เครื่องแสดงเวลาและปฏิทินแบบระยะไกล

TIME AND CALENDAR REMOTE MONITOR

โดย

นาย ปฐมพงษ์ พรอยู่ศรี

นาย วาทยุทธ บัวแย้ม

นาย วีระศักดิ์ ธรรมเสริมสกุล

นาย สมบัติ อินยิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2539

ปริญญานิพนธ์ ปีการศึกษา 2539

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องแสดงเวลาและปฏิทินแบบระยะไกล

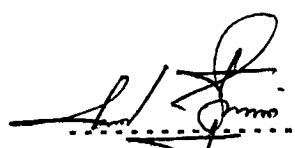
Time and Calendar Remote Monitor

ผู้จัดทำ นาย ปฐมพงษ์ พรอยู่ศรี เลขประจำตัว 37012100

นาย วาทยุทธ บัวแย้ม เลขประจำตัว 37012109

นาย วีระศักดิ์ ธรรมเสริมสกุล เลขประจำตัว 37012112

นาย สมบัติ อินยิน เลขประจำตัว 37012116



.....อาจารย์ที่ปรึกษา

(ผศ. สุพรรณ กุลพาณิชย์)

เครื่องแสดงเวลาและปฏิทินแบบระยะไกล

นาย ปฐมพงษ์ พรอยู่ศรี 37012100
นาย วาทยุทธ บัวแย้ม 37012109
นาย วีระศักดิ์ ธรรมเสริมสกุล 37012112
นาย สมบัติ อินยิน 37012116

อาจารย์ที่ปรึกษา
ผศ. สุพรรณ กุลพาณิชย์

บทคัดย่อ

โครงงาน เครื่องแสดงเวลาและปฏิทินแบบระยะไกล เป็นการประยุกต์การใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51 ในเรื่องติดต่อสื่อสารทางพอร์ทอนุกรม โดยการใช้มาตราฐานการส่งสัญญาณแบบ RS 232 และ RS 422 รวมทั้งการต่อขยายพอร์อินพุท-เอาท์พุท เพื่อเพิ่มการติดต่อกับส่วนการแสดงผลทั้ง แอลอีดี 7 Segment และ แอลอีดี Dot matrix , เพื่อการรับอินพุทคีย์บอร์ดสำหรับการตั้งค่า เวลาและปฏิทิน นอกจากนี้ยังประยุกต์การติดต่อและทำงานร่วมกันของไมโครคอนโทรลเลอร์ตระกูล MCS-51 ด้วยกันเองและกับชิพต่างๆ คือ RTC (Real Time Clock) เพื่อใช้เป็นสัญญาณนาฬิกาพื้นฐานสำหรับการส่งและแสดงผลเวลาและปฏิทิน โดยจะมีแบตเตอร์รี่แบคอัพข้อมูลของ Register ในRTC เพื่อป้องกันการผิดพลาดของเวลา และปฏิทิน , Digital - Temperature Sersor สำหรับการตรวจวัดอุณหภูมิ ที่บอร์ดสเลฟ , Analog To Digital สำหรับการรองรับการตรวจวัดค่าความชื้น รวมทั้งการประยุกต์การติดต่อกับไมโครคอมพิวเตอร์เพื่อใช้ในการ Down Load ข้อมูลเวลา,ปฏิทิน และข้อความข่าวสาร และ Up Load ผลของการตรวจวัดค่าของอุณหภูมิและสำรองสำหรับค่าความชื้นตามเวลาต่างๆ

TIME AND CALENDAR REMOTE MONITOR

By :

Mr.Pathompong Pornyoosri

Mr.Watayout Baouyam

Mr.Werasak Thammsermsakul

Mr.Sombutt Inyin

Advisor ;

Mr. Suphan Kumrapanit

Abstract

The Thesis TIME AND CALENDAR REMOTE MONITOR it application of microcontroller ,MCS - 51. In serial communication by standard serial communication RS 232 and RS 422. There port input-output expansion for communicate Display with LED 7-Segment and LED Dot matrix , and interface to receive input keyboard for set time and calendar .Other application interface and complementary with there flemry itself and other chip.The RTC chip (Real Time Clock) this clock pulse is time base for send and display time and calendar.Have battery backup data register of RTC.The requite protect fault of time and calendar.The Digital Temperature Sersor for detect temperature on slave Board . with Analog To Digital will support detect humidity . Finally Application interface with microcomputer for development solfware. It used Down Load data of time , calendar and information massage and Up Load data from detect temperature and data support humidity for some time.

กิตติกรรมประกาศ

ขอขอบพระคุณ ผศ. สุพรรณ กุลพาณิชย์ เป็นอย่างสูงที่กรุณาให้คำปรึกษา และ แนะนำการส่งสัญญาณแบบ อนุกรมทั้ง RS 232 และ RS 422 ตลอดทั้งการแก้ปัญหาในการทำงานทั้งหมด พร้อมให้ความอนุเคราะห์ ชุดแสดงผล LED DOT MATRIX เพื่อใช้ในโครงงานนี้ และสนับสนุนในด้านเครื่องมือที่ใช้สำหรับการทำโครงงานชิ้นนี้ทั้งหมด

ขอขอบพระคุณอาจารย์ประจำภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรมทุกท่านที่ให้คำปรึกษาในเรื่องที่เกี่ยวของของโครงงานชิ้นนี้

ขอขอบพระคุณ คุณพ่อ, คุณแม่ ที่ให้กำลังใจในการศึกษาและสนับสนุนการทำโครงงาน

ขอขอบคุณ เพื่อนๆ พี่ๆ น้องๆ ที่คอยให้กำลังใจตลอดมา

สารบัญ

สารบัญภาพ

บทที่ 1

ขอบเขตของโครงงาน

บทนำ

โครงงานเครื่องส่งเวลาและปฏิทินแบบแสดงผลระยะไกล (Time & Calendar Remote Monitor) เป็นโครงงานที่ใช้ทดสอบการทำงานของไมโครคอนโทรลเลอร์ตระกูล MCS - 51 ในการส่งสัญญาณแบบอนุกรมตามมาตราฐาน แบบ RS - 232 และมาตราฐานแบบ RS - 422 โดยใช้เวลา (Time) และปฏิทิน(Calendar)เป็นข้อมูลหลักในการส่งข้อมูล รวมทั้งข้อมูลที่เป็นข้อความ (Massage) สำหรับการแจ้งข่าวสาร

นอกจากนี้แล้วโครงงานเครื่องส่งเวลาและปฏิทินแบบแสดงผลระยะไกล ยังใช้สำหรับการทดสอบการติดต่อร่วม (interface) กับชิพที่เป็นส่วนประกอบของโครงงาน เช่น ชิพ DS 1202 ซึ่งเป็นตัวกำเนิดเวลาจริง (Real - Time - Clock) ที่เรียกย่อกันว่า RTC ในโครงงานนี้จะใช้เป็นตัวกำเนิดฐานเวลาให้การส่งเวลาและปฏิทิน ชิพ DS 1602 เป็นตัวตรวจจับอุณหภูมิแบบ ดิจิตอล (Digital Temperature Sensor) เป็นต้น รวมทั้งการใช้ซอฟร์แวร์ (Solf Ware) บนเครื่องคอมพิวเตอร์ในโครงงานนี้จะเป็นโปรแกรมที่พัฒนาบนวิสชวลเบสิค(Vistual Basic) สำหรับการอัพโหลด (Up-load) และการดาวร์โหลด (Down-load) ข้อมูลต่างที่จะกล่าวต่อไปในหัวข้อขอบเขตของโครงงานให้กับ ไมโครคอนโทรลเลอร์ตระกูล MCS - 51 ที่รองรับด้วย ซอฟร์แวร์ (Solf Ware) เอสเซมบลี (Assembly) สำหรับควบคุมการรับส่งสัญญาณและการควบคุมการแสดงผล รวมทั้งการตั้งค่าเวลาและปฏิทินจากคีบอดร์ และการดาวร์โหลดข้อมูลมาจากเครื่องคอมพิวเตอร์

ขอบเขตของโครงงาน

เครื่อง Time & Calendar Remote Monitor เป็นการทำงานร่วมกันของ อุปกรณ์หลักทั้งหมด 3 ชุดซึ่งประกอบด้วย

1 Microcomputer ทำหน้าที่

- Down - Load Data ซึ่ง Data ประกอบด้วย Time (hour:min:sec) ,Calendar (Day:date:month:year)และข้อความ(Massage) ผ่าน RS - 232 ไปให้ Master Control Board

- Up - Load Data จาก Master Control Board มายัง Microcomputer เพื่อ พิมพ์ผลการเก็บข้อมูลในการทำงานของระบบที่ทำการตรวจวัดค่าอุณหภูมิ (Temperature) และค่าที่ใช้รองรับค่าความชื้น (Humidity) จาก SlaveControl Board ผ่านทาง RS - 422 มาเก็บที่บัฟเฟอร์ใน Master Control Board ที่มีค่าวัน,เวลา ที่บันทึกค่าเหล่านั้นรวมอยู่ด้วย

2 Master Control Board ทำหน้าที่

- รับข้อมูล(Data)ประกอบด้วยเวลา (sec:min:hour) ,ปฏิทิน (Day:date:Month:Year) เพื่อทำการตั้งเวลาและปฏิทินให้กับระบบทั้งหมดโดยฐานเวลาจะใช้ชิพ RTC (Real-Time-Clock) เป็นฐานเวลา ซึ่งมี แบตเตอร์รี่ (Battery)ที่ใช้สำรองข้อมูลในขณะไฟฟ้าดับ

-ส่งเวลา (sec:min:hour) , ปฏิทิน (Day:date:Month:Year) และ ข้อความ (Massage) ผ่าน RS 422 ไปให้ Slave Control Board เพื่อการแสดงผลต่อไป

- รับ Dataประกอบด้วยTemperature,Humidity จาก Slave Control Board เพื่อบันทึกลงตารางสำหรับการการพิมพ์ โดย เครื่องไมโครคอมพิวเตอร์ (Microcomputer)

3 slave Control Board ทำหน้าที่

- รับ Dataประกอบด้วย Time(sec:min:hour),Calendar(Day:date:Month:Year) และข้อความ (Massage) จาก Master Control Board เพื่อนำมาแสดงผลบน LED แบบ DOT - METRIX ขนาด 8*8 จำนวน 8 หลัก ผ่าน RS 422

- ส่ง Data ประกอบด้วย ค่าอุณหภูมิ(Temperature) , ค่าความชื้น (Mimidity) ไปยัง Master Control Board ผ่าน RS 422

- Display data ทั้งหมด คือ Time(sec:min:hour) , Calendar(Day:date:Month:Year) ข้อความ (Massage),Temperature,Humidity

ลักษณะการทำงานของเครื่องแสดง Time & Calendar Remote Monitor เป็นการทำงานที่อาศัยระบบการส่งสัญญาณระยะไกลผ่าน ที่เป็นมาตราฐาน RS 422 ซึ่งมีข้อดีก็คือการส่งสัญญาณได้ไกลถึง 4000เมตรและการรับ LOAD ได้ 10 ตัว ต่อตัวส่ง 1 ตัว ในขณะเดียวกัน ความเร็วในการส่งข้อมูลสูงสุด (Maximum Data Rate) มีค่าได้ถึง 10 bps ต่างจากระบบการส่งสัญญารมาตรฐาน ที่เป็น RS 232 ที่การส่งสัญญาณได้ไกลเพียง 50 เมตรและ การรับ LOAD ได้ 1 ตัว ต่อตัวส่ง 1 ตัวและสามารถเปรียบเทียบคุณสมบัติที่สำคัญ และข้อดีต่างๆ จากตารางที่ 1

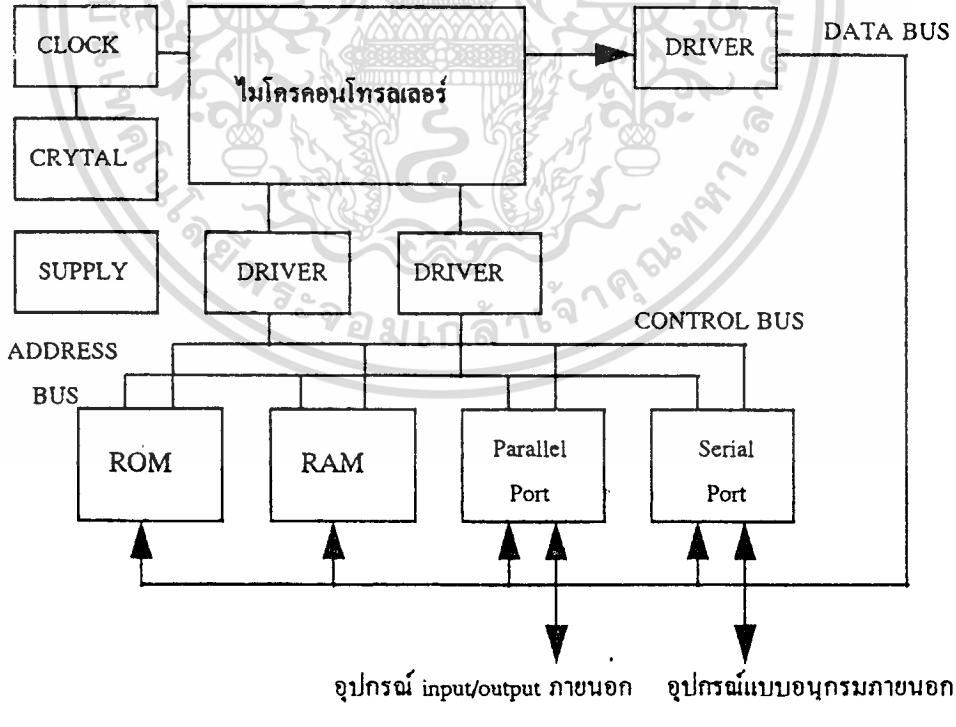
การควบคุมการทำงานของทั้ง Master Control Board และ slave Control Board จะใช้ Micro Controller ตระกูล MCS-51 ซึ่งจะแสดงรายละเอียดเกี่ยวกับโครงสร้าง และRgister ต่างๆ ที่เกี่ยวข้องกับระบบการคุมควบของ Time & Calendar Remote Monitor รวมถึงชิพ RTC , ชิพ Temperature sensor ดังรายละเอียดที่จะแสดงต่อไป

PARAMETER	RS - 232-C	RS - 423-A	RS - 422 -A	RS - 485
Mode of operation	Single - Ended	Single - Ended	Differential	Differential
Number of driver Number of receiver	1 Driver 1 Driver	1 Driver 10 Driver	1 Driver 10 Driver	1 Driver 1 Driver
Maximumum data bit per second(bps)	20K	100K	10M	10M
Max Common Mode Voltage	_+25V	_+6V	6V-(-0.25V)	12V-(-7V)
Driver Output	5V mim 25V max	3.6V mim 6.0V max	2V min	1.5Vmin
Driver load	3K -7K ohm	450 ohm min	100 ohm min	60 ohm min
Driver slew	30us Max	externally controller	NA	NA
Driver output shot circuit current limit	500mAto Vcc or GND	150 mAtoGND	150 mA to GND	150 mA to GND 250mA to8,12V
Driver output resistance (hight Z impedance)	NA 300 ohm	NA 60 Kohm	NA 60 Kohm	120 Kohm 120 Kohm
Receiver imput resistance	3 - 7 Kohm	4 Kohm	4 Kohm	12 Kohm
Maximum Cable length (ft)	50	4000	4000	4000
Receiver sensitivity	3	200mV	200mV	200mV

รูปที่ 1 ตารางที่เปรียบคุณสมบัติของระบบการส่งสัญญาณ มาตราฐาน RS แบบต่าง ๆ

บทที่ 2
แนวความคิดพื้นฐานของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์อาจจะเป็นคำที่ไม่ค่อยคุ้นเคยกันมากนัก เมื่อเปรียบเทียบกับคำว่าไมโครโปรเซสเซอร์ซึ่งมีการใช้งานแพร่หลายมากกว่า โดยมักจะเรียกว่าเป็นตัวประมวลผลกลางหรือ ซีพียู (CPU) ของระบบ เมื่อมีการนำไปใช้งานจำเป็นจะต้องมีไอซีประกอบภายนอกเพิ่มเติมเข้าไปให้ระบบที่สมบูรณ์ เช่น หน่วยความจำ และพอร์ตควบคุม เป็นต้น ดังแผนภาพในรูปที่ 2 โดยองค์ประกอบของไมโคร โปรเซสเซอร์จะเป็นบล็อกทั้งหมดที่อยู่ภายในเส้นประ บล็อกของการทำงานอื่น ๆ เป็นไอซีภายนอกที่จะต้องเพิ่มเติมเข้าไปให้ครบตามหน้าที่ที่ต้องการ สำหรับไมคอน โทรลเลอร์ก็มีหลักการพื้นฐานเช่นเดียวกัน เพียงแต่องค์ประกอบการทำงานเหล่านั้นหลายส่วน ได้รับการออกแบบให้บรรจุอยู่ภายในไอซีเพียงตัวเดียวเท่านั้น ดังนั้น การนำไปใช้งานก็ค่อนข้างจะสะดวก เพียงการต่อคริสตอลเพื่อเป็นฐานเวลาและแหล่งจ่ายไฟให้เท่านั้น ก็จะเป็นระบบคอมพิวเตอร์เพื่องานควบคุมที่สมบูรณ์ในบางครั้งจึงอาจมีการเรียกไมโครคอนโทรลเลอร์ว่าเป็น 1 chip microcomputer ก็ได้

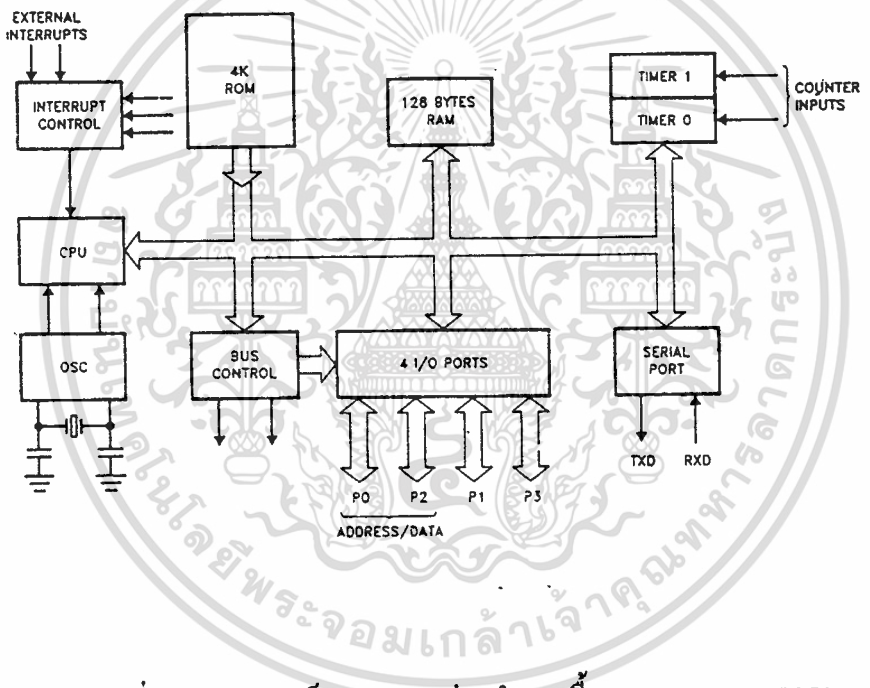


รูปที่ 2 หน่วยความจำการทำงานทั่วไปของระบบไมโครโปรเซสเซอร์

คุณลักษณะพื้นฐานของ 8051

จากแผนภาพในรูปที่ 3 แสดงให้เห็นถึงหน่วยการทำงานพื้นฐานของไมโครคอนโทรลเลอร์เบอร์ต่าง ๆ ที่จัดอยู่ภายในตระกูล MCS-51 นี้ ประกอบด้วย

- หน่วยประมวลผลกลางขนาด 8 บิต
- หน่วยประมวลผลสำหรับข้อมูลแบบบิต (Boolean Processor)
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำโปรแกรม 64 กิโลไบต์
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำข้อมูล 64 กิโลไบต์



รูปที่ 3 แผนภาพบล็อกแสดงหน่วยทำงานพื้นฐานของ MCS-8051

- หน่วยความจำโปรแกรมภายในขนาด 4 กิโลไบต์ แบบ EPROM (เบอร์ 8751) หรือแบบ ROM (เบอร์ 8051)
- หน่วยความจำแบบ RAM ภายในจำนวน 128 ไบต์
- พอร์ตอินพุต / เอาต์พุตแบบขนานจำนวน 32 เส้น ซึ่งสามารถแยกทำงานได้อย่างอิสระ
- วงจรนับ/จับเวลาขนาด 16 บิต จำนวนสองวงจร
- วงจรสื่อสารแบบอนุกรมแบบฟูลดูเพล็กซ์ (Full Dulplex)

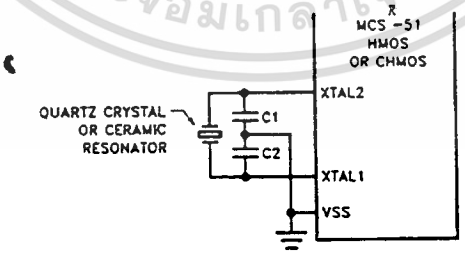
- วงจรควบคุมการอินเตอร์รัปต์จากแหล่งกำเนิดสัญญาณ 6 ประเภท พร้อมการกำหนดลำดับความสำคัญได้สองระดับ

- วงจรออสซิลเลเตอร์ภายใน

ฐานเวลาในการทำงานของซีพียูภายใน 8051

8051 มีวงจรออสซิลเลเตอร์อยู่ภายใน สำหรับการสร้างพัลส์ของสัญญาณนาฬิกา ซึ่งจะนำไปเป็นฐานเวลา หรือการกำหนดจังหวะการทำงานของหน่วยการทำงานทั้งหมดให้สอดคล้องกัน (Sinchronization) โดยปกติแล้วก็มักจะทำโดยการใช้คริสตอลเชื่อมต่อเข้ากับขาสัญญาณ XTAL1 และ XTAL2 พร้อมกับตัวเก็บประจุดังลักษณะในรูปที่ 4 หรืออาจจะเป็นสัญญาณนาฬิกาจากภายนอกก็ได้

พัลส์ความถี่ของสัญญาณนาฬิกาจะเรียกว่า Pulse (ใช้สัญลักษณ์เป็นตัวอักษร P) และคาบของสัญญาณนาฬิกานี้ เรียกว่า คาบเวลาออสซิลเลเตอร์ (Oscillator period) คาบเวลาออสซิลเลเตอร์จำนวนสองคาบ เรียกว่า State (ใช้สัญลักษณ์เป็นตัวอักษร S) ซึ่งจะนำไปใช้เป็นช่วงเวลาพื้นฐานการทำงานย่อยของไมโครคอนโทรลเลอร์ เช่น การนำคำสั่ง (Fetch) การถอดความหมาย (Decode) การประมวลผล (Execute) และการเขียนข้อมูล(Write) เป็นต้น ดังแสดงเป็นแผนภาพในรูปที่ 5 ช่วงเวลาของ State จำนวนหกครั้ง จะเรียกว่า แมชชีนไซเคิล (Machine cylce) ดังนั้นค่าหนึ่งแมชชีนไซเคิลจะใช้เวลา 12 คาบเวลาออสซิลเตอร์ ค่าของแมชชีนไซเคิลนี้จัดว่าเป็นช่วงเวลาที่น้อยที่สุดในการทำคำสั่งใดคำสั่งหนึ่ง ซึ่งหากว่าเป็นคำสั่งที่ซับซ้อนมากก็จะต้องใช้เวลานานสองถึงสามแมชชีนไซเคิล



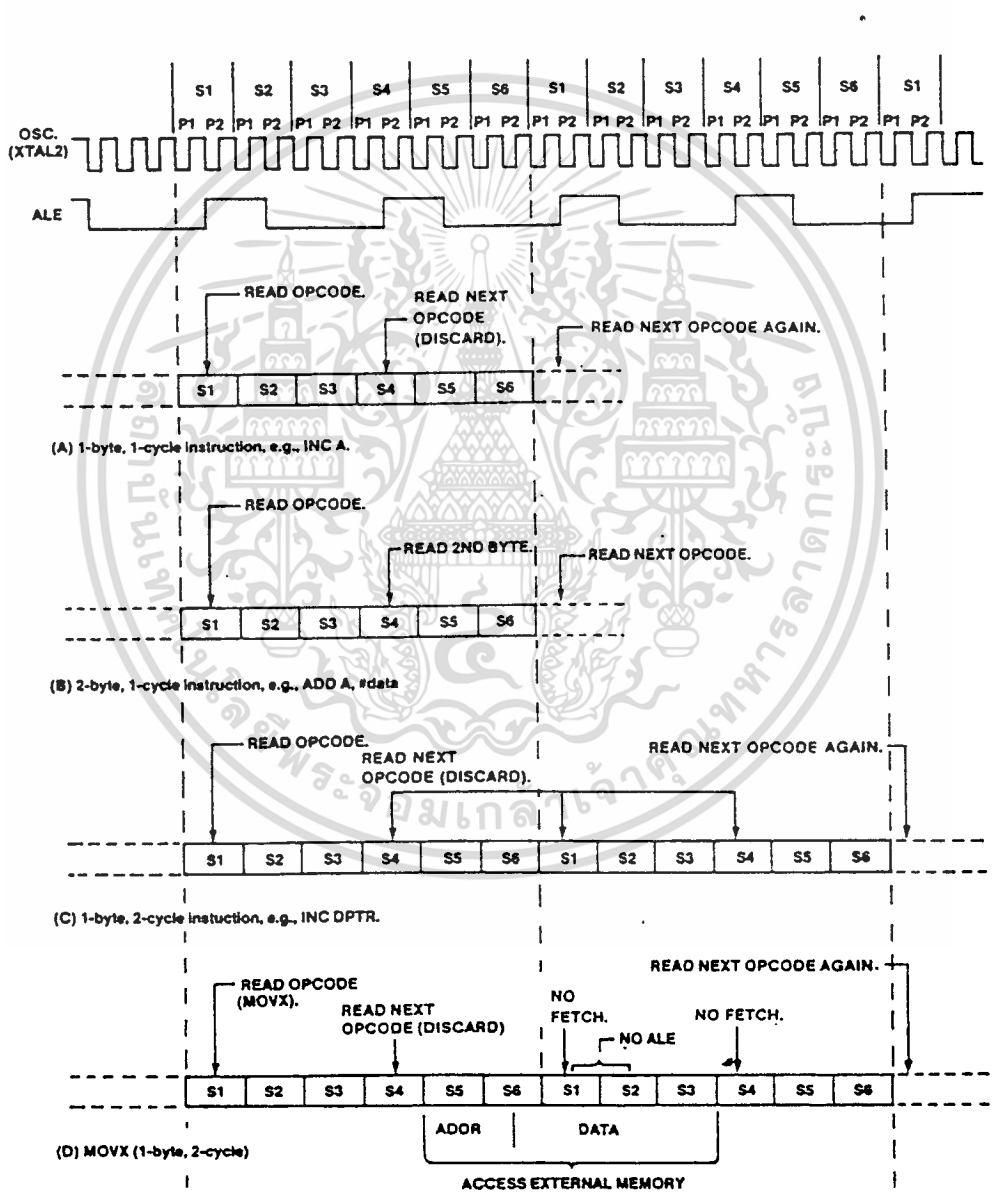
รูปที่ 4 แสดงการใช้คริสตอลภายนอกต่อเข้ากับวงจรออสซิลเลเตอร์

การคำนวณหาว่าเวลาที่ใช้ในการทำคำสั่งใดจนเสร็จสิ้น จะต้องดูว่าคำสั่งนั้นใช้จำนวนแมชชีนไซเคิลเป็นเท่าไรในการประมวลผล (ขอให้ดูข้อมูลจากภาคผนวก ข) เวลาที่ใช้จะคำนวณตามสูตร

$$T = \frac{C \times 12}{\text{Crystal Frequency}}$$

โดย C เป็นค่าจำนวนแมชชีนไซเคิลของคำสั่ง

Crystal Frequency เป็นค่าความถี่ของคริสตอลที่ใช้กับ 8051



รูปที่ 5 แสดงแผนภาพเวลาพื้นฐานของ 8051 และลำดับของช่วงเวลา stste ในการทำคำสั่งหนึ่งไบต์

<u>ตัวอย่างเช่น</u>

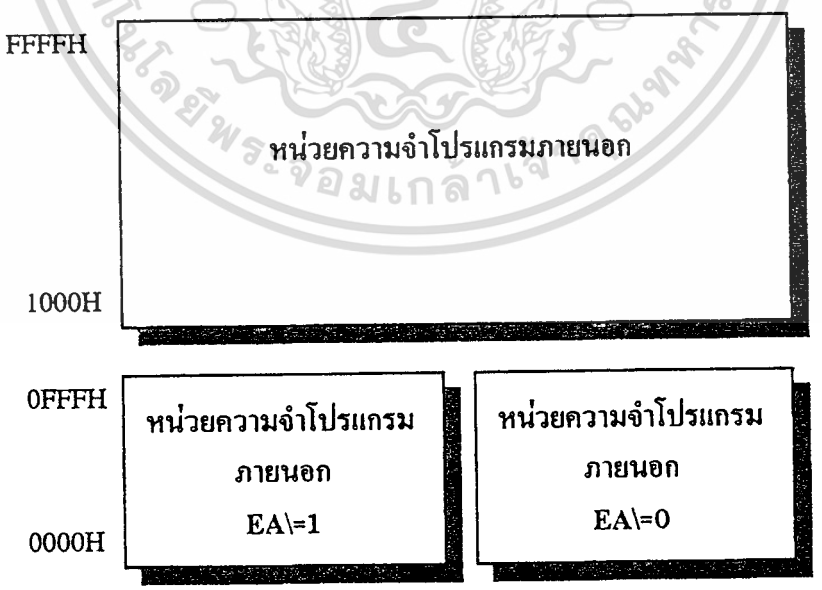
เวลาในการทำคำสั่ง ADD A,R1 ซึ่งต้องการ 1 แมชชีนไซเคิล
เมื่อใช้คริสตอล 16 เมกะเฮิร์ต จะเป็นเวลา 0.75 ไมโครวินาที และ
เมื่อใช้คริสตอล 12 เมกะเฮิร์ต จะเป็นเวลา 1 ไมโครวินาที เป็นต้น

อย่างไรก็ตาม ในบางครั้งอาจจะพบเห็นการใช้ค่าของคริสตอลเป็น 11.059 เมกะเฮิร์ต ทั้งนี้โดยมีเหตุผลเนื่องจาก สามารถนำค่าความถี่ที่ได้นี้ ไปใช้ในการเป็นฐานเวลาสำหรับการสร้างความถี่ในการรับ / ส่งข้อมูลอนุกรมซึ่งเป็นหน่วยการทำงานหนึ่งภายใน 8051 เอง โดยจะทำให้ได้ค่าที่ใกล้เคียงกับค่ามาตรฐานคือ 19200,9600,4800,2455,1200,และ 300 บิต/วินาที ซึ่งจะได้กล่าวถึงอีกครั้งหนึ่ง ต่อไป

หน่วยความจำโปรแกรมของ 8051

หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมของ 8051 เป็นบริเวณหน่วยความจำ สำหรับเก็บข้อมูลและคำสั่งใช้งานต่าง ๆ ซึ่งแม้ว่าจะไม่มีการจ่ายกระแสไฟฟ้าให้กับระบบ ข้อมูลเหล่านั้นก็ยังคงอยู่ไม่สูญหายโครงสร้างของหน่วยความจำโปรแกรม มีลักษณะเช่นเดียวกับหน่วยความจำที่บรรจุอยู่ในไอซี หน่วยความจำประเภทต่าง ๆ เช่น หน่วยความจำแบบ ROM (Read Only Memory) หรือ EPROM (Erasable ProgrammableRead Only Memory) เป็นต้นดังรูปที่ 6



รูปที่ 6 การจัดพื้นที่หน่วยความจำสำหรับ ไมโครคอนโทรลเลอร์ 8051

8051 สามารถอ่านข้อมูลหน่วยความจำโปรแกรมนี้ได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทของหน่วยความจำโปรแกรมเป็น 2 ลักษณะ ตามตำแหน่งของหน่วยความจำนั้น คือ หน่วยความจำโปรแกรมภายใน (Internal Program Memory) ซึ่งเป็นหน่วยความจำROM หรือ EPROM ที่อยู่ภายในตัวไอซีไมโครคอนโทรลเลอร์เอง และหน่วยความจำโปรแกรมภายนอก (External Program Memory) ซึ่งเป็นการใช้ไอซีหน่วยความจำมาทำหน้าที่เป็นหน่วยความจำโปรแกรมของระบบ

หน่วยความจำโปรแกรมภายใน

ไมโครคอนโทรลเลอร์เบอร์ต่าง ๆ ที่จัดอยู่ภายในตระกูล 8051 นี้ มีขนาดของหน่วยความจำโปรแกรมภายในแตกต่างกันออกไปเพื่อความเหมาะสมกับการนำไปใช้งานในลักษณะต่าง ๆ กันดังนี้

8051 และ 8052 มีหน่วยความจำแบบ ROM ขนาด 4 และ 8 กิโลไบต์ ตามลำดับ ประกอบอยู่ภายในไอซี และมีความเหมาะสมกับการนำไปใช้ในวงจรทางอุตสาหกรรมที่มีจำนวนการผลิตมาก เนื่องจากจะมีผลทำให้ต้นทุนค่าใช้จ่ายการผลิตต่อหน่วยลดลงได้มาก

8751 มีหน่วยความจำแบบ EPROM ขนาด 4 กิโลไบต์อยู่ภายในไอซี ข้อมูลที่จัดเก็บอยู่ภายในนี้สามารถใช้แสงอัลตร้าไวโอเลตลบ และนำกลับไปบรรจุโปรแกรมใหม่ได้อีกครั้งหนึ่งคล้ายคลึงกับไอซีหน่วยความจำEPROM ที่จะกล่าวถึงในหัวข้อถัดไปไมโคอนโทรลเลอร์เบอร์นี้เหมาะสมกับงานค้าน อุตสาหกรรมที่มีจำนวนการผลิตคราวละไม่มากนัก หรืออาจจะเป็นงานประเภทต้นแบบภายในห้องปฏิบัติการ เป็นต้น

8031 และ 8032 ไม่มีหน่วยความจำโปรแกรมอยู่ภายในตัวไอซีเลย ดังนั้นในการนำไปใช้งานจึงจำเป็นต้องอาศัยหน่วยความจำโปรแกรมภายนอกเสมอ ซึ่งการใช้งานลักษณะนี้จะมีผลทำให้ต้องเสียความสามารถบางประการ เกี่ยวกับพอร์ตอินพุต/เอาต์พุตของไมโครคอนโทรลเลอร์ ไปเนื่องจากต้องนำไปใช้เป็นสัญญาณควบคุม เกี่ยวกับการจัดการติดต่อหน่วยความจำภายนอกแทน

หน่วยความจำโปรแกรมภายนอก

หน่วยความจำโปรแกรมภายนอกเป็นการใช้หน่วยความจำ EPROM (หรือ ROM) เชื่อมต่อเข้ากับระบบของ 8051 โดยอาจจะมีสาเหตุได้หลายประการ เช่น เป็นการทดลองทำระบบต้นแบบจำนวนน้อย หรืออาจต้องการลดต้นทุนการผลิตเพราะราคาของไมโครคอนโทรลเลอร์แบบที่ไม่มีหน่วยความจำโปรแกรมภายใน ราคาจะต่ำกว่าแบบที่มีหน่วยความจำโปรแกรมภายในมาก เป็นต้น ในบางครั้งอาจจะมีสาเหตุจากความจำเป็นอื่น ๆ ที่ไม่อาจหลีกเลี่ยงได้ เช่น การที่หน่วยความจำภายในของไมโครคอนโทรลเลอร์มีขนาดความจุที่ไม่เพียงพอกับการเก็บโปรแกรม หรือ อาจจะเป็นว่าการที่ใช้ไอซีหน่วยความจำจะทำให้สามารถจัด

หาเครื่องมือ (Tools) ช่วยการพัฒนาระบบที่ใช้งานกันโดยแพร่หลายและราคาถูกได้ ซึ่งจะช่วยลดเวลาในการพัฒนาระบบลงได้มาก เป็นต้น

ไมโครคอนโทรลเลอร์เบอร์ต่าง ๆ ของตระกูล 8051 นี้ สามารถขยายให้ใช้งานหน่วยความจำภายนอกได้ทั้งสิ้น โดยกรณีที่มีหน่วยความจำโปรแกรมภายในอยู่แล้ว การอ้างตำแหน่งแอดเดรสที่มี ทั้งในหน่วยความจำโปรแกรมภายในและภายนอกนั้น จะต้องทำการพิจารณาระดับลอจิกของสัญญาณ EA ในขณะนั้นด้วย ดังจะได้อธิบายรายละเอียดในหัวข้อต่อไป

การเชื่อมต่อหน่วยความจำโปรแกรมภายนอกกับ 8051

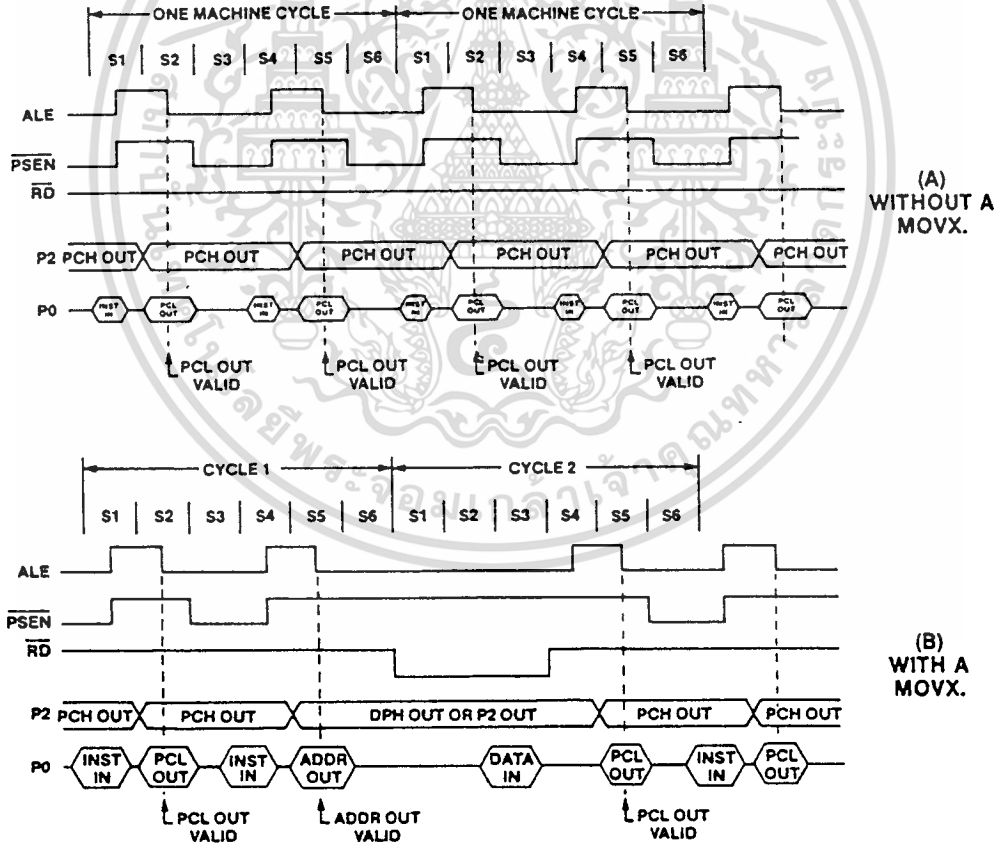
เนื่องจากระบบบัสแอดเดรสและบัสข้อมูลไมโครคอนโทรลเลอร์ 8051 เป็นลักษณะแบบใช้การมัลติเพล็กซ์จากพอร์ตเดียวกัน กล่าวคือในระยะเวลาเริ่มต้นเส้นสัญญาณเหล่านี้ของพอร์ตจะใช้ในการส่งค่าแอดเดรสของตำแหน่งที่ต้องการติดต่อด้วย ในช่วงเวลาต่อมาจึงจะเปลี่ยนไปเป็นสภาวะอิมพีแดนซ์สูงเพื่อใช้งานในฐานะของบัสข้อมูลแต่เนื่องจากEPROM ที่ใช้งานกันทั่วไปนั้นไม่ใช้การมัลติเพล็กซ์ และมีขาสัญญาณบัสแอดเดรสและบัสข้อมูลแยกออกจากกันโดยชัดเจนดังนั้นการเชื่อมต่อ EPROM เพื่อทำหน้าที่เป็นหน่วยความจำโปรแกรมจึงจำเป็นต้องมีวงจรประเภทแลตช์ (Latch) ประกอบเพิ่มเติมขึ้น เพื่อทำการค้างค่าของแอดเดรสที่ส่งออกมาจาก 8051 ในช่วงเวลาแรกให้กับขาสัญญาณแอดเดรสของ EPROM ต่อไป

จากตารางในรูปที่ 7 แสดงให้เห็นถึงสัญญาณต่าง ๆ ของ 8051 ซึ่งนำมาใช้ในการติดต่อกับหน่วยความจำภายนอก

สัญญาณ	คำจำกัดความ	ขาสัญญาณ	หน้าที่
EA	External Acess	31	เลือกประเภทหน่วยความจำภายในหรือภายนอก
ALE	Adress Enable	30	สัญญาณเอาต์พุตสำหรับการแลตช์ข้อมูลแอดเดรสจากบัส
P2.0-P2.3	Port 2	21-28	เป็นข้อมูลแอดเดรสไบต์สูงของหน่วยความจำ
P0.0-P0.7	Port 0	39-32	มัลติเพล็กซ์สัญญาณของบัสแอดเดรสและบัสข้อมูล
PSEN	Program Store Enable	29	สัญญาณระบุการ Read ให้กับหน่วยความจำ EPPROM

รูปที่ 7 สัญญาณของ 8051 ที่ใช้ในระหว่างการติดต่อ เพื่ออ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก

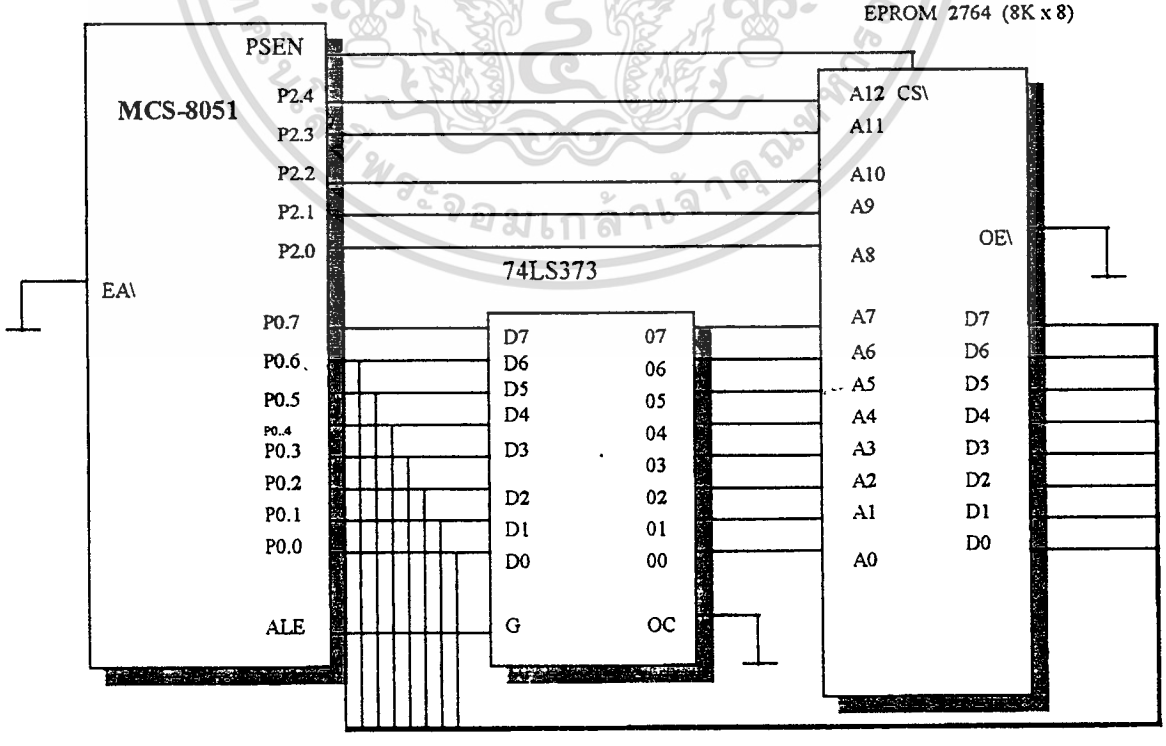
สัญญาณ EA (External Access) ใช้ในการกำหนดเลือกว่า จะอ่านข้อมูลมาจากหน่วยความจำโปรแกรมภายนอกหรือภายในตัวไอซีไมโครคอนโทรลเลอร์เอง ซึ่งหากเป็นระดับลอจิกต่ำจะอ่านข้อมูลมาจากหน่วยความจำโปรแกรมภายนอก และกรณีตรงข้ามจะอ่านข้อมูลมาจากหน่วยความจำโปรแกรมภายใน สิ่งที่ควรให้ความสังเกต คือ เมื่อมีการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายใน และมีการใช้งานแอดเดรสที่อยู่ในช่วงที่สูงเกินกว่าค่าสูงสุดของหน่วยความจำภายใน เช่น หากเป็นเบอร์ 8751 ซึ่งมีหน่วยความจำโปรแกรมภายในสูงสุดเพียง 4 กิโลไบต์ (ไม่เกินแอดเดรส OFFFH) แต่มีการอ้างถึงตำแหน่งที่สูงเกินกว่า 4 กิโลไบต์ เป็นต้น กรณีเช่นนี้ 8051 จะทำการอ่านค่าแอดเดรสที่สูงเกินกว่านี้มาจากหน่วยความจำโปรแกรมภายนอกโดยอัตโนมัติ



รูปที่ 8 แผนภาพสัญญาณเวลาแสดงการติดต่อกับหน่วยความจำโปรแกรมภายนอก

จากแผนภาพเวลาการติดต่อกับหน่วยความจำโปรแกรมภายนอกของ 8051 ในรูปที่ 8 จะเห็นว่าภายในช่วงเวลาของแมชชีนไซเคิลหนึ่ง ๆ นั้น 8051 มีการไปนำข้อมูลมาถึงสองครั้งด้วยกัน ดังนั้นการทำงานของ 8051 จึงจะไปอ่านข้อมูลโปรแกรมมาเป็นจำนวนทวีคูณเสมอ ในช่วงเวลาเริ่มต้นของการติดต่อกับหน่วยความจำภายนอก พอร์ต 0 จะเป็นค่าของแอดเดรสไบต์ต่ำ (A0-A7) และเวลาต่อมาจึงจะเป็นบัสข้อมูล การส่งค่าของแอดเดรสไบต์ต่ำนี้จะอยู่ในราวช่วงเวลาขอบขาลงของสัญญาณ ALE และจะยังคงอยู่จนเมื่อสัญญาณ PSEN เปลี่ยนไปเป็นระดับสัญญาณลอจิกต่ำ ดังนั้นการออกแบบวงจรจึงมักใช้สัญญาณ ALE นี้ในการทำให้ไอซีแลตช์ภายนอกค้างระดับสัญญาณแอดเดรสเหล่านี้ไว้ ส่วนสัญญาณ PSEN จะใช้ในการเลือกให้ไอซี EPROM ทำงานและอ่านข้อมูลกลับมา

ขณะเมื่อสัญญาณ PSEN เป็นระดับลอจิกต่ำ EPROM ก็จะทำการถอดระหัสค่าแอดเรสและส่งข้อมูลที่อยู่ในตำแหน่งนั้นออกมา โดยสัญญาณ PSEN นี้จะค้างสภาวะการเป็นลอจิกต่ำไว้ ช่วงเวลาหนึ่งเพื่อให้ข้อมูลถูกส่งออกมาจาก EPROM เรียบร้อยแล้ว จึงค่อยกลับไปเป็นระดับลอจิกสูงเช่นเดิม และในช่วงจังหวะขาขึ้นของสัญญาณ PSEN นี้เองที่ 8051 ทำการสุ่มอ่านข้อมูลเข้ามา สำหรับข้อมูลทางพอร์ต 2 ซึ่งเป็นค่าแอดเดรสไบต์สูง (A8-A15) นั้น จะถูกส่งออกมาในราวช่วงกึ่งกลางระหว่างที่สัญญาณ ALE เป็นลอจิกสูง ซึ่งก็เป็นเวลาใกล้เคียงกับการส่งค่าแอดเดรสไบต์ออกมาทางพอร์ต 0 สำหรับค่าแอดเดรสพอร์ต 2 นั้นจะค้างอยู่ตลอดเวลาของการติดต่อกับหน่วยความจำโปรแกรม



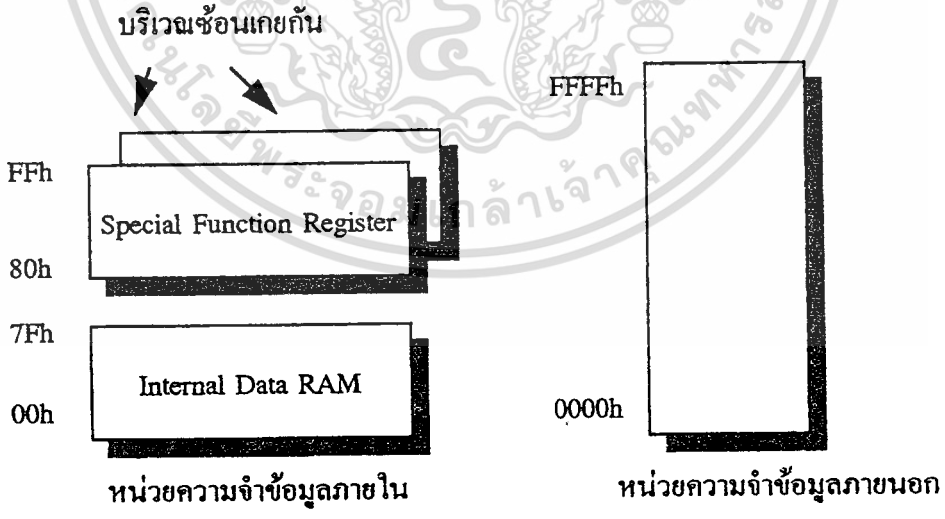
รูปที่ 9 การเชื่อมต่อหน่วยความจำภายนอกกับ 8051

จากวงจรในรูปที่ 9 แสดงให้เห็นถึงวิธีการเชื่อมต่อหน่วยความจำโปรแกรมภายนอกให้กับระบบ 8051 ด้วยไอซีหน่วยความจำ EPROM ขนาด 8K x 8 บิต เบอร์ 2764 ในรูปนี้ใช้สัญญาณ CS เชื่อมต่อโดยตรงเข้ากับขาสัญญาณ PSEN ของ 8051 ซึ่งจะมีผลทำให้ไม่ว่าจะทำคำสั่งใด ๆ ที่เกี่ยวข้องกับหน่วยความจำโปรแกรมแล้ว ก็จะเป็นการเลือกให้ EPROM นี้ทำงานตลอดเวลา

หน่วยความจำข้อมูลของ 8051

หน่วยความจำข้อมูล

หน่วยความจำข้อมูลมีหน้าที่สำหรับเก็บข้อมูล หรือตัวแปรที่เกิดขึ้นในขณะที่กำลังประมวลผลโปรแกรมไว้เป็นการชั่วคราว โดยพื้นฐานแล้วหน่วยความจำข้อมูลจัดเป็นหน่วยความจำ RAM แบบสแตติก ดังนั้นเมื่อไม่มีการจ่ายไฟฟ้าให้กับระบบ ก็จะมีผลทำให้ข้อมูลที่จัดเก็บไว้ภายในหน่วยความจำนี้สูญหายไป พื้นที่ของหน่วยความจำข้อมูลของ 8051 สามารถมีได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทออกเป็นสองลักษณะตามตำแหน่งที่ตั้งของหน่วยความจำนั้น ดังลักษณะของแผนภาพในรูปที่ 10 คือ หน่วยความจำโปรแกรมภายใน (Internal Data Memory) ซึ่งเป็น RAM ที่อยู่ภายในตัวของไอซีไมโครคอน โทรลเลอร์เอง และหน่วยความจำข้อมูลภายนอก (External Data Memory) ซึ่งเป็นการใช้ไอซีหน่วยความจำ RAM มาเพิ่มเติมเข้าไปในวงจรลักษณะเดียวกับการนำไอซี EPROM มาใช้งานเป็นหน่วยความจำโปรแกรมนั่นเอง



รูปที่ 10 การจัดพื้นที่หน่วยความจำข้อมูลสำหรับไมโครคอนโทรลเลอร์ 8051

หน่วยความจำขนาด 128 ไบต์แรก

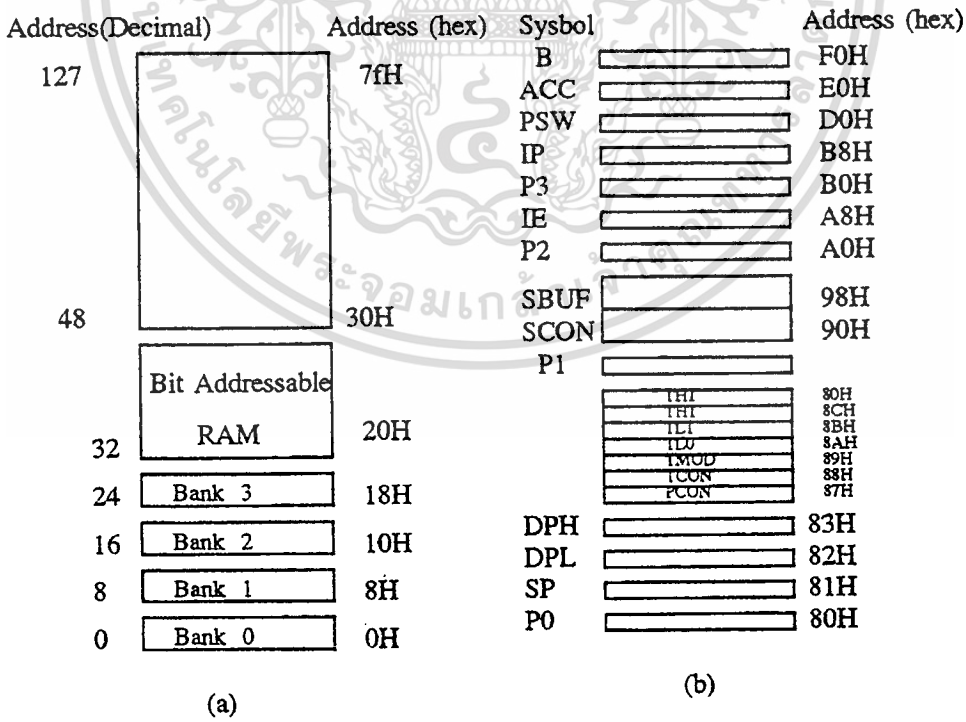
บริเวณนี้จะมีตำแหน่งแอดเดรสอยู่ในช่วง 00H-7FH ซึ่งยังได้มีการจำแนกย่อยออกไปอีกเป็นสามส่วนตามประเภทของการใช้งาน ดังนี้

บริเวณแอดเดรส 00H-1FH จำนวน 32 ไบต์ จำแนกออกเป็นกลุ่ม (หรือ แบงก์) ข้อมูลจำนวน 8 ไบต์ รวมทั้งหมดสี่กลุ่ม พื้นที่ข้อมูลในแต่ละกลุ่มจะถูกใช้งานในฐานะของรีจิสเตอร์ใช้งานทั่วไป ซึ่งมีชื่อเรียกว่า **รีจิสเตอร์ R0-R7** ดังตารางต่อไปนี้

แอดเดรส	รีจิสเตอร์แบงก์	ชื่อรีจิสเตอร์ใช้งาน
00H - 07H	0	R0 - R7
08H - 0FH	1	R0 - R7
10H - 17H	2	R0 - R7
18H - 1FH	3	R0 - R7

รูปที่ 11 ตารางการจัดหน่วยความจำตามชื่อของ รีจิสเตอร์

จะเห็นได้ว่าชื่อของรีจิสเตอร์ไม่ว่าจะอยู่ในรีจิสเตอร์แบงก์ใด ก็จะมีชื่อ R0 ถึง R7 เหมือนกันทั้งสิ้น (ดูรูปที่ 12) ดังนั้นในการใช้งานผู้ใช้จะต้องให้ความระมัดระวังว่าต้องการรีจิสเตอร์นั้น ๆ จาก



รูปที่ 12 การจัดพื้นที่หน่วยความจำข้อมูลภายใน (a) ช่วงตั้งแต่แอดเดรส 00-FFH และ (b)ช่วง 80-FFH ซึ่งกำหนดให้เป็นพื้นที่ของรีจิสเตอร์ใช้งานพิเศษ(Special-Function register)

แบงก์ใด การสวิตช์เลือกแต่ละกลุ่มของรีจิสเตอร์นี้ก็ทำได้ง่าย เพียงการกำหนดค่าของบิตที่อยู่ภายในรีจิสเตอร์ PSW เท่านั้นตามตาราง.รูปที่ 13 ต่อไปนี้

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

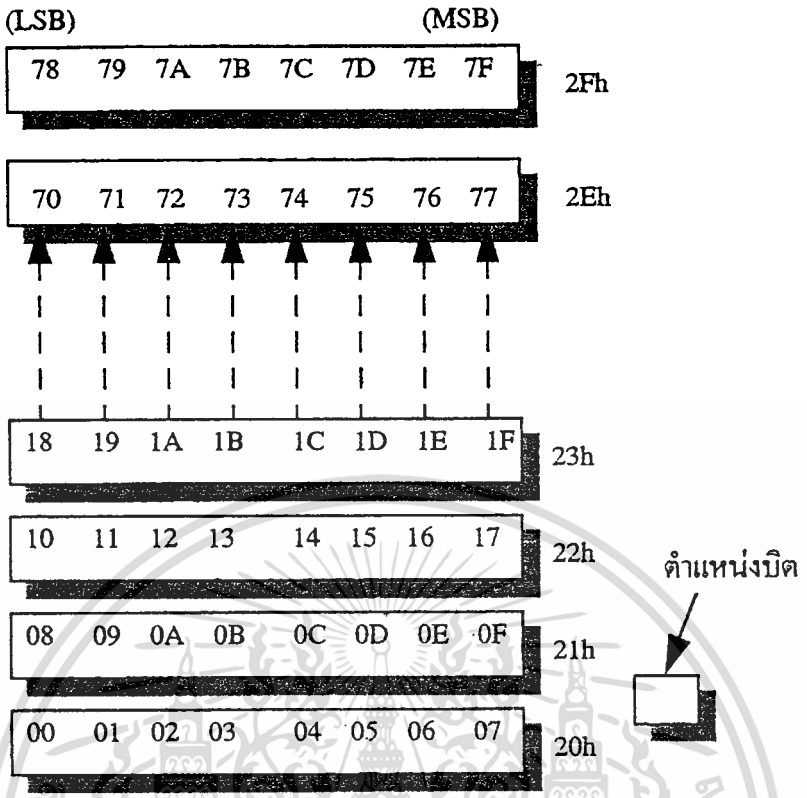
รูปที่ 13 การเลือกแบงก์ การใช้งานของรีจิสเตอร์ PSW

อย่างไรก็ตามโดยทั่วไปก็มักจะมีการใช้งานรีจิสเตอร์ R0-R7 เฉพาะในแบงก์ 0 เท่านั้น ดังนั้นพื้นที่ของแบงก์อื่น ๆ ที่เหลือก็สามารถนำมาใช้ในลักษณะของหน่วยความจำข้อมูลภายในปกติด้วยการอ้างถึงหมายเลขของแอดเดรสนั้น ๆ โดยตรง

บริเวณแอดเดรส 20H-2FH จำนวน 16 ไบต์ บริเวณพื้นที่เป็นส่วนสำหรับผู้ใช้ซึ่งจะมีความพิเศษต่างไปจากหน่วยความจำส่วนอื่น ๆ เนื่องจากผู้ใช้อาจสามารถอ้างถึงหน่วยความจำบริเวณนี้ได้ทั้งในลักษณะของ ไบต์ข้อมูลเช่นปกติหรืออาจจะเป็นบิตข้อมูลได้โดยตรง ดังนั้นหากว่ามองในลักษณะบิตข้อมูลแล้ว ก็จะมีพื้นที่ตัวแปรแบบบิตให้ใช้งานได้มากถึง 128 บิต โดยตำแหน่งแรกของบิตจะเป็นบิตซึ่งเริ่มต้นนับจากบิตนัยสำคัญต่ำสุด (LSB) ของแอดเดรส 2 เรื่อยไปจนกระทั่งถึงบิตที่ 127 ซึ่งเป็นบิตนัยสำคัญสูงสุด (MSB) ของแอดเดรส 2FH (ดูรูปที่ 14)

ความสามารถในการใช้งานพื้นที่ส่วนนี้แบบบิตข้อมูลโดยตรงนี้ นับว่าน่าสนใจมากและถือเป็นการใช้งาน 8051 อย่างเต็มประสิทธิภาพทีเดียว เนื่องจากว่า 8051 ได้รับการออกแบบมาสำหรับควบคุมเป็นพื้นฐานอยู่แล้ว ซึ่งส่วนมากงานลักษณะเช่นนี้หากเป็น การนำเข้าข้อมูลก็มักจะเป็นเพียงการอ่านค่าสถานะลอจิกของเส้นสัญญาณ หรือกรณีการส่งออกข้อมูลก็จะเป็นการกำหนดสภาวะลอจิกให้กับวงจรภายนอกผ่านทางบิตใดบิตหนึ่งอยู่แล้ว ดังนั้นหากว่ามีการกำหนดบิตหรืออ่านค่าของบิตโดยตรง แทนที่จะต้องทำลอจิกขั้นต้นกับข้อมูล ทั้งไบต์เพื่อต้องการทราบผลเพียงหนึ่งบิตเช่นที่กระทำกันในโปรเซสเซอร์โดยทั่วไป ก็จะเพิ่มความสะดวกและรวดเร็วในการเขียนโปรแกรมควบคุมมาก รายละเอียดในส่วนนี้จะได้กล่าวอีกครั้งหนึ่งเมื่อศึกษาถึงการใช้งานพอร์ตอินพุต/เอาต์พุตต่อไป

บริเวณแอดเดรส 30H - 7FH เป็นบริเวณที่สามารถนำไปใช้งานได้อย่างอิสระ โดยสามารถอ้างถึงได้เฉพาะในลักษณะของไบต์ข้อมูลตามปกติเท่านั้น



รูปที่ 14 หน่วยความจำข้อมูลภายในบริเวณที่อ้างถึงได้แบบบิต

รีจิสเตอร์หน้าที่พิเศษ

รีจิสเตอร์หน้าที่พิเศษ (SFR) เป็นรีจิสเตอร์สำหรับการควบคุมหน้าที่และการทำงานของอุปกรณ์หรือพอร์ตของ 8051 ทั้งหมด โดยที่ตำแหน่งอยู่ในบริเวณแอดเดรส 80H-FFH (ตามรูปที่ 15) การใช้งานรีจิสเตอร์หน้าที่พิเศษเหล่านี้สามารถทำได้ทั้งการระบุถึงชื่อของรีจิสเตอร์หรือตำแหน่งแอดเดรสที่เป็นของรีจิสเตอร์นั้นก็ได้

ตารางต่อไปนี้แสดงเห็นลักษณะการจัดพื้นที่หน่วยความจำ สำหรับรีจิสเตอร์หน้าที่พิเศษเหล่านี้โดยมีข้อสังเกตว่ารีจิสเตอร์ที่อยู่ในตำแหน่งแอดเดรสที่เป็นจำนวนทวีคูณของค่า 8 จะสามารถอ้างถึงในระดับบิตได้ด้วย (นั่นคือแอดเดรส 80H, 88H ,90H ,98H ,AOH , A8H , B0H ,B8H , D0H ,E0H และ F0H)

SFR MEMORY MAP

8 Byte

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

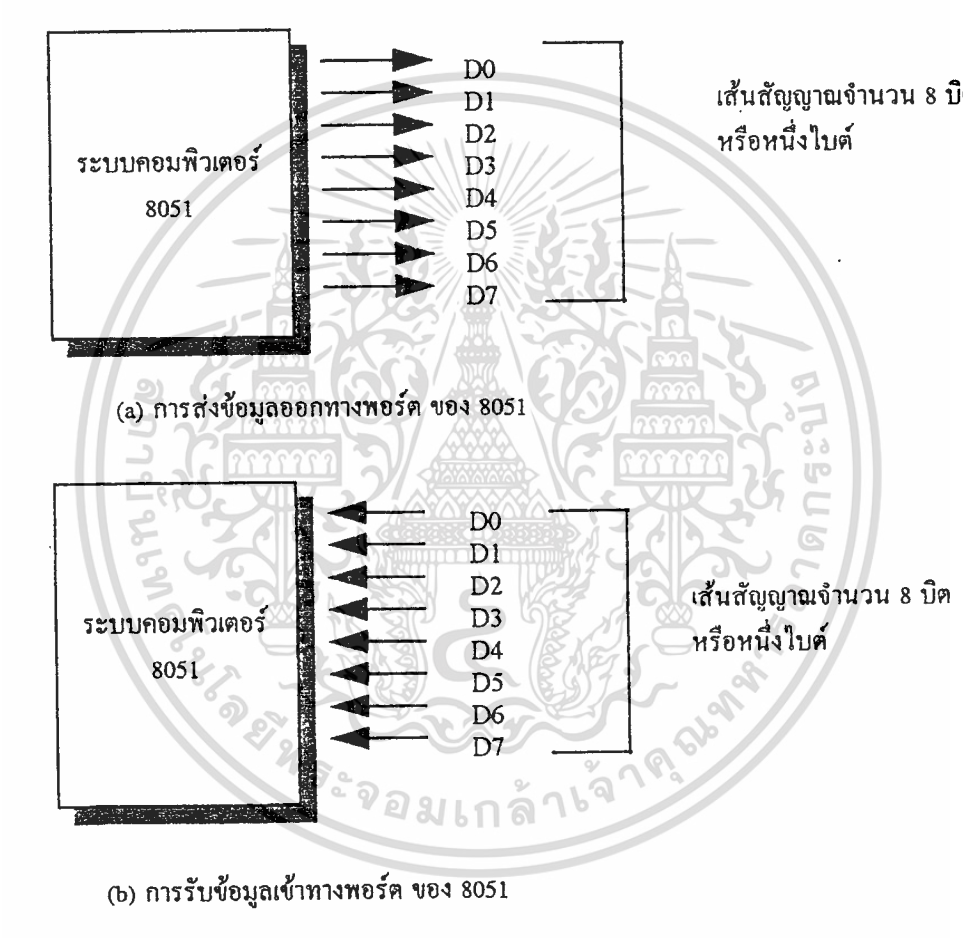
Bit Addressable

รูปที่ 15 รีจิสเตอร์ใช้งานพิเศษ (Special Function Register หรือ SFR)

พอร์ตอินพุต / เอาต์พุตของ 8051

พอร์ต มีความหมายถึง แอดเดรสหนึ่งที่ได้รับการกำหนดไว้เพื่อการโอนย้ายข้อมูลระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอก การกำหนดประเภทของการติดต่อขึ้นอยู่กับทิศทางการไหลของข้อมูลเมื่อพิจารณาจากไมโครคอนโทรลเลอร์เป็นหลัก (ดูรูปที่16) ดังนั้นการนำเข้าข้อมูลจากวงจรภายนอกจึงเรียกว่า การอินพุต (input) และในกรณีตรงกันข้ามเพื่อส่งออกข้อมูลก็จะเรียกว่า การเอาต์พุต (output)

เมื่อพิจารณาถึงวิธีการส่งข้อมูลภายในพอร์ตจะสามารถแยกประเภทของพอร์ตออกได้เป็นสองลักษณะ คือ **พอร์ตแบบขนาน (Parallel port)** ซึ่งทำการส่งจำนวนบิตข้อมูลทั้งหมดออกมาหรือนำเข้าไปพร้อมกันในคราวเดียว และ **พอร์ตแบบอนุกรม (Serial port)** ซึ่งทำการโอนย้ายข้อมูลคราวละบิต ๆ จนครบจำนวนแต่สำหรับในบทนี้จะได้กล่าวถึงเฉพาะในส่วนของพอร์ตแบบขนานเท่านั้น สำหรับการทำงานของพอร์ตแบบอนุกรมจะได้กล่าวภายหลังในหัวข้อต่อไป



รูปที่ 16 การอินพุต/เอาต์พุตขอ้มูลกับอุปกรณ์ภายนอกของ 8051

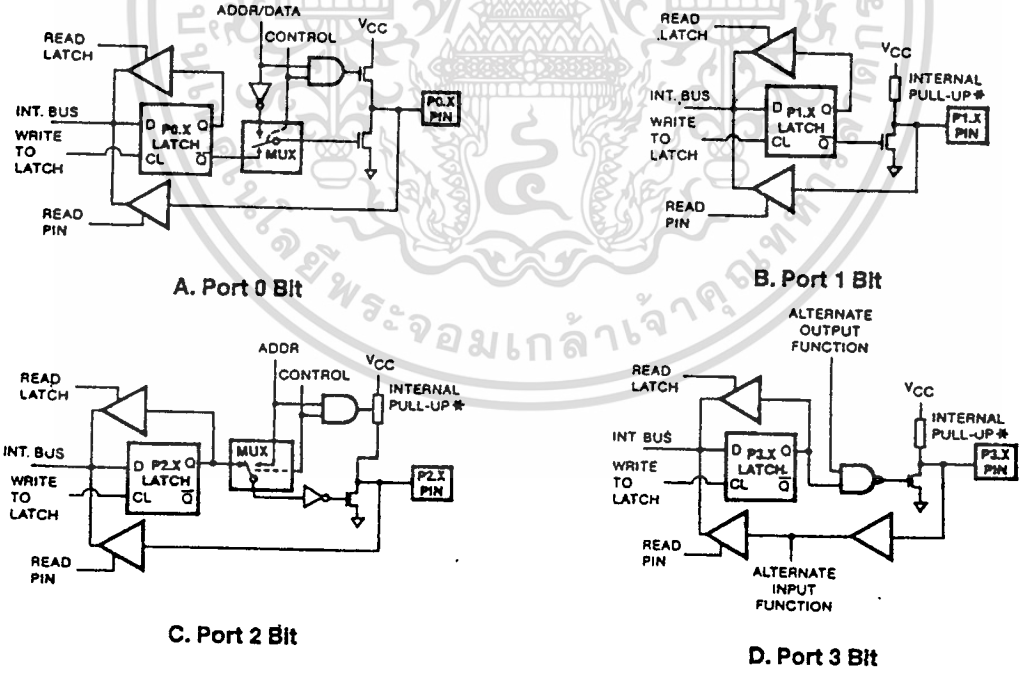
พอร์ตแบบขนานของ 8051

8051 มีโครงสร้างของพอร์ตที่สามารถใช้งานแบบขนานได้จำนวนทั้งหมดสี่พอร์ต เรียกชื่อเรียงตามลำดับว่า **พอร์ต 0, 1 , 2 ,และ3** และเป็นพอร์ตขนาด 8 บิตทั้งหมด การใช้งานพอร์ตสามารถทำได้ทั้งในลักษณะของเส้นสัญญาณเดี่ยว ๆหรือกลุ่มของสัญญาณได้นอกจากนี้พอร์ต 0, 2 และ 3 ยังสามารถนำไปใช้งานอื่น ๆ ที่ไม่ใช่เป็นพอร์ตอินพุต/เอาต์พุตได้ โดยพอร์ต 0 จะทำหน้าที่มัลติเพล็กซ์ ระหว่างบัสแอดเดรสไบต์และบัสข้อมูล สำหรับการติด

การติดต่อกับวงจรประกอบร่วมกับข้อมูลที่มีอัตรสไบต์สูง ซึ่งจะส่งออกมาทางพอร์ต 2 ดังได้อธิบายในบทที่ผ่านมา สำหรับพอร์ต 3 นั้น นอกเหนือไปจากความสามารถเช่นพอร์ตปกติแล้ว สามารถนำไปเป็นขาสัญญาณของการอินเตอร์รัปต์ต่าง ๆ ซึ่งรวมทั้งการสร้างสัญญาณควบคุม RD และ WR เพื่อทำหน้าที่อ่านหรือเขียนหน่วยความจำข้อมูลภายนอกด้วย การใช้งานพอร์ตลักษณะงานแบบอื่น ๆ ที่ไม่ใช่เป็นพอร์ตแบบอินพุต/เอาต์พุตนี้จะดำเนินการโดย 8051 เองโดยอัตโนมัติ

โครงสร้างการทำงานของพอร์ต 8051

จากลักษณะโครงสร้างของแต่ละบิตภายในพอร์ตทั้งหมดของ 8051 ซึ่งได้แสดงไว้ในรูปที่ 17 นั้นจะเห็นว่ามีความคล้ายคลึงกันตามลักษณะโครงสร้างที่เรียกว่า Quasi-bidirectional port ยกเว้นพอร์ต 0 ซึ่งเพียงแต่ไม่มีตัวต้านทานทำหน้าที่ Pull-upสัญญาณภายในเท่านั้นวงจรประกอบอื่นภายในยังมีฟลิปฟลอปแบบ D ซึ่งมีผลทำให้พอร์ตสามารถแลตซ์หรือค้างสภาวะของสัญญาณได้ นอกจากนี้ในส่วนเอาต์พุตของฟลิปฟลอปเฉพาะของพอร์ต 0 และพอร์ต 2 จะมี



รูปที่ 17 โครงสร้างของแต่ละบิตภายในพอร์ตอินพุต/เอาต์พุตของ 8051

โครงสร้างที่ทำหน้าที่คล้ายกับสวิตช์เพิ่มเติมขึ้น เพื่อควบคุมให้เอาต์พุตนี้ต่อเข้ากับส่วนของทรานซิสเตอร์ในระหว่างที่ไม่ได้มีการทำงานในลักษณะ ของบัสแอดเดรสหรือบัสข้อมูลด้วย สำหรับบัฟเฟอร์จำนวนสองตัวของทุกบิตในพอร์ตนั้นมีการทำงานแยกกันโดยอิสระโดยตัวที่อยู่ทางด้านบนจะยอมให้สัญญาณผ่านได้ก็ต่อเมื่อมีการอ่านค่าข้อมูลที่ค้างไว้ ส่วนอีกตัวหนึ่งซึ่งอยู่ทางด้านล่างจะถูกใช้งานเฉพาะเมื่อได้มีการอ่านสถานะของขาสัญญาณเท่านั้น

การใช้งานพอร์ตเป็นการอินพุต

การใช้งานพอร์ตเป็นการอินพุตข้อมูลจะต้องเริ่มต้นด้วยค่าอินพุตที่เป็น 1 ออกมาทางพอร์ตนั้นก่อนเป็นอันดับแรก เพื่อหยุดการทำงานของทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาต์พุตของพอร์ตนั้น ทำให้ขาสัญญาณของบิตถูกต่อเข้ากับตัวตานทานซึ่งทำหน้าที่เป็นตัว Pull-up ภายในซึ่งมีผลให้บิตนั้นๆ ของพอร์ต1, 2 และ 3 เป็นสภาวะของลอจิกสูง ตัวต้านทานนี้มีค่าประมาณ 50 kΩ ซึ่งเป็นค่าที่สูงมาก และทำให้อุปกรณ์ภายนอกสามารถขับสัญญาณของพอร์ตเหล่านี้เป็นลอจิกต่ำได้ง่าย หรับบิตของพอร์ต0 นั้นแม้ว่าจะมีหลักการทำงานที่คล้ายคลึงกันกับบิตของพอร์ตอื่นๆ แต่เนื่องจากไม่มีตัวต้านทานทำหน้าที่ Pull-up ภายในไว้ ทำให้ทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณของพอร์ดหยุดการทำงานก็จะเป็นผลให้ขาสัญญาณนี้อยู่ในสภาวะอิมพีแดนซ์สูงแทน

การใช้งานพอร์ตเป็นการเอาต์พุต

เมื่อมีการส่งข้อมูลที่มีค่าเป็น 0 ให้กับแต่ละบิตของพอร์ตทุกพอร์ต ข้อมูลนี้จะถูกส่งให้กับฟลิปฟลอปซึ่งจะค้างค่านี้ไว้ และมีผลทำให้ทรานซิสเตอร์ที่ทำหน้าที่ขับ สัญญาณเอาต์พุตนั้นทำงานดังนั้นขาสัญญาณก็จะมีสภาวะลอจิกเป็นลอจิกต่ำด้วย

ส่วนการส่งข้อมูลที่มีค่าเป็น 1 ออกมานั้น ในกรณีที่เป็นการทำงานในแต่ละ บิตของพอร์ต 1,2 หรือ 3 จะทำให้ทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาต์พุตนั้นหยุดการทำงาน มีผลทำให้ขาของสัญญาณเป็นลอจิกสูงด้วยตัวต้านทานที่ Pull-up อยู่ภายในนั้น แต่สำหรับการทำงานในแต่ละบิตทางพอร์ต 0 นั้นจะมีผลที่แตกต่างออกไป โดยขาสัญญาณณจะเป็นสภาวะอิมพีแดนซ์สูงแทน เนื่องจากไม่มีตัวต้านทานภายในเชื่อมต่ออยู่นั่นเอง ดังนั้นในการใช้งานพอร์ต 0 เป็นการเอาต์พุตข้อมูล จึงจำเป็นต้องใช้ตัวต้านทานภายนอก Pull-upสัญญาณไว้กับลอจิกสูงแทน

ความสามารถอีกประการหนึ่งเกี่ยวกับพอร์ตอินพุต/เอาต์พุตของ 8051 เป็นวิธีการอ่านค่าลอจิกจากพอร์ตซึ่งมีได้สองวิธีคือ การอ่านค่าลอจิกที่ขาสัญญาณ (Port pin) และการอ่านค่าลอจิกของการแลตช์ที่พอร์ต (Port latch) ดังจะสังเกตได้จากรูปที่ 17 วิธีการอ่านค่าจากพอร์ตทั้งสองแบบนี้จะช่วยให้ระบบทำงาน ได้ด้วยความถูกต้องมากยิ่งขึ้น

ตัวตั้งเวลาและตัวนับ (TIMER/COUNTER)

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 จะมีรีจิสเตอร์ที่ทำหน้าที่เป็นTimer/Counter ขนาด16 บิทจำนวน 2 ตัว คือTimer 0 และTimerl แต่ถ้าเป็นเบอร์ 8052 จะมี Timer/Counter เพิ่มมาให้อีก 1 ตัว คือ Timer 2 โดย Timer/Counter ทั้งสามตัวเป็นรีจิสเตอร์ที่อยู่ในหน่วยความจำบริเวณ SFR ซึ่งผู้ใช้สามารถกำหนดการทำงานให้เป็น Timer หรือ Counter ได้อย่างใดอย่างหนึ่งโดยรายละเอียดในแต่ละอย่างดังนี้

ในโหมด Timer ค่าของรีจิสเตอร์จะถูกเพิ่มค่าทุกๆ machine cycle ดังนั้นจึงสามารถคิดว่าการทำงานในโหมดนี้เป็นการนับ machine cycle ก็ได้และเนื่องจากใน 1 machine cycle ของ MCS-51ประกอบไปด้วย 12 oscillator period ดังนั้นอัตราเร็วในการนับ (Count Rate) จึงมีค่าเป็น 1/12 ของความถี่ออสซิเลเตอร์ที่ใช้ในระบบ

ในโหมด Counter ค่าในรีจิสเตอร์จะถูกเพิ่มค่าครั้งละหนึ่งตามการเปลี่ยนสถานะที่ขา T0,T1 หรือ T2 (ใน 8052) โดยการเปลี่ยนสถานะต้องเปลี่ยนค่าจาก 1 เป็น 0 (1 to 0 transition) MCS-51 จะตรวจสอบสถานะของลอจิกที่ขา T0,T1 โดยตรวจสอบ (sampled) ระหว่าง S5P2 ของแต่ละ machine cycle รายละเอียดในการตรวจสอบการเปลี่ยนสถานะลอจิกที่แต่ละขาจะมีลักษณะที่เหมือนกันดังนี้

เมื่อสถานะลอจิกที่ขา T0,T1 หรือ T2 มีค่าเป็น 1 (high) ใน S5P2 ของ cycle ใดของแต่ละ cycle ถัดไปที่ S5P2 เช่นกันหากสถานะลอจิกที่ขา T0,T1 หรือ T2 มีค่าเป็น 0 (Low) จะส่งผลให้ค่าที่อยู่ในรีจิสเตอร์ (Timer/Counter Register) ถูกเพิ่มขึ้นอีก 1 ในช่วง P3P1 ของ cycle ที่ถัดจาก cycle ซึ่งตรวจพบการเปลี่ยนสถานะของลอจิก ดังนั้น MCS-51 จำเป็นจะต้องใช้ machine cycle จำนวน 2 cycle (24 Oscillator Period) เพื่อตรวจสอบการเปลี่ยนสถานะลอจิกจาก 0 เป็น1ที่ขา T0,T1 หรือ T2 จึงทำให้อัตราการนับสูงสุดของ Counter มีค่าเท่ากับ 1/24 ของความถี่ออสซิลเลเตอร์ที่ใช้ในระบบ โดยไม่มีข้อจำกัดในเรื่อง duty cycle (อัตราส่วนของช่วงเวลาที่สัญญาณมีสถานะลอจิกเป็น 1 ต่อคาบเวลาของสัญญาณ)ของสถานะของสัญญาณแต่เพื่อให้มั่นใจว่าสถานะลอจิกจะถูกตรวจสอบ (sampled)เข้ามาอย่างน้อย 1 ครั้งก่อนที่มันจะเปลี่ยนระดับจึงควรให้สถานะลอจิกของสัญญาณคงค่า 0 หรือ 1 ไว้อย่างน้อย 1 mechine cycle เต็ม

นอกจากผู้ใช้จะสามารถเลือกการทำงานของรีจีสเตอร์ให้เป็น Timer หรือ Counter ได้แล้ว MCS-51 ยังอนุญาติให้ผู้ใช้กำหนดให้ Timer หรือ Counter มีการทำงานาที่พิเศษแยกย่อยลงไปอีกถึง 4 แบบ (โหมด 0,1,2,3,) ตามความเหมาะสมของการใช้งาน (T0,T1 มีให้เลือก 4 แบบ แต่ T2 มีให้เลือก 3 แบบ) ดังจะได้กล่าวต่อไปนี้

ตัวตั้งเวลาที่ศูนย์และตัวตั้งเวลาที่หนึ่ง(TIMER 0 and TIMER 1)

Timer/Counter 2 ตัวนี้อยู่ในไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์ ผู้ใช้สามารถที่จะเลือกการทำงานให้เป็น TimerหรือCounter ได้โดยการเปลี่ยนค่าบิทในรีจีสเตอร์ TMOD โดย TMOD) ดังแสดงในรูปที่ 18 โดยหากบิทนี้มีค่าเป็น 0 หมายถึงเลือกให้เป็น Timer (นับจำนวน machine cycle) ถ้าบิทนี้มีค่าเป็น 1 หมายถึงเลือกให้เป็น Counter (นับจำนวนการเปลี่ยนสถานะจาก 1 เป็น 0 ที่ขา T0 T1)

TMOD: TIMER/COUNTER MEDE CONTROL REGISTER,ไม่สามารถอ้างถึงแบบบิตได้

ชื่อบิต:TMOD ตำแหน่ง: 89h ค่าบิตเริ่มต้น: 0000 0000

GATE1	C/T1	M1	M0	GATE0	C/T0	M1	M0

ชื่อบิต	ตำแหน่ง	ความหมาย
GATE1	TMOD.7	บิตควบคุม GATE สำหรับ Timer 1
C/T1	TMOD.6	บิตกำหนดการทำงานแบบตัวนับหรือจับเวลาของ Timer1 โดยถ้าเป็นค่า 0 จะทำหน้าที่เป็นตัวจับเวลา
M1	TMOD.5	บิตบนสำหรับการกำหนดโหมทำงานของ Timer1
M0	TMOD.4	บิตล่างสำหรับการกำหนดโหมทำงานของ Timer1
GATE0	TMOD.3	บิตควบคุม GATE สำหรับ Timer0
C/T0	TMOD.2	บิตกำหนดการทำงานแบบตัวนับหรือจับเวลาของ Timer0 โดยถ้าเป็นค่า 0 จะทำหน้าที่เป็นตัวจับเวลา
M1	TMOD.1	บิตบนสำหรับการกำหนดโหมดทำงานของ Timer 0
M0	TMOD.0	บิตล่างสำหรับการกำหนดโหมดทำงานของ Timer0

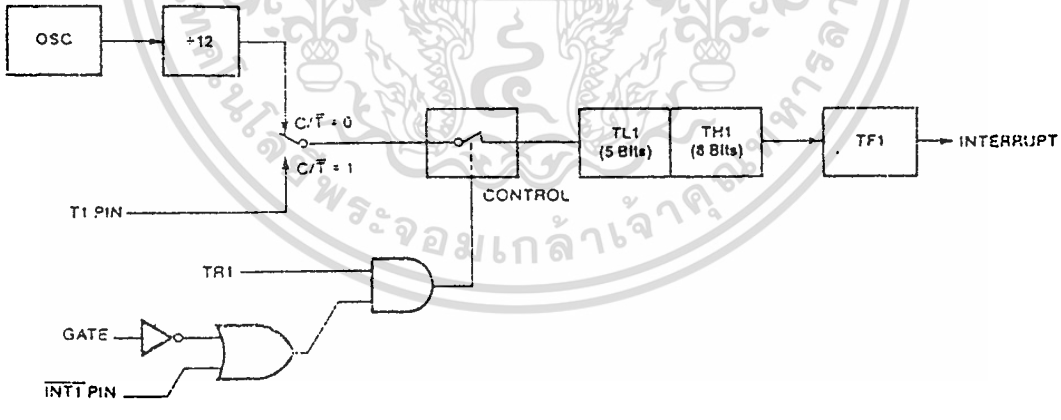
	SM1	MODE	ลักษณะการทำงาน	อัตราบอด
0	0	0	รีปรีจีสเตอร์	F ออสซิลเตอร์/ 12
0	1	1	8 - บิต UART	สามารถเปลี่ยนแปลงได้
1	0	2	9 - บิต UART	F ออสซิลเตอร์/32 หรือ F ออสซิลเตอร์/64
1	1	3	9 - บิต UART	สามารถเปลี่ยนแปลงได้

รูปที่ 18 ตารางแสดงบิทของ TMOD : Timer/Counter Mode Control Register

Timer/Counterทั้งสองตัวสามารถทำงานแตกต่างกันออกไป 4 แบบทั้งนี้โดยการเลือกด้วยค่าของบิท M0,M1 ในรีจิสเตอร์TMOD เช่นเดียวกับบิท C/T ดังแสดงในรูปที่ 1การทำงานในโหมด 0,1,2 จะคล้าย ๆ กันสำหรับ Timer/Counter ทั้งสอง แต่ในโหมด 3 ของ TIMER ทั้งสองจะทำงานที่ต่างกันออกไปจาก 3 โหมดแรก รายละเอียดการทำงานทั้ง 4 แบบของ TIMER ทั้งสองมีดังนี้

การทำงานของตัวตั้งเวลาในโหมดศูนย์ (Timer mode 0)

แต่ละ Timer/Counter ในโหมดนี้จะถูกกำหนดการทำงานให้เป็น Counter ขนาด 8 บิท ซึ่งจะถูกเพิ่มค่าต่อเมื่อสัญญาณจาก machine cycle หรือการตรวจพบการเปลี่ยนสถานะที่ขา T0,T1 ครบ 32ครั้งสแล้ว(เป็น Counter ขนาด 8 บิทที่ถูกหารด้วย 32)กล่าวอีกอย่างหนึ่งก็คือ Timer แต่ละตัวจะทำหน้าที่เป็นCounter เพื่อนับ machine cycle หรือการเปลี่ยนสถานะละจิกที่ขา T0,T1 ขนาด 13 บิทนั่นเอง โดยการนับสัญญาณ 32 ครั้งแรก จะใช้รีจิสเตอร์ Thx ซึ่งมีขนาด 8 บิททำให้Timer ทั้งสองทำงานเป็น Counter (นับจำนวน machine cycle หรือนับจำนวนการเปลี่ยนสถานะลอจิกที่ขา T0,T1 ขึ้นกับบิท C/T ในรีจิสเตอร์ TMODขนาด 13 บิท โดยมีการทำงานดังรูปที่ 19



รูปที่ 19 บล็อคการทำงานของTimer/counter 1 Mode 0 : 13 Bit Counter

จากรูปจะเห็นว่ารีจิสเตอร์ TL1และ TH1 ประกอบกันเข้าเป็น Counter ขนาด 13 บิท เมื่อค่าในรีจิสเตอร์ทั้งสองเปลี่ยนจาก 1 เป็น 0 ทั้งหมด จะมีผลไปเซ็ทบิท Timser Interpt Flag TF1 ซึ่งก่อให้เกิดอินเตอร์รัพท์ของ Timer ขึ้น (ควบคุมได้ด้วยรีจิสเตอร์ IE) อินพุทที่ใช้นับถูก enable ให้กับ Timer เมื่อบิท TR1 มีค่าเป็น 1 (ในรีจิสเตอร์ TCON ดังแสดงในภาพที่ 2)

และ Gate = 0 หรือ สถานะที่ขา INT1 เป็น 1 อย่างใดอย่างหนึ่ง การควบคุมด้วยบิททั้งสองมีดังแสดงในรูปที่ 19

การเซ็ทให้ Gate เป็น 1 จะยอมให้ Timer ถูกควบคุมโดยสถานะที่ขา INT1 ทำให้การนำ MCS-51 ไปใช้วัดความกว้างของพัลส์ทำได้ง่าย ส่วนการเลียร์ให้ Gate เป็น 0 จะมีผลให้การควบคุมอินพทที่ใช้นับ TIMER ทั้งสองกระทำได้โดยบิท TR1

บิท TR0,TR1 จะอยู่ในรีจิสเตอร์ TCON ดังแสดงในภาพที่ 2 ส่วนบิท Gate จะอยู่ในรีจิสเตอร์ TMOD ดังแสดงในภาพที่ 18 รีจิสเตอร์ขนาด 13 บิท ประกอบไปด้วย 8 บิทของรีจิสเตอร์ TH1 และ 5 บิทของรีจิสเตอร์ TL1 ส่วนบิทที่ 3 ของรีจิสเตอร์ TL1 ไม่ถูกกำหนดและไม่ถูกใช้ การเซ็ทค่าของบิท Run Flag (TR1) ก็ไม่ได้เคลียร์ค่าในรีจิสเตอร์ทั้งสอง

TCON: TIMER/COUNTER CONTROL REGISTER, สามารถอ้างถึงแบบบิตได้

ชื่อบิต: TCON ตำแหน่ง: 88h ค่าบิตเริ่มต้น:0000 0000

TF0	TR1	TF0	TR0	IE1	IT1	IE0	IT0

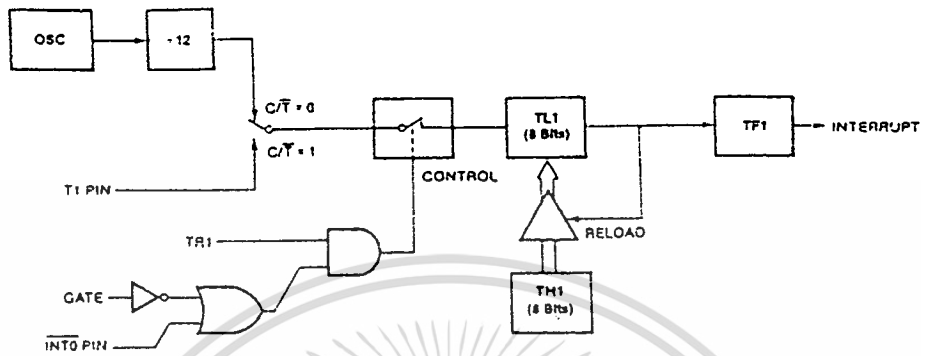
ชื่อบิต	ตำแหน่ง	ความหมาย
TF1	TCON.7	แฟล็กแสดงการโอเวอร์โฟลว์ของ Timer1
TR1	TCON.6	บิตกำหนดการทำงาน/หยุดทำงานของ Timer1
TF0	TCON.5	แฟล็กแสดงการโอเวอร์โฟลว์ของTimer0
TR0	TCON.4	บิตกำหนดการทำงาน/หยุดทำงานของ Timer0
IE1	TCON.3	แฟล็กแสดงการอินเตอร์รัปต์ของ INT1
IT1	TCON.2	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ INT1
IE0	TCON.1	แฟล็กแสดงการอินเตอร์รัปต์ของ INTO
IT0	TCON.0	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ INTO

รูปที่ 20 ตารางแสดงบิทของTCON : Timer/Counter Control Register

การทำงานของตัวตั้งเวลาในโหมดหนึ่ง (Timer mode 1)

การทำงานในโหมด 1 นี้มีการทำงานเหมือนในโหมด 0 ทุกประการ เว้นแต่ในรีจิส-เตอร์ Timer จะถูกใช้งานครบทั้ง 16 บิทเลย นั่นคือในโหมดนี้ Timer/Counter จะเป็น Counter หรือ Timer (ขึ้นกับบิท C/T ในรีจิสเตร์ TMOD เช่นกัน) ขนาด 16 บิทแทนการทำงานในโหมด 2 จะกำหนดให้รีจิสเตอร์ Timer/Counter ทำงานเป็น Counter ขนาด 8 bit (ใช้รีจิสเตอร์ Tlx) ซึ่งจะทำการโหลดค่าเองโดยอัตโนมัติด้วยค่าในรีจิสเตอร์ THxมื่อเกิด Overflow ในรีจิสเตอร์ Tlx ค่าในรีจิสเตอร์ THxนี้สามารถกำหนดได้ล่วงหน้าโดยซอฟท์แวร์ และจะไม่เปลี่ยนแปลงเมื่อถูกโหลดไปไว้ในรีจิสเตอร์ TL1 การทำงานของโหมด 2 ดังแสดงในรูปที่ 21

การทำงานในโหมดนี้มีไว้เพื่อใช้ในการกำเนิดสัญญาณอินเตอร์รัพท์เป็นจังหวะไปเรื่อยๆ หรือใช้เป็นตัวกำหนดเวลาการอินเตอร์รัพท์ที่คงที่



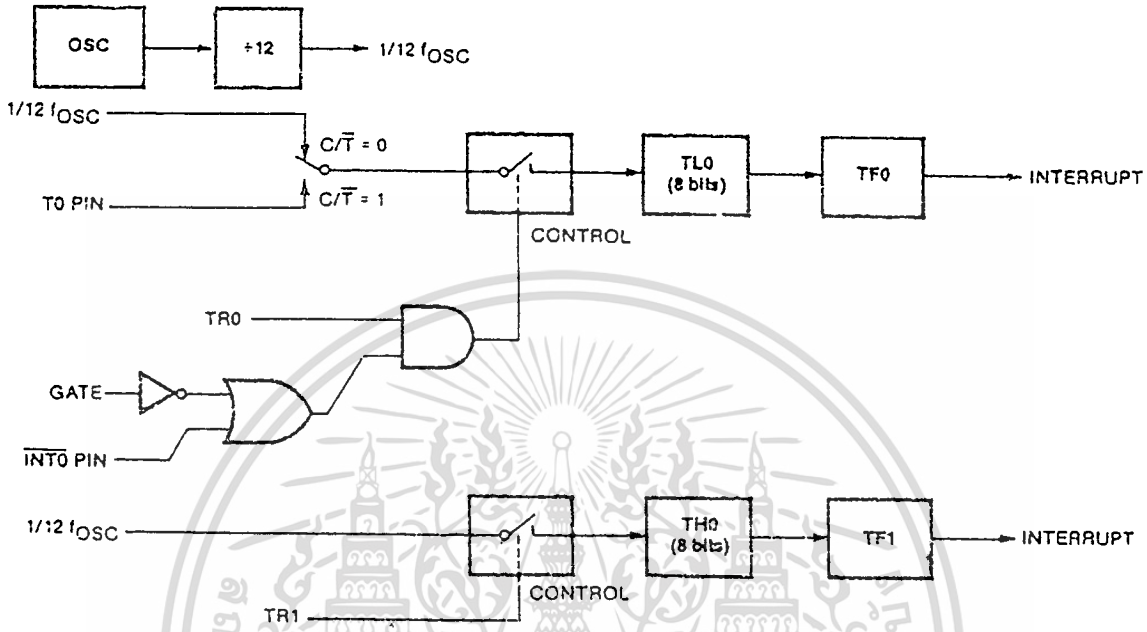
รูปที่ 21 บล็อคการทำงานของTimer/Counter 1 Mode 2 : 8 bit Auto reload

การทำงานของตัวตั้งเวลาในโหมดหนึ่ง (Timer mode 3)

ในการทำงานของ Timer/Counter โหมด 3 นี้ Timer 1 จะไม่มีการนับ ซึ่งมีผลเหมือนกับให้ค่า TR 1 = 0 แต่ Timer 0 จะมีการทำงานดังต่อไปนี้ Timer 0 ในโหมด 3 จะทำให้รีจิสเตอร์ TL0 และ TH0 ทำงานเหมือนเป็น Counter ขนาด 8 bit แยกต่างหากจากกัน 2 ตัว สำหรับการทำงานในโหมด 3 ของ Timer 0 แสดงในรูปที่ 22 ซึ่งจะเห็นว่ารีจิสเตอร์ TL0 ใช้การควบคุมจากบิท C/T,Gate,TR0,INT0 และ TF0 ส่วนรีจิสเตอร์ TH0 จะถูกบังคับให้เป็น Counter ที่นับจำนวน machine cycle และถูกควบคุมการทำงานโดยบิท TR1และTF1 ของ Timer 1 ดังนั้นขณะนี้รีจิสเตอร์ TH0 จะควบคุมการอินเาาตอร์รัพท์ของ Timer 1 ดังแสดงในรูปที่ 22

ในMODE 3 นี้ มีไว้เพื่อการใช้งานที่ต้องการ Timer หรือ Counter 8 บิท เพิ่มขึ้นดังนี้ เมื่อใช้ Timer 0 ในโหมด 3 8051 สามารถมองเหมือนว่ามันมี Timer/Counter 3 ตัว และ 8052 มี Timer/Counter 4 ตัว โดยเมื่อ Timer 0 กำลังทำงานอยู่ในโหมด 3 Timer 1 ยังสามารถใช้นับได้โดยการควบคุมสามารถทำได้ด้วยการบังคับให้ Timer 1สวิทช์ไปมาระหว่างโหมด 3 และโหมดอื่น(หาก Timer 1 อยู่ในโหมด 3จะหยุดการนับหากในโหมดอื่นจะนับไปเรื่อย ๆ) ดังนั้นจึงเปรียบเสมือนว่ามี Timer/Counter ใช้มากขึ้นอีก 1 ตัวจาก Timer1 (เพิ่มจากรีจิสเตอร์ TL0 และ TH0) โดยทั่วไปจะใช้ Timer 1เป็นตัวกำหนด Baud Rate แต่จริง ๆ แล้ว

Timer 1 สามารถถูกใช้งานใด ๆก็ได้ที่ไม่ต้องการการอินเตอร์รัพท์ ทั้งนี้เพราะการอิเตอร์รัพท์ของ Timer 1 ถูกใช้โดย Timer 0 ไปแล้วนั่นเอง



รูปที่ 22 บล็อคการทำงานTimer/Counter 0 Mode 3 : Two 8 bit Counters

การทำงานของตัวตั้งเวลาในโหมดหนึ่ง (Timer mode 3)

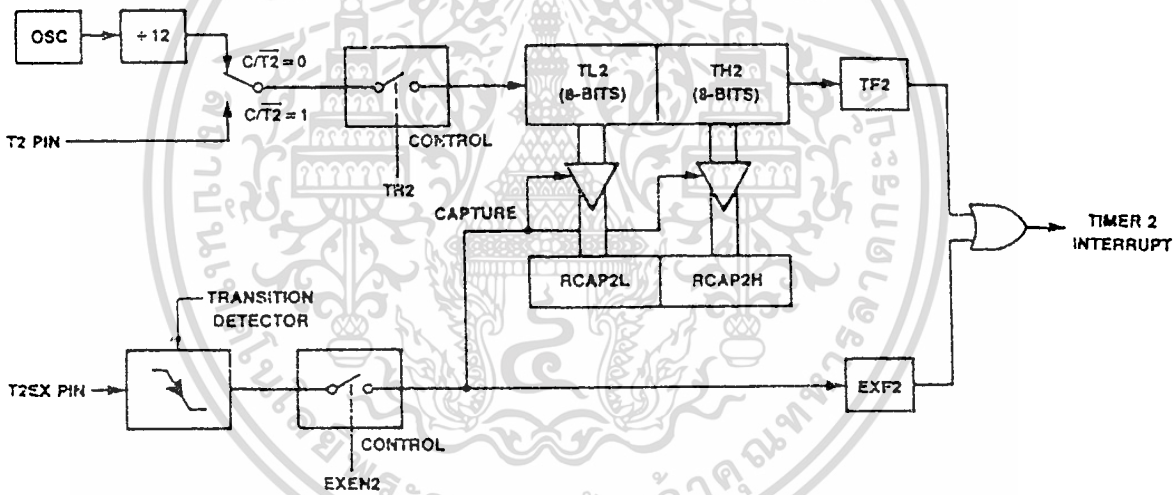
Timer 2 เป็น Timer/Counter ขนาด 16 บิทซึ่งมีเฉพาะใน 8052โดยมันสามารถทำงานเป็น Timer หรือ Counter อย่างใดอย่างหนึ่งเหมือน Timer 0 และTimer 1 ผู้ใช้สามารถเลือกการทำงานได้ด้วยบิท C/T2 ในรีจิสเตอร์ T2CON (รูปที่ 6) ใน Timer 2 นี้มีการทำงานอยู่ 3 โหมดด้วยกัน คือ Capture , Auto-Reload และ Baud Rate Generator ซึ่งสามารถเลือกได้โดยกำหนดค่าบิทในรีจิสเตอร์ T2CON ดังแสดงในตารางดังรูปที่ 23

RCLK+TCLK	CP/RL2	TR2	MODE
0	0	1	16-Bit auto-reload
0	1	1	16+Bit Capture
1	X	1	Baud Rate Generator
X	X	1	Off

รูปที่ 23 ตารางการกำหนดใช้งานในรีจิสเตอร์ TCON

การทำงานของแต่ละโหมดของ Timer 2 มีรายละเอียดดังนี้

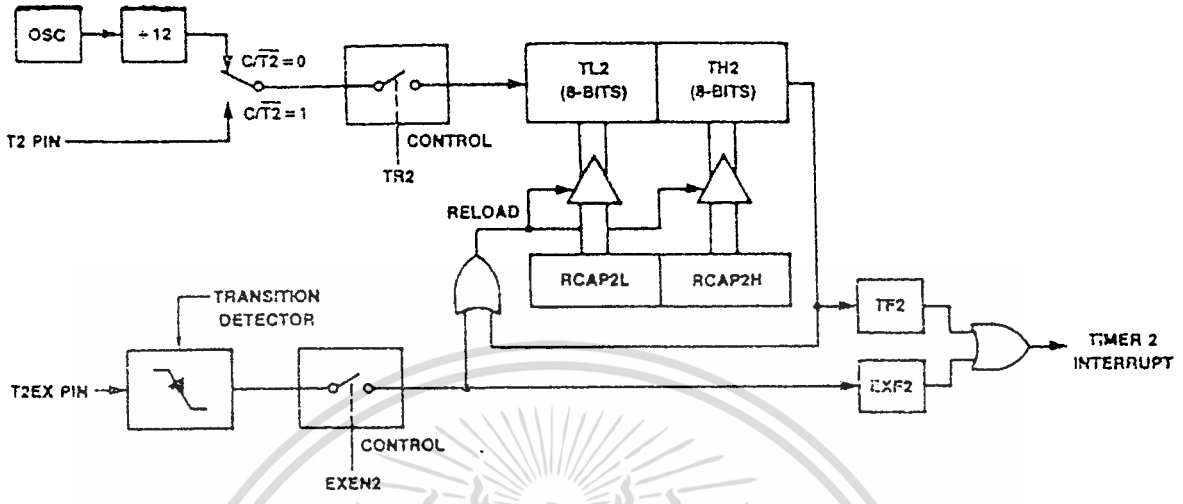
Capture Mode มี 2 Option ให้เลือกใช้คือสามารถถูกเลือกได้โดยบิท EXEN2 ในรีจิสเตอร์ T2CON ซึ่งถ้าบิทEXEN2 = 0 แล้วTimer 2 จะเป็นTimer หรือCounterขนาด16 บิท ซึ่งเมื่อเกิด Overflow จะไปเซ็ทบิท TF2 (Timer 2 Overflow Flag) ส่งผลให้เกิดอินเทอร์รัพท์ได้ แต่ถ้าบิท EXEN2 = 1 Timer2 จะยังคงทำงานเหมือนที่กล่างมาแล้ว แต่มีคุณสมบัติพิเศษที่เพิ่มเข้ามาคือสถานะละจิกที่มีการเปลี่ยนจาก 1 เป็น 0 ที่ขา T2EX ซึ่งจะทำให้ค่าปัจจุบัน (current value) ใน Timer 2 register (รีจิสเตอร์ TL2 , TH2) ถูกโหลด (Captured) ลงไปในรีจิสเตอร์ RCAP2L และ RCAP2H ตามลำดับ (RCAP2L และ RCAP2H เป็นรีจิสเตอร์ในบริเวณ SFR ที่มีเพิ่มขึ้นมาใน 8502) นอกจากนี้การเปลี่ยนระดับ (Transition) ที่ขา T2EX จะทำให้บิท EXF2 ใน T2CON ถูกเซ็ท และบิท EXF2 สามารถทำให้เกิดอินเทอร์รัพท์ได้เช่นเดียวกับ TF2 (Capture Mode สามารถอธิบายการทำงานได้ดังรูปที่ 24)



รูปที่ 24 บล็อคการทำงานของTimer 2 ใน Capture Mode

Auto-Reload Mode มี 2 Option ในการเลือกเช่นเดียวกัน ซึ่งถูกเลือกโดยบิท EXEN2 ในรีจิสเตอร์ T2CON โดยถ้าบิท EXXN2 = 0 แล้ว เมื่อค่าใน Timer 2 เปลี่ยนจาก 1 ไปเป็น 0 ทั้งหมด (นับครบ 16 บิท) ไม่เพียงแต่จะทำให้บิท TF2 ถูกเซ็ท แต่จะ ทำให้ค่าในรีจิสเตอร์ของ Timer 2 ถูกโหลดอีกครั้งด้วยค่า 16 บิทในรีจิสเตอร์ RCAP2L และ RCAP2H ซึ่งสามารถตั้งค่าไว้ล่วงหน้าได้ด้วยซอฟท์แวร์ ถ้าบิท EXEN2 = 1 Timer 2 จะยังคงทำงานเหมือนดังที่กล่าวมาแต่มีคุณสมบัติพิเศษเพิ่มขึ้นคือการเปลี่ยนสถานะจาก 1 เป็น 0 ที่ขา T2EX จะไปกระตุ้นให้มีการโหลดค่าจากรีจิสเตอร์ RCAP2L และ RCAP2H ไปยังรีจิสเตอร์ ของ Timer 2 ทันทีและเซ็ทบิท TF2 เช่นกัน กล่าวคือ การโหลดค่าจากรีจิสเตอร์ RCAP2L และ

RCAP2H ไปยังรีจิสเตอร์ของ Timer 2 จะเกิดขึ้นได้โดยไม่ต้องรอให้ Timer 2 เกิด Overflow นั่นเอง (ดูรูปที่ 25)



รูปที่ 25 บล็อคการทำงานของTimer 2 in Auto-Reload Mode

Baud Rate Generator Mode

ถูกเลือกโดยการให้บิท RCLK = 1 และ / หรือ TCLK = 1 ซึ่งจะอธิบายร่วมกับเรื่อง Serial Port ต่อไป และรายละเอียดของแต่ละบิทดังรูปที่ 26

T2CON: TIMER/COUNTER 2 CONTROL REGISTER สามารถอ้างถึงแบบบิตได้

ชื่อบิต: T2CON ตำแหน่ง: C8h ค่าบิตเริ่มต้น: 0000 0000

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2

ชื่อบิต	ตำแหน่ง	ความหมาย
TF2	T2CON.7	แฟล็กโอเวอร์โฟลว์ของ Timer 2
EXF2	T2CON.6	Timer 2 Extermal flag
RCLK	T2CON.5	Receive clock flag
TCLK	T2CON.4	Transmit clock flag
EXEN2	T2CON.3	limer 2 external enable flag
TR2	T2CON.2	แฟล็กกำหนดการทำงาน/หยุดการทำงานของ Timer2
C/T2	T2CON.1	แฟล็กเลือกโหมดการทำงานเป็นตัวนับหรือจับเวลา
CP/RL2	T2CON.0	แฟล็กแสดงโหมดการแสดง Capture/Reload

รูปที่ 26 ตารางแสดงบิทของ T2CON : Timer / Counter 2 Control Register

บทที่ 3
การสื่อสารข้อมูลแบบอนุกรมและการใช้งาน

การติดต่อแบบอนุกรม (SERIAL INTERFACE)

ใน MCS-51 มีSerial Portซึ่งสามารถรับและส่งข้อมูลแบบอนุกรมได้โดยผู้ใช้ไม่จำเป็นต้องต่อชิพอื่นเพิ่มอีกเลยทำให้มีความสะดวกในการนำไปประยุกต์ใช้งาน ที่ต้องมีการติดต่อข้อมูลแบบแนุกรมมาก Serial Port ที่มีใน MCS-51 สามารถทำงานได้ในแบบ Full Duplex หมายความว่ามันสามารถรับและส่งข้อมูล ได้พร้อม ๆ กัน โดยในการรับข้อมูลจะมีบพฟเฟอร์ให้ด้วย ทำให้ MCS-51 สามารถรับข้อมูลไบต์ที่สองซึ่งถูกส่งตามเข้ามาก่อนจะถูกอ่านจาก Receive Register ไปเก็บไว้ในหน่วยความจำ (แต่ถ้าไบต์แรกยังไม่ถูกอ่านเมื่อเวลาที่การรับของไบต์ที่สองสิ้นสุดลง หนางไบต์ในสองไบต์จะสูญหายไป)

Swrial Port จริง ๆ แล้วประกอบไปด้วยรีจิสเตอร์จนาด 8 บิท จำนวนสองตัว แต่ละตัวมีชื่อเรียกตามหนาที่ดังนี้คือ "รีจิสเตอร์ตัวรับ" (Receove Regoster) สำหรับการรับข้อมูล และ "รีจิสเตอร์ตัวส่ง" (Transmition Register) สำหรับการส่งข้อมูล รีจิสเตอร์ทั้งสองมีตำแหน่งเดียวกันใน SFR คือตรงกับตำแหน่งของรีจิสเตอร์ SBUF (99H) โดยการเข้าถึงข้อมูลของรีจิสเตอร์แต่ละตัว MCS-51จะทราบเองว่าผู้ใช้ต้องการติดต่อกับรีจิสเตอร์ ตัวใดโดยดูจากคำสั่ง ทั้งนี้เพราะในการเขียนข้อมูลไปที่รีจิสเตอร์ SBUF จะหมายถึงการโหลดค่าไปยัง Transmit register เพื่อส่งข้อมูลออกไปภายนอกส่วนการอ่านข้อมูลจากreceiveRegoster หมายถึงการรับข้อมูลจากภายนอกแบบอนุกรมการใช้งาน Serial PortในMCS-51มีความสะดวกและคล่องตัวสูงทั้งนี้เนื่องจากผู้ใช้สามารถกำหนดการทำงานแตกต่างกันได้ถึง 4ประเภท โดยการเลือกใช้งานที่มีแตกต่างกัน 4 ประเภทนี้มีจุดประสงค์ เพื่อความเหมาะสมกับการรับส่งข้อมูลแบบอนุกรมแต่ละชนิดดังนี้

ส่งข้อมูลแบบอนุกรมในโหมด 0 การทำงานแบบแรกหรือในโหมด 0 นี้ขา RXD มีไว้สำหรับการรับส่งข้อมูลแบบแนุกรมทั้งสองกรณี ส่วนขา TXD มีไว้เพื่อใช้กำเนิดสัญญาณ Shift Clock เพื่อเป็นตัวกำหนดจังหวะในการรับหรือส่งข้อมูล (ข้อมูลจะถูกรับหรือส่งตามสัญญาณ Shift Clock) ในโหมดนี้การรับส่งข้อมูลจะเป็นแบบ 8 บิท (8 Data Bits) โดยรับและส่งบิทต่ำสุดก่อน (LSB First) อัตราการรับส่งข้อมูล (BAUD Rate)ในการทำงานโหมด 0 ถูกกำหนดไว้ที่ 1/12 ของความถี่ ออสซิลเลเตอร์ที่ใช้ในระบบโดยในการทำงานโหมด 0 นี้จะไม่มี start bit และ stop bit เพราะข้อมูลที่รับหรือส่งออกไปถูกกำหนดโดยใช้สัญญาณจาก Shift Clock

ส่งข้อมูลแบบอนุกรมในโหมด 1 การทำงานแบบที่สอง หรือการทำงานโหมด 1 นี้ มีการรับและส่งข้อมูลครั้งละ 10 บิท โดยการส่งข้อมูลจะส่งผ่านทางขา TXD ส่วนการรับข้อมูลจะรับทางขา RXD ข้อมูลทั้ง 10 บิทจะประกอบไปด้วย 1 start bit(บิทที่ 0 มีค่าเป็น0) 8 data bits (รับและส่งบิทต่ำก่อน) และ 1 stop bit (มีค่าเป็น 1) โดยขณะทำการรับข้อมูลค่าของ stop bit ที่รับได้จะไปอยู่ในบิท RB 8 ของรีจิสเตอร์ SCON อัตราความเร็วในการรับหรือส่งข้อมูล (BAUD Rate) ของการรับและส่งข้อมูลในโหมดนี้สามารถเปลี่ยนแปลงได้ ดังจะได้กล่าวในรายละเอียดต่อไป

ส่งข้อมูลแบบอนุกรมในโหมด 2 การทำงานแบบที่สาม หรือการทำงานโหมด 2 จะมีการรับและส่งข้อมูลครั้งละ11บิทโดยข้อมูลถูกส่งผ่านทางขา TXD และรับมาผ่านทางขา RXD ข้อมูลที่รับและส่งทั้ง 11 ประกอบด้วย 1 start bit (บิทที่ 0 มีค่าเป็น 0) 8 data bits (บิทที่ 1-8) โดยรับหรือส่งบิทต่ำก่อนส่วนบิทที่ 9 เป็นบิทที่สามารถโปรแกรม ให้มีค่าเป็นศูนย์หรือหนึ่งก็ได้ (Programmable 9th Data bit)และบิทสุดท้ายคือ Stop bit (บิทที่ 10) จะมีค่าเป็น 1 เสมอ ดังนั้นจำนวนบิทที่รับส่งทั้งหมดจึงมี 11 บิท ตั้งแต่บิท 0 ถึงบิท 10 (1 Start bit + 8 Data bit + 1 Programmable bit + 1 stop bit)

ในขณะกำลังส่งข้อมูล บิทข้อมูลที่ 9 (the 9th data bit) จะเป็นค่าของบิท TB 8 ในรีจิสเตอร์ SCON ดังแสดงในรูปที่ 1 บิทนี้สามารถถูกกำหนดให้มีค่าเป็น 0 หรือ อย่างไรก็ได้ ส่วนใหญ่ในการใช้งานจริง มักจะใช้บิทนี้เป็นบิทสำหรับตรวจสอบข้อมูลที่รับหรือส่ง (parity bit) ว่าข้อมูลทารับส่งถูกต้องหรือไม่ โดยจะนำเอาบิท P (Psrity) ในรีจิสเตอร์ PSW ไปไว้ในบิท TB 8 ส่วนในขณะรับข้อมูล บิทข้อมูลบิทที่ 9 (the 9th data bit) จะไปปรากฏค่าอยู่ในบิท RB 8 ของรีจิสเตอร์ SCON โดยไม่สนใจ stop bit ค่า Baud Rate ในโหมดนี้สามารถตั้งให้เป็น 1/32 หรือ 1/64 ของความถี่ออสซิเลเตอร์ที่ใช้ในระบบ

ส่งข้อมูลแบบอนุกรมในโหมด 3 การทำงานขของ Serial Port แบบสุดท้าย คือ การทำงานในโหมด 3 ดยในการทำงานโหมดนี้ ข้อมูลจำนวน 11 บิทถูกส่งผ่านขา TXD และถูกรับผ่านขา RXD ข้อมูลทั้ง 11 บิทนี้ประกอบไปด้วย 1 start bit (บิท 0) 8 data bit (บิท 1-8) ส่วนบิทที่ 9 จะเป็นบิทที่โปรแกรมได้เหมือนในโหมด 2 (Programable 9th bit) และบิทสุดท้ายคือ 1 Stop bit (บิทที่ 10) อัตราความเร็วในการรับส่งข้อมูล (Baud Rate) สามารถเปลี่ยนแปลงได้ ดังจะได้ศึกษาในรายละเอียดต่อไป จะเห็นได้ว่าการรับส่งข้อมูลในโหมด 3 นี้จะเหมือนกับโหมด 2 ทุกอย่างยกเว้นค่า Baud Rate ซึ่งโหมดนี้สามารถเปลี่ยนแปลงเท่านั้น

การทำงานของ Serial Port ทั้ง 4 โหมดที่กล่าวมานี้ การส่งข้อมูลจะเริ่มทันทีเมื่อคำสั่งใด ๆ ที่ใช้รีจิสเตอร์ SBUF เป็นรีจิสเตอร์ปลายทาง (Destination Register) เช่น คำสั่ง MOV SBUF,A ส่วนในการรับข้อมูลจะเริ่มต้นโดยมีเงื่อนไขดังนี้

- ในโหมด 0 บิท R1 = 0 และ REN = 1

- ในโหมดอื่น ๆ การรับข้อมูลเริ่มเมื่อ MCS-51ได้รับ Start bitข้ามา ถ้าบิท REN = 1

การติดต่อสื่อสารแบบใช้ตัวประมวลผลหลายตัว

(MULTIPROCESSOR COMMNICATIONS)

การทำงานของ Serial Port โหมด 2 และ 3 ของ MCS-51 จะมีรูปแบบการใช้งานพิเศษนอกเหนือจากการรับส่งข้อมูลธรรมดา ตามที่ได้กล่าวมาแล้วการใช้งานพิเศษที่มีก็ คือ การติดต่อสื่อสารระหว่างซีพียูด้วยกันเอง (Multiprocessor communications) การทำงานแบบ Multiprocessor จะใช้เมื่อระบบประกอบไปด้วย ไมโครคอนโทรลเลอร์จำนวนหลานตัวเพื่อแบ่งแยกงานทำเป็นส่วน ๆ โดยมีไมโครคอน โทรลเสอร์ตัวหลักหรือตัวแม่ทำหน้าที่ควบคุมและรับข้อมูลจากไมโครคอนโทรลเลอร์ตัวลูกแต่ละตัวมาประมวลผล ดังนั้นทำให้ผู้ออกแบบระบบสามารถแบ่งงานที่ต้องการควบคุมทั้งหมดออกเป็นส่วนย่อย ๆ ให้แก่ไมโครคอนโทรล-เลอร์จำนวนมากกว่าหนึ่งตัวขึ้นไปทำการควบคุมแยกส่วนกัน โดยจะมีไมโครคอนโทรลเลอร์ตัวหลักหรือตัวแม่ทำหน้าที่ติดต่อสื่อสารกับไมโครคอนโทรลเลอร์ตัวลูกมาประมวลผล ทำให้ไมโครคอนโทรลเลอร์ตัวแม่สามารถควบคุมระบบทั้งหมดได้จากข้อมูลที่ได้รับมาจากไมโคร-คอนโทรลเลอร์ตัวลูก

ข้อดีของการทำงานในระบบ MULTIPROCESSER คือ

- ความน่าเชื่อถือของระบบที่สูงกว่าระบบที่ใช้ไมโครโปรเซสเซอร์ หรือไมโครคอนโทรลเลอร์เพียงตัวเดียวควบคุมการทำงาน

- ความเป็นโครงสร้างของระบบ (Module) ซึ่งทำให้การขยายระบบทำได้ง่าย ไม่ต้องกังวลกับการรบกวนการทำงาน ระหว่าง แต่ละส่วนอีกทั้งความเร็วในการควบคุมการทำงานรวมทั้งหมดก็ลดลงไปไม่มากนัก นอกจากนี้งานย่อยที่แบ่งการควบคุมสามารถเขียนโปรแกรมได้ง่ายกว่า เมื่อเปรียบเทียบกับการเขียนโปรแกรมสกหรับควบคะมงานย่อยที่เพิ่มขึ้นเมื่อใช้ไมโครคอนโทรลเลอร์ตัวเดียว การทำงานของ Serial Port โหมด 2 หรือ โหมด 3 ข้อมูลที่รับหรือส่งจะมีทั้งสิ้น 11 บิท ดังที่ได้กล่าวมาแล้วในตอนต้นโดยหากเรากำหนดให้ Serial Port โหมด 2 หรือโหมด 3 ทำงานในแบบ Mulriprocessor ด้วยการควบคุมที่บิท SM2ในรีจิสเตอร์ SCON ดังแสดงในรูปที่ 2 จะมีผลทำให้บิทที่ 9 ที่รับเข้ามาถูกนำไปไว้ที่บิท RB8 ของรีจิสเตอร์ SCON ตามด้วย stop bit เหมือนรับส่งข้อมูลตามทาได้กล่างมาแล้ว แต่เมื่อ stop bit ถูกรับเข้ามาแล้วหากในขณะนั้นบิทที่ RB8 มีค่าเป็น 1 อยู่จะทำให้มีปลไปกระตุ้นให้ฮาร์ดแวร์ในส่วน Serial Port Interrupt ให้เริ่มทำงานเพื่ออินเตอร์รัพท์ MCS-51 ต่อไป รายาละเอียดการทำงานมีดังต่อไปนี้คือ

เมื่อไมโครคอนโทรลเลอร์ตัวแม่ (Master Processor) ต้องการส่งข้อมูลจำนวนหนึ่งไปยังไมโครคอนโทรลเลอร์ตัวลูก(Slave)ตัวหนึ่งจากที่มีหลายตัวในระบบ ในขั้นแรกไมโครคอนโทรลเลอร์ตัวแม่จะต้องส่งข้อมูลขนาด 1 ไบต์ซึ่งค่านี้จะเป็นค่าที่ระบุตำแหน่ง(Address Byte) ของไมโครคอนโทรลเลอร์ตัวลูกที่เป็นเป้าหมายของ ไมโครคอนโทรลเลอร์ตัวแม่ต้องการติดต่อด้วยค่า Adress Byte ที่ไมโครคอนโทรลเลอร์ตัวแม่ส่งไปจะมีข้อแตกต่างจากข้อมูลทั่วไปที่รับส่งกันจริง ๆระหว่างไมโครคอนโทรลเลอร์ที่มีชื่อเรียกว่า Data byte ตรงที่บิทที่ 9 จะเป็น1 หากข้อมูลนั้นเป็น Adress Byte และจะเป็น 0 เมื่อข้อมูลนั้นเป็น Data Byte หากไมโครคอน-โทรลเลอร์ตัวลูกมีการเซ็ทบิท SM2 =1 แล้ว(บอกให้ MCS-51ทำงานในแบบ Multiprocessor) ถ้าข้อมูลที่รับเข้ามาเป็น Data Byte จะไม่สามารถอินเตอร์รัพท์ซีพียูได้ แต่ถ้าข้อมูลที่ได้รับเป็น Adress Byte (บิทที่ 9 มีค่าเป็น 1) ไมโครคอนโทรลเลอร์ตัวลูกทุกตัวจะถูกอินเตอร์รัพท์ การทำเช่นนี้เพื่อที่ไมโครคอนโทรลเลอร์ตัวลูกทุกตัวสามารถตรวจสอบได้ว่าข้อมูล Address Byte ที่ได้รับเข้ามาที่ค่าตรงกับตำแหน่งของตัวมันเองหรือไม่ หากไมโครคอนโทรลเลอร์ตัวลูกตัวใดมีค่าตำแหน่งของตัวมันเองตรงกับข้อมูล Adress Byte ที่ได้รับเข้ามาจะทำการเคลียร์บิท SM2 และเตรียมข้อมูลที่เป็น Data Byte ซึ่งจะตามเข้าหลังจากรับ Adress Byte เรียบร้อยแล้วส่วน ของไมโครคอนโทรลเลอร์ตัวลูกอื่น ๆ ที่ตรวจสอบข้อมูลดูแล้วปรากฏว่าไม่ตรงกับ Address ของตัวมันเองจะยังคงปล่อยให้บิท SM2 ของมันถูกเซ็ทต่อไป และกลับไปทำงานที่ค้างอยู่ก่อนได้รับการอินเตอร์รัพท์ต่อโดยไม่สนใจข้อมูลที่เป็น Data Byte ซึ่งตามเข้ามาหลัง Address Byte

บิท SM2 จะไม่มีผลในการทำงานของSerial Portในโหมด 0แต่การทำงานในโหมด1 บิท SM2 สามารถใช้เพื่อตรวจสอบ stop bit (Validity of the stopbit) โดยการรับข้อมูลของโหมด 1 ถ้าบิท SM2 = 1 การเกิด Receive Interrupt จะไม่ทำงานจนกว่า stop bit ที่รับเข้ามามีค่าเป็น 1

รีจิสเตอร์ ควบคุมการติดต่อแบบอนุกรม

รีจิสเตอร์ SCON (Serial Port Control And Status Register) เป็นรีจิสแตอร์ที่ใช้ควบคุมการทำงานของ Serial Port ภายใน MCS-51 โดยค่าของแต่ละบิทในรีจิสเตอร์ตัวนี้จะใช้สำหรับควบคุมและตรวจสอบการทำงานของ Serial Port ในMCS-51 ก่อนใช้งาน Serial Port ผู้เขียนโปรแกรมจำเป็นต้องทราบถึงความหมายของบิทต่างๆ ในรีจิสเตอร์ตัวนี้โดยค่าของข้อมูลแต่ละบิทของรีจิสเตอร์ SCON จะมีความหมายดังแสดงในรูปที่ 27

รีจิสเตอร์ตัวนี้ไม่เพียงแต่ใช้ควบคุมการทำงานของ Serial Port ในโหมดต่าง ๆ เท่านั้น เพราะบางบิทของรีจิสเตอร์นี้ยังใช้เป็นที่เก็บข้อมูลซึ่งเป็นบิทที่ 9 สำหรับการรับการส่งข้อมูลในโหมด 2 และ 3 (บิท TB 8 RB 8) และนอกจากนี้รีจิสเตอร์ SCON ยังมีบิทที่ทำหน้าที่อินเทอร์รัพท์ซีพียู (Serial Port Interupt) คือบิท TI และ RI อีกด้วย

ความเร็วในการส่งข้อมูล (Baud Rate)

Baud Rete หมายความถึงอัตราเร็วในการรับหรือส่งข้อมูล โดยใน MCS-51 ค่า Baud Rate สำหรับการรับหรือส่งข้อมูลจะมีค่าเท่าใดก็ขึ้นอยู่กับการทำงานในแต่ละ โหมดของ Serial Port ดังนี้

$$\text{ในโหมด 0 Baud Rate} = \frac{\text{ความถี่ออสซิสเลอร์ที่ใช้}}{12}$$

SCON: SERIAL PORT CONTROL REGISTER, สามารถอ้างถึงแบบบิตได้

ชื่อบิต: SCON ตำแหน่ง: 98h ค่าบิตเริ่มต้น: 0000 0000

SMO	SM1	SM2	REN	TB8	RB8	TI	RI

ชื่อบิต	ตำแหน่ง	ความหมาย
SM0	SCON.7	บิตเลือกโหมดการทำงาน
SM1	SCON.6	บิตเลือกโหมดการทำงาน
SM2	SCON.5	แฟล็กกำหนดการทำงานแบบมัลติโปรเซสเซอร์
REN	SCON.4	แฟล็กยอมให้มีการรับข้อมูล
TB8	SCON.3	ค่าของบิตที่ 9 สำหรับการส่งข้อมูลออก
RB8	SCON.2	ค่าของบิตที่ 9 ของข้อมูลที่รับเข้า
TI	SCON.1	แฟล็กแสดงการอินเตอร์รัปต์ภายหลังการส่งข้อมูล
RI	SCON.0	แฟล็กแสดงการอินเตอร์รัปต์เมื่อมีข้อมูลรับเข้า

รูปที่ 27 ตารางแสดงบิทของ SCON : Serial Port Control Register

ดังนั้นหากใช้คริสตอลความถี่ 12 MHzBaud Rate ของ Serial Port ในโหมด 0 จะมีค่าสูงถึง 1 Mhz เลยทีเดียว

- ในโหมด 2 ค่า Baud Rate ขึ้นอยู่กับค่าของบิท SMOD ซึ่งอยู่ในรีจิสเตอร์ PCON โดยถ้าบิท SMOD = 0 (ซึ่งเป็นค่านี้เมื่อเกิดการรีเซ็ท) Baud Rate จะเป็น 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้ แต่ถ้าบิท SMOD = 1Baud Rate จะเป็น 1/32 ของความถี่ออสซิลเลเตอร์ หรืออาจเขียนเป็นสูตรได้ว่า

$$\text{Baud Rate ในโหมด 2} = \frac{[2^{(\text{SMOD})} * (\text{Oscillator Frequency})]}{64}$$

ดังนั้น หากใช้คริสตรอความถี่ 12 Mhz ก็จะได้ความเร็วในการส่งข้อมูล(Baud Rate) สูงสุดในการทำงานโหมดนี้คือ 375K

- Baud Rate ในโหมด 1 และ 3 จะถูกกำหนดโดยอัตราการเกิด Over flow ของ Timer1(Timer 1 Over flow Rate) แต่ถ้าเป็น 8052 ซึ่งมี Timer / Counter เพิ่มมาอีก 1 ตัวเราสามารถใช้ Timer 2 ที่มีเพิ่มมานี้เป็นตัวกำหนด Baud Rate ได้ทำให้มี Timer จำนวน 2 ตัว (Timer 1 และ Timer 2) ที่สามารถนำมากำหนด Baud Rate โดยอาจใช้ตัวใดตัวหนึ่งในการกำหนด Baud Rate การรับ ส่วนอีกตัวหนึ่งสำหรับการส่งข้อมูล ทำให้การรับและการส่งมีค่า Baud Rate ที่แตกต่างกันได้

การใช้ Timer 1เป็นตัวสร้าง Baud Rates เมื่อ Timr1 ถูกใช้เป็นตัวกำหนด Baud Rate (Baud Rate Generator) สำหรับการทำงานของ Serial Port ในโหมด 1 และ 3 ค่าของ Baud Rate ที่ได้จะถูกกำหนดด้วยอัตราการเกิด Over flow ของ Timer 1 และขึ้นอยู่กับบิท SMOD ในรีจิสเตอร์ PCON ซึ่งเราอาจเขียนเป็นสมการที่ใช้คำนวณหา Baud Rate ได้ดังนี้

$$\text{Baud Rate โหมด 1,3} = \frac{2^{(SMOD)} * \text{Oscillator Frequency}}{32 * 12 * [256 - (TH1)]}$$

เราสามารถสร้าง Baud Rate ค่าต่ำ ๆ ได้ด้วย Timer 1 โดยการปล่อยให้ Timer 1 สามารถอินเตอร์รัพท์ซีพียูเพื่อทำการโหลดค่าเองใหม่ด้วยซอฟท์แวร์ขณะเกิด Overflow เนื่องจากในโหมด 16 บิทไม่มีการทำงานแบบ Auto-Reload ตารางในรูปที่ 28 เป็นค่า Baud Rate ค่าต่าง ๆ ที่ใช้กันมากและบอกว่าสามารถใช้ได้จาก Timer 1

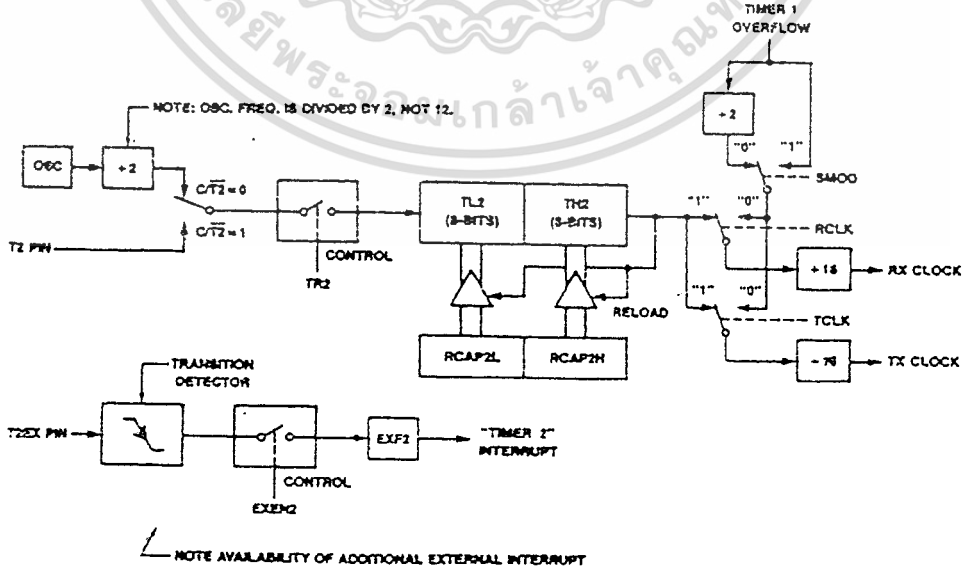
Baud Rate	Fosc	SMOD	Timer1		
			C/T	Mode	ReloadValue
Mode0 Max:1MHz	12MHz	X	X	X	X
Mode2 Max:375kHz	12MHz	1	X	X	X
Mode1,3Max:62.5kHz	12MHz	1	0	2	FFH
19.2KHz	11.059 MHz	1	0	2	FDH
9.6 KHz	11.059 MHz	0	0	2	FDH
4.8 KHz	11.059 MHz	0	0	2	FAH
2.4 KHz	11.059 MHz	0	0	2	F4H
1.2 KHz	11.059 MHz	0	0	2	E8H
137.5 KHz	11.986 MHz	0	0	2	1DH
110 KHz	6 MHz	0	0	2	72H
110 KHz	12MHz	0	0	1	FEEBH

รูปที่ 28 ตารางแสดงการเลือก Timer 1 ในการสร้าง Baud Rate

จากตารางจะเห็นว่าในการทำงานของ Serial Port โหมด 0 จะมีความเร็วในการส่งมากที่สุด เมื่อเปรียบเทียบกับโหมดอื่นที่คริสตอลความถี่เดียวกัน และ จะเห็นว่าการเลือกใช้คริสตอลความถี่ 11.0592 MH.z จะสามารถตั้งค่า Baud Rate ในโหมด 1 และ 3 เป็นมาตรฐานที่ใช้กันทั่วไปได้ เช่น 1200,2400,4800,9600,19200 จึงเป็นเหตุผลที่สำคัญในระบบควบคุมส่วนใหญ่เลือกใช้คริสตอลความถี่ 11.0592 MH.z มากกว่า 12 MH.z

ในตาราง 2 นอกจากจะแสดงค่าา Baud Rate ค่าต่าง ๆ เปรียบเทียให้เห็นแล้วตารางนี้ยังบอกค่าที่ต้องโหลดไปไว้ในรีจิสเตอร์ Timer/Counter ค่ามาตรฐานต่างๆให้เราทราบอีกด้วย ผู้เขียนโปรแกรมสามารถนำค่านี้ไปใช้ได้เลย แต่หากต้องการใช้ค่าBaud Rate อื่นที่นอกเหนือไปจากตารางนี้จะต้องคำนวณค่าที่ต้องโหลดให้กับ รีจิสเตอร์ของTimer/Counter เพื่อใช้สร้าง Baud Rate ด้วย

การใช้ Timer 2 ในการสร้าง Baud Rateใน 8052 ซึ่งมี Timer 2 เพิ่มขึ้นมาให้ผู้ใช้อีก 1 ตัว ซึ่งสามารถถูกเลือกให้มีการทำงานเป็นตัวกำหนด Baud Rate (Baud Rate Generator) ได้โดยการเซ็ทบิท TCLK และ/หรือ RCLK ในรีจิสเตอร์ T2CON (รูปที่ 7 ในเรื่อง Timer) จากรูปจะเห็นว่าบิท TCLK , RCLK จะเป็นสวิทช์เลือกระหว่างการใช้อัตราการเกิด Overflow ของ Timer 2 หรือ Timer 1 เป็นตัวสร้าง Baud Rate ดังนั้นในกรณีที่ใช้ 8052 ค่า Baud Rate สำหรับการส่งและการรับสามารถที่จะต่างกันได้ในเวลาเดียวกัน(ให้บิทRCLK มีค่าเป็น1 ส่วน TCLK มีค่าเป็น 0 หรือกลับกัน) การกำหนด RCLK และหรือ TCLK ให้เป็น 1 จะทำให้ Timer 2 ถูกเลือกใช้สร้าง Baud Rateและอยู่ในโหมดการสร้าง Baud Rate ดังแสดงในรูปที่ 29



รูปที่ 29 บล็อคแสดงการใช้ Timer 2 ในการสร้างBaud Rate

การใช้ Timer 2 ในการสร้าง Baud Rate (Baud Rate Generator Mode) มีการทำงานเหมือนโหมด Auto -Reload ตรงที่การเกิด Rollover ในรีจิสเตอร์ TH2 ทำให้ Timer 2 รีจิสเตอร์ (TL2 และ TH2) ถูกโหลดค้วยค่าขนาด 16 บิทจากรีจิสเตอร์ RCAP2L และ RCAP2H ซึ่งต้องได้รับการตั้งค่าไว้ล่วงหน้าด้วยซอฟท์แวร์เรียบร้อยแล้ว

เมื่อ Timer 2 ทำงานในโหมดนี้ Baud Rate ของการรับหรือการส่งข้อมูลในโหมด 1 และ 3 จะถูกกำหนดด้วยอุตราการเกิด Overflow ของรีจิสเตอร์ Timer 2 ตามสมการ

$$\text{Mode 1,3 Baud Rate} = \frac{\text{[Timer 2 Overflow Rate]}}{16}$$

Tuner สามารถถูกกำหนดสำหรับการทำงานเป็น Timer หรือ Counter อย่างใดอย่างหนึ่ง ในการทำงานทั่วไปที่นำ Timer/Counter มาใช้เป็นตัวกำหนด Baud Rate มักจะกำหนดให้เป็น Timer ซึ่งการทำงานในโหมด Timer ของ Timer 2 จะแตกต่างออกไปเล็กน้อยเมื่อมันถูกใช้ให้เป็นตัวกำหนด Baud Rate กล่าวคือ ปกติ Timer จะถูกเพิ่มค่าทุก ๆ machine cycle (ดังนั้นความถี่จึงเป็น 1/2 ของความถี่ออสซิลเลเตอร์) แต่เมื่อใช้เป็นตัวกำหนด Baud Rate มันจะถูกเพิ่มค่าทุก ๆ state time (ใน 1 machine cycle มี 6 state) นั่นคือใช้ความถี่เพียงครึ่งหนึ่งของความถี่ออสซิลเลเตอร์(1/2 ของ Pscillator frequency) ในกรณีนี้ Baud Rate จะถูกกำหนดโดยสมการ

$$\text{Mode 1,3 Baud Rate} = \frac{\text{[Oscillator Frequency]}}{\text{[32 * (65535 - (RCAP2H,RPAC2L)]}}$$

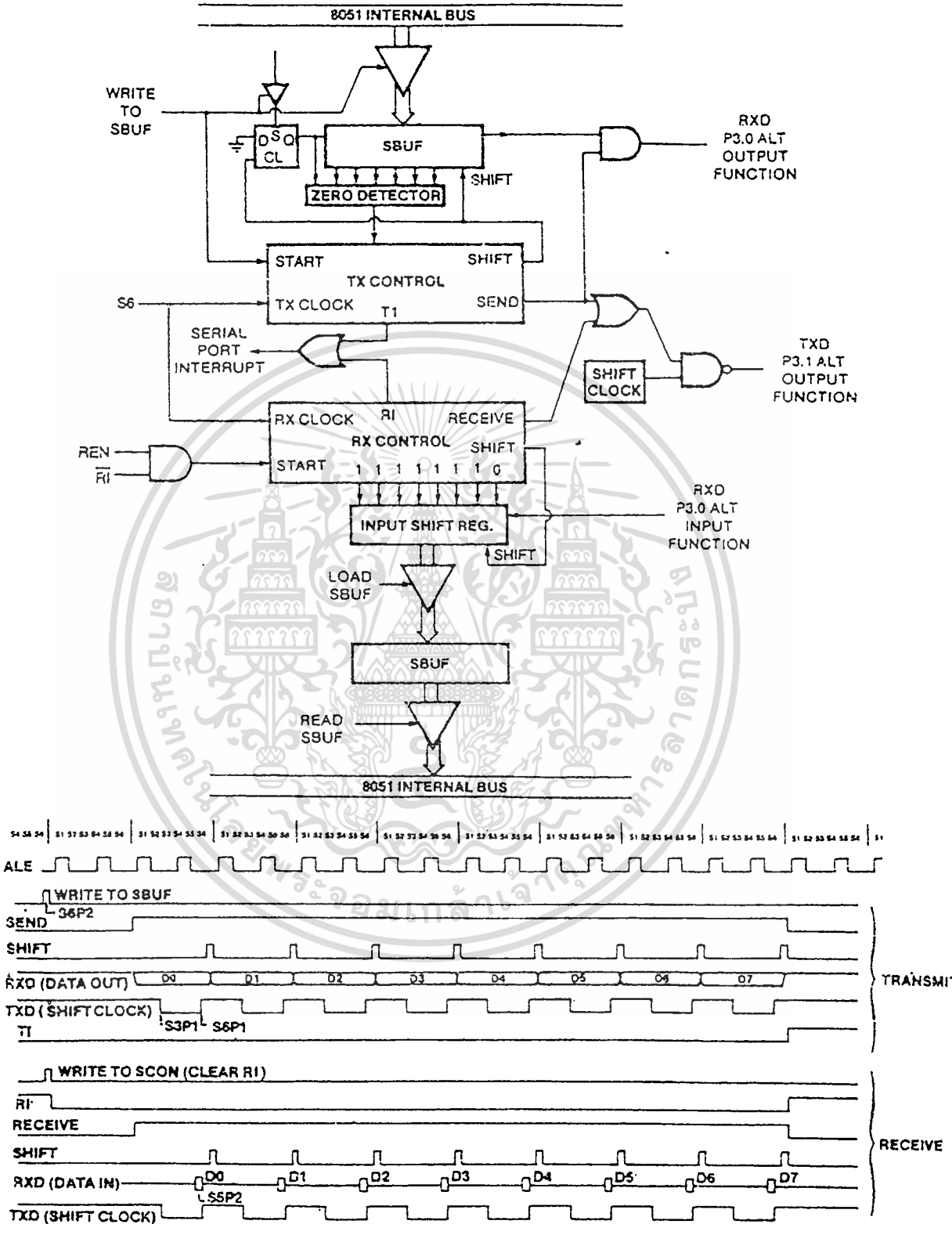
เมื่อ RCAP2L , RCAP2H เป็นที่ต้องโหลดไว้ในรีจิสเตอร์ RCAC2L และ RCAP2H ตามลำดับ ซึ่งถูกกำหนดเป็นจำนวนเต็ม ไม่คิดเครื่องหมาย ขนาด 16 บิท Timer 2 ในการทำงานเป็นตัวกำหนด Baud Rate มีรายละเอียดดังแสดงในรูปที่ 29 จากรูปนี้จะเห็นได้ว่า Timer 2 ถูกใช้เป็นตัวกำหนด Baud Rate ก็ต่อเมื่อบิท RCLK,TCLK ตัวใดตัวหนึ่งมีค่าเป็น 1 (RCLK + TCLK = 1) ในรีจิสเตอร์ T2CON (บิททั้งสองจะเป็นตัวเลือกระหว่างการใช้ Timer 1 หรือ Timer 2 เป็นตัวกำหนด Baud Rate) และเมื่อเปรียบเทียบกับภาพแสดงการทำงานของ Timer 2 จะสังเกตุได้ว่าการเกิด Rollover ในรีจิสเตอร์ TH2 จะไม่ทำให้บิท TF2 ถูกเซ็ท ดังนี้นการเกิด Rollover ของ Timer 2 เมื่อใช้เป็นตัวกำหนด Baud Rate จะไม่ทำให้เกิดอินเทอร์รัพท์ ผู้ใช้จึงไม่จำเป็นต้องห้ามการเกิดอินเทอร์รัพท์ของ Timer 2 ในระหว่างที่ถูกใช้เป็นตัวกำหนด Baud Rate และจากด้านล่างของรูปที่ 29 จะเห็นว่าถ้าบิท EXEN2 ถูกเซ็ทเป็น 1 การเปลี่ยนสถานะของสัญญาณจาก 1 เป็น 0 (1 to 0 transition) ที่ขา T2EX ของ

MCS-51 จะมีผลไปเซ็ทบิท EXF2ทำให้สามารถอินเทอร์รัพท์ซีพียูได้หากต้องการ โดยไม่มีการโหลดค่าจากรีจิสเตอร์ RCAP2L และ RCAP2H ไปยังรีจิสเตอร์ Timer 2 เลยทั้งนี้เพื่อให้ในระหว่างการใช้งาน Timer 2 เป็นตัวกำหนด Baud Rate ยังสามารถนำขา T2EX มาใช้เป็น External Inerrupt ได้ถ้าต้องการและจากรูปจะเห็นว่าบิท SMOD ไม่มีผลในการเพิ่มค่า Baud Rate สิ่งหนึ่งที่ควรระลึกไว้เสมอในการใช้งาน Timer 2 เป็นตัวกำหนด Baud Rate ผู้ใช้ไม่ควรพยายามอ่านหรือเขียนข้อมูลใด ๆ ลงไปยังรีจิสเตอร์ของ Timer 2 (TH2 , TL2) ทั้งนี้เพราะภายใต้ภาพการทำงานเช่นนี้ รีจิสเตอร์ของ Timer 2 จะถูกเพิ่มค่าทุกๆ state time ทำให้ข้อมูลที่ได้จากการอ่านหรือเขียนไม่เที่ยงตรง แต่เราอาจจะเขียนหรืออ่านข้อมูลจากรีจิสเตอร์ RCAP (RCAP2L , RCAP2H) ได้ถึงแม้การเขียนข้อมูลไปยังรีจิสเตอร์ทั้ง 2 นี้ อาจทำให้เกิดการผิดผลาดได้ หากข้อมูลที่เขียนลงไปตรงกับช่วงเวลาการโหลดค่าใหม่จากรีจิสเตอร์นี้ไปยังรีจิสเตอร์ของ Timer 2 ครั้งต่อไปพอดี (การเปลี่ยนค่าในรีจิสเตอร์ RCAP มีไว้เพื่อให้ผู้ใช้สามารถเปลี่ยนค่า Baud Rate ได้) เพราะฉะนั้น ในกรณีที่ผู้ใช้ต้องการเปลี่ยนแปลงค่าใด ๆ ในรีจิสเตอร์ของ Timer 2 และรีจิสเตอร์ RCAP ควรจะเคลียร์บิท TF2 เพื่อให้ Timer 2 หยุดนับเสียก่อน จากนั้นจึงสามารถกระทำการใด ๆ กับรีจิสเตอร์เหล่านี้ได้ตามต้องการ

การอธิบายการทำงานในโหมด 0 เพิ่มเติม

การทำงานในโหมดนี้ ดังได้อธิบายมาตอนต้นแล้วว่าข้อมูลถูกรับเข้าและ ส่งออกผ่านขา RXD ส่วนขา TXD ใช้เป็นตัวสร้าง Shift Clock เพื่อกำหนดจังหวะในการรับและส่งข้อมูลให้แก่วงจรภายนอก ในการทำงานโหมด 0 ข้อมูลถูกรับและส่ง โดยการเริ่มจาก start bit (0) ต่อด้วย data จำนวน 8 บิท โดยเริ่มที่บิทต่ำสุดก่อน (LSB First) และปิดท้ายด้วย stop bit (1) stop bit ที่รับได้จะถูกนำไปเก็บไว้ในบิท RB8 ของรีจิสเตอร์ SCON ค่า Baud Rate ในการทำงานโหมดนี้จะคงที่เท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้

ในรูปที่ 30 แสดงถึงโครงสร้างในการทำงานคร่าว ๆ ของ Serial Port ในโหมด 0 รวมทั้ง Timing diagram ที่เกี่ยวข้องสำหรับการส่งและการรับข้อมูล การส่งข้อมูลถูกทำให้เริ่มต้นโดยคำสั่งใด ๆ ที่ใช้รีจิสเตอร์ปลายทาง (destination register) สัญญาณ write to SBUF (ตามรูปที่ 30)ที่เกิดขึ้นขณะ S6P2จะเป็นการ โหลดค่า 1 ไปยังบิทที่ 9 ของ transmit shift register (จากรูป บิทที่ 9 คือ D-FLIP FLOF) และบอกให้วงจรควบคุมการส่งข้อมูล (TX Control Unit)เริ่มต้นส่งข้อมูลแผนผังเวลาภายในที่แสดงรายละเอียดการส่งข้อมูลจะเห็นได้ว่ามี 1 machine cykcle เต็มอยู่ระหว่างช่วงสัญญาณ write to SBUF และสัญญาณ SEND ที่ถูกกระตุ้นให้เริ่มทำงาน (ขณะเริ่ม active) ดังแสดงในรูปที่ 30



รูปที่ 30 บล็อคแสดงการทำงานและแผนภาพเวลาของ Serial Port Mode 0

สัญญาณ SEND ที่ถูกกระตุ้นให้ทำงาน (มีค่าเป็น 1) จะเป็นสัญญาณออกจากวงจรควบคุมการส่งข้อมูลเพื่อไปควบคุมให้ shift register ส่งข้อมูลออกไปทีละบิท ข้อมูลที่ส่งออกมาจะผ่านออกไปยังขา RXD ซึ่งตรงกับ P3.0 และสัญญาณ SEND นี้ยังไปควบคุมให้วงจร SHIP CLOCK ซึ่งทำหน้าที่สร้าง shift clock ผ่านออกไปทางขา TXD ซึ่งตรงกับ P3.1 โดย shift clock จะเป็นสัญญาณที่มีค่าเป็น Low ระหว่าง S6,S1,S2 ของทุก ๆmachine cycle ดังแสดงในภาพเมื่อถึง S6P2 ของทุก ๆ machine cycle ซึ่งสัญญาณ SEND ยังคงทำงาน (enable) อยู่ ข้อมูลของ transmit shift register จะถูกเลื่อนไปทางขวา 1 ตำแหน่ง ส่งผลให้ข้อมูลผ่านออกมาทางขา RXD ทีละ 1 บิท โดยเริ่มจากบิทที่ต่ำสุดก่อน ขณะที่ข้อมูลซึ่งถูกเลื่อนออกมาทางขวาทีละบิทนี้ ค่า 0 จะถูกนำมาแทนที่ตำแหน่งซึ่งถูกเลื่อนออกไป โดยจะเลื่อนเข้ามาทางด้านซ้ายของ transmit sthft register (มาจาก D-FLIP FLOP ที่มีขา D ต่อลงกราวนด์) การส่งข้อมูลจะดำเนินไปเรื่อย ๆ จนกระทั่งข้อมูลที่เป็นบิทสูงสุด (MSB) ที่ถูกโหลดไว้ในรีจิสเตอร์ SBUF ถูกเลื่อนไปอยู่ที่ตำแหน่งเอาท์พุท (Output position) ของ shift register พร้อมกับ 1 ที่ถูกโหลดไปที่ตำแหน่งที่ 9 ในตอนเริ่มต้นการส่งข้อมูลที่กล่าวในตอนแรกจะอยู่ติดถัดจากบิทสูงสุดมาทางซ้าย โดยทุกตำแหน่งทางซ้ายของมันจะมีค่าเป็น 0 หมด (มี 0 ทางซ้ายรวม 7 ตัว) ในสภาวะเช่นนี้จะส่งผลให้วงจรตรวจจับศูนย์ (ZERO Detector) ซึ่งทำงานโดยตรวจจับเพียง 7 บิท เริ่มทำงาน โดยการส่งสัญญาณไปบอกวงจรควบคุมการส่งข้อมูลให้ทำการเลื่อนข้อมูลครั้งสุดท้ายอีก 1 ครั้งเพื่อส่งบิทสูงสุดออกไป แต่บิทที่ 9 จะไม่ถูกส่งตามออกมา โดยจะยังคงอยู่ที่ตำแหน่งเอาท์พุทของ shift register เท่านั้น จากนั้นการส่งข้อมูลจะสิ้นสุดลงโดยวงจรส่วนควบคุมการส่งข้อมูลจะบังคับให้สัญญาณ SEND หยุดทำงาน (deactived SEND) เมื่อข้อมูลและบิทในรีจิสเตอร์ SBUF ถูกส่งออกไปภายนอกเรียบร้อยแล้ว บิท TI จะถูกเซ็ทเพื่ออินเทอร์รัพท์ซีพียูการกระทำทั้งสองนี้จะเกิดขึ้นที่ขณะ S1P1 ของ machine cycle ที่ 10 หลังจากที่มีสัญญาณ Write to SBUF

การรับข้อมูลในโหมด 0 เกิดขึ้นได้ก็ต่อเมื่อบิท REN = 1 และ RI = 0 (พิจารณาจากรูปที่ 30) โดยขณะ S6P2 ของ machine cycle ถัดไป วงจรควบคุมการรับ (RX control block) จะทำการเขียนข้อมูล 11111110B ไปที่ Receive shift register และในสัญญาณนาฬิกาในเฟสถัดไปสัญญาณ Receiveจากวงจรควบคุมการรับข้อมูลจะเริ่มต้นทำงาน(Active)โดยให้สัญญาณที่ได้จากการควบคุม NAND GATE สามารถส่งสัญญาณ Shift clock ที่ได้จะมีการเปลี่ยนสถานะทุก ๆ S3P1 และ S6P1 ของ ทุก ๆ machine cycle ในช่วงการรับข้อมูล โดยขณะ S6P2 ของแต่ละ machine cycle ซึ่งสัญญาณ Receive ยังคงทำงานอยู่ ข้อมูลของ Receive Shift Register จะถูกเลื่อนเข้ามาทางซ้าย 1 ตำแหน่งเนื่องจากมีข้อมูลที่รับได้จากภายนอกถูกเลื่อนจากทางขวามาไว้ที่รีจิสเตอร์ SBUF ค่าของข้อมูลที่เข้ามาจากทางขวาจะเป็นข้อมูลได้รับการตรวจสอบ (sampled) ที่ขา RXD ในช่วง S5P2 ของ machine cycle เดียวกัน

ขณะที่ข้อมูลถูกเลื่อนเข้าจากทางขวาทีละบิท ค่าของ 1 ช่วงถูกนำไปไว้ใน Rexeive shift register ในตอนแรกจะถูกเลื่อนออกไปทางซ้าย (Shift out) เมื่อ 0 ซึ่งถูกโหลดไปไว้ที่บิทต่ำสุดตั้งแต่ตอนแรก (11111110) ถูกเลื่อนมาอยู่ในตำแหน่งซ้ายสุดของ Shift Register จะทำให้เกิดสัญญาณส่งไปบอกวงจรส่วนควบคุมการรับข้อมูลอีก 1 บิท โดยการเลื่อนบิทมาทางซ้ายเมื่ออีกครั้งและส่งสัญญาณ LOAD SBUF เพื่อให้ข้อมูลเข้า ไปอยู่ในรีจิสเตอร์ SBUF การกระทำทั้งหมดนี้จะเกิดขึ้นในช่วง S1P1 ของ machine cycle ที่ 10 หลังจากที่มีการโหลดค่าไปที่รีจิสเตอร์ SCON ซึ่งไปเครียร์บิท R1 (ดูรายละเอียดจากรูปที่ 30) เมื่อการรับข้อมูลสิ้นสุดลงแล้ว สัญญาณ RECEIVE ขณะหยุดทำงาน และบิท RI จะถูกเซ็ทอีกครั้ง

ในการส่งและการรับข้อมูลของ Serial Port ในโหมด 0 นี้ จะพบว่าจังหวะในการรับกรือส่งข้อมูลของวงจรภายนอกที่รับข้อมูลขึ้นอยู่กับmachine cycle ของซีพียู นั่นคือ ข้อมูลจะถูกรับเข้ามาหรือส่งออกไปทุก ๆ machine cycle จึงทำ Baud Rate ของการทำงานโหมด 0 มีค่าเท่ากับ 1/12 ของความถี่ออสซิสเลเตอร์ (เท่ากับความถี่ของ machine cycle)

สัญญาณ Shift Clock ที่เกิดขึ้นขณะส่งข้อมูลจะเป็นสัญญาณเพื่อควบคุมจังหวะในการรับข้อมูลของวงจรภายนอกที่รับข้อมูลจาก MCS-51 ส่วนสัญญาณ Shift Clock ที่เกิดขึ้นขณะ MCS-51 รับข้อมูลจากภายนอกจะถูกสร้างขึ้น เพื่อเป็นสัญญาณบอกให้วงจรภายนอกส่งสัญญาณให้แก่ MCS-51

อธิบายการทำงานในโหมด 1 เพิ่มเติม

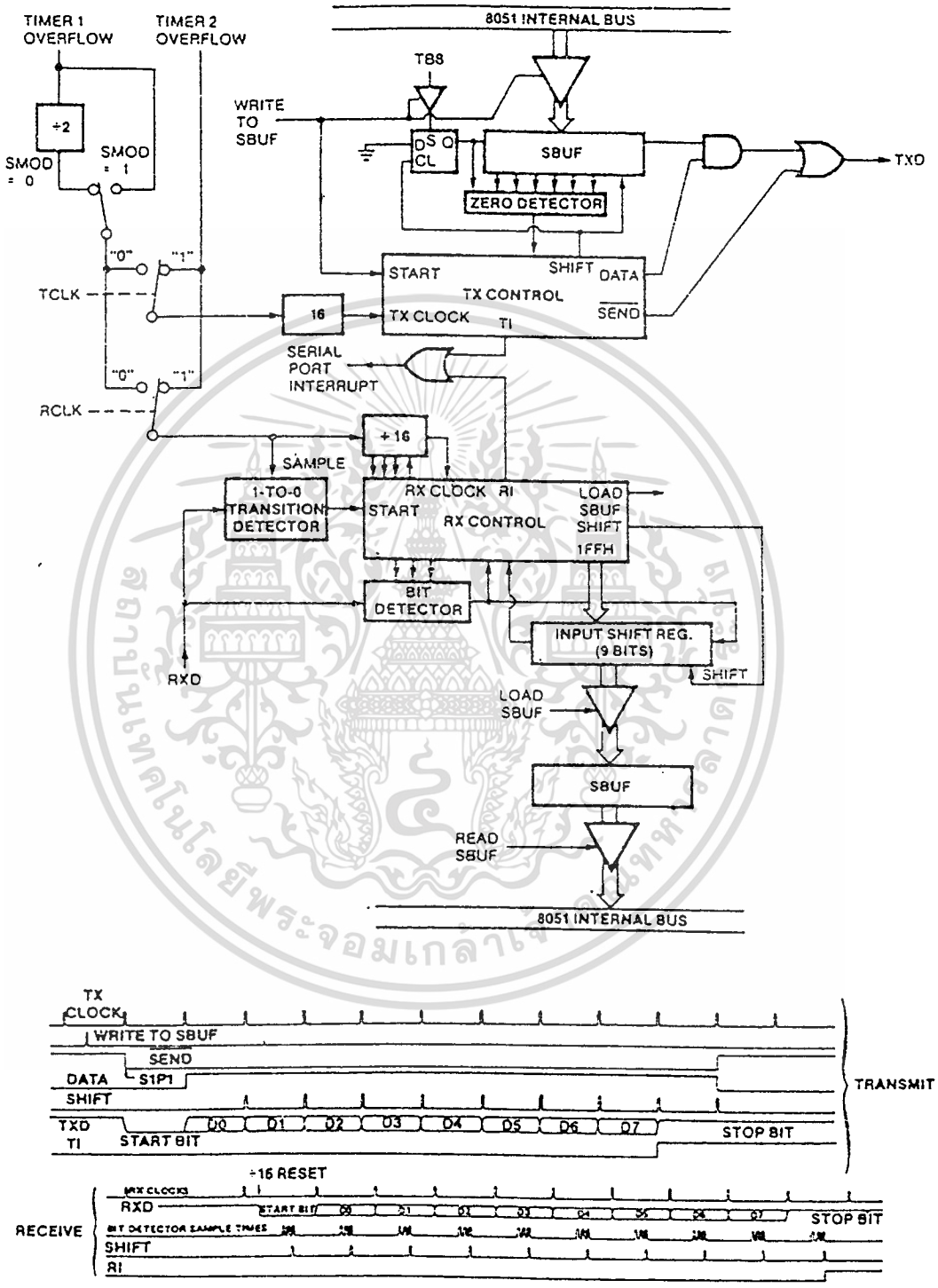
การทำงานของ Serial Port ในโหมด 1 นั้น ดังกล่าวมาแล้วว่าข้อมูล 10 บิทถูกส่งผ่านขา TXD และรับเข้าทางขา RXD โดยข้อมูลจะประกอบไปด้วย 1 start bit (0) 8 data bit (รับหรือส่งบิทต่ำสุดก่อน) ตามด้วย 1 stop bit (1) ในการรับ stop bit จะถูกนำไปเก็บไว้ในบิท RB8 ของรีจิสเตอร์ SCON ใน 8051 Baud Rate ถูกกำหนดโดยอัตราการเกิด Overflow ของ Timer 1 (Timer 1 Overflow Rate) ส่วนใน 8052 ซึ่งมี Timer 2 เพิ่มขึ้นมา ดังนั้น Baud Rate สามารถถูกกำหนดโดยอัตราการเกิด Overflow ของ Timer 1 หรือ Timer 2 อย่างใดอย่างหนึ่งหรือทั้งสองอย่าง (ตัวหนึ่งสำหรับการส่งและอีกตัวสำหรับการรับ) โดยใช้บิทRCLK และบิท TCLK เป็นตัวเลือก (รูปที่ 20ในเรื่อง Timer/Counter) รูปที่ 31 แสดงการทำงานของ Serial Port ในโหมดนี้อย่างคร่าว ๆ รวมทั้งแผนผังเวลาที่เกี่ยวข้องสำหรับการรับและการส่งข้อมูล

การส่งข้อมูลจะเริ่มต้นโดยคำสั่งใด ๆ ที่ใช้รีจิสเตอร์ SBUF เป็นรีจิสเตอร์ปลายทาง (Destination Register) สัญญาณ Write to SBUF มีอินพุทต่อลงกราวน์ และขา S ต่อกับสัญญาณWrite to SBUF) และจะไปกระตุ้นให้วงจรควบคุมการส่งข้อมูลทราบว่าขณะนี้กำลังมีความต้องการที่จะส่งข้อมูล การส่งข้อมูลจริงๆจะเริ่มในช่วง S1P1 ของ machine cycleที่ถัดจากการเกิด Rollover ครั้ง ถัดไปใน Counter ที่ถูกหารด้วย 16 (Counter จะถูกเพิ่มค่าเมื่อนับ

สัญญาณที่เป็นพัลส์ได้ครบ 16 ลูก)ดังนั้นการส่งข้อมูลแต่ละบิทจะสอดคล้อง (Synchronized) กับสัญญาณที่ได้จาก Counter ที่ถูกหารด้วย 16 ไม่ใช่กับสัญญาณ Wriet to SBUF

การส่งเริ่มต้นด้วยสัญญาณ SEND (Ative ที่สถานะ 0) ที่ถูกกระตุ้นให้ทำงาน และทำการใส่ start bit (0) ไปยังขา TXD เมื่อเวลาของการส่งบิทแรกผ่านไป ข้อมูลจะถูกส่งออกมาทีละบิทจากTransmit Shift Registerโดยบิทที่ส่งออกมาจะมาจากตำแหน่งเอาท์พุทของรีจิสเตอร์ไปที่ขา TXD สัญญาณ Shift จะให้ pulse ทีละสูกภายหลังการส่งข้อมูลผ่านไป 1 บิทขณะที่บิทข้อมูลถูกเลื่อนออกไปทางขวาค่า 0 จะถูกเลื่อนเข้ามาทางซ้าย (จาก D-FLIP FLOP)เมื่อบิทสูงสุดของข้อมูลอยู่ที่ตำแหน่งเอาท์พุทของ shift register แล้ว 1 ซึ่งถูกโหลดไปไว้ในตำแหน่งที่ 9 ในตอนแรกจะอยู่ถัดจากบิทสูงสุดไปทางซ้าย ส่วนตำแหน่งอื่น ๆทางซ้ายมีค่าเป็น 0 ทั้งหมด ในสภาวะเช่นนี้วงจรตรวจจับศูนย์ (ขนาด 7 บิท) จะเริ่มต้นทำงานการส่ง สัญญาณไปบอกวงจรควบคุมการส่งข้อมูลให้ทำการเลื่อนข้อมูล เป็นครั้งสุดท้ายอีก 1 ครั้งแล้วจึงหยุดการส่งข้อมูลพร้อมทั้งเซ็ทบิท T1เพื่ออินเทอร์รัพท์ซีพียู เหตุการณ์ที่เกิดขณะการส่งข้อมูลเสร็จสิ้นลงจะเกิดในช่วง Rollover ครั้งที่ 10 ของ Counter ที่หาร 16 หลังจากสัญญาณ Write to SBUF เกิดขึ้นการรับข้อมูลเริ่มต้นเมื่อมีการครวจพบการเปลี่ยนสถานะจาก 1 เป็น0 ที่ขาRXD (ตรวจพบ start bit) ซึ่งการตรวจจับสถานะนี้จะถูก Sampled ด้วยอัตรา 16 (อัตราเดียวกับการเกิด Rolloverซึ่งถูกส่งต่อไปให้Counter หาร 16 ต่อไป) ไม่ว่าค่า Baud Rate จะถูกกำหนดเป็นเท่าใดก็ตามเมื่อการเปลี่ยนสถานะของสัญญาณถูกตรวจพบ Counter หาร 16 จะถูกรีเซ็ททันทีและข้อมูล1FFH จะถูกนำไปเก็บไว้ในInput Shift Registerการรีเซ็ทCounter หาร 16 ก็เพื่อตั้งการ Rollover ให้ตรงกับการเริ่มต้นการรับข้อมูลของบิทต่อไปที่จะรับเข้ามา

จากรูป Counter จะถูกนับขึ้นไป 1 ก็ต่อเมื่อมีการเกิด Overflow ขึ้น 16 ครั้งหรือกล่าวง่าย ๆ ได้ว่า เมื่อเกิด Overflow 16 ครั้ง ค่าของ Counter จะถูกเพิ่มขึ้นทีละ 1 เสมอซึ่งเวลาในการรับข้อมูลแต่ละบิทจะนำเอาอัตราการเกิด Overflowมาเป็นตัวตรวจสอบนั่นคือแบ่งเวลาในการรับข้อมูลแต่ละบิทออกเป็น16 stateโดยมีตัวตรวจสอบข้อมูลในแต่ละบิท (BIT DETECTER) ทุก ๆ state ที่ 7,8,9 ของเวลาในข้อมูลแต่ละบิท ตัวตรวจสอบบิทจะตรวจสอบค่าสถานะที่ขา RXD ซึ่งมีข้อมูลรออยู่ซึ่งค่าที่รับเข้ามาจะถือว่าเป็น ข้อมูลที่ถูกต้องก็ต่อเมื่อการตรวจสอบค่าสถานะเป็นค่าเดียวกันอย่างน้อย 2 ใน 3ของ state ที่ 7,8,9 ที่ต้องทำเช่นนี้เพื่อเป็นการป้องกันสัญญาณรบกวน (noise rejection) แต่ในช่วงของการรับบิทแรก (start bit) ถ้าค่าที่ตรวจสอบได้ใน state 7,8,9 ปรากฏว่าไม่เป็น 0 วงจรที่ทำหน้าที่รับข้อมูลจะถูกเซ็ทและระบบจะเริ่มกลับไปตรวจหาการเปลี่ยนสถานะจาก 1 เป็น 0 ที่ขา RXD ต่อไป ที่ต้องทำเช่นนี้ก็เพื่อยกเลิกการรับข้อมูลหาก tart bit ไม่ถูกต้อง แต่ถ้า start bit ถูกตรวจสอบแล้วปรากฏว่ามีค่าเป็น 1 การรับข้อมูลบิทต่อไปจะเริ่มต้นทันที โดย start bit จะถูก shift ไปไว้ในInput Shift Register และบิทข้อมูลที่เหลือจะถูกรับตามมา



รูปที่ 31 บล็อคแสดงการทำงานของ Serial Port Mode 1 แบบใช้ TCLK , RCLK and Timer 2 are Present in the 8052 Only

ขณะที่ data bit เข้ามาทางขวา1(1FFH ที่ถูกโหลดไว้ก่อน)จะถูก shift out ไปทางซ้าย จนกระทั้งเมื่อ start bit ถูกเลื่อนไปถึงตำแหน่งซ้ายสุดในShift Register (ในโหมด 1 รีจิสเตอร์นี้จะมีขนาด 9 บิท) มันจะส่งสัญญาณไปออก RX control block ให้ทำการ shift ครั้งสุดท้ายอีก 1 ครั้ง เพื่อรับ stop bit ไปไว้ในบิทRB8 รีจิสเตอร์ SCON จากนั้นจึงส่งสัญญาณไปโหลดข้อมูลจาก Input Shift Register เพื่อนำไปเก็บไว้ในรีจิสเตอร์ SBUF และนำ stop bit ไปเก็บไว้ใน RB8 รวมทั้งการเซ็ทบิทRI จะถูกสร้างขึ้นก็ต่อเมื่อเงื่อนไขต่อไปนี้เป็นจริงขณะสัญญาณ shift pulse ลูกสุดท้ายถูกสร้างขึ้น

- บิท R1 ถูกเคลียร์ (R1 = 0) และ
- บิท SM2 = 0 หรือ stop bit = 1

ถ้าเงื่อนไขอย่างใดอย่างหนึ่งของทั้งสองอย่างนี้ไม่ตรงกลุ่มข้อมูลที่รับเข้ามาจะหายไปโดยไม่สามารถเรียกคืนได้ ถ้าเงื่อนไขทั้งสองถูกต้อง stop bit จะถูกนำไปเก็บไว้ในบิท RB8 ในรีจิสเตอร์ SCON ส่วนบิทที่เป็นข้อมูล 8 บิท (data 8 bit) จะถูกโหลดไปไว้ในรีจิสเตอร์ SBUF และบิท RI จะถูกเซ็ท เมื่อถึงตอนนี้ไม่ว่าเงื่อนไขข้างต้นจะตรงหรือไม่ ระบบการรับส่งข้อมูลจะถือว่าการรับข้อมูลจำนวน1ไบต์(รับข้อมูลเข้ามา10 บิทแต่เป็นdata จริง ๆ เพียง 8 บิท เสร็จสิ้นลงแล้ว และเริ่มเพื่อรับข้อมูลไบต์ต่อไป

อธิบายการทำงานในโหมด 2 และ 3 เพิ่มเติม

การทำงานของSerial Port ในโหมดนี้มีการรับและส่งข้อมูลครั้งละ11บิทโดยส่งข้อมูลผ่านขา TXD และรับข้อมูลผ่านขา RXD เหมือนทุกโหมดที่กล่าวมาแล้ว แต่ข้อมูลทั้ง 11 บิท จะประกอบไปด้วย 1 Start bit (0) , 8 data bits (รับและส่ง LSB ก่อน) , 1 Programable 9th bit และ 1 stop bit (1) โดยบิทที่มีเพิ่มเข้ามาในโหมดนี้คือบิทที่ 9 ซึ่งกำหนดให้เป็น 0 หรือ 1 ได้ ส่วนในการรับบิทที่ 9 จะไปปรากฏอยู่ที่บิท RB8 ในรีจิสเตอร์ SCON เช่นเดียวกับการส่งในโหมด 2 ค่า Baud Rate สามารถกำหนดให้เป็น 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้ ส่วนในโหมด 3 Baud Rateสามารถแปรค่าได้ โดยถูกกำหนดด้วย Timer 1 หรือ Timer 2 อย่างใดอย่างหนึ่งโดยมีบิท RCLK และ TCLK เป็นสวิทช์เลือก

รูปที่ 32 และ 33 แสดงแผนผังการทำงานของ Serial Port ในโหมด 2 และ โหมด 3 ในส่วนของการรับข้อมูลจะเหมือนกับในโหมด 3 ทุกอย่างแต่ในส่วนของ การส่งจะต่างกันกับโหมด 1 เพียงในเรื่องบิทที่ 9 ของ transmet shift register และส่งสัญญาณไปบอก TX control block ว่าขณะนี้กำลังาต้องการส่งข้อมูล การส่งข้อมูลที่เริ่มที่ S1P1 ของmachine cycle ถัดจากการเกิด Rollover ใน counter หาร 16 ครั้งถัดไป (ดังนั้นช่วงเวลาในการส่งแต่ละบิทจะซิงโครไนซ์กับ Counter หาร 16 ไม่ใช่กับสัญญาณ Write to SBUF)

การส่งข้อมูลเริ่มต้นด้วยการกระตุ้นให้สัญญาณ SEND ทำงาน ซึ่งจะใส่ค่า start bit ไปที่ขา TXD เมื่อเวลสของการส่งบิทแรกผ่านไป สัญญาณ data จะถูกกระตุ้นให้ทำงานตามมา ซึ่งจะเป็นการ enable output bit ของ transmit shift register ไปที่ขา TXD shift pulse ลูกแรกเกิดขึ้น (ดูในภาพที่ 14) เมื่อเวลาในการส่งผ่านไป 1 บิท (1 bit time) สัญญาณ shift ที่จะไปทำให้ 1 (ต่อไปจะกลายเป็น stop bit) ถูกใส่เข้าไปในตำแหน่งที่ 9 ของ shift register หลังจากนั้นจะมีเพียงที่ถูกใส่เข้าไปในShift Registerดังนั้นขณะที่ data bit ถูกshift out ออกไปทางขวา ศูนย์จะถูกใส่เข้าไปทางซ้ายการส่งจะดำเนินไปเรื่อย ๆ จนกระทั่งบิทTB8 ซึ่งถูกใส่ไว้ในตำแหน่งที่ 9 ของ shift register ในตอนแรก ถูกเลื่อนไปอยู่ที่ตำแหน่ง out put ของ shift register และทางด้านซ้ายของบิท TB8 จะมีค่าเป็น 1 (stop bit) จากการโหลดด้วย shift pulse ลูกแรก ส่วนตำแหน่งอื่นที่เหลือทางด้านซ้ายจะมีค่าเป็น 0 หมด สถาวะที่มี 0 อยู่ด้านซ้าย ของ stop bit จะถูกตรวจด้วยวงจรตรวจจับ 0 (ZERO DETECTOR) ซึ่งเมื่อตรวจจับได้ก็จะส่งสัญญาณไปบอก TX control block ให้ทำการ shift ครั้งสุดท้ายอีกหนึ่งครั้งแล้วจึงค่อยให้สัญญาณ SEND หยุดทำงาน รวมทั้งเซ็ทบิท TI เพื่อทำการ Rollover ครั้งที่ 11 ของ Counter หาร 16 หลังจากสัญญาณ Write to SBUF ถูกกระตุ้น

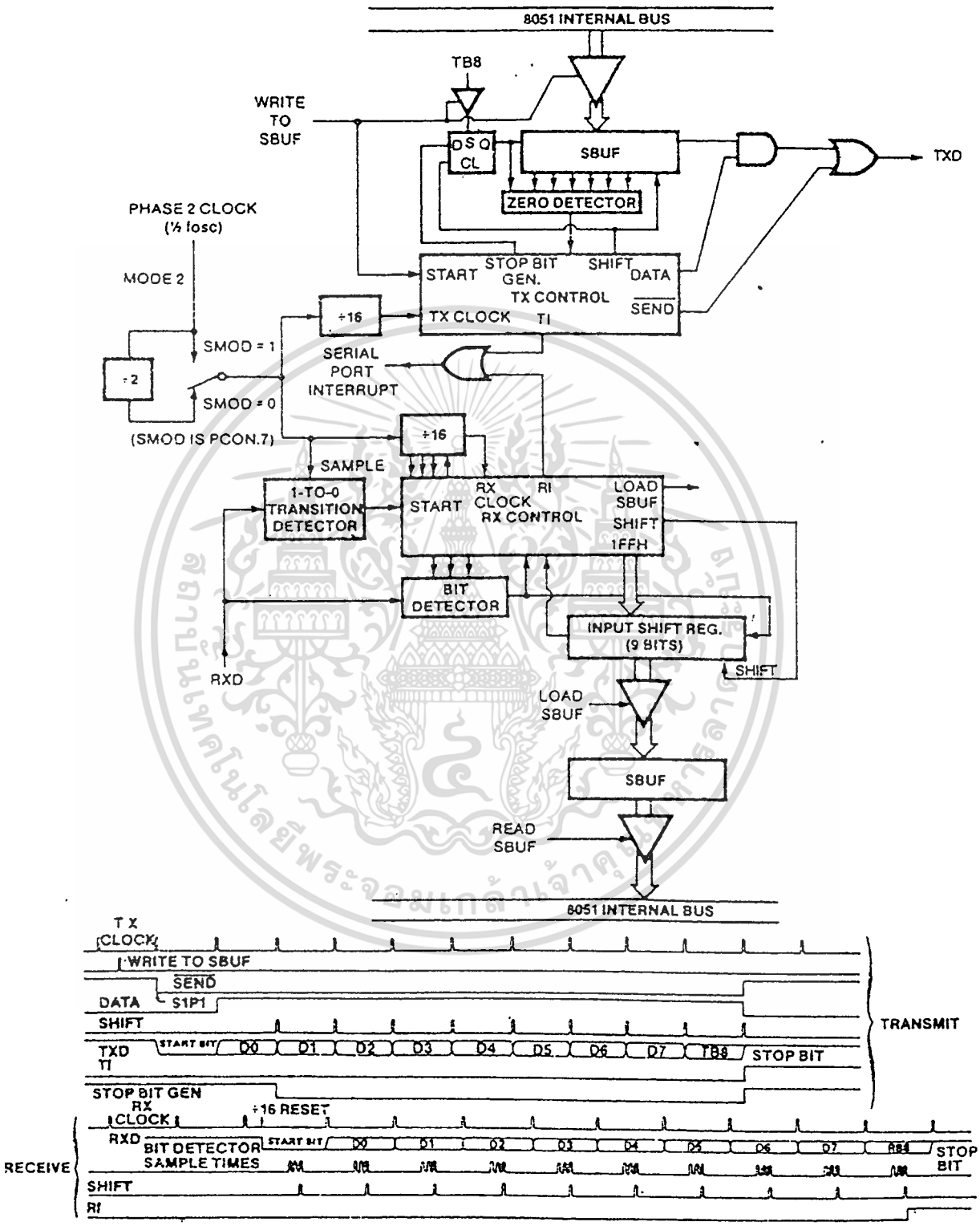
การรับข้อมูลเริ่มต้นด้วยการตรวจพบการเปลี่ยนสถานะจาก 1เป็น0ที่ขาRXD สำหรับจุดประสงค์นี้RXD จะถูกตรวจสอบด้วยอัตรา16 ครั้งเท่ากับการเกิดRollover ไม่ว่า Baud Rate จะถูกกำหนดให้มีค่าเป็นเท่าใดก็ตาม เหมือนในโหมด 1 เมื่อการเปลี่ยนแปลงถูกตรวจพบ Counter ซึ่งถูกหารด้วย 16 ถูกรีเซ็ททันที และค่า 1 FFH จะถูกเขียนไปที่ Input Shift Register ที่ state 7,8,9 ของเวลาในการรับข้อมูลแต่ละบิท ตัวสอบบิทจะตรวจสอบค่าของ RXD ซึ่งค่าที่ถูกรับเข้ามาเป็นข้อมูลจะต้องตรวจพบอย่างน้อย 2 ใน 3 ของการตรวจทั้ง 3 ครั้ง เพื่อเป็นการป้องกันสัญญาณรบกวนที่อาจจะเกิดขึ้นได้ แต่ถ้าบิทแรกที่ถูกรับเข้ามาถูกตรวจสอบพบว่าไม่เป็น 0 วงจรรับข้อมูลจะถูกรีเซ็ท และกลับไปเริ่มตรวจการเปลี่ยนสถานะที่ขา RXD ตามเดิม ทั้งนี้เพื่อไแน่ใจว่าบิทแรกที่เข้ามาเป็น start bit แน่นอนนั่นเองหากแรกตรวจพบว่ามีค่าเป็น 1 นั่นคือบิทนี้เป็น start bit แน่นอน การรับข้อมูลจะดำเนินต่อไปโดยจะทำการ Shift start bit ไปเก็บไว้ที่ Input shift register

ขณะที่ data bit ถูก shift มาทางขวาใน Input Shift register จะทำให้ 1FFH ซึ่งถูกนำไปไว้ที่ Input shift register ในตอนแรกถูก shift out ไปทางซ้าย เมื่อ start bit (1) ถูกเลื่อนไปถึงตำแหน่งซ้ายสุดในshift register(ในโหมด2 และ3 รีจิสเตอร์ขนาด 9 บิท)มันจะส่งสัญญาณไปบอกRXControl block ให้ทำการ shift ครั้งสุดท้างอีกหนึ่งครั้ง แล้วจึงส่งสัญญาณไปโหลดเอาข้อมูลจาก shift register มาไว้ที่รีจิสเตอร์ SBUF และนำบิทที่รับเข้ามาครั้งสุดท้ายไปเก็บไว้ที่บิท RB8 รวมทั้งเซ็ทบิท RI ซึ่งเหตุการณ์ทั้งหมดก็ต่อเมื่อเป็นไปตามเงื่อนไขต่อไปนี้ เมื่อสัญญาณ pulse ลูกสุดท้ายถูกสร้างขึ้น

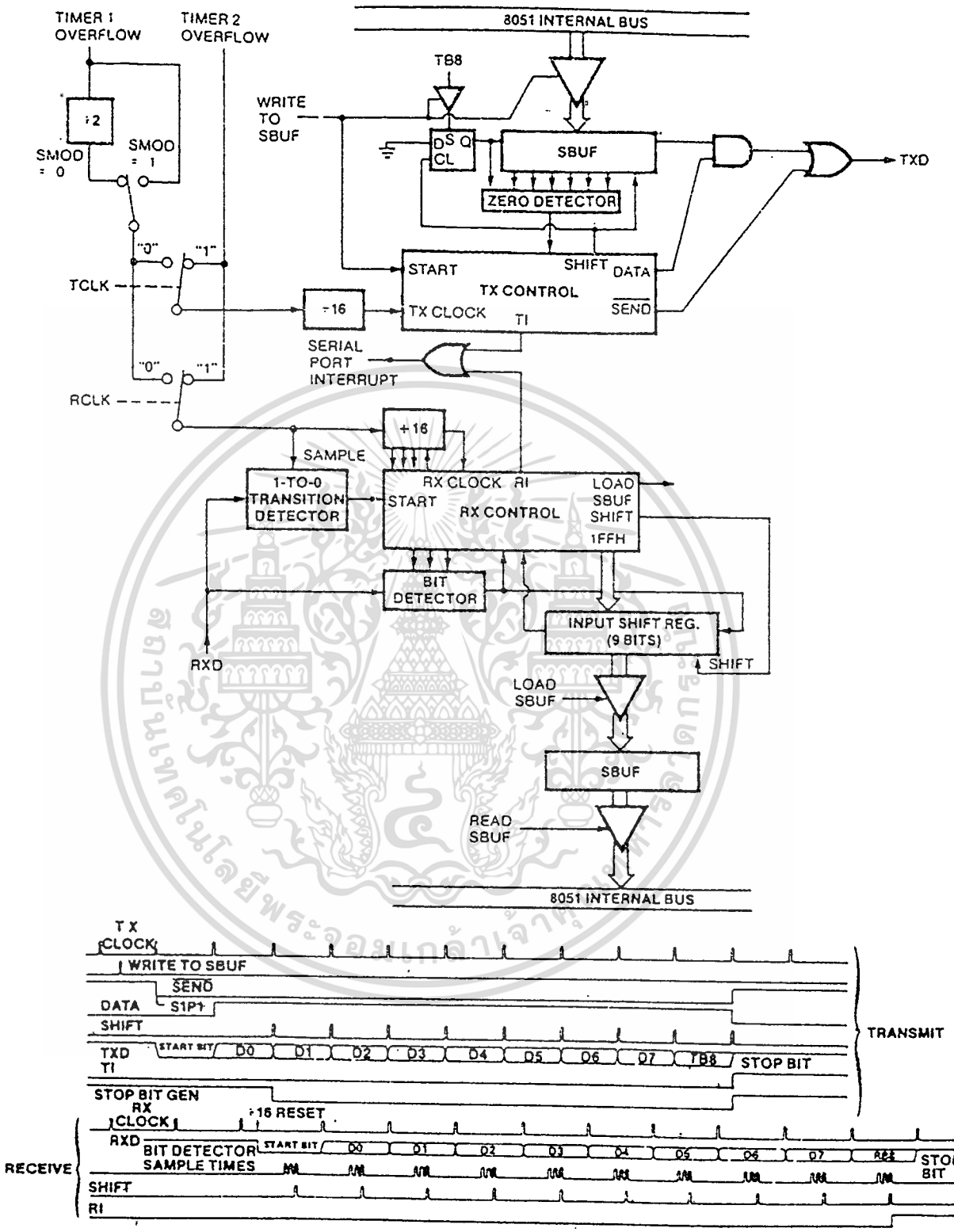
- บิท RI = 0

- บิท SM2 = 0 หรือ บิทที่ 9 ที่รับเข้ามามีค่าเป็น 1

ถ้ามีเงื่อนไขใดไม่ตรงตามนี้ข้อมูลกลุ่มที่ถูกรับเข้ามาทั้งหมดจะหายไปโดยไม่สามารถเรียกคืนกลับมาได้ และบิท RI จะไม่ถูกเซ็ทแต่ถ้าตรงตามเงื่อนไขทั้งสองอย่างนี้ บิทที่ 9 ที่รับเข้ามาจะถูกนำไปไว้ในบิท RB8 ในรีจิสเตอร์ SCON ส่วน data ทั้ง 8 จะถูกโหลดไปไว้ในรีจิสเตอร์ SBUF เมื่อเวลาในการรับผ่านไปอีก 1 บิทไม่ว่าเงื่อนไขทั้งสองจะเป็นอย่างไรระบบการรับข้อมูลก็จะกลับไปตรวจกาการเปลี่ยนสถานะจาก 0 เป็น 1 ที่ขา RXD ต่อ โดยถือว่ารับข้อมูลครบ 1 ไบต์ (1 ไบต์นี้จะนับรวม start bit , Programable bit , stop bit รวมเป็น 11 บิท) เรียบร้อยแล้ว สังเกตุว่าค่าของ stop bit ที่ถูกรับเข้ามาจะไม่เกี่ยวข้องกับรีจิสเตอร์ SBUF บิท RB8 หรือ บิท RI เลย



รูปที่ 32 บล็อคแสดงการทำงานในSerial Port Mode 2



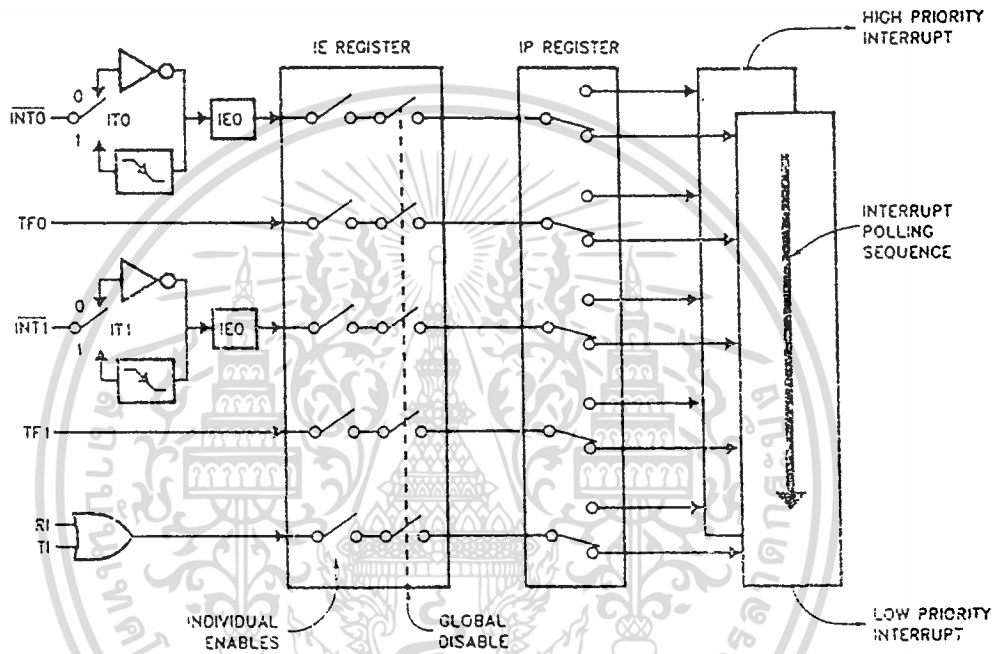
รูปที่ 33บล็อคแสดงการทำงานของ Serial Port Mode 3 แบบใช้ TCLK, RCLK, andTimer 2 ซึ่งมีเฉพาะใน 8052/8032 เท่านั้น

บทที่ 4

การอินเทอร์รัพท์และ อาร์.ที.ซี. (INTERRUPTS & RTC)

การอินเทอร์รัพท์

MCS-51 สามารถรับสัญญาณอินเทอร์รัพท์ที่เกิดขึ้นได้อย่างน้อย 5 ชนิด เช่น ในเบอร์ 8052 สามารถรับได้ 6 ชนิด ดังแสดงในรูปที่ 34



รูปที่ 34 แสดงจุดของการ Interrupt ใน MCS -51

สัญญาณอินเทอร์รัพท์

สัญญาณอินเทอร์รัพท์แต่ละชนิดที่ MCS-51 สามารถรับได้มีราบละเอียดดังต่อไปนี้

- External Interrupts (INT0 และ INT1) เป็นอินเทอร์รัพท์ที่เกิดขึ้นจากภายนอกโดยผู้เขียนโปรแกรมสามารถเลือกให้ MCS - 51 ตรวจสอบสัญญาณอินเทอร์รัพท์ที่เกิดขึ้นได้ 2 แบบด้วยกัน คือ แบบตรวจสอบระดับสัญญาณ (Level - Artivated) หรือ แบบตรวจสอบการเปลี่ยนสถานะของสัญญาณ (Transition - Artivated) อย่างใดอย่างหนึ่งขึ้นอยู่กับการควบคุมของบิท IT0 และ IT1 ในรีจิสเตอร์ TCON

เมื่อ MCS - 51 ตรวจสอบสัญญาณอินเทอร์รัพท์อย่างใดอย่างหนึ่งของทั้งสองนี้จะมีผลทำให้บิท IE0 หรือ IE1 ของรีจิสเตอร์ TCON ถูกเซ็ท ซึ่งบิททั้งสองจะทำหน้าที่เป็นตัวบอกสถานะ (FLAG) ให้ซีพียู ของMCS-51ทราบว่าขณะนี้ได้เกิดอินเทอร์รัพท์จากภายนอกขึ้น จากนั้นบิทที่ถูกเซ็ทเมื่อเกิดอินเทอร์รัพท์ (บิท IE0 หรือ IE1) นี้จะถูกเคลียร์โดยฮาร์ดแวร์

เมื่อซีพียูถูกย้ายไปทำงานที่โปรแกรมส่วนบริการอินเทอร์รัพท์ (interrup service routine) ก็ต่อเมื่ออินเทอร์รัพท์ ที่เกิดขึ้นนั้นเป็นชนิดที่ตรวจสอบได้ จากการเปลี่ยน สถานะของสัญญาณ (transition - activated) แต่ถ้าอินเทอร์รัพท์ที่เกิดขึ้นใช้การตรวจสอบจากระดับของสัญญาณ (level - activated) แล้วเมื่อซีพียูย้ายไปทำงานที่โปรแกรมบริการอินเทอร์รัพท์จะไม่มีการเคลียร์บิท IE0 หรือ IE1ให้ ดังนั้นวงจรภายนอกเดิมเองมิฉะนั้นโปรแกรมหลักที่ทำงานอยู่จะถูกอินเทอร์รัพท์ไปเรื่อย ๆจนกระทั่งสัญญาณอินเทอร์รัพท์กลับมีค่าเป็น0 อีกครั้ง

- Timer 0 และ Timer 1 Interrupts ถูกทำให้เกิดอินเทอร์รัพท์ได้โดยที่บิท TF0 และ TF1 ซึ่งถูกเซ็ทโดยการเปลี่ยนค่าจากการเป็น 1 ทั้งหมดมาเป็น 0 ทั้งหมดใน Timer/Counter Register (เกิด Roll Over) ของแต่ละตัว (ยกเว้น Timer 0 ในโหมด 3) เมื่อมีอินเทอร์รัพท์จาก Timer เกิดขึ้น บิท TF0 และ TF1 จะถูกเคลียร์โดยฮาร์ดแวร์เอง เมื่อซีพียูย้ายไปทำงานในส่วนโปรแกรมบริการอินเทอร์รัพท์

- Serial Port Interrupt สามารถทำให้เกิดอินเทอร์รัพท์ได้ โดยสัญญาณที่จะทำให้เกิดอินเทอร์รัพท์ได้มาจากบิท T1หรือ R1ที่นำมาผ่านเกท OR และบิทที่บอกการอินเทอร์รัพท์ทั้งสองนี้จะไม่ถูกเคลียร์โดยฮาร์แวร์เมื่อซีพียูไปทำงานใน โปรแกรมบริการอินเทอร์รัพท์ เพราะการเกิดอินเทอร์รัพท์ของ Serial Port อาจจะเกิดจากบิทR1หรือT1ก็ได้ ดังนั้นโปรแกรมในส่วนบริการอินเทอร์รัพท์จะต้องเป็นผู้ตรวจสอบเองว่า สัญญาณอินเทอร์รัพท์ที่เกิดขึ้นได้มาจากบิท R1 หรือ T1 และบิททั้งสองจะถูกเคลียร์ได้โดยซอฟท์แวร์เท่านั้น

- Timer 2 Interrupt ใน 8052 จะมีรีจิสเตอร์สำหรับ Timer 2 เพิ่มขึ้นมาอีก 1 ตัว ซึ่ง Timer 2 ที่มีเพิ่มมานี้ สามารถใช้อินเทอร์รัพท์ได้เหมือนเช่นใน Timer 0 และ Timer 1 แต่การเกิดอินเทอร์รัพท์ของ Timer 2 จะแตกต่างออกไปจาก Timer 0 และ Timer 1 ดังนี้ สัญญาณอินเทอร์รัพท์ของ Timer 2 เกิดขึ้นโดยการนำบิท TF2 และ EXF2 มาผ่านเกณฑ์OR โดยบิท TF2 และ EXF2 ทั้งสองจะไม่ถูกเคลียร์โดยฮาร์ดแวร์เมื่อซีพียูไปทำงานในส่วนโปรแกรมบริการอินเทอร์รัพท์ เหมือนในการเกิดอิเทอร์รัพท์ของ Serial Port ดังนั้นในโปรแกรมบริการอินเทอร์รัพท์จะต้องค้นหาเองว่าบิท TF2 หรือ EXF2 ตัวใดที่เป็นสาเหตุทำให้เกิดการอินเทอร์รัพท์ และบิทจะต้องได้รับการเคลียร์โดยซอฟท์แวร์เองด้วย

สัญญาณอินเทอร์รัพท์ทั้งหมดที่เกิดขึ้น ไม่ว่าจะเกิดาจากภายในหรือภายนอก จะมีผลไปเซ็ทบิทต่าง ๆ ดังได้กล่าวมาแล้ว ทำให้มีการอินเทอร์รัพท์เกิดขึ้น แต่เนื่องจากบิทที่ควบคุมการเกิดอินเทอร์รัพท์แต่ละชนิดดังได้กล่าวไปแล้ว สามารถควบคุมให้ถูกเซ็ทหรือเคลียร์ได้ จากสัญญาณอินเทอร์รัพท์ จะมีผลให้เกิดอินเทอร์รัพท์ขึ้นเช่นเดียวกัน โดยจะมีผลเหมือนกับว่ามันถูกเซ็ทหรือถูกเคลียร์โดยฮาร์ดแวร์ที่ต้อง การอินเทอร์รัพท์ซีพียูนั่น คือ อินเทอร์รัพท์ทั้งหมดที่กล่าวมา สามารถถูกยกเลิกได้ด้วยซอฟท์แวร์

สัญญาณอินเทอร์รัพท์แต่ละชนิดที่กล่าวไปแล้ว สามารถถูกควบคุมให้สามารถอินเทอร์รัพท์ MCS-51 ได้หรือไม่ ทั้งนี้โดยการควบคุมจากบิทต่าง ๆ ในรีจิสเตอร์ IE ดังแสดงในรูป35

IE: INTERRUPT ENABLE REGISTER, สามารถอ้างถึงแบบบิตได้

ชื่อบิต: IE ตำแหน่ง: A8h ค่าบิตเริ่มต้น: 0X00 0000

EA	-	ET2	ES	ET1	EX1	ET0	EX0

ชื่อบิต	ตำแหน่ง	ความหมาย
EA	IE.7	อีนาเบิล/อิสเอเบิลการเกิดอินเตอร์รัปต์โดยรวม
-	IE.6	-
ET2	IE.5	อีนาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ Timer2
ES	IE.4	อีนาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์พอร์ตอนุกรม
ET1	IE.3	อีนาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ Timer1
EX1	IE.2	อีนาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ INT1
ET0	IE.1	อีนาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ Timer0
EX0	IE.0	อีนาเบิล/ดิสเอเบิลการเกิดอินเตอร์รัปต์ INTO

รูปที่ 35 แสดงบิทต่าง ๆ ของรีจิสเตอร์ IE

จะเห็นได้ว่าบิท EA ในรีจิสเตอร์ IE สามารถควบคุมสัญญาณการเกิดอินเทอร์รัพท์ได้ทั้งหมดหากบิทนี้มีค่าเป็น0สัญญาณอินเทอร์รัพท์ทุกชนิดจะไม่สามารถอินเทอร์รัพท์ MCS-51 ได้ หากแต่บิทนี้มีค่าเป็น สัญญาณอินเทอร์รัพท์แต่ละชนิดจะมีอิสระในการควบคุมให้สามารถเกิดอินเทอร์รัพท์หรือไม่ก็ได้ (ควบคุมาจากบิท IE.0 - IE.5)

ตำแหน่งบิท IE5 , IE6 ไม่ถูกใช้ใน 8051 เพราะถูกสงวนไว้ใช้ใน MCS-51 เบอร์อื่น ๆ ที่สามารถจัดการกับสัญญาณอินเทอร์รัพท์ได้เพิ่มขึ้น ดังนั้นซอฟท์แวร์ของผู้ใช้ไม่ควรจะมีคำสั่งเขียนค่า I ลงไปในบิทตำแหน่งเหล่านี้ เพื่อให้เกิดโปรแกรมยังคงสามารถใช้กับไอซีเบอร์ใหม่ ๆ ในตระกูลนี้ได้นั่นเอง

การจัดลำดับความสำคัญ ของการอินเทอร์รัพท์

โครงสร้างการจัดลำดับความสำคัญ (Priority Lever Structure) ของสัญญาณอินเทอร์รัพท์แต่ละชนิด สามารถถูกเลือกระดับความสำคัญได้ 2 ระดับ โดยการเซ็ทหรือเคลียร์บิทในรีจิสเตอร์ IP ดังในรูปที่ 36

IP: INTERRUPT PRIORITY REGISTER สามารถอ้างถึงแบบบิตได้

ชื่อบิต: IP ตำแหน่ง: B8h ค่าบิตเริ่มต้น : 0000 0000

-	-	PT2	PS	PT1	PX	PT0	PX0

ชื่อบิต	ตำแหน่ง	ความหมาย
	IP.7	
	IP.6	
PT2	IP.5	ระดับความสำคัญของ Timer2
PS	IP.4	ระดับความสำคัญของพอร์ตอนุกรม
PT1	IP.3	ระดับความสำคัญของ Timer1
PX1	IP.2	ระดับความสำคัญของ INT1
PT0	IP.1	ระดับความสำคัญของ Timer0
PX0	IP.2	ระดับความสำคัญของ INTO

รูปที่ 36 ตารางแสดงจัดลำดับความสำคัญในการ Interrupt ของ Register IP

ระดับความสำคัญของการเกิดอินเทอร์รัพท์มีได้ 2 ระดับ ได้แก่

low Priority Interrupt : สัญญาณอินเทอร์รัพท์ชนิดนี้ สามารถถูกอินเทอร์รัพท์จาก High Priority Interrupt ได้ แต่ไม่สามารถถูกอินเทอร์รัพท์โดย Low Priority Interrupt ตัวอื่น ๆ ได้ High Priority Interrupt:สัญญาณอินเทอร์รัพท์ประเภทนี้ไม่สามารถถูกอินเทอร์รัพท์โดยสัญญาณอินเทอร์รัพท์ชนิดอื่น ๆ ได้เลย นั่นคือมีระดับความสำคัญสูงสุด

ถ้ามีการสัญญาณอินเทอร์รัพท์เกิดขึ้นพร้อมกัน 2 ชนิด โดยมีระดับความสำคัญในการอินเทอร์รัพท์ไม่เท่ากัน สัญญาณอินเทอร์รัพท์ที่มีระดับความสำคัญสูงกว่า (High Priority Interrupt) จะได้รับการบริการก่อนแต่ถ้ามีการขออินเทอร์รัพท์พร้อมกัน 2 ชนิด ซึ่งมีระดับความสำคัญเท่ากันลำดับในการบริการอินเทอร์รัพท์ภายใน MCS - 51 จะเป็นตัวกำหนดเองว่าอินเทอร์รัพท์ชนิดใดควรจะถูกบริการก่อน ดังนั้นภายในระดับความสำคัญของการเกิด

อินเทอร์รัพท์หนึ่ง ๆ ที่เกิดขึ้นจะมีระดับความสำคัญในการขออินเทอร์รัพท์ย่อยลงไปอีกระดับหนึ่ง Second Priority Structure) ดังรูปที่ 37

ระดับความสำคัญ	สัญญาณ	ความหมาย
1	IE0	อินเตอร์รัปต์ภายนอก 0
2	TF0	วงจรนับ/จับเวลา 0
3	IE1	อินเตอร์รัปต์ภายนอก 1
4	TF1	วงจรนับ/จับเวลา 1
5	RI หรือ TI	วงจรรับ/ส่งข้อมูลอนุกรม
6	TF2 หรือ EXF2	วงจรนับ/จับเวลา 2

รูปที่ 37 ลำดับความสำคัญของการ Interrupt

การจัดการกับอินเทอร์รัพท์ที่มี ระดับความสำคัญ เท่าเทียมกัน ที่เกิดขึ้นพร้อมกัน (Priority Within Level Struchure) ถูกใช้เพียงเพื่อแก้ปัญหาการเกิดสัญญาณอินเทอร์รัพท์พร้อมกันขึ้นมาเท่านั้น (Simultaneou Request Of The Same Priority Level) รีจิสเตอร์ IP มีบิทที่ไม่ถูกใช้งานอยู่จำนวนมาก คือ IP.7 และ IP.6 ดังนั้นซอฟท์แวร์ของผู้ใช้ไม่ควรมีการเขียนค่า 1 ไปที่ตำแหน่งบิทเหล่านั้น เพราะมันอาจถูกนำไปใช้ใน MCS-51 เบอร์ใหม่ ๆ ในอนาคตต่อไป

จัดการกับสัญญาณอินเทอร์รัพท์ ของ MCS-51

สัญญาณอินเทอร์รัพท์ของอินเทอร์รัพท์แต่ละชนิดจะถูกตรวจสอบ (Semple) ที่ทุก ๆ S5P2 ของทุก machine cycle ค่าที่ตรวจสอบจะถูกรับเข้ามาระหว่าง machine cycle ที่ตามมา โดยใน 8052 Timer 2 Interrup cycle จะต่างออกไปดังจะอธิบายใน หัวข้อเวลาในการตอบสนอง(Response Time)

การตอบสนองต่อสัญญาณอินเทอร์รัพท์ ซึ่งจะเห็นว่าประกอบไปด้วย cycle ต่าง ๆ กันคือ cycle ตรวจจับอินเทอร์รัพท์ (จะตรวจสถานะของบิทที่ทำให้เกิดการอินเทอร์รัพท์ซีพียูทุก ๆ S5P2 ของแต่ละ machine cycle) แต่ละ cycleการคนหาชนิดของอินเทอร์รัพท์ (POLLING CYCLE) โดยทำการตรวจสอบว่าเป็นอินเทอร์รัพท์ชนิดใด และ cycle การสร้างคำสั่ง LCALL ไปยังโปรแกรมส่วนบริการอินเทอร์รัพท์ที่เหมาะสม (ใช้ 2 cycle) และ cycle ของการทำคำสั่งที่อยู่ในโปรแกรมบริการอินเทอร์รัพท์โดยมีรายละเอียดดังนี้

ถ้าบิทที่ก่อให้เกิดอินเทอร์รัพท์ตัวใดตัวหนึ่งอยู่ในสถานะการเซ็ทที่ S5P2 ของ cycle ตรวจจับอินเทอร์รัพท์ซึ่งเป็น cycle ก่อนหน้า cycle การหาชนิดของอินเทอร์รัพท์ (POLLING CYCLE) ซึ่งจะทำการค้นหาว่าอินเทอร์รัพท์ชนิดใดเป็นผู้ขอเข้ามาใน cycleถัดไปก็จะทำคำสั่ง LCALL ไปยังโปรแกรมบริการอินเทอร์รัพท์ที่เหมาะสม cycle การทำคำสั่ง LCALL โดยฮาร์ดแวร์จะถูกกระทำสำเร็จก็ต่อเมื่อไม่ถูกป้อนกันโดยสถาวะดังต่อไปนี้

1. ซีพียูกำลังทำคำสั่งในโปรแกรม บริการ อินเทอร์รัพท์ของอินเทอร์รัพท์ซึ่งมีความสำคัญเทียบเท่าหรือสูงกว่าอยู่ในขณะนั้น

2. cycle ที่ครวจหาชนิดของอินเทอร์รัพท์ไม่ใช่ cycle สุดท้ายของคำสั่งที่ซีพียูกำลังปฏิบัติงานอยู่ นั่น คือซีพียูกำลังทำงานของคำสั่งใด ๆ ยังไม่เสร็จสิ้นขณะตรวจหาชนิดของอินเทอร์รัพท์พบ

3. คำสั่งที่กำลังปฏิบัติอยู่ในขณะนั้นเป็น RETI หรือคำสั่งใด ๆ ที่มีการเขียนข้อมูลไปยังรีจิสเตอร์ IE หรือ IP

สภาวะทั้งสามนี้จะป้องกันมิให้ซีพียูทำคำสั่ง LCALL ไปยังโปรแกรมบริการย่อย ๆ อินเทอร์รัพท์ขณะตรวจพบการขออินเทอร์รัพท์

สภาวะที่ 1 มีไว้เพื่อให้อินเทอร์รัพท์ที่มีความสำคัญสูงกว่าได้รับการบริการก่อน โดยอินเทอร์รัพท์ที่มีความสำคัญต่ำกว่า ไม่สามารถขัดจังหวะได้

สภาวะที่ 2 ทำให้แน่ใจว่าคำสั่งที่กำลังทำอยู่ในขณะนั้นจะถูกทำให้เสร็จก่อนการย้ายไปทำคำสั่งใด ๆ ที่โปรแกรมบริการอินเทอร์รัพท์ที่เกี่ยวข้อง

สภาวะที่ 3ทำให้แน่ใจว่า ถ้าคำสั่งที่ทำอยู่เป็น RET1 หรือการเข้าถึงข้อมูลในรีจิสเตอร์ IE หรือ IP แล้ว อย่างน้อยจะต้องมีคำสั่งอีก 1 คำสั่งถูกปฏิบัติก่อนอินเทอร์รัพท์ใด ๆ จะถูกบริการ

cycle การตรวจหาชนิดของอินเทอร์รัพท์ (POLLING CYCLE) จะถูกทำซ้ำในแต่ละ machine cycle และค่าที่รับเข้า (ชนิดของอินเทอร์รัพทที่ตรวจสอบได้) จะเป็นค่าที่ปรากฏเมื่อ S5P2 ของ machine cycle ก่อนหน้านี้สังเกตว่าถ้าบิทที่ทำให้อินเทอร์รัพท์ถูกเซ็ท (Active) แต่ไม่ถูกตอบสนองเพราะ 1 ใน 2 สภาวะที่กล่าวมาข้างต้น และปรากฏว่าไม่ active เมื่อสภาวะที่ป้องกันการทำคำสั่ง LCALL หมดไปแล้ว อินเทอร์รัพท์ที่เกิดจะไม่ได้รับการบริการ (ไม่มีการย้ายการทำงานไปยังโปรแกรมการอินเทอร์รัพท์) หรือกล่าวได้ว่าสถานะของอินเทอร์รัพท์แต่ละตัวซึ่งครั้งหนึ่งเคย active แต่ไม่ได้รับการตอบสนอง เพราะถูกป้องกันด้วยสภาวะดังกล่าวข้างต้น จะไม่ถูกเก็บค่าไว้ทั้งนี้เพราะในแต่ละ cycleของการตรวจหาชนิดของอินเทอร์รัพท์ ซึ่งเกิดขึ้นทุก machine cycle จะรับสถานะของอินเทอร์รัพท์ใหม่เข้ามาเสมอ

สังเกตว่าถ้าอินเทอร์รัพท์ที่มีความสำคัญสูงถูก active ก่อน S5P2 ของช่วง C3 ในรูปที่ 37 และไม่มีสภาวะที่ป้องกันการทำคำสั่ง LCALL มันก็จะได้รับการบริการในระหว่าง C5

และ C6 โดยปราศจากการบริการของอินเทอร์รัพท์ที่มีความสำคัญน้อยกว่า ถึงแม้จะได้รับการตรวจสอบค่าแล้วในช่วง C1 ดังนั้นซีพียูจะรับรู้การอินเทอร์รัพท์โดยการทำคำสั่ง LCALL ไปที่โปรแกรมบริการอินเทอร์รัพท์ที่เหมาะสม ในบางกรณีมันจะเคลียร์บิทก่อนที่ก่อให้เกิดอินเทอร์รัพท์ด้วยเลย แต่ในบางกรณีมันจะไม่เคลียร์ให้ ดังนั้นผู้ใช้ต้องบรรจุคำสั่งเคลียร์บิทเหล่านี้เองในซอฟท์แวร์ ทั้งนี้ขึ้นกับชนิดของอินเทอร์รัพท์ที่เกิดขึ้นในแต่เหตุการณ์ดังได้กล่าวไปแล้วในตอนต้น

- external interrupt flag (IE0 และ IF1) จะถูกเคลียร์ต่อเมื่อ MCS-51 ถูกเลือกให้ตรวจสอบสัญญาอินเทอร์รัพท์ แบบจากการเปลี่ยนระดับสัญญาณ(transition activated)คำสั่ง LCALL ที่กระทำโดยฮาร์ดแวร์จะPUSH ข้อมูลของProgram Counter ไปไว้ที่stack (ไม่มีการ save ค่าของรีจิสเตอร์ PSW)และโหลดค่าของ PC ใหม่ด้วยแอดเดรสที่ตรงกับ ตำแหน่งของโปรแกรมบริการอินเทอร์รัพท์ที่เกี่ยวข้องกับสัญญาณอินเทอร์รัพท์นั้น ดังแสดงในรูปที่ 38

แหล่งกำเนิดสัญญาณอินเตอร์รรัปต์	ตำแหน่งแอดเดรส (Hex)
IE0 อินเตอร์รัปต์ภายนอก	0003
TF0 วงจรนับ/จับเวลา 0	000B
IE1 อินเตอร์รัปต์ภายนอก 1	0013
TF1 วงจรนับ/จับเวลา 1	001B
R1 หรือ TI วงจรรับ/ส่งข้อมูลอนุกรม	0023

รูปที่ 38 ตำแหน่งแอดเดรส เมื่อเกิดอินเตอร์รรัปต์

คำสั่งในโปรแกรมบริการอินเทอร์รัพท์ จะถูกปฏิบัติไปเรื่อย ๆ จนกระทั่งพบคำสั่ง RETI ซึ่งเป็นตัวบอกซีพียูว่าการบริการอินเทอร์รัพท์สิ้นสุดลงแล้ว จากนั้นซีพียูจะไป POP เอาค่า 2 ไบต์สูงสุดจาก stack และโหลดให้กับ PC เพื่อให้สามารถกลับไปทำงานคำสั่งเดิมที่ปฏิบัติอยู่ก่อนได้รับอินเทอร์รัพท์สังเกตว่าคำสั่ง RET ธรรมดาจะสามารถ POP เอาค่า 2 ไบต์สูงสุดจาก STACK และโหลดให้กับ PC ได้เช่นเดียวกับคำสั่ง RET แต่หากใช้ RET แล้ว Interrupt control ststem จะยังคิดว่าอินเทอร์รัพท์ยังคงถูกกระทำอยู่

การอินเทอร์รัพท์จากภายนอก (EXTERNAL INTERRUPT)

แหล่งกำเนิดอินเทอร์รัพท์จากภายนอกสามารถถูก โปรแกรมให้มีการตรวจสอบแบบ level - activaled หรือ transition - activated โดยการเซ็ทหรือเคลียร์บิท IT1 , IT2 ในรีจิสเตอร์ TCON โดยถ้าบิท (ITx(IT0 หรือ IT1) มีค่าเป็น 0 external interrupt จะถูกตรวจ

สอบจากระดับของสัญญาณ (level-activated) โดยสถานะLOW ที่ขาIntx (INT0 หรือ INT1) จะทำให้เกิดอินเทอร์รัพท์ซีพียูถ้าบิทItx =1,external interrupt จะถูกตรวจสอบจากการเปลี่ยนระดับสัญญาณ 1ransition - activated (edge-trigged) ในโหมดนี้ถ้าการตรวจสอบที่ขา INTx พบว่ามีสถานะเป็น HIGH ใน 1 cycle และเป็น LOW ใน cycle ถ้าไปinterrupt request flag หรือบิท Iex (บิท IE0 หรือ IE1) ในรีจิสเตอร์ TCON จะถูกเซ็ท บิท Iexที่ถูกเซ็ทนี้จะทำให้เกิดการอินเทอร์รัพท์ซีพียูต่อไป

เพราะว่า external interrupt ถูกตรวจสอบ1ครั้งในแต่ละ machine cycle และสัญญาณที่เป็น HIGH หรือ LOW ควรจะคงค่าไว้อย่างน้อย 12 period ของความถี่ออสซิลเลเตอร์ เพื่อให้แน่ใจว่า สัญญาณอินเทอร์รัพท์ที่เกิดขึ้นจะถูกตรวจพบแน่นอน เช่น ในกรณีมีการเกิด external interrupt ถูกตรวจสอบจากการเปลี่ยนระดับข้อมูลสัญญาณ (transition - activated) แหล่งกำเนิดอินเทอร์รัพท์ภายนอกจะต้องคงค่าเป็น LOWอย่างน้อย 1 cycle เพื่อให้มั่นใจว่าการเปลี่ยนแปลงถูกตรวจพบอย่างแน่นอน ในกรณีนี้บิท Iex ซึ่งเป็น external interrupt flag จะถูกเคลียร์โดยอัตโนมัติ เมื่อซีพียูปฏิบัติคำสั่ง LCALL ไปที่ external interrupt routine

ถ้า external interrupt ถูกตรวจสอบจากระดับของสัญญาณ (level - activated) วงจรที่กำเนิดอินเทอร์รัพท์ภายนอก ต้องรักษาสถานะการขออินเทอร์รัพท์ให้กลับ มีค่าเหมือนเดิม (Deactive) ก่อนที่โปรแกรมบริการอินเทอร์รัพท์จะถูกกระทำเสร็จ มิฉะนั้นแล้ว โปรแกรมบริการอินเทอร์รัพท์เดียวกันนี้จะถูกทำงานซ้ำอีก

เวลาในการตอบสนอง อินเทอร์รัพท์ (RESPONES TIME)

ระดับของ INT0 และ INT1 จะถูก invert และถูกคงไว้ในบิทIE0 และ IE1 ที่ S5P2 ของแต่ละ machine cycle เช่นเดียวกับ Timer2 Overflow flagและEEF2รวมทั้ง Seriel Port flag (บิท RI , TI) จริง ๆ แล้วสถานะของบิททำหน้าที่อินเทอร์รัพท์ซีพียูจะยังไม่ถูกรับเข้ามาโดยวงจรภายใน จนกระทั้งถึง machine cycle ถัดไป

FLAG ของ TIMER 1 (TF0 และ TF1) จะถูกเซ็ทที่ S5P2 ของ cycle ที่ S5P2 ของ cycle ที่ TIMER เกิด overflowแล้วค่า FLAG นี้จะถูกรับโดยวงจรภายในใน cycle ถัดไป แต่ FLAG ของ TIMER 2 จะถูกเซ็ทที่S5P2 และถูกรับค่าในcycle เดียวกัน cycle ที่เกิด overflow

ถ้ามีการขออินเทอร์รัพท์และเงื่อนไข 3 ข้อที่กล่าวมาถูกต้องสำหรับ MCS-52 ที่จะรับรู้ว่ามีการขออินเทอร์รัพท์ได้คำสั่ง LCALL ที่เกิดขึ้นเองจาก MCS-51 จะทำให้ซีพียูภายใน MCS-51 ถูกย้ายการทำงานไปที่โปรแกรมการอินเทอร์รัพท์ที่ถูกเรียก และคำสั่งถัดไปที่จะปฏิบัติ จะเป็นคำสั่งแรกในโปรแกรมบริการอินเทอร์รัพท์เนื่องจากตัวคำสั่ง LCALL เองจะใช้เวลา 2 machine cycle ดังนั้นการตอบสนองต่อสัญญาณอินเทอร์รัพท์ที่เกิดขึ้น จะต้อง

ใช้เวลาอย่างน้อย 3machine cycleเต็ม ๆ ซึ่งเป็นเวลาระหว่างการเกิดสัญญาณอินเทอร์รัพท์กับการเริ่มต้นทำงานคำสั่งแรกในโปรแกรมบริการอินเทอร์รัพท์ ดังอธิบายได้ดังรูปที่ 38

ช่วงเวลาในการตอบสนองที่ยาวกว่านี้ อาจเกิดขึ้นได้หากการเรียกอินเทอร์รัพท์ถูกขัดไว้ด้วยเงื่อนไข 1 ใน 3 ข้อที่กล่าวไว้ในข้างต้น ถ้าอินเทอร์รัพท์ที่มีลำดับความสำคัญเท่ากันหรือสูงกว่ากำลังถูกกระทำอยู่ เวลา ในการรอคอยที่เพิ่มขึ้นจะขึ้นอยู่กับลักษณะของโปรแกรมบริการอินเทอร์รัพท์ที่กำลังปฏิบัติอยู่ถ้าคำสั่งที่กำลังกระทำอยู่ไม่อยู่ใน cycle สุดท้ายของคำสั่งนั้น เวลาในการรอคอยที่มีเพิ่มขึ้นจะไม่สามารถมากกว่า 3 cycle ได้เพราะคำสั่งที่ยาวที่สุดคือ MUL , DIV ใช้เวลาเพียง 4 cycle เท่านั้นและหากคำสั่งที่กำลังกระทำเป็น REI หรือการเขียนข้อมูลไปยังรีจิสเตอร์ IE หรือ IP เวลารอที่เพิ่มขึ้นนั้น ก็จะไม่มากกว่า 5 cycle (อย่างมากที่สุดอีก 1 cycle เพื่อให้กระทำคำสั่งที่ทำอยู่เสร็จกันอีก 4 cycle ที่ใช้ทำคำสั่งถัดไป ในกรณีที่คำสั่งถัดไปนั้นเป็น MUL หรือ DIV

ดังนั้น ในระบบที่มีการใช้อินเทอร์รัพท์เพียงชนิดเดียว เวลาที่ใช้สำหรับการตอบสนองอินเทอร์รัพท์จะมากกว่า 3 cycle และน้อยกว่า 9 cycle เสมอ

ฐานเวลาที่เป็นจริง (REAL TIME CLOCK)

ในระบบที่ไมโครคอนโทรลเลอร์เป็นตัวควบคุม บางครั้งอาจจำเป็นต้องมีการใช้เวลาเป็นตัวกำหนดการทำงานเช่น การกำหนดการทำงานให้เครื่องจักรเปิดหนือปิดในเวลาที่กำหนด การบันทึกเวลาการใช้งานเครื่องจักรในแต่ละวัน ดังนั้นหาในระบบที่ใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมสามารถทราบเวลาในขณะใด ๆ ได้ จะเป็นการสะดวกในการเขียนโปรแกรมที่เกี่ยวข้องกับเวลา และสะดวกในการออกแบบระบบ และในบทนี้เราจะกล่าวถึงชิพที่ทำหน้าที่เป็นตัวสร้างฐานเวลาให้แก่ไมโครคอนโทรลเลอร์โดยภายในชิพประเภทนี้จะมีเวลาที่เดินอย่างเที่ยงตรงอยู่ตลอดเวลา และสามารถส่งข้อมูลซึ่งเป็นเวลาขณะใดๆให้แก่ระบบได้

เราได้ศึกษามาแล้วว่าใน MCS-51 มีรีจิสเตอร์ที่ทำหน้าที่เป็นได้ทั้ง TIMER หรือ COUNTER ซึ่งผู้ใช้สามารถเขียนโปรแกรมให้ทำงานเป็นตัวสร้างฐานเวลาแก่ซีพียูได้ แต่เนื่องจากไม่สะดวกในการใช้งานจริงที่ต้องการทราบเวลาปัจจุบัน เนื่องจากต้องมีการคำนวณคาบเวลาให้ถูกต้อง และต้องมีการตั้งค่าใหม่ทุกครั้งที่เริ่มทำงาน หรือหากการทำงานเกิดผิดพลาดเวลาก็จะผิดตามไปด้วย เนื่องจากเราใช้รีจิสเตอร์ภายใน MCS-51 เป็นตัวกำหนดฐานเวลาดังนั้นจึงทำให้เกิดความจำเป็นที่จะต้องมีวงจรซึ่งทำหน้าที่สร้างฐานเวลาจริง (ต่อไปจะเรียกย่อว่า RTC ซึ่งย่อมาจาก REAL TIMER CLOCK) ให้แก่ระบบโดยมีการทำงานเป็นอิสระจากไมโครคอนโทรลเลอร์ และจำเป็นต้องทำงานอยู่ตลอดเวลาแม้ขณะไม่มีพลังงานจ่ายให้ระบบ นั้งนี้โดยการใช้แบตเตอรี่แบคอัพ RTC ที่ทำหน้าที่สร้างฐานเวลาจริงให้แก่ระบบในปัจจุบันนี้ได้พัฒนาขึ้นมาก จนมีขนาดเล็กลงเป็นไอซี ซึ่งใช้พลังงานน้อยมาก ทำให้การใช้แบตเตอรี่

สำรองข้อมูลจะสะดวกมากขึ้น ระบบควบคุมที่ใช้ RTC เป็นตัวกำหนดเวลารวมนั้น นอกจากจะทำให้มีความเที่ยงตรงในการทำงานมากขึ้นแล้ว ประโยชน์อย่างหนึ่งที่ได้นั่นคือทำให้มี TIMER/COUNTER ไว้เหลือใช้งานอย่างอื่นได้อีกด้วย

RTC ที่ทำหน้าที่เป็นตัวสร้างฐานเวลาให้แก่ระบบที่นิยมใช้กัน ในปัจจุบันนี้มีอยู่ด้วยกันหลายเบอร์ แต่เบอร์ที่ใช้งานได้ง่ายและสะดวกที่สุด คือ RTC ของบริษัท DALLAS SEMICONDUCTOR เบอร์ DS1202 "SERIALTIMEKEEPER"ซึ่งเราจะได้กล่าวต่อไปในบทนี้RTCจริงๆแล้วมีอยู่ด้วยกันหลายชนิดบางชนิด สามารถอินเทอร์รัพท์ไมโครคอนโทรลเลอร์ภายในคาบเวลาที่กำหนดได้ แต่เบอร์ที่เราจะกล่าวถึงนี้ สามารถให้ข้อมูลเกี่ยวกับเวลาแก่ระบบเท่านั้นการเลือกใช้งาน RTC ประเภทใดขึ้นอยู่กับผู้ออกแบบว่าต้องการความสามารถมากน้อยขนาดไหน

RTC เบอร์ DS1202 ที่เราจะกล่าวถึงนี้ มีความเที่ยงตรงในการติดต่อระหว่างตัวไอซีเองกับไมโครคอนโทรลเลอร์เพียง 3 เส้นเท่านั้น ทั้งนี้เนื่องจาก RTC เบอร์นี้ใช้การติดต่อรับส่งข้อมูลในแบบอนุกรม คุณสมบัติคร่าว ๆ ของ RTC เบอร์นี้มีดังนี้

- ทำหน้าที่นับวินาที นาที ชั่วโมง วันที่ของเดือน เดือน ปี รวมทั้งคำนวณปี อธิกสุรทินให้เองโดยอัตโนมัติ
- มีหน่วยความจำขนาด 24*8 สำหรับเก็บข้อมูลทั่ว ๆ ไป ส่วนใหญ่ไว้เก็บข้อมูลที่ต้องการแบคอัพในขณะที่ไม่มีพลังงานจ่ายให้แก่ระบบ เช่น พาสเวริด์ที่เปลี่ยนค่าได้ , เวลาที่ต้องการให้เครื่องจักรทำงานเริ่มทำงาน ซึ่งทำให้ไม่จำเป็นต้องแบคอัพหน่ายความจำทั้งระบบนั่นเอง
- ใช้การติดต่อแบบอนุกรม จึงใช้จำนวนสายในการเชื่อมต่อกับระบบเพียง 3 เส้นเท่านั้น
- ใช้ไฟเลี้ยงเพียง 2.5-5.5 โวลท์ และใช้กระแสเพียง 300nA เท่านั้น ที่ระดับแรงดัน 2.0โวลท์
- การโอนย้ายข้อมูลสามารถกระทำได้ทั้งในแบบ Single Byte หรือ Multiple Byte (Burst Mode) ไม่ว่าจะเป็นการเขียนหรืออ่านข้อมูล
- มีให้เลือกทั้งแบบ 8 PIN หรือ 16 PIN SOIC เพื่อใช้สำหรับแผ่นวงจรชนิด SURFACE MOUNT
- TTL COMPATIBLE (Vcc = 5 โวลท์)
- ช่วงอุณหภูมิในการใช้งานกว้างมาก ระหว่าง -40 องศา - +88 องศาเซลเซียส

รายละเอียดของ อาร์ . ที ซี. (RTC)

RTC เบอร์ DS1202 " Serial Time Keeper Chip " มี Real Time Clock/Calendar และ Static RAM ขนาด 24 ไบท์ โดยใช้การเดินสายเพียง 3 เส้น ในการเชื่อมต่อกับระบบเพื่อรับส่งข้อมูลเกี่ยวกับเวลา ข้อมูลที่ RTC DS1202 สามารถส่งให้ระบบประกอบด้วย

- วินาที (Seconds)
- นาที (Minutes)
- ชั่วโมง (Hour)
- วันที่ (Date)
- วัน (Day)
- เดือน (Month)
- ปี (Year)

วันที่ในวันสุดท้ายของเดือน จะถูกปรับโดยอัตโนมัติสำหรับเดือนที่มีจำนวนวันน้อยกว่า 31 วัน และมีการคำนวณจำนวนวันของเดือนกุมภาพันธ์ในปีอธิกสุรทินให้เอง ข้อมูลที่ส่งให้แก่ไมโครคอนโทรลเลอร์สามารถเลือกรูปแบบได้ทั้งแบบ 24 ชั่วโมง (0.00 น. ถึง 23.59 น.) หรือแบบ 12 ชั่วโมง (0.00 น. ถึง 12.00 น. โดยมีข้อมูลเพิ่มเพื่อบอกให้ทราบว่าเป็นช่วงกลางวันหรือกลางคืน)

การเชื่อม RTC DS1202 เข้ากับระบบ มีความสะดวกมากเนื่องจากใช้จำนวนสายเพียง 3 เส้น เท่านั้น เพราะใช้การติดต่อแบบอนุกรมจะใช้ ชนิดที่เป็น SYNCHRONOUS SERIAL COMMUNICATION ขาที่ต้องใช้ในการรับส่งข้อมูลทั้งสามคือ

- RST (Reset)
- I/O (DATA LINE)
- SCLK (SERIAL CLOCK)

เนื่องจากใน RTC DS1202 มีเวลาที่เดินอยู่ตลอดเวลา รวมทั้งหน่วยความจำนวนหนึ่ง ดังนั้นในการติดต่อกับ RTC DS1202 ผู้ใช้สามารถเลือกได้ว่าต้องการข้อมูลจากนาฬิกาหรือจากหน่วยความจำในไอซี(CLOCK /RAM)โดยการรับส่งสามารถกระทำได้ทั้งแบบทีละ1ไบท์ (SINGLE BYTE) หรือสามารถรับส่งกันคราวละหลายไบท์ (BURST MODE)นอกจากนี้ RTC DS1202 ยังถูกออกแบบให้สามารถใช้พลังงานได้น้อยมาก เพื่อให้สะดวกในการแบคอัพโดยทำให้สิ้นเปลืองพลังงาน จากแบตเตอรี่น้อยที่สุดโดยไอซีตัวนี้สามารถเก็บรักษาข้อมูลในหน่วยความจำและเวลาได้ที่กำลังไฟฟ้าน้อยกว่า 1 ไมโครวัตต์

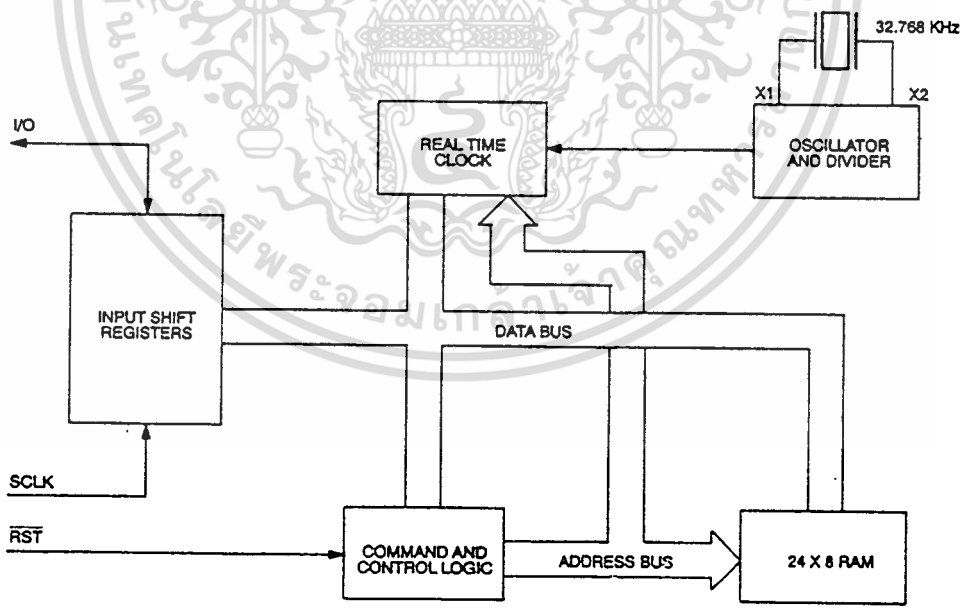
ในการรับส่งข้อมูลใด ๆ RST จะถูกทำให้มีสถานะเป็น 1 จากนั้นข้อมูลจำนวน 8 บิท จะถูกโหลดเข้าไปยัง SHIFT REGISTER ซึ่งข้อมูลขนาด 8 บิท จะประกอบด้วยคำสั่งในการควบคุม RTC และตำแหน่งของหน่วยความจำที่ต้องการติดต่อการรับข้อมูลแต่ละบิทจะกระทำที่ช่วงขอบขาขึ้นของสัญญาณ SCLK (SERIAL CLOCK)

ใน RTC DS1202 ประกอบด้วยหน่วยความจำขนาด 24 ไบท์ และรีจิสเตอร์ที่ทำหน้าที่เก็บเวลาของไอซีอีก 8 ตัว ดังแสดงในภาพที่ 4 โดยรีจิสเตอร์ทั้ง 8 นี้สามารถเข้าถึงได้เสมือนเป็นหน่วยความจำตำแหน่งหนึ่ง ๆ ดังนั้นต่อไปเราจะมองว่า RTC เบอร์นี้มีหน่วยความจำรวมทั้งสิ้น 32 ตำแหน่ง โดยในความเป็นจริง รีจิสเตอร์ทั้ง 8 กับหน่วยความจำขนาด 24 ไบท์จะมีค่าตำแหน่งซ้อนกัน

ข้อมูลขนาด 8 บิทแรก จะระบุตำแหน่งของความจำที่ต้องการติดต่อ (ทั้งหน่วยความจำทั่วไปและหน่วยความจำที่เป็นรีจิสเตอร์ซึ่งใช้เก็บเวลา) และบอกว่าเป็นการเขียนหรืออ่านข้อมูลจากหน่วยความจำตำแหน่งนั้น ๆ รวมทั้งระบุว่าให้การรับส่งข้อมูลเป็นแบบ SINGLE BYTE หรือ BURST MODE

หลังจากมีสัญญาณนาฬิกาเกิดขึ้น 8 ครั้ง(สัญญาณ SCLK)ในระหว่างการรับส่งข้อมูล 8 บิทแรกลงไปใน SHIFT REGISTER สัญญาณนาฬิกาต่อไปที่จะเกิดขึ้นจะเป็นการนำข้อมูลออกจาก RTC สำหรับการอ่าน หรือนำข้อมูลเข้า RTC สำหรับการเขียน จำนวนของสัญญาณนาฬิกาทั้งหมดที่เกิดขึ้นในการติดต่อครั้งหนึ่ง ๆ จึงเท่ากับ 8+8 ใน SINGLE BYTE MODE หรือ 8 บวกมากที่สุด 192 (8*24) สำหรับ BURST MODE

การทำงานของ RTC



ภาพที่ 39 โครงสร้างภายในของ RTC DS1202

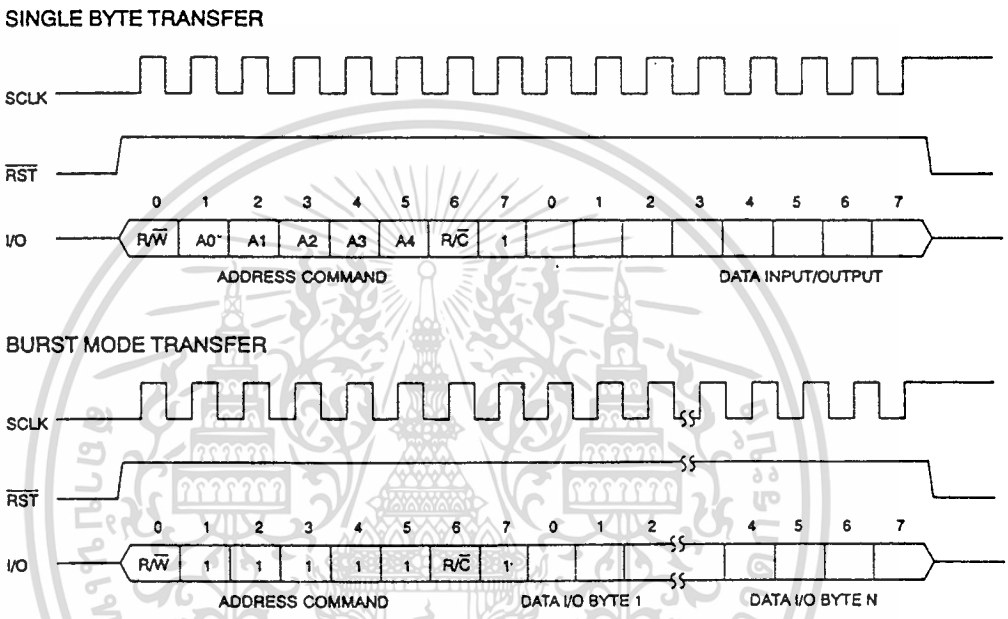
โครงสร้างภายในของ RTC DS1202 ประกอบไปด้วยส่วนที่สำคัญ ๆ ดังแสดงในรูปที่ 39 จะเห็นว่า RTC เบอร์นี้ประกอบไปด้วยส่วนที่สำคัญ ๆ ดังนี้คือ

- SHIFT REGISTER
- CONTROL LOGIC

- OSCILLATOR
- REAL TIME CLOCK และ
- RAM

โครงสร้างของ COMMAND BYTE ของ RTC

โครงสร้างของ COMMAND BYTE มีดังแสดงในภาพที่ 40



FUNCTION	BYTE N	SCLK n
CLOCK	8	72
RAM	24	200

ภาพที่ 40 แสดงโครงสร้างของ COMMAND BYTE

เนื่องจากในการรับส่งข้อมูลระหว่าง RTC DS1202 ในตอนเริ่มต้นไมโครคอน - โทรลเลอร์จะต้องส่งข้อมูลเพื่อกำหนดการทำงานให้แก่ RTC เสียก่อน ข้อมูลที่ RTC ได้รับในตอนเริ่มต้นจะมีขนาด 1 ไบท์ ซึ่งมีชื่อว่า COMMAND BYTE และเนื่องจากข้อมูลในไบท์นี้จะเป็นตัวกำหนดการทำงานของ RTC ดังนั้นแต่ละบิทในไบท์นี้จะมีความหมายแตกต่างกันไปดังนี้

- MSB (บิท 7) ต้องเป็น 1 เสมอ ถ้าเป็น 0 การทำงานต่อจากนี้จะถูกหยุดไว้หมด
- บิท 6 ถ้าเป็น 0 จะระบุว่าต้องการติดต่อกับ CLOCK/CALENDAR REGISTER ดังนั้นข้อมูลที่รับส่งกันจะเป็นเวลา หากบิทนี้มีค่าเป็น1 จะระบุว่าต้องการติดต่อกับหน่วยความจำ

- บิท 1 ถึง 5 เป็นตัวระบุตำแหน่งหน่วยความจำ (ทั้งรีจิสเตอร์และหน่วยความจำทั่วไป) ที่ต้องการเข้าถึง ไม่ว่าจะเป็นการอ่านหรือเขียนข้อมูล ซึ่งควบคุมด้วยบิท 0 ดังจะได้กล่าวต่อไป

- บิท 0 จะระบุว่าเป็นการอ่านหรือเขียนข้อมูล โดยถ้าเป็น 0 หมายถึง การเขียนข้อมูลลงไปยังไอซี หากเป็น1 หมายถึง การอ่านข้อมูลจากไอซีดังนั้นในการส่ง COMMAND BYTE ไปยัง RTC จะเริ่มต้นด้วยบิท 0 ก่อนเสมอ (LSB FIRST)

การรับส่งข้อมูลหลายไบท์ (BURST MODE)

BURST MODE หมายถึง การรับส่งข้อมูลครั้งละหลายไบท์ โดยสามารถกำหนดได้ว่าจะเป็นข้อมูลจาก CLOCK / CALENDAR หรือจาก RAM โดยหากเป็นการรับส่งข้อมูลกับ CLOCK /CALENDAR จะรับส่งกันครั้งละ 8 ไบท์ หากเป็นRAMจะรับส่งกันครั้งละ 24 ไบท์ (ดูรายละเอียดได้จากภาพที่ 40) คำสั่ง BURST MODE กำหนดโดย COMMAND BYTE ดังได้กล่าวมาแล้ว

ในการรับหรือส่งใน BURST MODE จะเริ่มต้นที่บิท 0 ของหน่วยความจำตำแหน่ง 0 ก่อนเสมอ ไม่ว่าจะเป็น CLOCK/CALENDAR หรือ RAM

BURST MODE นี้มีไว้เพื่อความสะดวกในการรับหรือส่งข้อมูลครั้งละจำนวนมาก ๆ ทำให้ไม่ต้องส่ง COMMAND BYTE หลายครั้งนั่นเอง

การป้องกันการเขียน (WRITE PROTECT COMMAND BYTE)

เวลาที่เดินอยู่ภายในรีจิสเตอร์ที่ทำหน้าที่เก็บเวลา และข้อมูลที่อยู่ในหน่วยความจำทั้ง 24 ตำแหน่ง สามารถป้องกันมิให้เขียนข้อมูลใด ๆ ซ้อนลงไป เพื่อป้องกันเหตุการณ์ที่เกิดขึ้นโดยไม่ได้ตั้งใจ ทั้งนี้โดยการควบคุมจาก WRITE PROTECT REGISTER ซึ่งเป็นรีจิสเตอร์ตำแหน่งที่7ของรีจิสเตอร์ที่ทำหน้าที่เก็บเวลาดังแสดงในภาพที่ 40 โดยมี WRITE PROTECT BIT ซึ่งเป็นบิทที่ 7 ของรีจิสเตอร์ตัวนี้เป็นตัวกำหนดการทำงานหาก WRITE PROTECT BIT เป็น 0 หมายถึง สามารถเขียนข้อมูลใด ๆ ลงไปยังรีจิสเตอร์ที่เก็บเวลา หรือหน่วยความจำได้ หากบิทนี้เป็น 1 หมายถึงRTC อยู่ในสถานะป้องกันการเขียนข้อมูล ดังนั้นก่อนการเขียนข้อมูลใด ๆ ไปยัง CLOCK / CALENDAR หรือ RAM WRITE PROTECT BIT ต้องเป็น 0 เสมอ โดยการให้ RST เป็น 1 และโหลด WRITE PROTECTION COMMAND BYTE (BEH) ตามด้วย DATA BYTE ที่มีค่าเป็น 00H (เคลียร์ให้ WRITE PROTECT BIT เป็น 0) หลังจากนั้น RST ต้องกลับมามีสถานะเป็น 0 ก่อนที่คำสั่งอื่น ๆ จะเริ่มต้นทำงานได้

ส่วนในบังคับให้ RTC อยู่ในสถานะป้องกันการเขียน ข้อมูลก็ต้องให้ RST เป็น1 แล้วโหลด COMMAND BYTE ที่มีค่า BEH ตามด้วยข้อมูล BOH (ให้ WRITE PROTECT BIT เป็น 1) ในการทำงานแบบ BURST MODE เราไม่สามารถเขียนค่าใด ๆ เข้าไปใน WRITE PROTECT BIT ได้

การควบคุมการทำงาน

การรับหรือส่งข้อมูลทั้งหมด จะต้องเริ่มโดยให้ RST INPUT เป็น 1 ก่อนเสมอ โดย RST มีหน้าที่หลักอยู่ 2 ประการดังนี้

1. RST จะเป็นสัญญาณบอกให้ส่วน CONTROL LOGIC ทำงาน โดยจะอนุญาตให้มีการเขียนหรืออ่านข้อมูลใน SHIFT REGISTER ใน ADDRESS/COMMAND SEQUENCE

2. RST ใช้เป็นสัญญาณในการหยุดการทำงานใด ๆ กับ RTC DS1202

ข้อมูลของRTC

ในตอนเริ่มต้นติดต่อระหว่าง RTC DS1202 กับไมโครคอนโทรลเลอร์ ไบท์แรกจะต้องเป็น COMMAND BYTE เสมอ หากใน COMMAND BYTE ระบุว่าเป็นการเขียนข้อมูลไปในไอซีข้อมูลจะถูกรับเข้ามาในช่วงขอบขาขึ้น (RISING EDGE) ของ SCLK เท่านั้น โดยเริ่มต้นด้วยบิท0ก่อนเสมอและหากเป็นคำสั่งให้รับส่งครั้งละ1 ไบท์ เมื่อข้อมูลได้รับเข้ามาครบแล้วสัญญาณSCLKที่รับได้เกินจะถูกละเลยไป หากเป็นคำสั่งให้รับส่งแบบBURST MODE ซึ่งรับส่งครั้งละ 24 ไบท์ ก็มีลักษณะเช่นเดียวกัน คือ เมื่อรับข้อมูลครบ 24 ไบท์แล้ว สัญญาณ SCLK ที่รับได้เกินจะถูกละเลยเช่นกัน

ข้อมูลเอ้าท์พุท

หลังจากรับ COMMAND BYTE แล้ว หากมีการระบุว่าเป็นการอ่านข้อมูลจากRTC DS1202 ข้อมูลจะถูกส่งออกจากตัวไอซีสู่ภายนอกในขณะช่วงขอบขาลง (FALLING EDGE) ของSCLK หลังจากมีการรับ COMMAND BYTE เรียบร้อยแล้ว นั่นคือบิทแรกที่ส่งออกจาก RTC จะเกิดขึ้นในช่วงขอบขาลงของสัญญาณนาฬิกาลูกที่ 9 นั่นเอง (ต่อจาก COMMAND BYTE)

เวลาและปฏิทิน(CLOCK & CALENDAR)

บิท 7 ของรีจิสเตอร์ที่เก็บค่าวินาที (ดู ในรูปที่ 40) จะเป็นตัวบอกให้ RTC DS1202 หยุดการทำงานของวงจรในส่วน OSCILLATOR เมื่อบิทนี้มีค่าเป็น1 ซึ่งเป็นผลให้นาฬิกาภายในชิพหยุดการทำงานไปด้วย และจะบังคับให้ไอซีอยู่ในสถานะ LOW POWER STANDBY MODE โดยใช้กระแสไม่เกิน 100nA และเมื่อบิทนี้เป็น 0 อีกครั้ง วงจร OSCILLATOR จะเริ่มทำงานต่อทันที

การตั้งเวลาโหมด AM-PM / 12-24 MODE

บิท 7 ของรีจิสเตอร์ที่เก็บค่าชั่วโมง (ดูในรูปที่ 40) ถูกกำหนดให้เป็น 12/24 HOUR MODE SELECT BIT นั่นคือเป็นตัวเลือกว่าจะให้รีจิสเตอร์นี้เก็บค่าชั่วโมงแบบ 12 ชั่วโมง หรือ 24 ชั่วโมง โดย

- บิทที่ 7 เป็น 1 จะเป็นการเลือกให้เก็บค่าแบบ 12 ชั่วโมง โดยมีบิท 5 จะเป็นตัวบอกว่าเป็นช่วงกลางวันหรือกลางคืน (AM / PM INDICATOR) โดย 1 จะหมายถึง PM และ 0 หมายถึง AM

- บิทที่ 7 เป็น 0 จะเป็นการเลือกให้เก็บค่าแบบ 24 ชั่วโมง และบิท 5 จะเป็นบิทที่แสดงหลักสิบตัวที่สองของชั่วโมง (20-23)

- บิทที่ 7 ของ WRITE PROTECT REGISTER จะเป็น WRITE PROTECT BIT ดังได้กล่าวมาแล้วในเรื่อง WRITE PROTECT COMMAND BYTE โดย 7 บิทแรกถูกบังคับให้เป็น 0 หมด ทำให้อ่านได้ค่าเป็น 0 เสมอ

เวลาและปฏิทินในโหมดการส่งหลายไบต์(CLOCK /CALENDAR BUST MODE)

COMMAND BYTE ที่มีค่า RAM จะเป็นการระบุว่าให้มีการเขียนข้อมูลในรีจิสเตอร์ที่เก็บเวลาในแบบ BURST MODE หาก COMMAND BYTE มีค่า FEH จะระบุว่าให้มีการอ่านข้อมูลในรีจิสเตอร์นี้ในแบบ BURST MODE เช่นกัน (ดูรายละเอียดในภาพที่ 4) ซึ่งใน CLOCK / CALENDAR BURST MODE นี้ จะมีการรับหรือส่งข้อมูลคราวละ 8 ไบท์ติดต่อกัน (รับหรือส่งข้อมูลในรีจิสเตอร์ที่เก็บค่าเวลาทั้ง 8 ตัว) โดยจะเป็นการรับหรือส่งขึ้นกับ COMMAND BYTE ดังได้กล่าวมาแล้ว โดยข้อมูลที่รับหรือส่งจะเริ่มต้นด้วยบิท 0 ของรีจิสเตอร์ 0 ก่อนเสมอ

การเขียนข้อมูลแบบหลายไบต์ในหน่วยความจำ (RAM BURST MODE)

COMMANDBYTE ที่มีค่า FEH จะเป็นการระบุว่าให้มีการเขียนข้อมูลในหน่วยความจำแบบ BURST MODE หาก COMMAND BYTE มีค่า FEH จะระบุว่าให้มีการอ่านข้อมูลในหน่วยความจำแบบ BURST MODE ทำนองเดียวกันใน CLOCK / CALENDAR BURST MODE จะมีข้อแตกต่างกันก็เพียงจำนวนข้อมูลที่รับหรือส่งเท่านั้น เพราะหน่วยความจำใน RTC DS1202 มีขนาด 24 ไบท์ ดังนั้น RAM BURST MODE นี้ข้อมูลจำนวน 24 ไบท์ จะรับส่งกัน โดยเริ่มต้นด้วยบิท 0 ของ หน่วยความจำตำแหน่ง 0 ก่อนเสมอ

การเลือกใช้คริสตอล

RTC DS1202 ใช้คริสตอลความถี่ 32.768 KHz เป็นตัวกำหนดคาบเวลาในการทำงาน โดยมักจะใช้ของบริษัท Daiwa เบอร์ DT26S หรือของบริษัท Seiko เบอร์ DS-VT 200 หรือเบอร์ที่เทียบเท่ากันคริสตอลความถี่ 32.752 KHz นี้สามารถต่อโดยตรงเข้ากับขา 2 และ 3 (x1,x2)ของDS1202 ได้เลยโดยค่าอิมพีแดนซ์ของตัวคริสตอลเองควรเป็นค่า CAPACITANCE (C_L) ขนาด 6 pF

บทที่ 5

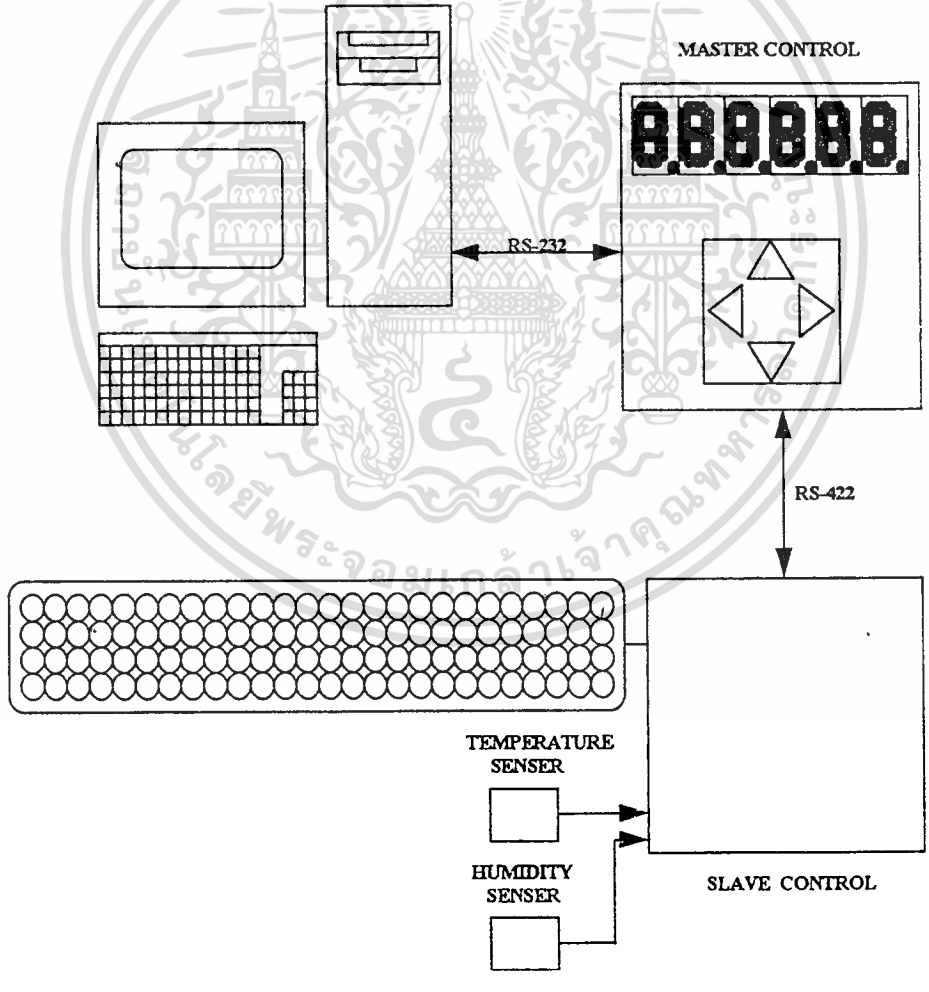
ส่วนประกอบฮาดร์แวร์ ของระบบ

โครงสร้างฮาดร์แวร์ของระบบ

โครงสร้างฮาดร์แวร์ของเครื่องแสดงเวลาและปฏิทินแบบระยะไกล (Time & Calendar Remote Monitor) มีส่วนประกอบหลักที่สำคัญ 2 ส่วนคือ

1. บอร์ดควบคุมหลัก (Master Control Board)
2. บอร์ดควบคุมรอง (Slave Control Board)

นอกจากนี้ยังประกอบด้วยส่วนอื่น ๆที่ต่อร่วมกับบอร์ดทั้งสอง ดังแสดงไว้ในรูปที่ 41



รูปที่ 41 บล็อคแสดงโครงสร้างฮาดร์แวร์ของระบบทั้งหมด

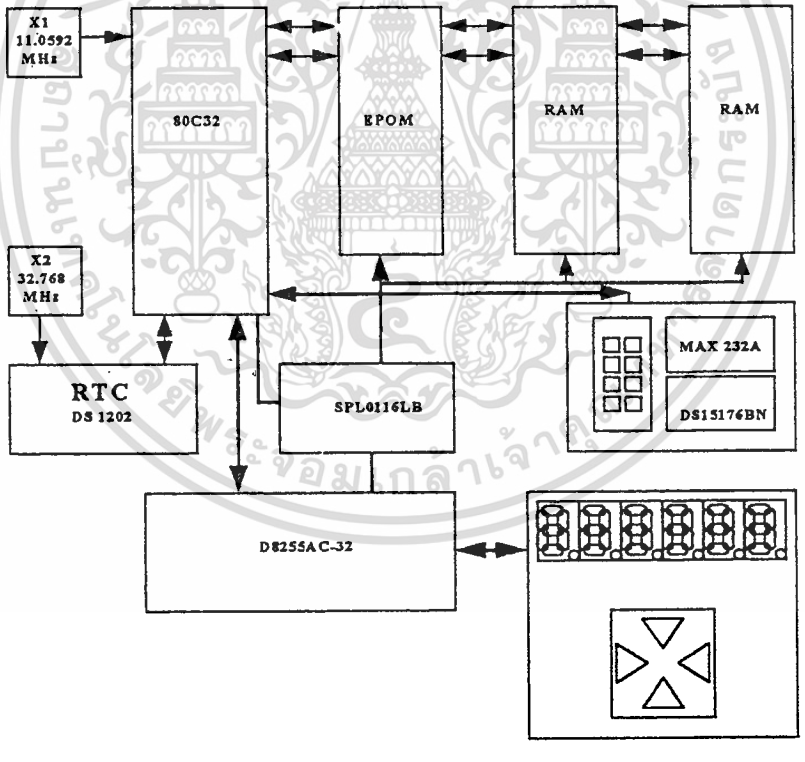
ฮาดร์แวร์ของบอร์ดควบคุมหลัก (MATER CONTROL BOARD)

การทำงานของ Master Board จะแสดงใน รูปที่ 42โดยมีส่วนประกอบและหน้าที่ที่สำคัญดังนี้คือ

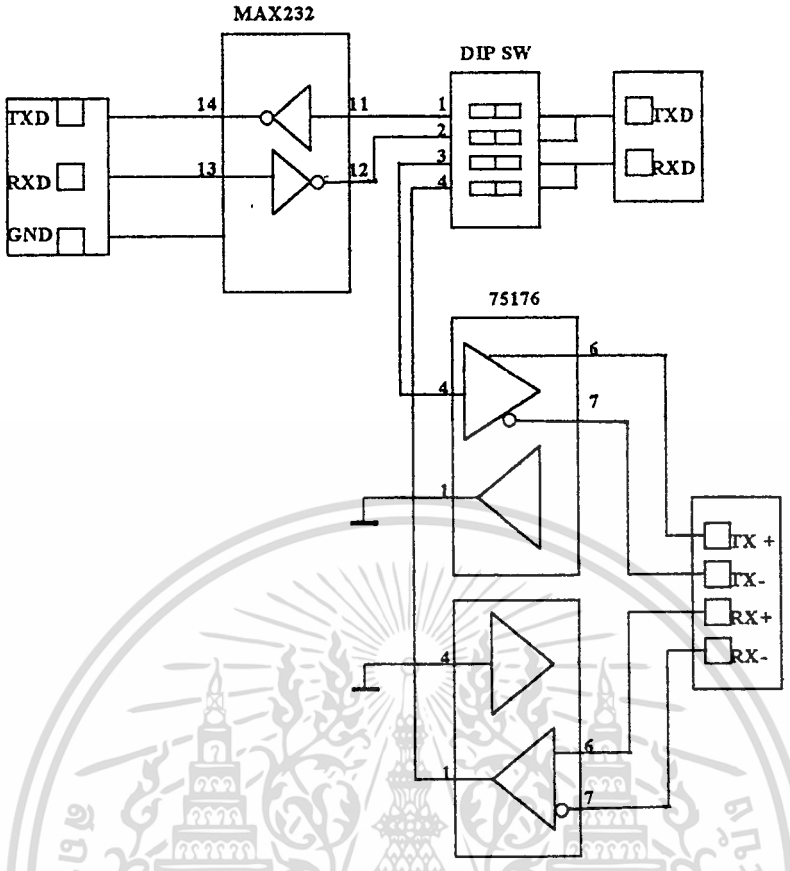
1 Control Board ที่ เป็นหัวใจหลักของการควบคุมโดยใช้ CPU 8032 และมี RTC ที่ใช้เป็นฐานเวลาโดยมีแบตเตอร์รี่Backup ในกรณี ที่เกิดกระแสไฟฟ้าขัดข้อง ในส่วนรายละเอียดของ RTC ตามที่ได้อธิบายไว้แล้วส่วนคุณสมบัติของทั้งหมดจะอธิบายต่อไป

2. ชุด Receiver & Driver ของ RS- 232 และ RS- 422 ดังแสดงไว้ในรูปที่ 43 โดยมีชิพ Max 232 จะทำหน้าที่Receive & Drive ของการสื่อสารอนุกรม RS 232 และ ชิพ 15176 จะทำหน้าที่ Receive & Drive ของการสื่อสารอนุกรม RS 422 และรายละเอียดชิพทั้งสองดูได้จากภาคผนวก

3. ชุด KeyBoard และ Display 7 Segment สำหรับการ SET ข้อมูลทั้งหมดใน RTC และใช้แสดงผล ข้อมูลจาก RTC และ Temperature,Humidity



รูปที่ 42 บล็อคแสดงการทำงานของของ Master control board



รูปที่ 43 block diagram การรับส่งสัญญาณ ของ Master control board

รายละเอียดเกี่ยวกับการใช้งาน MASTER BOARD

การปรับจัมพ์เปอร์

จัมพ์เปอร์ EA สำหรับเลือกใช้หน่วยความจำโปรแกรม (PROGRAM MAMORY) ตำแหน่งแอดเตรสเริ่มต้น OOOOH เป็น INT (INTERNAL) หรือEXT. (EXTERNAL)

จัมพ์เปอร์ RESE สำหรับเลือกสัญญาณรีเซท CPU จากวงจร RC หรือ MAX691

จัมพ์เปอร์ U2 SIZE สำหรับเลือกขนาดหน่วยความจำโปรแกรม U2 (EPROM) เป็นขนาด 8,16kbyte (2764,27128) หรือ 32kbyte (27256)

จัมพ์เปอร์ U3 SIZE สำหรับเลือกขนาดหน่วยความจำข้อมูล U3 (RAM) เป็น 8kbyte (6264) หรือ 32KByte (62256)

จัมพ์เปอร์ U4 TYPE สำหรับเลือกชนิดหน่วยความจำ U4 เป็น EPROM (หน่วยความจำโปรแกรม) หรือ RAM (หน่วยความจำข้อมูล)

จัมพ์เปอร์ U4 SIZEสำหรับเลือกขนาดหน่วยความจำโปรแกรมและข้อมูล U4 เป็น8kbyte (XX64) หรือ 32kbyte (XX256)

จัมพ์เปอร์ BACKUPสำหรับเลือก NO/OFF การสำรองข้อมูล(MAX691)ของ U3

จัมพ์เปอร์ WDสำหรับเลือก EN. (ENABLE) / DIS.(DISABLE) วงจร WATCHDOG (MAX691)

จัมพ์เปอร์ PF สำหรับเลือก EN.(ENABLE) / DIS.(DISABLE) วงจร POWER FAIL DETECTOR (MAX691)

* จัมพ์เปอร์ BACKUP, WDและPFจะใช้งานได้ก็ต่อเมื่อเสียบใช้งานชิพ MAX691 ด้วย*

การปรับจัมพ์เปอร์เลือกหน่วยความจำ

ไมโครคอนโทรลเลอร์ตระกูล 8031(32) สามารถต่อกับหน่วยความจำภายนอกได้ถึง 128 Kbyte โดยแบ่งออกเป็นสองส่วนคือ 64Kbyte เป็นหน่วยความจำโปรแกรม(PROGRAM MEMORY)และอีก 64 Kbyte เป็นหน่วยความจำข้อมูล (DATA MEMORY) ซึ่งหน่วยความจำทั้งสองส่วนนี้มีตำแหน่งแอดเดรสที่ 0000H - FFFFH เหมือนกัน แต่จะถูกแยกออกจากจากกันด้วยสัญญาณควบคุมที่ต่างกัน โดยสัญญาณ PSEN Ⓡ ใช้ควบคุมการอ่านและเขียนหน่วยความจำโปรแกรม(EPROM) สัญญาณ RDⓇ และ WR Ⓡ ใช้ควบคุมการอ่านและเขียนหน่วยความจำข้อมูลและพอรทอินพุท / เอาท์พุท และสำหรับการอ่านหน่วยความจำโปรแกรม และ ข้อมูล (PROGRAM AND DATA MEMORY) ใช้สัญญาณ GETⓇ ซึ่งสัญญาณนี้ได้จากการ AND สัญญาณ PSEN Ⓡ และ RD Ⓡหน่วยความจำส่วนนี้สามารถใช้ได้กับERROM หรือ RAM

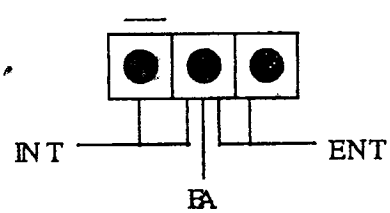
สำหรับบอร์ดควบคุมหลักได้จัดหน่วยความจำออกเป็น 3 ส่วนด้วยกัน คือ

U2 เป็นหน่วยความจำโปรแกรม (PROGRAM MEMORY) แอดเดรส 0000H - 7FFFH

U3 เป็นหน่วยความจำข้อมูล (DATA MEMORY) แอดเดรส 0000H - 7FFFH

U4 เป็นหน่วยความจำโปรแกรมและข้อมูล (PROGRAM AND DATA MEMORY) แอดเดรส8000H-F7FFHส่วนแอดเดรสF800H -FFFFH ใช้เป็นตำแหน่งของพอร์ทอินพุท/เอาท์พุท หน่วยความจำ U2 , U3และ U4 สามารถเลือกขนาด (SIZE) และชนิด (TYPE) ได้ด้วจัมพ์เปอร์ดังต่อไปนี้

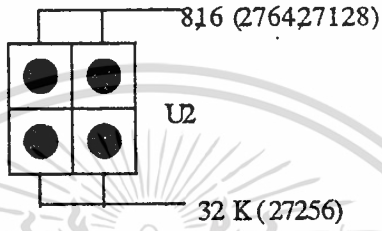
<u>การเลือกใช้หน่วยความจำโปรแกรมภายใน/ภายนอก (จัมพ์เปอร์ EA)</u>



รูปที่ 44 การเลือกใช้หน่วยความจำโปรแกรมภายใน/ภายนอก

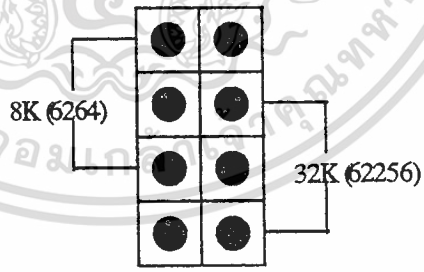
ในการใช้งานหน่วยความจำโปรแกรมที่แอดเดรสเริ่มต้น 0000 H สามารถเลือกใช้หน่วยความจำส่วนนี้ได้ด้วยจัมพ์เปอร์ EA ทั้งแบบใช้หน่วยความจำโปรแกรมภายใน (INTERMAL PROGRAM MEMORY) สำหรับ CPU เบอร์ 8051, 8052, 8751, 8752, 8052AHBASIC โดยจัมพ์เปอร์นี้ที่ตำแหน่งINT(INTERNAL)หรือใช้หน่วยความจำโปรแกรมภายนอก (EXTERNAL PROGRAM MEMORY) สำหรับ CPU เบอร์ 8051,8032ปรับจัมพ์เปอร์นี้ไปที่ ตำแหน่ง EXT. (EXTERNAL)

การเลือกขนาดหน่วยความจำ U2 (จัมพ์เปอร์ U2 SIZE)



รูปที่ 45 การเลือกขนาดหน่วยความจำ U2

จัมพ์เปอร์ U2 SIZE สำหรับเลือกขนาดหน่วยความจำโปรแกรม (PROGRAM MEMORY) ตำแหน่งU2 แอดเดรสเริ่มต้นที่ 0000H เป็น EPROM ใช้ได้ 3 ขนาดคือ 8,16 Kbyte (เบอร์ 2764, 27128) หรือ 32 Kbyte (เบอร์ 27256)



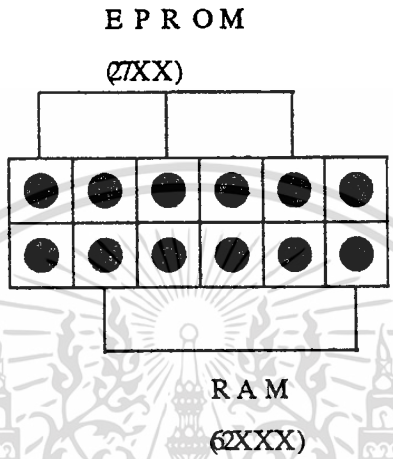
รูปที่ 46 การเลือกขนาดหน่วยความจำ U3

จัมพ์เปอร์ U3 SIZE สำหรับเลือกขนาดหน่วยความจำข้อมูล (DATA MEMORY) ตำแหน่ง U3 แอดเดรสเริ่มต้นที่ 0000H เป็น RAM ใช้ได้ 2 ขนาดคือ 8 Kbyte (เบอร์ 6264) หรือ 32 Kbyte (เบอร์ 62256)

*หน่วยความจำ U3 (RAM) นี้สามารถสำรองข้อมูลในช่วงไฟดับ (BACKUP) ได้โดยบนบอร์ด ANT-32 ต้องมีชิพ MAX691 แบตรี่ลิเธี่ยม และจัมพ์เปอร์ BACKUPต้องอยู่ที่ตำแหน่ง ON

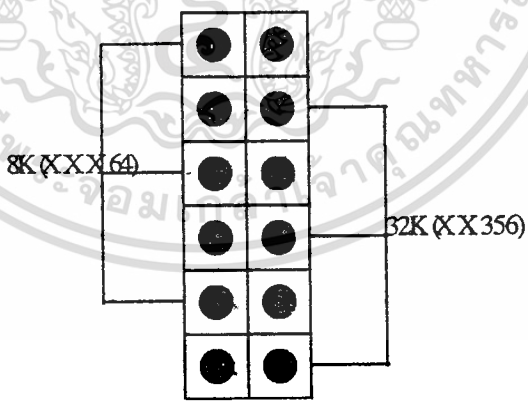
การเลือกชนิดหน่วยความจำ U4 (จัมพ์เปอร์ U4 TYPE)

จัมพ์เปอร์ U4 TYPE สำหรับเลือกชนิดหน่วยความจำโปรแกรมและข้อมูล (PROGRAM AND DATA MEMORY) ตำแหน่ง U4 แอดเดรสเริ่มต้นที่ 8000 เป็น EPROM (27X X X) หรือ RAM (62 X X X) และขนาดของ U4 เลือกได้ด้วยจัมพ์เปอร์ U4 SIZE



รูปที่ 47 การเลือกขนาดหน่วยความจำ U4 ตัวที่ 1

การเลือกขนาดหน่วยความจำ U4 (จัมพ์เปอร์ U4 SIZE)



รูปที่ 48 การเลือกขนาดหน่วยความจำ U4 ตัวที่ 2

จัมพ์เปอร์ U4 SIZE สำหรับเลือกขนาดหน่วยความจำโปรแกรมและข้อมูล (PROGRAM AND DATAMEMORY) ตำแหน่งU4 (เลือกเป็น EPROM หรือRAM จัมพ์เปอร์ U4) ได้ 2 ขนาด คือ 8 Kbyte (EPROM เบอร์ 2764, RAM เบอร์ 6264) หรือ 32 Kbyte

การใช้งานหน่วยความจำขนาด 32 Kbyte ที่ตำแหน่ง U4 นี้จะสามารถใช้ได้ 30 Kbyte (แอดเดรส8000H - F7FFH) เท่านั้น

การขยายอินพุท/เอาท์พุทบนบอร์ดควบคุมหลัก

8255 Programmable Peripheral Interface (PPI) เป็นชิพ พอร์ทแบบขนานที่เป็นที่นิยมใช้งานกันมากมาย สำหรับบอร์ด ANT - 32 ใช้พอร์ท 8255 จำนวน 2 ตัวทำหน้าที่เป็นพอร์ท ทำให้มีพอร์ทอินพุท/เอาท์พุทถึง 24 x 2 = 48 บิท โดยแบ่งเป็นUSER PORT 1 และ 2 มีตำแหน่งแอดเดรสดังนี้

USER PORT 1 (U10) แอดเดรส F800H + 8255 offset addr = actual addr

Port A	ตำแหน่งแอดเดรส	F800H + 00H = F800H
Port B	ตำแหน่งแอดเดรส	F800H + 01H = F801H
Port C	ตำแหน่งแอดเดรส	F800H + 02H = F802H
Mode Port	ตำแหน่งแอดเดรส	F800H + 03H = F803H

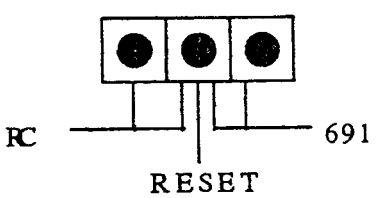
USER PORT 2 (U11) แอดเดรส FC00H + 8255 offset addr = actual addr

Port A	ตำแหน่งแอดเดรส	FC00H + 00H = FC00H
Port B	ตำแหน่งแอดเดรส	FC00H + 01H = FC01H
Port C	ตำแหน่งแอดเดรส	FC00H + 02H = FC02H
Mode Port	ตำแหน่งแอดเดรส	FC00H + 03H = FC03H

ก่อนที่จะใช้งานพอร์ท 8255 ผู้ใช้ต้องทำการกำหนดหมดการทำงาน (configulation) ของพอร์ท A, Bและ C ให้เป็นพอร์ทอินพุทหรือเอาท์พุท โดยทำการเขียนค่า control code ไปที่ Mode Port ซึ่ง ModePort นี้สามารถเขียนได้เท่านั้นไม่สามารถอ่านได้ในที่นี้จะกล่าวเฉพาะการทำงานในโหมด0 ซึ่งเป็นโหมดที่ใช้ได้สะดวกและง่ายต่อการทำความเข้าใจดังแสดงค่า control code ดังตารางในรูปที่ 50

Microprocesser Reset (จัมพ์เปอร์ Reset)

สัญญาณรีเซ็ท CPU บนบอร์ด ANT - 32 สามารถเลือกใช้ได้จาก 2 แหล่ง คือ RC (จากวงจรรีเซทของ MAX 691) โดยจัมพ์เปอร์ RESET



ในรูปที่ 49 การปรับจัมพ์เปอร์ RESET

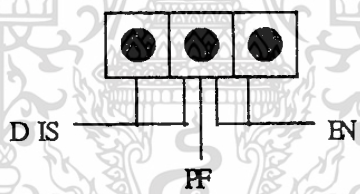
Port A (PA0 - PA7)	Port C บน (PB4 - PC7)	Port B (PB0 - PB7)	Port C ล่าง (PC0 - PC3)	Control Code (hex)
output	output	output	output	80H
output	output	output	input	81H
output	output	input	output	82H
output	output	input	input	83H
output	input	output	output	88H
output	input	output	input	89H
output	input	input	output	8AH
output	input	input	input	8BH
input	output	output	output	90H
input	output	output	input	91H
input	output	input	output	92H
input	output	input	input	93H
input	input	output	output	98H
input	input	output	input	99H
input	input	input	output	9AH
input	input	input	input	9BH

รูปที่ 50 แสดงตาราง Control codeของ 8255

ถ้าไม่ได้ใช้งาน MAX 691ให้ปรับจัมพ์เปอร์ไปที่ตำแหน่ง RCวงจร RC วงจร R/C รีเซท จะทำการรีเซท CPU ในช่วง power upเท่านั้น ส่วนสัญญาณรีเซท CPU ของ M AX691จะทำ การรีเซ็ท CPU ในช่วง power up และ power Down โดยที่ขา RESET จะเป็นลอจิก "1" เมื่อแรงดัน Vcc ตกลงต่ำกว่า4.75 โวลท์ประมาณ 50 มิลลิเซคคั่น (ms) ซึ่งก็หมายความว่า CPU จะถูกรีเซทเมื่อเริ่มจ่ายไฟด้วยพัลส์ที่มีความกว้าง 50 ms และจะถูกรีเซทอีกครั้งเมื่อไฟตก นอกจากนี้แล้วขา RESET จะถูกใช้เมื่อเลือกใช้งานWatchdog Timer ซึ่งจะได้กล่าวถึงในหัวข้อถัดไป

จัมพ์เปอร์ PF (Power Fail Detector)

วงจรตรวจจับแรงดันไฟตก (Power Fail Detector) สำหรับระบบไมโครฯนับว่ามีความสำคัญมากกับระบบที่ต้องการเก็บค่าพารามิเตอร์หรือข้อมูล บางอย่างลง RAM ก่อนที่ระบบจะหยุดทำงาน เพื่อที่ว่าเมื่อระบบเริ่มทำงานใหม่อีกครั้งจะได้นำเอาข้อมูลที่เก็บไว้ก่อน ไฟดับมาประมวลผลเพื่อใช้งานต่อไป โครงสร้างภายในของวงจรนี้เป็นวงจร voltage comparator โดยรับแรงดันอินพุทที่ต้องการตรวจสอบจากภายนอกเข้าที่ขา PEI (Power Fail Input) นำมาเปรียบเทียบกับแรงดันอ้างอิง (reference voltage) 1.3 โวลท์ ซึ่งขาเอาท์พุทคือขาPFO (Power Fail Output) จะเป็นลอจิก "O" เมื่อแรงดันที่ขา PEI ต่ำกว่า 1.3 โวลท์ ขา PFI รับแรงดันมาจากวงจร voltage divider ภายนอกซึ่งก็คือ Rx และ Ry โดยการทำการตรวจจับแรงดัน Vcc 5 โวลท์ ค่าอัตราส่วนของวงจร voltage divider สามารถกำหนดได้จากหลักการที่ว่าแรงดันที่ขา PFIจะตกลงถึงค่า 1.3 โวลท์ก่อนที่แรงดัน +5โวลท์จากแหล่งจ่ายไฟจะตกลงถึง4.75 โวลท์ โดยปกติแล้วขาPFOจะต่อเข้ากับขาอินเทอร์รัพท์ของ CPU เพื่อที่ว่าเมื่อเกิดสภาวะไฟตก CPU จะถูกอินเทอร์รัพท์เพื่อให้ CPU ไปทำขบวนการนำเอาข้อมูลที่จะเป็นเก็บลง RAM ก่อนที่แรงดัน Vcc จะตกลงต่ำกว่า 4.75 โวลท์ และที่แรงดันนี้เองที่ CPU จะถูกรีเซทอีกครั้ง (power down reset)



รูปที่ 51 การปรับจัมพ์เปอร์ Power Fail

ขา PRO ของMAX69จะต่อเข้าขา INTI ของ CPU สามารถเลือกใช้งานวงจร Power Fail Detector นี้ได้ด้วยจัมพ์เปอร์ PF โดยปรับไปที่ EN. (enable) หรือไม่ใช้งานโดยปรับไปที่ DIS. (disable)

การคำนวณหาค่า Rx, Ry

กำหนดค่าแรงดัน Vcc ในขณะไฟตกในช่วงที่ CPU ยังสามารถทำงานได้ที่ 4.8 โวลท์ และเพื่อให้ง่ายต่อการคำนวณจึงกำหนดค่า Ry = 10K สามารถคำนวณหาค่า Rx ได้ดังนี้

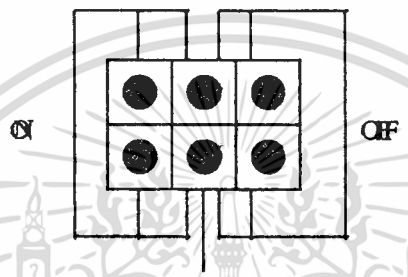
$$Rx = (4.8 - 1.3) / (1.3/Ty)$$
$$= 3.5 / (1.3/10K)$$
$$= 26.923K$$

เลือกใช้ค่า Rx = 27K, Ry = 10K

ค่าแรงดัน 4.8 โวลท์นี้สามารถเปลี่ยนแปลงได้ตามลักษณะของงานที่ใช้

จัมพ์เปอร์ BACKUP

การสำรองข้อมูล(data backup)ของCMOS RAM (U3) และRTC (DS1202)ในช่วงไฟดับ สำหรับบอร์ด ANT - 32 ใช้ MAX691 จัดการในส่วนของแบตเตอรี่ และสัญญาณ Chip Enable ของRAM โดย MAX691 รับแรงดันจากแบตเออรี่ลิเธี่ยม 3โวลท์ เข้าที่ขา Vbatt และแรงดัน Vout จาก MAX691 เข้าที่ขา Vcc ของ CMOS RAM และ RTC ในการใช้งานปกติใช้ไฟ 5 โวลท์ ที่ขา Vout จะเสมือนถูกต่อเข้ากับขา Vcc ภายใน MAX691 และจะต่อเข้ากับขา Vbatt เมื่อแรงดัน Vcc ต่ำกว่าแรงดันของแบตเตอรี่ ดังนั้นแบตเตอรี่จะถูกใช้งานเฉพาะในช่วงที่ไฟดับเท่านั้น



BACKUP

รูปที่ 52 การปรับจัมพ์เปอร์ Power backup

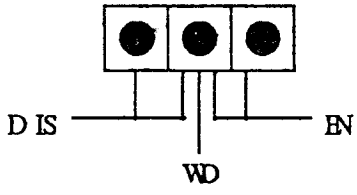
สำหรับ CMOS RAM เพื่อป้องกันข้อมูลสูญหายในขณะไฟดับ MAX691 ได้ออกแบบวงจรควบคุมสัญญาณ Chip Enable ของ RAM โดยที่ขา CE IN ของ MAX691 รับสัญญาณ Chip Enable มาจากวงจร Address Decoder และสัญญาณที่ขา CE OUT จากMAX691 จะต่อเข้าขา Chip Eanble ของ RAM สัญญาณที่ขา CE OUT จะเป็นไปตามสัญญาณที่ขา CE IN ก็ต่อเมื่อแรงดัน Vcc สูงเกิน 4.65 โวลท์ และจะเป็นลอจิก "1" เมื่อแรงดัน Vcc ต่ำกว่า 4.65 โวลท์เพื่อป้องกัน CPU เขียนค่าข้อมูลที่ไม่ถูกต้องลง RAM ในระหว่าง power up และ power down การสำรองข้อมูลของ RAM (U3) สามารถเลือก ON/OFF ได้ด้วยจัมพ์เปอร์ BLACKUP ซึ่งมีอยู่2 ตัว ส่วน RTC นั้นจะทำการสำรองข้อมูลตลอดเวลา ไม่สามารถ OFF ได้

Watchdog Timer (จัมพ์เปอร์ WD)

Watchdog Timer เป็นวงจรที่ทำหน้าที่ตรวจสอบการทำงานของระบบไมโครฯ ว่าทำงานในสภาวะปกติหรือไม่ ถ้าระบบทำงานผิดปกติหรือเกิดอาการ แฮงค์ วงจรส่วนนี้จะทำการรีเซท CPU ให้เริ่มทำงานใหม่อีกครั้ง Watchdog Timer จึงมีความสำคัญและจำเป็นมากสำหรับระบบไมโครฯ ซึ่งจะเป็นการเพิ่มเสถียรภาพการทำงานของระบบให้ดียิ่งขึ้น

หลักการทำงานของ Watchdog Timer ก็คือ CPU ต้องส่งสัญญาณไปกระตุ้นที่ขา WDI (WATCHDOG INPUT) ของ MAX691 โดยใช้พอร์ต P1.7 ที่ขา OSC IN และ OSC SEL ของ MAX691 ไม่ได้ต่อใช้งาน (ปล่อยลอยไว้) CPU ต้องทำการเปลี่ยนสภาวะ (toggle) ที่ขา WDI

ทุกๆ 1.6 วินาที (WatchdogtimeoutPeriod = 1.6 วินาที) โดยใช้คำสั่ง CPL P1.7 เพื่อให้แน่ใจว่าซอฟท์แวร์ ได้ทำงานอย่างถูกต้อง ถ้าฮาร์ดแวร์หรือซอฟท์แวร์เกิดทำงานผิดพบาด ซึ่งจะมีผลทำให้สภาวะที่ขา EDI ไม่เปลี่ยนแปลงตามเวลาที่กำหนดไว้



รูปที่ 53การปรับจัมพ์เปอร์ Watchdog Timer

MAX691 จะส่งสัญญาณรีเซทเป็นพัลส์บวกที่ขา RESET กว้าง50 ms เพื่อรีเซทให้ CPU กลับไปทำงานใหม่อีกครั้ง และที่ขา RESET นี้จะส่งพัลส์รีเซทออกมาทุก ๆ 1.6 วินาที จนกว่าจะมีการเปลี่ยนสถานะที่ขา WDI อีกครั้ง

การใช้งาน Watchdog Timer สามารถเลือกใช้งานได้ด้วยจัมพ์เปอร์ WD โดยปรับไปที่ EN.(enable) เมื่อต้องการใช้งาน หรือปรับไปที่ DIS (disable) เมื่อไม่ต้องการใช้และในกรณีต้องการเปลี่ยนค่า Watchdog Timeout Period เป็นค่าอื่น สามารถกรทำได้โดยต่อสายจัมพ์ที่ขา OSC SEL (ขา 8) ต่อลงground และเพิ่มคาปาซิเตอร์ CX ที่ขา 7 ตามตำแหน่งที่พิมพ์ไว้บนบอร์ด ANT - 32 ซึ่งจะมีผลทำให้ค่าความกว้างของพัลส์รีเซท (Reset Timeout Period) เปลี่ยนไปตามค่า Cx ด้วยดังแสดงไว้ในตารางในรูป ที่ 54

OSC SEL	OSC IN	Watchdog Timeout Period		Reset Timeout
ขา 8	ขา 7	Normal	After Reset	Period
Floating	Floating	1.6 Sec	1.6 sec	50 ms
low	Cx	(400 ms/47pF)Cx	1.6 sec/47pF) Cx	(200ms/47pF)Cx

รูปที่ 54 ตาราง การเลือกค่า Reset Pulse Width และ Watchdog Timeout

<u>ตัวอย่าง</u> ต้องการใช้งาน Watchdog Timer โดยกำหนดว่า Timeout Period = 1 วินาที ค่า Cx สามารถคำนวณโดยใช้สูตร

Cx = (Timeout Period 47 pF) / 400 ms

= (1 sec X 47 pF) / 0.4 sec

= 117.5 pF

เลือกใช้ค่า Cx = 100 pF

ค่า Watchdog Timeout Period สำหรับการใช้งานปกติ (Normal) และหลังจาก CPU ถูกรีเซท (Immediately After Reset) ค่าReset Timeout Period เมื่อใช้ค่า Cx 100pF หาได้จากสูตรไว้ในตารางที่ 2 ดังนี้

Watchdog Timeout Period (Normal) = (400ms / 47 pF) X 100pF

= 0.85 sec

Wctchdog Timeout Period (after Reset) = (1.6sec / 47 pF) X 100 pF

= 3.4 sec

Reset Timeout Period = (200ms / 47pF) X 100pF

= 0.43 sec

คำแนะนำสำหรับการใช้งาน Watchdog Timer

ในการเขียนโปรแกรมเพื่อใช้งาน Watchdog Timer ควรจะพัฒนาโปรแกรมด้วยภาษาแอสเซมบลี้ ซึ่งเมื่อเริ่มเขียนโปรแกรมต้องไม่ใช้ Watchdog Timer โดยปรับจัมพ์เปอร์ WD ไปที่ DIS (disable) และเมื่อเขียนโปรแกรมจนได้โปรแกรมที่ทำงานได้ตามที่ต้องการแล้วจึงทำการแทรกคำสั่ง CPL P1.7 โดยประมาณหรือใช้วิธีคำนวณดูว่า CPU จ ะวนกลับมาทำคำสั่งนี้ภายในเวลาไม่เกินค่าของ Watchdog Timeout ปรับจัมพ์เปอร์ WD ไปที่EN.(enable)เพื่อใช้งานWatchdog Timer ผู้ใช้สามารถตรวจสอบว่า CPUได้กลับมาทำคำสั่งนี้ภาในเวลาที่กำหนดทันหรือไม่โดสังเกตจากการทำงานของโปรแกรม ถ้า CPU กลับมาทำคำสั่งนี้ภายในเวลาที่กำหนดโปรแกรมจะทำงานได้ตามปกติเหมือนกับ การทำงานของโปรแกรมก่อนที่จะใช้งาน Watchdog Timer แต่ถ้า CPU จะถูกรีเซทโดยจะกลับไปทำงานในส่วนต้น ของโปรแกรมตลอดเวลา หรืออาจใช้ลอจิกโพรบ, ออสซิลโลสโคป ตรวจจับสัญญาณที่ขา P1.7 ของ CPUโดยต้องให้ระยะห่างระหว่างพัลส์แต่ละลูกไม่เกินค่า Watchdog Timeout Period

การใช้งาน REAL TIME CLOCK บนบอร์ดควบคุมหลัก

สำหรับการใช้งานระบบไมโครฯ ที่มีเวลามาเกี่ยวข้องด้วย จำเป็นต้องมีวงจรในส่วนที่ทำหน้าที่เป็นRTC (Real time Cleck) คือ นาฬิกาเวลาจริง ซึ่งบอร์ดควบคุมหลักใช้ชิพ RTC เบอร์ DS1202 Serial TimekeeperChip ของ DALLAS SEMICNDUCTOR โดยต่อร่วมกับอุปกรณ์ภายนอกเพียงเล็กน้อยและที่สำคัญคือ ไม่ต้องทำการปรับแต่ง ซึ่งเมื่อจะใช้ RTC นี้ต้องมีชิพ DS1202 และ MAX691 รวมทั้งคริสตอล32.768 Khzและแบตเตอรี่ลิเธี่ยมบนบอร์ดควบคุมหลักด้วย

DS1202 (U7) ประกอบไปด้วย Real Time Clock / Calender และ Static RAM ขนาด 24 ไบท์ ทำการอินเตอร์เฟสกับ CPU ในแบบอนุกรม โดยใช้สายเพียง 3 เส้นคือ (1) ขา RST® (Reset), (2) ขา I / 0(Data line) และ (3) ขา SCLK (Serial Clock) ขาสัญญาณทั้งสามนี้จะต่อเข้ากับขา P1.6, P1.4 และ P1.5 ของCPU ตามลำดับ เมื่อต้องการทราบค่าเวลา CPU ต้องทำการอ่านเวลาจาก RTC เพราะว่า DS1202 ไม่มีขาสำหรับไปอินเตอร์รัพท์ CPU สามารถเขียนหรืออ่านข้อมูลของ CLOCK หรือ RAM ได้ 2 วิธีคือ Single - byte และ Multiple - byte โดยทั้งสองวิธี CPU ต้องส่ง command byte (8 บิท) ให้ DS1202 เพื่อบอกให้ DS1202 ทราบว่าจะทำการเขียนหรืออ่าน CLOCK หรือ RAM พร้อมตำแหน่ง (address) และตามด้วยข้อมูล ในขณะที่กำลังติดต่อกับ DS1202 สัญญาณที่ขา RST® ต้องเป็นลอจิก "1" ขาSCLK จะเป็นสัญญาณSerial Clock เพื่อทำการเขียนหรืออ่านข้อมูล โดยจะใช้สัญญาณ Clock 1ลูกสำหรับข้อมูล 1 บิท ส่วนขา I / 0 เป็นข้อมูลอนุกรม โดยจะเป็นอินพุทเมื่อทำการเขียน และเป็นเอาท์พุท เมื่อทำการอ่าน โดยข้อมูลที่จะเขียนหรืออ่านจะเริ่มจากบิท 0 แล้วจะจบด้วยบิทที่ 7 ค่าของ commandbyte ในการเขียนหรืออ่าน CLOCK และ RAM แสดงไว้ดังตารางในรูปที่ 55

Register	Function	Command Address	Write=W Read=R	Range data (bcd)	7	6	5	4	3	2	1	0
0	Seccnds	80 80	W R	00-59	CH	10 SEC			SEC			
1	Minutes	82 83	W R	00-59	0	10 MIN			MIN			
2	12 Hrs 24 Hrs	84 85	W R	01-12 00-23		0 0			HOUR			
3	Date	86 87	W R	01-31	0	0	0		DATE			
4	Month	88 89	W R	01-12	0	0	0	1	MONT			
5	Day	8A 8B	W R	01-07	0	0	0	0	DAY			
6	Year	8C 8D	W R	00-99					YEAR			
7	Write Protect	8E 8F	W R	00-80		0	0	0	0	0	0	0

รูปที่ 55 ตาราง การเลือกค่า การอ่านและการเขียน Command Byte ของ RTC

SPECIFICATION ของ Master Board

CPU..... 80C32 CHMOS 8 - bits Microcontroller

CLOCK SPEED..... 11.0592 Mhz

INTERNAL RAM ..... 256 Byte

PROGRAM MEMORY..... U2 8 - 32 Kbyte : 2764, 27128,27256

(EPROM) default = 28 Pins socket

DATA MEMORY..... U3 8 - 32 Kbyte : 6264,62256(RAM -

defualt = 28 pins socket

PROGRAM & DATA MEMORY..... U4 8 - 32 Kbyte : 2764 27256 (EPROM)

6264 62256 (RAM)

2864 28256 (EEPROM)

default = 6264 8 Kbyte RAM

INTERNAL PORT..... 12 bits I / O (PORT1 TO T1 INTO INTI)

PORT..... USER PORT 1,2 48bits 8255 I / O

LCD PORT (DOT MATRIX ONLY)

OERIAL INTERFACE..... RS232C use MAX 232 chip

BACKPUP,WATCHDOG, PF DETECTOR.....use MAX 691 chip (options)

REAL TIME CLOCK.....use DS1202 chip (options)

DATA RETENTION TIME.....over 4 Years for RAM and RTC

CONNECTOR.....40 pins - System Ecpansion

26 pins - internal port

16 pins - User Port 1,2

14 pins - internal Port

3 pins - LCD Port

2 pins - RS232C

POWER CONSUMTION.....+ 5 VDC 168 mA (approximate)

SIZE..... 10.2 cm. x 14.2 cm.

ฮาร์ดแวร์ ของบอร์ดควบคุมรอง(SLAVE BOARD)

เนื่องจากจุดประสงค์ของ PROJECT นี้เพื่อการรับส่งข้อมูลแบบอนุกรม และต้องการแสดงผลออก DISPLAY BOARD ด้วยดังนั้นต้องหา CONTROLLLER ที่มีความสามารถในการตอบสนองการทำงานให้ตรงกับความต้องการ ด้วยเหตุผลดังกล่าวเราจึงต้องเลือกใช้ ANT-31 PJ

ANT-31PJ เป็นบอร์ดที่เน้นสำหรับงานพัฒนาโดยเฉพาะซึ่งตัวบอร์ดใช้ชิพเบอร์ 80C31 ซึ่งอยู่ในตระกูล MCS-51 ของ INTEL โดยเป็นตระกูลที่นิยมกันอย่างมากในงานควมคุมต่างๆ บนบอร์ดสามารถใส่หน่วยความจำได้ 2 ตัว โดยทำให้ใช้งานได้สูงสุดถึง 40 KBYTE พร้อมทั้ง 8255 USER PORT ที่มี I/O ในการใช้งาน 24 บิท นอกจากนี้ยังมี LCD PORT ที่สามารถต่อเข้ากับ LCD MODULE ได้โดยตรง และส่วนที่เป็น WORKING AREAจะมีจุดต่อสัญญาณต่างๆ ที่สำคัญรอไว้พร้อม ทำให้สะดวกอย่างมากต่อการพัฒนาส่วนเพิ่มเติมเข้าไปและยังมีระบบ RESET และ WATCH DOG จากชิพเบอร์ MAX 232 ซึ่งช่วยให้งานมีความแน่นอนยิ่งขึ้น ANT-31PJ มีความคล่องตัวในการพัฒนาสูง สามารถเลือกใช้ REM31 หรือ BASIC32 ซึ่งเป็นโปรแกรมที่ช่วยพัฒนาบอร์ดได้โดยผ่านเครื่องคอมพิวเตอร์

คุณสมบัติของบอร์ด SLAVE

CPU..... 80C31

CLOCK..... 11.0592 MHz

MOMORY.....U2 0/32 EPROM SOCKET

(2764,27128,27256)

0000-7FFFH (PMEM U3 8/8K (RAM #6264)ADDRESS

0000-1FFFH (DMEM)0R 8000-9FFFH(DMEM + PMEM)

PORT..... 24BIT8255USER PORT(26 PIN

CONNECTOR)

LCDPORT..... (DOT MATRIX ONLY)

SERIAL PORT..... (3 PIN CONNECTOR , # MAX232)

ONBOARD.....1 POWER LED12-PIN POWER

CONNECTOR

1RESET SWITCH (MAX1232)

1- 2-WAYS JUMPER (WATCH-DOGENABLE / DISABLE)

2 TEST POINT FOR CLIP (VCC & GND)

SIZE .....10.2 * 14.2 cm

รายละเอียดเกี่ยวกับการใช้งานของ SLAVE BOARD

หน่วยความจำ

การจัดหน่วยความจำบนบอร์ด ANT-31PJ แบ่งได้เป็น 2 SOCKET โดยสามารถใส่หน่วยความจำได้สูงสุด 40 Kbyte และสามารถเลือกเบอร์ต่าง ๆ ได้ดังนี้

SOCKET	จำนวน Kbyte	เบอร์หน่วยความจำ	แนวทางการใช้
U2	8 TO 32 Kbyte	2764,27128,27256	-สำหรับโปรแกรม MONITOR
U3	8 Kbyte	6264	- สำหรับเก็บข้อมูล หรือ ใช้ทดสอบโปรแกรม โดยสามารถเลือกตำแหน่ง ADDRESS เป็น 0000H หรือ 8000H

การเลือก ADDRESS ของ U3 นี้ สามารถเลือกใช้ได้ 2 ช่วงดังนี้

1. ADDRESS 0000-1FFFH โดยจะมองหน่วยความจำเป็น DATA MEMORY เท่านั้น ซึ่งสามารถใช้ งานทั่วไป หรือจะใช้กับ BASIC32 ก็ได้ (BASIC32 คือ EPROM ที่มี BASIC อยู่ในตัวให้สามารถ ใช้ภาษาเบสิกพัฒนาโปรแกรมได้)

2. ADDRESS 8000h-9FFFH โดยจะมองหน่วยความจำเป็นทั้ง DATA MEMORY & PROGRAM MEMORY ซึ่งใช้งานได้ทั่วไปยกเว้นที่สามารถใช้กับ REM31 ได้ด้วย โดยจะทำให้พัฒนาโปรแกรมสะดวกยิ่งขึ้น (REM31 คือ EPROM ที่มีโปรแกรมมอนิเตอร์เพื่อช่วยให้พัฒนาบนบอร์ดผ่านทางเครื่องคอมพิวเตอร์ได้)

การเลือกเบอร์หน่วยความจำ หรือการเลือกตำแหน่งของ ADDRESS สามารถทำได้ด้วยตัว JUMPER ที่อยู่ด้านล่างของ SOCKETโดยกำหนดตัว JUMPER โดยตรงกับตำแหน่งที่ต้องการ

การขยายอินพุทเอ้าท์พุทพอร์ท

SLAVE BOARD มีหน่วย INPUT & OUTPUT อยู่ 1 ตัว โดยผู้ใช้สามารถใช้งานได้อย่างมีอิสระ ซึ่งใช้เบอร์ 8255 ยอดนิยม ขั้วต่อเป็นแบบ 26 PIN-HEADER มาตราฐาน โดยจะสามารถใช้บอร์ดขยายต่าง ๆ ในตระกูล EX-SERIES ได้ทันที หรือจะต่อออกไปประยุกต์ใช้

งานต่าง ๆ ได้ตามต้องการตำแหน่ง ADDRESS ของ 8255 จะจัดอยู่ในส่วน DATA MEMORY โดยใช้ ADDRESS ดังต่อไปนี้

PORT	ADDRESS
PORT A	E000H
PORT B	E001H
PORT C	E002H
CONTROL PORT	E003H

การใช้งาน 8255 นี้ ต้องเซ็ท CONTROL PORT ก่อนเสมอ ซึ่งจะเป็นการกำหนดคุณสมบัติต่าง ๆ ของ PORT A,B,C ซึ่งสามารถดูจาก DATA SHEET ของ 8255 ได้อย่างไรก็ตามนิยมใช้ MODE 0 เป็นส่วนใหญ่ โดยกำหนดจะกำหนดให้ PORT A,B,C เป็น INPUT หรือ OUTPUTได้ตามต้องการซึ่งค่าที่ต้องกำหนดในCONTROL PORT จะดูได้จากตาราง Control port ในตารางในรูปที่ 50

วงจร RESET

บน SLAVE BOARDจะใช้ชิพเบอร์ MAX232โดยจะใช้สำหรับการ RESET ระบบรวมทั้งมีวงจร WATCH DOGให้พร้อมด้วย ซึ่งการ RESET แบบนี้จะได้ผลดีกว่าวงจรแบบ RC ทั่ว ๆ ไป โดยจะทำการ RESET ทั้งช่วง POWER UP และ POWER DOWN ซึ่งแน่ใจได้ว่าระบบการทำงานของไมโครคอนโทรลเลอร์จะไม่เกิดสภาพรวนจากการ RESET อย่างแน่นอน การ RESET จาก MAX 232 นี้จะตั้งระดับไว้ที่ 10%ของ VCC ซึ่งหมายถึงว่าระบบ RESETจะทำงานทันทีถ้า VCC ต่ำกว่า 4.5 V

วงจร WATCH DOG

ส่วนของวงจร WATCH DOG ก็จะยิ่งช่วยให้ระบบเกิดความเสถียรภาพมากยิ่งขึ้น โดยสามารถเลือก ENABLE หรือ DISABLE ได้ด้วย JUMPER ซึ่งอยู่ทางด้านขวาของชิพเบอร์ 74HCT138ในกรณีตั้งไว้ที่ DISABLE ระบบจะต่อขาRSTของ MAX1232 เข้ากับขาA0 ของ CPU ซึ่งหมายถึงว่าจะมีสัญญาณเข้ามาตลอดเวลา แต่ถ้าตั้งไว้ที่ ENABLE ระบบจะต่อเข้ากับสัญญาณ DECODE ในตำแหน่ง A000H เพราะฉะนั้นโปรแกรมที่เขียนขึ้นต้องส่งสัญญาณมากระตุ้นขานี้เสมอภายในเวลา 1.2 sec (การส่งสัญญาณมากระตุ้นทำได้โดยใช้คำสั่ง MOVX และอ้าง ADDRESS ที่ A000H ก็เพียงพอแล้ว โดยไม่ต้องสนใจว่าจะเป็นการ READ หรือ WRITE ใน REGISTER ตัวไหนแต่อย่างใด) ในกรณีที่สัญญาณขาดหายไป ซึ่งอาจจะเกิดจากการ HANG ของระบบ ตัว MAX1232 จะทำการ RESET ระบบทันทีโดยจะทำให้ระบบกลับมาทำงานได้ตาม

การ SET JUMPER

JUMPER EA สำหรับเลือกใช้หน่วยความจำโปรแกรม (PROGRAM MAMORY) ตำแหน่งแอดเตรสเริ่มต้น OOOOH เป็น INT . (INTERNAL) หรือ EXT. (EXTERNAL)

JUMPER RESE สำหรับเลือกสัญญาณรีเซท CPU จากวงจร RC หรือ MAX691

JUMPER U2 SIZE สำหรับเลือกขนาดหน่วยความจำโปรแกรม U2 (EPROM) เป็น 8,16kbyte (2764,27128) หรือ 32kbyte (27256)

JUMPER U3 SIZE สำหรับเลือกขนาดหน่วยความจำข้อมูล U3 (RAM) เป็น 8kbyte (6264) หรือ 32KByte (62256)

JUMPER BACKUP สำหรับเลือก NO/OFF การสำรองข้อมูล (MAX691) ของ U3

JUMPER WD สำหรับเลือก EN. (ENABLE) / DIS.(DISABLE) วงจร. WATCHDOG (MAX691)

การ SET JUMPER เลือกหน่วยความจำ

ไมโครคอนโทรลเลอร์ตระกูล 8031 (8032) สามารถต่อกับหน่วยความจำภายนอกได้ถึง 128 Kbyte โดยแบ่งออกเป็นสองส่วนคือ 64 Kbyteเป็นหน่วยความจำโปรแกรม (PROGRAM MEMORY)และอีก 64 Kbyte เป็นหน่วยความจำข้อมูล (DATA MEMORY) ซึ่งหน่วยความจำทั้งสองส่วนนี้มีตำแหน่งแอดเตรสที่ 0000H - FFFFH เหมือนกัน แต่จะถูกแยกออกจากจากกันด้วยสัญญาณควบคุมที่ต่างกัน โดยสัญญาณ PSEN Ⓡ ใช้ควบคุมการอ่านและเขียนหน่วยความจำโปรแกรม(EPROM)สัญญาณ RDⓇและWR Ⓡใช้ควบคุมการอ่านและเขียนหน่วยความจำข้อมูลและพอรท์อินพุท / เอาท์พุท และสำหรับการอ่านหน่วยความจำโปรแกรมและข้อมูล (PROGRAM ANF DATA MEMORY)ใช้สัญญาณ GETⓇซึ่งสัญญาณนี้ได้จากการ AND สัญญาณ PSEN Ⓡ และ RD Ⓡ หน่วยความจำส่วนนี้สามารถใช้ได้กับERROM หรือ RAM

บทที่ 6

ซอฟท์แวร์ของระบบ (SOFTWARE SYSTEM)

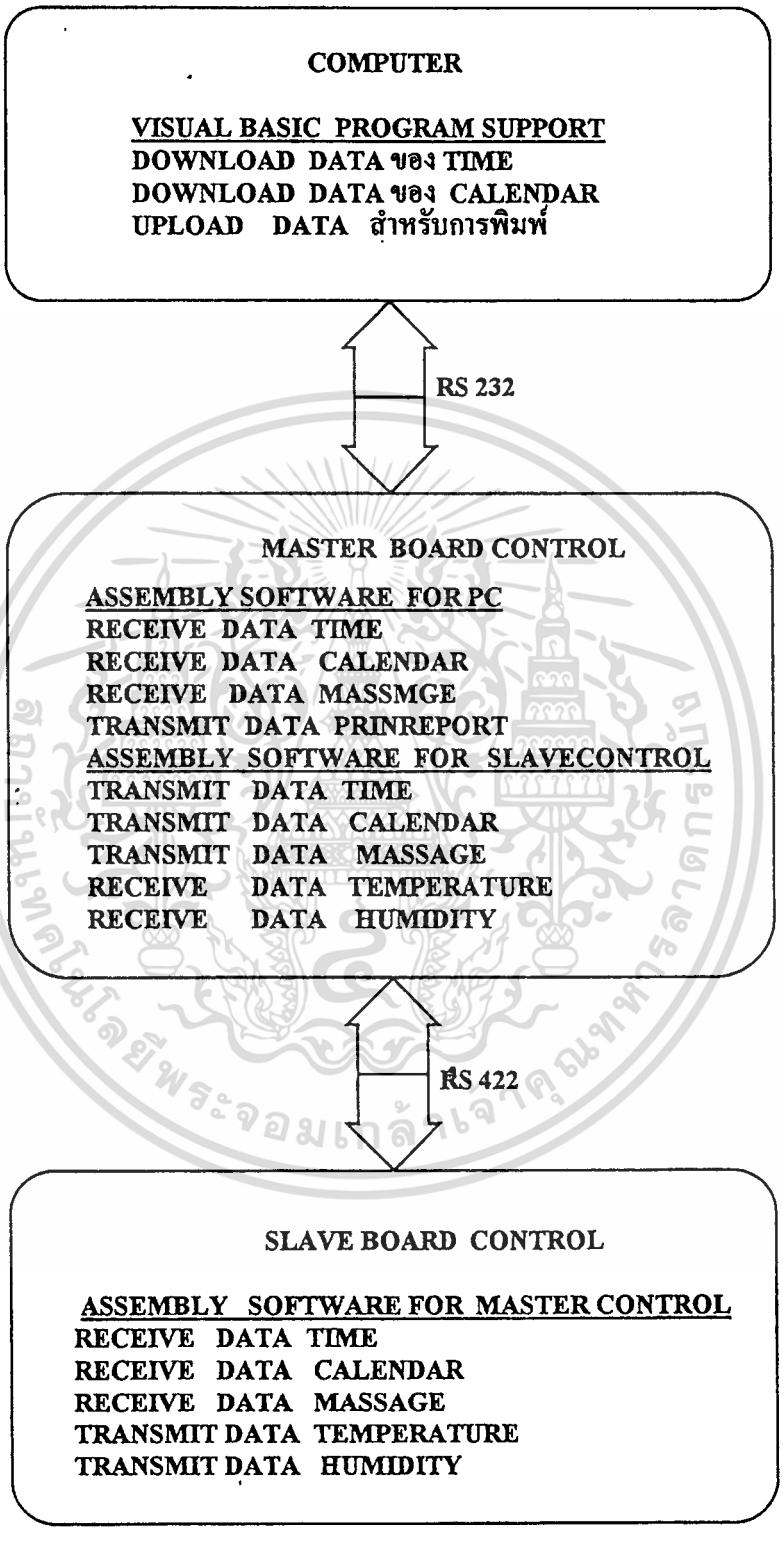
Software ของระบบจะแบ่งออกเป็น 3 ส่วนตามที่แสดงในรูปที่ 56 ซึ่งได้อธิบายรายละเอียดของแต่ละส่วนไว้แล้วในช่วงต่อไป การอธิบาย Software ของแต่ส่วนจะนำเสนอโดยการใช้ Flowchart และตามด้วย Soure Code ของแต่ละส่วน โดยสามารถแบ่งได้ดังนี้

การพัฒนาโปรแกรมด้วยวิชวลเบสิค (**Application By Visual Basic**)

Software ในส่วนนี้จะการพัฒนาจาก Command ของ Visual Basic โดยใช้ทำหน้าที่หลักในการ Down-Load ข้อมูลสำหรับการตั้งค่าเวลา (hour:min:sec)ตั้งค่าปฏิทิน(day:date:mount:year) นอกจากนั้นยังทำหน้าที่ Up-Load ข้อมูลสำหรับพิมพ์ ค่าการเปลี่ยนแปลงของ Temperature และ Humidity ณ วันเวลาที่แตกต่างกัน โดยโปรแกรมในส่วนนี้จะใช้ทำการติดต่อกับโปรแกรมใน Master Control Board ซึ่งรายละเอียดของโปรแกรมจะได้อธิบายในลำดับต่อไป

การพัฒนาโปรแกรมด้วยภาษาเอสเซมบลี้

Software ในส่วนนี้จะทำหน้าที่ในการควบคุม Master Control Board และ Slave Control Board โดยจะแยกกันโดยอิสระ แต่จะทำงานร่วมกันในการแลกเปลี่ยนข้อมูลของ ซึ่งกันและกัน โปรแกรม Assembly ใน Master Control Board จะทำหน้าในการอ่านค่า,ตั้งค่า,ส่งค่า RTC ไปให้กับ Slave Control Board เพื่อนำไปแสดงผลต่อไป ควบคุมการแสดงผล ของ Seven segment และ Key Board บน Master Control Board ส่วนโปรแกรม Assembly ใน Slave Control Board ทำหน้าที่ ในการควบคุมการแสดงผลที่ LED DOT METRIX , อ่านอุณหภูมิจากตัว Sensor ,อ่านค่าความชื้นจากวงจร Analog To Digital และส่งกลับไปยัง Master Control Board ส่วนการอธิบายรายละเอียดจะแสดงโดยใช้ Flow chart และ Source Code ในลำดับถัดไป

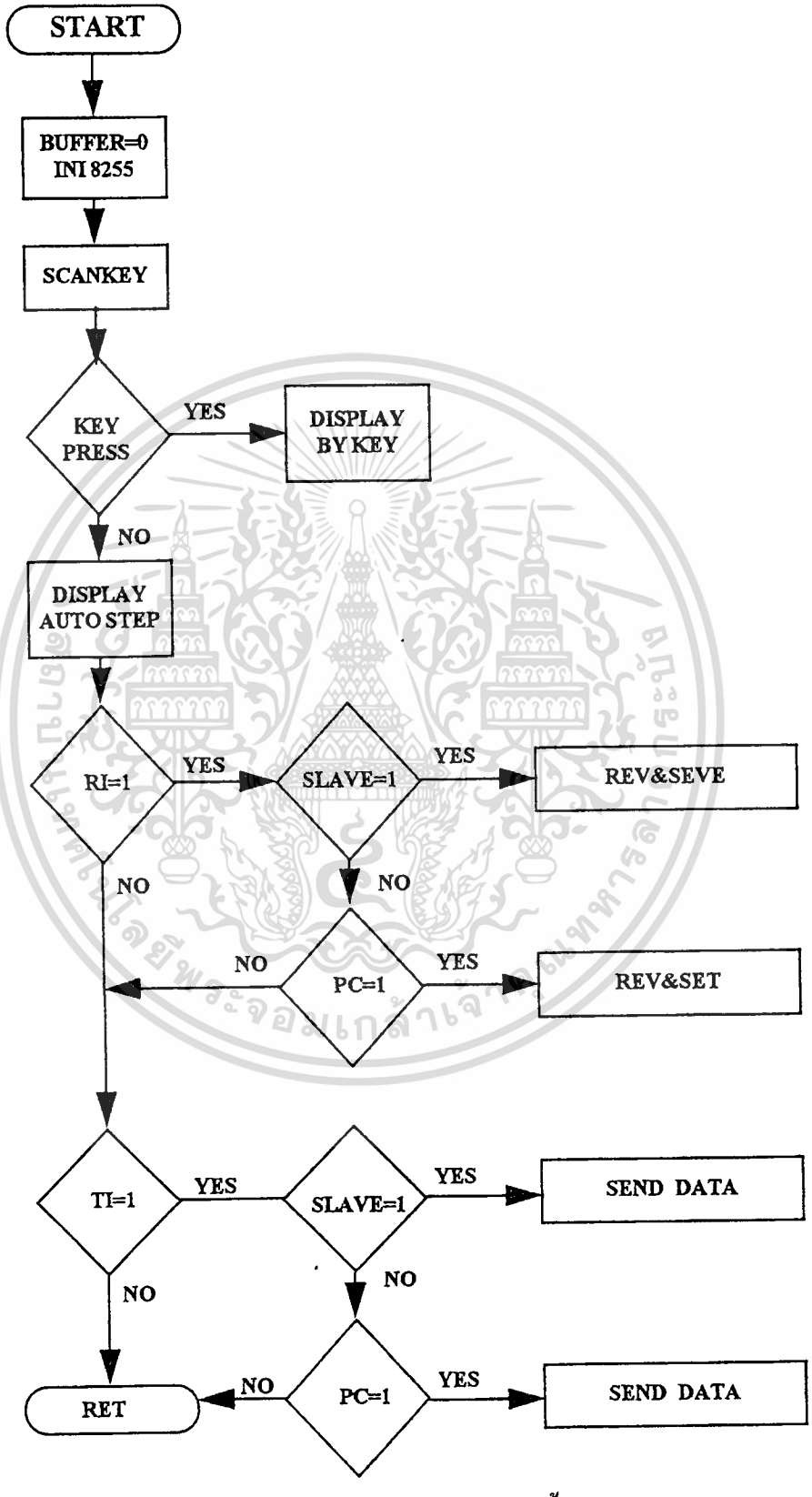


รูปที่ 56 แสดงโมดูลของ Application ของ Software ของระบบ

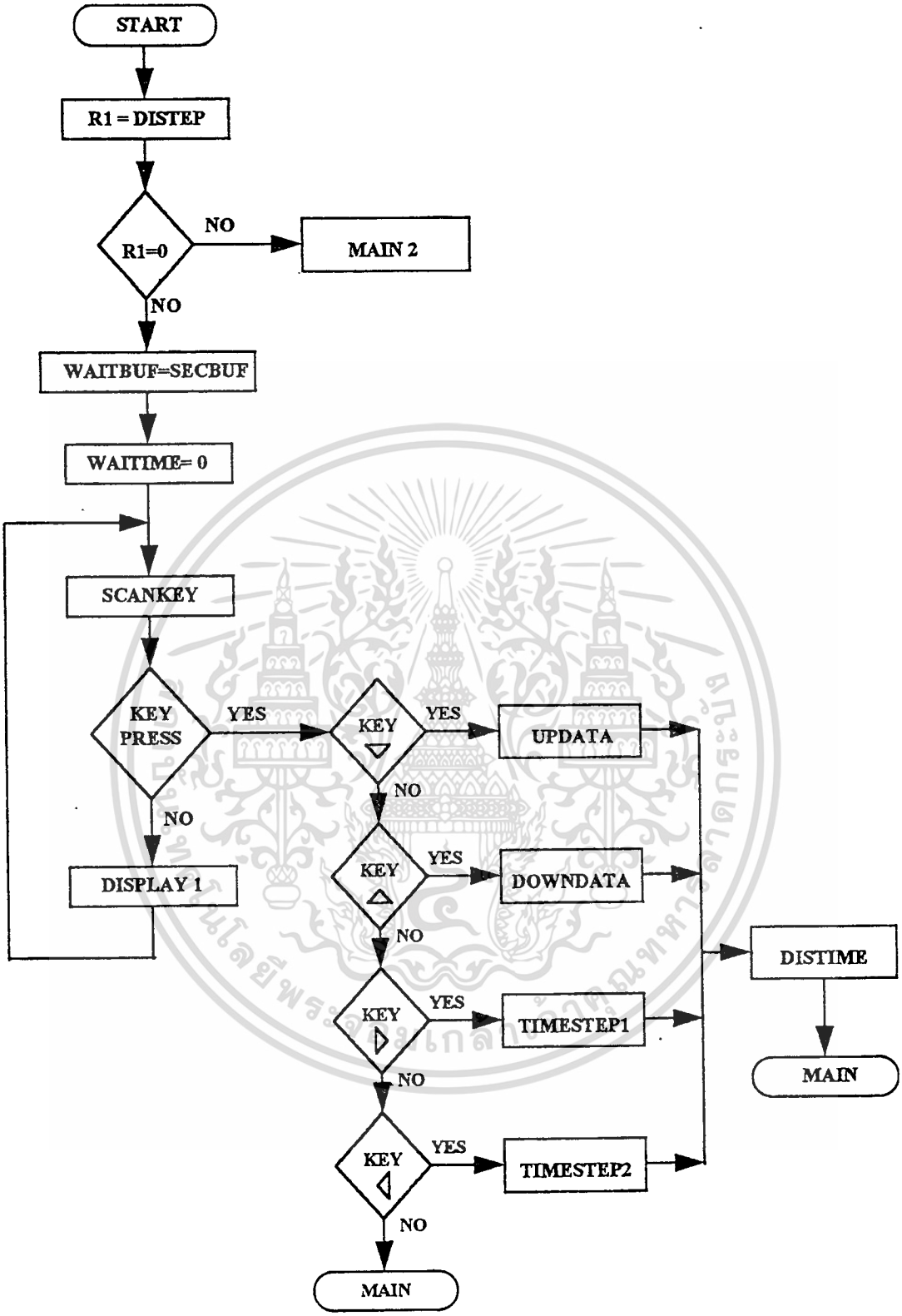
ซอฟท์แวร์เอสเซมบลี้ บน Master Control Board

เนื่องจากโครงสร้างทาง Hard Ware ของ Master Control Board จะใช้ CPU ตระกูล MCS-51 การควบคุมโดยโปรแกรมภาษา Assembly จึงมีลักษณะที่แตกต่างไปจากโปรแกรมภาษา Assembly ของ Z80 และ 8088โดยเฉพาะในเรื่องการใช้ Register แบบพิเศษ (SFR)

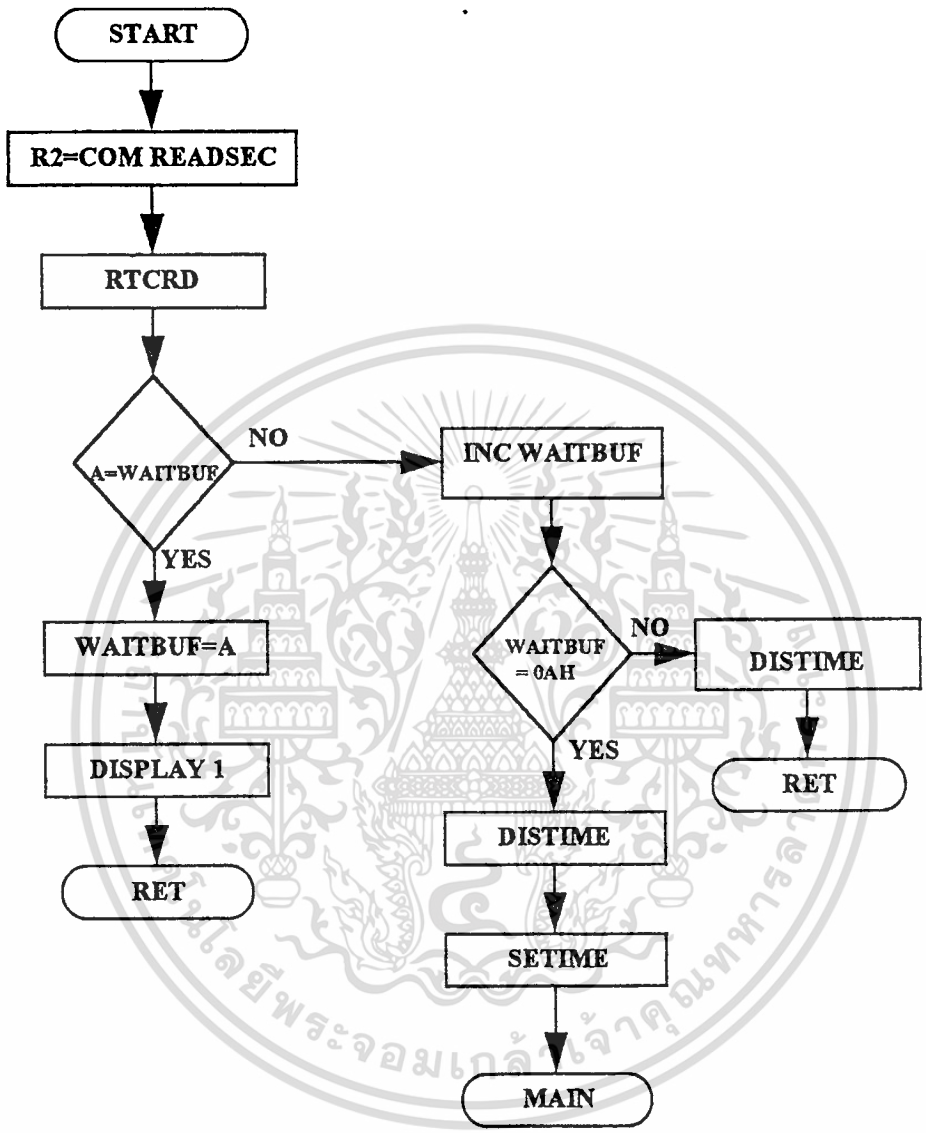
การควบคุมการทำงานทั้งหมดบน Master Control Board พอจะอธิบายจาก Flow chart ของระบบตามรูปที่ 57 ดังนี้คือ การกำหนดค่าเริ่มต้นในการทำงานของ Port 8255 เพื่อกำหนดให้Port A,Port B,และPort C บน(Upper)เป็น Output Port ,Port C ล่าง(Lower)จุดประสงค์หลักในก็เพื่อต้องการแสดงผลที่ 7 Segment และรับค่าการกด Key และจะเริ่มเข้าสู่ Monitor หลักของโปรแกรม ซึ่งจะทำการ Scan Key หากมีการกด Key ก็จะตรวจสอบว่าเป็น Key ที่กดนั้นคือ Key อะไร หากการตรวจสอบว่า Key ที่กดนั้น เป็นKey เป็น Keyup หรือ Keydown ลำดับต่อมาก็จะก็จะเป็นการเปลี่ยน Step การแสดงผล โดยจะเริ่มการแสดงผล Time, Calendar,temperature Humidity และคำว่า Presen ค้างไว้ ซึ่งการทำงานของแต่ส่วนได้แสดงไว้ใน Flowchart การกด Keyup หรือ Keydown ก็จะแสดงผลกลับทิศทางกันของทั้งการแสดงผลทั้ง 5 โปรแกรม ส่วนการกด Keylelf หรือ Keydown ในขณะที่มีการแสดงผล Time และ Calendar โปรแกรมก็จะเข้าสู่การ SET ค่าของ RTC การกด Keylelf หรือ Keydown ในขณะที่มีการแสดงผล ในโปรมแกรมอื่นๆ จะไม่มีการเปลี่ยนแปลง การ SET ค่า RTC 0 จะมีการกำหนดขอบเขตค่าที่เป็นจริงของแต่ละส่วน เช่น ค่า Hour จะมีค่า ตั้งแต่ 00-23 เป็นต้น เมื่อการ SET จบลงโปรแกรมก็จะกลับเข้าสู่ Main Monitor (โปรแกรมการแสดงผลที่ 7 Segment)ส่วนประกอบของSUB Program ได้แสดงบางส่วนไว้ใน Flowchart ตั้งแต่รูปที่ 58 ถึง 63 และ SoureCode ซึ่งได้แสดงในลำดับถัดไป



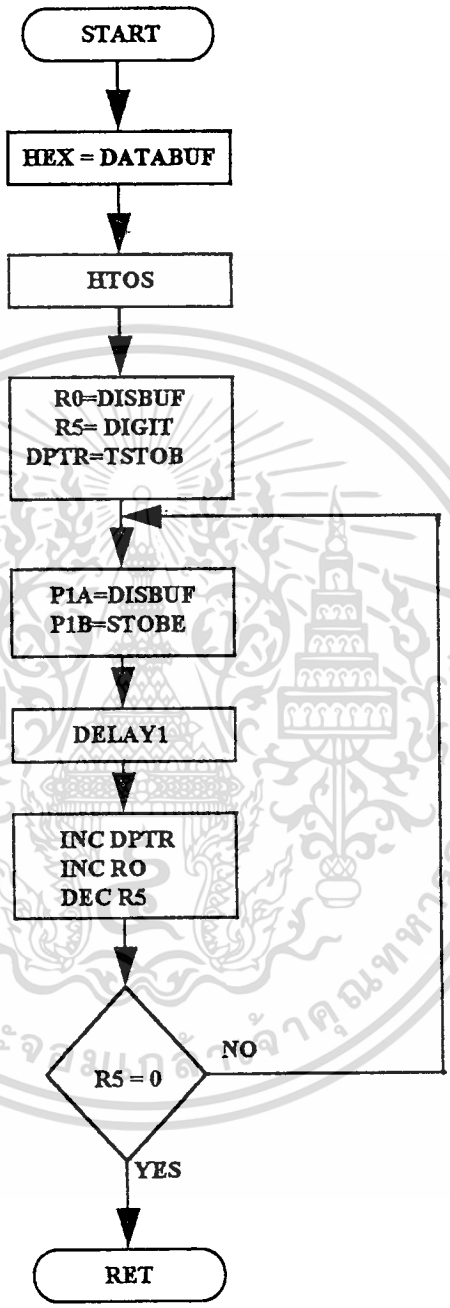
รูปที่ 57 บล็อคแสดงโปรแกรมหลักของระบบทั้งหมด



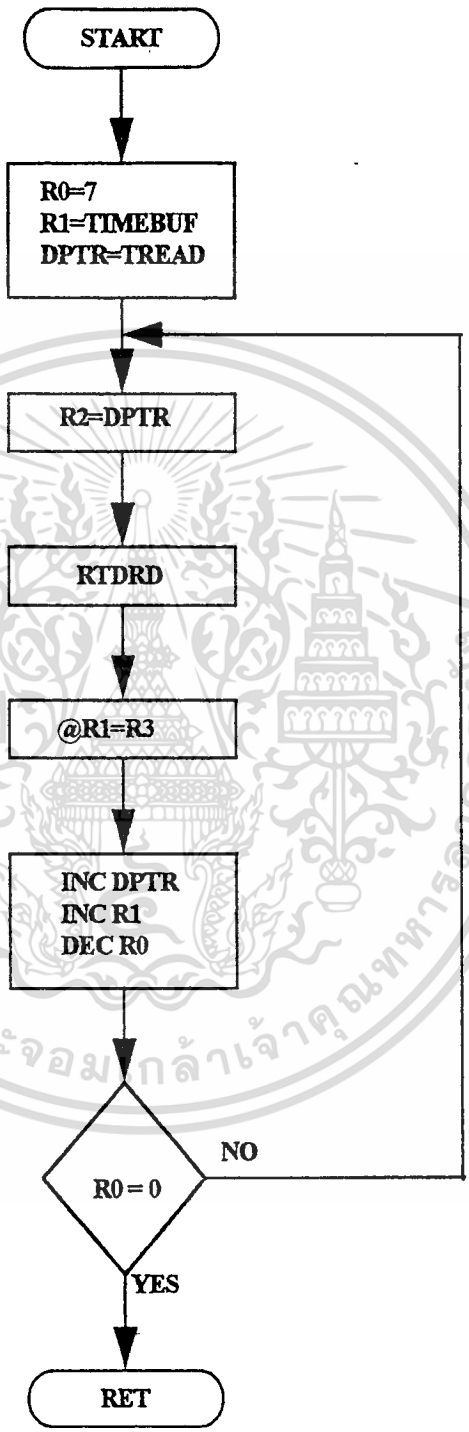
รูปที่ 58 แสดงโปรแกรมย่อยการตั้งเวลาด้วย คีย์บอร์ด



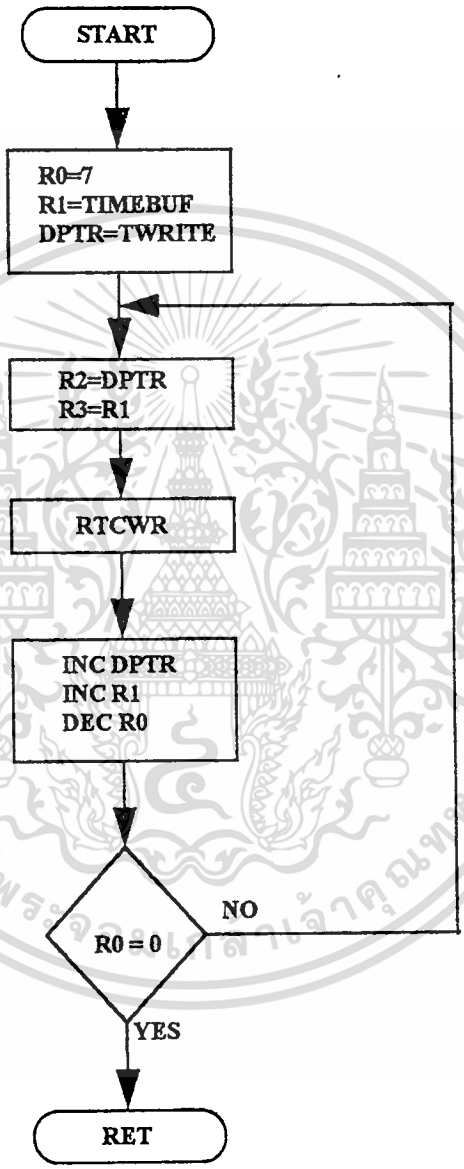
รูปที่ 59 แสดงโปรแกรมย่อยหน่วงเวลาสำหรับตั้งเวลาด้วย



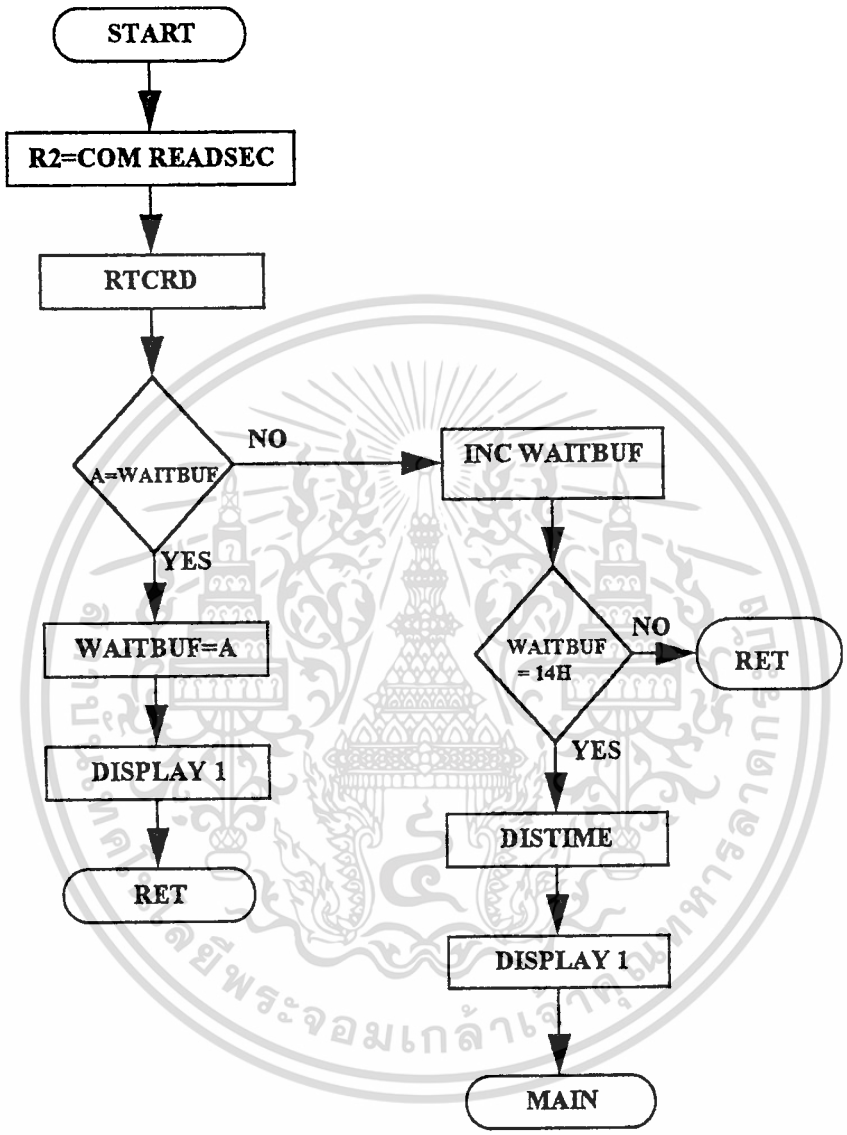
รูปที่ 60 แสดงโปรแกรมย่อยการแสดงผล 7 segment



รูปที่ 61 แสดงโปรแกรมย่อยการอ่าน RTC



รูปที่ 62 แสดงโปรแกรมย่อยการเขียน RTC



รูปที่ 63 แสดงโปรแกรมย่อยการหน่วงเวลาการแสดงผล

ซอฟท์แวร์บนบอร์ดควบคุมรอง (Slave Control Board)

ในส่วนของโปรแกรมของ Slave Control Board นั้นแบ่งออกเป็น 2 ส่วนคือ

1. โปรแกรมหลัก
2. โปรแกรมย่อย ซึ่งจะแบ่งออกได้เป็น
   - โปรแกรมย่อยของการแสดงผล Dot Matrix
   - โปรแกรมย่อยของการตรวจจับอุณหภูมิ
   - โปรแกรมย่อยของการตรวจจับความชื้น

โปรแกรมหลัก

การทำงานของโปรแกรมหลักคือการรับและส่งข้อมูลระหว่าง Slave Control Board และ Master Control Board ดัง Flow Chart ของโปรแกรมหลัก รูปที่ 64.

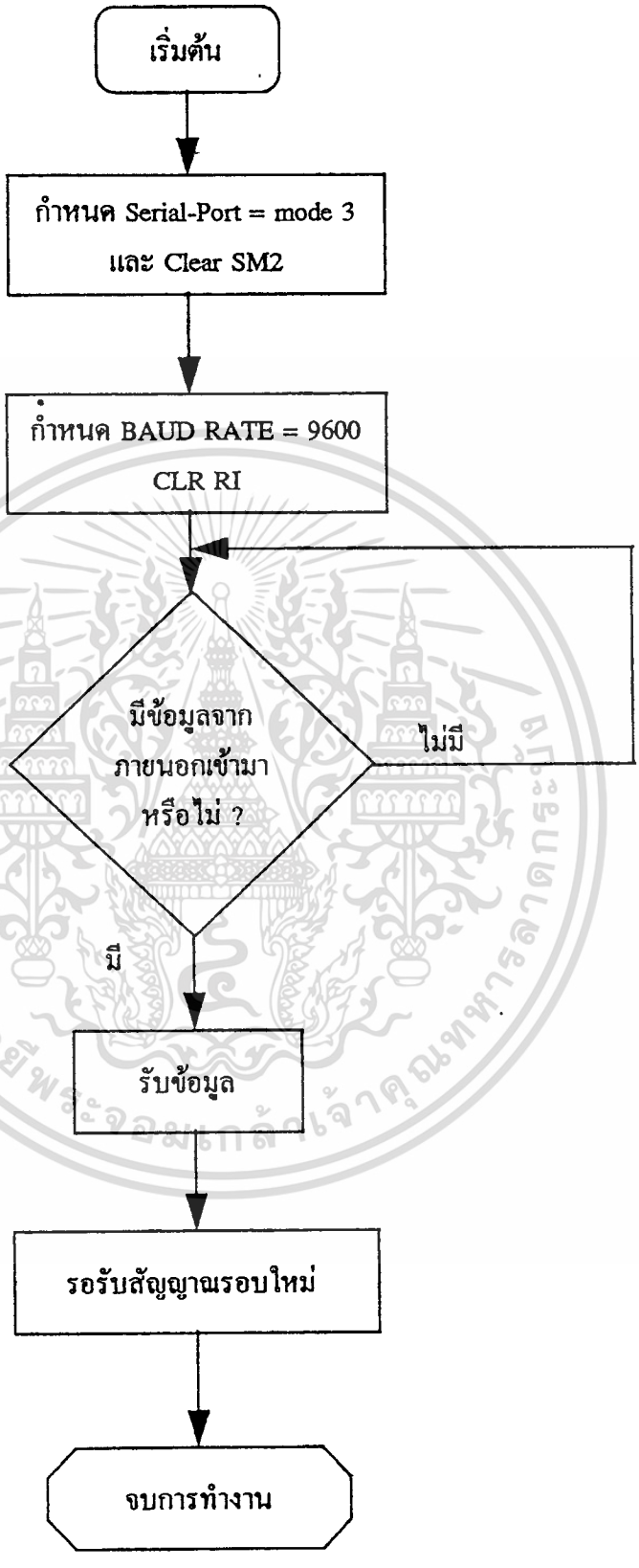
ทางด้านการรับนั้นเป็นหน้าที่หลักของ Slave Control Board เพราะเมื่อรับข้อมูลมาแล้วจะต้องแสดงผลออกทาง Display Board ข้อมูลที่รับเข้ามาจะมีด้วยกัน 3 ส่วนด้วยกันคือ

1. ข้อมูลของเวลา
2. ข้อมูลของ วัน เดือน ปี
3. ข้อมูลของข้อความ

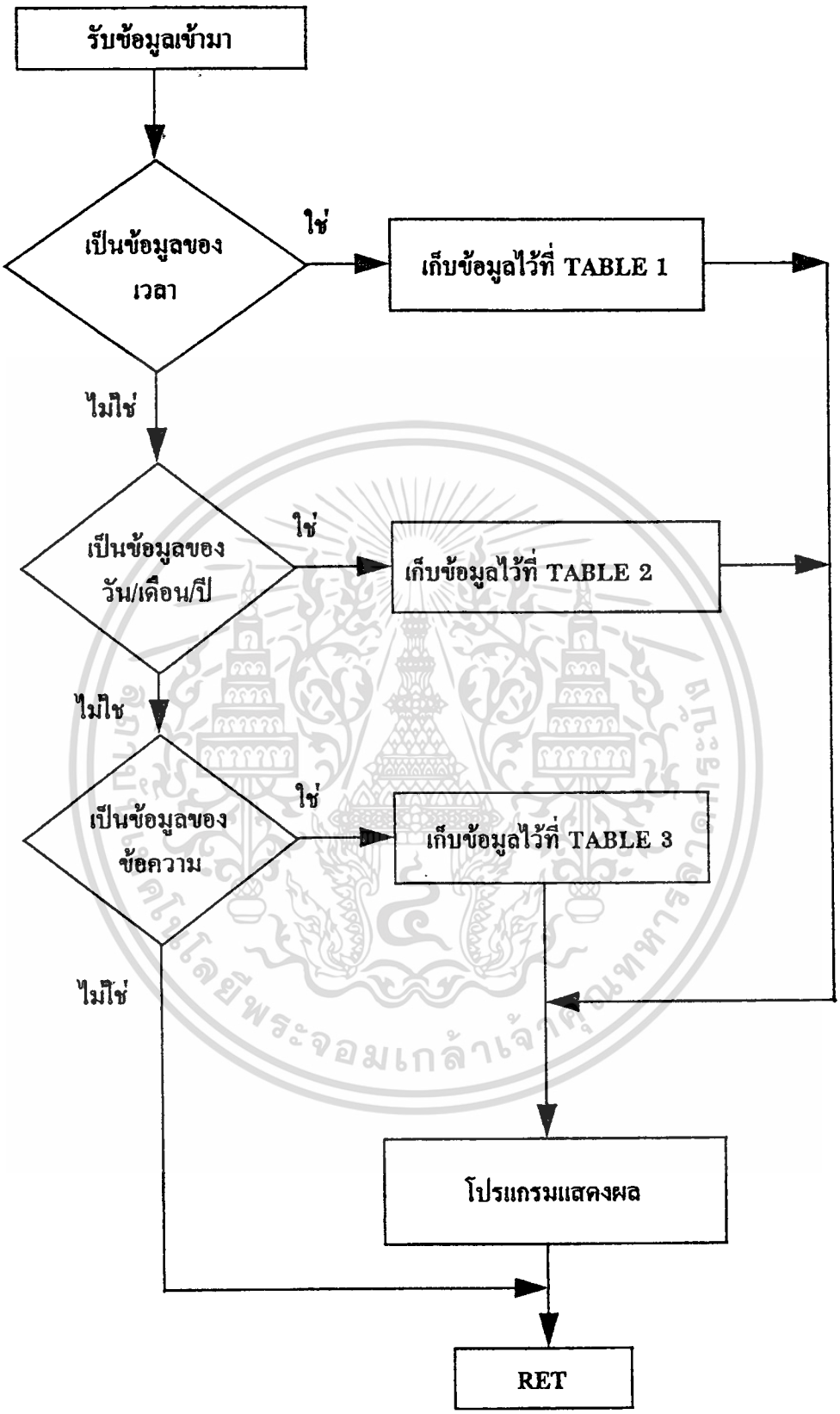
ดังนั้นโปรแกรมต้องทำการตรวจสอบข้อมูลที่เข้ามาว่าเป็น ข้อมูลของอะไรขึ้นอยู่กับเงื่อนไขการส่งโดยจะตรวจสอบตัวเครื่องหมายกำกับหน้าข้อมูลนั้นๆ เมื่อรับข้อมูลเข้ามาแล้วตรวจสอบการสิ้นสุดของข้อมูลเมื่อพบว่าข้อมูลแต่ละชุดเสร็จสิ้น การส่งหลังจากนั้นจะนำข้อมูลไปเก็บไว้ในตารางต่างๆกันซึ่งดูได้จาก Flow Chart ของโปรแกรมย่อยการรับข้อมูล รูปที่ 66. กำหนดให้แสดงผลออกทาง Display Board ทันทีหรือต้องการเก็บไว้ในตารางก่อนก็ได้

ในส่วนโปรแกรมหลักจะกระทำเช่นนี้ตลอด คือ รับข้อมูลของเวลา และ แสดงผลทุกๆ 1 วินาทีและจะตรวจจับอุณหภูมิและความชื้นทุกๆ 2 ชั่วโมง โดยจะนำข้อมูลไปเก็บไว้ในตารางแล้วให้แสดงผล ค่าอุณหภูมิ หรือ ความชื้นที่วัดได้ และทุกๆครั้งของการตรวจวัดจะต้องทำการส่งข้อมูลกลับไปให้ Master Control Board ทุกครั้ง ถ้าจะทำการส่งจะกระโดดไปที่โปรแกรมย่อยการส่งข้อมูล หรือถ้าต้องการแสดงผลข้อมูลออกทาง Display Board โปรแกรมหลักก็จะสั่งให้กระโดดไปยังโปรแแกรมย่อยการแสดงผล

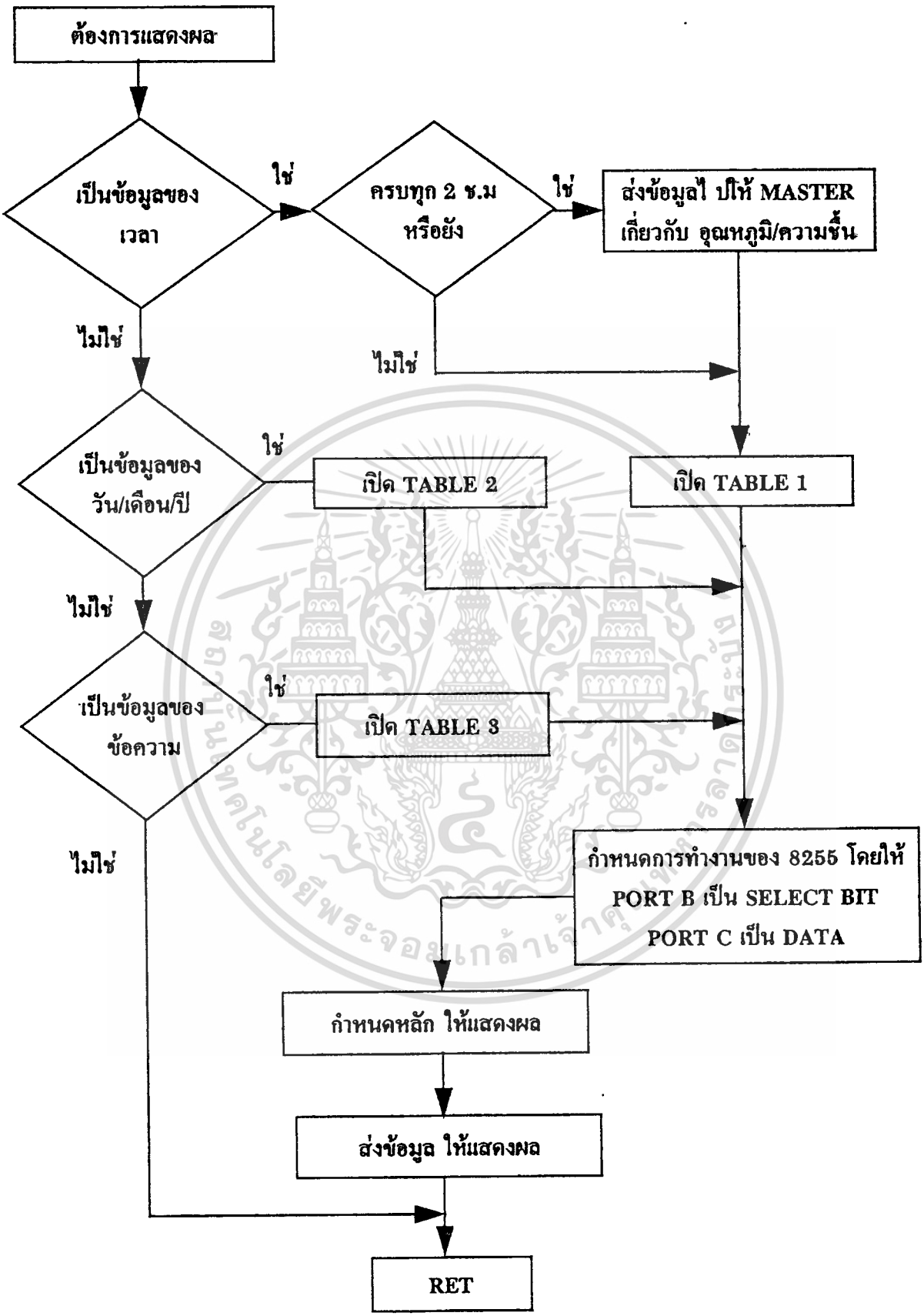
.



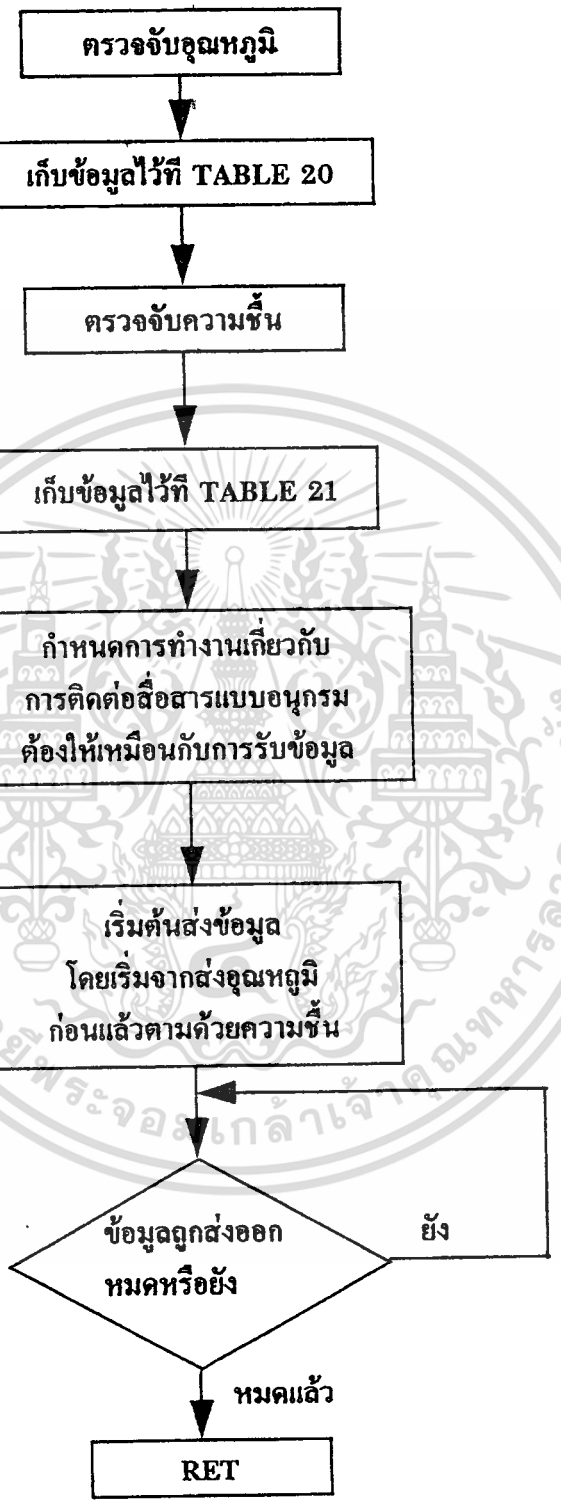
รูปที่ 64 บล็อคแสดงขั้นตอนการทำงานของโปรแกรมหลักของ Slave-Board



รูปที่ 65 บล็อคแสดงขั้นตอนการทำงานของโปรแกรมย่อยการรับข้อมูลของ Slave-Board



รูปที่ 66 บล็อคแสดงขั้นตอนการทำงานของโปรแกรมย่อยการแสดงผลของ Slave-Board



รูปที่ 67 บล็อคแสดงขั้นตอนการทำงานของโปรแกรมย่อยการส่งข้อมูลของ Slave-Board

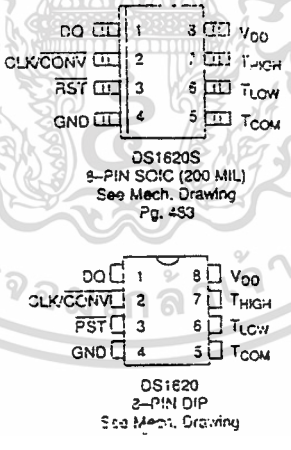
โปรแกรมการแสดงผลข้อมูล

การแสดงผลข้อมูลออกทาง DISPLAY BOARD จะแสดงตัวอักษรเป็นรหัส ASCII โดยต้องปู้อนข้อมูลเข้าในส่วนของตัว DISPLAY BOARD ต้องCOMPLEMENT ข้อมูลที่จะแสดงด้วยทุกๆค่าของข้อมูล ทั้งนี้เพราะตัวของ DISPLAY BOARD จะทำงานเมื่อ ACTIVE LOW

จากรูป FLOW CHART ในรูปที่ 66 โปรแกรมย่อยการแสดงผลจะต้องดึงข้อมูลออกมาจาก ตาราง ที่เก็บค่าข้อมูลต่างๆ ออกมา เมื่อรับรู้ว่าต้องการแสดงผลข้อมูลอะไร ก็จะไปกำหนดการทำงานของ 8255 โดยจะกำหนดให้ PORT B และ PORT C เป็น OUTPUT PORT การกำหนดให้ PORT A,B,C เป็น INPUT หรือ OUTPUT โดยจะต้องเข้าไปกำหนดใน CONTROL PORT ซึ่ง ค่าต่าง ๆในการกำหนดจะดูได้จาก ตาราง ในโปรมย่อยการแสดงผลนี้จะกำหนดให้ PORT B เป็น SELECT BIT (0 - 8 BIT) และ PORT C เป็น DATA ของตัวอักษร

โปรแกรมย่อยการตรวจวัดอุณหภูมิ

การวัดอุณหภูมิจะใช้ IC # DS1620 (Digital Thermostat and Thermostat) ซึ่งรูปร่างลักษณะของ IC ตัวนี้ รวมทั้งขาต่างๆ ดูจากรูปที่ 68 ซึ่งในส่วนรายละเอียดตัวโปรแกรมของการตรวจวัดอุณหภูมิจะดูได้จากโปรแกรมแอสเซมบลีของ SLAVE CONTROL BOARD ดู FLOW CHART การทำงานได้จากรูปที่ 67



รูปที่ 68 แสดงขาของ ชิพ ตัววัดอุณหภูมิ

ค่าอุณหภูมิที่วัดได้ต้องทำการเปลียนค่าจาก DIGITAL OUTPUT (BINARY) ไปเป็น DIGITAL OUTPUT (HEXIMAL) แล้ทำการเปลี่ยนให้เป็น DECIMAL OUTPUT อีกที แล้วนำค่านี้ไปเก็บในตารางที่เรากำหนดไว้ในตอนต้นของโปรแกรมหลัก ถ้าต้องการแสดงผลออกทาง DISPLAY BOARD ก็จะนำค่า DECIMAL OUTPUT ออกมาแสดง ถ้าต้องการส่งข้อมูลไปให้ MASTER BOARD ก็จะนำค่านี้ส่งออกเช่นกัน

คุณสมบัติของ DS 1620

1. ไม่ต้องต่ออุปกรณ์การวัดอุณหภูมิภายนอกเพราะในตัวมันเองมี SENSOR อุณหภูมิติดอยู่ใต้ผิวด้านบนของ IC เอง ซึ่งมีการทำงานตาม block diagram ตามรูปที่ 69

2. วัดอุณหภูมิได้ตั้งแต่ -55 องศา C ถึง+125 องศาC โดยเพิ่มขึ้นทีละ 0.5 องศา หรือ -67 องศา F ถึง 257 องศา F โดยเพิ่มขึ้นทีละ 0.9 องศา F

3. ค่าที่วัดได้เป็น DIGITAL 9 BIT ซึ่งสามารถดูการแปลงค่าจาก เลขฐานสอง ไปเป็นเลขฐานสิบ และ เลขฐานสิบหก จากตารางรูปที่ 69.

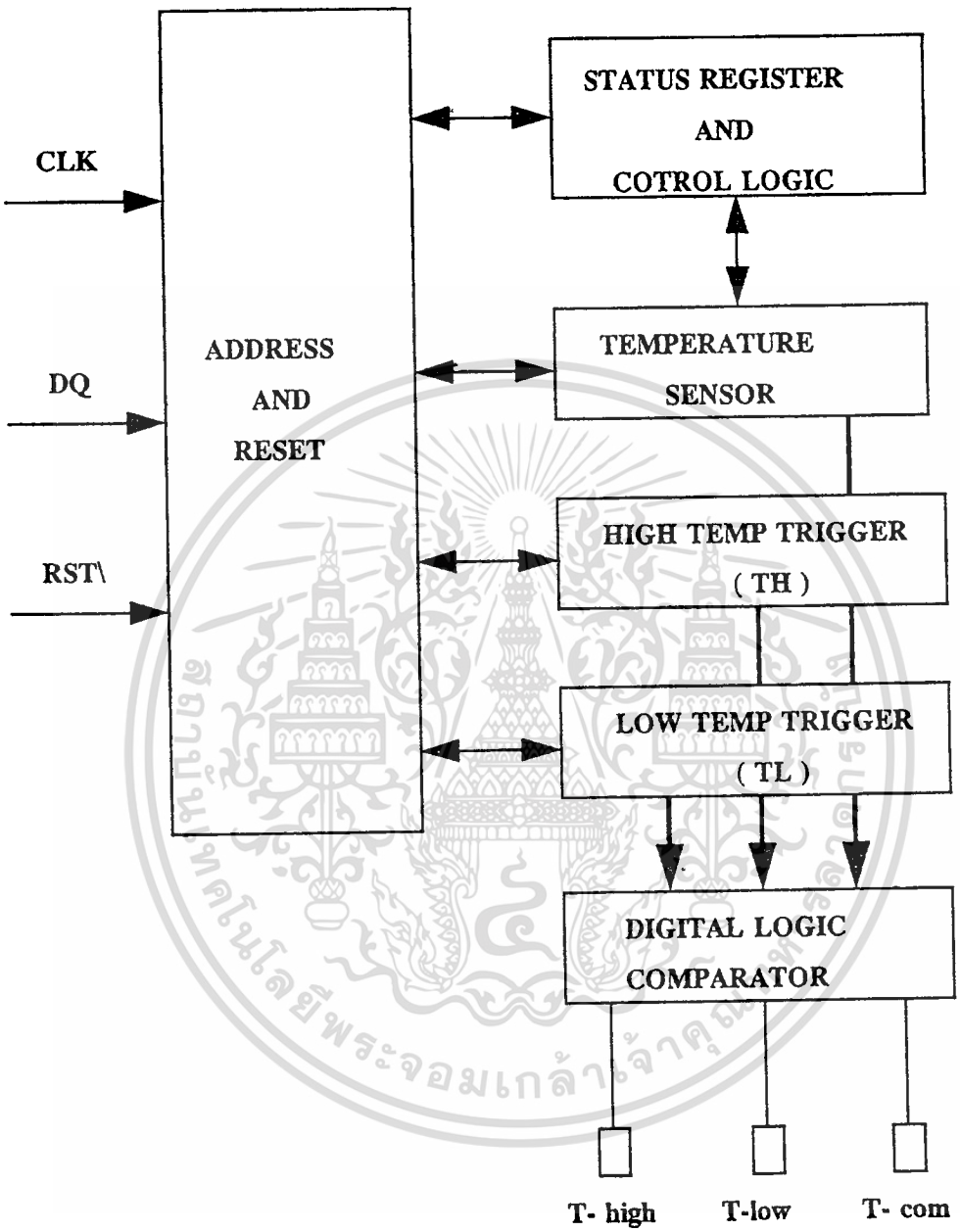
4. ใช้เวลาในการ CONVERT เพียง 1 วินาที

5. การติดต่อกับ CPU จะใช้เพียง 3 ขาคือ

- CLK/CONV เป็นขาสัญญาณนาฬิกาควบคุมการอ่าน
- RST เป็นขา reset
- DQ เป็นขา data output

TEMP	DIGITAL OUTPUT BINARY CODE	DIGTAL OUTPUT HEXIMAL CODE
+ 125 องศา C	0 1111 1010	00FAH
+ 25 องศา C	0 0011 0010	0032H
+ 0.5 องศา C	0 0000 0001	0001H
0 องศา C	0 0000 0000	0000H
-0.5 องศา C	1 1111 1111	01FFH
- 25 องศา C	1 1100 1110	01CEH

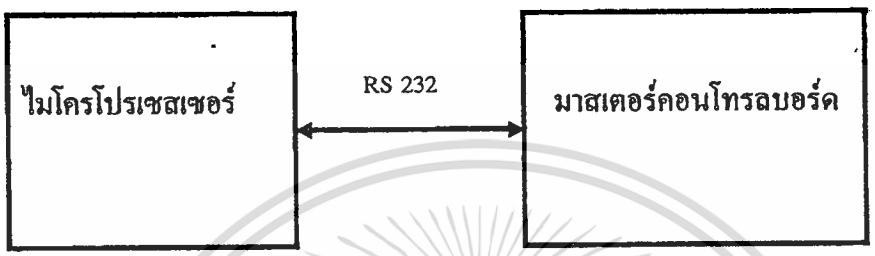
รูปที่ 69 ตารางตัวอย่างการ CONVERT ค่าอุณหภูมิ



รูปที่ 70 บล็อกไดอะแกรมการทำงานของ DS1620

การพัฒนาโปรแกรมด้วย VISUAL BASIC

สำหรับการติดต่อสื่อสาร ระหว่างไมโครคอมพิวเตอร์ กับ มาสเตอร์คอนโทรลบอร์ดนั้น เราจะเชื่อมต่อกันด้วย มาตรฐานการสื่อสาร RS-232 ซึ่งเป็นการสื่อสารแบบอนุกรม เพื่อใช้ในการอัปโหลด ดาวน์โหลด ระหว่าง ไมโครคอมพิวเตอร์ กับ มาสเตอร์คอนโทรลบอร์ด ดังรูปที่ 71



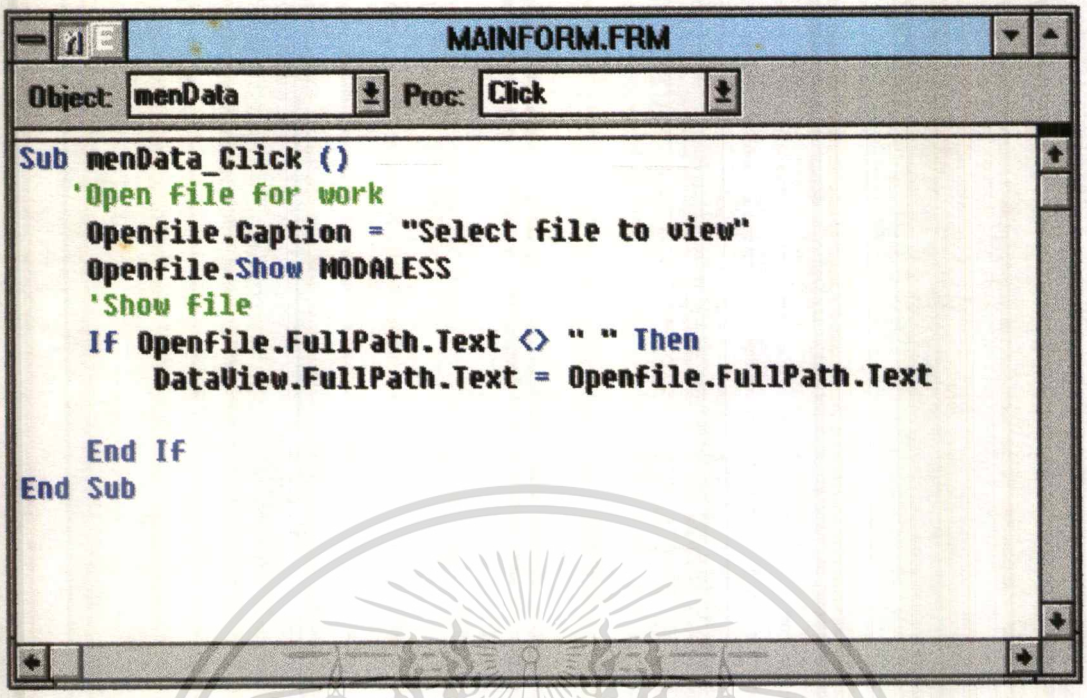
รูปที่ 71 การเชื่อมต่อไมโครคอมพิวเตอร์ กับ มาสเตอร์คอนโทรลบอร์ด

จากรูปที่ 71การที่ไมโครคอมพิวเตอร์ กับ มาสเตอร์คอนโทรลบอร์ด จะติดต่อกันได้นั้น เราจะต้องเขียนโปรแกรมการใช้งานขึ้นมารองรับทั้งสองส่วน สำหรับในส่วนโปรแกรมการใช้งานบนเครื่องคอมพิวเตอร์นั้น เราใช้โปรแกรม Visual Basic เป็นเครื่องมือในการพัฒนาโปรแกรมสำหรับการใช้งานดังลักษณะที่จะกล่าวถึงพอเป็นสังเขป

หลักการโปรแกรมของ Visual Basic

ในการเขียนโปรแกรมโดยทั่วไป โปรแกรมเมอร์จะต้องกำหนดการไหลของโปรแกรมตั้งแต่ต้นจนจบโปรแกรมได้ ไม่ว่าโปรแกรมจะกระโดดไปทำงานที่โปรแกรมย่อยใดอย่างไร ตอนสุดท้ายจะต้องกลับมาโปรแกรมหลักแล้วจบที่จุดนั้น ด้วยเหตุผลนี้โปรแกรมเมอร์จะต้องมีทางเลือกออปชันย่อยของผู้ใช้ทุกออปชัน จึงทำให้การเขียนโปรแกรมเกิดข้อผิดพลาดขึ้นเสมอที่เรียกว่า "Bug" จนกล่าวกันโดยทั่วไปว่า "ไม่มีโปรแกรมใดไม่มีบัก"

แต่การเขียนโปรแกรมของ Visusl Basic ใช้ภาพและการมองเห็น ที่เรียกว่า "Vsual" ทำให้มีข้อผิดพลาดน้อยลงโดย Visual Basic จะเก็บออปเจกต์ต่างๆไว้ในส่วนที่เรียกว่า "ฟอร์ม" (Form) โดยออปเจกต์เหล่านี้จะถูกกำหนดให้ทำงานตามเหตุการณ์หรือ อีเวนท์ (Event) ที่กำหนด ซึ่งอาจเป็นการ Click , DbClock เป็นต้น เหตุการณ์อื่นที่ไม่ระบุไว้ ก็จะไม่มีผลกับออปเจกต์นั้น ลักษณะโปรแกรมเช่นนี้เรียกว่า Event-Driven ซึ่ง Visual Basic ได้กำหนดส่วนหัวและส่วนหางเรียบร้อยแล้วดังรูปที่ 72



รูปที่ 72 การเขียนโปรแกรมแบบ Event-Driven

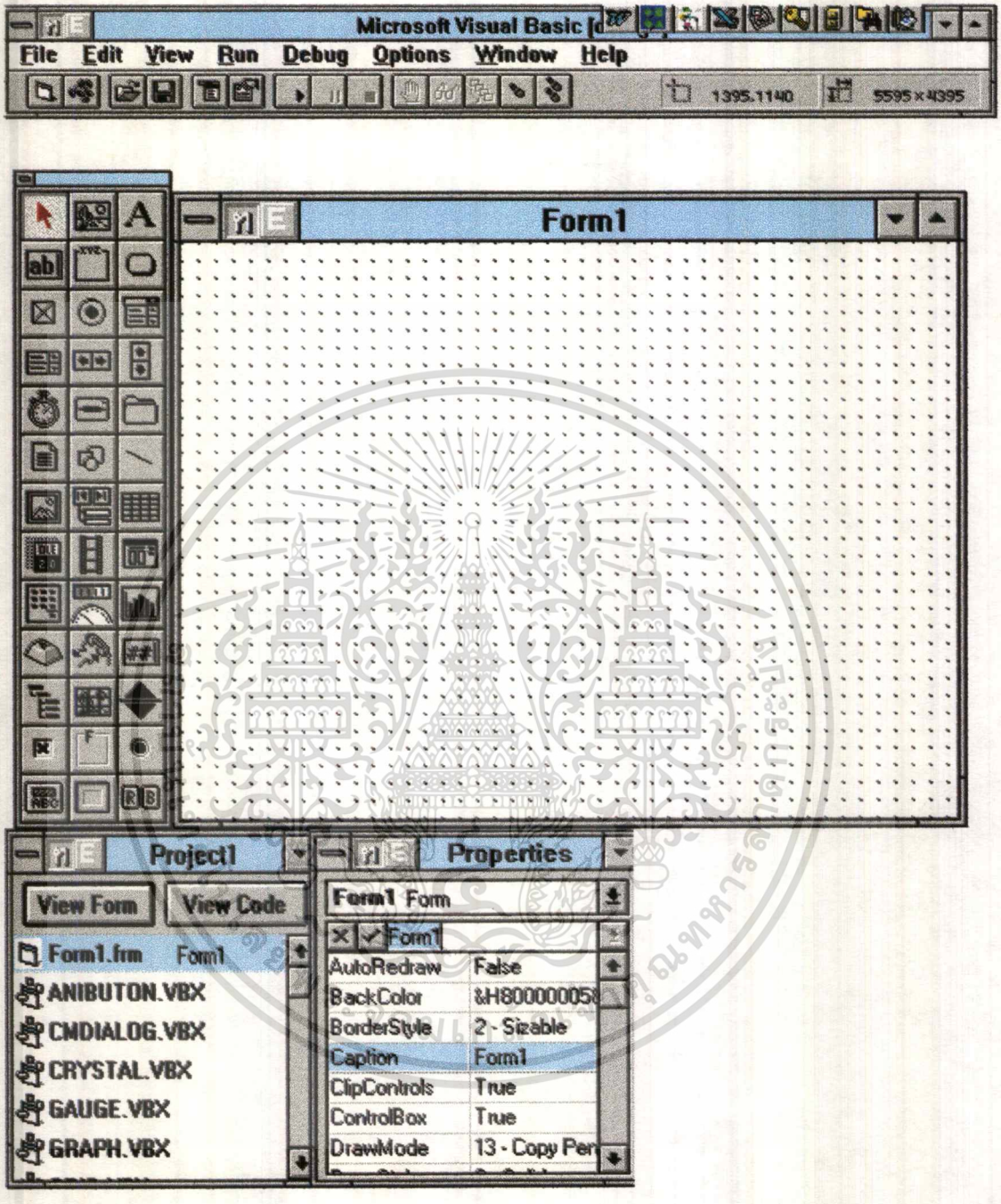
จากรูปที่ 72 จะเห็นว่าการเขียนโปรแกรมแบบ Event-Driven นี้ทำให้โปรแกรมเมอร์สามารถเขียน หรือพัฒนาโปรแกรมได้ง่ายขึ้น เพียงแค่ใส่คำสั่งให้โปรแกรมทำงานเท่านั้นข้อดีของ Visual Basic พอสรุปได้ดังนี้

- สามารถใช้งานได้สะดวก และง่ายในการพัฒนาโปรแกรม
- โปรแกรมที่พัฒนาด้วย Visual Basic จะทำงานบนวินโดว์
- สะดวก และง่ายสำหรับผู้ใช้งาน โปรแกรม ที่พัฒนาเสร็จแล้ว

นอกจากที่กล่าวมาแล้ว Visual Basic ยังมีคุณสมบัติอื่นๆ อีก เช่น ฟอร์น สไตล์ ขนาด สี ความกว้าง ความสูง เป็นต้น จึงทำให้ผู้พัฒนาโปรแกรม กำหนดคุณสมบัติได้อย่างงายดาย

วินโดว์ของโปรแกรม Visual Basic

หลังจากผู้ใช้เรียกใช้งาน Microsoft Visual Basic แล้วก็จะเข้าสู่การทำงานของ Visual Basic ซึ่งวินโดว์ของ Visual Basic ที่ใช้พัฒนาโปรแกรมเป็นดังรูปที่ 75 ซึ่งโปรแกรมที่พัฒนาบนVisual Basic for Windows จะมีลักษณะภาษาที่คล้ายคลึงกับภาษาอังกษฤที่เราใช้กันอยู่ทุกวันดังนั้นเราสามารถทำความเข้าใจกับโปรมได้ง่าย เหมาะสำหรับการพัฒนาโปรแกรม

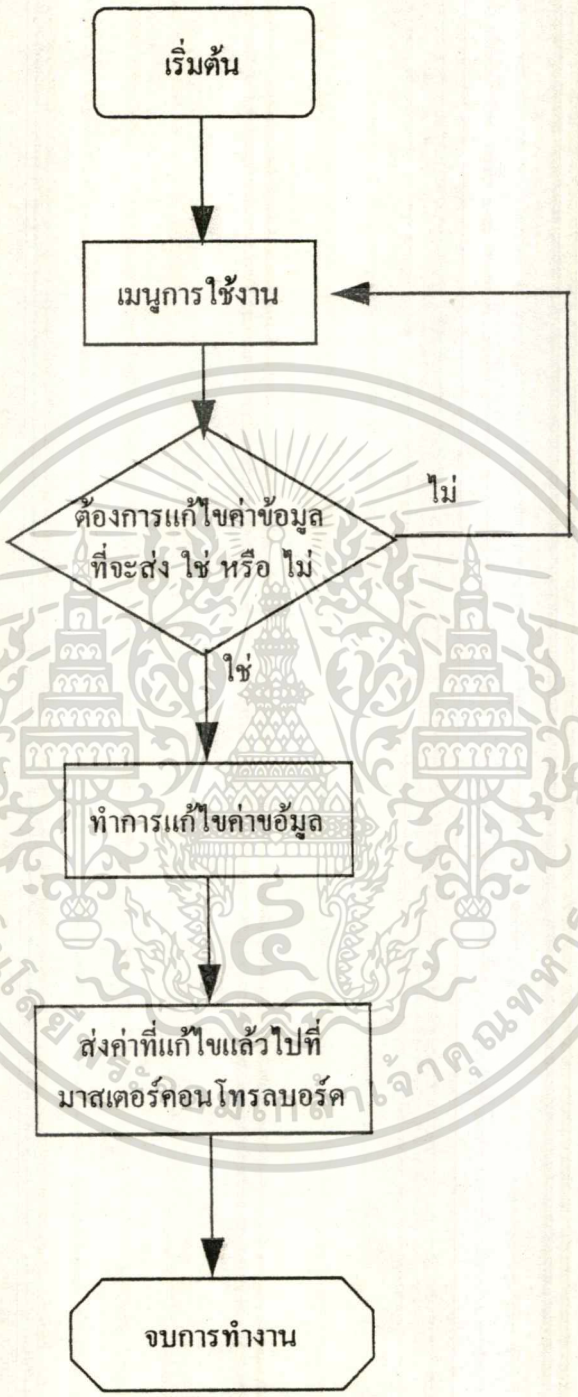


รูปที่ 73 แสดงฟอร์มของ Visual Basic for Windows

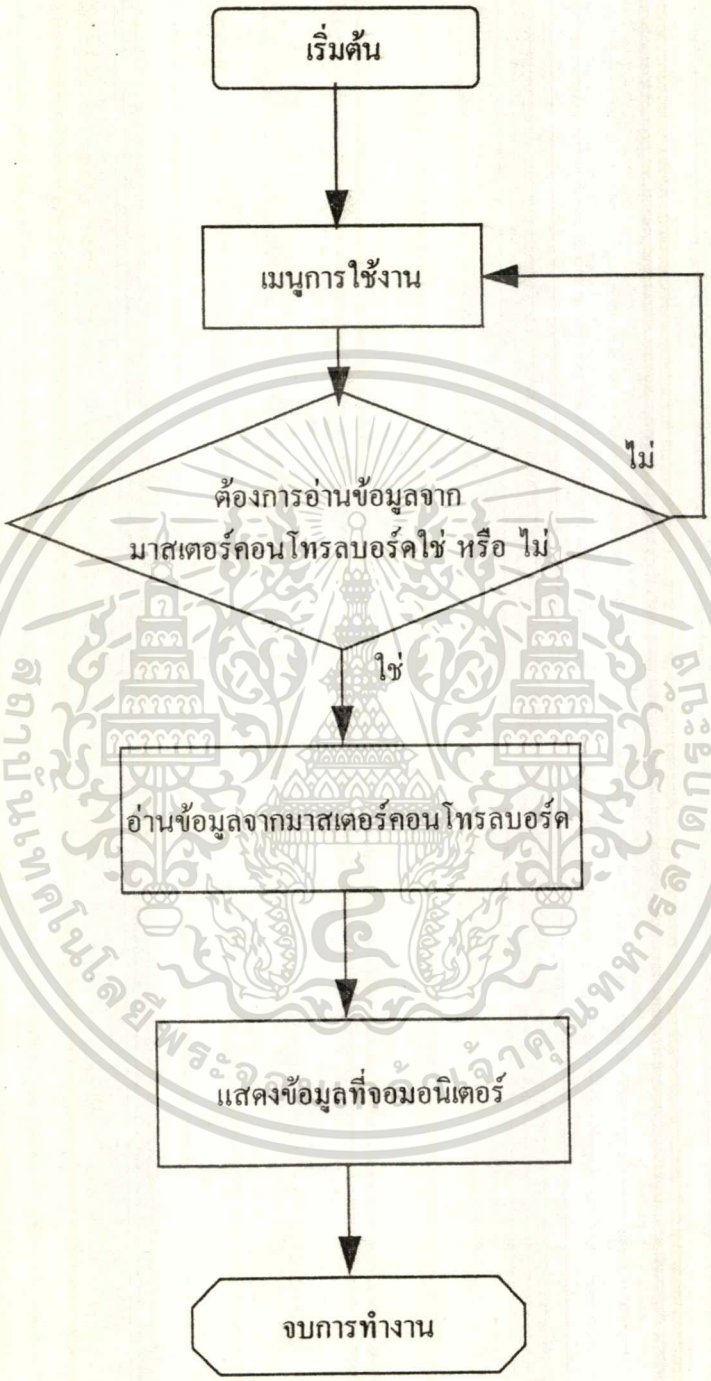
การพัฒนาโปรแกรมการใช้งาน Time and Calendar Remote Monitor

ในการพัฒนาโปรแกรมนั้นก่อนเราจะต้องทราบถึงจุดประสงค์ในการใช้งาน การทำงาน และ เงื่อนไขการทำงาน ของโปรแกรมที่เราจะทำการพัฒนาเสียก่อนเราถึงจะทำการพัฒนาโปรแกรมขึ้นมาได้ สำหรับโปรแกรมการใช้งานและการทำงานของCalendar and Clock Remote นั้นเราจะใช้สำหรับการเซ็ตค่า วัน เดือน ปี เวลา ส่งไปให้กับมาสเตอร์คอนโทรลบอร์ดใช้แสดง

ผลออกทางจอแสดงผล และใช้ในการอ่านค่า วัน เดือน ปี เวลา อุณหภูมิ ที่ส่งมาจากมาสเตอร์คอนโทรลบอร์ด ซึ่งมี Flow Chart การทำงานดังรูปที่ 74 และ รูปที่ 75

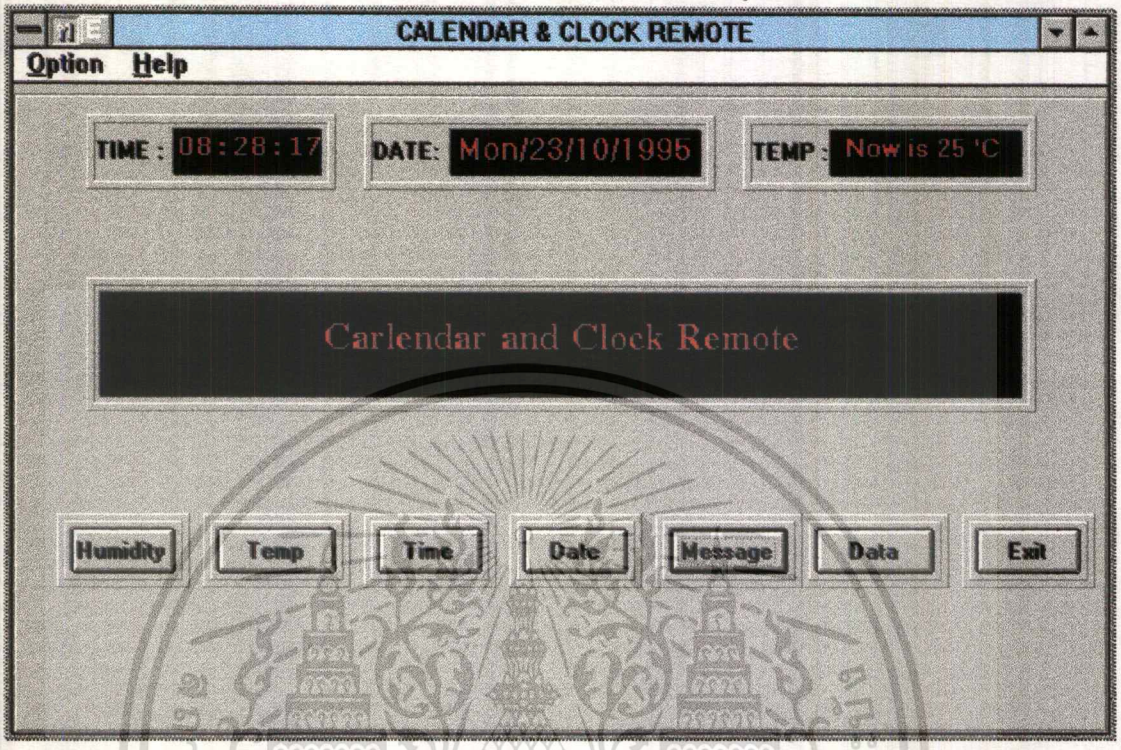


รูปที่ 74 การส่งข้อมูลจากไมโครคอมพิวเตอร์ ไปที่มาสเตอร์คอนโทรลบอร์ด



รูปที่ 75 การอ่านข้อมูลจากมาสเตอร์คอนโทรลบอร์ดเข้ามาที่ไมโครคอมพิวเตอร์

จาก Flow Chart ในรูปที่ 74 และ 75 เราสามารถนำเงื่อนไขของการทำงาน และจุดประสงค์ ของโครงงานมาทำการสร้าง และเขียนโปรแกรมการใช้งานด้วย Visual Basic ได้แบบฟอร์มการใช้งานของการทำงานบน เครื่องไมโครคอมพิวเตอร์ดังรูปที่ 76



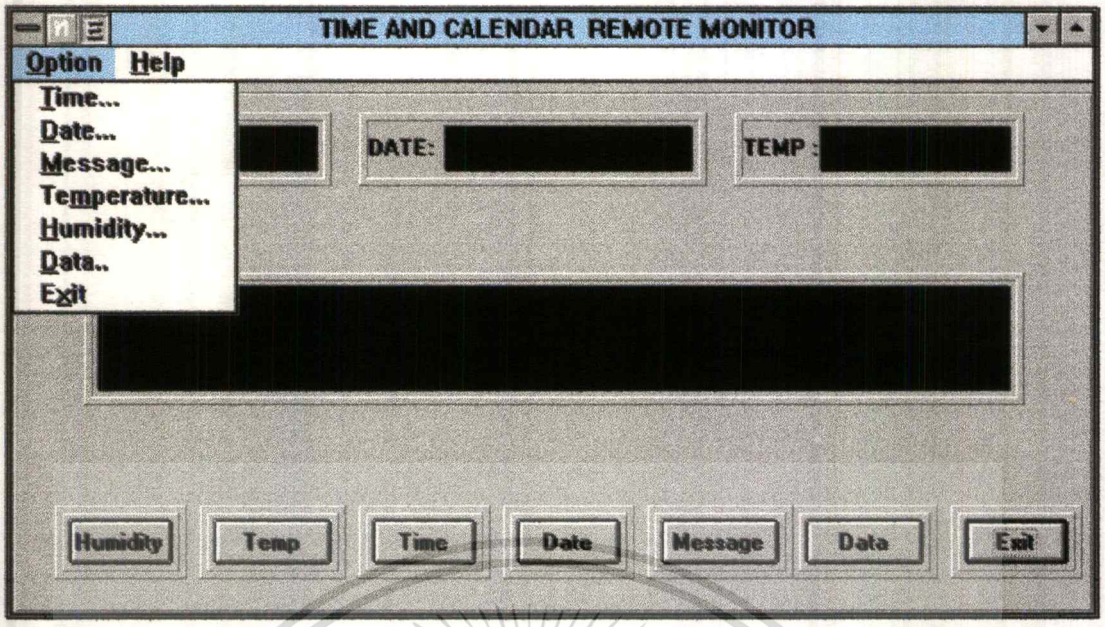
รูปที่ 76 แสดงการใช้งานจริงของโปรแกรมของระบบ

ขั้นตอนการใช้งานโปรแกรม

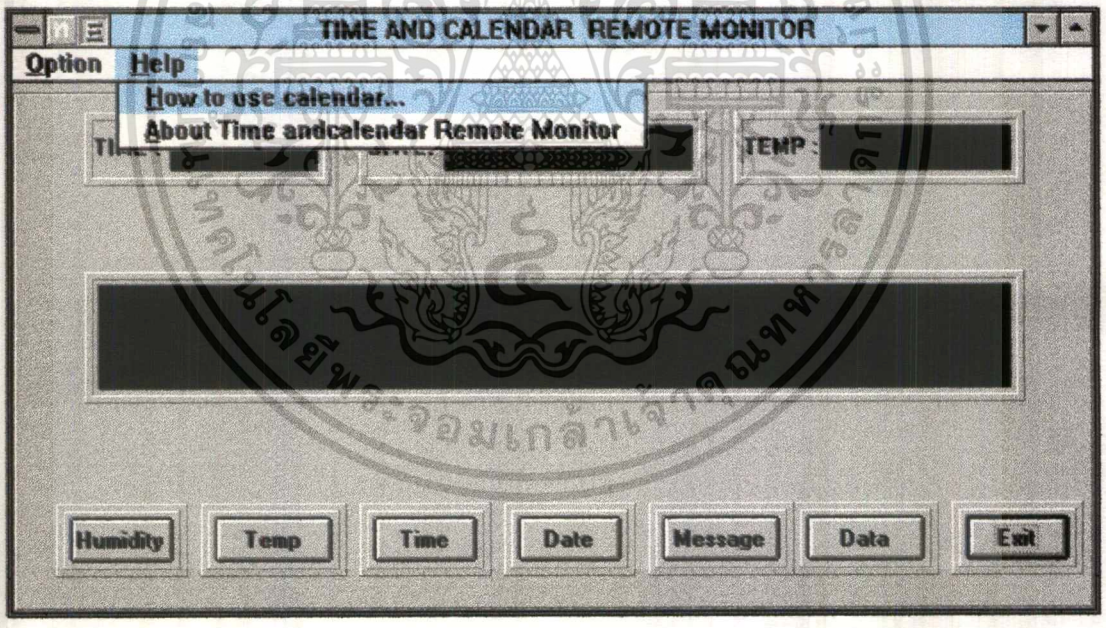
1. เชื่อมต่อมาสเตอร์คอนโทรลบอร์ดเข้าที่ Port Com2 ของ เครื่อง PC
2. ทำการรันโปรแกรม บนการทำงานของ Microsoft Windows
3. Clickที่เมนู RUN ของ Program Manager แล้วเขียน Command lineดังนี้คือ

   A:\Calendar\Calendar.EXE

หลังจากนั้นก็สามารถเลือกใช้งานโปรแกรมได้ตามคำสั่งดังรูปที่ 77(a) และรูปที่ 77(b)



รูปที่ 77(a) แสดงเมนู การใช้ งานของโปรแกรม



รูปที่ 77 (b) แสดงเมนูการใช้งานของโปรแกรม

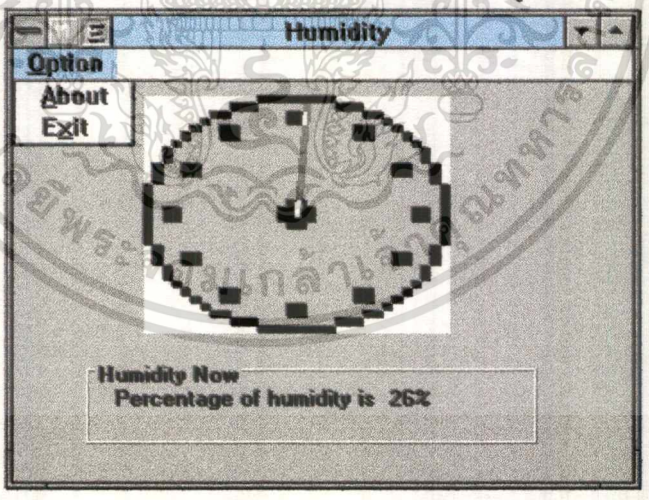
หน้าที่ของเมนูใช้งานต่างๆมีดังนี้คือ

1. เมนู Option เลือกเมนูการทำงานหลักของโปรแกรม
2. เมนู Time.. เลือกเพื่อการตั้งค่าเวลา
3. เมนู Date.. เลือกเพื่อการตั้งค่า วัน วันที่ เดือน และ ปี
4. เมนู Message.. เลือกเพื่อการส่ง Message ให้กับ Master Control Board
5. เมนู Temperature ขอดูค่าอุณหภูมิ
6. เมนู Humidity ขอดูค่าความชื้น
7. เมนู Data.. เลือกเพื่อการใช้งาน เกี่ยวกับ File ของข้อมูลที่ต้องการ
8. เมนู Exit เป็นการออกจากโปรแกรมการใช้งาน
9. เมนู Help ขอดูการใช้งานของโปรแกรม
10. เมนู About ขอดู รายละเอียดเกี่ยวกับผู้เขียนโปรแกรม

ผลการทดลองใช้งานโปรแกรม Time and Calendar Remote Monitor

การขอดูค่าความชื้น

@ Click ที่ปุ่ม Humidity หรือเลือกที่ Option menu แล้ว Click Humidity...จากนั้นฟอร์มแสดงค่าเปอร์เซนต์ความชื้นจะโชว์ขึ้นมาให้เราทราบค่าดังรูปที่ 78

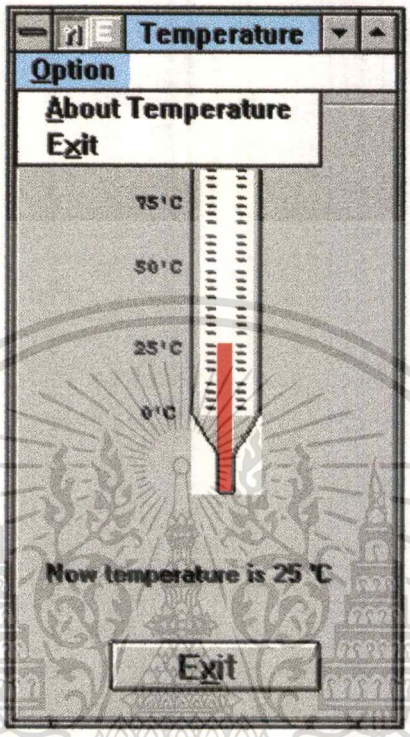


รูปที่ 78 ฟอร์มแสดงค่าความชื้น

- เมื่อจะออกจากฟอร์ม Humidity ก็ Click ที่ปุ่ม Exit จาก From

การขอดูค่าอุณหภูมิ

@ Click ที่ปุ่ม Temp หรือเลือก Temperature menu จาก Option menu จากนั้น ฟอร์มแสดงค่าอุณหภูมิจะโชว์ดังรูปที่ 79



รูปที่ 79 ฟอร์มแสดงค่าอุณหภูมิ

เมื่อจะออกจากฟอร์ม Tempurative ก็ Click ที่ปุ่ม Exit จาก From

การตั้งค่าเวลา

@Click ที่ปุ่ม Time หรือเลือกที่คำสั่ง Time จาก Menu Option จากนั้นเครื่องจะ โชว์ InputBox เพื่อให้เราตั้งค่าเวลาตามรูปแบบที่กำหนดให้คือ hh:min:sec เมื่อตั้งเวลาถูกต้องแล้วกด OK ถ้าไม่ต้องการตั้งกด Cancel ดังรูปที่ 80

รูปที่ 80 Input Box สำหรับการแก้ไขเวลา

การตั้งค่า วัน,เดือน,ปี

@ กดที่ปุ่ม Date หรือเลือกที่ Option Menu แล้วเลือกไปที่คำสั่งDate.... ซึ่งเครื่องก็จะแสดง Input Box สำกรับแก้ไข วัน,วันที่ ,เดือน, ปี ขึ้นมาตาม โดยมี From ดังต่อไปนี้คือ Ddd/dd/mm/yyyy โดยวันจะต้องได้โดยการใส่ชื่อย่อคือ

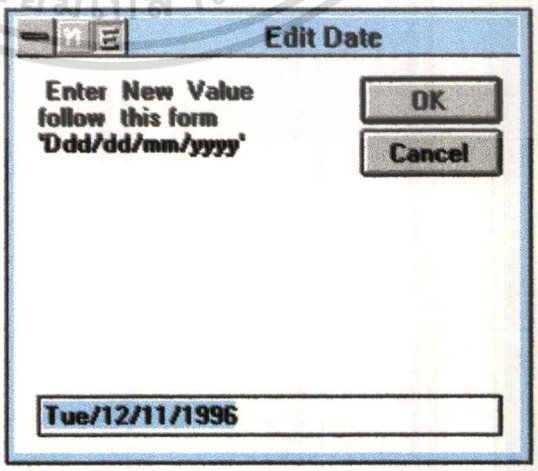
-Day ให้ป้อนค่าดังนี้ Sun, Mon,Tue,Wed,Thu, Fri และ Sat

-Date ตั้งได้ตั้งแต่ 1-31

-เดือน ตั้งได้ตั้งแต่ 1-12

-ปี ตั้งค่าได้ตั้งแต่ 0000-9999

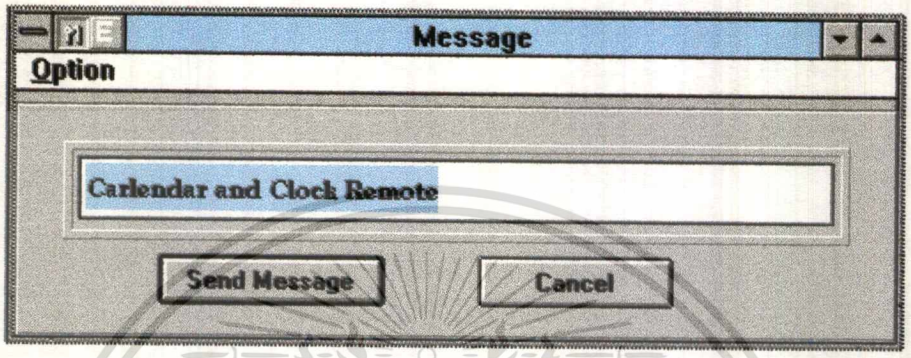
เมื่อใส่ค่าแล้วกด OK ดังรูปที่ 81



รูปที่ 81 แสดง Input Box สำหรับการแก้ไข วัน, วันที่ ,เดือน, ปี

การส่งข้อความ (Message) ให้กับ Master Control Board

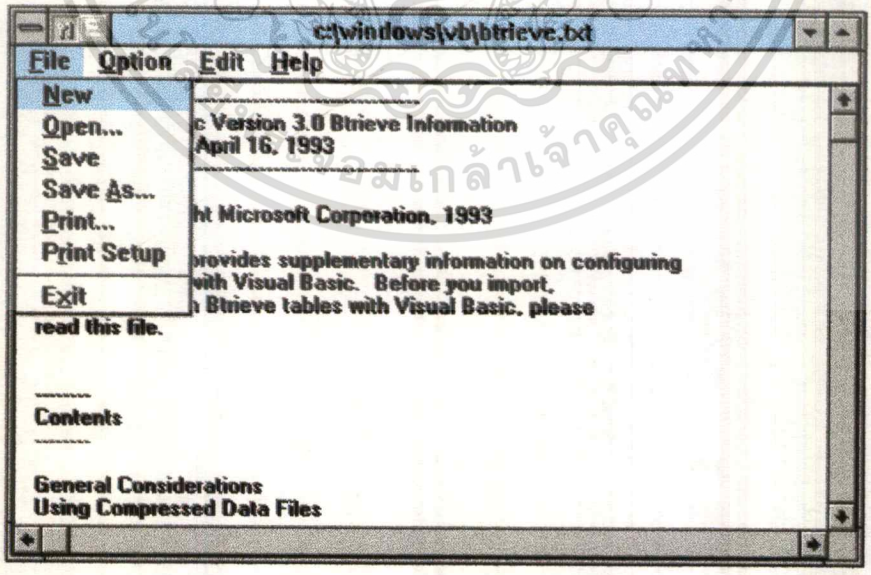
@กดที่ปุ่ม Message หรือ เลือกที่ Menu Option แล้วเลือกที่คำสั่ง Message...จากนั้นก็จะโชว์ฟอร์ม Messageขึ้นมาเพื่อให้เราเขียนข้อความที่จะส่งเมื่อเสร็จจากนั้นกดปุ่ม Send Message หรือกดปุ่ม Enter ถ้าไม่ต้องการส่งก็กดที่ปุ่ม Cancel ดังรูปที่ 82



รูปที่ 82 แสดงฟอร์มการส่งข้อความ

การเปิด File,SaveFile และการอ่านข้อมูล

เราสามารถที่จะเลือกดู File ที่เราเก็บไว้ หรือ Save File ที่เราจะเก็บข้อมูลได้โดยการกดปุ่ม Date หรือ Click ที่ Option Menu แล้วเลือกที่คำสั่ง Data... จากนั้นเครื่องก็จะให้เราเปิดเอกสารที่เราต้องการดูได้โดยมีแบบฟอร์มการใช้งานดังรูปที่ 83



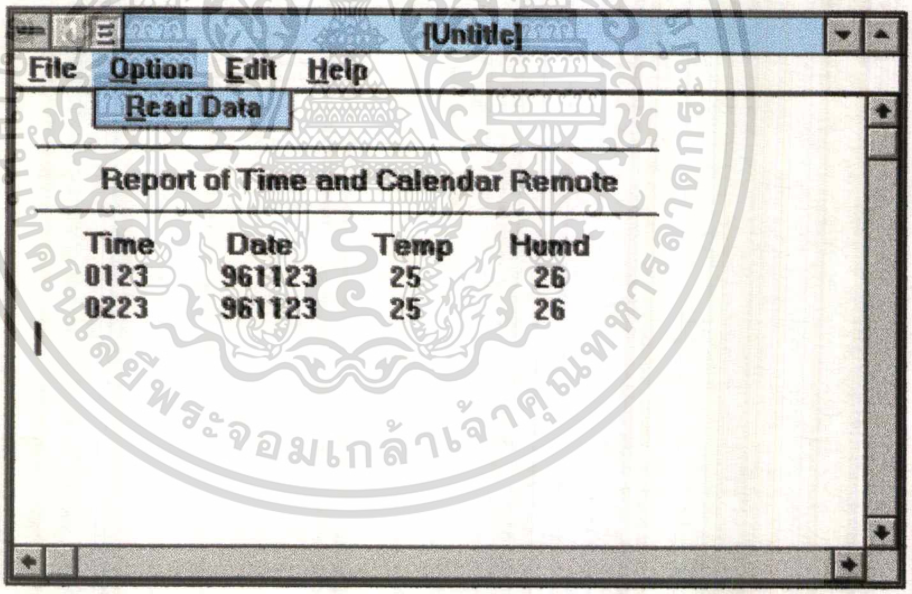
รูปที่ 83 แสดงฟอร์มเกี่ยวกับการจัดการข้อมูล

การเลือกใช้เมนูคำสั่งของ DataViewFrom

- New การเปิดหรือสร้าง File ใหม่
- Open การเปิด File
- Save การบันทึก File
- Save As การบันทึก File เก่าด้วยชื่อ File ใหม่
- Print Setup การ Set เครื่องพิมพ์
- Print การพิมพ์เอกสารออกทางเครื่องพิมพ์
- Exit การออกจาก ฟอร์ม DataView
- Option เป็นการเข้าสู่เมนู Read Data เพื่อการอ่านข้อมูลจาก Master Control Board

ผลการทดลองการอ่านข้อมูลที่เก็บไว้ในบัฟเฟอร์ของ Master Control Board

@ เมื่อเราที่เลือกคำสั่ง Read Data จากเมนู Option ของ DataView form เราก็จะได้ข้อมูลที่เก็บอยู่ในบัฟเฟอร์ของ Master Control Board แสดงออกมาให้เห็นดังรูปที่ 84.



รูปที่ 84 แสดงข้อมูลที่อ่านมาจากบัฟเฟอร์ของ Master Control Board

การออกจาก Program

@โดยการกดปุ่ม Exit หรือ เลือกที่ Option menu แล้วเลือกที่คำสั่ง Exit

ปัญหาที่เกิดขึ้นและวิธีการแก้ไขในโครงงาน

ปัญหาที่เกิดขึ้นในการทำโครงงานเครื่องแสดงเวลาและปฏิทินแบบระยะไกล สามารถแบ่งออกเป็นสองลักษณะใหญ่ ๆ ตามโครงสร้างของระบบคือ

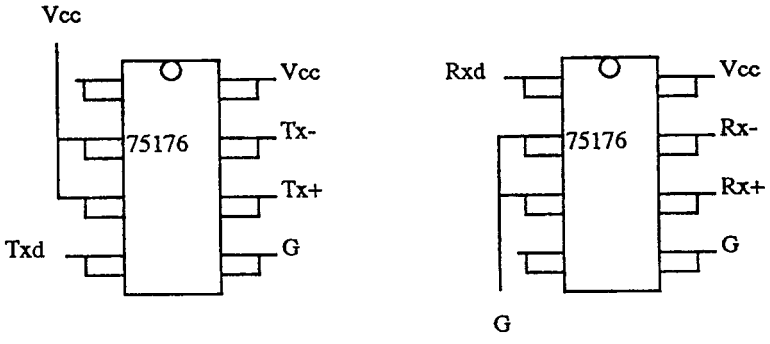
1. ปัญหาและการแก้ไขทางฮาร์ดแวร์
2. ปัญหาและการแก้ไขทางซอฟท์แวร์

เนื่องจากระบบการทำงานทั้งหมดของจะประกอบด้วยส่วนต่าง ๆตามที่ได้อธิบายไว้ในบทที่ 1.ซึ่งการทำงานในส่วนต่าง ๆ ของระบบจะต้องทำการควบคุมการทำงานของฮาร์ดแวร์ด้วยให้เป็นตามเงื่อนไขให้ได้มากที่สุดโดยใช้โปรแกรมซอฟท์แวร์ที่เหมาะสมของส่วนนั้นๆ เช่น การที่ Master Control Board จะต้องมีฮาร์ดแวร์รองรับการส่งข้อมูล แบบอนุกรมตามมาตรฐาน RS 232 และ RS 422 ในขณะที่ Slave Control Board จะต้องมีฮาร์ดแวร์ รองรับการส่งข้อมูลตามมาตรฐาน RS 422 เพียงอย่างเดียว ซึ่งจะพบว่าการทำงานของทั้งระบบต้องมีระบบทางฮาร์ดแวร์ อยู่ 2 ส่วน และการทำงานของระบบทางด้านซอฟท์แวร์จะแบ่งออกได้ทั้งหมด 3 ส่วน คือ โปรแกรม Visual Basic สำหรับการ Upload และ Download จากเครื่องไมโครคอมพิวเตอร์ไปที่ Master Control Board ผ่าน RS 232 และต้องรองรับโปรแกรมซอฟท์แวร์ด้วยเอสเซมบลี้ รวมทั้งต้องส่งข้อมูลผ่านRS 422 ไปยัง Slave Control Board เพื่อนำข้อที่ต้องการไปแสดงผลที่ LED Dotmetrix ต่อไป

จะเห็นว่าการทำงานร่วมกันของระบบดังกล่าว จะต้องใช้สมาชิกเขียนโปรแกรมสำหรับใช้ควบคุมการทำงานของทั้ง 3 ส่วนขึ้นมาปัญหาที่สำคัญก็คือ การเขียนโปรแกรมทั้งหมดให้ทำงานร่วมกันให้เป็นไปตามจุดประสงค์ของโครงงาน ซึ่งก็ประสบปัญหาต่าง ๆอย่างมาก

ปัญหาและการแก้ไขทางฮาร์ดแวร์

- การต่อชิพ IC สำหรับการรับส่งข้อมูลแบบอนุกรมของ RS 422 ซึ่งในโครงงานนี้จะใช้ ชิพ DS 15176 ซึ่งสามารถใช้ได้กับการ รับส่งข้อมูลแบบอนุกรม แบบ RS 485 ได้เช่นกัน ชิพ DS 15176 ภายในจะประกอบด้วย Drivers 1 ชุดและ Receiver 1 ชุด แต่เมื่อนำมาใช้ในการรับส่งข้อมูลแบบอนุกรมของ RS 422 จะต้องนำ DS 15176 2 ตัว มาทำเป็น Drivers 1 ชุดและ Receiver 1 ชุด เนื่องจากการ รับส่งข้อมูลแบบอนุกรมของ RS 422 จะเป็นการรับส่งข้อมูลแบบ 4 สาย คือ การส่ง 2 สาย (Tx+,Tx-) และการรับ 2 สาย (Rx+,Rx-) ดังนั้นการต่อขาการรับส่งต้องมีการจัดการกับขาที่ไม่ได้ใช้งานอย่างถูกต้องเพื่อป้องกันความผิดพลาดในการใช้งานโดยการต่อขาที่ไม่ได้ใช้งานตามรูปที่ 85



(a) การต่อใช้งานเป็น Drivers (b) การต่อใช้งานเป็น Receivers

รูปที่ 85 แสดงการต่อใช้งานชิพ DS 75176

- การต่อชิพ RTC (DS 1202) ในการใช้งานชิพนี้จะต้องมีการใช้ Cytal สำหรับเป็นฐานเวลาให้กับ RTC ทำให้ การต่อ Cytal จะต้องต่อตามที่แสดงไว้ในรายละเอียดการใช้งานในของ Master Control Board และต้องทำการตั้งค่าเริ่มต้นให้กับ RTC ทุกครั้งที่เริ่มต้นการใช้งานทำให้อาจจะเกิดความผิดพลาดขึ้นถ้าไม่ใส่ค่าเริ่มต้นคือ นาฬิกาจะไม่เดิน เนื่องจากค่าเริ่มต้นอาจไม่อยู่ในช่วงที่ RTC ทำการอ่านค่าเวลาได้ และต้องระวังการรบกวนของสัญญาณภายนอก ที่จะทำให้ Cytal มีค่าความถี่แตกต่างไปจากค่าตามที่ระบุไว้ ซึ่งผลก็คือ นาฬิกาจะเดินผิดไปจากความเป็นจริง

- การต่อชิพ Digital Temperature Sensor (DS 1620) เป็นชิพที่ใช้สำหรับการตรวจวัดค่าอุณหภูมิที่ Slave Control Board โดยจะทำการแปลงค่าอุณหภูมิมาเป็นสัญญาณดิจิตอลและสามารถอ่านได้ 2 หน่วยคือ หน่วยของศาเซลเซียส และองศาฟาเรนไฮต์ การอ่านค่าอุณหภูมิจากชิพ DS 1620 จะเป็นการอ่านแบบ อนุกรมพัลซ์ เช่นเดียวกับชิพ RTC การต่อใช้งานต้องให้เป็นไปตามที่ระบุตามที่กำหนดมิฉะนั้นจะทำให้ ไม่มีสัญญาณเอาท์พุทออกมาจากชิพ DS 1620

- การต่อชิพ Analog To Digital Converter (Max 192) เป็นชิพที่รองรับการตรวจจับการเปลี่ยนแปลงของค่าความชื้นโดยทำการแปลงค่า อินพุทที่เป็น Analog ตั้งแต่ 0-5 โวลท์ DC ให้เป็นค่า Digital ขนาด 11 บิต แต่การอ่านจะใช้ค่าที่ 8 บิตทางด้านซ้ายเพื่อให้มีความไวในการแปลงค่าลดลงเนื่องจากการแปลงค่าของชิพ Max 192 มีความไวในการวัดประมาณ 10 ns และการต่อใช้งานจำเป็นต้องมีแรงดันอ้างอิงที่ศูนย์โวลท์ โดยต้องมีการปรับค่าอินพุทเพื่อให้การแปลงค่าที่เป็นดิจิตอลมีค่าที่เป็นเลขฐานสิบหกตรงกับค่าเลขฐานสิบตรงกับแรงดันอินพุท

ปัญหาและการแก้ไขทางซอฟท์แวร์

- ปัญหาในการพัฒนาโปรแกรมในการ Upload และ Download ด้วย Visual Basic ซึ่งจะเป็นส่วนที่ทำงานบนเครื่องไมโครคอมพิวเตอร์นั้น จะเกิดขึ้นในกรณีของโปรแกรมการติดต่อ

สื่อสารกับ พอร์ตอนุกรม (Com1 หรือ Com2) ซึ่งปัญหาที่เกิดขึ้นคือ โปรแกรมที่เขียนไว้แล้วเพื่อรอรับข้อมูลที่ส่งมาจาก Master Control Board เพื่อนำข้อมูลเข้ามาใช้งานจริง แต่เมื่อไปต่อเข้ากับ Master Control Board จริงๆแล้วข้อมูลที่นำเข้ามาได้นั้นไม่ถูกต้องตามที่ต้องการเนื่องจาก แบบฟอร์มการส่งข้อมูลเข้ามาของ Master Control Board นั้นไม่ถูกต้องกับเงื่อนไขของโปรแกรมที่เขียนไว้ในตอนแรกจึงทำให้เกิดปัญหาขึ้นในขั้นตอนการทดลองโปรแกรม รวมทั้งระบบ การแก้ปัญหาคือต้องทำการแก้ไขเงื่อนไขของโปรแกรม Visual Basic ในส่วนของการติดต่อกับ พอร์ตอนุกรม พร้อมทั้งเพิ่มเงื่อนไขบางส่วนเข้าไปในโปรแกรม เพื่อให้สอดคล้องกับข้อมูลที่ได้รับมาจาก Master Control Board จึงทำให้สามารถแก้ไขปัญหาที่เกิดขึ้นได้

- ปัญหาในการพัฒนาโปรแกรม Assembly สำหรับใช้ในการควบคุมการทำงานของ Master Control Board โดยรายละเอียดทางด้านฮาร์ดแวร์ของบอร์ด และซอฟท์แวร์ ได้แสดงไว้ในบทที่ 5 และบทที่ 6 ปัญหาที่เกิดขึ้นในการเขียนโปรแกรมควบคุมจะเกิดขึ้นในขั้นตอนการพัฒนาโปรแกรม คือ การทดลองโปรแกรมควบคุมนั้นจะทำได้จากสองส่วน คือการทดลองบน Single Board และทดลองบน Remote Monitor ซึ่งการทดลองโปรแกรมทั้งสองแบบจะประสบปัญหาที่เหมือนกันคือ หน่วยความจำแรมภายในบางส่วนจะถูกนำไปใช้ในโปรแกรมมอนิเตอร์ของ Single Board และ Remote Monitor และปัญหาอีกอย่างหนึ่งก็คือการใช้การอินเทอร์รัพท์จะถูกกระทำแบบจำลองตำแหน่งที่เป็นจริง มาเป็นแบบ Visual เมื่อนำมาทดลองกับการทำงานจริงบน Master Control Board จะต้องตั้งตำแหน่งไว้มิฉะนั้นจะเกิดการทำงานที่ไม่ตรงกับการอินเทอร์รัพท์ที่เป็นจริงของ CPU MCS - 51 นอกจากนี้ยังเกิดปัญหากับการติดต่อกับชิพที่ต้องทำงานร่วมกันคือ ชิพ RTC (DS 1202) ที่ต้องมีการติดต่อแบบอนุกรมพัลซ์ในแต่ละคำสั่งโดยต้องทำการอ่านทุก ๆ ครั้งที่ต้องการแสดงผลและต้องการส่งข้อมูล การอ่านค่ามาจากชิพ RTC ซึ่งจะได้เลขฐานสิบหก จะต้องนำมาแปลงเป็น ASCII Code สำหรับการส่งข้อมูลและแปลงเป็น Segment Code สำหรับการแสดงผลที่ 7-Segment ในการส่งจะส่งข้อมูลของเวลาทุก ๆ เวลาก่อนการเปลี่ยนแปลงที่จะเกิดขึ้น

- ปัญหาการพัฒนาโปรแกรม ASSEMBLY สำหรับใช้ในการควบคุมการทำงานทางด้าน Slave Control Board การเขียนโปรแกรมภาษา ASSEMBLY นั้นจะซับซ้อนและมีรายละเอียดปลีกย่อยมากพอสมควร ดังที่กล่าวมาแล้วในหัวข้อการควบคุมการทำงานของบอร์ดควบคุมรอง (Slave Board) โปรแกรมในส่วนนี้ประกอบด้วย 5 โปรแกรมด้วยกันคือ

- โปรแกรมการรับข้อมูล
- โปรแกรมการส่งข้อมูล
- โปรแกรมการแสดงผล
- โปรแกรมการตรวจจับอุณหภูมิ
- โปรแกรมการตรวจจับ ADC

ปัญหาแต่ละโปรแกรม

- โปรแกรมการรับข้อมูล การรับข้อมูลจะใช้การตรวจสอบการ อินเตอร์รัปและตรวจสอบ โปรโตคอล การรับส่งข้อมูล ก่อนการทำงานจะต้องกำหนดค่าต่างๆที่เกี่ยวข้องกับการทำงานของ Serial Port และหลังจากการรับข้อมูลจะต้องคืนค่าต่างๆเหล่านี้ให้เหมือนเดิมด้วยมิฉะนั้นการรับข้อมูลผิดพลาดในการรับข้อมูลครั้งต่อไป

- โปรแกรมการส่งข้อมูล การส่งข้อมูลจะใช้การตรวจสอบสถานะของบิต TI และก่อนการส่งทุกครั้งต้องกำหนดค่าต่างๆที่เกี่ยวกับการติดต่อทาง พอร์ตอนุกรม ให้ตรงกับทางด้านตัวรับข้อมูล และ การส่งข้อมูลตัวแรกจะต้องกำหนดให้บิต TI เป็น 1 เพียงครั้งเดียว หลังจากนั้นสามารถส่งได้โดยการตรวจสอบสถานะของบิต TI ได้เลย

- โปรแกรมการแสดงผลข้อมูล บางคำสั่งเมื่อดูรูปแบบคำสั่งถูกต้องแต่เวลาใช้งานจริงกลับไม่สามารถทำงานได้ เช่น คำสั่งSJMP หรือคำสั่ง PUSH , POP ซึ่ง 2 คำสั่งนี้จะใช้ในโปรแกรมย่อยที่ซ้อนกันหลายโปรแกรมย่อยนั้นทำให้บางครั้งการทำงานผิดพลาดได้

การแก้ปัญหาทาง ซอร์ฟแวร์ ต้องพยายามจัดรูปแบบการทำงานอย่าให้มีหลายโปรแกรมย่อยซ้อนกันมากนั้น และ คำสั่ง SJMP ต้องคำนึงถึงความสามารถของตัวคำสั่งเองด้วย ถ้าเป็นไปได้พยายามหลีกเลี่ยงการใช้คำสั่งนี้ในกรณีที่การกระโดดมีระยะใกล้เคียงกับความสามารถของตัวคำสั่ง(+128 และ -127)สิ่งที่สำคัญในการเขียนโปรแกรมภาษาASSEMBLY คือจะต้องทำความเข้าใจคุณสมบัติของตัวคำสั่งนั้นๆให้ดีและต้องพยายามฝึกฝน เพื่อเพิ่มทักษะและเพิ่มเทคนิคการเขียนโปรแกรม และการส่งข้อมูลไปยัง บอร์ดแสดงผลจะต้องส่งข้อมูลกำหนดหลักว่าจะให้หลักไหนแสดงผลออกไปก่อน หลักจากนั้นถึงจะตามด้วยข้อมูลที่จะแสดงผล ซึ่งข้อมูลเหล่านี้เป็นรหัส ACSII ต้องกลับค่าข้อมูลจาก 0 เป็น 1 หรือจาก 1 เป็น 0 (ใช้คำสั่ง CPL A) ทั้งนี้เพราะบอร์ดแสดงผลจะทำงานที่ สถานะต่ำ (ACTIVE LOW)

- โปรแกรมการตรวจจับอุณหภูมิ และ ตรวจสอบการแปลงข้อมูลจาก อะนาลอก เป็นดิจิตอล จะต้องเขียนโปรแกรมให้ตรงกับ TIMING DIAGRAM

บทที่ 7

ซอร์สโปรแกรมของระบบ

ซอร์สโปรแกรมของระบบจะประกอบด้วย

1 ซอร์สโปรแกรมของ Master Control Board

2 ซอร์สโปรแกรมของ Slave Control Board

3 ซอร์สโปรแกรมของ Visual Basic

ซอร์สโปรแกรมของ Master Control Board

```
;FILENAME        MASTER CONTROL.ASM
;DESCRPTION      CONTROL ALL SYSTEM FOR ANT32
;HARDWARE        MASTER BOARD : ANT-32 V2
;ASSEMBLER       SXA51
;PROGRAMMER      S IN-YIN
;DEPT            INSTRUMENTATION  KMITL
;*************************
;*****INTERNAL RAM *******
;*************************
ORG             8200H
P1A             EQU  0FC00H
P1B             EQU  0FC01H
P1C             EQU  0FC02H
P1P             EQU  0FC03H
TBTI            EQU  09000H    ;TABLE RECEIVE TIME
TBCAL           EQU  09010H    ;TABLE RECEIVE CALENDAR
TBSTI           EQU  09020H    ;TABLE FOR SEND TIME
TBTEM           EQU  09040H    ;TABLE TEMPERATURE
TBHUM           EQU  09050H    ;TABLE HUMIDITY
TBMASS          EQU  09060H    ;TABLE MASSAGE
TBREPO          EQU  09100H    ;TABLE REPORT
SECBUF          EQU  20H
MINBUF          EQU  21H
HOURBUF         EQU  22H
YEARBUF         EQU  23H
MONTBUF         EQU  24H
```

```
DATEBUF     EQU  25H
DAYBUF      EQU  26H
TEMBUF      EQU  27H
HUMBUF      EQU  28H
STEPSET     EQU  29H
MAX         EQU  2AH
MIN         EQU  2BH
DECBUF      EQU  2CH
BUFSET      EQU  2DH
HIGBUF      EQU  2EH
LOWBUF      EQU  2FH
NEWKEY      EQU  30H
DISTEP      EQU  31H
WAITBUF     EQU  32H
WAITIME     EQU  33H
DISBUF      EQU  34H
HEXBUF      EQU  3AH
BUFF        EQU  3DH
WAITB       EQU  3EH
WAITI       EQU  3FH
DATABI      EQU  P1.4
CLKBIT      EQU  P1.5
RSTBIT      EQU  P1.6
;***** MAIN MONITOR *****
;ORG     0000H
;*****SERVICE SERIAL INTERRUPT*****
        ORG         08100H      ;VECTOR SERIAL INTERRUPT
        LJMP        SONG
        ORG         08105H      ;VECTOR TIME0  INTERRUPT
        LJMP        TIMES
        ;***** SERVICE INTERRUPT *****
        ORG         08150H      ;START SERVICE INTERRUPT
SONG:   PUSH        PSW
        PUSH        ACC
        JBC         RI,XX
        JBC         TI,YY
```

```
ZZ:     POP     ACC
        POP     PSW
        RETI
XX:     MOV     A,SBUF
        CJNE    A,#'@',ZZ
        LCALL   RECEIVE
        SJMP    ZZ
YY:     LCALL   SENDTIME
        SJMP    ZZ
TIMES:  PUSH    PSW
        PUSH    ACC
        LCALL   RPORT       ;SAVE REPORT FOR SEND
        POP     ACC
        POP     PSW
        RETI
        ;*********************************
        ;*****COMMUNICATETION SUB ****
        ;*********************************
        ORG     08200H
        LJMP    MAIN0       ;START MAIN MONITOR
        ;*****RECEIVE DATA*****
        ;REG A
RECEIVE: LCALL  RBYTE
        CJNE    A,#'&',X1   ;ID TIME
        LCALL   SAVE1
        RET
X1:     CJNE    A,#'#',X2   ;ID  CALENDAR
        LCALL   SAVE2
        RET
X2:     CJNE    A,#'T',X3   ;ID TEMPERATURE
        LCALL   SAVE3
        RET
X3:     CJNE    A,#'H',X4
        LCALL   SAVE4       ;ID  HUMIDITY
        RET
X4:     CJNE    A,#'*',X5
```

```
        LCALL   SAVE5       ;ID MASSAGE
        RET
X5:     CJNE    A,#'I',X6
        LCALL   SENDREPO    ;SEND REPORT
        RET
X6:     RET
        ;*****RECEIVE & SAVE TIME FORM PC*****
        ;REG A,DPTR
SAVE1:  MOV     DPTR,#TBTI
        MOV     A,#'@'
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#'&'
        MOVX    @DPTR,A
        CLR     A
SAVE12: LCALL   RBYTE
        CJNE    A,#'$',SAVE10
        INC     DPTR
        MOVX    @DPTR,A
        SJMP    CONTIME
SAVE10: INC     DPTR
        MOVX    @DPTR,A
        CLR     A
        SJMP    SAVE12
CONTIME: MOV    R0,#03H     ;LOOP CONVERT
        MOV     SECBUF,#00
        MOV     MINBUF,#00
        MOV     HOURBUF,#00
        MOV     R1,#SECBUF
        MOV     DPTR,#TBTI+2
CONT2:  MOVX    A,@DPTR
        MOV     R2,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R3,A
        LCALL   ATOH
```

```
        MOV     @R1,A
        DJNZ    R0,CONT1
        LCALL   SETIME
        RET
CONT1:  INC     R1
        INC     DPTR
        SJMP    CONT2
        ;*****  SETTIME FOR RTC ******
        ;REG R5,R6
SETIME: CLR     P1.6        ;RST=0
        SETB    P1.5        ;SCLK=1
        LCALL   RTCDL
        MOV     R6,#08EH    ;WR PROTECTION ON
        MOV     R7,#00H
        LCALL   BYTEWR
        MOV     R6,#80H     ;SETSEC
        MOV     R7,SECBUF
        LCALL   BYTEWR
        MOV     R6,#082H    ;SETMIN
        MOV     R7,MINBUF
        LCALL   BYTEWR
        MOV     R6,#084H    ;SETHOUR
        MOV     R7,HOURBUF
        LCALL   BYTEWR
        MOV     R6,#08EH    ;WR PROTECTION OFF
        MOV     R7,#00H
        LCALL   BYTEWR
        RET
        ;******RECEIVE & SET CALENDAR FORM PC ****
        ;REG A,R0,R1,R2,R3
SAVE2:  MOV     DPTR,#TBCAL
        MOV     A,#'@'
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#'#'
        MOVX    @DPTR,A
```

```
        CLR     A
SAVE22: LCALL   RBYTE
        CJNE    A,#'$',SAVE20
        INC     DPTR
        MOVX    @DPTR,A
        SJMP    CONCAL
SAVE20: INC     DPTR
        MOVX    @DPTR,A
        SJMP    SAVE22
CONCAL: MOV     R0,#04H
        MOV     YEARBUF,#00
        MOV     MONTBUF,#00
        MOV     DATEBUF,#00
        MOV     DAYBUF,#00
        MOV     R1,#YEARBUF
        MOV     DPTR,#TBCAL+2   ;CONVERT ASCII TO HEX
CONC2:  MOVX    A,@DPTR
        MOV     R2,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R3,A
        LCALL   ATOH
        MOV     @R1,A
        DJNZ    R0,CONC1
        LCALL   SETCAL
        RET
CONC1:  INC     R1
        INC     DPTR
        SJMP    CONC2
        ;***** SET CALENDAR  FOR RTC *****
        ;REG  R6,R7
SETCAL: CLR     P1.6            ;RST=0
        SETB    P1.5            ;SCLK=1
        LCALL   RTCDL
        MOV     R6,#08EH        ;WR PROTECTION ON
        MOV     R7,#00H
```

```
        LCALL   BYTEWR
        MOV     R6,#08CH        ;SETYEAR
        MOV     R7,YEARBUF
        LCALL   BYTEWR
        MOV     R6,#088H        ;SETMONTH
        MOV     R7,MONTBUF
        LCALL   BYTEWR
        MOV     R6,#086H        ;SETDATE
        MOV     R7,DATEBUF
        LCALL   BYTEWR
        LCALL   BYTEWR
        MOV     R6,#08AH        ;SETDAY
        MOV     R7,DAYBUF
        LCALL   BYTEWR
        MOV     R6,#08EH        ;WR PROTECTION OFF
        MOV     R7,#00H
        LCALL   BYTEWR
        RET
        ;*****RECEIVE TEMPERATURE FROM PC *****
SAVE3:  MOV     DPTR,#TBTEM     :SAVE TEMP
        MOVX    @DPTR,A
SAVE32: LCALL   RBYTE
        CJNE    A,#'$',SAVE30
        SJMP    CONTEM
SAVE30: INC     DPTR
        MOVX    @DPTR,A
        SJMP    SAVE32
CONTEM: MOV     DPTR,#TBTEM+1       ;CONVERT ASCII TO HEX
        MOVX    A,@DPTR
        MOV     R2,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R3,A
        LCALL   ATOH
        MOV     TEMBUF,A
        RET
```

```
        ;*****RECEIVE HUMIDITY FROM SLAVE *****
        ;REG A R2 R3
SAVE4:  MOV     DPTR,#TBHUM     ;SAVE HUMIDITY
        MOVX    @DPTR,A
SAVE42: LCALL   RBYTE
        CJNE    A,#'$',SAVE40
        SJMP    CONHUM
SAVE40: INC     DPTR
        MOVX    @DPTR,A
        SJMP    SAVE42
CONHUM: MOV     DPTR,#TBHUM+1   ;CONVERT ASCII TO HEX
        MOVX    A,@DPTR
        MOV     R2,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     R3,A
        LCALL   ATOH
        MOV     TEMBUF,A
        RET
        ;*****RECEIVE MASSAGE FROM PC *****
        ;REG A R2 R3
SAVE5:  MOV     DPTR,#TBMASS    ;SAVE MASSAGE
        MOV     A,#'@'
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#'*'
        MOVX    @DPTR,A
        CLR     A
SAVE52: LCALL   RBYTE
        CJNE    A,#'S',SAVE50   ;SAVE END TABLE MASSAGE
        INC     DPTR
        MOVX    @DPTR,A
        RET
SAVE50: INC     DPTR
        MOVX    @DPTR,A
        SJMP    SAVE52
```

```
        ;*****SAVE TABLE FOR REPORT*****
        ;REG A, DPTR
RPORT:  MOV     DPTR,#TBREPO
        MOV     A,#'@'
        MOVX    @DPTR,A
        LCALL   READ
        MOV     DPH,HIGBUF
        MOV     DPL,LOWBUF
        MOV     A,#'!'          ;START BUFFER BLOCK
        MOVX    @DPTR,A
        CLR     A
        MOV     A,HOURBUF       ;DATA 2
        LCALL   HTOA
        MOV     A,R2
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R3
        MOVX    @DPTR,A
        CLR     A
        MOV     A,MINBUF        ;DATA 1
        LCALL   HTOA
        MOV     A,R2
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R3
        MOVX    @DPTR,A
        CLR     A
        MOV     A,DATEBUF       ;DATA 3
        LCALL   HTOA
        MOV     A,R2
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R3
```

```
MOVX    @DPTR,A
CLR     A
MOV     A,MONTBUF    ;DATA 4
LCALL   HTOA
MOV     A,R2
INC     DPTR
MOVX    @DPTR,A
INC     DPTR
MOV     A,R3
MOVX    @DPTR,A
CLR     A
MOV     A,YEARBUF    ;DATA 5
LCALL   HTOA
MOV     A,R2
INC     DPTR
MOVX    @DPTR,A
INC     DPTR
MOV     A,R3
MOVX    @DPTR,A
CLR     A
MOV     A,TEMBUF     ;DATA 6
LCALL   HTOA
MOV     A,R2
INC     DPTR
MOVX    @DPTR,A
INC     DPTR
MOV     A,R3
MOVX    @DPTR,A
CLR     A
MOV     A,HUMBUF     ;DATA 7
LCALL   HTOA
MOV     A,R2
INC     DPTR
MOVX    @DPTR,A
INC     DPTR
MOV     A,R3
```

```
        MOVX    @DPTR,A
        CLR     A
        MOV     A,#'&'          ;END BUFFER BLOCK
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#'$'          ;END TABLE
        MOVX    @DPTR,A
        MOV     R2,#096H
        MOV     R3,#0F0H
        LCALL   DPCOM
        JC      YAYA
        JNC     YAYA1
        RET
YAYA1:  MOV     A,#'$'
        INC     DPTR
        MOVX    @DPTR,A
        MOV     HIGBUF,#091H    ;START  TABLE FOR REPORT
        MOV     LOWBUF,#001H
        RET
YAYA:   MOV     HIGBUF,DPH      ;LOOP SAVE
        MOV     LOWBUF,DPL
        RET
        ;*****SAVE TIME TO TABLE*****
SAVETI: LCALL   READ
        MOV     DPTR,#TBSTI
        MOV     R0,#09H
        MOV     A,#'@'
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#'&'          ;HEAD TABLE
        MOVX    @DPTR,A
        MOV     R1,#SECBUF
SAVETI2: MOV    A,@R1
        LCALL   HTOA
        MOV     A,R2
```

```
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,R3
        MOVX    @DPTR,A
        DJNZ    R0,SAVETI1
        MOV     A,#'$'          ;END TABLE
        INC     DPTR
        MOVX    @DPTR,A
        RET
SAVETI1: INC    R1
        SJMP    SAVETI2
        ;******SEND TIME ******
SENDTIME: CALL  START5
        MOV     DPTR,#TBSTI
LOOP:   MOVX    A,@DPTR
        CJNE    A,#'$',THEN
        CALL    SENO
        MOV     A,#0DH
        CALL    SENO
        MOV     A,#0AH
        CALL    SENO
        RET
THEN:   ACALL   SENO
        INC     DPTR
        SJMP    LOOP
SENO:   MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        RET
START5: MOV     SCON,#50H
        MOV     IE,#00H
        MOV     PCON,#00H
        MOV     TMOD,#20H
        MOV     TH1,#0FDH
        SETB    TR1
```

```
        RET
        ;******SEND  REPORT TO PC*******
SENDREPO: CALL      START5
        MOV         DPTR,#TBREPO
LOOP1:  MOVX        A,@DPTR
        CJNE        A,#'$',THEN1
        CALL        SENO
        MOV         A,#0DH
        CALL        SENO
        MOV         A,#0AH
        CALL        SENO
        RET
THEN1:  ACALL       SENO
        INC         DPTR
        SJMP        LOOP1
        ;*****WAIT  FOR SEND TIME*****
PLAY4:  MOV         R6,#081H        ;READ SEC ONLY
        LCALL       BYTERD
        MOV         SECBUF,R7
        MOV         A,R7
        CJNE        A,WAITBUF,WAI   ; SEC CHANGE
        MOV         WAITBUF,A
        RET
WAI:    MOV         WAITBUF,A
        LCALL       SENDTIME
        RET
        ;*****WAIT  FOR SAVE REPORT *****
PLAY5:  MOV         A,MINBUF
        CJNE        A,#015H,PA1     ;15 MIN
        MOV         R5,BUFF
        CJNE        R5,#00H,POX1
        LCALL       RPORT           ;SAVE  REPORT
        MOV         BUFF,#0FFH
POX1:   RET
PA1:    CJNE        A,#030H,PA2     ;30 MIN
        MOV         R5,BUFF
```

```
        CJNE    R5,#00H,POX1
        LCALL   RPORT       ;SAVE REPORT
        MOV     BUFF,#0FFH
        RET
PA2:    CJNE    A,#045H,PA3 ;45 MIN
        MOV     R5,BUFF
        CJNE    R5,#00H,POX1
        LCALL   RPORT       ;SAVE REPORT
        MOV     BUFF,#0FFH
        RET
PA3:    CJNE    A,#059H,PA4 ;59 MIN
        MOV     R5,BUFF
        CJNE    R5,#00H,POX1
        LCALL   RPORT       ;SAVE REPORT
        MOV     BUFF,#0FFH
        RET
PA4:    MOV     BUFF,#00H
        RET
        ;***** RBYTE SUB ******
        ;OUT = A
        ;REG = A
RBYTE:  JNB     RI,$        ;WAIT FOR RECEIVE OK
        CLR     RI
        MOV     A,SBUF
        RET
        ;***** SBYTE SUB *****
        ; SEND BYTE
        ; IN  = A
        ; REG = NO
SBYTE:  JNB     TI,$        ;WAIT FOR SEND OK
        CLR     TI
        MOV     SBUF,A
        RET
        ;***** ATOH SUB *****
        ;ASCII TO HEX CONVERT
        ;IN  = R2,R3   30H,41H
```

```
        ;OUT = A      0AH
        ;REG = A,R2
ATOH:   MOV           A,R2
        CALL          ATOHS
        SWAP          A
        MOV           R2,A
        MOV           A,R3
        CALL          ATOHS
        ORL           A,R2
        RET
ATOHS:  CJNE          A,#'A',$+3
        JC            ATOHS1
        ADD           A,#9
ATOHS1: ANL           A,#0FH
        RET
        ; ***** HTOA SUB *****
        ; CONVERT HEX TO ASCII
        ; IN  = A
        ; OUT = R2,R3
        ; REG = A,R2,R3
HTOA:   PUSH          ACC
        SWAP          A
        CALL          HTOAS
        MOV           R2,A
        POP           ACC
        CALL          HTOAS
        MOV           R3,A
        RET
HTOAS:  ANL           A,#0FH
        CJNE          A,#0AH,$+3
        JNC           HTOAS1
        ORL           A,#30H
        RET
HTOAS1: SUBB          A,#9
        ORL           A,#40H
        RET
```

```
        ;***** SET MODE SERIAL******
SERIAL: MOV     PCON,#00H       ;CLR
        MOV     TMOD,#020H      ;TIMER1 AUTO-RELOAD MODE
        MOV     TH1,#0FDH       ;BAUDRATE 9600 BPS
        MOV     TCON,#040H      ;TIMER1 ON
        RET
        ;*********************************
        ;*****START  MAINMONITOR*****
        ;*********************************
MAIN0:  MOV     SCON,#50H       ;SERRIAL MODE 1
        MOV     IE,#090H        ;ENABLE  SERIAL  INTERRUPT
        MOV     IP,#010H        ;PRIORIYT SERIAL INTERRUPT
        LCALL   SERIAL          ;SERIAL MODE1  BUADRATE 9600
        MOV     HIGBUF,#091H    ;START TABLE  FOR REPORT
        MOV     LOWBUF,#01H
        LCALL   INIRTC
        MOV     DISTEP,#00H     ;START DISPLAY
MAIN:   MOV     R6,#081H        ;READ SEC ONLY
        LCALL   BYTERD
        MOV     SECBUF,R7
        MOV     WAITBUF,SECBUF  ;START AUTO STEP
        MOV     WAITIME,#00H
START:  MOV     SCON,#50H       ;SERRIAL MODE 1
        MOV     IE,#090H        ;ENABLE  SERIAL  INTERRUPT
        MOV     IP,#010H        ;PRIORIYT SERIAL INTERRUPT
        LCALL   SERIAL          ;SERIAL MODE1  BUADRATE 9600
        MOV     HIGBUF,#091H    ;START TABLE  FOR REPORT
        MOV     LOWBUF,#01H
        LCALL   INI8255
S1:     LCALL   SCANKEY
        MOV     R0,NEWKEY
        CJNE    R0,#0F0H,WAHTKEY    :NO PRESS KEY
        LCALL   SAVETI              ;SAVE TIME FOR SEND
        LCALL   PLAY5               ;SAVE REPORT 15 MIN
        LCALL   DISPLAY1            ;MANUAL  DISPLAY
```

```
        LCALL   PLAY4
        SJMP    START
WAHTKEY: CJNE   R0,#0D0H,KEY01  ;UP KEYPRESS
        LCALL   KEYDIS1
        SJMP    START
KEY01:  CJNE    R0,#0E0H,KEY02  ;DOWN KEYPRESS
        LCALL   KEYDIS2
        SJMP    START
KEY02:  CJNE    R0,#0B0H,KEY03  ;LELF KEYPRESS
        LJMP    MAIN1           ;SET PROGRAM
KEY03:  CJNE    R0,#070H,SODA   ;RIGHT KEYPRESS
        LJMP    MAIN1           ;SET PROGRAM
SODA:   SJMP    S1
        ;*****DELAY  SUB*****
        ;REG = R2,R3
DELAY1: MOV     R3,#06H
ABB:    MOV     R2,#017H
        DJNZ    R2,$
        DJNZ    R3,ABB
        RET
DELAY2: MOV     R3,#0F0H
ABB1:   MOV     R2,#0F0H
        DJNZ    R2,$
        DJNZ    R3,ABB1
        RET
        ;***** SET8255 SUB *****
        ;REG = A
INI8255: MOV    A,#088H
        MOV     DPTR,#P1P
        MOVX    @DPTR,A
        RET
        ;***** INITIAL &WRITE RTC *****
INIRTC: CLR     P1.6            ;RST=0
        SETB    P1.5            ;SCLK=1
        LCALL   RTCDL
        MOV     R6,#08EH        ;WR PROTECTION ON
```

```
        MOV     R7,#00H
        LCALL   BYTEWR
        MOV     R6,#80H         ;INISEC
        MOV     R7,#00H
        LCALL   BYTEWR
        MOV     R6,#082H        ;INIMIN
        MOV     R7,#00H
        LCALL   BYTEWR
        MOV     R6,#084H        ;INIHOUR
        MOV     R7,#00H
        LCALL   BYTEWR
        MOV     R6,#08CH        ;INIYEAR
        MOV     R7,#96H
        LCALL   BYTEWR
        MOV     R6,#088H        ;INIMONT
        MOV     R7,#09H
        LCALL   BYTEWR
        MOV     R6,#086H        ;INIDATE
        MOV     R7,#21H
        LCALL   BYTEWR
        MOV     R6,#08AH        ;INIDAY
        MOV     R7,#01H
        LCALL   BYTEWR
        MOV     R6,#08EH        ;WR PROTECTION OFF
        MOV     R7,#00H
        LCALL   BYTEWR
        RET
        ;*****SCANKEY SUB*****
        ;REG = A,R0
SCANKEY: MOV    DPTR,#P1C       ;READKEY FORM PORT C
        MOVX    A,@DPTR
        MOV     R0,A
        CJNE    R0,#0F0H,RBA1
        MOV     NEWKEY,R0
        SJMP    SCAN1
RBA1:   MOV     NEWKEY,R0
```

```
        LCALL   DELAY1
SCAN1:  MOV     DPTR,#P1C
        MOVX    A,@DPTR
        CJNE    A,NEWKEY,SCANKEY
        MOV     NEWKEY,A
        RET
        ;*****DISPLAY 7SEGMENT SUB*****
        ;REG = A,R0,R5
DISPLAY11: CPL  P1.7            ;WATCH DOG
        LCALL   HTOS
        MOV     R0,#DISBUF
        MOV     R5,#06H
        MOV     DPTR,#TBSTOB
MAINDIS: CLR    A
        MOVC    A,@A+DPTR
        MOV     R6,A
        PUSH    DPH
        PUSH    DPL
        MOV     A,@R0
        MOV     DPTR,#P1A       ;OUT DATA
        MOVX    @DPTR,A
        MOV     DPTR,#P1B       ;STOBE
        MOV     A,R6
        MOVX    @DPTR,A
        LCALL   DELAY1
        POP     DPL
        POP     DPH
        INC     DPTR
        INC     R0
        DJNZ    R5,MAINDIS
        RET
TBSTOB: DB      0EFH,0DFH,0FBH
        DB      0F7H,0FEH,0FDH
DIS22:  LCALL   HTOS
        MOV     R0,#DISBUF+2
        MOV     R5,#02H
```

```
        MOV     DPTR,#TBSTOB+2
        SJMP    MAINDIS
        ;*****WRITE RTC SUB*****
        ;IN = R6
        ;OUT= R7
BYTEWR: CLR     P1.4        ;COMMAND WRITE
        LCALL   RTCDL
        SETB    P1.6        ;RST=1
        LCALL   RTCDL
        MOV     B,#8        ;SEND  COMMAND
        CLR     C
BYTEWR1: MOV    A,R6
        RRC     A
        MOV     R6,A
        MOV     P1.4,C
        LCALL   CLKWR
        DJNZ    B,BYTEWR1
        MOV     B,#8        ;SEND  DATA
        CLR     C
BYTEWR2: MOV    A,R7
        RRC     A
        MOV     R7,A
        MOV     P1.4,C
        LCALL   CLKWR
        DJNZ    B,BYTEWR2
        CLR     P1.6        ;RST=0
        LCALL   RTCDL
        RET
CLKWR:  SETB    P1.5
        LCALL   RTCDL
        CLR     P1.5
        LCALL   RTCDL
        RET
RTCDL:  MOV     R3,#4       ;DELAY
        DJNZ    R3,$
        RET
```

ซอร์สโปรแกรมของ Slave Control Board

```
; Program for slave-board
; recieve data from master-board
; Program running message
; Date-Time-Temperature-Humidity
; PROGRAMMER     VIRASAK.T
; INSTRUMENT / KMITL
ORG         0000H
LCALL       MAIN

CONA        EQU         0E000H
CONB        EQU         0E001H
CONC        EQU         0E002H
CONT        EQU         0E003H
READY       EQU         0A000H
TABLE1      EQU         1650H  ; TIME CALEENDAR
TABLE2      EQU         1690H  ;Message
TABLEH      EQU         1700H
REG         EQU         0028H

; ********* INTERRUPT VECTOR ********

            ORG     0023H

            PUSH    ACC
            PUSH    PSW
            JB      RI,$+6
            JBC     TI,ENDSER
            MOV     A,SBUF
            CJNE    A,#'@',ENDSER
            JNB     RI,$
            CLR     RI
            MOV     A,SBUF
            CJNE    A,#'&',SAS1
            LCALL   TIMECHK
            JMP     ENDSER
```

```
SAS1:    CJNE    A,#'*',ENDSER
         LCALL   MESSAGE
ENDSER:  POP     PSW
         POP     ACC
         RETI
SEND:    MOV     SCON,#50H
         MOV     PCON,#00H
         MOV     TMOD,#20H
         MOV     TH1,#0FDH
         MOV     IE,#00H
         SETB    TR1
         MOV     DPTR,#TABLET
         CLR     A
         SETB    TI  ;ADD
L00P:    MOVX    A,@DPTR
         CJNE    A,#'~',DAX
         SETB    EA
         SETB    ES
         RET
         JMP     ENDSER
DAX:     JNB     TI,$
         CLR     TI
         MOV     A,SBUF
         INC     DPTR
         CLR     SBUF
         SJMP    L00P

; *********  Start main program  *********

         ORG     0100H

MAIN:    MOV     SCON,#52H
         MOV     PCON,#00H
         MOV     TMOD,#20H
         MOV     TH1,#0FDH
         SETB    EA
```

```
        SETB    ES
        SETB    PS
        SETB    REN
        SETB    TR1
        CLR     SM2
        LCALL   CLEAR
        LCALL   CLEAR1
        LCALL   MAIN1
        LCALL   CLEAR
        LCALL   CLEAR1
        LCALL   DISP1

WDOG:   MOV     DPTR,#READY
        MOV     A,#11H
        MOVX    @DPTR,A
        SJMP    WDOG
TIMECHK: MOV    DPTR,#TABLE1
LOOP0:  CJNE    A,#0AH,THEN0
        LCALL   CLEAR1
        LCALL   CLEAR
        LCALL   SHOWTIME
        LCALL   CLEAR1
        LCALL   DISP1
        LCALL   CLEAR1
        LCALL   TEMP
        LCALL   CLEAR1
        LCALL   DISP1
        RET
THEN0:  MOVX    @DPTR,A
        CLR     SBUF
        CLR     A
        INC     DPTR
        JNB     RI,$
        CLR     RI
        MOV     A,SBUF
        JMP     LOOP0
```

```
MESSAGE: MOV     DPTR,#TABLE2
  LOOP2: CJNE    A,#'$',THEN2
         LCALL   CLEAR1
         LCALL   CLEAR
         LCALL   ALL
         SETB    21H.0
         LCALL   DISP
         LCALL   CLEAR
         LCALL   CLEAR1
         LCALL   DISP1
         RET
  THEN2: MOVX    @DPTR,A
         CLR     SBUF
         CLR     A
         INC     DPTR
         JNB     RI,$
         CLR     RI
         MOV     A,SBUF
         JMP     LOOP2
CALENDAR: LCALL  CLEAR
         LCALL   CLEAR1
         LCALL   SHOWCAL
         LCALL   ALL
         LCALL   DISP0
         LCALL   REAR1
         LCALL   CLEAR
         LCALL   CLEAR1
         LCALL   DISP1
         RET
  CLEAR: CLR     A
         MOV     R1,A
         MOV     R0,A
         MOV     R2,A
         MOV     R3,A
         RET
```

```
CLEAR1: CLR     A
        CLR     B
        MOV     20H,#00H
        MOV     21H,A
        MOV     22H,A
        MOV     23H,A
        RET

CLEARt: MOV     DPTR,#TABLETIME    ; --:--:--
        CLR     A
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        INC     DPTR
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    @DPTR,A
        RET

REAR1:  MOV     DPTR,#TABLECAL+9
        MOV     R0,#20H
WAB:    CLR     A
        MOVX    @DPTR,A
        INC     DPTR
        DJNZ    R0,WAB
        RET

; ++++++++++ TEMP ++++++++++
```

```
; P1.0 = DQ
; P1.1 = CLK\CONV\
; P1.2 = RST\

MAIN1:  MOV     R7,#0CH     ;WRITE CONFIG
        MOV     R6,#82H     ;SHOT=0
        LCALL   WDAT16
        LCALL   DELAYX
        MOV     R7,#0EEH
        MOV     R6,#00H
        LCALL   WDAT16
        LCALL   DELAYX

MAIN2:  LCALL   DDATA
        LCALL   DDATA
        RET

;++++++++ WDATA SUB ++++++++++
; WDATA SUBRUTINE

WDAT16: CLR     P1.0        ;CLR DATA
        LCALL   DELAY1
        SETB    P1.2
        LCALL   DELAY1
        MOV     A,R7        ;
        LCALL   WDATAX
        MOV     A,R6        ;
        LCALL   WDATAX
        LCALL   DELAY1
        CLR     P1.2
        RET

WDATAX: MOV     B,#08H          ;WRITE 8 BIT DATA IN=A)
WDATAX1:RRC     A
        MOV     P1.0,C
        LCALL   CLKR
```

```
        DJNZ    B,WDATAX1
        RET

; +++++++++++ RDATA SUB +++++++++++
; RAED DATA SUBRUTINE

RDATA:  CLR     P1.0
        LCALL   DELAY1
        SETB    P1.2
        LCALL   DELAY1
        MOV     A,R7
        LCALL   WDATAX

        SETB    P1.0
        MOV     B,#08H
RDATA3: CLR     P1.1
        LCALL   DELAY1
        MOV     C,P1.0      ;READ DATA
        RRC     A
        SETB    P1.1
        LCALL   DELAY1
        DJNZ    B,RDATA3
        MOV     R6,A
RDATA8: MOV     B,#08H
RDATA7: CLR     P1.1
        LCALL   DELAY1
        MOV     C,P1.0
        RRC     A
        SETB    P1.1
        LCALL   DELAY1
        DJNZ    B,RDATA7
        MOV     REG,A
        LCALL   DELAY1
        CLR     P1.2
        RET
; +++++++++++ CLKF SUB +++++++++++
```

```
        ; A FALLING ADGE

CLKF:   SETB    P1.1
        LCALL   DELAY1
        CLR     P1.1
        LCALL   DELAY1
        RET
; +++++++++ CLKR SUB ++++++++++
; A RISING ADGE
CLKR:   CLR     P1.1
        LCALL   DELAY1
        SETB    P1.1
        LCALL   DELAY1
        RET
; ++++++++++ DDATA SUB ++++++++++
; DISPLAY DATA SUBRUTINE
DDATA:  MOV     R7,#0AAH
DDATA1: LCALL   RDATA
        MOV     A,R6        ;LSB DATA
        LCALL   DIV
        MOV     DPL,A
        MOV     A,REG
        LCALL   DIV
        MOV     DPH,A
        LCALL   HTOD
        MOV     A,R3
        MOV     R1,#00H
        MOV     R2,#00H
        MOV     R3,#00H
        LCALL   HTOA        ;ASCII
        MOV     A,R2
        MOV     DPTR,#TABLET+2
        MOVX    @DPTR,A
        INC     DPTR
        CLR     A
        MOV     A,R3
```

```
        MOVX    @DPTR,A      ;TABLET: " T1T2 "
        CLR     A
        MOV     R0,A
        MOV     R1,A
        MOV     R2,A
        MOV     R3,A
        CLR     A
        MOV     R4,A
        MOV     R5,A
        MOV     R6,A
        MOV     R7,A
        RET

; ++++++++++ DIV SUB ++++++++++
; DIV SUBRUTINE
; REG = A,B
DIV:    MOV     B,#2
        DIV     AB
        RET

; ++++++++++ DELAY SUB ++++++++++
; DELAY SUBRUTINE
; REG = R1,R4

DELAY1: MOV     R4,#2
        DJNZ    R4,S
        RET

DELAYX: MOV     R4,#0
DELX1:  MOV     R1,#0
        DJNZ    R1,S
        DJNZ    R4,DELX1
        RET

; ********* S1S2 - M1M2 - H1H2 *********
```

```
SHOWTIME: MOV     DPTR,#TABLE1+3   :
          MOVX    A,@DPTR
          PUSH    ACC
          CLR     A

          MOV     DPTR,#TABLE1+4
          MOVX    A,@DPTR
          PUSH    ACC
          CLR     A

          MOV     DPTR,#TABLE1+5
          MOVX    A,@DPTR
          PUSH    ACC
          CLR     A

          MOV     DPTR,#TABLE1+6
          MOVX    A,@DPTR
          PUSH    ACC
          CLR     A

          MOV     DPTR,#TABLE1+7
          MOVX    A,@DPTR
          PUSH    ACC
          CLR     A

          MOV     DPTR,#TABLE1+8
          MOVX    A,@DPTR
          PUSH    ACC
          CLR     A
          MOV     DPTR,#TABLETIME+1
          POP     ACC
          MOVX    @DPTR,A        ;__H2:____:____
          CLR     A
          MOV     DPTR,#TABLETIME
          POP     ACC
```

```
        MOVX    @DPTR,A         ;H1H2:____:____
        CLR     A
        MOV     DPTR,#TABLETIME+4  ;4
        CLR     A
        POP     ACC
        MOVX    @DPTR,A         ;H1H2:__M2:____
        MOV     DPTR,#TABLETIME+3  ;3
        CLR     A
        POP     ACC
        MOVX    @DPTR,A         ;H1H2:M1M2:____
        MOV     DPTR,#TABLETIME+7  ;7
        CLR     A
        POP     ACC
        MOVX    @DPTR,A         ;H1H2:M1M2:__S2
        MOV     DPTR,#TABLETIME+6  ;6
        CLR     A
        POP     ACC
        MOVX    @DPTR,A         ;H1H2:M1M2:S1S2
        LCALL   CLEAR
        RET

GONNA:  CJNE    A,#'4',BAX
        MOV     DPTR,#TABLE1+6
        MOVX    A,@DPTR
        CJNE    A,#'0',BAX
        LCALL   CALENDAR
BAX:    RET

; ********* STORE DATA TEMP TO TEMP TABLE *********

TEMP:   MOV     DPTR,#TABLE1+3  ; S1
        MOVX    A,@DPTR
        CJNE    A,#'0',NON
        CLR     A
        INC     DPTR    ;S2
        MOVX    A,@DPTR
```

```
        CJNE    A,#'0',NON
        CLR     A
        INC     DPTR    ;M1
        MOVX    A,@DPTR
        CJNE    A,#'0',GONNA
        SJMP    CLK
        RET
CLK:    CLR     A
        INC     DPTR    ;M2
        MOVX    A,@DPTR
        CJNE    A,#'0',NON
        CLR     A
        INC     DPTR    ;H1
        MOVX    A,@DPTR
        MOV     R2,A
        CLR     A
        INC     DPTR    ;H2
        MOVX    A,@DPTR
        MOV     R3,A
        LCALL   ATOH
        MOV     R2,#00H
        MOV     R3,#00H
        CJNE    A,#06H,CHKT1
        LCALL   GOGO
NON:    RET
CHKT1:  CJNE    A,#07H,CHKT2
        LCALL   GOGO
        RET

GOGO:   MOV     DPTR,#TABLET+2
        MOVX    A,@DPTR
        PUSH    ACC
        INC     DPTR
        MOVX    A,@DPTR
        PUSH    ACC
        MOV     DPTR,#TABLEtemp+17
```

```
        CLR     A
        POP     ACC
        MOVX    @DPTR,A
        MOV     DPTR,#TABLEtemp+16
        POP     ACC
        MOVX    @DPTR.A
        CLR     A
        LCALL   ALL
        LCALL   CLEAR1
        SETB    23H.4
        LCALL   DISt
        LCALL   CLEAR1
        LCALL   CLEAR
        LCALL   SEND
        RET
; ******** DAILY - YEAR - MONTH - DATE ********
SHOWCAL: MOV    DPTR,#TABLE1+15  : YYMMDDII YY
        CLR     A
        MOVX    A,@DPTR
        CJNE    A,#'0',BACK
        INC     DPTR        +16
        MOVX    A,@DPTR
CHECK1x: CJNE   A,#'1',CHECK2x
        MOV     DPTR,#TABLE3        : _MONDAY.
        LCALL   STORE6D
        MOV     DPTR,#TABLE1+13
        MOVX    A,@DPTR
        MOV     DPTR,#TABLECAL+17
        MOVX    @DPTR,A
        CLR     A
        MOV     DPTR,#TABLE1+14
        MOVX    A,@DPTR
        MOV     DPTR,#TABLECAL+18
        MOVX    @DPTR,A
        CLR     A
        LCALL   CHECK1m
```

```
BACK:     RET

CHECK2X:  CJNE      A,#'2',CHECK3x
          MOV       DPTR,#TABLE4          ; _TUESDAY,
          LCALL     STORE7D
          MOV       DPTR,#TABLE1+13
          MOVX      A,@DPTR
          MOV       DPTR,#TABLECAL+18
          MOVX      @DPTR,A
          CLR       A
          MOV       DPTR,#TABLE1+14
          MOVX      A,@DPTR
          MOV       DPTR,#TABLECAL+19
          MOVX      @DPTR,A
          CLR       A
          LCALL     CHECK1m
          RET

STORE6D:  MOVX      A,@DPTR     : TABLE3  TABLE7  TABLE9
          PUSH      ACC
          CLR       A
          INC       DPTR
          MOVX      A,@DPTR     : _M      _F      _S
          PUSH      ACC
          CLR       A
          INC       DPTR
          MOVX      A,@DPTR     ; _MO     _FR     _SU
          PUSH      ACC
          CLR       A
          INC       DPTR
          MOVX      A,@DPTR     ; _MON    _FRI    _SUN
          PUSH      ACC
          CLR       A
          INC       DPTR
          MOVX      A,@DPTR     ; _MOND   _FRID   _SUND
          PUSH      ACC
```

```
CLR     A
INC     DPTR
MOVX    A,@DPTR     ;_MONDA    _FRIDA    _SUNDA
PUSH    ACC
CLR     A
INC     DPTR
MOVX    A,@DPTR     ; _MONDAY  _FRIDAY   _SUNDAY
PUSH    ACC
CLR     A
INC     DPTR
MOVX    A,@DPTR     :  _FRIDAY,  _SUNDAY,
PUSH    ACC

MOV     DPTR,#TABLECAL+16   : _123456,
POP     ACC
MOVX    @DPTR,A
CLR     A
MOV     DPTR,#TABLECAL+15   : ______Y,
POP     ACC
MOVX    @DPTR,A
CLR     A
MOV     DPTR,#TABLECAL+14   : _____AY,
POP     ACC
MOVX    @DPTR,A
CLR     A
MOV     DPTR,#TABLECAL+13   : ____DAY,
POP     ACC
MOVX    @DPTR,A
CLR     A
MOV     DPTR,#TABLECAL+12   : ___NDAY,
POP     ACC
MOVX    @DPTR,A
CLR     A
MOV     DPTR,#TABLECAL+11   : __ONDAY,
POP     ACC
MOVX    @DPTR,A
```

```
                CLR     A
                MOV     DPTR,#TABLECAL+10   ; MONDAY,
                POP     ACC
                MOVX    @DPTR,A
                CLR     A
                MOV     DPTR,#TABLECAL+9    ; _MONDAY,
                POP     ACC
                MOVX    @DPTR,A
                CLR     A
                RET

CHECK1m:        MOV     DPTR,#TABLE1+11
                MOVX    A,@DPTR
                MOV     R2,A
                CLR     A
                INC     DPTR        ;+12
                MOVX    A,@DPTR
                MOV     R3,A
                CLR     A
                LCALL   ATOH
                SJMP    CHECKm
                RET
CHECKm:         CJNE    A,#01H,CHECK2m
                MOV     DPTR,#TABLEm1   ;,JANUARY, (,1234567,)
                LCALL   STOREm7
                RET

STOREm4:        MOV     A,DPL
                MOV     R1,#06H
                ADD     A,R1
                MOV     DPL,A
                CLR     A
                MOV     R1,A
                MOVX    A,@DPTR         ;,
                PUSH    ACC
```

```
        LCALL   QUITE
        MOVX    A,@DPTR         :,1
        PUSH    ACC

        LCALL   QUITE
        MOVX    A,@DPTR         :,12
        PUSH    ACC

        LCALL   QUITE
        MOVX    A,@DPTR         :,123
        PUSH    ACC

        LCALL   QUITE
        MOVX    A,@DPTR         :,1234
        PUSH    ACC

        LCALL   QUITE
        MOVX    A,@DPTR         :,1234,
        PUSH    ACC

        LCALL   QUITE
        MOVX    A,@DPTR         :,1234,`
        PUSH    ACC

        MOV     DPTR,#TABLE1+16
        MOVX    A,@DPTR
BD1:    CJNE    A,#'1',BD2
        MOV     DPTR,#TABLECAL+19
        LCALL   RESTORE4m
        RET
BD2:    CJNE    A,#'2',BD3
        MOV     DPTR,#TABLECAL+20
        LCALL   RESTORE4m
        RET
BD3:    CJNE    A,#'3',BD4
        MOV     DPTR,#TABLECAL+22
```

```
        LCALL   RESTORE4m
        RET
BD4:    CJNE    A,#'4',BD5
        MOV     DPTR,#TABLECAL+21
        LCALL   RESTORE4m
        RET
BD5:    CJNE    A,#'5',BD6
        MOV     DPTR,#TABLECAL+19
        LCALL   RESTORE4m
        RET
BD6:    CJNE    A,#'6',BD7
        MOV     DPTR,#TABLECAL+21
        LCALL   RESTORE4m
        RET
BD7:    CJNE    A,#'7',BD8
        MOV     DPTR,#TABLECAL+19
        LCALL   RESTORE4m
BD8:    RET
RESTORE4m: POP  ACC
        POP     ACC
HK4:    POP     ACC
        CJNE    A,#'`',EDY4
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#TABLE1+9
        MOVX    A,@DPTR
        POP     DPL
        POP     DPH
        MOVX    @DPTR,A
        INC     DPTR
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#TABLE1+10
        CLR     A
        MOVX    A,@DPTR
```

```
        POP     DPL
        POP     DPH
        MOVX    @DPTR,A
        LCALL   CLEAR
        MOV     A,#'.'
        INC     DPTR
        MOVX    @DPTR,A
        RET
EDY4:   MOVX    @DPTR,A
        CLR     A
        INC     DPTR
        SJMP    HK4
        RET
QUITE:  CLR     A
        MOV     A,DPL
        DEC     A
        MOV     DPL,A
        CLR     A
        RET

DISP:   MOV     R0,#03H     ; MESSAGE
DIS:    LCALL   CLEAR1
        SETB    B.1
        SETB    21H.0
        LCALL   DISPLAY
        CLR     A
        LCALL   DISP2
        LCALL   DISP3
        LCALL   DISP4
        LCALL   DISP5
        LCALL   DISP6
        DJNZ    R0,DIS
        LCALL   CLEAR1
        MOV     DPTR,#TABLE2+2
        MOV     R1,#38H
REC:    CLR     A
```

```
        MOVX    @DPTR,A
        INC     DPTR
        DJNZ    R1,REC
        RET

DISt:   LCALL   CLEAR1      ; SHOW TEMP
        MOV     R0,#03H
DIt:    LCALL   CLEAR1
        SETB    B.1
        SETB    23H.4
        LCALL   DISPLAY
        CLR     A
        LCALL   DISP2
        LCALL   DISP3
        LCALL   DISP4
        LCALL   DISP5
        LCALL   DISP6
        DJNZ    R0,DIt
        LCALL   CLEAR1
        RET

DISP0:  LCALL   CLEAR       ; CALENDAR
        MOV     R0,#03H
DIS0:   LCALL   CLEAR1
        SETB    22H.0
        SETB    B.1
        LCALL   DISPLAY
        CLR     A
        LCALL   DISP2
        LCALL   DISP3
        LCALL   DISP4
        LCALL   DISP5
        LCALL   DISP6
        DJNZ    R0,DIS0
        LCALL   CLEAR1
        RET
```

```
DISP1:  LCALL   CLEAR1   ;SHOW TIME
        SETB    20H.7
        SETB    B.1
        MOV     A,#80H
        LCALL   DIGIT8
        LCALL   DIGIT8
        RET
DISPLAY1: LCALL CONTROL
        LCALL   SEGMENT
        PUSH    DPH
        PUSH    DPL
        MOV     A,#00H
        LCALL   NEXT0
RUNN1:  POP     DPL
        POP     DPH
        MOVX    @DPTR,A
        CLR     A
        LCALL   DELAY
        RET
CONTROL: MOV    DPTR,#CONT
        MOV     A,#80H
        MOVX    @DPTR,A
        CLR     A
        RET

SEGMENT: MOV    DPTR,#CONB
        MOV     A,R4
        ANL     A,#0FFH
        MOVX    @DPTR,A
        INC     DPTR
        RET

        END
```

ซอร์สโปรแกรม Visual Basic

Aboutform.frm

```
Sub Command1_Click ()
    Unload Aboutdem
End Sub

Sub Form_Load ()
    ' Set position form
    Left = (Screen.Width - Width) / 2
    Top = (Screen.Height - Height) / 2
    Image1.Left = 880
    Image1.Top = 1500
    Image2.Left = 880
    Image2.Top = 1500
End Sub

Sub Timer1_Timer ()
    If Timer1.Interval <> 2000 Then
        Timer1.Interval = 2000
        Image2.ZOrder
    Else
        Timer1.Interval = 500
        Image2.ZOrder 1
        Image1.ZOrder
    End If
End Sub
```

DataViewform.frm

```
' declaration variable
 Dim HWidth, HHeight, I, Mssg
 Const CF_TEXT = 1

Sub Form_Load ()
    DataView.FullPath.Text = Openfile.FullPath.Text
    Caption = DataView.FullPath.Text
    Form_Resize
End Sub

Sub Form_Activate ()
     If FullPath.Text = " " Then
       Unload DataView
     Else
      ' set Text1 inside form
      Form_Resize
     ' Initialization
       NL$ = Chr$(13) + Chr$(10)
       T$ = Space$(32000)
       Ndx& = 1
      ' Load file to be displayed
       Open FullPath.Text For Input As #1
       ' Load file up to 32000 bytes
         Do Until EOF(1)
           Line Input #1, a$
           a$ = Space$(2) + a$ + NL$
          If Ndx& + Len(a$) >= 32000 Then
              Exit Do
```

```
            Else
                Mid$(T$, Ndx&, Len(a$)) = a$
                Ndx& = Ndx& + Len(a$)
            End If
        Loop
        ' close file
        Close #1
        ' set caption of this form based on file listed
        Caption = FullPath.Text
        ' display file contents in the Text Box
        Text1.Text = RTrim$(T$)
    End If
End Sub

Sub Form_Resize ()
    ' Adjust the Text box size in form
    Text1.Move 0, 0, ScaleWidth, ScaleHeight
End Sub

Sub menAboutCalen_Click ()
    ' Load AboutForm
    Aboutdem.Show MODAL
End Sub

Sub menCopy_Click ()
    ' Copy cut text to Clipboard
    Clipboard.SetText Text1.SelText
End Sub

Sub menCut_Click ()
```

```
    ' Recieved value from user
     Work$ = Text1.Text
     Wstart% = Text1.SelStart
     Wlength% = Text1.SelLength
    ' Copy message to clipboard
     Clipboard.SetText Mid$(Work$, Wstart% + 1, Wlength%)
    ' cut message for out to
     Work$ = Left$(Work$, Wstart%) + Mid$(Work$, Wstart% + Wlength% + 1)
     Text1.Text = Work$
    ' position of curser
    Text1.SelStart = Wstart%
End Sub

Sub menExit_Click ()
    If Caption = "[Untitle]" And Text1.Text <> " " Then
        Msg$ = " Do you want save this file?"
       M% = MsgBox(Msg$, 4, "DataView")
         If M% = 6 Then
           SSavefile.Show MODAL
           Unload DataView
         Else
           Unload DataView
        End If
     Else
       ' Exit DataView form
        Unload DataView
     End If
End Sub

Sub menHowtouse_Click ()
```

```
    ' Load Howtouse form
    FileMsg "USEMSG.MSG", 9
End Sub

Sub menNew_Click ()
    If FileName = NewFile Then
       DataView.Caption = "[Untitle]"
       DataView.Text1.Text = " "
       Exit Sub
    Else
      ' Wraning User for save file was used
       If Text1.Text <> " " Then
       Msg$ = " Do you want save this file"
       N% = MsgBox(Msg$, 4, "DataView")
          If N% = 4 Then
             Exit Sub
          End If
       End If
    End If
    ' Make New file
     DataView.Caption = "[Untitle]"
     DataView.Text1.Text = " "
End Sub

Sub menOpen_Click ()
    ' Wraning User for save file was used
     If Caption <> " " Or NewFile Then
        Msg$ = " Do you want save this file"
        Msg$ = Msg$ + "Click OK for save this file"
          M% = MsgBox(Msg$, (4 + 32), "DataView")
```

```
        If   M% = 7 Then
             ' open Form work
              Openfile.Caption = "Select a file for view"
              Openfile.Show MODAL
             ' Load file
                If Openfile.FullPath.Text <> " " Then
                   DataView.FullPath.Text = Openfile.FullPath.Text
                   Text1.Text = " "
                End If
          Else
             MsgBox (" Pleas select 'save' from menu File")
             Exit Sub
          End If
    End If
End Sub
Sub menPaste_Click ()
   ' Get working parameters
    Work$ = Text1.Text
    Wstart% = Text1.SelStart
    Wlength% = Text1.SelLength
   ' Cut out text, if any, and insert Clipboard text
    Work$ = Left$(Work$, Wstart%) + Clipboard.GetText() + Mid$(Work$, Wstart% +
Wlength% + 1)
    Text1.Text = Work$
    ' Position edit cursor
     Text1.SelStart = Wstart%
    ' Set focus back to text box
    Text1.SetFocus
End Sub
```

```
Sub menPrint_Click ()
    If Text1.Text <> " " Then
       Printer.Print Text1.Text
       Printer.NewPage
       Printer.EndDoc
    Else
       Msg = MsgBox("Not Doccument")
       Exit Sub
    End If
End Sub

Sub menPrintsetup_Click ()
    CMDialog1.Flags = &H40&
    CMDialog1.Action = 5
End Sub

Sub menReadData_Click ()
   ' Mainform.Comm2.InputLen = 0
   ' Caption = "[Untitle]"
   NL$ = Chr$(13) + Chr$(10)
   Mainform.Comm2.Output = "@" + "!" + NL$
   If Mainform.Comm2.InBufferCount Then
         DT$ = Mainform.Comm2.Input
         Text1.Text = DT$
       End If
      Print Text1.Text
End Sub

Sub menSave_Click ()
```

```
' Check Title for new file
  If DataView.Caption <> FullPath Then
    SSavefile.Show MODAL
  Else
    Open Caption For Output As #1
      Print #1, DataView.Text1.Text
    Close #1
  End If
End Sub

Sub menSaveAs_Click ()
  SSavefile.Show MODAL
End Sub
```

Howtouseform.frm

```
Sub menExit_Click ()
  'Exit Howtouseform
  Unload Howtouseform
End Sub

Sub menTopcHumidity_Click ()
    FileMsg "USEMSG.MSG", 2
End Sub

Sub menTopicData_Click ()
    ' show Datausing message
    FileMsg "USEMSG.MSG", 7
End Sub
```

```
Sub menTopicDate_Click ()
    ' show data using message
    FileMsg "USEMSG.MSG", 5
End Sub

Sub menTopicExit_Click ()
    ' show Exit using message
    FileMsg "USEMSG.MSG", 8
End Sub

Sub menTopicMessage_Click ()
   ' show data using message
    FileMsg "USEMSG.MSG", 6
End Sub

Sub menTopicTemp_Click ()
     FileMsg "USEMSG.MSG", 3
End Sub

Sub menTopicTime_Click ()
     FileMsg "USEMSG.MSG", 4
End Sub

Sub menUsing_Click ()
     FileMsg "FILEMSG.MSG", 1
End Sub
```

Humidityform.frm

```
Sub Form_Load ()
    NL$ = Chr$(13) + Chr$(10)
  ' send command to MCS-8051
    If Mainform.Comm2.InBufferCount Then
       ' Read the "OK" response data in the serial port.
        InString$ = Mainform.Comm2.Input
        HumVal$ = Mid$(InString$, 19, 2)
        Humidityfrm.Gauge1.Value = HumVal$
        Humidityfrm.Label1.Caption = "Percentage of humidity is " & HumVal$ & "%"
    End If
End Sub

Sub menAbout_Click ()
    Aboutdem.Label1.Caption = "HUMIDITY SHOW"
    Aboutdem.Show MODAL
End Sub

Sub Timer1_Timer ()
     NL$ = Chr$(13) + Chr$(10)
      If Mainform.Comm2.InBufferCount Then
        ' Read the "OK" response data in the serial port.
         InString$ = Mainform.Comm2.Input
         HumVal$ = Mid$(InString$, 19, 2)
         Humidityfrm.Gauge1.Value = HumVal$
         Humidityfrm.Label1.Caption = "Percentage of humidity is " & HumVal$ & "%"
     End If
 End Sub

 Sub Timer2_Timer ()
     Timer1.Enabled = False
```

```
End Sub

Sub menExit_Click ()
    Unload Humidityfrm
End Sub
```

Mainform.frm

```
Const PORT = 2

Sub CmdData_Click ()
    menData_Click
End Sub

Sub CmdDate_Click ()
    mendate_Click
End Sub

Sub CmdExit_Click ()
    Unload Mainform
End Sub

Sub CmdHumidity_Click ()
    Humidityfrm.Show MODELESS
End Sub
Sub CmdMessage_Click ()
    menMessage_Click
End Sub

Sub CmdTemp_Click ()
```

```
    menTemp_click
End Sub

Sub CmdTime_click ()
    menTime_click
End Sub

Sub Digital_DblClick ()
    msg = " Do you want setting time?"
    msg = MsgBox(msg, (32 + 1), "Edit Time")
    If msg = 1 Then
     menTime_click
    Else
      Exit Sub
    End If
End Sub

Sub Form_Load ()
    NL$ = Chr$(13) + Chr$(10)
   ' show form
    Show
   ' set parameter for communication
   SetFocus
   Comm2.CommPort = PORT
   Comm2.Settings = "9600,N,8,1"
   Comm2.InputLen = 0
   Comm2.PortOpen = True
   Left = 0
   Top = 960
    Form_Resize
```

```
End Sub

Sub Form_Resize ()
    Frame3D1.Move 0, 0, ScaleWidth, ScaleHeight
End Sub

Sub LabelDate_DblClick ()
    mendate_Click
End Sub

Sub LabelTemp_Click ()
    NL$ = Chr$(13) + Chr$(10)
    If Comm2.InBufferCount Then
      ' Read the "OK" response data in the serial port.
      InString$ = Comm2.Input
      TTmpVal$ = Mid$(InString$, 17, 2)
      Tempfrm.Label1.Caption = "Now is " & TTmpVal$ & "'C "
      LabelTemp.Caption = "Now is " & TTmpVal$ & "'C "
    End If
End Sub

Sub menAboutcalendar_Click ()
   ' Load Aboutdem show
   Aboutdem.Show MODAL
End Sub

Sub menData_Click ()
    ' Open file for work
     Openfile.Caption = "Select file to view"
     Openfile.Show MODALESS
```

```
    ' Show file
    If Openfile.FullPath.Text <> " " Then
        DataView.FullPath.Text = Openfile.FullPath.Text
    End If
End Sub

Sub mendate_Click ()
    Dim Today
    NL$ = Chr$(13) + Chr$(10)
    If Comm2.InBufferCount Then
        DDate$ = Mainform.Comm2.Input
        NNowdate$ = Mid$(DDate$, 15, 2) + "/"
        NNow1date$ = NNowdate$ + Mid$(DDate$, 13, 2) + "/"
        NNow2date$ = NNow1date$ + Mid$(DDate$, 11, 2) + "/"
        NNow3date$ = NNow2date$ + "19" + Mid$(DDate$, 9, 2)
        NNow4date$ = Mid$(NNow3date$, 3, 11)
            If Mid$(NNow3date$, 1, 2) = "01" Then
              NNow5date$ = "Sun" + NNow4date$
            ElseIf Mid$(NNow3date$, 1, 2) = "02" Then
              NNow5date$ = "Mon" + NNow4date$
            ElseIf Mid$(NNow3date$, 1, 2) = "03" Then
              NNow5date$ = "Tue" + NNow4date$
            ElseIf Mid$(NNow3date$, 1, 2) = "04" Then
              NNow5date$ = "Wed" + NNow4date$
            ElseIf Mid$(NNow3date$, 1, 2) = "05" Then
              NNow5date$ = "Thu" + NNow4date$
            ElseIf Mid$(NNow3date$, 1, 2) = "06" Then
              NNow5date$ = "Fri" + NNow4date$
            ElseIf Mid$(NNow3date$, 1, 2) = "07" Then
              NNow5date$ = "Sat" + NNow4date$
```

```
        End If
    End If
        DDm$ = " Enter  New  Value  follow  this form 'Ddd/dd/mm/yyyy'"
        NewDate$ = InputBox$(DDm$, "Edit Date", NNow5date$)
        LabelDate.Caption = NewDate$
        NNewdate$ = Mid$(NewDate$, 13, 2)
        NNewdate$ = NNewdate$ + Mid$(NewDate$, 8, 2)
        NNewdate$ = NNewdate$ + Mid$(NewDate$, 5, 2)
        NewDdate$ = Mid$(NewDate$, 1, 3)
            If NewDdate$ = "Sun" Then
              Comm2.Output = "@" + "#" + NNewdate$ + "01" + "$"
            ElseIf NewDdate$ = "Mon" Then
              Comm2.Output = "@" + "#" + NNewdate$ + "02" + "$"
            ElseIf NewDdate$ = "Tue" Then
              Comm2.Output = "@" + "#" + NNewdate$ + "03" + "$"
            ElseIf NewDdate$ = "Wed" Then
              Comm2.Output = "@" + "#" + NNewdate$ + "04" + "$"
            ElseIf NewDdate$ = "Thu" Then
              Comm2.Output = "@" + "#" + NNewdate$ + "05" + "$"
            ElseIf NewDdate$ = "Fri" Then
              Comm2.Output = "@" + "#" + NNewdate$ + "06" + "$"
            ElseIf NewDdate$ = "Sat" Then
              Comm2.Output = "@" + "#" + NNewdate$ + "07" + "$"
            Else
              msg = "Incorrect Data.. Please Click Date button for new data..If you want
set? "
               msg = MsgBox(msg, 64, "Error")

              Exit Sub
           End If
```

```
        Today = NewDate$ ' Get current date and time.
        Mainform.Label2.Caption = "Today is " & Format(Today, "dddd,mmmm dd,
yyyy")
End sub

Sub menExit_Click ()
    ' Exit all program
    Unload Tempfrm
    Unload Humidityfrm
    Unload Mainform
End Sub

Sub menHowtouse_Click ()
    ' Load How to use form show
    Howtouseform.Show MODAL
End Sub

Sub menHumidity_Click ()
    NL$ = Chr$(13) + Chr$(10)
   If Mainform.Comm2.InBufferCount  Then
      ' Read the "OK" response data in the serial port.
      InString$ = Mainform.Comm2.Input
      HumVal$ = Mid$(InString$, 19, 2)
      Humidityfrm.Gauge1.Value = HumVal$
      Humidityfrm.Label1.Caption = "Percentage of humidity is " & HumVal$ & "%"
    Else
      Humidityfrm.Label1.Caption = "Not received data"
    End If
     Humidityfrm.Show MODELESS
End Sub
```

```
Sub menMessage_Click ()
    Timer2.Enabled = False
    Mainform.Comm2.Output = "@" + "%' Chr$(13)"
    DataMessage$ = Mainform.Comm2.Input
    Messagefrm.Text1.Text = DataMessage$
    Messagefrm.Show MODELESS
End Sub

Sub menTemp_click ()
    NL$ = Chr$(13) + Chr$(10)
    If Comm2.InBufferCount Then
      ' Read the "OK" response data in the serial port.
       InString$ = Comm2.Input
       TTmpVal$ = Mid$(InString$, 17, 2)
       Tempfrm.Gauge2.Value = TTmpVal$
       Tempfrm.Label1.Caption = "Now tempurature is " & TTmpVal$ & "'C "
       LabelTemp.Caption = "Now is " & TTmpVal$ & "'C "
    End If
       Tempfrm.Show MODELESS
End Sub

Sub menTime_click ()
    Dim msg, NowTime$, NewTime$    ' Declare variables.
    Timer3.Enabled = False
    NL$ = Chr$(13) + Chr$(10)
       If Mainform.Comm2.InBufferCount Then
        NowTime$ = Mainform.Comm2.Input
          If Len(NowTime$) < 21 Then
            Digital.Cls
```

```
        Else
          Digital.Cls
          NNowTime$ = Mid$(NowTime$, 7, 2) + ":"
          NNow1Time$ = NNowTime$ + Mid$(NowTime$, 5, 2) + ":"
          NNow2Time$ = NNow1Time$ + Mid$(NowTime$, 3, 2)
        End If
    End If
        msg = "Enter a new time in the form hh:mm:ss"
        NewTime$ = InputBox$(msg, "Time Set", NNow2Time$)   ' Get user input.
        If Len(NewTime) > 0 Then    ' Check if valid.
          NNow2Time$ = NewTime$  ' Set time.
          msg = "System time now set to "
          msg = msg & NNow2Time$ & "."  ' Put time in message.
          Digital_Click
        Else
          Digital_Click
          Exit Sub
        End If
      Digit$ = Mid$(NewTime$, 7, 2)
      Digit$ = Digit$ + Mid$(NewTime$, 4, 2)
      Digit$ = Digit$ + Mid$(NewTime$, 1, 2)
     Mainform.Comm2.Output = "@" + "&" + Digit$ + "$"
End Sub
Sub Timer1_Timer ()
    ' Keep track of current second
    Static LastSecond
    ' Are we into a new second yet?
    If Second(Now) = LastSecond Then
      Exit Sub
    Else
```

```
    LastSecond = Second(Now)
  End If

  ' Update time variables
  Hnum = Hour(Now)
  Mnum = Minute(Now)
  Snum = Second(Now)
  ' Update digital display, whether visible or not
  Digital.CurrentX = 0
  Digital.CurrentY = 0
  TTime$ = Time$
  Digital.Cls
  Digital.Print TTime$
End Sub

Sub Timer3_Timer ()
  Mainform.Comm2.InputLen = 0
  NL$ = Chr$(13) + Chr$(10)
  'Mainform.Comm2.Output = "@" + "&" + "$" + NL$
  If Mainform.Comm2.InBufferCount Then
    NowTime$ = Mainform.Comm2.Input
    If Len(NowTime$) < 21 Then
      Digital.Cls
    Else
      Digital.Cls
      NNowTime$ = Mid$(NowTime$, 7, 2) + ":"
      NNow1Time$ = NNowTime$ + Mid$(NowTime$, 5, 2) + ":"
      NNow2Time$ = NNow1Time$ + Mid$(NowTime$, 3, 2)
      Digital.Print NNow2Time$
    End If
```

```
    End If
End Sub
```

Messageform.frm

```
Sub Command1_Click ()
    menSend_Click
End Sub

Sub Command2_Click ()
     Unload Messagefrm
End Sub

Sub menExit_Click ()
    Unload Messagefrm
End Sub

Sub menSend_Click ()
  NL$ = Chr$(13) + Chr$(10)
   Mainform.Label2.Caption = Text1.Text
   Words$ = Text1.Text
   Mainform.Comm2.Output = "*" & Words$ + "$" + NL$
      Hide
End Sub
```

Openfileform.frm

```
  ' Declarations for GETFILE.FRM
  Const TEXTFLAG = 0
  Const FILEFLAG = 1
```

```
Const DIRFLAG = 2
Dim SelectFlag As Integer

Sub ExitForm ()
    ' User might want different patterns next time
    FileTypes.Clear
    ' Don't unload, simply hide
    OpenFile.Hide
End Sub

Sub FillLabel1 ()
    ' Display directory part of path
    Label1.Caption = Dir1.Path

    ' If directory string is too long, squish it down
    If Label1.Width > 2200 Then
        ' Extract drive part
        a$ = Left$(Dir1.Path, 3)
        b$ = Mid$(Dir1.Path, 4)
        ' Extract last subdirectory part
        Do While InStr(b$, "\")
            b$ = Mid$(b$, InStr(b$, "\") + 1)
        Loop
        ' Squish out middle part
        Label1.Caption = a$ + "...\" + b$
    End If
End Sub

Function GetFileType$ ()
    ' Get pattern description from combo box
```

```
    Tmp$ = FileTypes.Text
    ' Find position of parentheses
    p1 = InStr(Tmp$, "(") + 1
    p2 = InStr(Tmp$, ")")
    ' Return part between parentheses
    If p1 > 0 And p2 > p1 Then
        GetFileType$ = LCase$(Mid$(Tmp$, p1, p2 - p1))
    Else
        GetFileType$ = "*.*"
    End If
End Function

Sub Command1_Click ()
    ' OK button; some errors can happen
    On Error GoTo ErrorTrap
    ' Was the last change to the filename in Text1?
    If SelectFlag = TEXTFLAG Then
        File1.FileName = Text1.Text
        ' We're done if FullPath was set
        If FullPath <> "" Then
            On Error GoTo 0
            ExitForm
        End If
    ' Update directory list
    Dir1.Path = File1.Path
    ' Was user only selecting a new directory?
    ElseIf SelectFlag = DIRFLAG Then
        Dir1.Path = Dir1.List(Dir1.ListIndex)
        Dir1_Change
    ' Set FullPath to selected file
```

```
    Else
        If Right$(Dir1.Path, 1) = "\" Then
            FullPath.Text = Dir1.Path + Text1.Text
        Else
            FullPath.Text = Dir1.Path + "\" + Text1.Text
        End If
        ' All done
        ExitForm
    End If
    DataView.FullPath.Text = OpenFile.FullPath.Text
    DataView.Show MODALESS
    Exit Sub

ErrorTrap:
    Beep
    Resume Next
End Sub

Sub Command2_Click ()
    ' Cancel button; indicate by erasing FullPath
    FullPath = ""
      'All done
    Unload OpenFile
End Sub

Sub Dir1_Change ()
    ' User selected new subdirectory
    FillLabel1
    ' Update filename
    File1.FileName = Dir1.Path + "\" + File1.Pattern
```

```
    File1.Pattern = GetFileType$()
    ' Update drive list
    Drive1.Drive = Dir1.Path
    ' Update name of file
    Text1.Text = File1.Pattern
    ' Set last change to directory
    SelectFlag = DIRFLAG
End Sub

Sub Drive1_Change ()
    ' User changed drive; update directory
    Dir1.Path = Drive1.Drive

    ' Display current pattern
    Text1.Text = File1.Pattern
    ' Set last change to directory
    SelectFlag = DIRFLAG
End Sub

Sub File1_Click ()
    ' User clicked on new filename
    Text1.Text = File1.FileName
    ' Set last change to filename
    SelectFlag = FILEFLAG
End Sub

Sub FileTypes_Click ()
    ' User selected new pattern from combo box
    File1.Pattern = GetFileType$()
    ' Display pattern until a file is selected
```

```
    Text1.Text = File1.Pattern
End Sub

Sub Form_Load ()
    ' Center form on screen
    OpenFile.Left = (Screen.Width - OpenFile.Width) / 2
    OpenFile.Top = (Screen.Height - OpenFile.Height) / 2
End Sub

Sub Form_KeyUp (KeyCode As Integer, Shift As Integer)
    ' Watch only for Alt plus N, D, T or V key
    If Shift = 4 Then
        Select Case KeyCode

        ' Alt+N
        Case 78
            Text1.SetFocus
        ' Alt+D
        Case 68
            Dir1.SetFocus
        ' Alt+T
        Case 84
            FileTypes.SetFocus
        ' Alt+V
        Case 86
            Drive1.SetFocus
        End Select
    End If
End Sub
```

```
Sub Text1_Change ()
    ' Set last change to File Name field
    SelectFlag = TEXTFLAG
End Sub
```

Ssavefileform.frm

```
' Declarations for SAVEFILE.FRM
Const TEXTFLAG = 0
Const FILEFLAG = 1
Const DIRFLAG = 2
Dim SelectFlag As Integer

Sub ExitForm ()
   ' User might want different patterns next time
    FileTypes.Clear
   ' Don't unload, simply hide
    SSaveFile.Hide
End Sub

Sub FillLabel1 ()
    ' Display directory part of path
     Label1.Caption = Dir1.Path
    ' If directory string is too long, squish it down
     If Label1.Width > 2200 Then
       ' Extract drive part
        a$ = Left$(Dir1.Path, 3)
        b$ = Mid$(Dir1.Path, 4)
        ' Extract last subdirectory part
        Do While InStr(b$, "\")
```

```
            b$ = Mid$(b$, InStr(b$, "\") + 1)
        Loop
        ' Squish out middle part
        Label1.Caption = a$ + "...\" + b$
    End If
End Sub

Function SaveFileType$ ()
    ' Get pattern description from combo list
    Tmp$ = FileTypes.Text
    ' Find position of parentheses
    p1 = InStr(Tmp$, "(") + 1
    p2 = InStr(Tmp$, ")")
    ' Return part between parentheses
    If p1 > 0 And p2 > p1 Then
        SaveFileType$ = LCase$(Mid$(Tmp$, p1, p2 - p1))
    Else
        SaveFileType$ = "*.*"
    End If
End Function

Sub Command1_Click ()
    ' OK button; some errors can happen
    On Local Error GoTo ErrorTrap
    ' Was user only selecting a new directory?
    If SelectFlag = DIRFLAG Then
        Dir1.Path = Dir1.List(Dir1.ListIndex)
        Dir1_Change
        SelectFlag = TEXTFLAG
    ' Try to return indicated filename
```

```
ElseIf InStr(Text1.Text, "\") Then
    Tmp$ = Text1.Text
  ' Trim back to last \
    Do Until Right$(Tmp$, 1) = "\"
       Tmp$ = Left$(Tmp$, Len(Tmp$) - 1)
    Loop
  ' Trim off \ if not root directory
    If Len(Tmp$) > 3 Then
       Tmp$ = Left$(Tmp$, Len(Tmp$) - 1)
    End If
  ' Set indicated directory
    Dir1.Path = Tmp$
  ' Trim path off left of filename
    Do
       p1 = InStr(Text1.Text, "\")
       Text1.Text = Mid$(Text1.Text, p1)
    Loop While InStr(Text1.Text, "\")
' Filename shown with no user-entered path
Else
    ' Get working copy of filename
    Tmp$ = Trim$(Text1.Text)
    ' First character must be alphabetic
    If Not Left$(Tmp$, 1) Like "[a-zA-Z]" Then
       Tmp$ = ""
    End If
    ' Check for position of period
    p1 = InStr(Tmp$, ".")
    If p1 > 0 Then
       If Len(Tmp$) - p1 > 3 Then Tmp$ = ""
       If p1 > 9 Then Tmp$ = ""
```

```
        Else
            If Len(Tmp$) > 9 Then Tmp$ = ""
        End If
        ' Proceed only if filename seems OK
        If Tmp$ <> "" Then
            ' Build full path to file
            If Right$(Dir1.Path, 1) = "\" Then
                FullPath = Dir1.Path + Tmp$
            Else
                FullPath = Dir1.Path + "\" + Tmp$
            End If
                'Return full path
            ExitForm
        ' Not a good filename
        Else
            Beep
            Text1.SetFocus
        End If
        Open FullPath.Text For Output As #1
         Print #1, DataView.Text1.Text
        Close #1
    End If
     Exit Sub
ErrorTrap:
    Beep
    Resume Next
End Sub

Sub Command2_Click ()
    ' Cancel button; indicate by erasing FullPath
```

```
    FullPath = " "
    ' All done
    ExitForm
End Sub

Sub Dir1_Change ()
    ' User selected new subdirectory
    FillLabel1
    ' Update filename
    File1.FileName = Dir1.Path + "\" + File1.Pattern
    ' Update drive list
    Drive1.Drive = Dir1.Path
    ' Update name of file
    Text1.Text = File1.Pattern
    ' Set last change to directory
    SelectFlag = DIRFLAG
End Sub

Sub Drive1_Change ()
    ' User changed drive; update directory
    Dir1.Path = Drive1.Drive
    ' Display current pattern
    Text1.Text = File1.Pattern
    ' Set last change to directory
    SelectFlag = DIRFLAG
End Sub

Sub File1_Click ()
    ' User clicked on new filename
    If File1.ListIndex <> -1 Then
```

```
        Tmp$ = File1.FileName
        File1.ListIndex = -1
        Text1.Text = Tmp$
    End If
    ' Set last change to filename
    SelectFlag = FILEFLAG
End Sub

Sub FileTypes_Click ()
    ' User selected new pattern from combo box
    File1.Pattern = SaveFileType$()
    ' Display pattern until a file is selected
    Text1.Text = File1.Pattern
End Sub

Sub Form_Load ()
    ' Center form on screen
    SSaveFile.Left = (Screen.Width - SSaveFile.Width) / 2
    SSaveFile.Top = (Screen.Height - SSaveFile.Height) / 2
End Sub

Sub Form_KeyUp (KeyCode As Integer, Shift As Integer)
    ' Watch only for Alt plus N, D, T or V key
    If Shift = 4 Then
        Select Case KeyCode
        ' Alt+N
        Case 78
            Text1.SetFocus
        ' Alt+D
        Case 68
```

```
            Dir1.SetFocus
        ' Alt+T
        Case 84
            FileTypes.SetFocus
            ' Alt+V
        Case 86
            Drive1.SetFocus
        End Select
    End If
End Sub

Sub Text1_Change ()
    ' Set last change to File Name field
    SelectFlag = TEXTFLAG
End Sub
```

Tempform.frm

```
Sub Command1_Click ()
    Unload Tempfrm
End Sub

Sub Form_Load ()
    If Mainform.Comm2.PortOpen = False Then
    Mainform.Comm2.PortOpen = True
    End If
        Mainform.Comm2.Output = "!" + "$" + NL$
    If Mainform.Comm2.InBufferCount Then
    ' Read the "OK" response data in the serial port.
      InString$ = Mainform.Comm2.Input
```

```
        TempValue$ = Mid$((Right$(InString$, 5)), 1, 2)
        TempValue$ = Mainform.Comm2.Input
        Gauge2.Value = Val(TempValue$)
    End If
        Label1.Caption = "Now temperature is " & Gauge2.Value & " 'C"
    Form_Resize
End Sub

Sub Form_Resize ()
    Frame3D1.Move 0, 0, ScaleWidth, ScaleHeight
End Sub

Sub menAboutTemp_Click ()
    Aboutdem.Label1.Caption = " TEMPERATURE "
    ' show about form
    Aboutdem.Show MODAL
End Sub

Sub menExit_Click ()
    Unload Tempfrm
End Sub
```

ภาคผนวก

วิธีการใช้งาน TIME&CALENDAR REMOTE MONITOR

1. การ UP-LOAD &DOWN-LOAD

- ต่อสาย RS-232 จาก MASTER CONTROLเข้ากับ COMPUTOR ทางSERIALPORT

ซึ่งจะเป็น COM1 (DB9) หรือ COM2 (DB25)

-SET DIPSWITCH 

RS - 232

RS - 422

- ON POWER ของ MASTER CONTROL

-UPDATE DATA ของTIME (HH-MM-SS) และCALENDA(DD-MM-YY) ,(DAY)

-การ UPLOAD-DOWNLOAD จะกระทำผ่านโปรมแกรม VISTUAL BASIC

2.การตั้งเวลาที่ MASTER CONTROL

2.1 การตั้งเวลา

-กด ▽ หรือ △ ให้แสดงผลที่ TIMEDISPLAY 

-ถ้าต้องการ ตั้งวินาที (SECOND)

-กด ◁ หรือ ▷ ให้แสดงผลที่ SECONDS DISPLAY 

-กด ▽ หรือ △ ให้แสดงผลที่ DATA DISPLAYที่ต้องการ 

-ถ้าไม่ต้องการ SET ต่อไป ก็รอประมาณ 5 SEC จะDISPLAY การ SETก็สำเร็จ 

-ถ้าต้องการตั้งนาที (MINUTE)

-กด ◁ หรือ ▷ ให้แสดงผลที่ MINUTES DISPLAY 

-กด ▽ หรือ △ ให้แสดงผลที่ DATA DISPLAYที่ต้องการ 

-ถ้าไม่ต้องการ SET ต่อไป ก็รอประมาณ 5 SEC จะDISPLAY การ SETก็สำเร็จ 

-ถ้าต้องการชั่วโมง (HOUR)

-กด ◁ หรือ ▷ ให้แสดงผลที่ HOUR DISPLAY

-กด ▽ หรือ △ ให้แสดงผลที่ DATA DISPLAYที่ต้องการ

-ถ้าไม่ต้องการ SET ต่อไป ก็รอประมาณ 5 SEC จะDISPLAY การ SETก็สำเร็จ

3.การตั้งปฏิทิน(CALENDAR)

-ถ้าต้องการ ตั้งปี (YEAR)

-กด ◁ หรือ ▷ ให้แสดงผลที่ YEAR DISPLAY

-กด ▽ หรือ △ ให้แสดงผลที่ DATA DISPLAYที่ต้องการ

-ถ้าไม่ต้องการ SET ต่อไป ก็รอประมาณ 5 SEC จะDISPLAY การ SETก็สำเร็จ

-ถ้าต้องการตั้งเดือน (MONTH)

-กด ◁ หรือ ▷ ให้แสดงผลที่ MONTH DISPLAY

-กด ▽ หรือ △ ให้แสดงผลที่ DATA DISPLAYที่ต้องการ

-ถ้าไม่ต้องการ SET ต่อไป ก็รอประมาณ 5 SEC จะDISPLAY การ SETก็สำเร็จ

-ถ้าต้องการตั้งวันที่ (DATE)

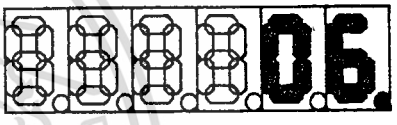
-กด ◁ หรือ ▷ ให้แสดงผลที่ DATE DISPLAY

-กด ▽ หรือ △ ให้แสดงผลที่ DATA DISPLAYที่ต้องการ

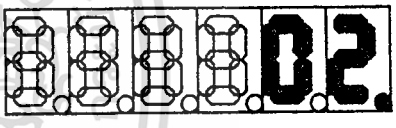
-ถ้าไม่ต้องการ SET ต่อไป ก็รอประมาณ 5 SEC จะDISPLAY
การ SETก็สำเร็จ

-ถ้าต้องการตั้งวัน (DAY)

-กด ◁ หรือ ▷ ให้แสดงผลที่ DAY DISPLAY

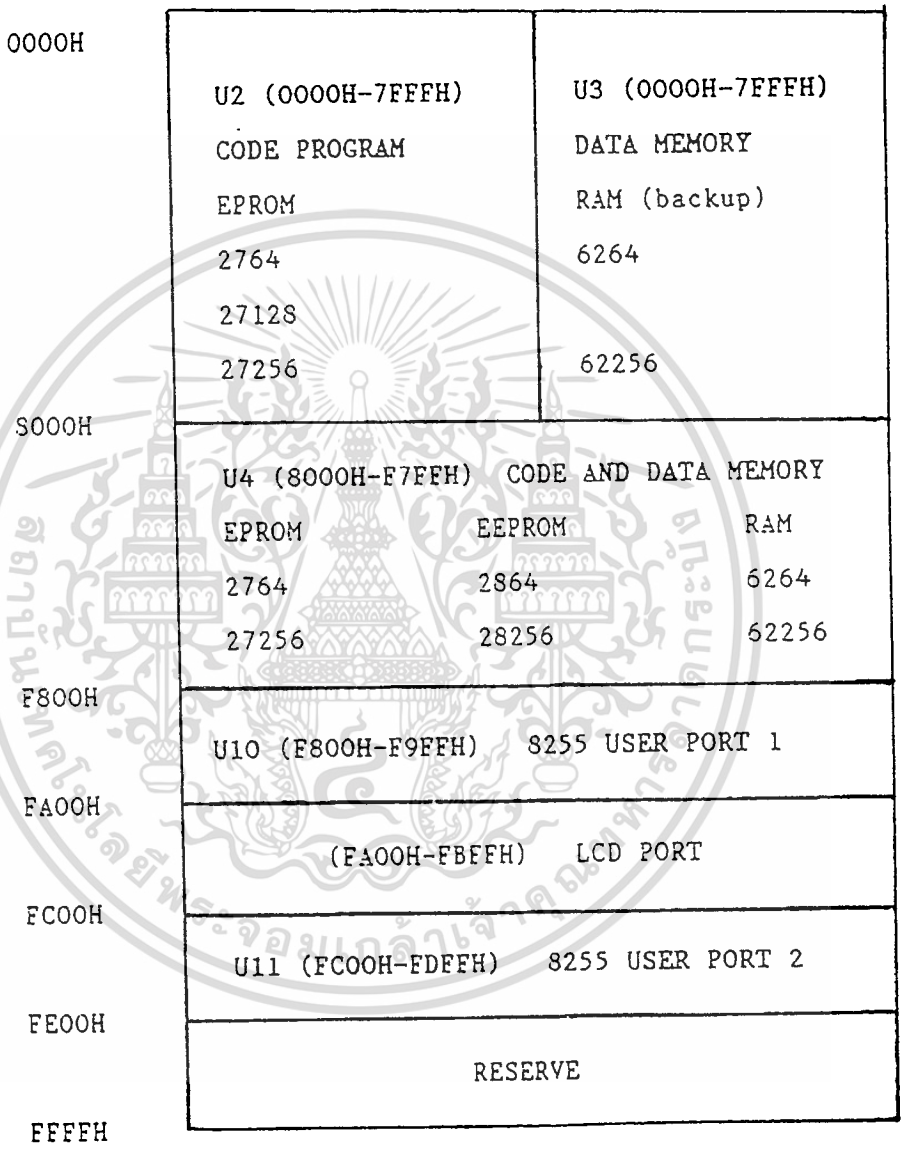


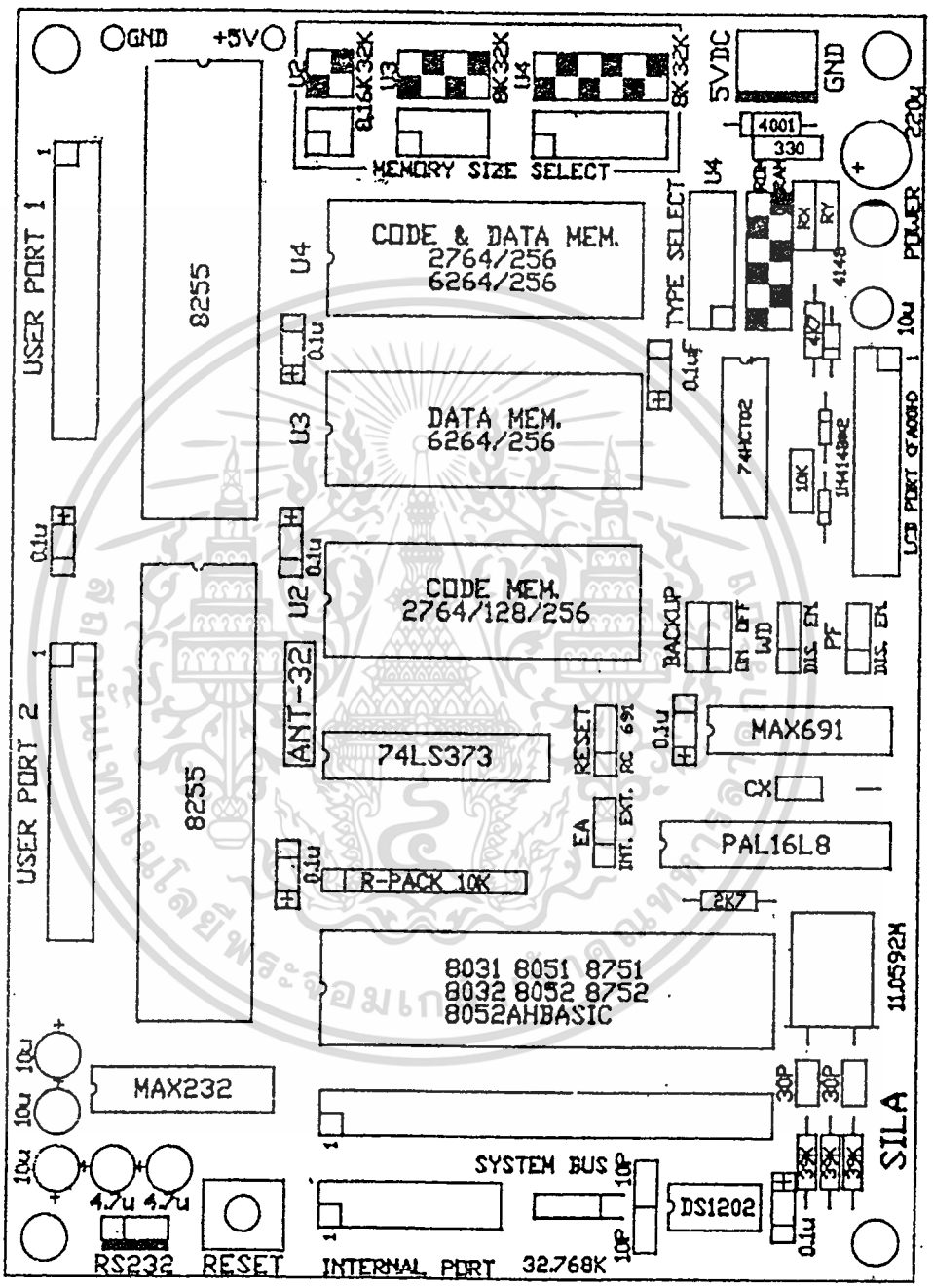
-กด ▽ หรือ △ ให้แสดงผลที่ DATA DISPLAYที่ต้องการ

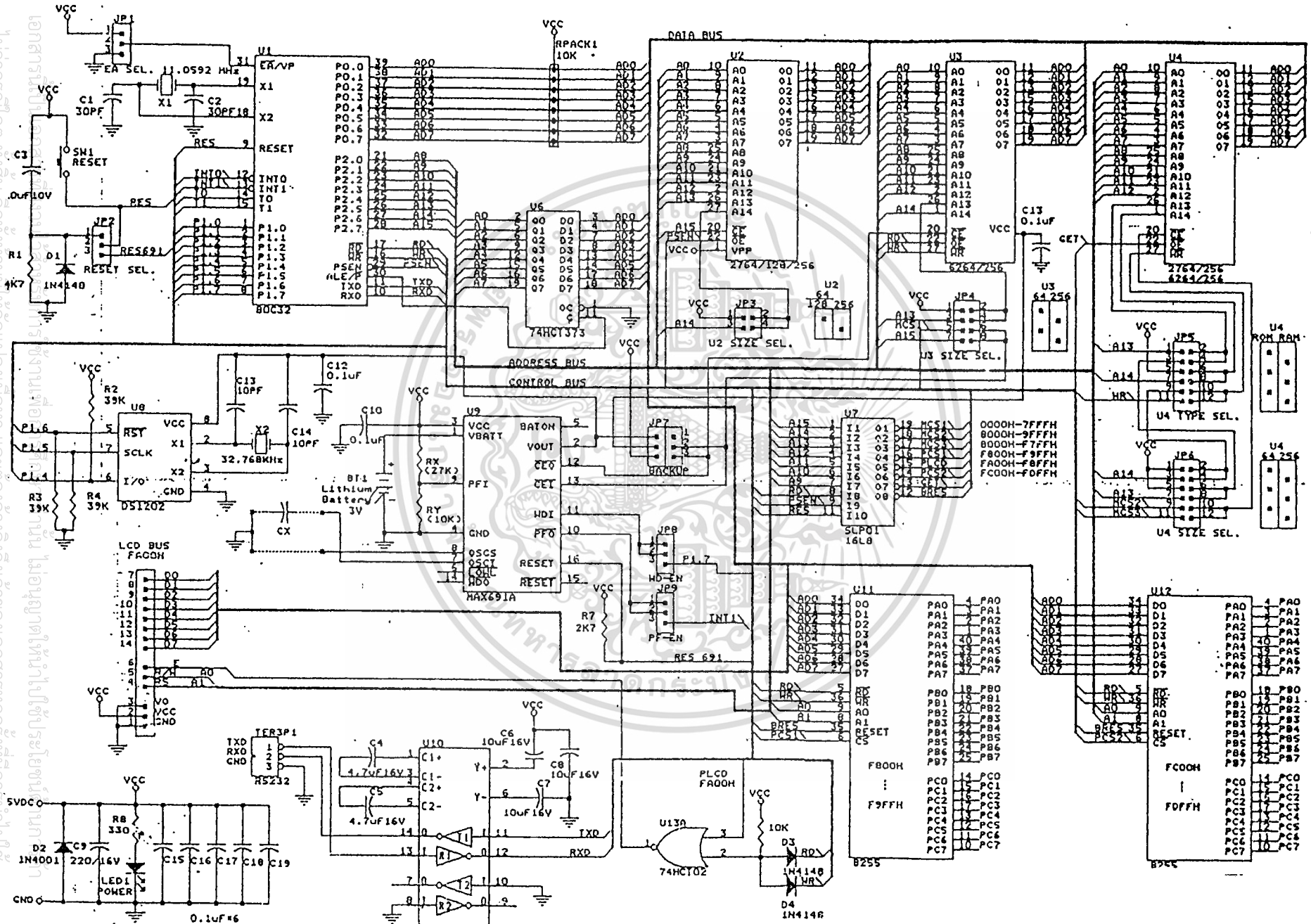


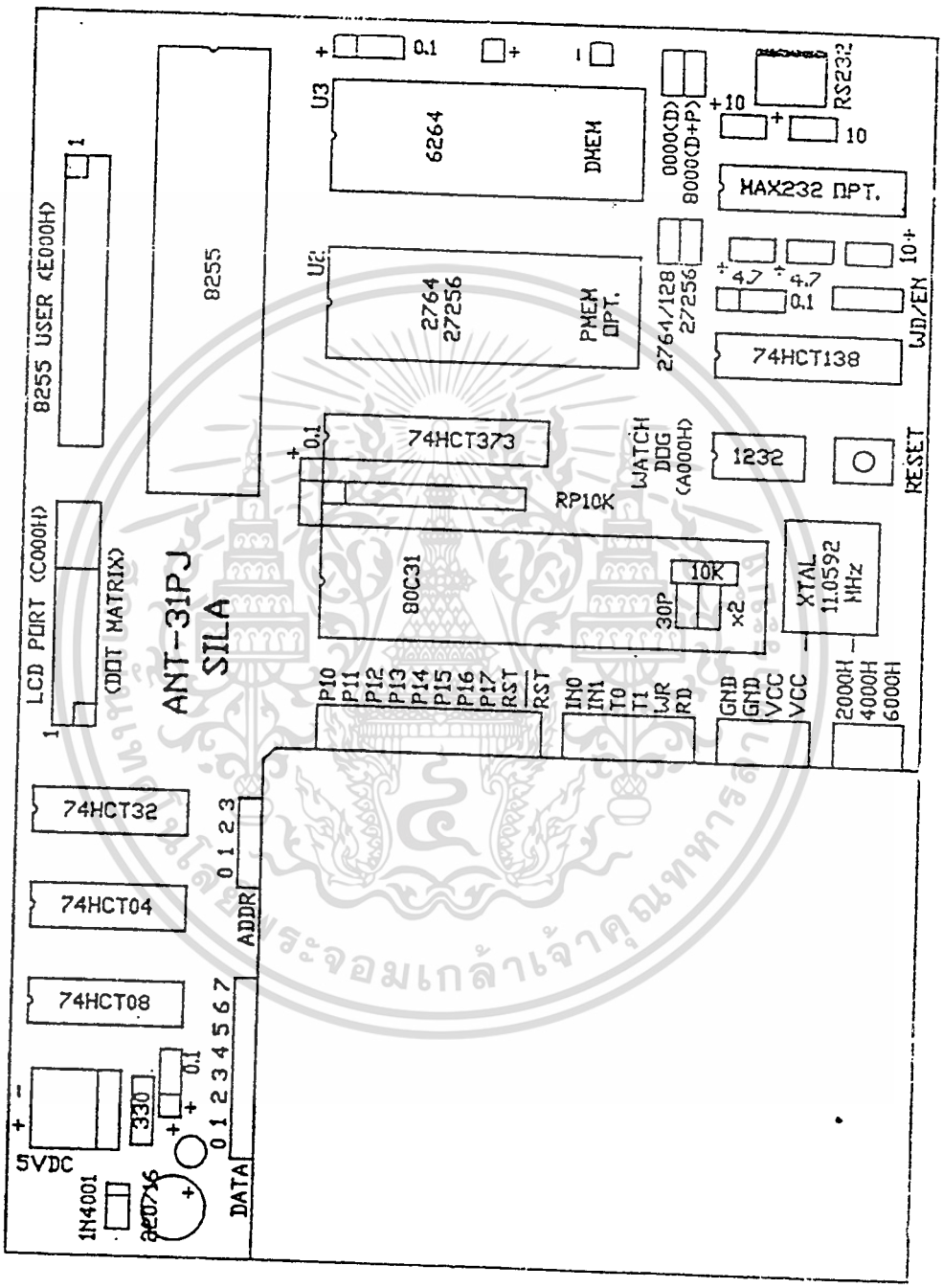
-ถ้าไม่ต้องการ SET ต่อไป ก็รอประมาณ 5 SEC จะDISPLAY
การ SETก็สำเร็จ

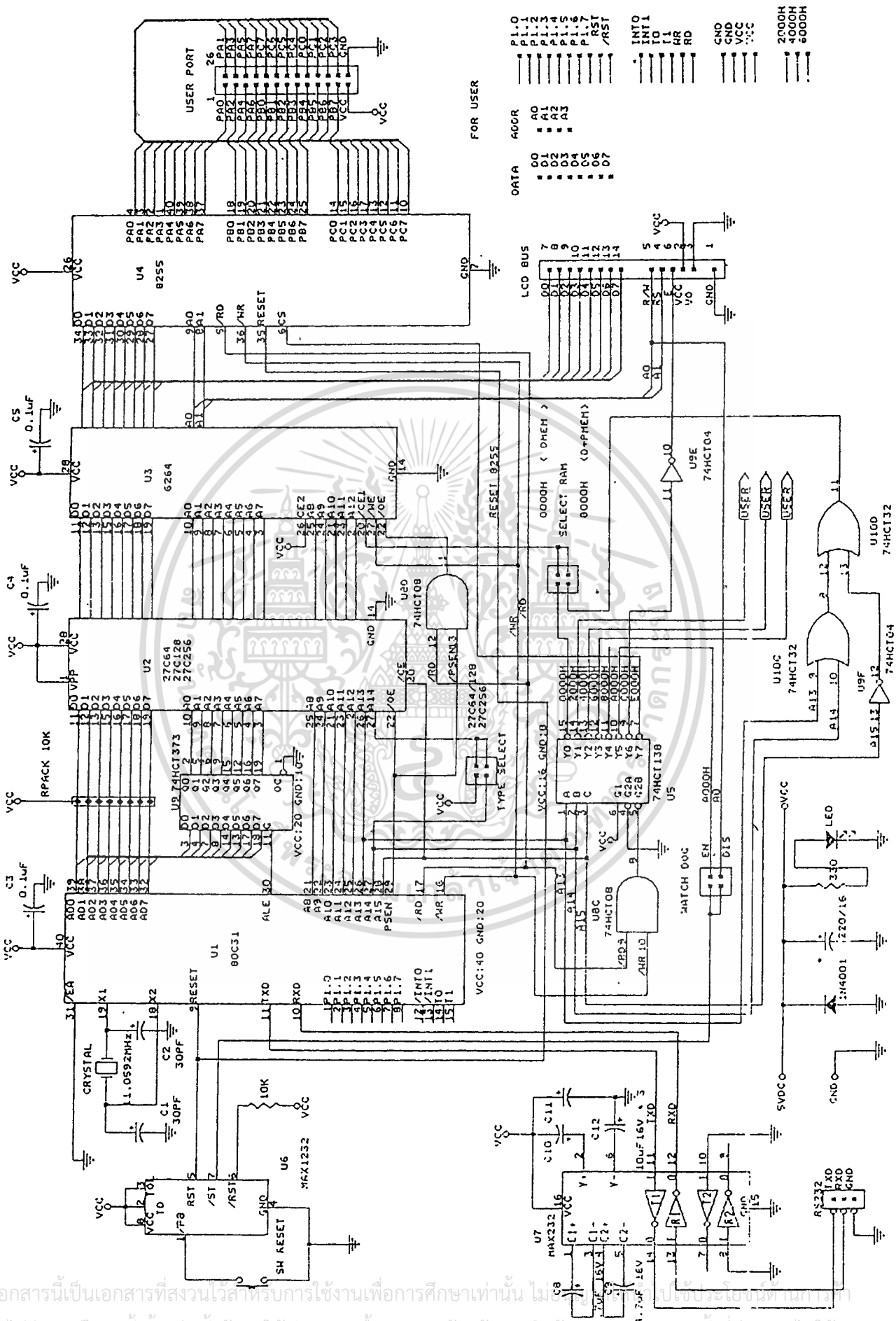
ANT-32 MEMORY MAP

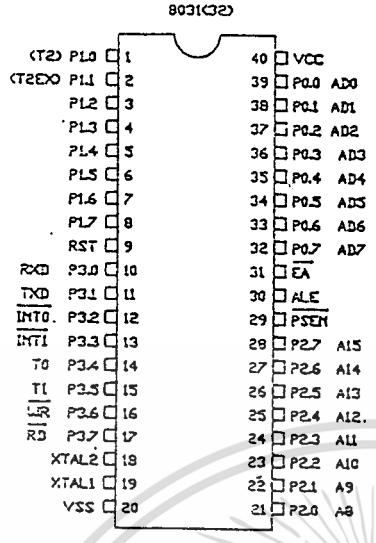


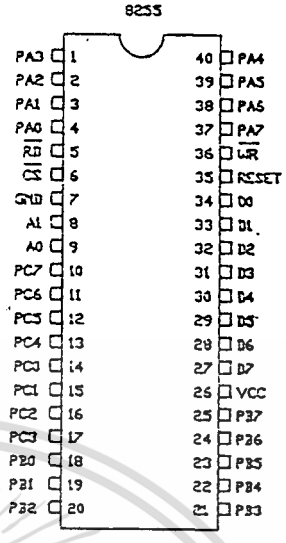


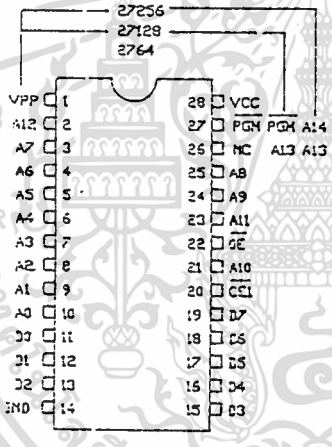

DATA BUS
ADDRESS BUS
CONTROL BUS
LCD BUS
FAOOH
PLCO
FAOOH
U1
80C32
U2
2764/128/256
U3
6264/256
U4
2764/256
6264/256
U5
74HCT373
U6
DS1202
U7
SLP01
16L8
U8
DS1202
U9
MAX691A
U10
MAX232
U11
8255
U12
8255
U13A
74HCT02
RPACK1
10K
EA SEL.
11.0592 MHz
SW1
RESET
RESET SEL.
RES691
U2 SIZE SEL.
U3 SIZE SEL.
U4 TYPE SEL.
U4 SIZE SEL.
BACKUP
Lithium
Battery
3V
32.768KHz
RS232
LED1
POWER
0000H-7FFFH
8000H-9FFFH
8000H-F7FFH
F800H-F9FFH
FA00H-FBFFH
FC00H-FDFFH
FBOOH
F9FFH
FCOOH
FDFFH
INT1
RES 691
HD_EN
PF_EN
BATON
VOUT
CEO
CET
HDI
PFO
RESET
OSCS
OSCI
CONC
HDO
TXD
RXD
RD
WR
ALE/P
PSEN
5VDC
GND
VCC

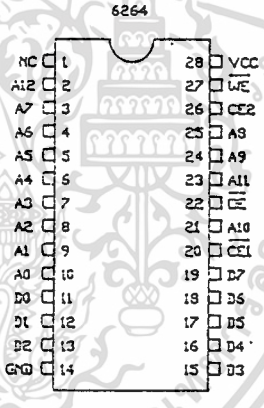


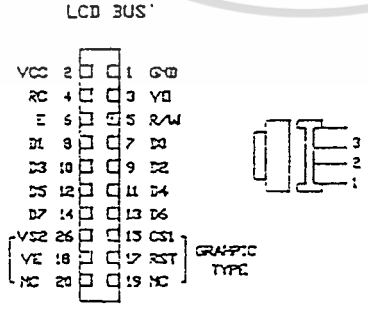


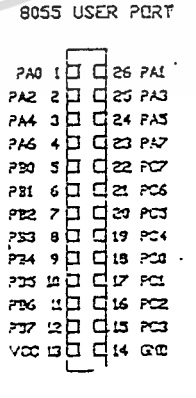




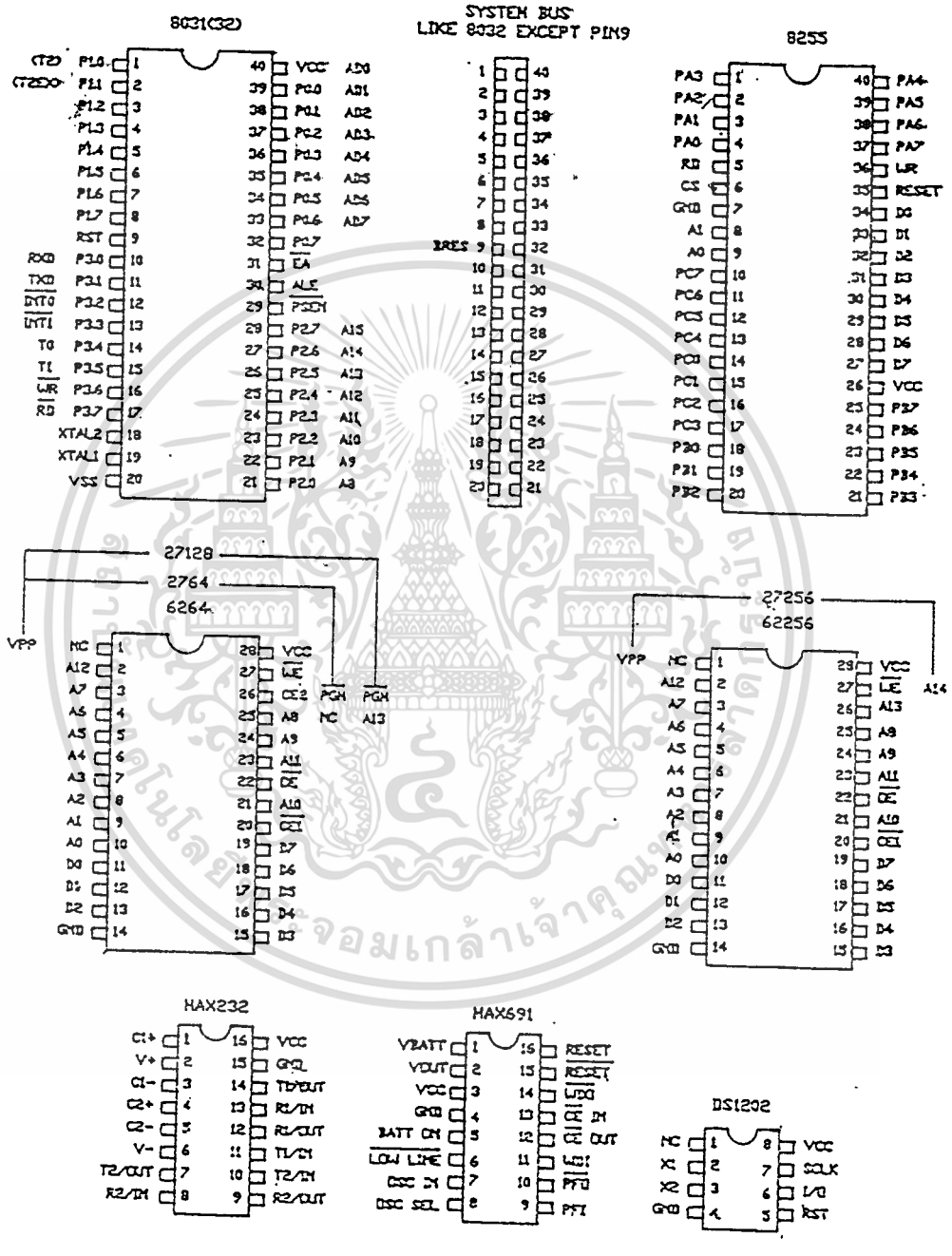








ภาพแสดงรายละเอียดของ CONNECTOR และ CHIP (ต่อ)



1 Kenneth J. Ajaya , " The Microcontrol Controller Architecture Programming , and Application " , West Publishing Company , 1991

2 Dallas Semiconduction Corporation , " DS 1202 Data sheet" , Dallas Semiconduction Corporation , 1993

3 Ross Neson , " Running Visual Basic For windows ", Microsoft Corporation, 1994

4 Peter W. Gofton , " Mastering Serial Communication " , SYBEX INC ,1994

5 สุนทร วิทูสุรพจน์ , " การโปรแกรมภาษาแอสเซมบลี ของ ไมโครคอนโทรลเลอร์ตระกูล-8051" , บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2537

6 ราบินเดอร์ ศรีกิจจาภรณ์ , " คู่มือการใช้งาน Visual Basic สำหรับ windows " , บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) , 2538

7 จิรศักดิ์ เหลืองอุไร , " คัมภีร์การใช้งานการสื่อสารอนุกรมบน PC " , บริษัทซีเอ็ด ยูเคชั่น จำกัด (มหาชน) , 2538