การบันทึกสภาพอากาศ
WEATHER RECORDER

โดย

นาย อดิศร เข็มวิชัย

นาย อมรกุล อัมรามร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

หัวข้อปริญญานิพนธ์	การบันทึกสภาพอากาศ
ชื่อนักศึกษา	นาย อดิศร เข็มวิชัย
	นาย อมรกุล อัมรามร
ภาควิชา	เทคนิคอุตสาหกรรม
อาจารย์ที่ปรึกษา	ผศ. ไพศาล สิทธิโยภาสกุล

คณะวิศวกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้ปริญญานิพนธ์ ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

..... ประธานกรรมการ
( )

..... กรรมการ
( )

..... กรรมการ
( )

..... กรรมการ
( )

..... กรรมการ
( )

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

PROJECT REPORT	WEATHER RECORDER
BY	MR. ADISORN KEMWICHAI
	MR. AMONKUL UMMARAMON
DEPARTMENT OF	INDUSTRIAL TECHNOLOGY
ADVISER	ASST PROF. PAISARN SITIYOPASAKUL

ACCEPTED BY THE FACULTY OF ENGINEERING, KING MONGKUT'S INSTITUTE OF TECNOLOGY LADKRABANG IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE.

PROJECT REPORT COMMITTEE

..... Chairman

(.....)

..... Committee

(.....)

..... Committee

(.....)

..... Committee

(.....)

..... Committee

(.....)

หัวข้อปริญญานิพนธ์	การบันทึกสภาพอากาศ
ชื่อนักศึกษา	นาย อดิศร เข็มวิชัย
	นาย อมรกุล อัมรามร
ภาควิชา	เทคนิคอุตสาหกรรม
อาจารย์ที่ปรึกษา	ผศ. ไพศาล สิทธิโยภาสกุล

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้ มีเนื้อหาเกี่ยวกับ การเก็บบันทึกข้อมูลสภาพอากาศ ประกอบด้วย อุณหภูมิ ความเข้มแสง โดยอ้างอิงเวลากับไอซีสร้างฐานเวลาจริง ซึ่งข้อมูลที่เก็บทั้งหมดจะเป็นข้อมูลดิจิตอล จัดเก็บไว้ในส่วนหน่วยความจำของฮาร์ดแวร์ และสามารถส่งข้อมูลที่จัดเก็บไว้ในส่วนของฮาร์ดแวร์ผ่านพอร์ตสื่อสารอนุกรม ไปเก็บยังฮาร์ดดิสก์ของคอมพิวเตอร์ เพื่อนำข้อมูลนั้นมาแสดงผลผ่านส่วนของซอฟต์แวร์โปรแกรมที่อยู่ในคอมพิวเตอร์ โดยแสดงข้อมูลในรูปแบบกราฟเชิงเส้น เปลี่ยนแปลงตามเวลา ในช่วงเวลาหนึ่งวัน ซึ่งจะทำให้เราสามารถสังเกตถึงความเปลี่ยนแปลงของสภาพอากาศ ได้โดยทำการเปรียบเทียบข้อมูลที่เก็บไว้ และสามารถใช้ข้อมูลเหล่านี้ หาความน่าจะเป็นของสภาพอากาศล่วงหน้าได้ โดยปริญญานิพนธ์นี้ประกอบด้วยส่วนของฮาร์ดแวร์ซึ่งเป็นบอร์ดคอนโทรล ที่ออกแบบขึ้น และใช้โปรแกรมภาษาแอสเซมบลีในการเขียนโปรแกรมควบคุม ส่วนซอฟต์แวร์โปรแกรมของคอมพิวเตอร์ใช้โปรแกรม Visual Basic 6 ในการเขียนโปรแกรมจัดการข้อมูล

PROJECT REPORT WEATHER RECORDER

BY MR. ADISORN KEMWICHAI

MR. AMONKUL UMMARAMON

DEPARTMENT OF INDUSTRIAL TECHNOLOGY

ADVISER ASST PROF. PAISARN SITIYOPASAKUL

ABSTRACT

This project have detail about weather data record. It consist of Temperature data, and Light data. This data with refer by Real time clock IC. All data is digital and record in the memory of hardware. The hardware can send data in memory to computer disk by serial port. That data can display in software Program on computer in linear graph form, on period day. Form data we can see the difference and calculate probability of the weather. This project consist of hardware, is control Board and software program on computer using Visual Basic 6 Program.

กิตติกรรมประกาศ

โครงงานนี้สามารถสำเร็จได้ด้วยดี เพราะได้รับคำแนะนำจาก อ.ไพศาล สิทธิโยภาสกุล ทั้งในด้าน ฮาร์ดแวร์ และ ซอฟแวร์ รวมทั้งเอกสารอ้างอิง ขอขอบคุณ คุณ วุฒิชัย ชื่นดี ที่ช่วยในเรื่องสถานที่ และ อุปกรณ์ในการทำโครงงานนี้ ขอขอบคุณ คุณ สุทธิศักดิ์ พงศ์ธนาพานิช ที่ให้คำแนะนำในด้านโปรแกรม Visual Basic ขอขอบคุณเพื่อนๆทุกคนที่ได้ให้ความช่วยเหลือในโครงงานนี้ และของคุณตัวเองที่พยายามมาจนมีวันนี้ ขอขอบคุณพระคุณ บิดา มารดา ซึ่งเป็นผู้ให้กำลังใจตลอดมา

คณะผู้จัดทำ

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	
บทคัดย่อภาษาอังกฤษ	
กิตติกรรมประกาศ	
บทที่ 1 บทนำ	**1**
วัตถุประสงค์	1
ขอบเขต	1
บทที่ 2 ทฤษฎีที่ใช้ในโครงงาน	**4**
ทฤษฎีที่สำคัญเกี่ยวกับ AT89S8252	4
SPI (Serial Peripheral Interface)	6
ทฤษฎีเบื้องต้นเกี่ยวกับระบบบัส I^2C	13
ทฤษฎีระบบการสื่อสารอนุกรมแบบหนึ่งสาย (1 – Wire^{TM} Serial Bus)	23
บทที่ 3 การออกแบบ	**30**
การออกแบบฮาร์ดแวร์	30
ภาควัดอุณหภูมิ	31
ภาควัดความเข้มแสง	35
ภาค ADC	37
ภาคฐานเวลาเรียลไทม์	39
ภาคหน่วยความจำ	45
ภาคคอนโทรลเลอร์	46
ภาคการเชื่อมต่อกับคอมพิวเตอร์	48
การเขียนโปรแกรมควบคุม	49
การออกแบบซอฟแวร์โปรแกรม	50
หลักการของซอฟแวร์	50
การแสดงผลออกทางจอภาพ	51
การตั้งเวลาให้ฮาร์ดแวร์	51

การเก็บข้อมูล 52
บทที่ 4 สรุปและวิจารณ์ **53**
เอกสารอ้างอิง
ภาคผนวก

สารบัญรูป

รูป	หน้า
1.1 แสดงบล็อกไดอะแกรมส่วนประกอบทางฮาร์ดแวร์	2
1.2 แสดงลักษณะของกราฟที่ใช้แสดงผล	3
2.1 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MSC-51 แบบแฟลช AT89S8252	5
2.2 รูปแบบของการเชื่อมต่อของ SPI อย่างง่าย	6
2.3 ไดอะแกรมของการทำงานของส่วน SPI ภายในไมโครคอนโทรลเลอร์ AT89S8252	8
2.4 การเชื่อมต่อสัญญาณในระบบ SPI ระหว่างอุปกรณ์มาสเตอร์และสเลฟ	9
2.5 ไดอะแกรมเวลาของการถ่ายทอดข้อมูล SPI เมื่อบิต CPHA เป็น “0”	10
2.6 ไดอะแกรมเวลาของการถ่ายทอดข้อมูล SPI เมื่อบิต CPHA เป็น “1”	10
2.7 ผังแสดงการเชื่อมต่อของอุปกรณ์ต่างๆ บนระบบบัส I^2C	14
2.8 แสดงวงจรเอาต์พุตของอุปกรณ์ในระบบบัส I^2C	15
2.9 การต่อตัวต้านทานพูลอัปบนสายสัญญาณในระบบบัส I^2C	16
2.10 การต่อตัวต้านทาน Rs เพื่อลดสัญญาณรบกวนขนาดใหญ่ที่อาจเข้ามาในระบบบัส I^2C	16
2.11 ไดอะแกรมเวลาสถานะต่างๆในบัส I^2C	18
2.12 รูปแบบของข้อมูลที่กำหนดแอดเดรสที่ใช้ในการอ้างอิง 7 บิต	19
2.13 รูปแบบของข้อมูลอนุกรมที่ใช้ในการติดต่ออุปกรณ์บนบัส I^2C เมื่อใช้ในการอ้างอิงถึง 7 บิต	20
2.14 รูปแบบของข้อมูลอนุกรมที่ใช้ในการติดต่ออุปกรณ์บนบัส I^2C เมื่อใช้ในการอ้างอิงถึง 10 บิต	20
2.15 วงจรตัวอย่างการต่อไมโครคอนโทรลเลอร์ MCS-51 กับอุปกรณ์ระบบบัส I^2C	21
2.16 การเชื่อมต่อบนระบบบัสหนึ่งสาย	24
2.17 ไทม์สล็อตการรีเซตและการตอบรับของอุปกรณ์บนระบบบัสหนึ่งสาย	26
2.18 ไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์ ซึ่งตรงกับไทม์สล็อตการเขียนข้อมูลของอุปกรณ์สเลฟ	27
2.19 ไทม์สล็อตการเขียนข้อมูล “1” ของอุปกรณ์มาสเตอร์ ซึ่งตรงกับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์สเลฟ	28
2.20 ไทม์สล็อตการเขียนข้อมูล “0” ของอุปกรณ์มาสเตอร์	28

รูป หน้า

3.1 แสดงบล็อกไดอะแกรม ส่วนประกอบทางฮาร์ดแวร์ 30

3.2 การจัดขาของ DS1820 31

3.3 โครงสร้างการทำงานภายในของไอซีตรวจจับอุณหภูมิ 31

3.4 การจัดสรรพื้นที่ สแครตช์แพด ใน DS1820 32

3.5 การเชื่อมต่อ DS1820 กับไมโครคอนโทรลเลอร์ MCS-51 34

3.6 แสดงลักษณะข้อมูลของ DS1820 35

3.7 แสดงการต่อใช้งานเบื้องต้น 36

3.8 แสดงบล็อกไดอะแกรมภายในของ ADC0808 37

3.9 แสดงการต่อใช้งาน ADC0808 39

3.10 โครงสร้างภายในของไอซีเรียลไทม์คล็อกเบอร์ DS1307 40

3.11 (ก) การจัดสรรหน่วยความจำแรมภายใน DS1307 41

(ข) รายละเอียดของค่ารีจิสเตอร์เก็บค่าเวลาและรีจิสเตอร์ควบคุมของ DS1307

3.12 รูปแบบของข้อมูลสำหรับติดต่อกับ DS1307 ในโหมดการเขียนข้อมูล 43

3.13 รูปแบบข้อมูลเพื่อใช้ในการติดต่อ DS1307 ในโหมดการอ่านข้อมูล 43

3.14 แสดงการเชื่อมต่อไมโครคอนโทรลเลอร์ MCS-51 กับไอซีเรียลไทม์คล็อก DS1307 44

3.15 แสดงลักษณะการต่อใช้งานแรมขนาด 8 กิโลไบต์ 45

3.16 แสดงการต่อใช้งานร่วมกับ SPI LODE 46

3.17 แสดงลักษณะภายนอกของไอซี DS1833 47

3.18 แสดงลักษณะการต่อใช้งานไอซี DS1833 47

3.19 แสดงการต่อใช้งาน DS275 48

3.20 แสดงลักษณะของคอนเนคเตอร์ DB-9 48

3.21 แสดงลักษณะของข้อมูลที่นำมาจัดเก็บในตาราง 50

สารบัญตาราง

ตาราง	หน้า
2.1 แสดงหน้าที่พิเศษของพอร์ต 1 ใน AT89S8252	5
3.1 สรุปขั้นตอนการติดต่อกับ DS1820	33
3.2 แสดงคุณลักษณะทางไฟฟ้าของ SPI-0509CO	35
3.3 แสดงสถานะการเลือกสัญญาณอนาลอกของ ADC0808	38
3.4 แสดงรายละเอียดการเลือก Rate Select	42

บทที่ 1

บทนำ

ในปัจจุบันนี้สภาพอากาศของประเทศไทย มีการเปลี่ยนแปลงแตกต่างจากในอดีต ช่วงฤดูกาลก็คลาดเคลื่อน จากเดิม ฤดูหนาวบางปีก็หนาวเย็นจัด อีกทั้งอุณหภูมิในเวลากลางวันและกลางคืน ของฤดูหนาวยังมีความแตกต่างกันมาก ช่วงเย็นถึงช่วงเช้า อุณหภูมิเย็นจัด แต่พอช่วงกลางวันอุณหภูมิกลับสูงมาก ส่วนในฤดูร้อนระยะเวลา กลับยาวออกไป อีกทั้งอุณหภูมิยังสูงมากและเวลากลางวันในฤดูร้อนก็ยาวขึ้นมาก เนื่องจากเวลาที่พระอาทิตย์ขึ้นและตกเปลี่ยนไปดังนั้น การที่จะทราบถึงข้อมูลเหล่านี้อย่างละเอียด จึงจำเป็นต้องทำการจัดเก็บบันทึกข้อมูลเหล่านี้ไว้และใช้วิธีการทางสถิติเทียบหาความน่าจะเป็นของสภาพอากาศที่เกิดขึ้น แต่การที่จะคอยมาจดบันทึกข้อมูลเหล่านี้ เป็นเรื่องที่ค่อนข้างเสียเวลาและไม่สะดวกอย่างมากหากสถานที่นั้นอยู่หางไกล เพราะฉนั้นจึงมีแนวความคิดในการสร้างครื่องมือที่สามารถบันทึกข้อมูลสภาพอากาศได้อัตโนมัติขึ้น

วัตถุประสงค์

จากแนวความคิดเบื้องต้น เพื่อต้องการเก็บบันทึกข้อมูลของสภาพอากาศ ได้อย่างอัตโนมัติ ซึ่งทำให้เกิดความสะดวกและลดเวลาในการจดบันทึก อีกทั้งยังสามารถเก็บบันทึกข้อมูลได้ตลอด 24 ชั่วโมง และข้อมูลที่เก็บเหล่านั้นยังสามารถนำไปใช้แสดงผลและประมวลผลในคอมพิวเตอร์ได้อย่างสะดวก ซึ่งขอบเขตของโครงงานมีดังนี้

ขอบเขต

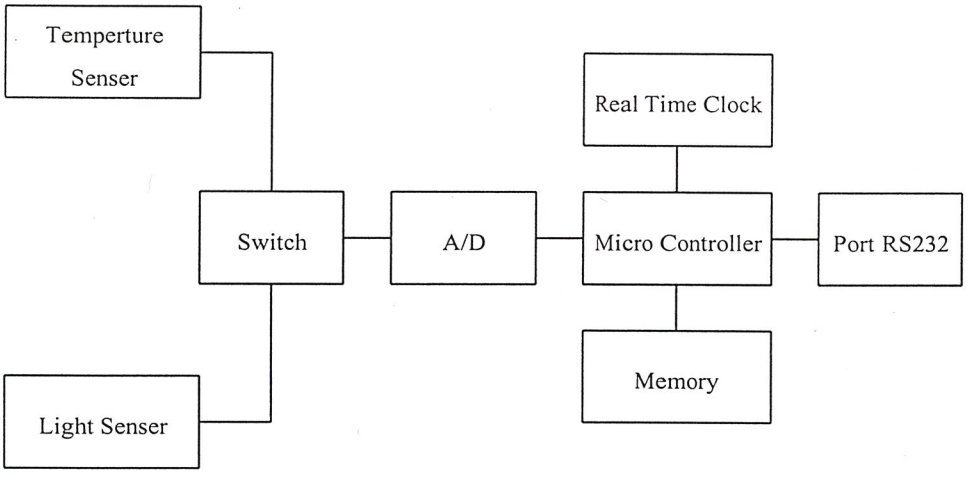
สามารถแบ่งแยกขอบเขตได้เป็น 2 ส่วน

1. ส่วนฮาร์ดแวร์ (Hard Ware)
2. ส่วนซอฟต์แวร์ (Soft Ware)

ส่วนฮาร์ดแวร์

สามารถทำการเก็บค่าข้อมูลสภาพอากาศ โดยประกอบด้วย ค่าอุณหภูมิ ค่าความเข้มแสง โดยเก็บเป็นข้อมูลหนึ่งวัน โดยสามารถอ้างอิงเวลาจริงจากวงจรสร้างฐานเวลาเรียลไทม์ (Real Time Clock) และข้อมูลที่จัดเก็บเป็นข้อมูลแบบดิจิตอล (Digital) เก็บไว้ในส่วนของหน่วยความจำของฮาร์ดแวร์ และสามารถทำการส่งข้อมูลเพื่อไปแสดงผลและประมวลผลที่คอมพิวเตอร์ (Computer) ได้

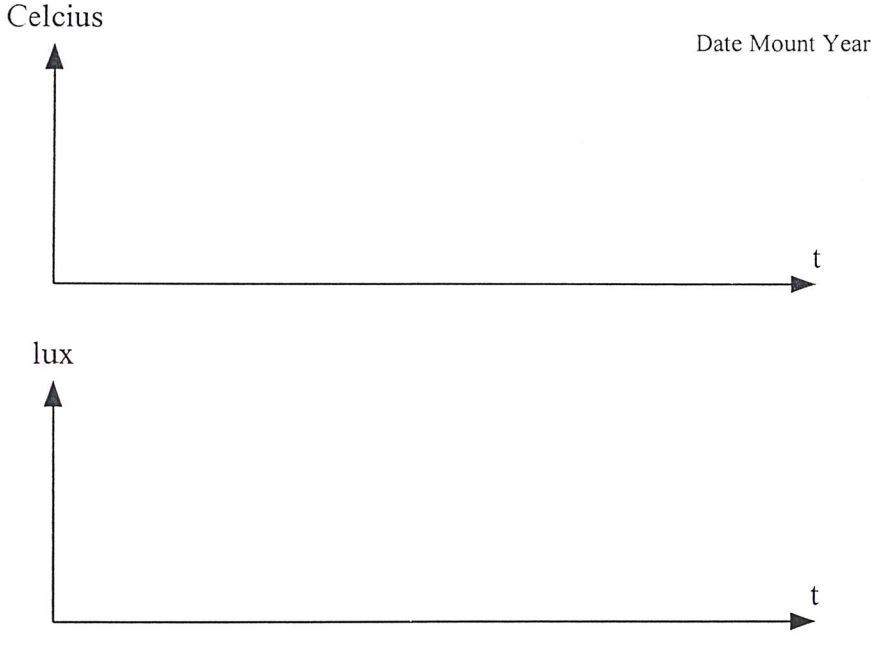
โดยข้อมูลทำการเก็บบันทึกจากเซนเซอร์ (Sensor) 2 ตัวคือ ตัวเซนเซอร์อุณหภูมิ และตัวเซนเซอร์แสง ซึ่งข้อมูลทั้ง 2 ชนิดที่ได้จากตัวเซนเซอร์จะเป็นข้อมูลอนาลอก (Analog) เพราะฉนั้นจึงต้องทำการแปลงข้อมูลจากอนาลอกไปเป็นข้อมูลดิจิตอล โดยใช้วงจร ADC (Analog to Digital Converter) และใช้วงจรสวิตช์ในการเลือกสัญญาณอนาลอกที่จะทำการแปลง แล้วเก็บข้อมูลที่ทำการแปลงเรียบร้อยแล้วไว้ในส่วนของหน่วยความจำ โดยสร้างข้อมูลเวลาจากวงจรฐานเวลาเรียลไทม์ และการส่งข้อมูลไปยังคอมพิวเตอร์จะทำผ่านพอร์ตสื่อสารอนุกรม (Serial Port) RS-232 โดยมีส่วนของไมโครคอนโทรลเลอร์ (Micro Controler) เป็นตัวควบคุมและจัดการทำงาน ซึ่งมีบล็อกไดอะแกรม (Block Diagram) ส่วนประกอบทางฮาร์ดแวร์ดังนี้



รูปที่ 1.1 แสดงบล็อกไดอะแกรมส่วนประกอบทางฮาร์ดแวร์

ส่วนซอฟต์แวร์

ซอฟต์แวร์โปรแกรม (Software Program) มีหน้าที่จัดการข้อมูลที่ส่งมาจากฮาร์ดแวร์ โดยเก็บข้อมูลนั้นไว้ในส่วนของฮาร์ดดิสก์ (Hard Disk) ของคอมพิวเตอร์ และนำข้อมูลนั้นมาแสดงผล โดยข้อมูลของ อุณหภูมิและความเข้มแสง จะแสดงในรูปแบบของกราฟเชิงเส้นเปลี่ยนแปลงตามเวลา ของข้อมูลหนึ่งวัน โดยระบุวัน เดือน ปี อย่างถูกต้อง และสามารถนำข้อมูลเหล่านั้น เก็บไว้เพื่อใช้อ้างอิงและใช้เปรียบเทียบเพื่อแสดงถึงการเปลี่ยนแปลงสภาพอากาศในช่วงเวลาที่ผ่านมาและสามารถจะใช้เพื่อคำนวณความหน้าจะเป็นของสภาพอากาศล่วงหน้าได้ โดยข้อมูลของอุณหภูมิจะแสดงในหน่วยขององศาเซลเซียส (Celcius) ส่วนความเข้มแสงจะแสดงในหน่วยลักซ์ (lux) และแกนเวลาจะเปลี่ยนแปลงในช่วง 24 ชั่วโมง



รูปที่ 1.2 แสดงลักษณะของกราฟที่ใช้แสดงผล

บทที่ 2

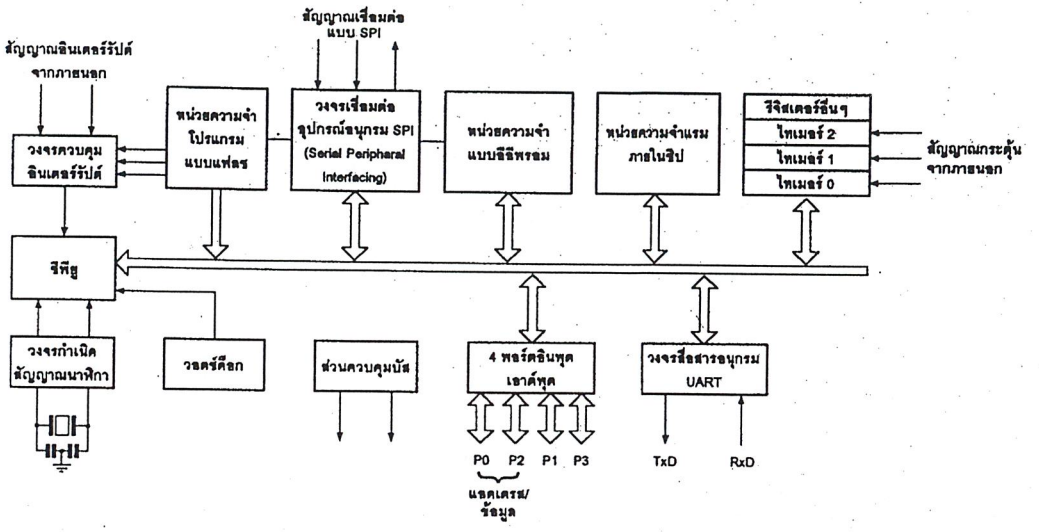
ทฤษฎีที่ใช้ในโครงงาน

ทฤษฎีที่สำคัญเกี่ยวกับ AT89S8252

คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์ ตระกูล MSC-51 เบอร์ AT89S8252

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมแบบแฟลชสามารถลบและเขียนใหม่ได้ 1000 ครั้ง
- หน่วยความจำพื้นฐานเป็นหน่วยความจำแบบแรม และมีหน่วยความจำแบบอีอีพรอม
- ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาต์พุต
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ (Full Duplex)
- ไทม์เมอร์/เคาร์เตอร์ขนาด 16 บิต 3 ตัว
- สามารถรองรับแหล่งกำเนิดการเกิดอินเตอร์รัปต์ได้ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป
- มีวงจรสื่อสารอนุกรมแบบ SPI
- มีวอตช์ด็อกไทเมอร์

สำหรับในรูปที่ 2.1 เป็น โครงสร้างพื้นฐานของ AT89S8252 แสดงวงจรเชื่อมต่ออนุกรมแบบ SPI ซึ่งในไมโครคอนโทรลเลอร์นี้ ใช้ในการเขียนข้อมูลลงในหน่วยความจำโปรแกรมโดยไม่ต้องถอดชิปออกจากระบบหรือเรียกว่าการโปรแกรมในวงจร ไทม์เมอร์ เคาน์เตอร์ ขนาด 16 บิต ที่เพิ่มเติมเข้ามาอีกตัวหนึ่งเป็นไทม์เมอร์ 2 และวงจรวอตช์ด็อกที่ใช้ในการตรวจสอบการทำงานผิดพลาดของซีพียู



รูปที่ 2.1 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MSC-51 แบบแฟลช AT89S8252

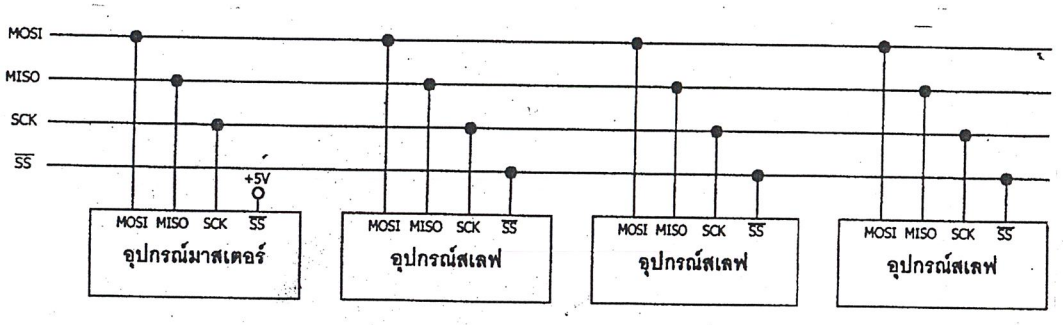
หน้าที่พิเศษของพอร์ต 1 ใน AT89S8252

ตารางที่ 2.1 แสดงหน้าที่พิเศษของพอร์ต 1 ใน AT89S8252

ขา	หน้าที่พิเศษ
P1.0	ขา T2 เป็นขาอินพุตนับค่าของไทมเมอร์/เคาร์เตอร์ 2 และเป็นขาเอาต์พุตของการกำเนิดสัญญาณนาฬิกาโดยไทม์เมอร์ 2 (clock out)
P1.1	ขา T2EX เป็นขาอินพุตทริกเกอร์สำหรับแคปเจอร์/รีโหลด และควบคุมทิศทางของสัญญาณ
P 1.4	ขา SS (Slave Select) เป็นขาเลือกการติดต่อในกรณีที่ไมโครคอนโทรลเลอร์เป็นอุปกรณ์สเลฟ ในระบบการติดต่อแบบ SPI
P 1.5	ขา MOSI (Master data input ,Slave data output) ใช้ในการติดต่อกับพอร์ต SPI
P 1.6	ขา MISO (Master data input , Salve data output) ใช้ในการติดต่อกับพอร์ต SPI
P 1.7	ขา SCK (Master clock output) เป็นขาสัญญาณนาฬิกาของการติดต่อกับพอร์ต SPI

SPI (Serial Peripheral Interface)

คือส่วนที่เชื่อมต่ออุปกรณ์อนุกรมหรือ SPI นี้เป็นการเชื่อมต่อไมโครคอนโทรลเลอร์เข้ากับอุปกรณ์ภายนอกที่มีลักษณะการถ่ายทอดข้อมูลในลักษณะอนุกรมแบบซิงโครนัส (synchronus) ความเร็วสูง ในการสื่อสารข้อมูลแบบซิงโครนัสจะต้องใช้สัญญาณนาฬิกาเพื่อกำหนดจังหวะในการถ่ายทอดข้อมูลร่วมกัน และต้องกำหนดสถานภาพของอุปกรณ์ในระบบอย่างชัดเจน เนื่องจากในการเชื่อมต่อแบบ SPI สามารถต่อพ่วงอุปกรณ์ได้เป็นจำนวนมาก โดยต้องมีการกำหนดตัวควบคุมหลักเป็นตัวแม่หรือมาสเตอร์ (master) และอุปกรณ์ที่นำมาต่อพ่วงในระบบเป็นอุปกรณ์ลูกหรือตัวสเลฟ (slave) ดังแสดงในรูปแบบของ SPI อย่างง่ายในรูปที่ 2.2



รูปที่ 2.2 รูปแบบของการเชื่อมต่อของ SPI อย่างง่าย

อุปกรณ์ตัวแม่หรือมาสเตอร์โดยส่วนใหญ่จะเป็นไมโครคอนโทรลเลอร์ และอุปกรณ์ตัวลูกหรือสเลฟมักเป็นไอซีที่ทำหน้าที่เฉพาะพิเศษ เช่นหน่วยความจำ ไอซีขยายพอร์ตหรือ UART เป็นต้น แต่ด้วยการใช้ความสามารถพิเศษของ SPI อาจสร้างโครงข่ายระบบควบคุมแบบมัลติโปรเซสเซอร์ได้ โดยทำการเชื่อมต่อไมโครคอนโทรลเลอร์หลายตัวเข้าด้วยกันโดยผ่านทางขาเชื่อมต่อของระบบ SPI นั่นหมายความว่า อุปกณ์ตัวลูกหรือสเลฟนั้นก็คือไมโครคอนโทรลเลอร์หรือไมโครโปรเซสเซอร์อีกชุดหนึ่ง

คุณสมบัติของ SPI ในไมโครคอนโทรลเลอร์ AT89S8525

- จัดการสื่อสารข้อมูลแบบฟูลดูเพล็กซ์ ใช้สายสัญญาณในการถ่ายทอดข้อมูล 3 เส้นในลักษณะซิงโครนัส (3- Wire synchronous data tranfer)
- สามารถทำงานเป็นได้ทั้งอุปกรณ์มาสเตอร์และสเลฟ
- ความถี่ของการถ่ายทอดข้อมูลสูงสุด 1.5 MHz
- สามารถเลือกให้ถ่ายทอดข้อมูลในบิต LSB หรือ MSB ก่อนได้
- เลือกอัตราการถ่ายทอดข้อมูลได้ 4 อัตรา
- สามารถถ่ายทอดสัญญาณอินเตอร์รัปต์เมื่อการถ่ายทอดสัญญาณสิ้นสุดลง
- มีการป้องกันการเขียนข้อมูลชนกัน
- เมื่อทำงานเป็นอุปกรณ์สเลฟสามารถออกจากโหมดประหยัดพลังงานแบบไอเดิลได้เมื่อมีการร้องขอให้ติดต่อกับระบบ SPI

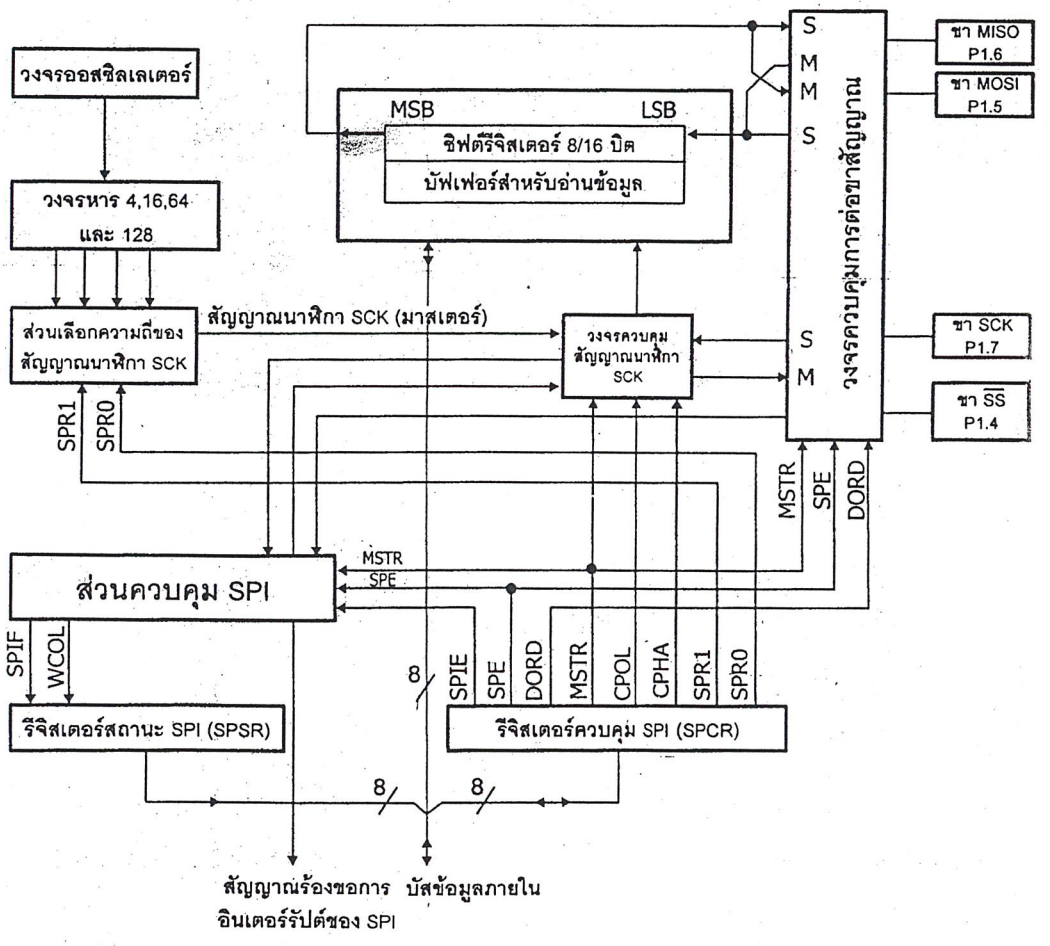
ขาสัญญาณของการเชื่อมต่อแบบ SPI

มีด้วยกัน 4 ขา คือ

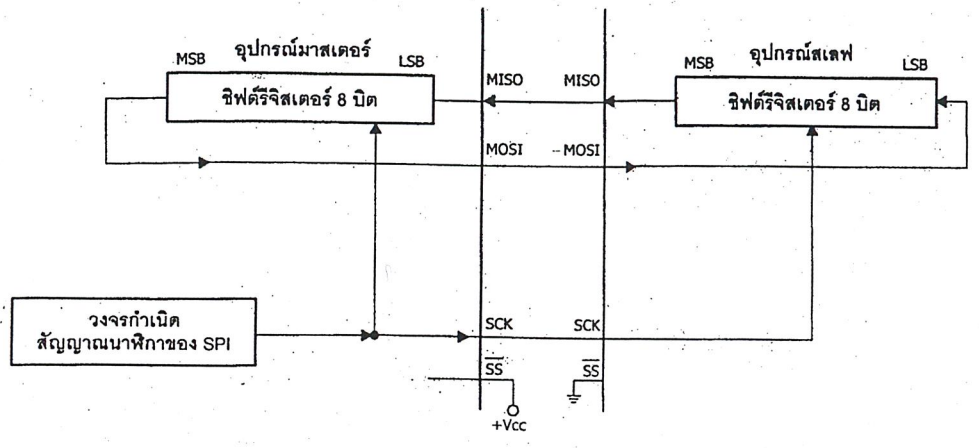
1. **ขา MOSI (Master Out Slave In)** สำหรับอุปกรณ์มาสเตอร์ขานี้จะเป็นข้อมูลออกสำหรับส่งไปยังอุปกรณ์สเลฟ ในขณะที่ถ้าเป็นอุปกรณ์สเลฟ ขานี้จะเป็นข้อมูลเข้า
2. **ขา MISO(Master In Slave Out)** สำหรับอุปกรณ์มาสเตอร์ ขานี้จะเป็นขาข้อมูลเข้าจากอุปกรณ์สเลฟ ในขณะถ้าเป็นอุปกรณ์สเลฟ ขานี้จะเป็นขาข้อมูลออกส่งไปยังอุปกรณ์มาสเตอร์
3. **ขา SCK (SPI clock)** สำหรับอุปกรณ์มาสเตอร์ ขานี้จะเป็นขาส่งสัญญาณนาฬิกาออกไปยังอุปกรณ์สเลฟเพื่อกำหนดจังหวะการถ่ายทอดข้อมูลให้ตรงกัน ในขณะที่ถ้าเป็นอุปกรณ์สเลฟ ขานี้จะเป็นขารับสัญญาณนาฬิกาจากอุปกรณ์มาสเตอร์
4. **ขา SS (Slave Select)** ใช้ในการเลือกอุปกรณ์สเลฟในกรณีที่มีการต่ออุปกรณ์สเลฟพ่วงกันหลายตัว ทำงานที่ลอจิก “0”

การทำงานของ SPI

ในรูปที่ 2.3 แสดงไดอะแกรมของการทำงานของส่วนเชื่อมต่ออุปกรณ์อนุกรมหรือ SPI ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชใน AT89S8252 จะเห็นได้ว่า หัวใจในการทำงานของรีจิสเตอร์ควบคุม SPI (SPI Control register) สำหรับการเชื่อมต่อสายสัญญาณในระบบ SPI ระหว่างอุปกรณ์มาสเตอร์และสเลฟแสดงดังรูปที่ 2.4



รูปที่ 2.3 ไดอะแกรมของการทำงานของส่วน SPI ภายในไมโครคอนโทรลเลอร์ AT89S8252

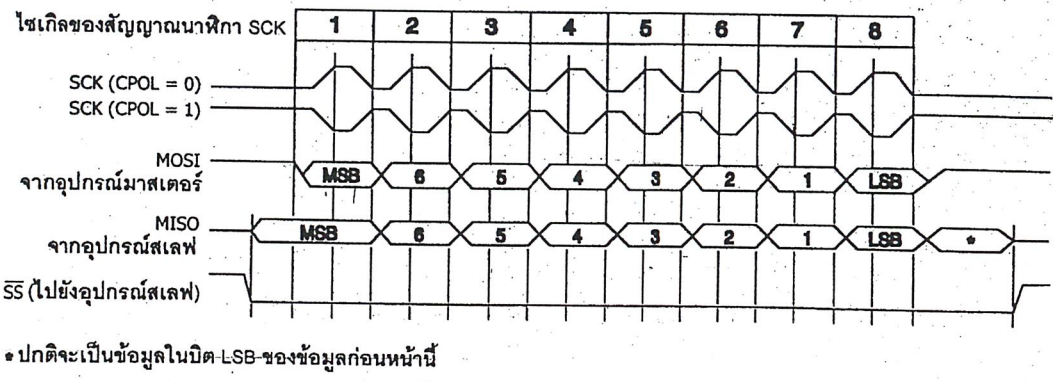


รูปที่ 2.4 การเชื่อมต่อสัญญาณในระบบ SPI ระหว่างอุปกรณ์มาสเตอร์และสเลฟ

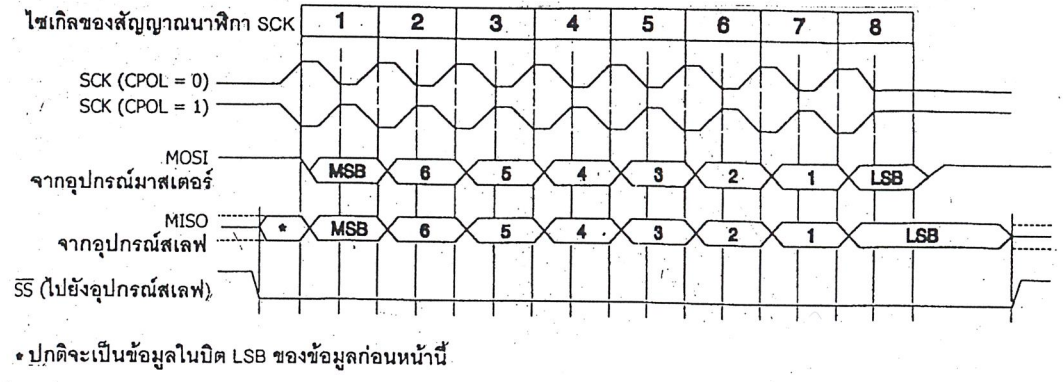
ในการอธิบายการทำงานของส่วน SPI จะใช้รูปที่ 2.3 และ 2.4 ประกอบกัน เมื่อเริ่มต้นการทำงานของขา SCK จะส่งสัญญาณนาฬิกาจากอุปกรณ์มาสเตอร์ไปยังอุปกรณ์สเลฟ โดยสัญญาณนาฬิกานี้ได้มาจากวงจรออสซิลเลเตอร์ภายในผ่านการหารค่าความถี่ด้วย 4 หรือ 16 หรือ 64 หรือ 128 ขึ้นอยู่กับการกำหนดในบิต SPR0 และ SPR1 ในรีจิสเตอร์ควบคุม SPI จากนั้นสัญญาณนาฬิกาจะได้รับการควบคุมการรับเข้าหรือส่งออกโดยวงจรควบคุมสัญญาณนาฬิกา

การเขียนข้อมูลลงในรีจิสเตอร์ข้อมูลของอุปกรณ์สเลฟ เริ่มต้นด้วยซีพียูทำการกระตุ้นให้วงจรกำเนิดสัญญาณนาฬิกาทำงาน ข้อมูลที่ต้องการเขียนจะออกจากอุปกรณ์มาสเตอร์ซึ่งในที่นี้คือไมโครคอนโทรลเลอร์ในลักษณะอนุกรมผ่านทางขา MOSI ไปเข้ายังขา MOSI ของอุปกรณ์สเลฟ หลังจากเลื่อนข้อมูลครบ 1 ไบต์วงจรกำเนิด วงจรกำเนิดสัญญาณนาฬิกาของส่วน SPI จะหยุดทำงานแล้วทำการเซ็ตบิต SPIF เพื่อแจ้งการถ่ายทอดข้อมูลสิ้นสุดลง ถ้าหากบิต SPIE ในรีจิสเตอร์ควบคุม SPI (SPCR) และบิต ES ในรีจิสเตอร์ SCON ได้รับการเซตไว้ด้วย จะเป็นการอีนาเบิลให้เกิดการอินเตอร์รัปต์ขึ้น

ขา SS เป็นขาสำหรับเลือกอุปกรณ์สเลฟ ถ้ามีการเชื่อมต่ออุปกรณ์สเลฟตัวใดเข้าสู่ระบบการติดต่อ SPI ที่ขานี้ต้องได้รับลอจิก “0” ในกรณีเป็นอุปกรณ์มาสเตอร์ต้องทำให้ขานี้มีลอจิกเป็น “1” หากอุปกรณ์สเลฟใดที่ขานี้ได้รับลอจิก “1” จะเป็นการดิสเอเบิลการติดต่อกับระบบ SPI ในทันทีและขา MOSI จะสามารถใช้งานเป็นขาพอร์ตอินพุตได้เท่านั้น



รูปที่ 2.5 ไดอะแกรมเวลาของการถ่ายทอดข้อมูล SPI เมื่อบิต CPHA เป็น "0"



รูปที่ 2.6 ไดอะแกรมเวลาของการถ่ายทอดข้อมูล SPI เมื่อบิต CPHA เป็น "1"

ในรูปที่ 2.5 และ 2.6 เป็นไดอะแกรมเวลาของรูปแบบการถ่ายทอดข้อมูลในส่วน SPI ซึ่งได้รับการกำหนด โดยบิต CPHA และ CPOL ในรีจิสเตอร์ควบคุม SPI

เมื่อนำไมโครคอนโทรลเลอร์ AT89S8252 มาใช้งานในด้าน SPI ขาพอร์ต 1.4-1.7 ซึ่งก็คือขา SS,MOSI,MISO และ SCK ไม่ควรนำไปใช้งานอื่นใดอีก ซึ่งอาจส่งผลกระทบต่อการจัดสรรพอร์ตของไมโครคอนโทรลเลอร์ บ้างแต่ไม่มากนัก เนื่องจากในไมโครคอนโทรลเลอร์นี้มีจำนวนขาพอร์ตให้ใช้มากถึง 32 บิต นำไปใช้ในงาน SPI เพียง 4 บิต ยังคงเหลือให้ใช้งานได้อีก 28 บิต ซึ่งน่าจะเพียงพอสำหรับการสร้างระบบควบคุมที่ทรงประสิทธิภาพ โดยเฉพาะอย่างยิ่งการใช้

หน่วยความจำโปรแกรมภายในไมโครคอนโทรลเลอร์ยิ่งทำให้การใช้งานพอร์ตสามารถกระทำได้อย่างเต็มที่

รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานในของส่วน SPI ในไมโครคอนโทรลเลอร์ AT89S8252

การทำงานของส่วนเชื่อมต่ออุปกรณ์อนุกรมหรือ SPI ของ ไมโครคอนโทรลเลอร์ AT89S8252 มีรีจิสเตอร์ที่ต้องเกี่ยวข้องทั้งสิ้น 3 ตัว ดังมีรายละเอียดดังต่อไปนี้

1. **รีจีสเตอร์ข้อมูล SPI หรือ SPDR (SPI Data register)**

มีแอดเดรสอยู่ที่ 86H ในพื้นที่ของรีจีสเตอร์ฟังก์ชั่นพิเศษ หรือ SFR มีขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต ใช้ในการเก็บข้อมูลที่เกิดการถ่ายทอดขึ้นในการทำงานของส่วน SPI ชื่อเรียกบิตต่างๆภายในรีจีสเตอร์ SPDR เรียงลำดับจากบิต MSB คือบิต SPD7-SPD0 รีจีสเตอร์ตัวนี้มีความพิเศษคือ เมื่อเกิดการรีเซตขึ้น ข้อมูลภายในรีจีสเตอร์ SPDR นี้จะไม่มีการเปลี่ยนแปลง

2. **รีจีสเตอร์ควบคุม SPI หรือ SPCR (SPI Control register)**

มีแอดเดรสอยู่ที่ D5H ในพื้นที่ของรีจีสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต เมื่อเกิดการรีเซตค่าของรีจีสเตอร์ตัวนี้จะเป็น 000001XXB (X หมายถึงค่าเดิมก่อนหน้าที่จะเกิดการรีเซต) มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

SPIE (SPI Interrupt Enable) : ใช้อีนาเบิลการอินเตอร์รัปต์อันเนื่องมาจากการถ่ายทอดข้อมูลของ SPI เสร็จสิ้นลง บิตนี้ต้องทำงานร่วมกับบิต SPIF ในรีจีสเตอร์ SPSR และบิต ES ในรีจีสเตอร์ IE การอินเตอร์รัปต์จะเกิดขึ้นได้เมื่อบิต SPIF, SPIE และ ES เป็น "1" ทั้งหมด ถ้าบิต SPIE เป็น "0" จะเป็นการดิสเอเบิลการอินเตอร์รัปต์ในระบบ SPI

SPE (SPI Enable) : ใช้อีนาเบิลการทำงานของ SPI เมื่อบิตนี้เซตเป็น "1" วงจรควบคุมภายในไมโครคอนโทรลเลอร์ AT89S8252 จะทำการเชื่อมต่อวงจรของ SS, MOSI, MISO และ SCK เข้ากับขา P1.4, P1.5, P1.6 และ P1.7 ของพอร์ต 1 ทันที เพื่อเตรียมการทำงานของ SPI

DORD (Data Order) : ใช้เลือกลำดับการถ่ายทอดข้อมูลของ SPI

"0" เลือกถ่ายทอดข้อมูลในบิต MSB ก่อน

"1" เลือกถ่ายทอดข้อมูลในบิต LSB ก่อน

MSTR (Master / Slave Select) : ใช้เลือกโหมดการทำงานของ SPI ของอุปกรณ์

"0" เลือกทำงานเป็นอุปกรณ์แบบสเลฟ

"1" เลือกทำงานเป็นอุปกรณ์แบบมาสเตอร์

CPOL (Clock Polarity) : ใช้ร่วมกับบิต CPHA ในการกำหนดจังหวะการทำงานของสัญญาณนาฬิกา SCK (พิจารณาไดอะแกรมเวลาในรูปที่ 2.5 และ 2.6 ประกอบ)

"0" ทำให้สัญญาณนาฬิกา SCK ของอุปกรณ์มาสเตอร์เป็นลอจิก "0" เมื่อไม่มีการส่งข้อมูล

"1" ทำให้สัญญาณนาฬิกา SCK เป็น "1" เมื่ออยู่ในสภาวะไอเดิล

CPHA (Clock Phase) : ใช้ร่วมกับบิต CPOL ในการกำหนดรูปแบบของสัญญาณนาฬิกา SCK (ใช้ไดอะแกรมเวลาในรูปที่ 2.5 และ 2.6 พิจารณาประกอบ)

SPR1-SPR0 (SPI Clock Rate Select) : ใช้กำหนดอัตราของสัญญาณนาฬิกา SCK การกำหนดนี้ต้องกระทำที่อุปกรณ์มาสเตอร์เท่านั้น

"00" อัตราสัญญาณนาฬิกามาจากความถี่สัญญาณนาฬิกาหลักหาร 4

"01" อัตราสัญญาณนาฬิกามาจากความถี่สัญญาณนาฬิกาหลักหาร 16

"10" อัตราสัญญาณนาฬิกามาจากความถี่สัญญาณนาฬิกาหลักหาร 64

"11" อัตราสัญญาณนาฬิกามาจากความถี่สัญญาณนาฬิกาหลักหาร 128

3. รีจิสเตอร์สถานะ SPI หรือ SPSR (SPI Status register)

มีแอดเดรสอยู่ที่ AAH ในพื้นที่ของรีจิสเตอร์ฟังชั่นพิเศษหรือ SFR มีขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต มีการใช้งานเพียง 2 บิต เมื่อเกิดการีเซตค่าของรีจิสเตอร์ตัวนี้จะเป็น 00XXXXXXB (X หมายถึงเป็นค่าเดิมก่อนที่จะเกิดการรีเซต) รายละเอียดของการใช้งานมีดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SPIF	WCOL	-	-	-	-	-	-

SPIF (SPI Interrupt Flag) : เมื่อการถ่ายทอดข้อมูลของ SPI เสร็จสิ้นลง บิตนี้จะเซตและจะทำให้เกิดการอินเตอร์รัปต์ขึ้นถ้าบิต SPIF ในรีจิสเตอร์ SPSR และบิตES ในรีจิสเตอร์ IE เป็น "1" การเคลียร์บิตนี้จะกระทำได้ด้วยวิธีการทาง ซอฟต์แวร์โดยการอ่านค่าของรีจิสเตอร์ SPSR และโดยการเข้าถึงข้อมูลในรีจิสเตอร์ข้อมูล SPI หรือ SPDR

WCOL (Write Collision Flag) : ใช้แสดงการชนกันของข้อมูล เหตุการณ์นี้จะเกิดขึ้นเมื่อรีจิสเตอร์ข้อมูล SPI หรือ SPDR ได้รับการเขียนในขณะที่กำลังถ่ายทอดข้อมูลอยู่ ส่งผลให้ค่าของรีจิสเตอร์ SPDR ที่อ่านได้มีค่าไม่ถูกต้อง บิตนี้จะเซตเมื่อเกิดเหตุการณ์ดังกล่าวขึ้น สามารถเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์โดยการอ่านค่าของรีจิสเตอร์ SPSR หลังเกิดการเซตที่บิตนี้ และโดยการเข้าถึงรีจิสเตอร์ SPDR

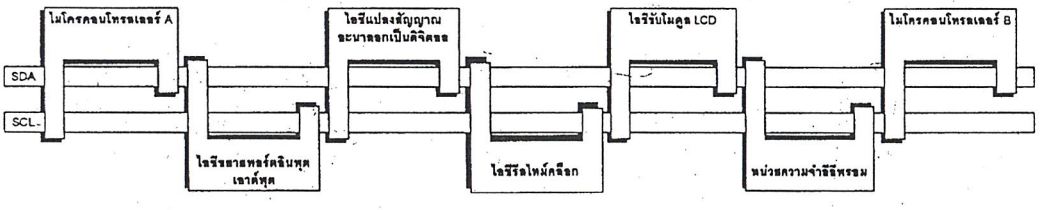
การใช้ประโยชน์จากส่วน SPI

นอกเหนือไปจากการใช้งานของ SPI ในการติดต่อสื่อสารข้อมูลแบบอะซิงโครนัสแล้ว ในไมโครคอนโทรลเลอร์ AT89S8252 ยังใช้ความสามารถของส่วน SPI ในการเขียนข้อมูลลงในหน่วยความจำโปรแกรมภายในตัวชิปด้วย จึงทำให้กระบวนการที่เรียกว่า การโปรแกรมหน่วยความจำ โปรแกรมในระบบหรือในวงจร (In-System Programming : ISP) ทำให้สามารถแก้ไขหน่วยความจำโปรแกรมได้โดยไม่ต้องถอดตัวไมโครคอนโทรลเลอร์ออกจากระบบเพียงต่อสายเข้าที่พอร์ต SPI คือขา SS, MOSI, MISO และ SCK (ซึ่งก็คือขา P1.4-P1.7) แล้วทำการรันซอฟต์แวร์ที่เขียนขึ้นเพื่อการโปรแกรมในระบบ ก็สามารถแก้ไขหน่วยความจำได้แล้ว

ทฤษฎีเบื้องต้นเกี่ยวกับระบบบัส I^2C

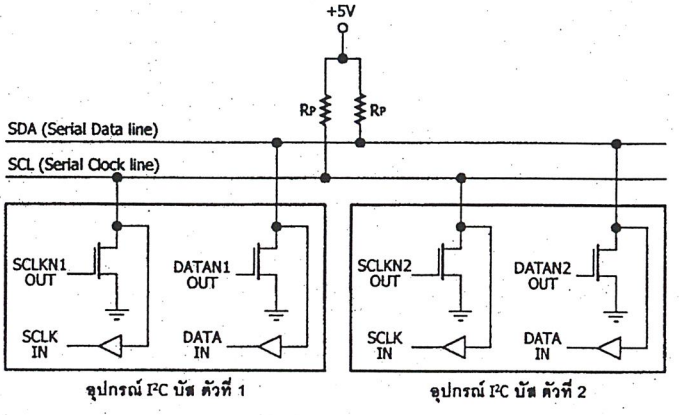
I^2C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซีโดยบัส I^2Cโดยได้รับการพัฒนาขึ้นโดยฟิลิปส์ (Philips)โดยจุดมุ่งหมายหลักคือ ต้องการให้ ไอซีหรือโมดูลสามารถติดต่อสั่งงานและควบคุมภายใต้สัญญาณเพียง 2 เส้น เส้นหนึ่งคือ สายข้อมูล อีกเส้นหนึ่งคือ สายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือต่อพ่วงกันไปส่วนการกำหนดแอดเดรสหรือตำแหน่งสำหรับการติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบนบัส I^2C มีชื่อเรียกอีกอย่างหนึ่งเป็นทางการว่า สายข้อมูลอนุกรม SDA (Serial Data line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกอีกอย่างหนึ่งว่า สายสัญญาณนาฬิกาอนุกรม หรือ SCL (Serial Clock line) ในการอธิบายต่อไปนี้จะเรียกสายสัญญาณทั้งสองอีกอย่างว่า สาย SDA และ SCL



รูปที่ 2.7 ผังแสดงการเชื่อมต่อของอุปกรณ์ต่างๆ บนระบบบัส I^2C

ในรูปที่ 2.8 แสดงผังการเชื่อมต่ออุปกรณ์ต่างๆบนบัส I^2C จะเห็นได้ว่า อุปกรณ์ทีทำการเชื่อมต่อบนบัส I^2C มีหลากหลาย ไม่ว่าจะเป็นไอซีที่ขยายพอร์ตอินพุตเอาต์พุต (I/O Expander),ไอซีจะแปลงสัญญาณอนาลอกเป็นดิจิตอล(ADC),และแปลงสัญญาณดิจิตอลเป็นอนาลอก (DAC), ไอซีไทม์คล็อก (RTC),ไอซีขับโมดูล LCD, หน่วยความจำอีอีพรอม และไมโครคอนโทลเลอร์



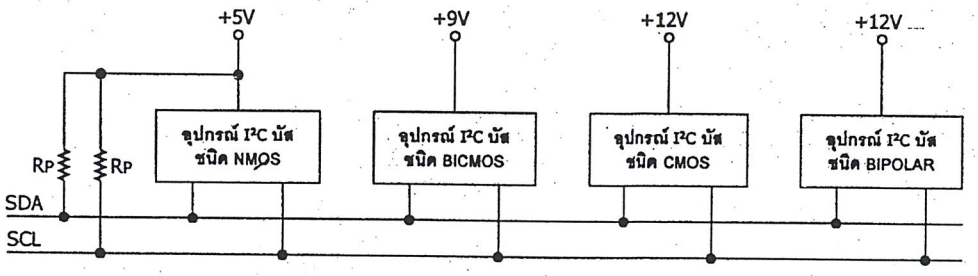
รูปที่ 2.8 แสดงวงจรเอาต์พุตของอุปกรณ์ในระบบบัส I^2C

คุณสมบัติโดยทั่วไปของบัส I^2C

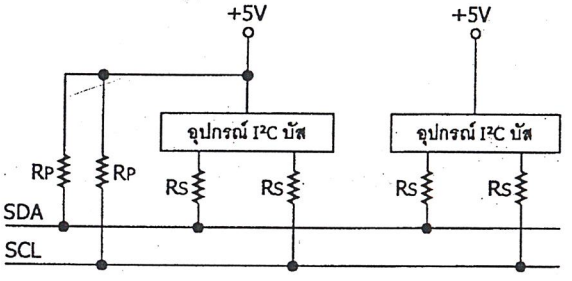
สายสัญญาณ SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (bi-direction line) ต้องมีการต่อตัวต้านทานพูลอัปกับแรงดัน +5V ไว้ตลอดเวลา เพื่อให้สายสัญญาณมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งานทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุตของอุปกรณ์ที่ต่ออยู่บนบัส I^2C ต้องมีลักษณะที่เป็นวงจรเดรนเปิด (open-drain) หรือคอลเล็คเตอร์เปิด (open-collector) ดังแสดงรายละเอียดในรูปที่ 2.8

อัตราการถ่ายทอดข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติ (standard mode) และสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง (fast mode) อุปกรณ์ที่ต่อร่วมกันอยู่บนบัส I^2C จะต้องมีความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400pF การเข้าถึงอุปกรณ์บนบัส I^2C ใช้ข้อมูลสำหรับการเข้าถึง 2 ค่าคือ 7 บิต (7-bit addresing) หรือ 10 บิต (10-bit addressing)

ข้อเด่นอีกประการหนึ่งของบัส I^2C คือ สามารถเชื่อมอุปกรณ์ที่ใช้ไฟเลี้ยงไม่เท่ากันให้สามารถติดต่อสื่อสารกันได้โดยอุปกรณ์บนบัส I^2C ตัวหนึ่งอาจใช้ไฟเลี้ยง +5V ในขณะที่อีกตัวหนึ่งใช้ไฟเลี้ยง +12V การต่อกันบนบัส I^2C สามารถกระทำได้ในลักษณะเดียวกันกับกรณีที่อุปกรณ์ทั้งสองใช้ไฟเลี้ยงเท่ากัน กล่าวคือ ให้สาย SDA และ SCL ของอุปกรณ์แต่ละตัวต่อเข้าด้วยกัน และต้องต่อตัวต้านทานพูลอัป (Rp) เข้ากับแรงดัน +5V ไว้ด้วยเสมอ ดังแสดงในรูปที่ 2.9



รูปที่ 2.9 การต่อตัวต้านทานพูลอัปบนสายสัญญาณในระบบบัส I^2C



รูปที่ 2.10 การต่อตัวต้านทาน Rs เพื่อลดสัญญาณรบกวนขนาดใหญ่ที่อาจเข้ามาในระบบบัส I^2C

ในกรณีที่อาจมีแรงดันไฟกระชากขนาดใหญ่ปะปนเข้ามาในบัส I^2C ที่ขา SDA และ SCL ของอุปกรณ์แต่ละตัวจะต้องต่อตัวต้านทานอนุกรมกับขา SDA และ SCL เรียกว่า Rs ก่อนต่อเข้าบัส I^2C ดังแสดงในรูปที่ 2.10

หลักการของบัส I^2C

บัส I^2C ประกอบด้วยสายสัญญาณ 2 เส้น ดังที่กล่าวมาแล้วคือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส หรือเรียกว่า โปรโตคอล (protocol) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ตัวใดติดต่อกันอยู่ และอุปกรณ์ตัวใดเป็นตัวรับหรือตัวส่ง ต่อจากนี้จะขออธิบายลักษณะ หน้าที่ และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส I^2C เพื่อเป็นข้อตกลงก่อนที่จะอธิบายการทำงานของระบบบัส I^2C ต่อไป

อุปกรณ์ที่เป็นตัวสร้างข้อมูลหรือส่งข้อมูล เรียกว่า ตัวส่ง (transmitter)

อุปกรณ์ที่เป็นตัวรับข้อมูล เรียกว่า ตัวรับ (receiver) ในอุปกรณ์บนบัส I^2C สามารถเป็นได้ทั้งตัวส่งและตัวรับ บางอุปกรณ์ทำหน้าที่เป็นตัวรับได้อย่างเดียว

อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I^2C เรียกว่า มาสเตอร์ (master)

อุปกรณ์ที่ควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าบนบัส I^2C เรียกว่า สเลฟ (slave)

ข้อกำหนด 2 ประการสำคัญของการติดต่อบนบัส I^2C คือ

(1) การถ่ายทอดข้อมูลจะเกิดขึ้นได้ก็ต่อเมื่อสายบัสว่างเท่านั้น

(2) ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายสัญญาณต้องงรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้นสัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควมคุมแทน

สภาวะที่เกิดขึ้นบนบัส I^2C

มีด้วยกัน 5 สภาวะดังนี้

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

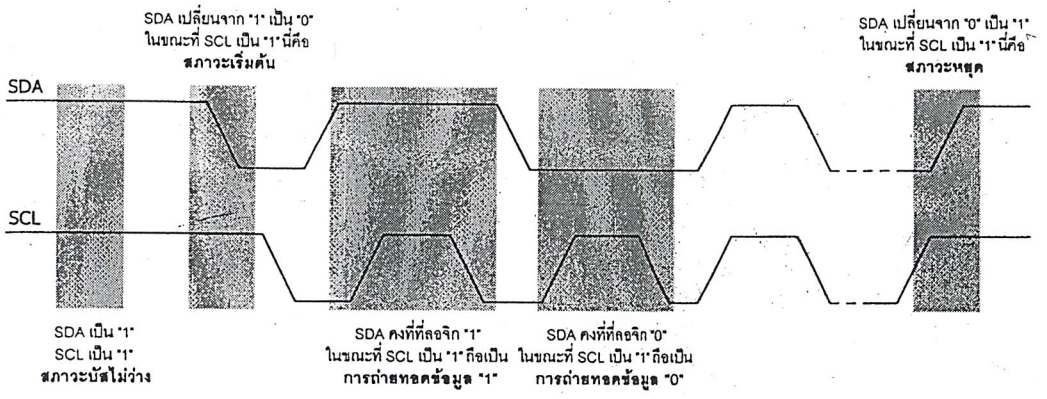
(2) **เริ่มต้นการถ่ายทอดข้อมูล (stop data tranfer)** เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า **สภาวะเริ่มต้น (START)**

(3) **หยุดการถ่ายทอดข้อมูล (stop data tranfer)** เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า **สภาวะหยุด (STOP)**

(4) **ข้อมูลดำรงอยู่บนบัส (data valid)** สภาวะที่เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่าเป็น “0” หรือ “1” ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ ต่อเมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกที่สูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมาย

เป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดนั้นเกิดความผิดพลาดขึ้น

(5) **รับรู้ข้อมูล (acknowledge)** เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งที่ทำการส่งข้อมูลมา 1 บิต เรียกว่า **บิตรับรู้ (acknowledge bit)** มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา เพื่อตอบสนองบิตรับรู้ที่ส่งมาจากตัวส่ง ทางด้านตัวรับจะส่งบิตรับรู้ที่มีสถานะลอจิกต่ำลงบนบัส อุปกรณ์สเลฟที่ถูกอ้างอิงถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว



รูปที่ 2.11 ไดอะแกรมเวลาสถานะต่างๆในบัส I^2C

ในรูปที่ 2.11 เป็นไดอะแกรมเวลาที่แสดงถึงสภาวะต่างๆ บนบัส I^2C ไม่ว่าจะป็นสภาวะว่าง, เริ่มถ่ายทอดข้อมูล, รับรู้ข้อมูล, รับรู้ และหยุดการถ่ายทอดข้อมูล

การทำงานบนบัส I^2C

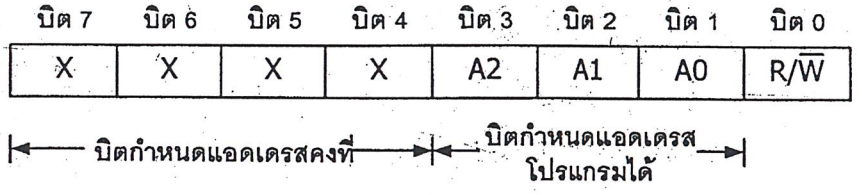
ก่อนที่เริ่มต้นการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่างๆ ที่ต่ออยู่บนบัส ต้องมีการอ้างถึงเสียก่อนโดยการอ้างถึงอุปกรณ์บนบัส I^2C นั้นจะใช้การอ้างอิงถึง 7 บิตหรือ 10 บิต ในกรณีที่มีอุปกรณ์ต่ออยู่บนบัสไม่มาก ใช้การอ้างอิงถึง 7 บิตก็เพียงพอ แต่ถ้ามีอุปกรณ์ต่ออยู่บนบัสมากกว่า 127 แอด

เดรส จำเป็นต้องใช้การอ้างอิงถึง 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้วก็เริ่มต้นการถ่ายทอดข้อมูลกันต่อไป

ดังนั้นหัวใจสำคัญอันดับแรกของการทำงานบนบัส I^2C คือการอ้างอิงถึงอุปกรณ์แต่ละตัว ซึ่งในที่นี้จะอธิบายรายละเอียดถึงการอ้างอิง 2 รูปแบบ

การอ้างอิงแบบ 7 บิต (7 – bit addressing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างอิงถึงอุปกรณ์ที่ต้องการติดต่อ หรือ ข้อมูลกำหนดแอดเดรส โดยมีรูปแบบแสดงไว้ดังรูปที่ 2.12 ใน 7 บิตรวมทั้งบิต MSB ด้วยจะเป็นแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ โดยแบ่งปัน บิตกำหนดแอดเดรสคงที่ (fixed address bit) จำนวน 4 บิตซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิตเป็นบิต ที่กำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) โดยผู้ที่ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I^2C ส่วนในบิต LSB ที่ใช้กำหนดการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้นๆ หากบิต LSB เป็น “0” หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์นั้นๆ หากเป็น “1” จะเป็นการอ่านข้อมูลจากเสลฟ

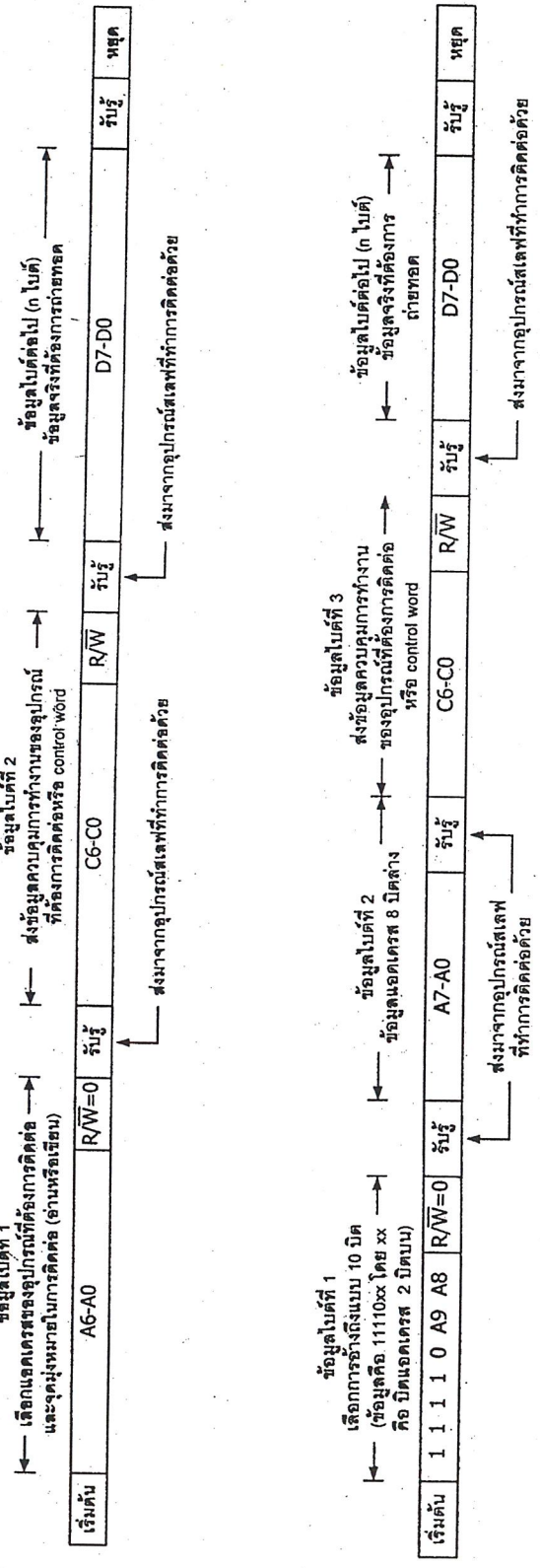


รูปที่ 2.12 รูปแบบของข้อมูลที่กำหนดแอดเดรสที่ใช้ในการอ้างอิง 7 บิต

ข้อมูลในไบต์ต่อมาคือ ข้อมูลควบคุม (control byte) ในอุปกรณ์แต่ละตัวจะมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่างไอซีขยายพอร์ตมีข้อมูลควบคุมที่ให้กำหนดว่า บิตใดเป็นบิตอินพุต บิตใดเป็นเอาต์พุต ในขณะที่ ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดการทำงานให้เป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลในไบต์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง (data)

หลังจากที่การถ่ายทอดข้อมูลแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้ ในรูปที่ 2.13 แสดงรูปแบบข้อมูลอนุกรมที่เกิดขึ้นในการติดต่อบนบัส I^2C ของการอ้างอิงถึงแบบ 7 บิต



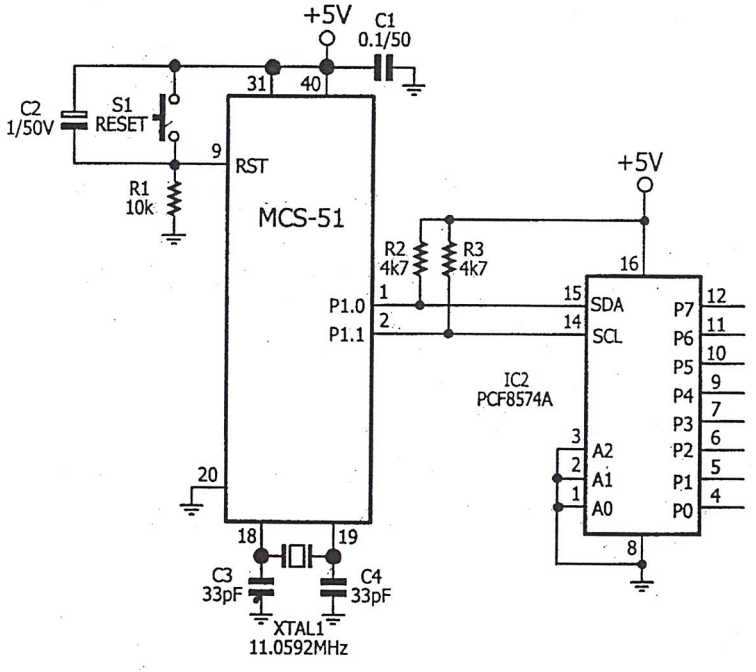
รูปที่ 2.13 รูปแบบขอข้อมูลอนุกรมที่ใช้ในการติดต่ออุปกรณ์บนบัส I^2C เมื่อใช้ในการอ้างอิงถึง 7 บิต

รูปที่ 2.14 รูปแบบขอข้อมูลอนุกรมที่ใช้ในการติดต่ออุปกรณ์บนบัส I^2C เมื่อใช้ในการอ้างอิงถึง 10 บิต

การอ้างอิงถึง 10 บิต

ในการอ้างถึงแบบนี้ยังคงใช้รูปแบบข้อมูลอนุกรมที่เหมือนกับแบบ 7 บิต หากแต่จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในข้อมูลไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิตเป็นบิตแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่า ต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟที่ต้องการติดต่อด้วย ข้อมูลไบต์ถัดไปเป็นข้อมูลควบคุม ข้อมูลที่หลังจากนั้นจะเป็นข้อมูลจริงที่ใช้ในการติดต่อ

เช่นเดียวกับการอ้างอิงถึง 7 บิต หลังจากการถ่ายทอดข้อมูลครบทุกไบต์ ต้องมีสภาวะการรับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้ ในรูปที่ 2.14 แสดงรูปแบบข้อมูลอนุกรมของการอ้างอิงถึงแบบ 10 บิต



รูปที่ 2.15 วงจรตัวอย่างการต่อไมโครคอนโทรลเลอร์ MCS51 กับอุปกรณ์ระบบบัส I^2C

การติดต่ออุปกรณ์ระบบบัส I^2C กับไมโครคอนโทรลเลอร์ MCS-51

สามารถทำได้ง่ายมากเพียงใช้ขาพอร์ต 2 ขา โดยกำหนดให้ขาหนึ่งเป็น SDA อีกขาหนึ่งเป็น SCL และต่อตัวต้านทานประมาณ 4.7K พูลอัปที่ขาพอร์ตทั้งสองขา เพียงเท่านี้ก็สามารถติดต่อสื่อสารกับอุปกรณ์ระบบบัส I2C ได้แล้ว

ในรูปที่ 2.15 เป็นวงจรตัวอย่างการต่อไมโครคอนโทรลเลอร์ MCS-51 เข้ากับระบบบัส I^2C จากวงจรใช้ขาพอร์ต P1.0 เป็นขา SDA และ P1.1 เป็นขา SCL อุปกรณ์ที่ทำการติดต่อด้วยคือไอซีขยายพอร์ตอินพุตเอาต์พุต เบอร์ PCF8574

การเขียนโปรแกรมติดต่อกับ I^2C

เริ่มต้นด้วยการสร้างสภาวะมาตฐานของบัส I^2C อันประกอบด้วย สภาวะเริ่มต้น, สภาวะสิ้นสุดการส่งข้อมูล, สภาวะหยุดและสัญญาณบนขา SCL

การสร้างสภาวะเริ่มต้น

1. เมื่อต้องการติดต่อกับบัส I^2C สิ่งแรกที่ต้องทำสำหรับไมโครคอนโทลเลอร์ซึ่งถือว่าเป็นอุปกรณ์มาสเตอร์คือ การทำให้บัสว่างด้วยการกำหนดให้ขา SCL และขา SDA มีลอจิกเป็น “1” ทั้งคู่
2. จากนั้นทำให้ขา SDA มีลอจิกเป็น “0” โดยที่ขา SCL ยังคงเป็นลอจิก “1” อยู่
3. การกำหนดให้ขา SCL มีลอจิกเป็น “0” ถึงตอนนี้ทั้ง SCL และ SDA มีลอจิกเป็น “0” ทั้งคู่พร้อมที่จะติดต่อได้แล้ว

การสร้างสภาวะหยุด

1. เมื่อต่องการหยุดส่งข้อมูลจะต้องส่งสภาวะหยุดออกไป โดยที่ตอนแรกต้องกำหนดให้ขา SCL และ SDA เป็นลอจิก “0” ทั้งคู่ก่อน
2. กำหนดให้ขา SLC มีลอจิกเป็น “1” โดยที่ SDA ยังคงมีลอจิกเป็น “0”
3. จะต้องให้ขา SDA มีลอจิกเป็น “1” ซึ่งทำให้ระบบบัสกลับเข้าสู่บัสว่างอีกครั้งพร้อมที่จะรับหรือส่งข้อมูลออกไป

การส่งข้อมูลลอจิก "0" และลอจิก "1"

หลังจากที่ทำการส่งบิตเริ่มต้นแล้ว ลำดับต่อไปคือ จะต้องส่งข้อมูลควบคุมซึ่งจะเป็นขบวนของลอจิก "0" และลอจิก "1" สำหรับการส่งข้อมูลลอจิก "0" ต้องดำเนินการตามขั้นตอนดังต่อไปนี้

1. ทำให้ขา SDA เป็น "0" สำหรับการส่งข้อมูลลอจิก "0"
2. ทำให้ขา SCL เป็น "1" สำหรับการป้อนสัญญาณนาฬิกา ในขณะที่ขา SDA ยังคงเป็น "0" อยู่
3. จากนั้นทำให้ขา SCL กลับมามีสถานะเป็นลอจิก "0" เหมือนเดิม

ในขณะที่ส่งข้อมูลลอจิก "1" มีขั้นตอนดังนี้

1. ทำให้ขา SDA เป็น "1" สำหรับการส่งข้อมูลลอจิก "1"
2. ทำให้ขา SCL เป็น "1" สำหรับการป้อนสัญญาณนาฬิกา ในขณะที่ขา SDA ยังคงเป็น "1" อยู่
3. จากนั้นทำให้ขา SCL กลับมามีสถานะเป็นลอจิก "0" เหมือนเดิม

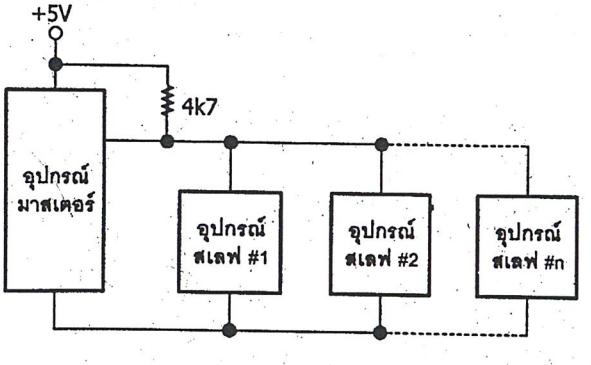
ทฤษฎีระบบการสื่อสารอนุกรมแบบหนึ่งสาย (1 – Wire^{TM} Serial Bus)

ระบบการสื่อสารข้อมูลแบบนี้ผู้คิดค้นคือ ดัลลัสเซมิคอนดักเตอร์ ระบบการสื่อสารแบบนี้เป็นระบบที่มีความชาญฉลาดและใช้จำนวนสายสัญญาณเพียง 1 เส้นเท่านั้น โดยไม่ต้องมีสายสัญญาณนาฬิกามาควบคุมจังหวะการถ่ายทอดข้อมูลเหมือนกับการสื่อสารอนุกรมในแบบอื่นๆ เนื่องจากสายข้อมูลนั้นจะทำหน้าที่เสมือนหนึ่งเป็นสายสัญญาณนาฬิกาในตัว ส่วนค่าของข้อมูลจะพิจารณาจากลักษณะของรูปสัญญาณที่ปรากฏบนสายสัญญาณในแต่ละช่องของเวลาหรือต่อไปนี้จะขอเรียกว่า ไทม์สล็อต (Time-slot) โดยคาบเวลาต่ำสุดและสูงสุดของสถานะต่างๆในการสื่อสารข้อมูลในแต่ละไทม์สล็อต มีการกำหนดขอบเขตไว้อย่างชัดเจน การถ่ายทอดข้อมูลจะเกิดในแต่ละไทม์สล็อตนั้น รูปแบบการถ่ายทอดข้อมูลจะเป็นแบบอะซิงโครนัสในระดับบิต ไม่มีการกำหนดความยาวของข้อมูลเป็นระดับไบต์ ระบบการสื่อสารแบบนี้จะเหมาะที่จะใช้ในการสื่อสารข้อมูลระหว่างไอซีบนแผงวงจรเดียวกัน หรือสร้างเป็นโครงข่ายสื่อสารแบบทวิสต์แพร์ก็ได้

คุณสมบัติทางเทคนิคของระบบบัสหนึ่งสาย

สายสัญญาณบนระบบบัสหนึ่งสายนี้จะเป็นสายสัญญาณแบบสองทิศทาง แต่ข้อมูลจะสามารถเดินทางได้ในทิศทางเดียวในเวลาช่วงหนึ่งๆ นั่นคือมีลักษณะคล้ายกับระบบสื่อสารแบบฮาล์ฟดูเพล็ก (half-duplex) ตัวอย่างที่เห็นได้ชัดคือ การใช้งานวิทยุสื่อสารหรือวิทยุสมัครเล่น อุปกรณ์บนระบบบัสต้องมีการระบุอย่างชัดเจนว่าตัวใดเป็นอุปกรณ์มาสเตอร์ ตัวใดเป็นอุปกรณ์สเลฟ โดยส่วนใหญ่อุปกรณ์มาสเตอร์คือไมโครคอนโทรลเลอร์ ส่วนอุปกรณ์สเลฟได้แก่ ไอซีตรวจจับอุณหภูมิ,ไอซีหน่วยความจำแรม เป็นต้น อุปกรณ์มาสเตอร์จะเป็นตัวจัดเตรียมความพร้อมของสายสัญญาณและควบคุมการถ่ายทอดข้อมูลบนสายสัญญาณนั้น ข้อมูลทั้งหมดไม่ว่าจะเป็นข้อมูลควบคุมหรือข้อมูลใช้งานจะถูกส่งลงบนสายสัญญาณที่มีอยู่เพียงเส้นเดียวนี้ทั้งหมดในระหว่างการทำงานอุปกรณ์มาสเตอร์และสเลฟสามารถเป็นได้ทั้งตัวส่งและตัวรับ ขึ้นอยู่กับเงื่อนไขของการทำงานในขณะนั้น ยกตัวอย่าง หากมีการเขียนข้อมูลจากอุปกรณ์มาสเตอร์ไปยังอุปกรณ์สเลฟ ตัวส่งคืออุปกรณ์มาสเตอร์ตัวรับคืออุปกรณ์สเลฟ ในทางตรงข้าม หากเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ ตัวส่งจะกลายเป็นอุปกรณ์สเลฟและตัวรับคืออุปกรณ์มาสเตอร์ ในระบบบัส 1 ระบบต้องมีอุปกรณ์มาสเตอร์เพียงตัวเดียวเท่านั้น

สายสัญญาณของระบบบัสนี้ต้องกำหนดสภาวะปกติไว้ที่ลอจิกสูง สามารถทำได้โดยต่อตัวต้านทานประมาณ 4.7 K พูลอัปไฟเลี้ยง +5V ดังนั้นอุปกรณ์ที่นำเข้ามาต่อบนระบบนี้จึงต้องออกแบบให้ภาคเอาต์พุตที่ต้องต่อกับสายสัญญาณมีลักษณะเป็นคอลเล็กเตอร์เปิดหรือเดรนเปิด ในรูปที่ 2.16 แสดงไดอะแกรมของการสื่อสารข้อมูลแบบอนุกรมหนึ่งสายเบื้องต้น



รูปที่ 2.16 การเชื่อมต่อบนระบบบัสหนึ่งสาย

คุณสมบัติไทม์สล็อต

อุปกรณ์มาสเตอร์จะเป็นอุปกรณ์เพียงตัวเดียวบนระบบบัสหนึ่งสายนี้ที่สามารถทำการอินนิเชียลสายสัญญาณได้ โดยอุปกรณ์มาสเตอร์จะกำเนิดจุดเริ่มต้นของไทม์สอล็ตด้วยการทำให้สายสัญญาณเป็นลอจิกต่ำในช่วงเวลาหนึ่ง จากนั้นจะทำให้กลับมาเป็นลอจิกสูง ถ้าหากอุปกรณ์สเลฟต้องการส่งข้อมูลมายังอุปกรณ์มาสเตอร์อุปกรณ์สเลฟจะเป็นตัวควบคุมสภาวะของสายสัญญาณต่อไป จนเสร็จสิ้นกระบวนการแต่ถ้าหากอุปกรณ์มาสเตอร์ต้องการจะส่งข้อมูลก็จะสามารถต่อไปได้เลย

ฟังก์ชั่นของ ไทม์สล็อตที่กำหนดโดยอุปกรณ์มาสเตอร์มีด้วยกัน 4 ฟังชั่นคือ ไทม์สล็อตของการรีเซต (RESET), การอ่านข้อมูล (READ DATA), การเขียนข้อมูล "1" (WRITE ONE), การเขียนข้อมูล "0" (WRITE ZERO) ไทม์สล็อตรีเซตใช้ในการเริ่มต้นติดต่อกับอุปกรณ์สเลฟในขณะที่ ไทม์สล็อตการอ่านจะใช้สำหรับอ่านข้อมูลที่จะส่งมาจากอุปกรณ์สเลฟ ส่วน ไทม์สล็อตการเขียนข้อมูล "1" และ "0" ใช้สำหรับการเขียนข้อมูลไปยังอุปกรณ์สเลฟผ่านสายสัญญาณของระบบ

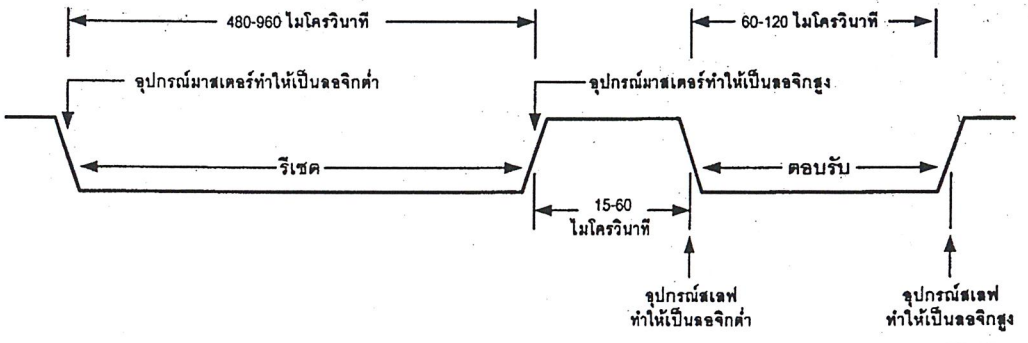
ทางด้านอุปกรณ์สเลฟมีฟังก์ชั่นของไทม์สล็อตอยู่ทั้งสิ้น 3 ฟังชั่นคือ ไทม์สล็อตของการตอบสนอง (PRESENCE), การเขียนข้อมูล "1" (WRITE ONE) และการเขียนข้อมูล "0" (WRITE ZERO) ไทม์สล็อตของการตอบสนองการตอบสนองใช้สำหรับตอบสนองการติดต่อจากอุปกรณ์มาสเตอร์ โดยอุปกรณ์สเลฟของตัวที่ถูกเลือกจากอุปกรณ์มาสเตอร์จะต้องส่งสัญญาณตอบสนองลงบนสายสัญญาณเพื่อแจ้งให้อุปกรณ์มาสเตอร์ทราบว่า ขณะนี้สามารถติดต่อกันได้แล้ว ส่วนไทม์สล็อตการเขียนข้อมูล "1" และ "0" ใช้สำหรับส่งข้อมูลไปยังมาสเตอร์ผ่านสายสัญญาณของระบบซึ่งจะสัมพันธ์กับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์

การแยกแยะฟังก์ชั่นของแต่ละไทม์สล็อตจะใช้ความยาวของคาบเวลาและลักษณะของรูปสัญญาณเป็นตัวกำหนด และทุกครั้งที่มีการเปลี่ยนแปลงฟังก์ชั่นต้องทำให้สายสัญญาณอยู่ในสภาวะว่างเสมอ ซึ่งก็คือการทำให้สายสัญญาณเป็นลอจิกสูงอย่างน้อยเป็นเวลา 1 ไมโครวินาที

ไทม์สล็อตการรีเซ็ตและการตอบสนอง

อุปกรณ์มาสเตอร์ทำให้เกิดการรีเซตบนสายสัญญาณเพื่อแจ้งแก่อุปกรณ์สเลฟ โดยการทำให้สายสัญญาณเป็นลอจิกต่อเนื่องอย่างน้อย 480 ไมโครวินาที และจะต้องทำให้สายสัญญาณกลับมาเป็นลอจิกสูงภายใน 480 ไมโครวินาทีหลังจากนั้น หากมีอุปกรณ์สเลฟต่ออยู่บนสายสัญญาณจะมีการตอบสนองสายสัญญาณรีเซตนั้นด้วยสัญญาณตอบสนอง (PRESENCE) โดยการทำให้สาย

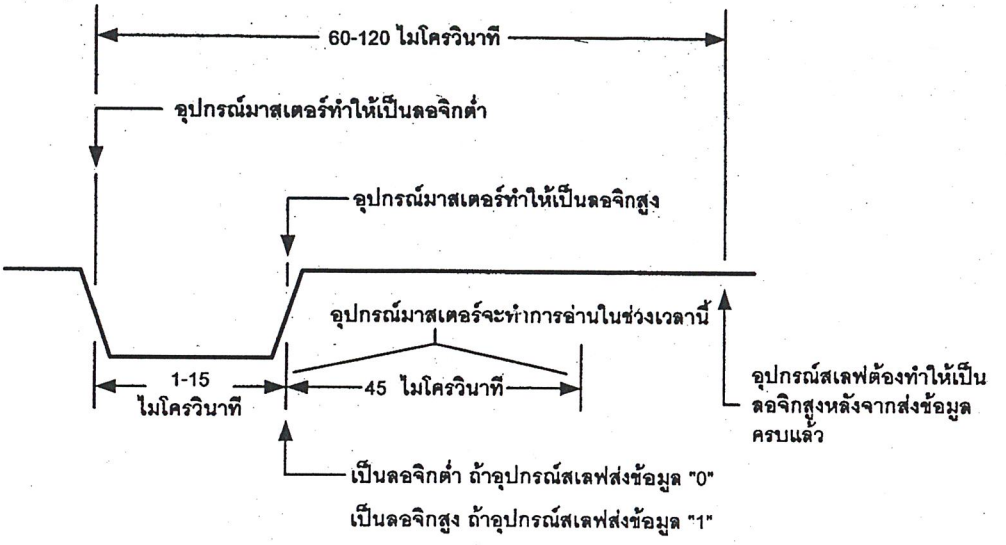
สัญญาณเป็นลอจิกต่ำต่อเนื่องนานประมาณ 60-240 ไมโครวินาที หลังจากสัญญาณรีเซตปรากฏประมาณ 15-60 ไมโครวินาที ในรูปที่ 2.17 แสดงไทม์สล็อตของการรีเซตและการตอบสนอง



รูปที่ 2.17 ไทม์สล็อตการรีเซตและการตอบรับของอุปกรณ์บนระบบบัสหนึ่งสาย

ไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์และการเขียนข้อมูลของอุปกรณ์สเลฟ

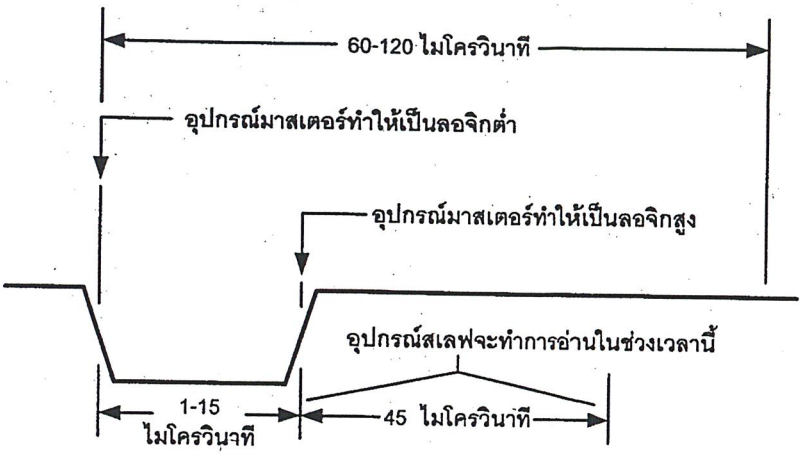
เมื่อต้องการอ่านข้อมูลจากอุปกรณ์สเลฟ อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 1-15 ไมโครวินาที จากนั้นต้องทำให้สถานะของสายกลับมาเป็นลอจิกสูง อุปกรณ์สเลฟจะส่งข้อมูลมาให้อุปกรณ์มาสเตอร์โดยถ้าข้อมูลเป็น “0” อุปกรณ์สเลฟจะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 45 ไมโครวินาที แล้วทำให้สายสัญญาณกลับมาสู่สถานะลอจิกสูงอีกครั้ง แต่ถ้าเป็นข้อมูล “1” อุปกรณ์สเลฟจะทำให้สายสัญญาณเป็นลอจิกสูงต่อเนื่องไปอีก 45 ไมโครวินาที รวมเวลาทั้งหมดในไทม์สล็อตนี้ประมาณ 60-120 ไมโครวินาที นั่นคือในไทม์สล็อตนี้ต้องใช้เวลารวมไม่เกิน 120 ไมโครวินาที ในขณะที่อุปกรณ์มาสเตอร์จะใช้เวลาในการอ่านอยู่ระหว่าง 15 และ 60 ไมโครวินาทีหลังจากเริ่มต้นไทม์สล็อตนี้ ในรูปที่ 2.18 แสดงรูปสัญญาณไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์ซึ่งจะมีลักษณะเหมือนกับการเขียนข้อมูลของอุปกรณ์สเลฟ และไทม์สล็อตทั้งสองจะเกิดขึ้นในช่วงเวลาเดียวกัน กล่าวคือ เมื่ออุปกรณ์มาสเตอร์อ่าน อุปกรณ์สเลฟก็ต้องทำการเขียนข้อมูล



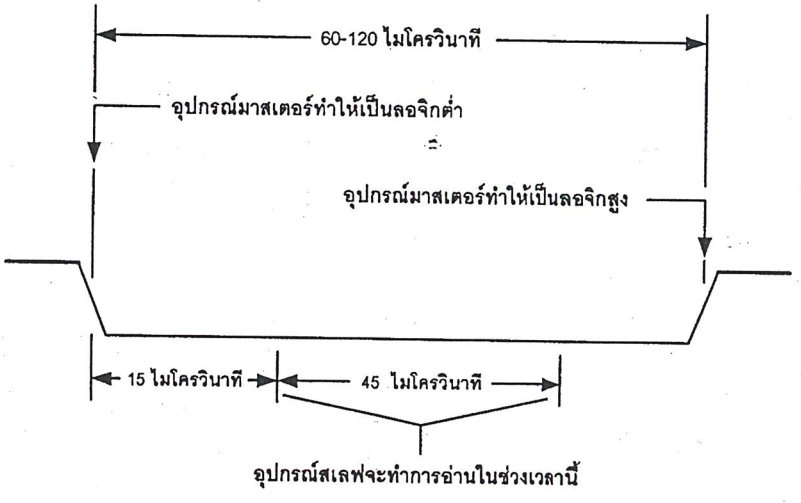
รูปที่ 2.18 ไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์ ซึ่งตรงกับไทม์สล็อตการเขียนข้อมูลของอุปกรณ์สเลฟ

ไทม์สล็อตการเขียนข้อมูลของอุปกรณ์มาสเตอร์

เมื่ออุปกรณ์มาสเตอร์ต้องการเขียนข้อมูล อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 1-15 ไมโครวินาที จากนั้นต้องทำให้สถานะของสายสัญญาณกลับมาเป็นลอจิกสูงแล้วดำเนินการเขียนข้อมูลได้ทันที ถ้าข้อมูลที่ต้องการเขียนไปยังอุปกรณ์สเลฟเป็น “0” อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 45 ไมโครวินาที แล้วทำให้สายสัญญาณกลับมาสู่สถานะสูงอีกครั้ง แต่ถ้าต้องการเขียนข้อมูล “1” อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกสูงต่อเนื่องไปอีก 45 ไมโครวินาที รวมเวลาทั้งหมดในไทม์สล็อตนี้ประมาณ 60-120 ไมโครวินาที ในรูปที่ 2.19 แสดงในรูปสัญญาณของไทม์สล็อตการเขียนข้อมูลของอุปกรณ์มาสเตอร์ซึ่งจะมีลักษณะเหมือนกับการอ่านข้อมูลของอุปกรณ์สเลฟ และไทม์สล็อตทั้งสองจะเกิดขึ้นในช่วงเวลาเดียวกันกล่าวคือ เมื่ออุปกรณ์มาสเตอร์เขียน อุปกรณ์สเลฟก็ต้องทำการอ่านข้อมูล



รูปที่ 2.19 ไทม์สล็อตการเขียนข้อมูล "1" ของอุปกรณ์มาสเตอร์ ซึ่งตรงกับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์สเลฟ



รูปที่ 2.20 ไทม์สล็อตการเขียนข้อมูล "0" ของอุปกรณ์มาสเตอร์

รูปแบบของการอ่านข้อมูลแบบหนึ่งสาย (1 – Wire^{TM} Communication protocol)

ในการติดต่อสื่อสารข้อมูลในระบบบัสหนึ่งสาย อุปกรณ์มาสเตอร์จะสามารถติดต่อกับอุปกรณ์สเลฟได้ครั้งละ 1 ตัวเท่านั้น ดังนั้นอุปกรณ์สเลฟแต่ละตัวจะต้องมีข้อมูลที่กำหนดแอดเดรสเฉพาะตัว โดยจะเก็บไว้ในหน่วยความจำรอมภายในอุปกรณ์สเลฟตัวนั้นๆ โดยปรกติอุปกรณ์สเลฟในระบบบัสหนึ่งสายของดัลลัสนี้จะมีหน่วยความจำขนาด 64 บิตหรือ 8 ไบต์ สำหรับการเก็บข้อมูลต่างๆที่สำคัญของอุปกรณ์แต่ละตัว ซึ่งประกอบด้วย

1. รหัสของตระกูล จำนวน 8 บิต
2. เลขหมายประจำตัว (serial number) จำนวน 48 บิต
3. รหัสตรวจสอบความผิดพลาด (CRC : Cyclical Redundancy Check) จำนวน 8 บิต

ผู้ใช้งานสามารถอ่านข้อมูลประจำตัวของอุปกรณ์สเลฟได้ด้วยการใช้คำสั่งอ่านหน่วยความจำรอม (ReadRom) ในกรณีที่บนสายสัญญาณมีอุปกรณ์สเลฟเพียงตัวเดียวไม่จำเป็นต้องอ้างแอดเดรสในการติดต่อ

รูปแบบการติดต่อบนระบบบัสหนึ่งสายจะเริ่มต้นขึ้นเมื่ออุปกรณ์มาสเตอร์ทำการรีเซตและกำหนดแอดเดรสของอุปกรณ์ที่ทำการติดต่อ ถ้าหากมีอุปกรณ์สเลฟ เพียงตัวเดียวสามารถข้ามขั้นตอนการติดต่อกับหน่วยความจำรอมในอุปกรณ์สเลฟได้ จะเรียกวิธีดังกล่าว การไม่ติดต่อหน่วยความจำรอม หรือ สคิปรอม (Skip ROM) จากนั้นรอการตอบรับของอุปกรณ์สเลฟ เมื่อการตอบรับสมบูรณ์ก็จะสามารถเริ่มขั้นตอนการอ่านหรือเขียนข้อมูลต่อไป

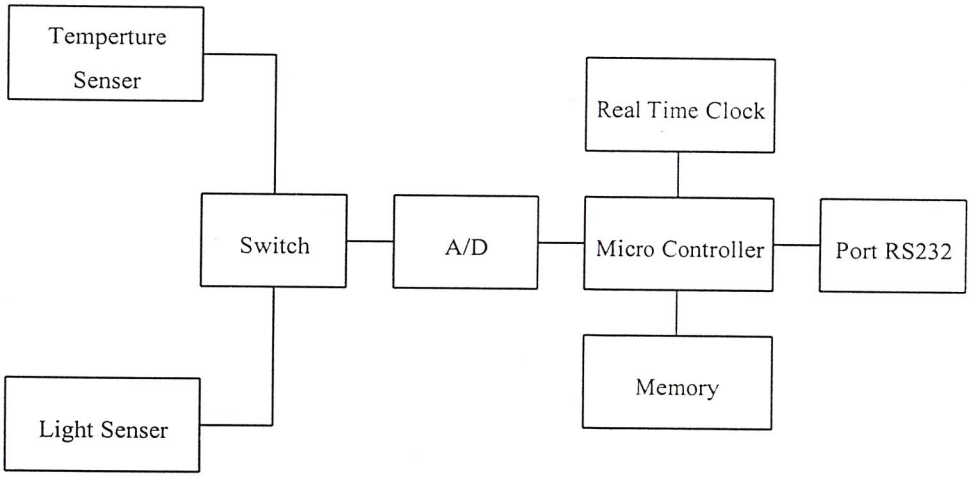
บทที่ 3

การออกแบบ

การออกแบบในส่วนของโครงงานนี้ เพื่อให้สามารถตอบสนองการทำงานได้ถูกต้องทั้งในการเก็บบันทึกค่า และในส่วนของการแสดงผล สามารถแบ่งส่วนของการออกแบบได้เป็น 2 ส่วน ด้วยกันคือ การออกแบบในส่วนของฮาร์ดแวร์ และการออกแบบในส่วนของซอฟแวร์

การออกแบบฮาร์ดแวร์

การออกแบบในส่วนนี้มีส่วนประกอบตามบล็อกไดอะแกรมที่ได้กล่าวถึงไว้แล้วในบทนำ คือ

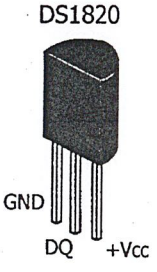


รูปที่ 3.1 แสดงบล็อกไดอะแกรม ส่วนประกอบทางฮาร์ดแวร์

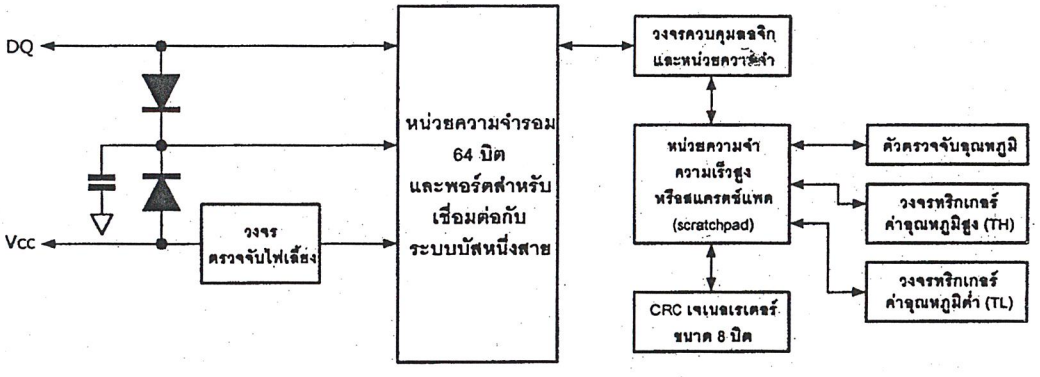
โดยจะอธิบายการออกแบบส่วนต่างๆ ตามบล็อกไดอะแกรม และแสดงวงจรสมบรูณ์ของส่วนฮาร์ดแวร์ไว้ในส่วนของภาคผนวก และในการออกแบบฮาร์ดแวร์นี้จะรวมส่วนของการออกแบบโปรแกรมภาษาแอสเซมบลีที่ใช้ควบคุมการทำงานของฮาร์ดแวร์ ไว้ในส่วนนี้ด้วย

1. ภาควัดอุณหภูมิ

ในภาควัดอุณหภูมิใช้ไอซี DS1820 ซึ่งเป็นไอซีตรวจจับอุณหภูมิที่ใช้ในการติดต่อแบบระบบบัสหนึ่งสาย ที่ขาต่อไปใช้งานเพียง 3 ขา คือ DQ ซึ่งเป็นขาเชื่อมต่อกับระบบบัส, ขาไฟเลี้ยงภายนอก และขากราวด์ ดังแสดงการจัดขาของไอซี DS1820 ในรูปที่ 3.2 และมีโครงสร้างการทำงานภายในดังแสดงในรูปที่ 3.3

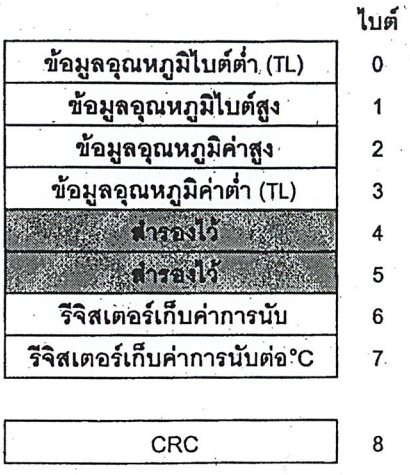


รูปที่ 3.2 การจัดขาของ DS1820



รูปที่ 3.3 โครงสร้างการทำงานภายในของไอซีตรวจจับอุณหภูมิ

หัวใจสำคัญของ DS1820 อยู่ที่ตัวตรวจจับอุณหภูมิและหน่วยความจำความเร็วสูงที่เรียกว่าสแครตช์แพด (scratchpad) ซึ่งมีขนาด 9 ไบต์ มีการจัดสรรหน่วยความจำในส่วนนี้แสดงในรูปที่ 3.4



รูปที่ 3.4 การจัดสรรพื้นที่สแครตช์แพด ใน DS1820

เมื่อวัดอุณหภูมิได้ก็จะนำค่าที่ได้นี้มาเก็บไว้ใน สแครตช์แพด ที่ไบต์ 0 และ 1 ทั้งนี้เนื่องจากไอซี DS1820 สามารถให้ข้อมูลของอุณหภูมิได้ละเอียดถึง 16 บิต เมื่อนำมาแปลงเป็นข้อมูลฐานสิบจึงสามารถแสดงความละเอียดของค่าอุณหภูมิได้ถึง 0.5 องศาเซลเซียสและ 0.9 องศาฟาเรนไฮต์โดยมีย่านวัดอุณหภูมิที่ –55 ถึง +125 องศาเซลเซียสหรือ -67 ถึง +257 องศาฟาเรนไฮต์ โดยค่าขององศาฟาเรนไฮต์ต้องใช้การแปลงหน่วยเข้ามาช่วย ใช้เวลาในการแปลงหน่วยอุณหภูมิเป็นดิจิตอลประมาณ 200 มิลลิวินาที สามารถกำหนดขอบเขตของอุณหภูมิที่ทำการวัดได้ และให้แจ้งเตือนเมื่อค่าอุณหภูมิสูงขึ้นหรือลดต่ำลงถึงค่าที่กำหนด โดยค่าของอุณหภูมิที่กำหนดจะเก็บไว้ที่ สแครตช์แพดในไบต์ 2 และ 3

คำสั่งเพื่อควบคุมการทำงานของ DS1820

ในการติดต่อกับไอซี DS1820 จะมีคำสั่งที่ต้องส่งให้แก่ DS1820 เพื่อกำหนดการทำงาน คำสั่งที่ใช้มากที่สุดมีด้วยกัน 3 คำสั่งดังนี้

1. **คำสั่งที่ไม่ติดต่อกับหน่วยความจำหรือสคิปรอม (Skip Rom)** เนื่องจากในการใช้งาน DS1820 โดยปกติแล้วจะมี DS1820 อยู่บนสายสัญญาณเพียงเส้นเดียว จึงไม่จำเป็นต้องใช้ข้อมูลกำหนดแอดเดรส ดังนั้นจึงไม่จำเป็นต้องติดต่อกับหน่วยความจำรอมเพื่ออ่านข้อมูล ข้อมูลของคำสั่งสคิปรอมที่ต้องส่งให้ DS1820 คือ 0CCH

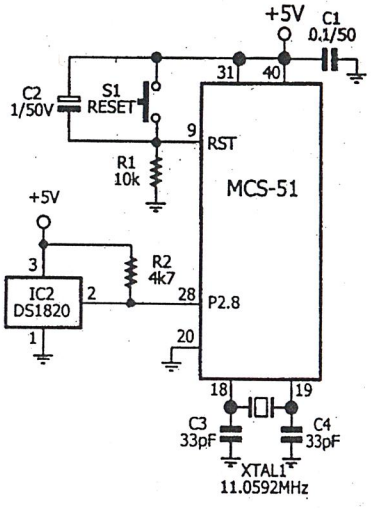
2. **คำสั่งแปลงอุณหภูมิ (Convert T)** มีค่าเท่ากับ 44H เมื่อส่งคำสั่งนี้ให้ DS1820 จะต้องทำการวนลูปอย่างน้อย 200 มิลลิวินาที เพื่อให้ DS1820 ได้ใช้เวลาในการแปลงค่าอุณหภูมิเป็นข้อมูลดิจิตอลมาเก็บไว้ในสแครตช์แพด
3. **คำสั่งอ่านข้อมูลจากสแครตช์แพด (Read Scratchpad)** มีค่าเท่ากับ 0BEH เมื่อส่งคำสั่งนี้ DS1820 จะทยอยส่งข้อมูลค่าของอุณหภูมิ ออกมาทั้งหมด 9 ไบต์ ดังมีรายละเอียดดังนี้

ขั้นตอนที่	การทำงานของอุปกรณ์มาสเตอร์	ข้อมูลหรือสภาวะ	รายละเอียด
1	ตัวส่ง	รีเซต	สร้างสัญญาณรีเซต
2	ตัวรับ	ตอบรับ	รอการตอบรับจาก DS1820
3	ตัวส่ง	0CCH	คำสั่ง Skip Rom
4	ตัวส่ง	44H	คำสั่งแปลงอุณหภูมิ(Convert T)
5	ตัวรับ	ข้อมูล 1 ไบต์	อ่านแฟลก Busy 8 ครั้ง
6	ตัวส่ง	รีเซต	สร้างสัญญาณรีเซต
7	ตัวรับ	ตอบรับ	รอการตอบรับจาก DS1820
8	ตัวส่ง	0CCH	คำสั่ง Skip Rom
9	ตัวส่ง	0BEH	คำสั่งการอ่านจากสแครตช์แพด
10	ตัวรับ	ข้อมูล 9 ไบต์	อ่านค่าอุภูมิจากสแครตช์แพด
11	ตัวส่ง	รีเซต	สร้างสัญญาณรีเซต
12	ตัวรับ	ตอบรับ	รอการตอบรับจาก DS1820
13	-	-	ทำการคำนวณค่าที่ได้จาก DS1820 เป็นเลขฐานสิบแล้วนำไปแสดงผลหรือใช้งานอื่นต่อไป

ตารางที่ 3.1 สรุปขั้นตอนการติดต่อกับ DS1820

การเชื่อมต่อกับไมโครคอนโทรลเลอร์ MCS-51

แสดงวงจรการเชื่อมต่อในรูปที่ 3.5 ใช้ขาพอร์ตเพียง 1 ขาเท่านั้นสำหรับการเชื่อมต่อกับ DS1820 โดยต้องมีตัวต้านทานค่า 4.7 K ต่อพูลอัปกับไฟเลี้ยง +5V จากนั้นจึงทำการเขียนโปรแกรมเพื่อติดต่อกัน โดยใช้รูปแบบการติดต่อตามมาตรฐานระบบบัสหนึ่งสายของดัลลัส



รูปที่ 3.5 การเชื่อมต่อ DS1820 กับไมโครคอนโทรลเลอร์ MCS-51

การเขียนโปรแกรมเพื่อติดต่อกับ DS1820

จากรายละเอียดของรูปแบบการสื่อสารในระบบบัสหนึ่งสายที่กล่าวตั้งแต่ต้น สามารถนำมาใช้เพื่อเป็นข้อมูลในการเขียนโปรแกรมติดต่อ โดยจะต้องเขียนโปรแกรมย่อยเพื่อสร้างไทม์สล็อตของฟังชั่นต่างๆ เริ่มจากโปรแกรมย่อยการรีเซต (RESET), โปรแกรมย่อยการรอการตอบรับจาก DS1820 (PRESENCE), โปรแกรมย่อยการอ่านเขียนข้อมูลกับ DS1820

อนึ่งในการติดต่อกับ DS1820 ในโครงงานนี้จะมี DS1820 เพียงตัวเดียวจึงไม่จำเป็นต้องใช้คำสั่งเพื่อติดต่อกับหน่วยความจำรอมภายใน DS1820 นั่นคือ จะติดต่อแบบสคิปรอม (Skip Rom) ดังมีรูปแบบการติดต่อสรุปให้เห็นได้อย่างชัดเจนในตารางที่ 3.1

ค่าอุณหภูมิ ที่ถูกทำการปรับตั้งไว้ใน DS1820 ในเทอมของ ½ องศาเซลเซียส LSB ซึ่งจะเป็นไปตามรูปแบบของข้อมูล 9 บิต

MSB	64	31	16	8	4	2	1	½
1	1	1	0	0	1	1	1	0

= -25 องศาเซลเซียส

รูปที่ 3.6 แสดงลักษณะข้อมูลของ DS1820

โดยในส่วนของโครงงานนี้ข้อมูลอุณหภูมิที่ได้จะเก็บเป็น ค่าองศาเซลเซียลในรูปแบบข้อมูลดิจิตอล 8 บิต เพื่อสะดวกในการจัดเก็บ และการนำมาแสดงผล

2. ภาควัดความเข้มแสง

ในภาควัดความเข้มแสง เป็นอีกส่วนที่สำคัญกับโครงงานนี้ โดยการอ่านค่าจะอ่านในรูปแบบของแรงดันทางไฟฟ้าโดยใช้คุณสมบัติทางไฟฟ้าของสารกึ่งตัวนำ เมื่อมีแสงมาตกกระทบรอยต่อจะทำให้คุณสมบัติทางไฟฟ้าเปลี่ยนแปลงไป แต่มีข้อจำกัดของอุปกรณ์สารกึ่งตัวนำชนิดนี้คือมีการทำงานไม่เป็นลิเนียร์ ซึ่งมีผลกระทบทำให้ค่าที่ได้มีค่าผิดพลาดขึ้น โดยในโครงงานนี้เลือกใช้โฟโต้ทรานซิสเตอร์ เบอร์ SPI – 0509CO ทำหน้าที่เป็นตัวเซนเซอร์ โดยมีคุณสมบัติทางไฟฟ้าคือ

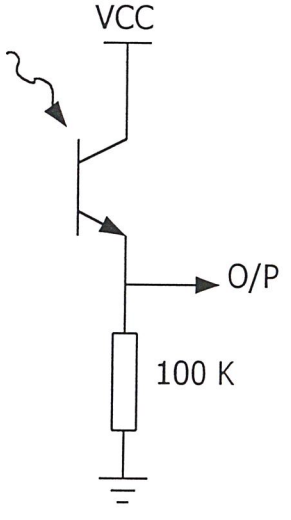
	SYMBOL	
	VCEO (V)	30
Absolute Max Rating	PC (mW)	10
	Topr (C)	-20~+80
	MIN (mA)	1.0
Light Current	TYP (mA)	30

	VCE (V)	3
	Ev (lux)	1000
Dark Current	MAX (uA)	0.1
	VCE (V)	10
$\Delta\varnothing$	TYP (deg)	+3
λ_p	TYP (nm)	880

ตารางที่ 3.2 แสดงคุณลักษณะทางไฟฟ้าของ SPI-0509CO

ลักษณะการต่อใช้งาน

การต่อใช้งานจะมีขาใช้งานเพียงสองขา คือขา คอลเลคเตอร์ และขาอีมิตเตอร์ โดยการจัดไบอัส จะกระทำโดยการที่มีแสงมาตกกระทบระหว่างรอยต่อของขาคอลเลคเตอร์ และขาอีมิตเตอร์ โดยเอาต์พุตที่ได้จะเป็นแรงดันที่ตกคร่อมโหลด เมื่อแสงที่ตกกระทบนั้นมีความสว่างมากเท่าใด ก็จะเปรียบเหมือนมีแรงดันไบอัสเพิ่มขึ้น ทำให้มีแรงดันตกคร่อมโหลดสูงขึ้นตามไปด้วย โดยมีลักษณะการต่อใช้งานเบื้องต้นดังรูปที่ 3.7

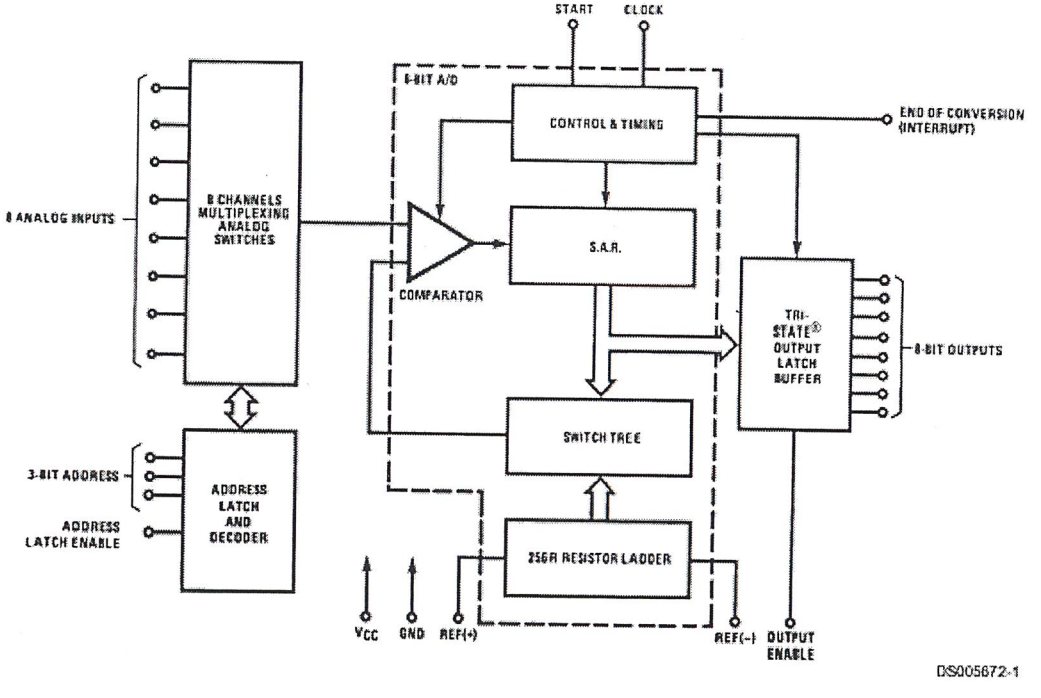


รูปที่ 3.7 แสดงการต่อใช้งานเบื้องต้น

ซึ่งสัญญาณที่ได้จากเอาต์พุตจะเป็นสัญญาณไฟฟ้าซึ่งเป็นสัญญาณอนาลอก ไม่สามารถนำมาติดต่อกับส่วนของฮาร์ดแวร์เพื่อทำการจัดเก็บไว้ในหน่วยความจำได้ ดังนั้นจะต้องทำการแปลงสัญญาณไฟฟ้าที่ได้ให้อยู่ในรูปดิจิตอลก่อน ซึ่งกระทำได้โดยวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลหรือที่เรียกว่าวงจร ADC (Analog to Digital Converter) ที่กำลังจะกล่าวถึงในหัวข้อต่อไป

3. ภาค ADC

ในภาคแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลนี้ จะเป็นส่วนที่ทำหน้าที่ในการแปลงข้อมูลทางไฟฟ้าที่ได้จากตัวเซนเซอร์มาเป็นสัญญาณดิจิตอลเพื่อให้สามารถนำข้อมูลที่ได้มาติดต่อกับส่วนของฮาร์ดแวร์ได้ โดยในการออกแบบของฮาร์ดแวร์ของโครงงานนี้เลือกใช้คอนเวอร์เตอร์ (Convertor) ขนาดของข้อมูลดิจิตอล 8 บิต ซึ่งจะมีจำนวนของข้อมูลที่ละเอียดถึง 256 ค่า และยังสามารถรองรับกับส่วนของคอนโทรลเลอร์ที่ใช้ในการออกแบบนี้ด้วย



รูปที่ 3.8 แสดงบล็อกไดอะแกรมภายในของ ADC0808

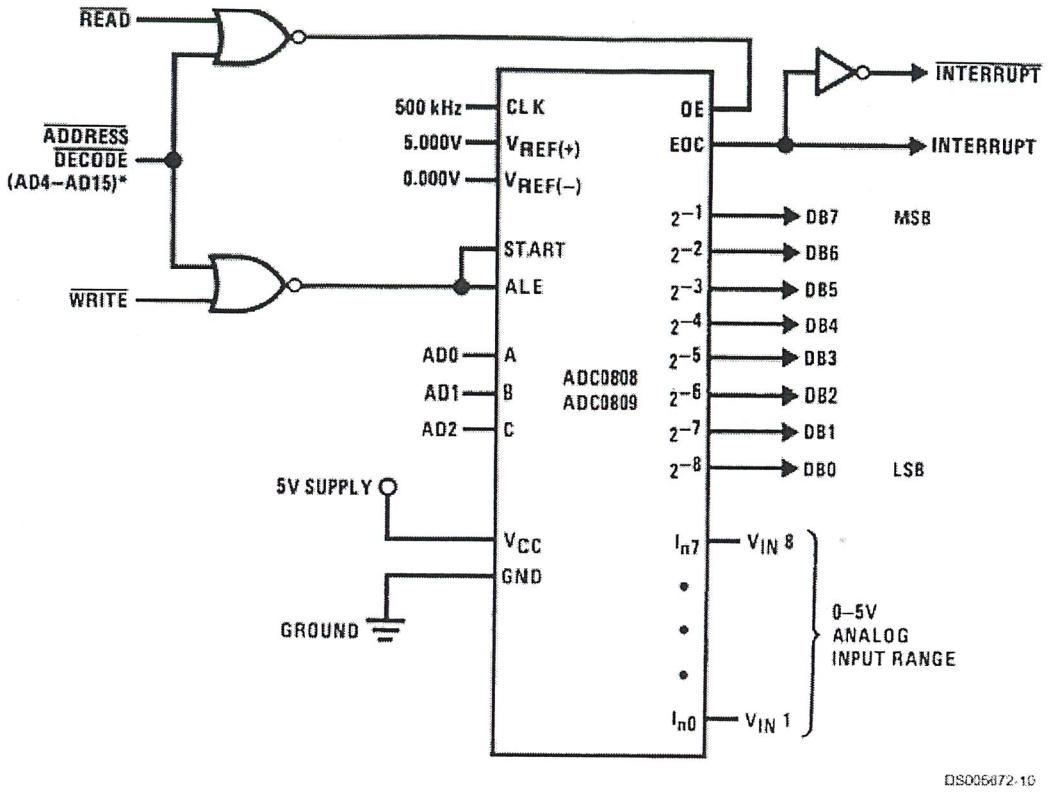
ในการออกแบบของโครงงานนี้เลือกใช้งานไอซีที่แปลงสัญญาณอนาลอกเป็นดิจิตอลเบอร์ ADC0808 ที่ออกแบบให้สามารถใช้งานร่วมกับตัวไมโครคอนโทรลเลอร์ได้โดยตรง และภายในตัวไอซียังออกแบบให้มีส่วนของวงจรมัลติเพล็กซ์สวิตซ์ (Multiplex Switch) ขนาด 8 ช่องทางไว้ให้สามารถใช้งานในกรณีที่มีอินพุตหลายตัวได้ โดยจะมีส่วนของวงจรแลทซ์ (Latch) ทำหน้าที่ในการแลทซ์ค่าตำแหน่งแอดเดรสของสวิทช์ไว้ โดยข้อมูลที่ทำการแปลงแล้วก็จะถูกแลทซ์ไว้ด้วย ซึ่งข้อมูลที่ได้จะถูกเก็บไว้ในส่วนของบัฟเฟอร์ด้านเอาต์พุต โดยจะสามารถอ่านข้อมูลได้เมื่อทำการส่งสัญญาณไปที่ขาอีนาเบิ้ลของตัวคอนเวอร์เตอร์ โดยมีบล็อกไดอะแกรมภายในดังรูปที่ 3.8 และแสดงส่วนของตารางเลือก สัญญาณอนาลอกดังตารางที่ 3.3

SELECTED ANALOG CANNEL	ADDRESS LINE		
	C	B	A
IN0	L	L	L
IN1	L	L	H
IN2	L	H	L
IN3	L	H	H
IN4	H	L	L
IN5	H	L	H
IN6	H	H	L
IN7	H	H	H

ตารางที่ 3.3 แสดงสถานะการเลือกสัญญาณอนาลอกของ ADC0808

ลักษณะการต่อใช้งาน

การต่อใช้งานตัวไอซี ADC0808 กับไอซีคอนโทรลเลอร์ จะทำได้โดยการส่งสัญญาณการอ่าน และเขียนข้อมูลจากคอนโทรลเลอร์ไปยังไอซีคอนเวอร์เตอร์ พร้อมทั้งจะต้องส่งสัญญาณดีโคดไอซีตามไปด้วย โดยต่อรวมกับสัญญาณ การอ่าน และเขียน ซึ่งทำได้โดยต่อผ่านไอซีแนนเกท เพื่อกลับสัญญาณให้เป็น ลอจิกสูงก่อนนำไปเข้ายังขาควมคุมของไอซีคอนเวอร์เตอร์ สามารถแสดงการต่อใช้งานได้ดังรูปที่ 3.9



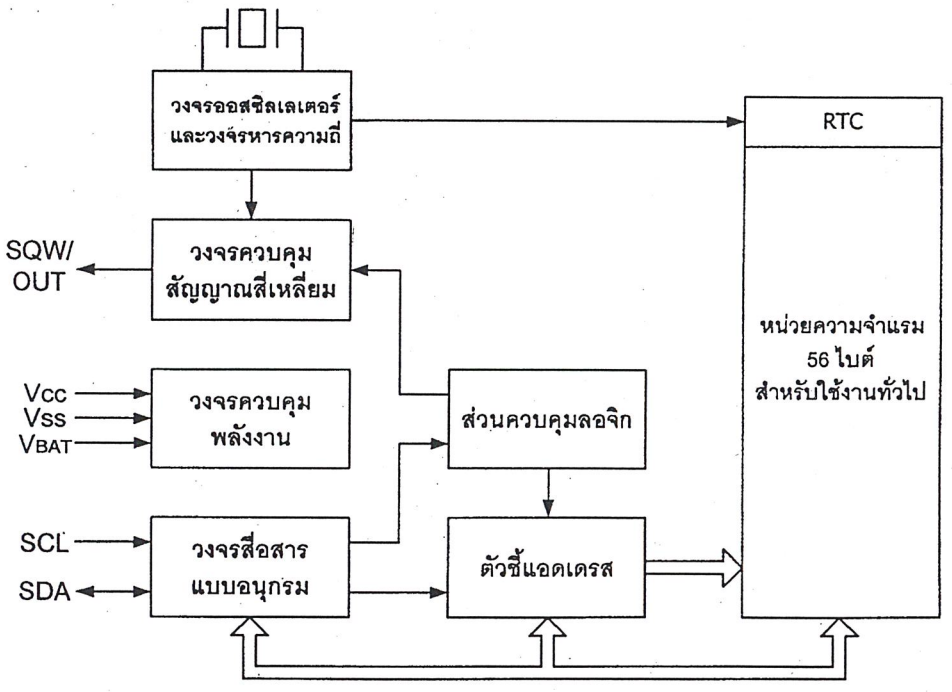
รูปที่ 3.9 แสดงการต่อใช้งาน ADC0808

4. ภาคฐานเวลาเรียลไทม์

ในภาคฐานเวลาเรียลไทม์ เป็นส่วนที่สร้างข้อมูลเวลาให้กับฮาร์ดแวร์ เพื่อใช้อ้างอิงในการเก็บข้อมูล ซึ่งในภาคนี้จะใช้ ไอซี DS1307 ไอซีสร้างฐานเวลาเรียลไทม์ โดย ไอซี DS1307 จัดการเชื่อมต่อในรูปแบบบัส I^2C โดยการทำงานเป็นอุปกรณ์สเลฟเสมอ ดังนั้นการติดต่อเพื่อใช้งานจึงต้องกำหนดรูปแบบตามที่กำหนดไว้ในรูปแบบการติดต่อแบบ I^2C ในรูปที่ 3.10 แสดงส่วนประกอบหลักที่สำคัญและไดอะแกรมการทำงานของ DS1307 วงจรออสซิลเลเตอร์ถือเป็นหัวใจหลักของไอซี เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลเวลาจริง ในขณะที่ DS1307 ทำงาน ที่ขา SQW/OUT จะมีสัญญาณพัลส์สี่เหลี่ยมส่งออกมมาตลอดเวลาในกรณีที่มีการอีนาเบิลวงจรกำเนิดสัญญาณพัลส์ที่รีจิสเตอร์ควบคุม ค่าความถี่ของสัญญาณนี้สามารถเลือกได้ 4 ค่าคือ 1Hz , 4.096 Hz ,8.192kHz และ 32kHz พร้อมกันนั้นก็จะมีการเก็บค่าของเวลาในหน่วยความจำนอนโวลาไทม์แรม

ซึ่งมีขนาดรวม 64 ไบต์ แต่จัดสรรให้เก็บข้อมูลเวลา 8 ไบต์และเป็นความจำสำหรับเก็บข้อมูลทั่วไปสำหรับผู้ใช้อีก 56 ไบต์

วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสภาวะของไฟเลี้ยงไอซี หากไฟเลี้ยงต่ำกว่า 1.25xVBAT ก็จะควบคุมให้ DS1307 หยุดการทำงานรีเซตค่าตัวนับแอดเดรสภายในทำให้ไม่สามารถติดต่อกับ DS1307 ได้ ดังนั้นในการใช้งาน DS1307 ต้องระมัดระวังอย่าให้ไฟเลี้ยงตกต่ำกว่า 1.25xVBAT หรือประมาณ 3.75 V ในกรณีที่ใช้ VBAT เท่ากับ 3V หากไฟเลี้ยงมีค่าต่ำกว่า VBAT ไอซี DS1307 จะเข้าสู่โหมดสำรองข้อมูลกระแสต่ำทันที จะไม่มีการส่งสัญญาณพัลซ์ออกมาที่ขา SQW/OUT แต่วงจรสร้างฐานเวลายังคงทำงานเพื่อให้ค่าของเวลาเดินไปอย่างไม่ผิดพลาด เมื่อมีไฟเลี้ยงปรากฏขึ้นอีกครั้ง DS1307 ก็จะสามารถให้ค่าของเวลาที่เป็นจริงแก่ผู้ใช้งานได้ต่อไป



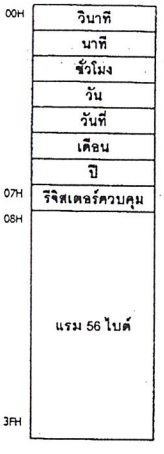
รูปที่ 3.10 โครงสร้างภายในของไอซีเรียลไทม์คล็อกเบอร์ DS1307

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

การจัดสรรหน่วยความจำภายใน DS1307

ในรูปที่ 3.11 (ก) แสดงการจัดสรรพื้นที่ของหน่วยความจำภายใน DS1307 พื้นที่ 7 ไบต์แรกตั้งแต่ แอดเดรส 00H-06H เป็นพื้นที่ของรีจิสเตอร์ค่าเวลาใช้ในการเก็บค่าข้อมูลเกี่ยวกับเวลา ไบต์ต่อมาที่แอดเดรส 07H เป็นพื้นที่ของรีจิสเตอร์ควบคุมการทำงานของ DS1307 ในรูปที่ 3.11 (ข) แสดงรายละเอียดของรีจิสเตอร์ค่าเวลาและรีจิสเตอร์ควบคุมของ DS1307

ด้วยการจัดสรรพื้นที่แบบนี้ ทำให้ผู้ใช้งานสามารถเรียกข้อมูลเวลาออกมาได้ตามต้องการ โดยไม่จำเป็นต้องอ่านออกมาทั้งหมดก็ได้ ค่าของเวลาทั้งหมดจะอยู่ในรูปของเลขฐานสิบ สำหรับการแสดงค่าเวลาในรูปของชั่วโมง สามารถเลือกได้ว่าต้องการแบบ 12 หรือ 24 ชั่วโมงโดยกำหนดที่บิตที่ 6 ของแอดเดรส 02H และเมื่อเลือกแบบ 12 ชั่วโมง ที่บิตที่ 5 ในแอดเดรสเดียวกันจะใช้ในการแสดงค่า AM/PM โดยถ้าบิตนี้เป็น "1" หมายถึง ค่าช่วงเวลาในขณะนี้เป็นช่วงเวลาเที่ยงวัน ในกรณีที่เป็นแบบ 24 ชั่วโมง บิตนี้จะใช้ในการแสดงค่า 2 ของหลักสิบในหน่วยชั่วโมง



(ก)

	บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0	ค่าของข้อมูล
	CH	ข้อมูลวินาที (หลักสิบ)			ข้อมูลวินาที (หลักหน่วย)				00-59
	x	ข้อมูลนาที (หลักสิบ)			ข้อมูลนาที (หลักหน่วย)				00-59
	x	12 ชั่วโมง / 24 ชั่วโมง	ชั่วโมง (หลักสิบ) / AM/PM	ข้อมูลชั่วโมง (หลักสิบ)	ข้อมูลชั่วโมง (หลักหน่วย)				01-12 / 00-23
	x	x	x	x	x	ข้อมูลวันในสัปดาห์			1-7
	x	x	ข้อมูลวันที่ (หลักสิบ)		ข้อมูลวันที่ (หลักหน่วย)				01-28/29 / 01-30 / 01-31
	x	x	x	ข้อมูลเดือน (หลักสิบ)	ข้อมูลเดือน (หลักหน่วย)				01-12
	ข้อมูลปี (หลักสิบ)				ข้อมูลปี (หลักหน่วย)				00-99
	OUT	x	x	SQWE	x	x	RS1	RS0	

(ข)

รูปที่ 3.11 (ก) การจัดสรรหน่วยความจำแรมภายใน DS1307

(ข) รายละเอียดของค่ารีจิสเตอร์เก็บค่าเวลาและรีจิสเตอร์ควบคุมของ DS1307

รีจิสเตอร์ควบคุม

มีแอดเดรสควบคุมอยู่ที่ 07H มีรายละเอียดของแต่ละบิตดังนี้

OUT (Outputcontrol) : ใช้ในการควบคุมระดับลอจิกที่ขา SQW/OUT ในกรณีที่ดิสเอเบิลการกำเนิดสัญญาณสี่เหลี่ยม โดยถ้าบิตนี้เป็น "1" ที่ขา SQW/OUT ก็จะเป็น "1" ถ้าบิตนี้เป็น "0" ที่ขา SQW/OUT ก็จะเป็น "0" ด้วย

SQWE (Square Wave Enable) : ใช้ในกรณีอีนาเบิลวงจรกำเนิดสัญญาณสี่เหลี่ยมที่ขา SQW/OUT ถ้าต้องการให้มีสัญญาณสี่เหลี่ยมออกมาให้กำหนดบิตนี้เป็น "1"

RS1, RS0 (Rate Select) : ใช้ในการเลือกความถี่ของสัญญาณสี่เหลี่ยมที่ขา SQW / OUT ดังมีรายละเอียดดังต่อไปนี้

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

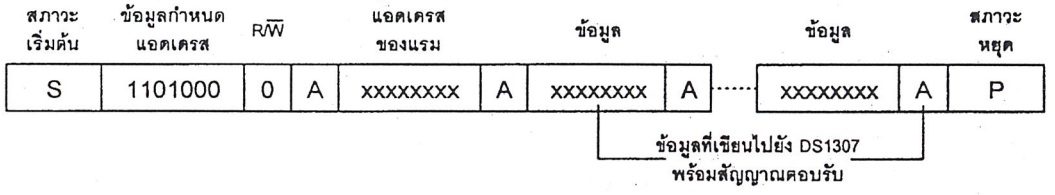
ตารางที่ 3.4 แสดงรายละเอียดการเลือก Rate Select

โหมดการทำงานของ DS1307

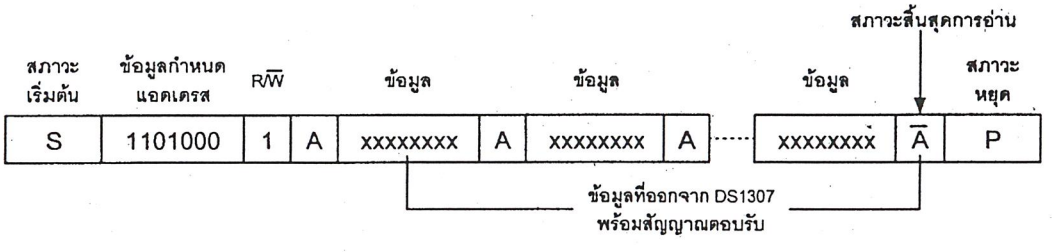
มีด้วยกัน 2 โหมดคือ โหมดเขียนข้อมูลและโหมดอ่านข้อมูล ในการใช้งาน DS1307 ตามปกติจะใช้งานเฉพาะการอ่านข้อมูลเท่านั้น เนื่องจากไมโครคอนโทรลเลอร์จะติดต่อกับ DS1307 เพื่ออ่านข้อมูลของเวลาไปใช้งาน ในโหมดการเขียนข้อมูลจะถูกใช้งานก็ต่อเมื่อต้องการตั้งค่าเวลาใหม่และต้องการเขียนข้อมูลลงในหน่วยความจำใช้งานทั่วไป อย่างไรก็ตามเมื่อเริ่มต้นการติดต่อกับ DS1307จำเป็นอย่างยิ่งที่จะต้องเข้าสู่โหมดการเขียนข้อมูลก่อนเพื่อกำหนดแอดเดรสที่ต้องการอ่านข้อมูล จากนั้นจึงเปลี่ยนโหมดการทำงานมาเป็นโหมดการอ่านข้อมูลต่อไป

โหมดการเขียนข้อมูล มีรูปแบบดังรูปที่ 3.12 เริ่มต้นเมื่อไมโครคอนโทรลเลอร์ ทำการกำหนดสภาวะเริ่มต้น (START :S) จากนั้นส่งข้อมูลกำหนดแอดเดรส 1101000 ตามด้วยข้อมูลเลือกการเขียน นั่นคือค่า 0 จากนั้นจะรอการตอบรับจาก DS1307 เมื่อมีการตอบรับมาเรียบร้อย ก็เริ่มเขียนข้อมูลลงไปครั้งละแอดเดรสหลังจากการเขียนข้อมูลลงไปแต่ละแอดเดรส จะต้องหยุดการตอบรับจาก DS1307 ทุกครั้งจึงจะสามารถเขียนข้อมูลต่อไปได้ เมื่อเขียนข้อมูลเรียบร้อยแล้วให้ส่งสภาวะการหยุด (STOP : P) เป็นอันสิ้นสุดกระบวนการเขียนข้อมูล

โหมดการอ่านข้อมูล มีรูปแบบดังรูปที่ 3.13 เริ่มต้นการทำงานเหมือนกับโหมดการเขียนข้อมูลคือ ไมโครคอนโทรลเลอร์กำหนดสภาวะเริ่มต้นแล้วทำการส่งข้อมูลกำหนดแอดเดรสตามด้วยข้อมูลเลือกการอ่านซึ่งเท่ากับ 1 จากนั้นรอการตอบรับจาก DS1307 เมื่อตอบรับเรียบร้อยแล้ว DS1307 จะทยอยส่งข้อมูลออกมาให้ไมโครคอนโทรลเลอร์คราวละ 1 แอดเดรสหรือ 1 ไบต์ โดยแอดเดรสที่เลือกอ่านข้อมูลจะต้องมีการกำหนดมาล่วงหน้าด้วยโหมดการเขียนข้อมูล วิธีง่ายๆ คือเข้าสู่โหมดการเขียนข้อมูลก่อน เมื่อถึงจังหวะที่ต้องการเขียนข้อมูล ให้ทำการสร้างสภาวะเริ่มต้นและส่งข้อมูลกำหนดแอดเดรสใหม่อีกครั้ง ตามด้วยเลือกโหมดการอ่านข้อมูล ข้อมูลที่ออกมาจาก DS1307 ก็จะเป็นแอดเดรสที่กำหนดไว้ก่อนหน้านี้



รูปที่ 3.12 รูปแบบของข้อมูลสำหรับติดต่อกับ DS1307 ในโหมดการเขียนข้อมูล

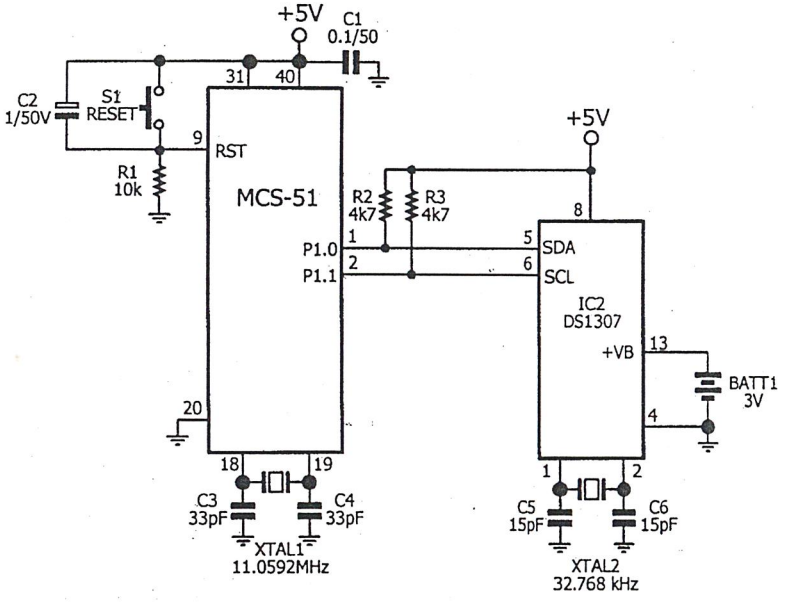


รูปที่ 3.13 รูปแบบข้อมูลเพื่อใช้ในการติดต่อ DS1307 ในโหมดการอ่านข้อมูล

การเชื่อมต่อ DS1307 กับไมโครคอนโทรลเลอร์ MSC-51

ดังแสดงในรูปที่ 3.14 จะเห็นได้ว่ามีลักษณะการต่อไม่เหมือนกับอุปกรณ์บนระบบบัส I^2C ตัวอื่นๆ ทุกประการ และสามารถที่จะต่อไอซีทั้งหมดรวมกันบนสาย SDA และ SCL ได้ เป็นการย้ำให้เห็นถึงความสามารถพิเศษของระบบบัส I^2C ที่ผู้ใช้งานสามารถเชื่อมต่ออุปกรณ์ระบบบัส I^2C ได้ถึง 3 ตัว 3 ลักษณะการทำงาน โดยใช้สัญญาณเพียง 2 เส้น

จากวงจรในรูปที่ 3.14 DS1307 จำเป็นต้องต่อแบตเตอรี่ไว้ตลอดเวลาไม่ว่าจะใช่งานหรือไม่ ทั้งนี้เพื่อรักษาการทำงานของวงจรภายใน DS1307 ให้ยังคงทำงานต่อเนื่องไป เมื่อใดที่ไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูล ก็จะได้ข้อมูลเวลาที่เป็นจริงตลอดเวลา



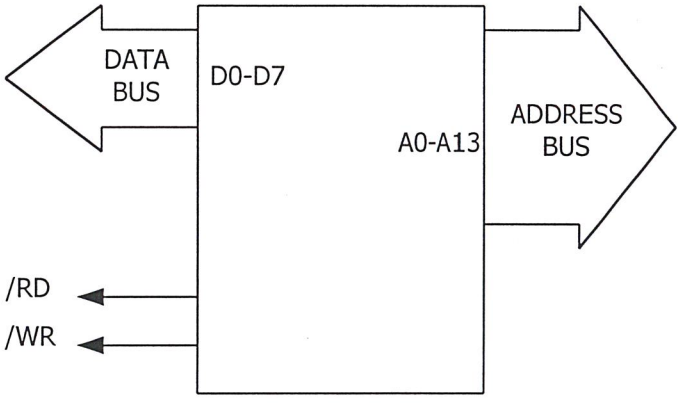
รูปที่ 3.14 แสดงการเชื่อมต่อไมโครคอนโทรลเลอร์ MCS-51 กับไอซีเรียลไทม์คล็อก DS1307

5. ภาคหน่วยความจำ

ภาคหน่วยความจำของส่วนฮาร์ดแวร์จะเป็นส่วนที่ทำหน้าที่ในการเก็บบันทึกข้อมูล ที่ได้จากตัวเซนเซอร์ โดยในการออกแบบฮาร์ดแวร์ส่วนนี้จะใช้หน่วยความจำแรม (Random Access Memory : RAM) เพื่อทำหน้าที่ในการจัดเก็บข้อมูล ซึ่งจากคุณสมบัติของแรมที่สามารถ เขียนข้อมูลซ้ำได้ ทำให้สามารถนำมาใช้งานในการเก็บบันทึกค่าซ้ำใหม่ได้ตลอดเวลา โดยข้อมูลที่เก็บไว้จะคงอยู่เสมอเมื่อยังมีไฟเลี้ยงจ่ายให้กับตัวแรม โดยในการออกแบบเลือกใช้ไอซีเบอร์ 6264 ซึ่งเป็นไอซีหน่วยความจำ มีบัส (Bus) ข้อมูลขนาด 8 บิต และความจุขนาด 64 กิโลบิต (8 ไบต์) เพื่อรองรับการเก็บข้อมูลของตัวโครงงานนี้ และได้ออกแบบเพิ่มเติมให้สามารถต่อแรมเพิ่มเป็นขนาด 256 กิโลบิต ได้

ลักษณะการต่อใช้งาน

การต่อใช้งานแรม จะทำการต่อดาต้าบัสของแรมกับด้าต้าบัสของตัวคอนโทรลเลอร์เพื่อให้สามารถทำการติดต่อข้อมูลถึงกันได้ โดยจะมีแอดเดรสบัสทำหน้าที่ระบุตำแหน่งของข้อมูลที่ได้ทำการจัดเก็บไว้ โดยจะใช้แอดเดรสบัส A0-A13 ในการระบุแอดเดรสของแรมขนาด 8 กิโลไบต์ โดยแอดเดรส ไบต์ต่ำ จะได้จากการแลทซ์แอดเดรสจากดาต้าบัส ส่วนขาสัญญาณการอ่านและเขียนแรมจะต่อมาจากคอนโทรลเลอร์เพื่อใช้ในการอ่านและเขียนข้อมูลของแรม



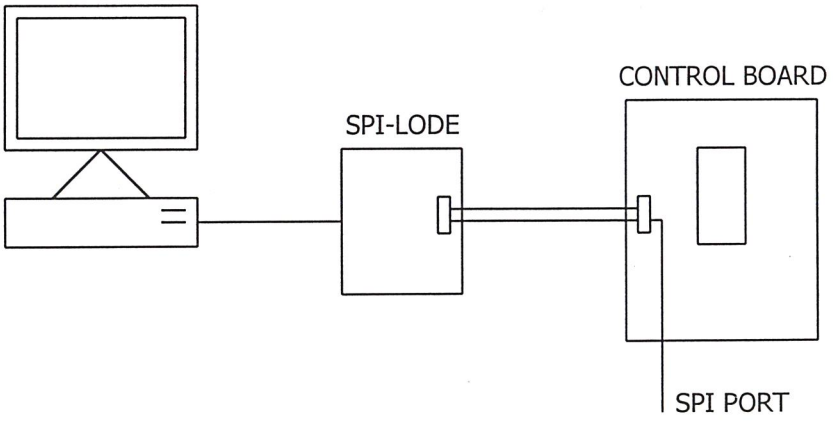
รูปที่ 3.15 แสดงลักษณะการต่อใช้งานแรมขนาด 8 กิโลไบต์

6. ภาคคอนโทรลเลอร์

ในการออกแบบนี้ส่วนของคอนโทรลเลอร์จะเป็นส่วนที่สำคัญมาก เพราะส่วนนี้ต้องทำหน้าที่ควบคุมการทำงานทั้งหมดของส่วนฮาร์ดแวร์ โดยการออกแบบของโครงงานนี้เลือกใช้ไอซีไมโครคอนโทรลเลอร์ MCS-51 เบอร์ AT89S8252 มาใช้งานในส่วนนี้ ซึ่งไอซีคอนโทรลเลอร์ตัวนี้มีคุณสมบัติเบื้องต้นคือ แฟลซเมมโมรี (Flash Memory) ขนาด 8 กิโลไบต์ และอีอีพรอม ขนาด 2 กิโลไบต์ และมีคุณสมบัติที่สำคัญเกี่ยวกับ SPI ซึ่งทำให้ชิปตัวนั้มีความสามารถเขียนโปรแกรมข้อมูลภายในตัวชิปโดย ผ่าน SPI-LOAD ทำให้มีความสะดวกอย่างมากในการที่จะทำการพัฒนาโปรแกรม โดยที่แฟลซเมมโมรี สามารถที่เขียนซ้ำได้ 1000 ครั้ง และส่วนอีอีพรอม เขียนซ้ำได้ 100,000 ครั้ง ซึ่งส่วนของโปรแกรมหลักทั้งหมดจะเขียนลงในส่วนของแฟลซเมมโมรีนี้ โดยผ่านส่วน SPI โดยในการออกแบบจะออกแบบให้มีส่วนของ SPI พอร์ตอยู่ในโครงงานทำให้สามารถโหลดโปรแกรมลงในตัวไอซีคอนโทรลเลอร์ ได้บนบอร์ดฮาร์ดแวร์ และเมื่อโหลดโปรแกรมเรียบร้อยแล้วสามารถนำพอร์ตที่เกี่ยวข้องกับส่วน SPI ไปใช้งานด้านอื่นได้ด้วย

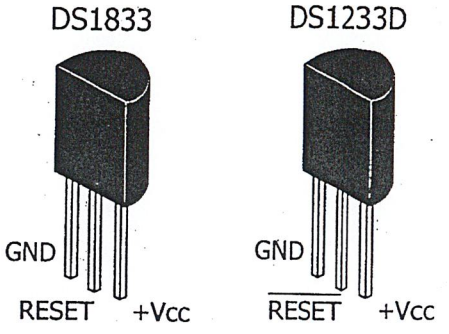
แสดงการต่อใช้งานร่วมกับ SPI LOAD

โดยบอร์ด SPI LOAD เป็นผลิตภัณฑ์ ของบริษัทศิลา-รีเสิรซ์ ที่ใช้ในการพัฒนาโปรแกรมของไอซีคอนโทรลเลอร์ของบริษัท ATMEL ในชิปตระกูล AT89Sxx ซึ่งมีแฟลซเมมโมรีอยู่ภายในโดยแสดงการต่อใช้งานดังรูปที่ 3.16



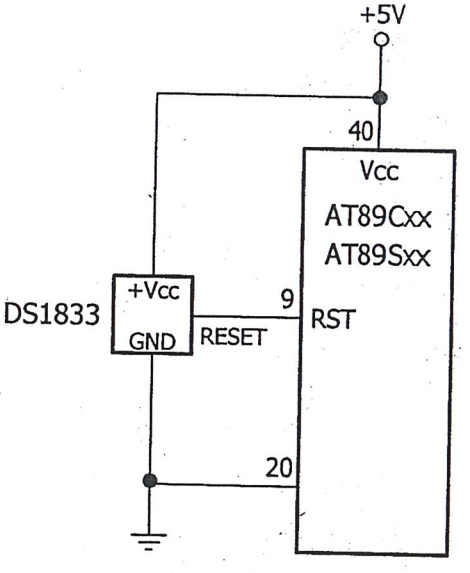
รูปที่ 3.16 แสดงการต่อใช้งานร่วมกับ SPI LODE

โดยในการติดต่อ ใช้งานกับบอร์ด SPI-LODE สัญญาณรีเซตของบอร์ดทั้งสองจะต้องสัมพันธ์กัน ซึ่งทำได้โดยการใช้ วงจรรีเซตแบบ เพาเวอร์ออน โดยใช้ไอซี DS1833 ซึ่งเป็นไอซีสร้างสัญญาณรีเซตแบบเพาเวอร์ออน มาควบคุม โดยมีลักษณะเป็นไอซี 3 ขา ดังรูป



รูปที่ 3.17 แสดงลักษณะภายนอกของไอซี DS1833

แสดงการต่อใช้งานไอซี DS1833



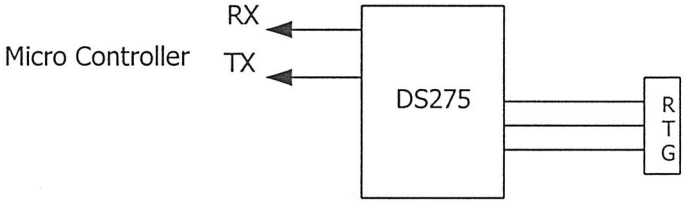
รูปที่ 3.18 แสดงลักษณะการต่อใช้งานไอซี DS1833

7. ภาคการเชื่อมต่อกับคอมพิวเตอร์

ในการออกแบบโครงงานนี้ส่วนการเชื่อมต่อกับคอมพิวเตอร์ เพื่อส่งข้อมูลจากฮาร์ดแวร์ให้กับคอมพิวเตอร์นั้น จะกระทำการส่งข้อมูลแบบอนุกรม โดยผ่านพอร์ตอนุกรม RS-232 โดยในตัวคอนโทรลเลอร์จะมีส่วนที่ใช้ในการติดต่อสื่อสารแบบอนุกรมอยู่แล้ว แต่ในการติดต่อกับคอมพิวเตอร์จะต้องทำการปรับระดับแรงดันของพอร์ตก่อนเพื่อให้สามารถติดต่อสื่อสารได้ โดยในการออกแบบนี้จะใช้ไอซี DS275 ซึ่งไอซีตัวนี้ออกแบบมาเพื่อเป็นตัวปรับระดับสัญญาณของพอร์ตสื่อสารอนุกรม ระหว่างตัวไมโครคอลโทรลเลอร์และคอมพิวเตอร์โดยเฉพาะ

รูปแสดงการต่อใช้งานไอซี DS275

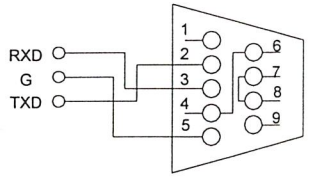
การต่อใช้งาน จะต่อขาสัญญาณ RX และ TX จากคอนโทรลเลอร์ซึ่งเป็นขาต่อใช้งานสำหรับการติดต่อสื่อสารอนุกรม ไปยังไอซี DS275 ซึ่งจะทำหน้าที่ในการปรับระดับสัญญาณของข้อมูล ในการติต่อกับคอมพิวเตอร์ แสดงการต่อใช้งานดังรูปที่ 3.19



รูปที่ 3.19 แสดงการต่อใช้งาน DS275

สายสัญญาณอินเตอร์เฟส

การเชื่อมต่อกับ คอมพิวเตอร์ ผ่านพอร์ตสื่อสารอนุกรม จะใช้ คอนเนคเตอร์ DB-9 โดยมีลักษณะการเชื่อมต่อดังรูป



รูปที่ 3.20 แสดงลักษณะของคอนเนคเตอร์ DB-9

8. การเขียนโปรแกรมควบคุม

การออกแบบโปรแกรมควบคุมฮาร์ดแวร์จะเขียนด้วยโปรแกรมภาษาแอสเซมบลี ของไมโครคอนโทรลเลอร์ MCS-51 และทำการแอสเซมเบอร์โดยโปรแกรม แอสเซมเบลอร์ SXA51 โดยการเขียนโปรแกรมควบคุมจะใช้โปรแกรมอีดิตเตอร์ (Editor) เป็นตัวเขียนโดยบันทึกแฟ้มด้วยนามสกุล ASM (File.asm) แล้วนำมาแอสเซมเบลอร์ด้วยโปรแกรม SXA51 เพื่อได้แฟ้มนามสกุล HEX (File.hex) แล้วนำมาโหลดลงตัวคอนโทรลเลอร์โดยผ่านตัว SPI LOAD ซึ่งรายละเอียดของโปรแกรมควบคุมแสดงดังอัลกอริทึม(Algorithm)ดังนี้

Algorithm

- เก็บค่าชั่วโมง,นาที ลงแรม
- เก็บ วัน,เดือน,ปี
- เปรียบเทียบวัน
    - ไม่เท่ากัน ใส่ตัวสิ้นสุดข้อมูล แล้วกลับไปเก็บ วัน,เดือน,ปี
    - เท่ากัน เปรียบเทียบเวลาทุก 5 นาที
        - เก็บค่า อุณหภูมิและแสง
        - หน่วงเวลาไป 60 วินาที
        - อ่านค่าวัน แล้วกลับไปเปรียบเทียบ

ซึ่งจากรายละเอียดทั้งหมดของภาคต่างๆ ในการออกแบบฮาร์ดแวร์ ได้แสดงวงจรสมบรูณ์ไว้ในส่วนของภาคผนวก รวมทั้ง ลายวงจรพิมพ์ และรายละเอียด การลงอุปกรณ์ ของตัวบอร์ดควบคุม รวมทั้งโฟลว์ชาร์ท และโปรแกรมควบคุม

การออกแบบซอฟแวร์โปรแกรม

ในการออกแบบส่วนนี้จะใช้โปรแกรม Visual Basic 6 ในการเขียนโปรแกรม โดยโปรแกรมส่วนนี้ออกแบบเพื่อใช้ในการจัดการ การติดต่อเพื่อส่งข้อมูลที่ได้จากส่วนฮาร์ดแวร์ และนำข้อมูลที่ได้มาจัดเก็บในรูปแบบตาราง แบ่งออกตามวัน เวลาของข้อมูล อ้างอิงตามข้อมูลเวลาที่ได้บันทึกไว้ของฮาร์ดแวร์ โดยมีรูปแบบของตารางดังรูปที่ 3.21

DAY DATE MONTH YEAR		
TIMES	TEMP (^{O}C)	LIGHT (lux)
HH:MM:SS		

รูปที่ 3.21 แสดงลักษณะของข้อมูลที่นำมาจัดเก็บในตาราง

และนำข้อมูลที่อยู่ในตาราง มาแสดงผลในรูปแบบของกราฟเชิงเส้น อ้างอิงตามแกนเวลา

โดยโครงสร้างของโปรแกรมประกอบด้วย ส่วนควบคุมการติดต่อระหว่างฮาร์ดแวร์และคอมพิวเตอร์ ส่วนที่ทำหน้าที่ในการจัดการข้อมูล ส่วนการแสดงผล และจะมีส่วนที่ทำหน้าที่ในการตั้งเวลาของฮาร์ดแวร์

หลักการของซอฟแวร์

ในการออกแบบซอฟแวร์เพื่อใช้ในการติดต่อรับข้อมูลที่ได้จากฮาร์ดแวร์ รวมไปถึงการส่งข้อมูลวันเวลาเพื่อให้ฮาร์ดแวร์เริ่มทำงานในตอนแรก จำเป็นอย่างยิ่งที่จะต้องออกแบบซอฟแวร์ให้สามารถทำงานสัมพันธ์กันกับฮาร์ดแวร์ ซึ่งในการออกแบบโปรแกรมของโครงงานนี้ได้ใช้โปรแกรม Visual Basic 6 ในการออกแบบ โดยลักษณะของซอฟแวร์ตามโครงสร้างนั้นจะมีหน้าต่างโปรแกรมหลักหนึ่งหน้าต่าง และจะมีเมนูบาร์เพื่อเลือกการใช้งานที่หน้าต่างหลักนี้ เพื่อรองรับความต้องการของผู้ใช้ ซึ่งจากโครงสร้างของซอฟแวร์นี้ สามารถที่จะแสดงรายละเอียดได้ดังนี้ รายละเอียดของข้อมูลตามวันเวลาที่เริ่มการบันทึก, ลักษณะกราฟเชิงเส้นของอุณหภูมิและความเข้ม

แสงในวันจากไฟล์ที่เปิดอยู่, สามารถโอนย้ายข้อมูลจากฮาร์ดแวร์มาไว้ที่เครื่องคอมพิวเตอร์เพื่อเก็บบันทึกเป็นไฟล์ได้รวมทั้งยังสามารถนำไฟล์นี้ไปเปิดดูที่โปรแกรมจัดการเกี่ยวกับ text ไฟล์ได้ยกตัวอย่างเช่นโปรแกรม EXCEL , NOTEPAD , EDITPLUS เป็นต้น

การแสดงผลออกทางจอภาพ

การแสดงผลออกทางจอภาพนั้น จะใช้การแสดงผลในรูปแบบของตารางและรูปกราฟเชิงเส้น เพื่อให้ผู้ใช้สามารถที่จะวิเคราะห์ถึงจุดต่างๆ ของค่าอุณหภูมิและค่าความเข้มแสงในแต่ละช่วงเวลาของวันได้อย่างถูกต้อง

การแสดงผลแบบตาราง จะแสดงลักษณะของอุณหภูมิและความเข้มแสงตามลำดับของเวลาโดยถ้าเป็นวันแรกของการเก็บข้อมูลที่ฮาร์ดแวร์ เมื่อนำมาแสดงผลในซอฟแวร์ค่าแรกที่เก็บบันทึกได้จะเริ่มต้นที่ตำแหน่งเวลานั้นและหากเป็นวันต่อๆมาค่าเริ่มต้นจะอยู่ที่เวลา 0:00 นาฬิกา โดยซอฟแวร์ที่ออกแบบนี้สามารถจะค้นหาข้อมูลต่างๆ ที่ผู้ใช้งานต้องการ โดยดูได้จากเมนูบาร์

การแสดงผลแบบกราฟ เพื่อให้ผู้ใช้งานสามารถที่จะวิเคราะห์ถึงลักษณะการเปลี่ยนแปลงของค่าอุณหภูมิและค่าความเข้มแสงในวันที่ผู้ใช้งานต้องการทราบ โดยสามารถแบ่งการแสดงผลได้เป็น 2 รูปแบบ คือ กราฟแสดงผลค่าอุณหภูมิและ กราฟแสดงผลค่าความเข้มแสง และทั้ง 2 ชนิดสามารถแสดงลักษณะกราฟได้หลายรูปแบบเช่น กราฟแบบเส้น1 มิติ, กราฟแบบเส้น 3 มิติ, กราฟแท่ง 1 มิติ กราฟแท่ง 3 มิติ โดยที่กราฟแสดงผลนี้ ผู้ใช้งานสามารถคลิกที่บริเวณกราฟ ก็จะมีการแสดงผลออกมาว่าที่ตำแหน่งนั้นๆ ของกราฟ เป็นช่วงเวลาใด มีระดับอ้างอิงที่เท่าใด

การตั้งเวลาให้ฮาร์ดแวร์

การตั้งเวลาให้กับส่วนฮาร์ดแวร์ เพื่อให้ฮาร์ดแวร์เริ่มทำการเก็บบันทึกค่าต่างๆ ที่วันเวลาเวลาจริงโดยค่าเวลาต่างๆ ที่ตั้งให้กับตัวฮาร์ดแวร์ประกอบด้วย วัน, เดือน, ปี, ชั่วโมง, นาที, วินาที ซึ่งการตั้งเวลานี้สามารถเลือกสั่งงานได้ที่เมนูบาร์ของโปรแกรม

การเก็บข้อมูล

ในการเก็บบันทึกข้อมูลที่ได้จากฮาร์ดแวร์ไว้ในส่วนของคอมพิวเตอร์นี้ จะเก็บบันทึกอยู่ในรูปแบบของแฟ้มข้อมูล และแฟ้มข้อมูลที่เก็บบันทึกได้สามารถที่จะนำไปเปิดในโปรแกรมที่สนับสนุนทางด้าน text ไฟล์ได้

บทที่ 4

สรุปและวิจารณ์

จากการทำงานที่ผ่านมาทั้งหมดในโครงงานนี้ ตั้งแต่การเตรียมการหาข้อมูล แล้วนำข้อมูลที่ได้มาทำการวางแนวทางการออกแบบ โดยศึกษาถึงพื้นฐานของการวัดค่าอุณหภูมิ การวัดค่าแสง การเก็บบันทึกข้อมูล การอินเตอร์เฟสกับคอมพิวเตอร์ การเขียนโปรแกรมจัดการข้อมูล นำมาสู่การหาอุปกรณ์และเทคโนโลยีที่นำมาใช้กับการออกแบบนี้ โดยแยกการออกแบบได้เป็นสองส่วนคือ ส่วนฮาร์ดแวร์ และส่วนซอฟแวร์ แล้วนำไปสู่การออกแบบและการทำงานจนงานสำเร็จลุล่วง โดยปัญหาที่เกิดขึ้นในการทำงาน ส่วนการออกแบบฮาร์ดแวร์พบปัญหาเกี่ยวกับการใช้งานอุปกรณ์ผิดพลาดในการทดลอง การออกแบบผิดพลาด และปัญหาเกี่ยวกับการศึกษาทดลองโปรแกรมควบคุมฮาร์ดแวร์ซึ่งไม่สามารถทำการทดลองได้จริง ซึ่งทำการทดลองเพียงส่วนหนึ่งเท่านั้น โดยต้องทำการทดลองจริงเมื่อฮาร์ดแวร์เสร็จแล้ว ส่วนการออกแบบซอฟแวร์ซึ่งใช้โปรแกรมภาษา Visaul Basic 6 พบปัญหาเกี่ยวกับการควบคุมติดต่อพอร์ตไม่สามารถที่จะทำการเคลียรบัฟเฟอร์ของพอร์ตได้ แต่โปรแกรมตัวนี้ออกแบบให้สามารถรองรับการเขียนโปรแกรมการจัดการข้อมูล และการแสดงผลของข้อมูลได้ดี เพราะฉนั้นจึงยังเลือกใช้โปรแกรมตัวนี้ต่อในโครงงานนี้

ผลการทดลองใช้งานจริงของฮาร์ดแวร์ร่วมกับซอฟแวร์สามารถใช้งานได้อย่างสมบรูณ์ค่าข้อมูลที่เก็บบันทึกไว้ในฮาร์ดแวร์สามารถส่งไปยังคอมพิวเตอร์และเก็บบันทึกลงแฟ้มข้อมูลได้ถูกต้อง และสามารถนำมาแสดงผลด้วยกราฟได้ถูกต้อง โดยค่าข้อมูลที่ได้ค่าของอุณหภูมิสามารถแสดงผลไดใกล้เคียง ส่วนค่าแสงจะมีค่าผิดพลาดเล็กน้อยในช่วงเริ่มต้นและช่วงปลายของตัวเซนเซอร์เนื่องจากกราฟการทำงานไม่ลิเนี่ยร

และจากการทำงานทั้งหมดสามารถได้รับความรู้ความสามารถเพิ่มเติมจากประสบการณ์ในการทำงาน ทั้งการทดลองจริง และคำแนะนำที่ได้รับจากอาจารย์ที่ปรึกษา เพื่อที่สามารถจะนำไปใช้ในการทำงานต่างๆ ต่อไปในอนาคต

เอกสารอ้างอิง

1. ชัยวัฒน์ ลิ้มพรจิตวิไลมล,วรพจน์ กรแก้ววัฒนกุล,เรียนรู้และปฏิบัติการ ไมโครคอนโทรลเลอร์MCS-51,อินโนเวตีฟ เอ กซ์เพอริเมนต์ จำกัด,2542.

2. ธีรวัฒน์ ประกอบผล,การประยุกต์ใช้งาน ไมโครคอนโทรลเลอร์,สมาคมส่งเสริมเทคโนโลยี(ไทย-ญี่ปุ่น),2541

3. สมยศ จุณณะปิยะ,การประยุกต์ใช้งาน ไมโครคอนโทรลเลอร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง,2543

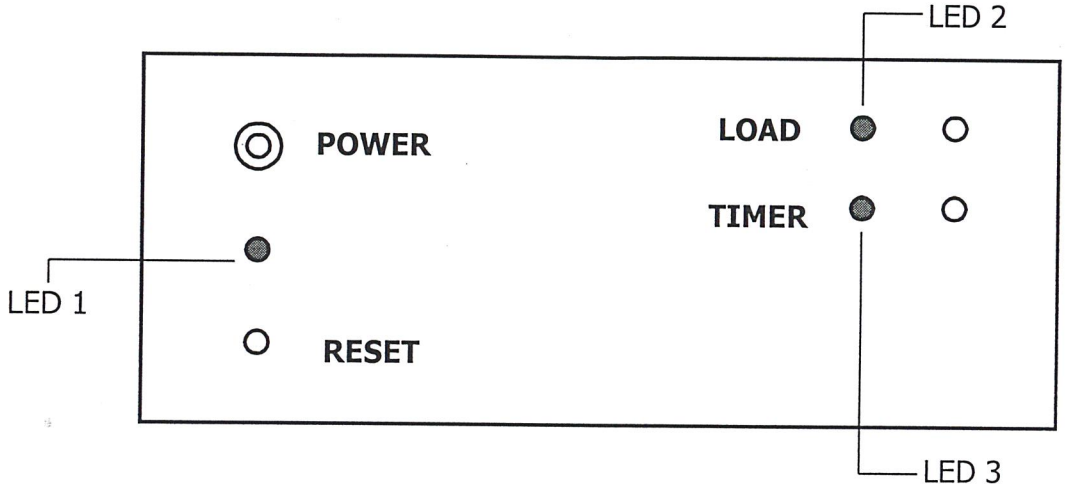
4. สุทธิ์ศักดิ์ พงศ์ธนาพานิช, Visual Basic 5.0 professicnal ,การใช้คำสั่งและคอนโทรล Activex ซีเอ็ดยูเคชั่น,2543

5. ฉันทวุฒิ พืชผล,พิชิต สันติกุลานนท์,คู่มือเรียน Visual Basic6, โปรวิชั่น จำกัด,2537

6. กิตติ ภักดีวัฒนะกุล;จำลอง ครูอุตสาหะ Visual Basic6 ฉบับโปรแกรมเมอร์,เคทีพีคอมพ์ แอนด์ คอนซัลน์ จำกัด,2542

ภาคผนวก

คู่มือการใช้งาน



- เมื่อกดสวิตช์ POWER เป็นสถานะออน LED 1 จะติด เพื่อบอกว่าเครื่องกำลังทำงานและกำลังเก็บค่าอยู่
- เมื่อกดสวิตช์ LOAD LED 2 จะติดเพื่อบอกว่าเครื่องอยู่ในสภาวะพร้อมที่จะทำการโหลดข้อมูล โดยให้ทำการเรียกโปรแกรมการโหลดข้อมูลมาใช้งาน และLED 2 จะดับเมื่อทำการโหลดข้อมูลเรียบร้อยแล้ว และสวิตช์ตัวนี้ต้องกดก่อนที่จะทำการปิดเครื่องเพื่อสิ้นสุดการเก็บค่าเสมอ
- เมื่อกดสวิตช์ TIMER LED 3 จะติดเพื่อบอกว่าเครื่องอยู่ในสภาวะพร้อมที่จะทำการตั้งเวลา โดยให้ทำการเรียกโปรแกรมการตั้งเวลามาใช้งาน และLED 3 จะดับเมื่อทำการตั้งเวลาเรียบร้อยแล้ว
- เมื่อกดสวิตช์ RESET จะเป็นการทำการรีเซตการทำงานของเครื่องให้กลับไปเริ่มต้นทำงานเก็บข้อมูลตั้งแต่ตำแหน่งเริ่มต้นใหม่ โดยทุกครั้งที่ทำการโหลดข้อมูลหรือตั้งเวลาเรียบร้อยแล้วจะต้องทำการรีเซตเครื่องเสมอ

<u>หมายเหตุ</u> การโหลดข้อมูลและการตั้งเวลาต้องทำในขณะที่เครื่องอยู่ในสภาวะออน และต้องต่อสายสัญญาณจาก SERIAL PORT ไปยังคอมพิวเตอร์แล้วเท่านั้น

การใช้งาน Soft Ware โปรแกรม

เมื่อเรียกโปรแกรมใช้งานขึ้นมาแล้ว โปรแกรมจะแสดงหน้าต่างที่มีเมนูบาร์คำสั่งขึ้นมา ประกอบด้วย คำสั่ง File, Graph, และ Tool ขึ้นมา โดยมีรายละเอียดดังนี้

File

- Open:เป็นคำสั่งที่ใช้ในการเรียกแฟ้มข้อมูลที่มีอยู่ขึ้นมาดู
- Load:เป็นคำสั่งที่ใช้ในการโหลดข้อมูลจากฮาร์ดแวร์
- Exit:เป็นคำสั่งที่ใช้ออกจากโปรแกรม

Graph

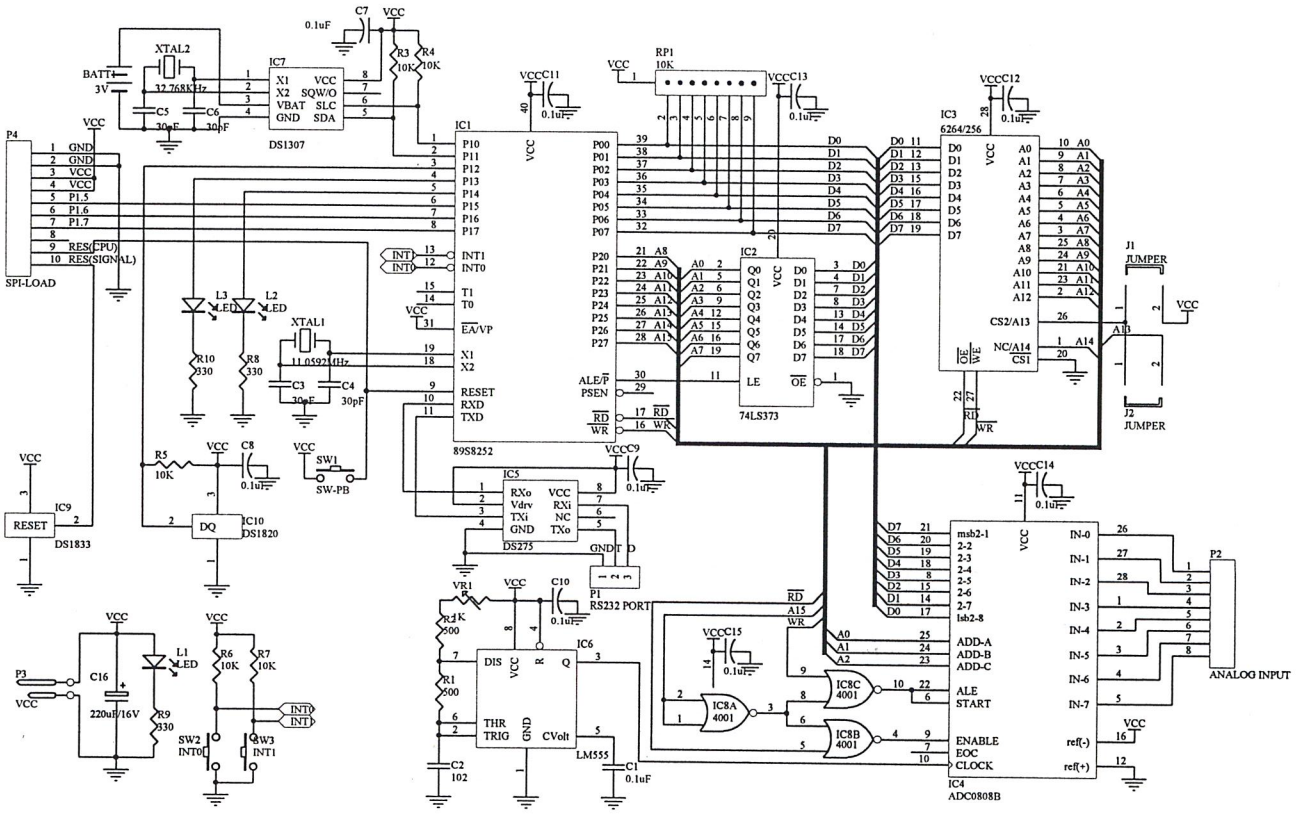
- Temp-Graph:เป็นคำสั่งที่ใช้ในการแสดงกราฟข้อมูลของอุณหภูมิ
- Light-Graph: เป็นคำสั่งที่ใช้ในการแสดงกราฟข้อมูลของแสง

โดยทั้งสองคำสั่งจะใช้ได้เมื่อมีค่าข้อมูลอยู่ในตารางแล้วเท่านั้น

Tool

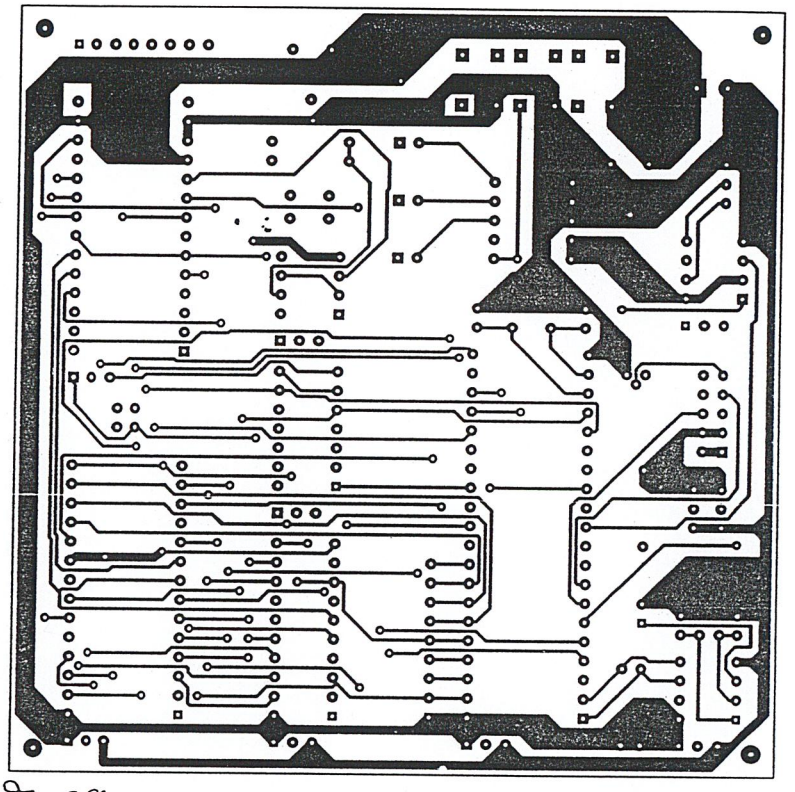
- Set Port:เป็นคำสั่งที่ใช้ในการเลือกพอร์ตอนุกรมของคอมพิวเตอร์ที่จะต่อใช้งาน โดยเมื่อเรียกใช้โปรแกรมทุกครั้งจะต้องทำการเซตพอร์ตก่อนเสมอ
- Set Time:เป็นคำสั่งที่ใช้ใน ดูเวลา และตั้งวัน เวลา ให้กับฮาร์ดแวร์เพื่อใช้ในการอ้างอิงการทำงาน
- Find Temp:เป็นคำสั่งที่ใช้ในการหาค่าข้อมูลของอุณหภูมิที่ต้องการจากตาราง
- Find Light:เป็นคำสั่งที่ใช้ในการหาค่าข้อมูลของแสงที่ต้องการจากตาราง

Circuit



Title	CONTROL BOARD		
Size: B	Number		Revision
Date:	15-May-2000	Sheet of	
File:	D:\Project\circuit\Pproj_1.sch	Drawn By:	

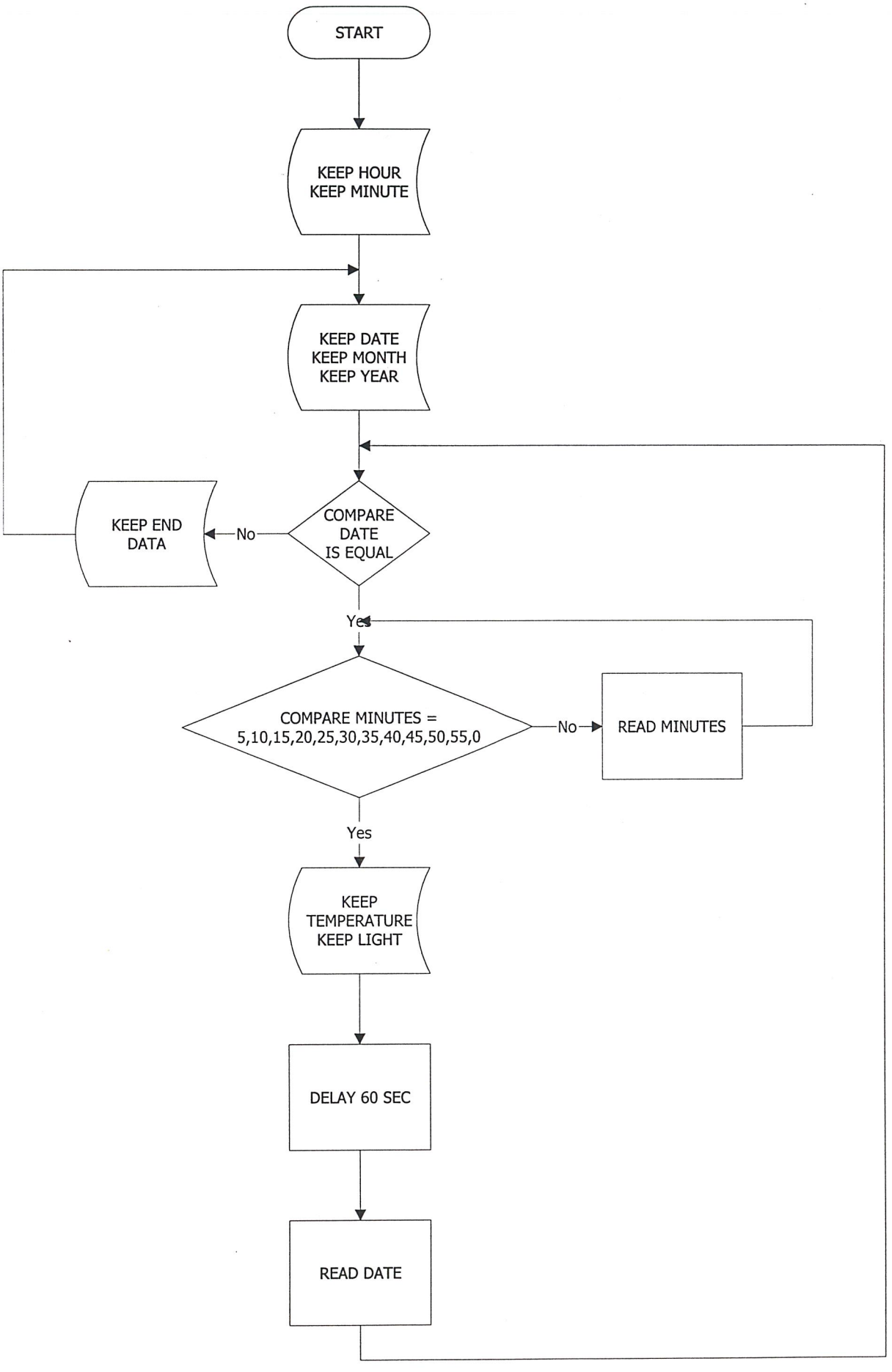
4 28



2 28

Assambly Program

CONTROL PROGRAM CHART



```
;*************************************************************
;FILENAME       PROJ.ASM PROGRAM
;DESCRIPTION    WEATHER RECORD PROJECT PROGRAM
;HARDWARE       MAIN BROAD
;ASSEMBLER      SXA51
;DATE           APRIL15,2000
;SOFTWARE ENG   A.KHEMWICHAI

;*******************************
;DEFINE PORT&PIN NAME
;*******************************

SCL             BIT       P1.0                ; SCL I2C Bus
SDA             BIT       P1.1                ; SDA I2C Bus
ONEWIRE         BIT       P1.2                ; 1-Wire Interface (DS1820 Temp Sensor
INT_LDE         BIT       P1.3                ; Interrupt 0 Status
INT_RTC         BIT       P1.4                ; Interrupt 1 Status

;*******************************
;DEFINE USER REGISTER
;*******************************

FLAG            EQU       2FH                 ; User Flag
;    7    6    5    4    3    2    1    0
;                                   BUSY I2C_ACK
I2C_ACK         BIT       FLAG.0              ; Define BUSY As bit
BUSY            BIT       FLAG.1              ; Define I2C Acknowledge As bit

ONEWIRE_DATA    EQU       20H                 ; For Keep Onewire Data
TEMP            EQU       21H                 ; For Keep Temp Data (Temp L Only)
I2C_ADDR        EQU       22H                 ; For Keep I2C Address
I2C_DATA        EQU       23H                 ; For Keep I2C Data
LIGHT_DATA      EQU       24H                 ; For Keep Light Data
DATE_T          EQU       25H                 ; For Keep Date Temporary
MINUTES_T       EQU       26H                 ; For Keep Minutes Temporary

SECONDS         EQU       27H                 ; For Keep Seconds
MINUTES         EQU       28H                 ; For Keep Minutes
HOURS           EQU       29H                 ; For Keep Hours
DAY             EQU       2AH                 ; For Keep Day
DATE            EQU       2BH                 ; For Keep Date
MONTH           EQU       2CH                 ; For Keep Month
YEAR            EQU       2DH                 ; For Keep Year
CONTROL         EQU       2EH                 ; For Keep Control Byte

L_BYTE          EQU       30H                 ; Keep Ram Address Low Byte
H_BYTE          EQU       31H                 ; Keep Ram Address High Byte

;*******************************
;DEFINE I2C SLAVE ADDRESS
;*******************************

RTC_ID          EQU       11010000B           ; RTC Slave Address

;*******************************
;DEFINE LIGHT ADDRESS
;*******************************

LIGHT_ADDR      EQU       8000H               ; Light Address

;*******************************
;PROGRAM
;*******************************

                ORG       0000H               ; Reset Vector

START:          SJMP      INITIAL             ; Jump Pass Interrupt Service

;*******************************
;EXTERNAL INTERUPT 0
;*******************************

                ORG       0003H               ; IE0 Vector

EX_INT_0:       AJMP      SERIAL              ; Jump To Interrupt Service Program
```

```
;*******************************
;EXTERNAL INTERUPT 1
;*******************************

                ORG     0013H               ; IE1 Vector

EX_INT_1:       AJMP    SET_RTC             ; Jump To Interrupt Service Program

;*******************************
;MAIN PROGRAM
;*******************************

                ORG     0030H

INITIAL:        CLR     INT_LDE             ; Initial Controller Status
                CLR     INT_RTC
                MOV     L_BYTE,#00H
                MOV     H_BYTE,#00H
                MOV     SP,#32H

                LCALL   RTC_RD              ; Read RTC From DS1307

                MOV     A,HOURS             ; Keep Hours To Ram
                ACALL   RAM_WR
                MOV     A,MINUTES           ; Keep Minutes To Ram
                ACALL   RAM_WR

MAIN:           MOV     IE,#10000101B       ; Enable EA,EX1,EX0

                MOV     A,DATE              ; Keep Date To Ram
                ACALL   RAM_WR
                MOV     A,MONTH             ; Keep Month To Ram
                ACALL   RAM_WR
                MOV     A,YEAR              ; Keep Year To Ram
                ACALL   RAM_WR

                MOV     DATE_T,DATE
MAIN_1:         MOV     A,DATE              ; Compare Date
                XRL     A,DATE_T
                JZ      MAIN_2              ; Jump If Fail
                MOV     A,#0FFH             ; Move End Of Data To Ram
                ACALL   RAM_WR
                SJMP    MAIN

MAIN_2:         MOV     MINUTES_T,#00H
MAIN_3:         MOV     A,MINUTES           ; Compare Minutes = 5,10,15,20,25,
                XRL     A,MINUTES_T         ; 30,35,40,45,50,55,0
                JZ      MAIN_4              ; Jump If True
                MOV     A,#55H
                XRL     A,MINUTES_T
                JZ      TIME_CHK            ; Jump If Fail
                MOV     A,MINUTES_T
                ADD     A,#05H
                DA      A
                MOV     MINUTES_T,A
                SJMP    MAIN_3

MAIN_4:         ACALL   DS1820_RST          ; DS1820 Reset
                ACALL   DS1820_PRES         ; DS1820 Presence
                MOV     ONEWIRE_DATA,#0CCH  ; Write Skip Rom
                ACALL   DS1820_WR           ; DS1820 Write
                MOV     ONEWIRE_DATA,#44H   ; Write Convert Command
                ACALL   DS1820_WR           ; DS1820 Write

                SETB    BUSY                ; Set Bit BUSY

PRES_CHK:       ACALL   DS1820_RST          ; DS1820 Reset
                ACALL   DS1820_PRES         ; DS1820 Presence
                JB      BUSY,PRES_CHK       ; Wait For Busy
                NOP                         ; Delay
                NOP
                NOP
                NOP
                ACALL   DS1820_RST          ; DS1820 Reset
                ACALL   DS1820_PRES         ; DS1820 Presence
                MOV     ONEWIRE_DATA,#0CCH  ; Write Skip Rom
```

```
                ACALL   DS1820_WR           ; DS1820 Write
                MOV     ONEWIRE_DATA,#0BEH  ; Write Read Scratchpad Command
                ACALL   DS1820_WR           ; DS1820 Write

                ACALL   DS1820_RD           ; Read DS1820
                MOV     TEMP,ONEWIRE_DATA   ; Get First Byte As TEMP (L)

                ACALL   DS1820_RST          ; DS1820 Reset
                ACALL   DS1820_PRES         ; DS1820 Presence

                MOV     A,TEMP              ; Get Temp Data
                CLR     C                   ; Clear Carry Flag
                RRC     A                   ; Rotate ACC To Right With Carry
                ACALL   RAM_WR

                MOV     DPTR,#LIGHT_ADDR    ; Read Light Data
                MOV     A,#00H
                MOVX    @DPTR,A             ; Start Convertion
                ACALL   DELAY_200us         ; Ckeck End Of Convertion Bit
                MOVX    A,@DPTR             ; Move Data To ACC
                MOV     LIGHT_DATA,A
                MOV     A,#00H
                MOVX    @DPTR,A             ; Start Convertion
                ACALL   DELAY_200us         ; Check End of Convertion Bit
                MOVX    A,@DPTR             ; Move Data To ACC
                CLR     C                   ; Clear Carry Flag
                ADDC    A,LIGHT_DATA        ; Average Light Data
                RRC     A
                ACALL   RAM_WR

                ACALL   DELAY_60s           ; Delay 60 s

TIME_CHK:       LCALL   RTC_RD              ; Read RTC From ds1307
                AJMP    MAIN_1              ; Jump To Main

;*******************************
;WRITE RAM
;*******************************

RAM_WR:         MOV     DPL,L_BYTE          ; Restore Ram Address
                MOV     DPH,H_BYTE
                MOVX    @DPTR,A             ; Move Data To Ram
                INC     DPTR                ; Increment Pointer
                MOV     H_BYTE,DPH          ; Keep Ram Address
                MOV     L_BYTE,DPL
                RET                         ; Return

;*******************************
;DELAY
;*******************************

DELAY_200us:    MOV     R6,#2EH             ; Each Loop = 200 us
DELAY_200us_1:  NOP
                NOP
                DJNZ    R6,DELAY_200us_1
                RET

DELAY_1s:       MOV     R7,#100             ; Each Loop = 1 s
DELAY_1s_1:     MOV     R6,#10
DELAY_1s_2:     MOV     R5,#0E6H
DELAY_1s_3:     NOP
                NOP
                DJNZ    R5,DELAY_1s_3
                DJNZ    R6,DELAY_1s_2
                DJNZ    R7,DELAY_1s_1
                RET

DELAY_60s:      MOV     R4,#60              ; Each Loop = 60 s
DELAY_60s_1:    ACALL   DELAY_1s
                DJNZ    R4,DELAY_60s_1
                RET

;*******************************
;EXTERNAL INTERUPT 0
;*******************************
```

```
SERIAL:     SETB    INT_LDE         ; Show Interrupt 0 Status
            CLR     EA              ; Disable Interrupt
            PUSH    ACC
            MOV     A,#0FEH         ; Move End Of Data To Ram
            ACALL   RAM_WR

            MOV     SCON,#52H       ; Set Mode 1 Rx Enable
            MOV     TMOD,#20H       ; Set TMOD
            MOV     TH1,#0FDH       ; Set Baud = 9600
            SETB    TR1             ; Start Timer 1

WAIT_S:     ACALL   R_XD
            MOV     R0,A            ; Compare Data = 'S'
            XRL     A,#53H
            JNZ     WAIT_S
            MOV     A,R0
            ACALL   T_XD

            MOV     DPTR,#0000H     ; Initial DPTR
SEND_1:     MOVX    A,@DPTR         ; Send Data To Computer
            ACALL   T_XD
            INC     DPTR            ; Increment DPTR
            XRL     A,#0FEH         ; Check End Of Data
            JNZ     SEND_1

            POP     ACC
            SETB    EA              ; Enable Interrupt
            CLR     INT_LDE

            RETI                    ; Return From Interupt Service

;*****************************
;EXTERNAL INTERRUPT 1
;*****************************

SET_RTC:    SETB    INT_RTC         ; Show Interrupt 1 Status
            CLR     EA              ; Disable Interrupt
            PUSH    ACC             ; Keep ACC To Stack

            MOV     SCON,#52H       ; Set Mode 1 Rx Enable
            MOV     TMOD,#20H       ; Set TMOD
            MOV     TH1,#0FDH       ; Set Baud = 9600
            SETB    TR1             ; Start Timer 1

WAIT_T:     ACALL   R_XD
            MOV     R0,A            ; Compare Data ='T'
            XRL     A,#54H
            JNZ     WAIT_T
            MOV     A,R0
            ACALL   T_XD

            ACALL   RTC_RD          ; Send Time To Computer
            MOV     A,DATE
            ACALL   T_XD            ; Date
            MOV     A,MONTH
            ACALL   T_XD            ; Month
            MOV     A,YEAR
            ACALL   T_XD            ; Year
            MOV     A,HOURS
            ACALL   T_XD            ; Hours
            MOV     A,MINUTES
            ACALL   T_XD            ; Minutes
            MOV     A,SECONDS
            ACALL   T_XD            ; Seconds

WAIT_Q:     ACALL   R_XD
            MOV     R0,A
            XRL     A,#51H          ; Compare Data = 'Q'
            JZ      QUIT            ; Jump If True
            MOV     A,R0
            XRL     A,#53H          ; Compare Data = 'S'
            JNZ     WAIT_Q

            ACALL   R_XD            ; Set Up RTC
            MOV     DATE,A          ; Set Date
            ACALL   R_XD
```

```
                MOV     MONTH,A             ; Set Month
                ACALL   R_XD
                MOV     YEAR,A              ; Set Year
                ACALL   R_XD
                MOV     HOURS,A             ; Set Hours
                ACALL   R_XD
                MOV     MINUTES,A           ; Set Minutes
                ACALL   R_XD
                MOV     SECONDS,A           ; Set Seconds
                ACALL   RTC_WR

                ACALL   RTC_RD              ; Read RTC
                MOV     A,DATE              ; Read Date
                ACALL   T_XD
                MOV     A,MONTH             ; Read Month
                ACALL   T_XD
                MOV     A,YEAR              ; Raed Year
                ACALL   T_XD
                MOV     A,HOURS             ; Read Hours
                ACALL   T_XD
                MOV     A,MINUTES           ; Read Minutes
                ACALL   T_XD
                MOV     A,SECONDS           ; Read Seconds
                ACALL   T_XD

QUIT:           POP     ACC                 ; Restore ACC From Stack
                SETB    EA                  ; Enable Interrupt
                CLR     INT_RTC
                RETI                        ; Return From Interrupt Service

R_XD:           JNB     RI,$                ; Wait Until RX Already
                CLR     RI                  ; Clear RI
                MOV     A,SBUF              ; Get Data From SBUF
                RET                         ; Return

T_XD:           JNB     TI,$                ; Wait Until TX Already
                CLR     TI                  ; Clear TI
                MOV     SBUF,A              ; Send Data To SBUF
                RET                         ; Return

;*************************************************************
;TEMPERATURE "DS1820"
;*************************************************************
;*******************************
;DS1820 DATA READ
;*******************************

DS1820_RD:      MOV     R4,#8               ; Set Loop 8 Times
                CLR     A                   ; Clear ACC
DS1820_RD_LOOP:CLR      ONEWIRE             ; Clear ONEWIRE
                NOP
                NOP
                SETB    ONEWIRE             ; Set ONEWIRE
                NOP
                NOP
                NOP
                NOP
                MOV     C,ONEWIRE           ; Get ONEWIRE To Carry Flag
                ACALL   ONEWIRE_DELAY       ; Delay 75 us
                RRC     A                   ; Rotate Right ACC With Carry Flag
                DJNZ    R4,DS1820_RD_LOOP   ; Do Until 8 Times
                MOV     ONEWIRE_DATA,A      ; Move ACC To ONEWIRE_DATA
                RET

;*******************************
;DS1820 DATA WRITE
;*******************************

DS1820_WR:      MOV     R4,#8               ; Set Loop 8 Times
                MOV     A,ONEWIRE_DATA      ; Get ONEWIRE_DATA
DS1820_WR_LOOP:RRC      A                   ; Rotate Right ACC With Carry Flag
                JNC     DS1820_WR_L         ; Carry Flag Was Set?
                CLR     ONEWIRE             ; Set => TX High
                NOP                         ; Delay
                NOP
                NOP
```

```
                NOP
                SETB    ONEWIRE             ; Set ONEWIRE
                ACALL   ONEWIRE_DELAY       ; Delay 75 us
                SJMP    DS1820_WR_NX        ; Jump To Next Write
DS1820_WR_L:    CLR     ONEWIRE             ; Clear => TX Low
                ACALL   ONEWIRE_DELAY       ; Delay 75 us
                SETB    ONEWIRE             ; Set ONEWIRE
                NOP                         ; Delay
                NOP
                NOP
                NOP
DS1820_WR_NX:   DJNZ    R4,DS1820_WR_LOOP   ; Do Until 8 Times
                RET                         ; Return

;*******************************
;DS1820 RESET
;*******************************

DS1820_RST:     CLR     ONEWIRE             ; Clear ONEWIRE
                ACALL   DELAY_1ms           ; Delay
                SETB    ONEWIRE             ; Set ONEWIRE
                MOV     R4,#8               ; Delay
                DJNZ    R4,$
                RET                         ; Return

;*******************************
;DS1820 RECEIVE PRRESENCE PULSE
;*******************************

DS1820_PRES:    MOV     R4,#8               ; Set Loop Wait 1
DS1820_PRES1:   MOV     R3,#0               ; Set Loop Wait 2
DS1820_PRES2:   JNB     ONEWIRE,DS1820_PRES3; Check ONEWIRE Was Clear?
                DJNZ    R3,DS1820_PRES2     ; Wait Loop Check 2
                DJNZ    R4,DS1820_PRES1     ; Wait Loop Check 1
                RET                         ; Return

DS1820_PRES3:   JNB     ONEWIRE,$           ; Wait Until ONEWIRE Set
                MOV     R4,#8               ; Delay
                DJNZ    R4,$
                CLR     BUSY                ; Clear BUSY Flag
                RET                         ; Return

;*******************************
;ONEWIRE_DELAY
;*******************************

ONEWIRE_DELAY:  MOV     R6,#12H             ; Each Loop = 75 us
ONEWIRE_DELAY1:NOP
                NOP
                DJNZ    R6,ONEWIRE_DELAY1
                RET

DELAY_1ms:      MOV     R6,#0E6H            ; Each Loop = 1 ms
DELAY_1ms1:     NOP
                NOP
                DJNZ    R6,DELAY_1ms1
                RET

;************************************************************
;************************************************************
;REAL TIME CLOCK  "DS1307"
;************************************************************
;*******************************
;RTC READ
;*******************************

RTC_RD:         MOV     I2C_ADDR,#RTC_ID    ; Set RTC As I2C Write Slave
                ACALL   I2C_SLAVE           ; Connect Slave

                MOV     I2C_DATA,#00H       ; Set Slave Address 00H
                ACALL   I2C_DATA_WR         ; Write Data To Slave

                ACALL   RTC_RD1             ; Connect RTC Read
                MOV     SECONDS,I2C_DATA    ; Read Data To SECONDS
                ACALL   I2C_NACK_BIT        ; Send Not Acknowledge
```

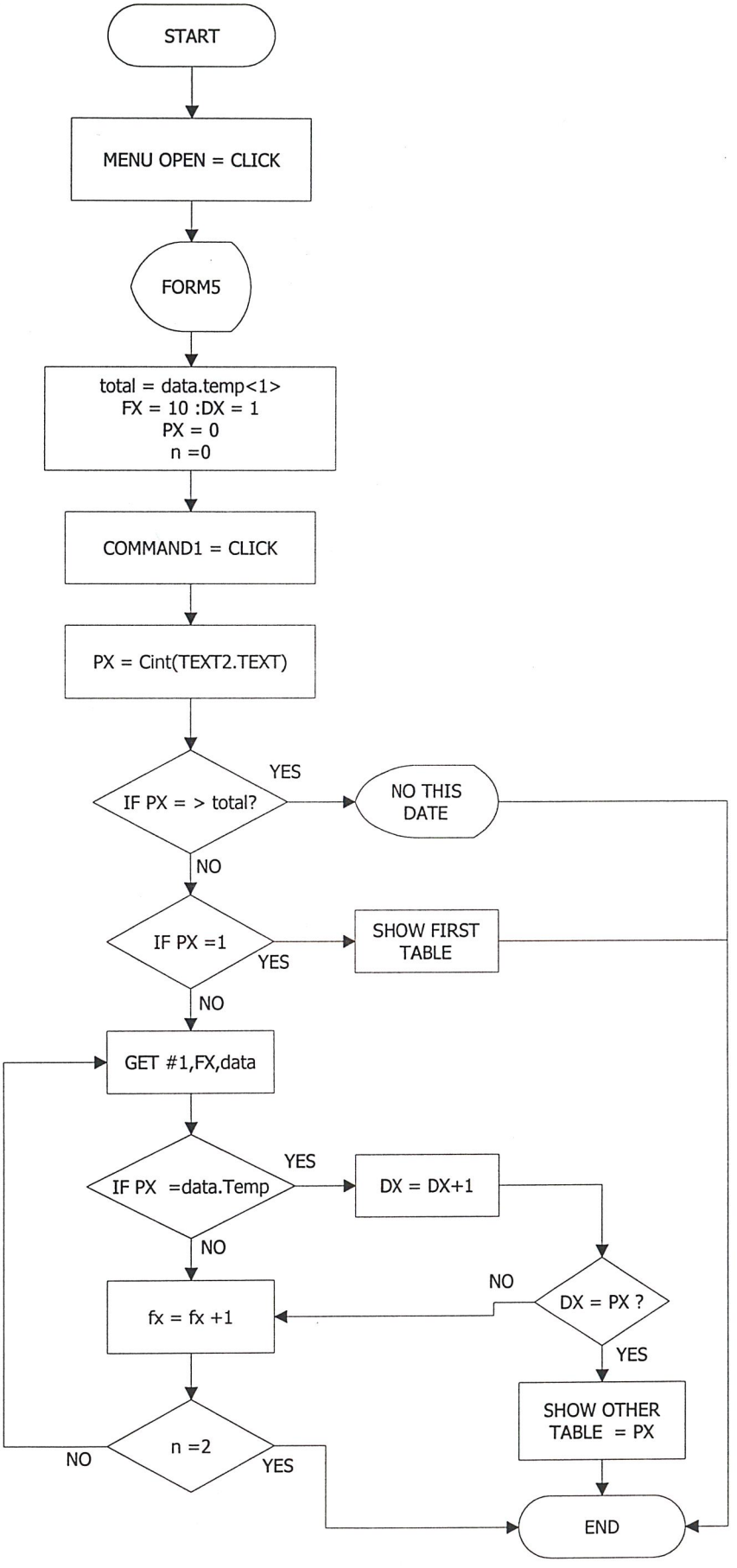
```
                ACALL   RTC_RD1             ; Connect RTC Read
                MOV     MINUTES,I2C_DATA    ; Read Data To MINUTES
                ACALL   I2C_NACK_BIT        ; Send Not Acknowledge

                ACALL   RTC_RD1             ; Connect RTC Read
                MOV     HOURS,I2C_DATA      ; Read Data To HOURS
                ACALL   I2C_NACK_BIT        ; Send Not Acknowledge

                ACALL   RTC_RD1             ; Connect RTC Read
                MOV     DAY,I2C_DATA        ; Read Data To DAY
                ACALL   I2C_NACK_BIT        ; Send Not Acknowledge

                ACALL   RTC_RD1             ; Connect RTC Read
                MOV     DATE,I2C_DATA       ; Read Data To DATE
                ACALL   I2C_NACK_BIT        ; Send Not Acknowledge

                ACALL   RTC_RD1             ; Connect RTC Read
                MOV     MONTH,I2C_DATA      ; Read Data To MONTH
                ACALL   I2C_NACK_BIT        ; Send Not Acknowledge

                ACALL   RTC_RD1             ; Connect RTC Read
                MOV     YEAR,I2C_DATA       ; Read Data To YEAR
                ACALL   I2C_NACK_BIT        ; Send Not Acknowledge

                ACALL   RTC_RD1             ; Connect RTC Read
                MOV     CONTROL,I2C_DATA    ; Read Data To CONTROL
                ACALL   I2C_NACK_BIT        ; Send Not Acknowledge

                ACALL   I2C_STOP            ; Send Stop Condition
                RET                         ; Return

RTC_RD1:        MOV     I2C_ADDR,#RTC_ID+1  ; Set RTC As I2C Read Slave
                ACALL   I2C_SLAVE           ; Connect Slave
                ACALL   I2C_DATA_RD         ; Read Data From Slave
                RET                         ; Return

;*******************************
;RTC WRITE
;*******************************

RTC_WR:         MOV     I2C_ADDR,#RTC_ID    ; Set RTC As I2C Write Slave
                ACALL   I2C_SLAVE           ; Connect Slave

                MOV     I2C_DATA,#00H       ; Set Slave Address 00H
                ACALL   I2C_DATA_WR         ; Write Data To Slave

                MOV     I2C_DATA,SECONDS    ; Write SECONDS To RTC
                ACALL   I2C_DATA_WR         ; Write Data To Slave

                MOV     I2C_DATA,MINUTES    ; Write MINUTES To RTC
                ACALL   I2C_DATA_WR         ; Write Data To Slave

                MOV     I2C_DATA,HOURS      ; Write HOURS To RTC
                ACALL   I2C_DATA_WR         ; Write Data To Slave

                MOV     I2C_DATA,DAY        ; Write DAY To RTC
                ACALL   I2C_DATA_WR         ; Write Data To Slave

                MOV     I2C_DATA,DATE       ; Write DATE To RTC
                ACALL   I2C_DATA_WR         ; Write Data To Slave

                MOV     I2C_DATA,MONTH      ; Write MONTH To RTC
                ACALL   I2C_DATA_WR         ; Write Data To Slave

                MOV     I2C_DATA,YEAR       ; Write YEAR To RTC
                ACALL   I2C_DATA_WR         ; Write Data To Slave

                MOV     I2C_DATA,CONTROL    ; Write CONTROL To RTC
                ACALL   I2C_DATA_WR         ; Write Data To Slave

                ACALL   I2C_STOP            ; Send Stop Condition
                RET                         ; Return

;*******************************
;DATA READ
;O/P:           I2C_DATA
```

```
;*******************************

I2C_DATA_RD:    PUSH      ACC                 ; Push ACC
                CLR       A                   ; Clear A
                MOV       R5,#8               ; Set Loop 8 Times
I2C_DATA_RD1:   ACALL     I2C_DELAY           ; Delay
                SETB      SCL                 ; Set SCL
                ACALL     I2C_DELAY           ; Delay
                MOV       C,SDA               ; Get SCL To Carry Flag
                RLC       A                   ; Rotate ACC To Left With Carry
                CLR       SCL                 ; Clear SCL
                DJNZ      R5,I2C_DATA_RD1     ; Do Until 8 Times
                MOV       I2C_DATA,A          ; Move Data To I2C_DATA
                POP       ACC                 ; Pop ACC
                RET                           ; Return

;*******************************
;DATA WRITE
;I/P:           I2C_DATA
;*******************************

I2C_DATA_WR:    PUSH      ACC                 ; Push ACC
                SETB      I2C_ACK             ; Set ACK bit
                MOV       A,I2C_DATA          ; Get Data
                MOV       R5,#8               ; Set Loop 8 Times
I2C_DATA_WR1:   RLC       A                   ; Rotate ACC To Left With Carry
                MOV       SDA,C               ; Move Carry Fiag To SDA
                ACALL     I2C_CLK             ; Pulse I2C Clock
                DJNZ      R5,I2C_DATA_WR1     ; Do Until 8 Times
                SETB      SDA                 ; Set SDA
                ACALL     I2C_DELAY           ; Delay
                SETB      SCL                 ; Set SCL
                ACALL     I2C_DELAY           ; Delay
                JB        SDA,I2C_DATA_WR2    ; Check Acknowledge From Slave
                CLR       I2C_ACK             ; Clear ACK bit
I2C_DATA_WR2:   CLR       SCL                 ; Clear SCL
                POP       ACC                 ; Pop ACC
                RET                           ; Return

;*******************************
;SLAVE CONNECT
;I/P:           I2C_ADDR
;O/P FLAG:      I2C_ACK
;*******************************

I2C_SLAVE:      PUSH      ACC                 ; Push ACC
                SETB      I2C_ACK             ; Set ACK bit
                MOV       A,I2C_ADDR          ; Get Slave Address
                ACALL     I2C_START           ; Send Start Condition

                MOV       R5,#8               ; Set Loop 8 Times
I2C_SLAVE1:     RLC       A                   ; Rotate ACC To Left With Carry
                MOV       SDA,C               ; Move Carry Flag To SDA
                ACALL     I2C_CLK             ; Pulse I2C Clock
                DJNZ      R5,I2C_SLAVE1       ; Do Until 8 Times

                SETB      SDA                 ; Set SDA
                ACALL     I2C_DELAY           ; Delay
                SETB      SCL                 ; Set SCL
                ACALL     I2C_DELAY           ; Delay
                JB        SDA,I2C_SLAVE2      ; Check Acknowledge From Slave
                CLR       I2C_ACK             ; Clear ACK

I2C_SLAVE2:     CLR       SCL                 ; Clear SCL
                POP       ACC                 ; Pop ACC
                RET                           ; Return

;*******************************
;START CONDITION
;*******************************

I2C_START:      SETB      SCL                 ; Set SCL
                SETB      SDA                 ; Set SDA
                ACALL     I2C_DELAY           ; Delay
                CLR       SDA                 ; Clear SDA During SCL Set
                ACALL     I2C_DELAY           ; Delay
```

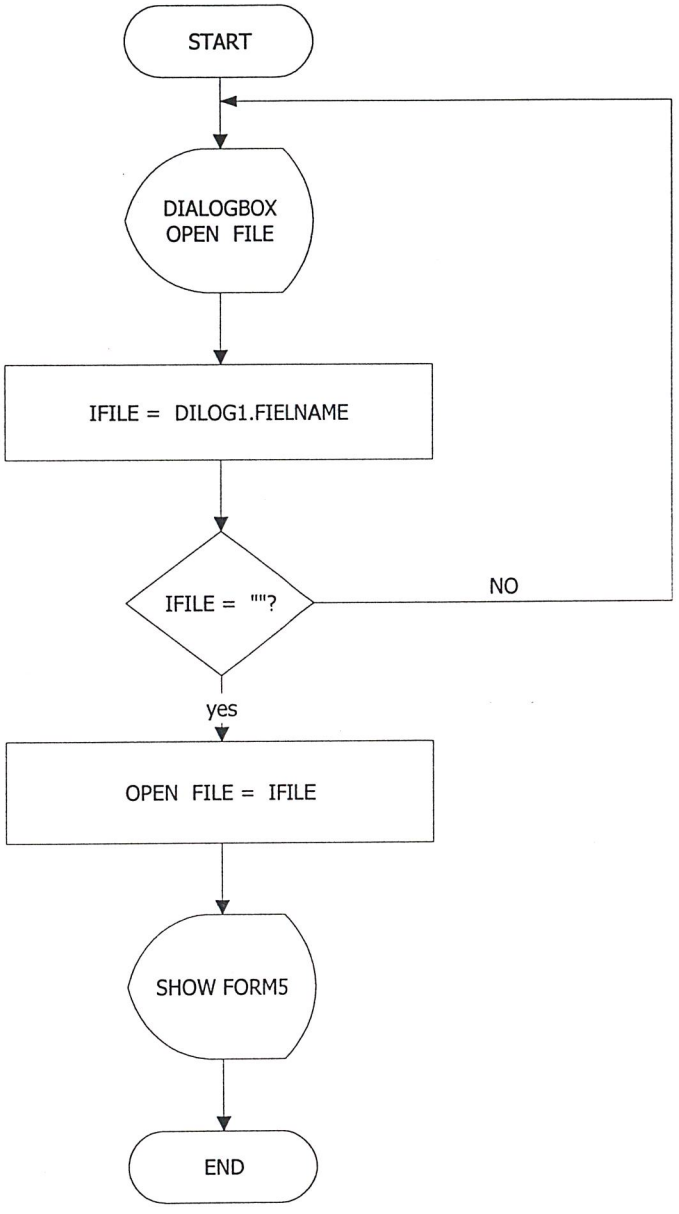
```
                CLR     SCL                     ; Clear SCL
                RET                             ; Return

;*******************************
;STOP CONDITION
;*******************************

I2C_STOP:       CLR     SDA                     ; Clear SDA
                ACALL   I2C_DELAY               ; Delay
                SETB    SCL                     ; Set SCL
                ACALL   I2C_DELAY               ; Delay
                SETB    SDA                     ; Set SDA During SCL Set
                RET                             ; Return

;*******************************
;I2C CLOCK
;*******************************

I2C_CLK:        ACALL   I2C_DELAY               ; Pulse SCL
                SETB    SCL
                ACALL   I2C_DELAY
                CLR     SCL
                RET                             ; Return

;*******************************
;I2C NOT ACKNOWLEDGE
;*******************************

I2C_NACK_BIT:   SETB    SDA                     ; Set SDA
                ACALL   I2C_DELAY               ; Delay
                ACALL   I2C_CLK                 ; Pulse I2C Clock
                RET                             ; Return

;*******************************
;I2C DELAY
;*******************************

I2C_DELAY:      MOV     R6,#0CH                 ; Each Loop = 50 us
I2C_DELAY1:     NOP
                NOP
                DJNZ    R6,I2C_DELAY1
                RET                             ; Return

;*************************************************************
END
;*************************************************************
```

Visual Basic Program

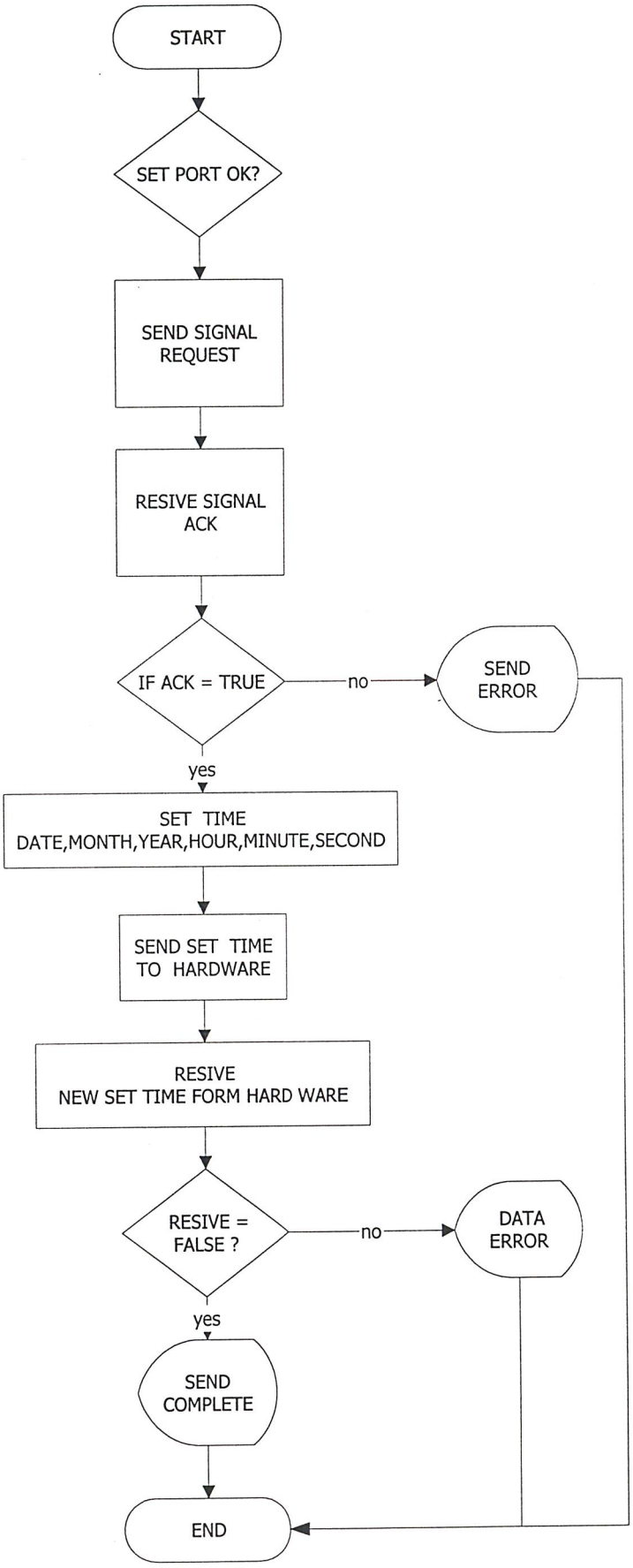
FLOW CHART SHOW TABLE



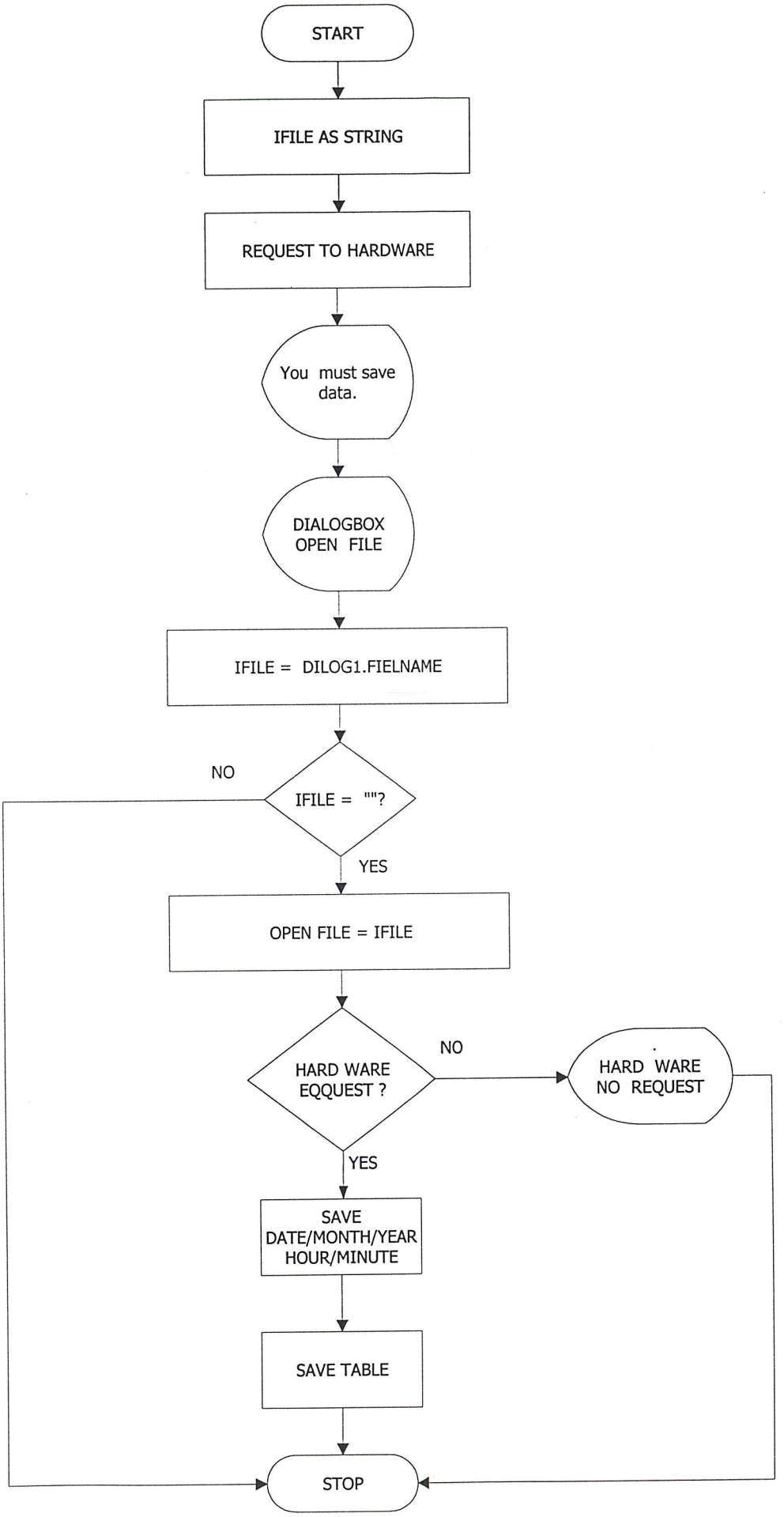
FLOW CHART MENU OPEN



FLOW CHART SET TIME



Flow Chart Load



```
VERSION 5.00
Object = "{648A5603-2C6E-101B-82B6-000000000014}#1.1#0"; "MSCOMM32.OCX"
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Object = "{5E9E78A0-531B-11CF-91F6-C2863C385E30}#1.0#0"; "MSFLXGRD.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Begin VB.Form Form1
   Caption         =   "Record Program "
   ClientHeight    =   9225
   ClientLeft      =   165
   ClientTop       =   795
   ClientWidth     =   6465
   LinkTopic       =   "Form1"
   ScaleHeight     =   9225
   ScaleWidth      =   6465
   StartUpPosition =   3  'Windows Default
   Begin VB.Timer Timer1
      Enabled         =   0   'False
      Interval        =   200
      Left            =   240
      Top             =   8760
   End
   Begin VB.TextBox Text1
      BackColor       =   &H00FFFF00&
      Height          =   372
      Left            =   480
      TabIndex        =   2
      Top             =   960
      Width           =   735
   End
   Begin MSComDlg.CommonDialog dialog1
      Left            =   5880
      Top             =   8520
      _ExtentX        =   688
      _ExtentY        =   688
      _Version        =   393216
   End
   Begin MSComctlLib.ProgressBar ProgressBar1
      Height          =   255
      Left            =   1440
      TabIndex        =   1
      Top             =   8880
      Width           =   2655
      _ExtentX        =   4683
      _ExtentY        =   450
      _Version        =   393216
      Appearance      =   1
      Scrolling       =   1
   End
   Begin MSCommLib.MSComm BCComm
      Left            =   5880
      Top             =   8640
      _ExtentX        =   794
      _ExtentY        =   794
      _Version        =   393216
      DTREnable       =   -1  'True
   End
   Begin MSFlexGridLib.MSFlexGrid MSFlexGrid1
      Height          =   6735
      Left            =   360
      TabIndex        =   0
      Top             =   1800
      Width           =   5655
      _ExtentX        =   9975
      _ExtentY        =   11880
      _Version        =   393216
      Rows            =   289
      Cols            =   3
      BackColor       =   16776960
      BackColorBkg    =   16744576
      GridColorFixed  =   8388736
      TextStyleFixed  =   3
      GridLinesFixed  =   3
   End
   Begin VB.Label Label4
      Caption         =   "Statrus"
      Height          =   375
```

```
      Left            =   480
      TabIndex        =   6
      Top             =   600
      Width           =   735
   End
   Begin VB.Line Line1
      BorderColor     =   &H00000080&
      BorderWidth     =   2
      Index           =   1
      X1              =   4920
      X2              =   4680
      Y1              =   840
      Y2              =   1200
   End
   Begin VB.Shape Shape1
      BorderColor     =   &H00FF0000&
      BorderStyle     =   6  'Inside Solid
      Height          =   615
      Left            =   1800
      Top             =   720
      Width           =   4095
   End
   Begin VB.Line Line1
      BorderColor     =   &H00000080&
      BorderWidth     =   2
      Index           =   0
      X1              =   2640
      X2              =   2400
      Y1              =   840
      Y2              =   1200
   End
   Begin VB.Label Label3
      Caption         =   "yy"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H8000000D&
      Height          =   495
      Left            =   5040
      TabIndex        =   5
      Top             =   840
      Width           =   735
   End
   Begin VB.Label Label2
      Alignment       =   2  'Center
      Caption         =   "mm"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00800000&
      Height          =   735
      Left            =   2400
      TabIndex        =   4
      Top             =   840
      Width           =   2415
   End
   Begin VB.Label Label1
      Caption         =   "dd"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
```

```
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00800000&
      Height          =   495
      Left            =   2040
      TabIndex        =   3
      Top             =   840
      Width           =   495
   End
   Begin VB.Menu MnuFile
      Caption         =   "&File"
      Begin VB.Menu MnuOpen
         Caption         =   "Open"
      End
      Begin VB.Menu MnuLoad
         Caption         =   "Load"
      End
      Begin VB.Menu MnuFileBar0
         Caption         =   "-"
      End
      Begin VB.Menu MnuExit
         Caption         =   "Exit"
      End
   End
   Begin VB.Menu MnuGraph
      Caption         =   "Graph"
      Begin VB.Menu MnuTG
         Caption         =   "Temp_Graph"
         Begin VB.Menu MnuTemp
            Caption         =   "Temp"
         End
         Begin VB.Menu MnuTemp1
            Caption         =   "Temp1"
         End
      End
      Begin VB.Menu MnuLg
         Caption         =   "Light_Graph"
         Begin VB.Menu MnuLight
            Caption         =   "Light"
         End
         Begin VB.Menu MnuLight1
            Caption         =   "Light1"
         End
      End
   End
   Begin VB.Menu MnuTool
      Caption         =   "Tool"
      Begin VB.Menu MnuSetPort
         Caption         =   "Set Port"
      End
      Begin VB.Menu MnuSetTime
         Caption         =   "SetTime"
      End
      Begin VB.Menu MnuToolbar0
         Caption         =   "-"
      End
      Begin VB.Menu MnuFundTemp
         Caption         =   "Find Temp"
         Begin VB.Menu MnuList
            Caption         =   "List"
         End
      End
      Begin VB.Menu MnuFindLight
         Caption         =   "Find  Light"
         Begin VB.Menu MnuListLight
            Caption         =   "List"
         End
      End
   End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
```

```
' PROGRAM  RECORD
Option Explicit

Private Sub Form_Load()
    ' adjust column width
    MnuGraph.Enabled = False    ' Set Mnu befor operate
    MnuLoad.Enabled = False
    MnuSetTime.Enabled = False

    With MSFlexGrid1
        .ColWidth(0) = 700
        .ColWidth(1) = 950
        .ColWidth(2) = 950
        .FixedAlignment(1) = flexAlignCenterCenter
        .FixedAlignment(2) = flexAlignCenterCenter
    End With
        Call set_table  ' set format table
        ProgressBar1.Align = 2
        ProgressBar1.Visible = False
FormLoadExit:
Exit Sub
End Sub

Private Sub Form_Resize()
    MSFlexGrid1.Width = Abs(ScaleWidth - 800)
    MSFlexGrid1.Height = Abs(ScaleHeight - 2000)
End Sub
Public Sub set_table()
Dim Mn, Hr, i As Integer
Dim j As Long
Dim add As String

MSFlexGrid1.FormatString = "^Time       |^   Temp  ( C  í )        |^    Light    (
j = 0
For i = 0 To 2
j = i
MSFlexGrid1.Col = j
MSFlexGrid1.row = 0
 MSFlexGrid1.CellFontBold = True
j = j + 1
Next i

j = 2
Hr = 0
Mn = 0
MSFlexGrid1.TextMatrix(1, 0) = "0" & ":" & "00"
    For i = 2 To 288
      Mn = Mn + 5
          If Mn > 55 Then
              Mn = 0
              Hr = Hr + 1
          End If
    If Mn = 0 Then
       MSFlexGrid1.TextMatrix(j, 0) = str(Hr) & ":00"
       ElseIf Mn = 5 Then
       MSFlexGrid1.TextMatrix(j, 0) = str(Hr) & ":05"
       Else
       MSFlexGrid1.TextMatrix(j, 0) = str(Hr) & ":" & str(Mn)
       End If
    j = j + 1
    Next i
End Sub

Public Function SaveOneDay()
       myhour(1) = CInt(Hex(Asc(BCComm.Input)))
       myminute(1) = CInt(Hex(Asc(BCComm.Input)))
       mydate(1) = CInt(Hex(Asc(BCComm.Input)))
       mymonth(1) = CInt(Hex(Asc(BCComm.Input)))
       myyear(1) = CInt(Hex(Asc(BCComm.Input)))
       header                ' Save header Table
End Function

Private Function Dlg1(d As String) As String
Dim filena As String
With dialog1
     .DialogTitle = d
```

```
      .CancelError = False
      .Filter = "for all file (*.txt)|*.txt"
      .ShowSave
               If Len(.FileName) = 0 Then
                  Exit Function
               End If
       filena = .FileName
    Dlg1 = filena
  End With
End Function

Public Function header()
Dim a, i, c, p, j, total As Integer
Dim n As Integer

        Open ifile For Random As #1 Len = Len(data)
        ' Save Header File
        data.Temp = "Date     "
        data.Light = "Month    "
        data.crlf = Chr(13) + Chr(10)
        Put #1, 2, data

        data.Temp = CStr(mydate(1))
        data.Light = CStr(mymonth(1))
        data.crlf = Chr(13) + Chr(10)
        Put #1, 3, data

        data.Temp = "Year     "
        data.Light = "(20-00)"
        data.crlf = Chr(13) + Chr(10)
        Put #1, 4, data

        data.Temp = "   20    "
        If myyear(1) < 9 Then
           data.Light = "0" & CStr(myyear(1))
        Else
            data.Light = CStr(myyear(1))
        End If
        data.crlf = Chr(13) + Chr(10)
        Put #1, 5, data

        data.Temp = "Hour     "
        data.Light = "Minute   "
        data.crlf = Chr(13) + Chr(10)
        Put #1, 6, data

        data.Temp = CStr(myhour(1))
        data.Light = CStr(myminute(1))
        data.crlf = Chr(13) + Chr(10)
        Put #1, 7, data

        data.Temp = "Temp     "
        data.Light = "Light    "
        data.crlf = Chr(13) + Chr(10)
        Put #1, 8, data
'   Loop for Save
n = 0
i = 9: c = 1: p = 1: j = 0: total = 1
Do While n = 0
   Ax(c) = Asc(BCComm.Input)      ' resive  new ascii  form port
           If Ax(c) = 255 Then    ' detect  FF

              data.Temp = "0               "
              data.Light = "0                "
              data.crlf = Chr(13) + Chr(10)
              Put #1, i, data

              mydate(1) = CInt(Hex(Asc(BCComm.Input)))   ' resive  write new date
              mymonth(1) = CInt(Hex(Asc(BCComm.Input)))
              myyear(1) = CInt(Hex(Asc(BCComm.Input)))

              data.Temp = "Date     "                    ' write new date to file
              data.Light = "Month    "
              data.crlf = Chr(13) + Chr(10)
              i = i + 1
              Put #1, i, data                            ' i table 1
```

```
                data.Temp = CStr(mydate(1))
                data.Light = CStr(mymonth(1))
                data.crlf = Chr(13) + Chr(10)
                i = i + 1
                Put #1, i, data                          ' i table 2

                data.Temp = "Year     "
                data.Light = "(20-00)"
                data.crlf = Chr(13) + Chr(10)
                i = i + 1
                Put #1, i, data                         ' i tabel 3

                data.Temp = "    20   "
                If myyear(1) < 9 Then
                data.Light = "0" & CStr(myyear(1))
                Else
                data.Light = CStr(myyear(1))
                End If
                data.crlf = Chr(13) + Chr(10)
                i = i + 1
                Put #1, i, data                         ' i table 4

                data.Temp = "Temp     "
                data.Light = "Light    "
                data.crlf = Chr(13) + Chr(10)
                i = i + 1
                Put #1, i, data                         ' i table 5

                total = total + 1
                GoTo jump
                Else: End If                            ' end  ASC = 255

        If Ax(c) = 254 Then    ' detect FE or detect end of date
                BCComm.PortOpen = False      ' close port  & insert "00" end of table
                For j = i To (i + 2)
                data.Temp = "0        "
                data.Light = "0        "
                data.crlf = Chr(13) + Chr(10)
                Put #1, j, data
                Next j

                data.Temp = "TOTAL    "
                data.Light = CStr(total)
                Put #1, 1, data

                Close #1
                Exit Function
                Else: End If                ' end of detect FE (Temp)
                data.Temp = CStr(Ax(c))

                Ax(p) = Asc(BCComm.Input)   ' new  ascii form port
                If Ax(p) = 254 Then         ' detect FE at Light
                BCComm.PortOpen = False     ' close port & insert " 00 " at end of tabl
                For j = i To (i + 2)
                data.Temp = "0        "
                data.Light = "0        "
                data.crlf = Chr(13) + Chr(10)
                Put #1, j, data
                Next j

                data.Temp = "TOTAL      "
                data.Light = CStr(total)
                Put #1, 1, data
                Close #1
                Exit Function
                Else: End If             ' end detect  Light
                data.Light = CStr(Ax(p))
                data.crlf = Chr(13) + Chr(10)
                Put #1, i, data
jump:
                        i = i + 1
 Loop
End Function
Private Sub MnuExit_Click()
End
```

```
End Sub
Private Sub MnuLight_Click()
Form2.Show
End Sub

Private Sub MnuLight1_Click()
Form6.Show
End Sub

Private Sub MnuList_Click()
Dim LL As Long
LL = 1
List1 (LL)
End Sub

Private Sub MnuListLight_Click()
Dim LL As Long
LL = 2
List1 (LL)
End Sub

Private Sub MnuLoad_Click()
Dim chkfrm As String
Dim Sa As String
Dim indata As Integer
Dim a As Integer

 MsgBox " You must save data "
 Sa = " Save File "
 ifile = Dlg1(Sa)

 With BCComm
        .InputLen = 1
        .PortOpen = True
        .Settings = "9600,N,8,1"

 End With
 On Error Resume Next
 BCComm.Output = Chr(83)  ' send char 'S'
 a = 0
 Do
 If BCComm.InBufferCount >= 1 Then
           Do
           indata = Asc(BCComm.Input)
           If indata = 83 Then:
           Text1.Text = "  Ready "
           a = 1
            If Chr(83) <> Chr(indata) Then
                BCComm.PortOpen = False
                Text1.Text = "NON"
                MsgBox " ERROR !" + Chr(10) + Chr(13) + " CHEK AGAIN  AND START BOT
                Text1.Text = "    "
                a = 1: Exit Sub
            End If
        SaveOneDay
     Loop Until a = 1
  Else: a = 0
 End If
 Loop Until a = 1
 BCComm.PortOpen = False    ' off port again
End Sub



Private Sub MnuOpen_Click()
MnuGraph.Enabled = True
Dim filena As String
With dialog1
     .DialogTitle = "Open"
     .CancelError = False
     .Filter = "for all file (*.txt)|*.txt"
     .ShowOpen
              If Len(.FileName) = 0 Then
                 Exit Sub
              End If
```

```
      filena = .FileName
   ifile = filena
 End With
Form4.Show          '   operat at  FROM4
                        If chkfrm = "file error" Then Unload Form4
End Sub

Private Sub MnuSetPort_Click()
   port = InputBox(" Please Enter Port ", "Enter Port ", 0, 100, 200)
   If (CInt(Asc(port))) > 57 Then
      MsgBox " Error inset ,insert again"
      ElseIf (CInt(Asc(port))) < 49 Then
      MsgBox " Error inset ,insert again"
   Else
      BCComm.CommPort = CInt(port)
      MnuLoad.Enabled = True
       MnuSetTime.Enabled = True
   End If
End Sub

Private Sub MnuSetTime_Click()
Form5.Show
End Sub

Private Sub MnuTemp_Click()
Form3.Show
End Sub

Private Sub MnuTemp1_Click()
Form7.Show
End Sub

Private Sub Timer1_Timer()
Static p
p = p + 1
Select Case p
Case 1
                Label1.ForeColor = &HC0&
                Label2.ForeColor = &HFF0000
                Label3.ForeColor = &HC000C0
Case 2
                Label1.ForeColor = &HFF0000
                Label2.ForeColor = &HC0&
                Label3.ForeColor = &HC000C0
Case 3
                Label1.ForeColor = &HFF0000
                Label2.ForeColor = &HC000C0
                Label3.ForeColor = &HC0&
p = 0
End Select
End Sub

Public Function List1(LL As Long)
Dim list, l As String
Dim count, k, i As Integer
Dim j As Long
' input light want to find
Do
list = InputBox("How much the light do you want?", _
                "Find     ", "", 300, 200)
Loop Until list <> ""

    'Select entire grid and remove bold formatting
    '(to remove the results of previous finds)
   With MSFlexGrid1
            .FillStyle = flexFillRepeat
            .Col = LL
            .row = 1
            .ColSel = LL
            .RowSel = MSFlexGrid1.Rows - 1
            .CellFontBold = False
   End With

    'Initialize ProgressBar to track search
       ProgressBar1.Visible = True
    'Search the grid cell by cell for find interger
```

```
    MSFlexGrid1.FillStyle = flexFillSingle

    count = 0
    j = 0
    For i = 1 To 288
        j = j + 1
        k = ((100 / 288) * i)       ' control slider1
        ProgressBar1.Value = k
            'If current cell matches find text box
               l = CInt(MSFlexGrid1.TextMatrix(j, LL))   ' Convert text of cell as i
               If l = CInt(list) Then
                '...select cell and format bold
                MSFlexGrid1.Col = LL
                MSFlexGrid1.row = j
                MSFlexGrid1.CellFontBold = True
               count = count + 1
               End If
    Next i
    ProgressBar1.Visible = False 'hide ProgressBar
    If count = 0 Then
            MsgBox " No  data list "
      Else: MsgBox "   Total    " & CStr(count) & "   point this day . "
      End If
End Function
```

```
VERSION 5.00
Object = "{65E121D4-0C60-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCHRT20.OCX"
Begin VB.Form Form2
   Caption         =   "Light  Graph "
   ClientHeight    =   6660
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   10260
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   14.25
      Charset         =   222
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   LinkTopic       =   "Form2"
   ScaleHeight     =   6660
   ScaleWidth      =   10260
   StartUpPosition =   3  'Windows Default
   Begin VB.TextBox Text1
      Height          =   495
      Left            =   2640
      TabIndex        =   6
      Top             =   720
      Width           =   1215
   End
   Begin MSChart20Lib.MSChart MSChart1
      Height          =   9495
      Left            =   120
      OleObjectBlob   =   "Form2.frx":0000
      TabIndex        =   3
      Top             =   1440
      Width           =   15135
   End
   Begin VB.ListBox List1
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   222
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   1020
      Left            =   13080
      TabIndex        =   2
      Top             =   240
      Width           =   1815
   End
   Begin VB.TextBox Text2
      Height          =   495
      Left            =   840
      TabIndex        =   1
      Top             =   720
      Width           =   1215
   End
   Begin VB.Label Label4
      Caption         =   "Time"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   840
      TabIndex        =   7
      Top             =   360
      Width           =   1215
   End
```

```
   Begin VB.Label Label3
      Caption         =   "Type Chart"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   11640
      TabIndex        =   5
      Top             =   240
      Width           =   1455
   End
   Begin VB.Label Label2
      Caption         =   "Reference"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   2640
      TabIndex        =   4
      Top             =   360
      Width           =   1695
   End
   Begin VB.Label Label1
      Caption         =   "Lighting   Graph"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   24
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00400000&
      Height          =   735
      Left            =   5880
      TabIndex        =   0
      Top             =   480
      Width           =   4095
   End
End
Attribute VB_Name = "Form2"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
Private Sub Form_Load()
LoadChartData
FillChartTypeLB
LoadChartData
SetAxes
End Sub

Private Sub SelectChartType()
    Select Case List1.ListIndex
    Case 0: MSChart1.chartType = VtChChartType3dBar
    Case 1: MSChart1.chartType = VtChChartType2dBar
    Case 2: MSChart1.chartType = VtChChartType3dLine
    Case 3: MSChart1.chartType = VtChChartType2dLine
    Case 4: MSChart1.chartType = VtChChartType3dArea
    Case 5: MSChart1.chartType = VtChChartType3dCombination
    Case 6: MSChart1.chartType = VtChChartType2dCombination
    End Select
```

```
    MSChart1.Layout

 End Sub

Private Sub FillChartTypeLB()
    ' Fill chart type.
    With List1
        .AddItem "3D Bar"
        .AddItem "2D Bar"
        .AddItem "3D Line"
        .AddItem "2D Line"
        .AddItem "3D Area"
        .AddItem "3D Combination"
        .AddItem "2D Combination"

        ' Set 2D-line as default.
        .ListIndex = VtChChartType2dLine
        .Selected(.ListIndex) = True
    End With
End Sub

Private Sub List1_Click()
    SelectChartType
End Sub
Private Sub LoadChartData()
Dim SportRate(1 To 288, 1 To 2) As Variant
Dim y%, u%
Dim t%, r%
'  Set the data

    For u% = 1 To 288
                    SportRate(u%, 2) = CVar(Form1.MSFlexGrid1.TextMatrix(u%, 2)) -
    Next u%

    MSChart1.ChartData = SportRate

 ' Set  Axis  x
 With MSChart1
        For t% = 1 To 288
        .row = t%
        r% = t% + 1
        .RowLabel = r%
        Next t%
  End With
End Sub

Private Sub SetAxes()
    Dim p%
    ' Set X-axis title.
    With MSChart1.Plot.Axis(VtChAxisIdX, 1).AxisTitle
        .Visible = True
        .Text = "Time"
        .VtFont.Name = "Arial"
        .VtFont.Size = 15
        .VtFont.Style = VtFontStyleBold
        ' Set text color to Blue.
        .VtFont.VtColor.Set 0, 0, 255
    End With
    ' Set Y-axis title.
    With MSChart1.Plot.Axis(VtChAxisIdY, 1).AxisTitle
        .Visible = True
        .Text = "Lux"
        .VtFont.Name = "Arial"
        .VtFont.Size = 15
        .VtFont.Style = VtFontStyleBold
        ' Set text color to Blue.
        .VtFont.VtColor.Set 0, 0, 255
    End With
    ' Set Row and Column labels.
End Sub
Private Sub mschart1_PointSelected(series As Integer, dataPoint As Integer, _
mouseflag As Integer, cancel As Integer)
Dim i As Long
i = CLng(dataPoint)
Text2.Text = Form1.MSFlexGrid1.TextMatrix(i, 0)
Text1.Text = Form1.MSFlexGrid1.TextMatrix(i, 2)
```

```
End Sub
```

```
VERSION 5.00
Object = "{65E121D4-0C60-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCHRT20.OCX"
Begin VB.Form Form3
   Caption         =   "Tempeture  Graph "
   ClientHeight    =   3195
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   4680
   LinkTopic       =   "Form3"
   ScaleHeight     =   3195
   ScaleWidth      =   4680
   StartUpPosition =   3  'Windows Default
   Begin VB.TextBox Text1
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   2640
      TabIndex        =   6
      Top             =   840
      Width           =   1215
   End
   Begin MSChart20Lib.MSChart MSChart1
      Height          =   9255
      Left            =   0
      OleObjectBlob   =   "Form3.frx":0000
      TabIndex        =   3
      Top             =   1680
      Width           =   14895
   End
   Begin VB.TextBox Text2
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   840
      TabIndex        =   2
      Top             =   840
      Width           =   1215
   End
   Begin VB.ListBox List1
      Height          =   1020
      Left            =   12840
      TabIndex        =   1
      Top             =   240
      Width           =   2055
   End
   Begin VB.Label Label4
      Caption         =   "Time"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   840
      TabIndex        =   7
      Top             =   480
      Width           =   1215
   End
```

```
    Begin VB.Label Label3
       Caption         =   "Type Chart"
       BeginProperty Font
          Name            =   "MS Sans Serif"
          Size            =   12
          Charset         =   222
          Weight          =   700
          Underline       =   0   'False
          Italic          =   0   'False
          Strikethrough   =   0   'False
       EndProperty
       Height          =   735
       Left            =   11280
       TabIndex        =   5
       Top             =   240
       Width           =   1575
    End
    Begin VB.Label Label2
       Caption         =   "Referlence"
       BeginProperty Font
          Name            =   "MS Sans Serif"
          Size            =   12
          Charset         =   222
          Weight          =   700
          Underline       =   0   'False
          Italic          =   0   'False
          Strikethrough   =   0   'False
       EndProperty
       Height          =   615
       Left            =   2640
       TabIndex        =   4
       Top             =   480
       Width           =   1575
    End
    Begin VB.Label Label1
       Caption         =   "Tempeture  Graph "
       BeginProperty Font
          Name            =   "MS Sans Serif"
          Size            =   24
          Charset         =   222
          Weight          =   700
          Underline       =   0   'False
          Italic          =   0   'False
          Strikethrough   =   0   'False
       EndProperty
       Height          =   615
       Left            =   5520
       TabIndex        =   0
       Top             =   480
       Width           =   4575
    End
End
Attribute VB_Name = "Form3"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
Private Sub Form_Load()
LoadChartData
FillChartTypeLB
LoadChartData
SetAxes
End Sub

Private Sub SelectChartType()
    Select Case List1.ListIndex
    Case 0: MSChart1.chartType = VtChChartType3dBar
    Case 1: MSChart1.chartType = VtChChartType2dBar
    Case 2: MSChart1.chartType = VtChChartType3dLine
    Case 3: MSChart1.chartType = VtChChartType2dLine
    Case 4: MSChart1.chartType = VtChChartType3dArea
    Case 5: MSChart1.chartType = VtChChartType3dCombination
    Case 6: MSChart1.chartType = VtChChartType2dCombination
    End Select
    MSChart1.Layout
```

```
 End Sub

Private Sub FillChartTypeLB()
    ' Fill chart type.
    With List1
        .AddItem "3D Bar"
        .AddItem "2D Bar"
        .AddItem "3D Line"
        .AddItem "2D Line"
        .AddItem "3D Area"
        .AddItem "3D Combination"
        .AddItem "2D Combination"

        ' Set 2D-line as default.
        .ListIndex = VtChChartType2dLine
        .Selected(.ListIndex) = True
    End With
End Sub
Private Sub List1_Click()
    SelectChartType
End Sub
Private Sub LoadChartData()
Dim SportRate(1 To 288, 1 To 2) As Variant
Dim y%, u%
Dim t%, r%
'  Set the data

    For u% = 1 To 288
                    SportRate(u%, 1) = CVar(Form1.MSFlexGrid1.TextMatrix(u%, 1)) -
    Next u%

    MSChart1.ChartData = SportRate

 ' Set  Axis  x
 With MSChart1
        For t% = 1 To 288
        .row = t%
        r% = t% + 1
        .RowLabel = r%
        Next t%
  End With
End Sub

Private Sub SetAxes()
    Dim p%

    ' Set X-axis title.
    With MSChart1.Plot.Axis(VtChAxisIdX, 1).AxisTitle
        .Visible = True
        .Text = "Time"
        .VtFont.Name = "Arial"
        .VtFont.Size = 15
        .VtFont.Style = VtFontStyleBold
        ' Set text color to Blue.
        .VtFont.VtColor.Set 0, 0, 255
    End With
    ' Set Y-axis title.
    With MSChart1.Plot.Axis(VtChAxisIdY, 1).AxisTitle
        .Visible = True
        .Text = "CELCIUS"
        .VtFont.Name = "Arial"
        .VtFont.Size = 15
        .VtFont.Style = VtFontStyleBold
        ' Set text color to Blue.
        .VtFont.VtColor.Set 0, 0, 255
    End With
    ' Set Row and Column labels.

End Sub
Private Sub mschart1_PointSelected(series As Integer, dataPoint As Integer, _
mouseflag As Integer, cancel As Integer)
Dim i As Long
i = CLng(dataPoint)
Text2.Text = Form1.MSFlexGrid1.TextMatrix(i, 0)
Text1.Text = Form1.MSFlexGrid1.TextMatrix(i, 1)
```

```
End Sub
```

```
VERSION 5.00
Begin VB.Form Form4
   Caption         =   "Select Record"
   ClientHeight    =   5085
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   9075
   ForeColor       =   &H00000000&
   LinkTopic       =   "Form4"
   ScaleHeight     =   5085
   ScaleWidth      =   9075
   StartUpPosition =   3  'Windows Default
   Begin VB.Timer Timer1
      Enabled         =   0   'False
      Interval        =   500
      Left            =   8640
      Top             =   4680
   End
   Begin VB.CommandButton Command1
      Caption         =   "present"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   -1  'True
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   6720
      TabIndex        =   14
      Top             =   4200
      Width           =   1935
   End
   Begin VB.TextBox Text2
      Alignment       =   2  'Center
      BackColor       =   &H00FFFFFF&
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   5040
      TabIndex        =   13
      Top             =   4200
      Width           =   1215
   End
   Begin VB.TextBox Text1
      Height          =   495
      Left            =   1920
      TabIndex        =   0
      Text            =   "Text1"
      Top             =   360
      Width           =   6495
   End
   Begin VB.Image Image1
      Height          =   690
      Left            =   7800
      Picture         =   "Form4.frx":0000
      Top             =   1560
      Width           =   690
   End
   Begin VB.Shape Shape4
      BorderColor     =   &H80000009&
      Height          =   975
      Left            =   120
      Top             =   3960
      Width           =   8775
   End
   Begin VB.Shape Shape3
```

```
      BorderColor     =   &H80000009&
      FillColor       =   &H00FFFFFF&
      Height          =   2535
      Left            =   120
      Top             =   1320
      Width           =   8775
   End
   Begin VB.Label Label12
      Caption         =   "Which a day do you want to see ?"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00404000&
      Height          =   495
      Left            =   360
      TabIndex        =   12
      Top             =   4200
      Width           =   4215
   End
   Begin VB.Label Label11
      Caption         =   "11"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H000000C0&
      Height          =   495
      Left            =   3120
      TabIndex        =   11
      Top             =   2880
      Width           =   495
   End
   Begin VB.Shape Shape2
      FillStyle       =   0  'Solid
      Height          =   135
      Index           =   1
      Left            =   2760
      Shape           =   3  'Circle
      Top             =   3120
      Width           =   255
   End
   Begin VB.Shape Shape2
      FillStyle       =   0  'Solid
      Height          =   135
      Index           =   0
      Left            =   2760
      Shape           =   3  'Circle
      Top             =   2880
      Width           =   255
   End
   Begin VB.Label Label10
      Caption         =   "10"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H000000C0&
      Height          =   495
      Left            =   2280
      TabIndex        =   10
```

```
      Top             =   2880
      Width           =   375
   End
   Begin VB.Label Label9
      Caption         =   "Strat Time "
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   240
      TabIndex        =   9
      Top             =   2880
      Width           =   1575
   End
   Begin VB.Line Line2
      BorderWidth     =   2
      X1              =   5400
      X2              =   5160
      Y1              =   2160
      Y2              =   2520
   End
   Begin VB.Line Line1
      BorderWidth     =   2
      X1              =   3120
      X2              =   2880
      Y1              =   2160
      Y2              =   2520
   End
   Begin VB.Label Label8
      Caption         =   "2008"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H000000C0&
      Height          =   495
      Left            =   5640
      TabIndex        =   8
      Top             =   2160
      Width           =   1215
   End
   Begin VB.Label Label7
      Alignment       =   2  'Center
      Caption         =   "January7"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H000000C0&
      Height          =   495
      Left            =   3120
      TabIndex        =   7
      Top             =   2160
      Width           =   2055
   End
   Begin VB.Label Label6
      Caption         =   "6"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
```

```
      Charset          =   222
      Weight           =   700
      Underline        =   0   'False
      Italic           =   0   'False
      Strikethrough    =   0   'False
   EndProperty
   ForeColor       =   &H000000C0&
   Height          =   495
   Left            =   2400
   TabIndex        =   6
   Top             =   2160
   Width           =   495
End
Begin VB.Label Label5
   Caption         =   "Start Record"
   BeginProperty Font
      Name             =   "MS Sans Serif"
      Size             =   14.25
      Charset          =   222
      Weight           =   700
      Underline        =   0   'False
      Italic           =   0   'False
      Strikethrough    =   0   'False
   EndProperty
   Height          =   495
   Left            =   240
   TabIndex        =   5
   Top             =   2160
   Width           =   1815
End
Begin VB.Shape Shape1
   DrawMode        =   4  'Mask Not Pen
   Height          =   1095
   Left            =   120
   Top             =   120
   Width           =   8775
End
Begin VB.Label Label4
   Caption         =   "4"
   BeginProperty Font
      Name             =   "MS Sans Serif"
      Size             =   14.25
      Charset          =   222
      Weight           =   700
      Underline        =   0   'False
      Italic           =   0   'False
      Strikethrough    =   0   'False
   EndProperty
   ForeColor       =   &H00008000&
   Height          =   495
   Left            =   2400
   TabIndex        =   4
   Top             =   1440
   Width           =   615
End
Begin VB.Label Label3
   Caption         =   "Day"
   BeginProperty Font
      Name             =   "MS Sans Serif"
      Size             =   14.25
      Charset          =   222
      Weight           =   700
      Underline        =   0   'False
      Italic           =   0   'False
      Strikethrough    =   0   'False
   EndProperty
   Height          =   495
   Left            =   3000
   TabIndex        =   3
   Top             =   1440
   Width           =   1215
End
Begin VB.Label Label2
   Caption         =   " total Record"
   BeginProperty Font
      Name             =   "MS Sans Serif"
```

```
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00000000&
      Height          =   495
      Left            =   240
      TabIndex        =   2
      Top             =   1440
      Width           =   1815
   End
   Begin VB.Label Label1
      Caption         =   "File name"
      BeginProperty Font
         Name            =   "LilyUPC"
         Size            =   21.75
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H00FF0000&
      Height          =   495
      Left            =   240
      TabIndex        =   1
      Top             =   360
      Width           =   1935
   End
End
Attribute VB_Name = "Form4"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub Command1_Click()
Dim Dx  As Integer
Dim h(1) As String
Dim n As Integer
Command1.Caption = " OK "
Dx = 1: Fx = 10: n = 1
h(1) = "Date           "
           Px = CInt(Text2.Text)
               If Px > total Then
               MsgBox " Input more than total "
               Close #1
               Unload Me
               Exit Sub
               Else: End If
Command1.Enabled = False
 If Px = 1 Then
    First_day
    Exit Sub
 Else
    Do
      Get #1, Fx, data
           If h(1) = data.Temp Then
              Dx = Dx + 1
                      If Dx = Px Then
                      Second_Day
                      Exit Sub
                      Else: End If
           Else
               End If
    Fx = Fx + 1
    Loop Until n = 2
End If
End Sub
Private Sub Form_Load()
On Error GoTo noText
Timer1.Enabled = True
Open ifile For Random As #1 Len = Len(data)
```

```
     Get #1, 1, data
     total = CInt(data.Light)

    Get #1, 3, data
           mydate(1) = CInt(data.Temp)
           mymonth(1) = CInt(data.Light)

    Get #1, 5, data
'          myyear(1) = CInt(data.Temp)
           myyear(1) = CInt(data.Light)

    Get #1, 7, data
           myhour(1) = CInt(data.Temp)
           myminute(1) = CInt(data.Light)
'  S how  data date form 4
ShowDate1                    '  Call Functin Show First Time
chkfrm = "file pass"
GoTo line
noText:
         MsgBox " ¢éÍÁÙÅ·Õèà»Ô´ äÁèãªè¢éÍÁÙÅ¢Í§â»Ãá¡ÃÁ"
         Close #1
         chkfrm = "file error"
         Exit Sub
line:
End Sub
Public Function ShowDate1()
Dim ip As String
Text1.Text = ifile
 Label4.Caption = CStr(total)
 Label6.Caption = CStr(mydate(1))
 Label7.Caption = month(mymonth(1))
    If myyear(1) <= 9 Then
      ip = "0" & CStr(myyear(1))
    Else: End If
Label8.Caption = "20" & ip
Label10.Caption = CStr(myhour(1))
Label11.Caption = CStr(myminute(1))
 End Function
Public Function First_day()
Dim i, co, a As Integer
Dim row As Long
row = 1
 For i = 1 To 288
            Form1.MSFlexGrid1.TextMatrix(row, 1) = " 0 "
            row = row + 1
    Next i
    row = 1
    For i = 1 To 288
            Form1.MSFlexGrid1.TextMatrix(row, 2) = " 0 "
            row = row + 1
    Next i

   i = 9
   a = 1
    row = ((myhour(1) * 12) + 1) + (myminute(1) / 5)

  '      Show  this Day
         Form1.Label1.Caption = CStr(mydate(1))
         Form1.Label2.Caption = month(mymonth(1))
         If myyear(Px) <= 9 Then
         op = "0" & CStr(myyear(1))
         Else: End If
         Form1.Label3.Caption = "20" & op
         Form1.Timer1.Enabled = True
    Do
           Get #1, i, data
            Form1.MSFlexGrid1.TextMatrix(row, 1) = data.Temp
            Form1.MSFlexGrid1.TextMatrix(row, 2) = data.Light
                   co = CInt((Form1.MSFlexGrid1.TextMatrix(row, 1)))
                   If co = 0 Then
                           Form1.MSFlexGrid1.TextMatrix(row, 1) = "0"
                           Close #1: Exit Function
                   Else
                       i = i + 1
                       row = row + 1
                           If row > 288 Then
```

```
                                a = 2
                                Else: End If
                        End If
    Loop Until a = 2
Close #1
End Function
Public Function Second_Day()
Dim i, co, a As Integer
Dim row As Long
Dim ta As Integer
Dim op As String

row = 1
For i = 1 To 288
            Form1.MSFlexGrid1.TextMatrix(i, 1) = " 0 "
    row = row + 1
    Next i
 row = 1
    For i = 1 To 288
            Form1.MSFlexGrid1.TextMatrix(i, 2) = " 0 "
   row = row + 1
    Next i

ta = Fx + 1
                        Get #1, ta, data
                        mydate(Px) = CInt(data.Temp)
                        mymonth(Px) = CInt(data.Light)

ta = Fx + 3
                        Get #1, ta, data
                        myyear(Px) = CInt(data.Light)

'       Show  this Day
                        Form1.Label1.Caption = CStr(mydate(Px))
                        Form1.Label2.Caption = month(mymonth(Px))
                        If myyear(Px) <= 9 Then
                        op = "0" & CStr(myyear(Px))
                        Form1.Label3.Caption = "20" & op
                        Else: End If


 '  Insert  Data  on Table Form1.MSFlexgrid1
row = 1
a = 1: ta = Fx + 5
    Do
           Get #1, ta, data
            Form1.MSFlexGrid1.TextMatrix(row, 1) = data.Temp
            Form1.MSFlexGrid1.TextMatrix(row, 2) = data.Light
                    co = CInt((Form1.MSFlexGrid1.TextMatrix(row, 1)))
                    If co = 0 Then
                            Form1.MSFlexGrid1.TextMatrix(row, 1) = "0"
                            Close #1: Exit Function
                    Else
                        ta = ta + 1
                        row = row + 1
                            If row > 288 Then
                            a = 2
                            Else: End If
                    End If
    Loop Until a = 2
Close #1
End Function

Private Sub Timer1_Timer()
Static p
p = p + 1
Select Case p
Case 1
         Text2.BackColor = &H10F801
Case 2
         Text2.BackColor = &H1C0C0
            p = 0
End Select
End Sub
```

```
VERSION 5.00
Begin VB.Form Form5
   BackColor       =   &H8000000B&
   Caption         =   "Set  Time "
   ClientHeight    =   5340
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   6945
   LinkTopic       =   "Form1"
   ScaleHeight     =   5340
   ScaleWidth      =   6945
   StartUpPosition =   3 'Windows Default
   Begin VB.CommandButton Command1
      Caption         =   "Send "
      Height          =   495
      Left            =   720
      TabIndex        =   40
      Top             =   2160
      Width           =   1335
   End
   Begin VB.TextBox Text3
      BackColor       =   &H00FF0000&
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H0000FFFF&
      Height          =   495
      Left            =   4560
      TabIndex        =   39
      Top             =   2280
      Width           =   1455
   End
   Begin VB.CommandButton Command2
      Caption         =   "Start"
      Height          =   495
      Left            =   3240
      TabIndex        =   38
      Top             =   2280
      Width           =   975
   End
   Begin VB.TextBox Text34
      BackColor       =   &H80000004&
      BorderStyle     =   0  'None
      Height          =   375
      Left            =   5640
      TabIndex        =   37
      Text            =   "Second"
      Top             =   3960
      Width           =   735
   End
   Begin VB.TextBox Text33
      BackColor       =   &H80000004&
      BorderStyle     =   0  'None
      Height          =   375
      Left            =   5520
      TabIndex        =   36
      Text            =   "Second"
      Top             =   480
      Width           =   615
   End
   Begin VB.TextBox Text32
      BackColor       =   &H80000004&
      BorderStyle     =   0  'None
      Height          =   375
      Left            =   1680
      TabIndex        =   35
      Text            =   "Month"
      Top             =   480
      Width           =   735
   End
```

```
Begin VB.TextBox Text31
   BackColor       =   &H80000004&
   BorderStyle     =   0  'None
   Height          =   255
   Left            =   720
   TabIndex        =   34
   Text            =   "Date"
   Top             =   480
   Width           =   735
End
Begin VB.CommandButton Command3
   Caption         =   "Quit"
   Height          =   375
   Left            =   720
   TabIndex        =   33
   Top             =   2760
   Width           =   1335
End
Begin VB.TextBox Text30
   Height          =   375
   Left            =   5640
   TabIndex        =   32
   Top             =   4320
   Width           =   735
End
Begin VB.TextBox Text29
   Height          =   372
   Left            =   3240
   TabIndex        =   31
   Top             =   2880
   Width           =   972
End
Begin VB.TextBox Text28
   BackColor       =   &H0000FFFF&
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   9.75
      Charset         =   222
      Weight          =   700
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   ForeColor       =   &H00FF0000&
   Height          =   372
   Left            =   4560
   TabIndex        =   30
   Top             =   2880
   Width           =   1452
End
Begin VB.TextBox Text27
   BackColor       =   &H80000004&
   BorderStyle     =   0  'None
   Height          =   288
   Left            =   4560
   TabIndex        =   28
   Text            =   "Minute"
   Top             =   3960
   Width           =   612
End
Begin VB.TextBox Text26
   BackColor       =   &H80000004&
   BorderStyle     =   0  'None
   Height          =   288
   Left            =   3600
   TabIndex        =   27
   Text            =   "Hour"
   Top             =   3960
   Width           =   612
End
Begin VB.TextBox Text25
   BackColor       =   &H80000004&
   BorderStyle     =   0  'None
   Height          =   288
   Left            =   2520
   TabIndex        =   26
```

```
    Text            =   "Year"
    Top             =   3960
    Width           =   612
End
Begin VB.TextBox Text24
    BackColor       =   &H80000004&
    BorderStyle     =   0  'None
    Height          =   288
    Left            =   1440
    TabIndex        =   25
    Text            =   "Month"
    Top             =   3960
    Width           =   612
End
Begin VB.TextBox Text23
    BackColor       =   &H80000004&
    BorderStyle     =   0  'None
    Height          =   288
    Left            =   480
    TabIndex        =   24
    Text            =   "Date"
    Top             =   3960
    Width           =   612
End
Begin VB.TextBox Text22
    Height          =   372
    Left            =   4560
    TabIndex        =   23
    Top             =   4320
    Width           =   612
End
Begin VB.TextBox Text21
    Height          =   372
    Left            =   3600
    TabIndex        =   22
    Top             =   4320
    Width           =   612
End
Begin VB.TextBox Text20
    Height          =   372
    Left            =   2520
    TabIndex        =   21
    Top             =   4320
    Width           =   612
End
Begin VB.TextBox Text19
    Height          =   372
    Left            =   1440
    TabIndex        =   20
    Top             =   4320
    Width           =   612
End
Begin VB.TextBox Text18
    Height          =   372
    Left            =   480
    TabIndex        =   19
    Top             =   4320
    Width           =   612
End
Begin VB.TextBox Text17
    BackColor       =   &H8000000A&
    BorderStyle     =   0  'None
    Height          =   288
    Left            =   4560
    TabIndex        =   16
    Text            =   "Statrus"
    Top             =   1920
    Width           =   732
End
Begin VB.TextBox Text16
    BackColor       =   &H8000000A&
    BorderStyle     =   0  'None
    Height          =   288
    Left            =   4560
    TabIndex        =   14
    Text            =   "Minute"
```

```
   Top             =   480
   Width           =   612
End
Begin VB.TextBox Text15
   BackColor       =   &H8000000A&
   BorderStyle     =   0  'None
   Height          =   288
   Left            =   3600
   TabIndex        =   13
   Text            =   "Hour"
   Top             =   480
   Width           =   612
End
Begin VB.TextBox Text14
   BackColor       =   &H8000000A&
   BorderStyle     =   0  'None
   Height          =   288
   Left            =   2640
   TabIndex        =   12
   Text            =   "Year"
   Top             =   480
   Width           =   612
End
Begin VB.TextBox Text13
   Alignment       =   2  'Center
   Height          =   405
   Left            =   5760
   TabIndex        =   11
   Top             =   840
   Width           =   255
End
Begin VB.TextBox Text12
   Height          =   405
   Left            =   5520
   TabIndex        =   10
   Top             =   840
   Width           =   255
End
Begin VB.TextBox Text11
   Height          =   372
   Left            =   4800
   TabIndex        =   9
   Top             =   840
   Width           =   252
End
Begin VB.TextBox Text10
   Height          =   372
   Left            =   4560
   TabIndex        =   8
   Top             =   840
   Width           =   252
End
Begin VB.TextBox Text9
   Height          =   372
   Left            =   3840
   TabIndex        =   7
   Top             =   840
   Width           =   252
End
Begin VB.TextBox Text8
   Height          =   372
   Left            =   3600
   TabIndex        =   6
   Top             =   840
   Width           =   252
End
Begin VB.TextBox Text7
   Height          =   372
   Left            =   2880
   TabIndex        =   5
   Top             =   840
   Width           =   252
End
Begin VB.TextBox Text6
   Alignment       =   2  'Center
   Height          =   372
```

```
      Left            =   2640
      TabIndex        =   4
      Top             =   840
      Width           =   252
   End
   Begin VB.TextBox Text5
      Height          =   372
      Left            =   1920
      TabIndex        =   3
      Top             =   840
      Width           =   252
   End
   Begin VB.TextBox Text4
      Height          =   372
      Left            =   1680
      TabIndex        =   2
      Top             =   840
      Width           =   252
   End
   Begin VB.TextBox Text2
      Height          =   372
      Left            =   960
      TabIndex        =   1
      Top             =   840
      Width           =   252
   End
   Begin VB.TextBox Text1
      Height          =   372
      Left            =   720
      TabIndex        =   0
      Top             =   840
      Width           =   252
   End
   Begin VB.Frame Frame1
      Caption         =   "Set Time"
      Height          =   1452
      Left            =   240
      TabIndex        =   15
      Top             =   120
      Width           =   6495
   End
   Begin VB.Frame Frame2
      Caption         =   "Initial "
      Height          =   1692
      Left            =   2760
      TabIndex        =   17
      Top             =   1680
      Width           =   3975
   End
   Begin VB.Frame Frame3
      Caption         =   "Send time"
      Height          =   1692
      Left            =   240
      TabIndex        =   18
      Top             =   1680
      Width           =   2292
   End
   Begin VB.Frame Frame4
      Caption         =   "Form hardwear"
      Height          =   1572
      Left            =   240
      TabIndex        =   29
      Top             =   3480
      Width           =   6495
   End
End
Attribute VB_Name = "Form5"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub Command1_Click()
Dim a, b, c As Integer    ' for date
Dim d, e, f As Integer    ' for month
Dim g, h, i As Integer  ' for century
```

```
Dim j, k, l  As Integer   ' for hour
Dim m, n, o As Integer    ' for minute
Dim p, q, r As Integer

With Form1.BCComm
        .InputLen = 1
        .PortOpen = True
        .Settings = "9600,N,8,1"
 End With
'On Error Resume Next
a = CInt(Text1.Text)
b = CInt(Text2.Text)

' LOOP DATE
Select Case a
   Case 0: c = b
           If c = 0 Then
           MsgBox "invalid date": Form1.BCComm.PortOpen = False: Exit Sub
           Else: End If
   Case 1: c = 16 + b
   Case 2: c = 32 + b
   Case 3: c = 48 + b
           If b > 2 Then
           MsgBox "invalid date": Form1.BCComm.PortOpen = False: Exit Sub
           Else: End If
   Case Else: MsgBox " invalid date": Form1.BCComm.PortOpen = False: Exit Sub
End Select

d = CInt(Text4.Text)
e = CInt(Text5.Text)
'LOOP MONTH
Select Case d
   Case 0: f = e
           If f = 0 Then
           MsgBox "invalid month": Form1.BCComm.PortOpen = False: Exit Sub
           Else: End If
   Case 1: f = 16 + e
           If e > 2 Then
           MsgBox "invalid month": Form1.BCComm.PortOpen = False: Exit Sub
           Else: End If
   Case Else: MsgBox " invalid month": Form1.BCComm.PortOpen = False: Exit Sub
End Select


j = CInt(Text8.Text)
k = CInt(Text9.Text)
'LOOP Hour
Select Case j
   Case 0: l = k
   Case 1: l = 16 + k
   Case 2: l = 32 + k
           If k > 3 Then
           MsgBox "invalid hour":  Form1.BCComm.PortOpen = False: Exit Sub
           Else: End If
   Case Else: MsgBox " invalid hour": Form1.BCComm.PortOpen = False: Exit Sub
End Select


m = CInt(Text10.Text)
n = CInt(Text11.Text)
'LOOP MINUTE
Select Case m
   Case 0: o = n
   Case 1: o = 16 + n
   Case 2: o = 32 + n
   Case 3: o = 48 + n
   Case 4: o = 64 + n
   Case 5: o = 80 + n
   Case Else: MsgBox " invalid time": Form1.BCComm.PortOpen = False: Exit Sub
End Select

p = CInt(Text12.Text)
q = CInt(Text13.Text)
'LOOP Second
Select Case p
   Case 0: r = q
```

```
    Case 1: r = 16 + q
    Case 2: r = 32 + q
    Case 3: r = 48 + q
    Case 4: r = 64 + q
    Case 5: r = 80 + q
    Case Else: MsgBox " invalid time": Form1.BCComm.PortOpen = False: Exit Sub
End Select


g = CInt(Text6.Text)
h = CInt(Text7.Text)
'LOOP CENTURY
Select Case g
    Case 0: i = h
    Case 1: i = 16 + h
    Case 2: i = 32 + h
    Case 3: i = 48 + h
    Case 4: i = 64 + h
    Case 5: i = 80 + h
    Case 6: i = 96 + h
    Case 7: i = 112 + h
    Case 8: i = 128 + h
    Case 9: i = 144 + h
   Case Else: MsgBox " invalid century"
End Select


Form1.BCComm.Output = Chr(83)  '

Form1.BCComm.Output = Chr(c)    ' SAND DATE
Form1.BCComm.Output = Chr(f)    ' SAND MONTH
Form1.BCComm.Output = Chr(i)    ' SAND CENTURY
Form1.BCComm.Output = Chr(l)    ' SAND MONTH
Form1.BCComm.Output = Chr(o)    ' SAND MINUTE
Form1.BCComm.Output = Chr(r)    ' SAND Second

FormHardware              ' CALL  FUNCTION
line:
Form1.BCComm.PortOpen = False  ' off port againEnd Sub
End Sub

Private Sub Command2_Click()
 With Form1.BCComm
        .InputLen = 1
        .PortOpen = True
        .Settings = "9600,N,8,1"
 End With
 On Error Resume Next
 Form1.BCComm.Output = Chr(84)  ' send char 'T'
 a = 0
 Do
 If Form1.BCComm.InBufferCount >= 1 Then
           Do
           indata = Asc(Form1.BCComm.Input)
           If indata = 84 Then:
           Text3.Text = "  Ready "
           Text28.Text = Chr(indata)
           a = 1

                If Chr(84) <> Chr(indata) Then
                    Form1.BCComm.PortOpen = False
                    Text3.Text = "NON"
                    Text28.Text = "NON"
                    MsgBox "                    ERROR !                    "
                    + Chr(10) + Chr(13) + " CHEK AGAIN  AND START BOTTON"
                    a = 1
                    Text3.Text = "     "
                    Text28.Text = "    "
                    Exit Sub
                 End If
           FormHardware
           Loop Until a = 1
  Else: a = 0
 End If
 Loop Until a = 1
Form1.BCComm.PortOpen = False      ' off port again
```

```
 Command1.Enabled = True
End Sub

Private Sub Command3_Click()
With Form1.BCComm
        .InputLen = 1
        .PortOpen = True
        .Settings = "9600,N,8,1"
 End With
 On Error Resume Next
 Form1.BCComm.Output = Chr(81)   ' send char 'Q'
 Form1.BCComm.PortOpen = False
Unload Me
End Sub

Private Sub Form_Load()
  Form1.BCComm.CommPort = CInt(port)
  Command1.Enabled = False
End Sub

Private Sub Text1_Change()
If Text1.Text = "" Then: Exit Sub
Text1.Text = con(Asc(Text1.Text))
End Sub

Private Sub Text10_Change()
If Text10.Text = "" Then: Exit Sub
Text10.Text = con(Asc(Text10.Text))
End Sub

Private Sub Text11_Change()
If Text11.Text = "" Then: Exit Sub
Text11.Text = con(Asc(Text11.Text))
End Sub

Private Sub Text12_Change()
If Text12.Text = "" Then: Exit Sub
Text12.Text = con(Asc(Text12.Text))
End Sub

Private Sub Text13_Change()
If Text13.Text = "" Then: Exit Sub
Text13.Text = con(Asc(Text13.Text))

End Sub

Private Sub Text2_Change()
If Text2.Text = "" Then: Exit Sub
Text2.Text = con(Asc(Text2.Text))
End Sub



Private Sub Text4_Change()
If Text4.Text = "" Then: Exit Sub
Text4.Text = con(Asc(Text4.Text))
End Sub

Private Sub Text5_Change()
If Text5.Text = "" Then: Exit Sub
Text5.Text = con(Asc(Text5.Text))
End Sub

Private Sub Text6_Change()
If Text6.Text = "" Then: Exit Sub
Text6.Text = con(Asc(Text6.Text))
End Sub

Private Sub Text7_Change()
If Text7.Text = "" Then: Exit Sub
Text7.Text = con(Asc(Text7.Text))
End Sub

Private Sub Text8_Change()
If Text8.Text = "" Then: Exit Sub
```

```
Text8.Text = con(Asc(Text8.Text))
End Sub

Private Sub Text9_Change()
If Text9.Text = "" Then: Exit Sub
Text9.Text = con(Asc(Text9.Text))
End Sub
Public Function con(a As Integer) As String
Dim check As Integer
Dim error, b As Integer
Dim str As String
b = 0
check = 1
Do
If (a < 48 Or a > 57) Then
    Do
    MsgBox "Error insert.You must insert again"
    check = 1: b = 1
    Loop Until check = 1
End If
Loop Until check = 1
If b = 1 Then
 a = 45
 con = Chr(a)
 Exit Function
End If
con = Chr(a)
End Function

Private Sub FormHardware()
Dim time(5) As Integer
Dim a As Integer
 a = 0
 Do
 If Form1.BCComm.InBufferCount >= 1 Then
           For i = 0 To 5
           time(i) = Asc(Form1.BCComm.Input)
           Next i

            Text18.Text = CStr(Hex(time(0)))
            Text19.Text = CStr(Hex(time(1)))
            Text20.Text = CStr(Hex(time(2)))
            Text21.Text = CStr(Hex(time(3)))
            Text22.Text = CStr(Hex(time(4)))
            Text30.Text = CStr(Hex(time(5)))
           a = 1
 Else: a = 0
 End If
 Loop Until a = 1
 MsgBox " Send time complete "
 Form1.BCComm.PortOpen = False     ' off port again
 End Sub
```

```
VERSION 5.00
Object = "{65E121D4-0C60-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCHRT20.OCX"
Begin VB.Form Form6
   Caption         =   "Light Graph "
   ClientHeight    =   9120
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   13350
   LinkTopic       =   "Form6"
   ScaleHeight     =   9120
   ScaleWidth      =   13350
   StartUpPosition =   3 'Windows Default
   Visible         =   0   'False
   Begin VB.TextBox Text1
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   615
      Left            =   2880
      TabIndex        =   6
      Top             =   720
      Visible         =   0   'False
      Width           =   1815
   End
   Begin VB.TextBox Text2
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   615
      Left            =   720
      TabIndex        =   5
      Top             =   720
      Visible         =   0   'False
      Width           =   1815
   End
   Begin MSChart20Lib.MSChart MSChart1
      Height          =   7455
      Left            =   480
      OleObjectBlob   =   "Form6.frx":0000
      TabIndex        =   4
      Top             =   1560
      Width           =   12735
   End
   Begin VB.ListBox List1
      Height          =   780
      Left            =   11520
      TabIndex        =   3
      Top             =   360
      Width           =   1215
   End
   Begin VB.Label Label4
      Caption         =   "Reference"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   2880
      TabIndex        =   7
```

```
      Top             =   360
      Visible         =   0   'False
      Width           =   1575
   End
   Begin VB.Label Label3
      Caption         =   "Type  Chart "
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   9.75
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   11520
      TabIndex        =   2
      Top             =   0
      Width           =   1215
   End
   Begin VB.Label Label2
      Caption         =   "Time"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   720
      TabIndex        =   1
      Top             =   360
      Visible         =   0   'False
      Width           =   1695
   End
   Begin VB.Label Label1
      BorderStyle     =   1  'Fixed Single
      Caption         =   "Lighting    Graph"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   24
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   975
      Left            =   5400
      TabIndex        =   0
      Top             =   360
      Width           =   4215
   End
End
Attribute VB_Name = "Form6"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub Form_Load()
LoadChartData
FillChartTypeLB
LoadChartData
SetAxes
End Sub

Private Sub SelectChartType()
    Select Case List1.ListIndex
    Case 0: MSChart1.chartType = VtChChartType3dBar
    Case 1: MSChart1.chartType = VtChChartType2dBar
    Case 2: MSChart1.chartType = VtChChartType3dLine
```

```
    Case 3: MSChart1.chartType = VtChChartType2dLine
    Case 4: MSChart1.chartType = VtChChartType3dArea
    Case 5: MSChart1.chartType = VtChChartType3dCombination
    Case 6: MSChart1.chartType = VtChChartType2dCombination
    End Select
    MSChart1.Layout

 End Sub

Private Sub FillChartTypeLB()
    ' Fill chart type.
    With List1
        .AddItem "3D Bar"
        .AddItem "2D Bar"
        .AddItem "3D Line"
        .AddItem "2D Line"
        .AddItem "3D Area"
        .AddItem "3D Combination"
        .AddItem "2D Combination"

        ' Set 2D-line as default.
        .ListIndex = VtChChartType2dLine
        .Selected(.ListIndex) = True
    End With
End Sub

Private Sub List1_Click()
    SelectChartType
End Sub
Private Sub LoadChartData()
Dim SportRate(1 To 20, 1 To 2) As Variant
Dim y%, u%
Dim t%, r%
'  Set the data
    y% = 0
    For u% = 1 To 280 Step 14
                    y% = y% + 1
                    SportRate(y%, 2) = CVar(Form1.MSFlexGrid1.TextMatrix(u%, 2)) -
    Next u%

    MSChart1.ChartData = SportRate

 ' Set  Axis  x
 ' Set  Axis  x
 With MSChart1
        t = 0
        For w = 1 To 280 Step 14
        y = CLng(w)
        t = t + 1
       .row = t
        .RowLabel = Form1.MSFlexGrid1.TextMatrix(y%, 0)
        Next w
  End With
End Sub

Private Sub SetAxes()
    Dim p%
    ' Set X-axis title.
    With MSChart1.Plot.Axis(VtChAxisIdX, 1).AxisTitle
        .Visible = True
        .Text = "Time"
        .VtFont.Name = "Arial"
        .VtFont.Size = 15
        .VtFont.Style = VtFontStyleBold
        ' Set text color to Blue.
        .VtFont.VtColor.Set 0, 0, 255
    End With
    ' Set Y-axis title.
    With MSChart1.Plot.Axis(VtChAxisIdY, 1).AxisTitle
        .Visible = True
        .Text = "Lux"
        .VtFont.Name = "Arial"
        .VtFont.Size = 15
        .VtFont.Style = VtFontStyleBold
        ' Set text color to Blue.
        .VtFont.VtColor.Set 0, 0, 255
```

```
    End With
    ' Set Row and Column labels.
End Sub
Private Sub mschart1_PointSelected(series As Integer, dataPoint As Integer, _
mouseflag As Integer, cancel As Integer)
Dim i As Long
i = CLng(dataPoint)
Text2.Text = Form1.MSFlexGrid1.TextMatrix(i, 0)
Text1.Text = Form1.MSFlexGrid1.TextMatrix(i, 2)
End Sub
```

```
VERSION 5.00
Object = "{65E121D4-0C60-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCHRT20.OCX"
Begin VB.Form Form7
   Caption         =   "Tempeture Graph"
   ClientHeight    =   8670
   ClientLeft      =   60
   ClientTop       =   345
   ClientWidth     =   13605
   LinkTopic       =   "Form7"
   ScaleHeight     =   8670
   ScaleWidth      =   13605
   StartUpPosition =   3  'Windows Default
   Begin VB.TextBox Text1
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   2280
      TabIndex        =   6
      Top             =   1080
      Visible         =   0   'False
      Width           =   1215
   End
   Begin MSChart20Lib.MSChart MSChart1
      Height          =   6495
      Left            =   360
      OleObjectBlob   =   "Form7.frx":0000
      TabIndex        =   5
      Top             =   1920
      Width           =   12975
   End
   Begin VB.ListBox List1
      Height          =   1020
      Left            =   11160
      TabIndex        =   4
      Top             =   480
      Width           =   1815
   End
   Begin VB.TextBox Text2
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   14.25
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   720
      TabIndex        =   3
      Top             =   1080
      Visible         =   0   'False
      Width           =   1215
   End
   Begin VB.Label Label4
      Caption         =   "Time"
      BeginProperty Font
         Name            =   "MS Sans Serif"
         Size            =   12
         Charset         =   222
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      Height          =   495
      Left            =   720
      TabIndex        =   7
      Top             =   720
```

```
      Visible          =   0   'False
      Width            =   1215
   End
   Begin VB.Label Label3
      Caption          =   "Type Chart"
      BeginProperty Font
         Name             =   "MS Sans Serif"
         Size             =   12
         Charset          =   222
         Weight           =   700
         Underline        =   0   'False
         Italic           =   0   'False
         Strikethrough    =   0   'False
      EndProperty
      Height           =   495
      Left             =   11160
      TabIndex         =   2
      Top              =   120
      Width            =   1575
   End
   Begin VB.Label Label2
      Caption          =   "Reference"
      BeginProperty Font
         Name             =   "MS Sans Serif"
         Size             =   12
         Charset          =   222
         Weight           =   700
         Underline        =   0   'False
         Italic           =   0   'False
         Strikethrough    =   0   'False
      EndProperty
      Height           =   495
      Left             =   2280
      TabIndex         =   1
      Top              =   720
      Visible          =   0   'False
      Width            =   1335
   End
   Begin VB.Label Label1
      BorderStyle      =   1  'Fixed Single
      Caption          =   "Tempeture     G raph "
      BeginProperty Font
         Name             =   "MS Sans Serif"
         Size             =   24
         Charset          =   222
         Weight           =   700
         Underline        =   0   'False
         Italic           =   0   'False
         Strikethrough    =   0   'False
      EndProperty
      Height           =   975
      Left             =   4440
      TabIndex         =   0
      Top              =   360
      Width            =   5655
   End
End
Attribute VB_Name = "Form7"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub Form_Load()
LoadChartData
FillChartTypeLB
LoadChartData
SetAxes
End Sub

Private Sub SelectChartType()
    Select Case List1.ListIndex
    Case 0: MSChart1.chartType = VtChChartType3dBar
    Case 1: MSChart1.chartType = VtChChartType2dBar
    Case 2: MSChart1.chartType = VtChChartType3dLine
    Case 3: MSChart1.chartType = VtChChartType2dLine
```

```
    Case 4: MSChart1.chartType = VtChChartType3dArea
    Case 5: MSChart1.chartType = VtChChartType3dCombination
    Case 6: MSChart1.chartType = VtChChartType2dCombination
    End Select
    MSChart1.Layout

 End Sub

Private Sub FillChartTypeLB()
    ' Fill chart type.
    With List1
        .AddItem "3D Bar"
        .AddItem "2D Bar"
        .AddItem "3D Line"
        .AddItem "2D Line"
        .AddItem "3D Area"
        .AddItem "3D Combination"
        .AddItem "2D Combination"

        ' Set 2D-line as default.
        .ListIndex = VtChChartType2dLine
        .Selected(.ListIndex) = True
    End With
End Sub
Private Sub List1_Click()
    SelectChartType
End Sub
Private Sub LoadChartData()
Dim SportRate(1 To 20, 1 To 2) As Variant
Dim y%, u%
Dim t%, r%
'  Set the data
   y% = 0
    For u% = 1 To 280 Step 14
                    y% = y% + 1
                    SportRate(y%, 1) = CVar(Form1.MSFlexGrid1.TextMatrix(u%, 1)) -
    Next u%

    MSChart1.ChartData = SportRate

 ' Set  Axis  x
 With MSChart1
        t = 0
        For w = 1 To 280 Step 14
        y = CLng(w)
        t = t + 1
       .row = t
        .RowLabel = Form1.MSFlexGrid1.TextMatrix(y%, 0)
        Next w
  End With
End Sub

Private Sub SetAxes()
    Dim p%
    ' Set X-axis title.
    With MSChart1.Plot.Axis(VtChAxisIdX, 1).AxisTitle
        .Visible = True
        .Text = "Time"
        .VtFont.Name = "Arial"
        .VtFont.Size = 15
        .VtFont.Style = VtFontStyleBold
        ' Set text color to Blue.
        .VtFont.VtColor.Set 0, 0, 255
    End With
    ' Set Y-axis title.
    With MSChart1.Plot.Axis(VtChAxisIdY, 1).AxisTitle
        .Visible = True
        .Text = "CELCIUS"
        .VtFont.Name = "Arial"
        .VtFont.Size = 15
        .VtFont.Style = VtFontStyleBold
        ' Set text color to Blue.
        .VtFont.VtColor.Set 0, 0, 255
    End With
    ' Set Row and Column labels.
```

```
End Sub
Private Sub mschart1_PointSelected(series As Integer, dataPoint As Integer, _
mouseflag As Integer, cancel As Integer)
Dim i As Long
i = CLng(dataPoint)
Text2.Text = Form1.MSFlexGrid1.TextMatrix(i, 0)
Text1.Text = Form1.MSFlexGrid1.TextMatrix(i, 1)
End Sub
```

```
Attribute VB_Name = "Module1"
Public chkfrm As String
Public mydate(10) As Integer
Public mymonth(10) As Integer
Public myyear(10) As Integer
Public myhour(10) As Integer
Public myminute(10) As Integer

Public ifile As String
Public data As RECORD
Public Ax(5) As Integer
Public port As String
Public total As Integer
Public Fx As Integer
Public Px As Integer

Type RECORD
     Temp  As String * 15
     Light As String * 15
     crlf As String * 2
End Type
Public Function month(t As Integer) As String
Select Case t
       Case 1: month = "  JANUARY"
       Case 2: month = "  FABRUARY"
       Case 3: month = "  MARCH"
       Case 4: month = "  APRIL"
       Case 5: month = "  MAY"
       Case 6: month = "  JUNE"
       Case 7: month = "  JULY"
       Case 8: month = "  AUGUST"
       Case 9: month = "  SATEMBER"
       Case 10: month = "  OCTORBER"
       Case 11: month = "  NOVEMBER"
       Case 12: month = " DECEMBER"
       Case Else
       month = " NO...NO"
End Select
End Function

Public Function ERR()
MsgBox "  ªéÍÁÙÅ·Õè·èÒ¹à»Ô´  äÁèàËÁÒÐÊÁ¡Ñºâ»Ãá¡ÃÁ "
Close #1
End Function
```

Data Sheet

Features

- Compatible with MCS-51™ Products
- 8K Bytes of In-System Reprogrammable Downloadable Flash Memory
  - SPI Serial Interface for Program Downloading
  - Endurance: 1,000 Write/Erase Cycles
- 2K Bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
- 4V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Nine Interrupt Sources
- Programmable UART Serial Channel
- SPI Serial Interface
- Low-power Idle and Power-down Modes
- Interrupt Recovery From Power-down
- Programmable Watchdog Timer
- Dual Data Pointer
- Power-off Flag

Description

The AT89S8252 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of downloadable Flash programmable and erasable read only memory and 2K bytes of EEPROM. The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 instruction set and pinout. The on-chip downloadable Flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with downloadable Flash on a monolithic chip, the Atmel AT89S8252 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89S8252 provides the following standard features: 8K bytes of downloadable Flash, 2K bytes of EEPROM, 256 bytes of RAM, 32 I/O lines, programmable watchdog timer, two data pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89S8252 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

The downloadable Flash can be changed a single byte at a time and is accessible through the SPI serial interface. Holding RESET active forces the SPI bus into a serial programming interface and allows the program memory to be written to or read from unless Lock Bit 2 has been activated.

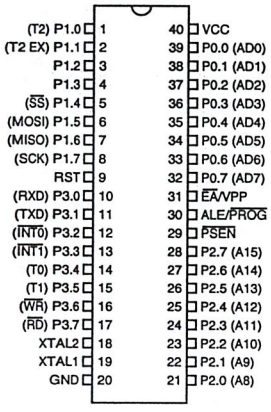
8-bit Microcontroller with 8K Bytes Flash

AT89S8252

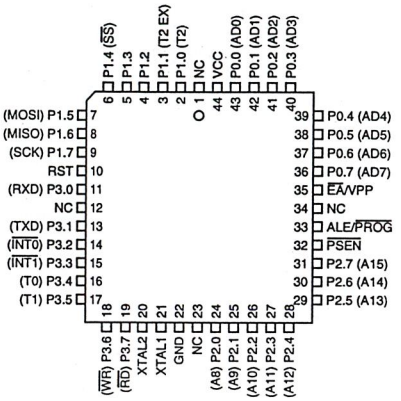
Rev. 0401E–02/00

Pin Configurations

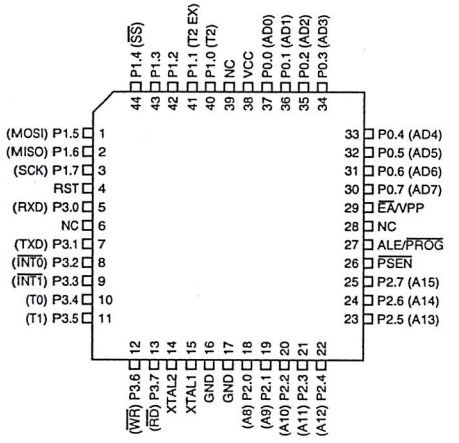
PDIP



PLCC



PQFP/TQFP



Pin Description

VCC

Supply voltage.

GND

Ground.

Port 0

Port 0 is an 8-bit open drain bbi-didirectional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

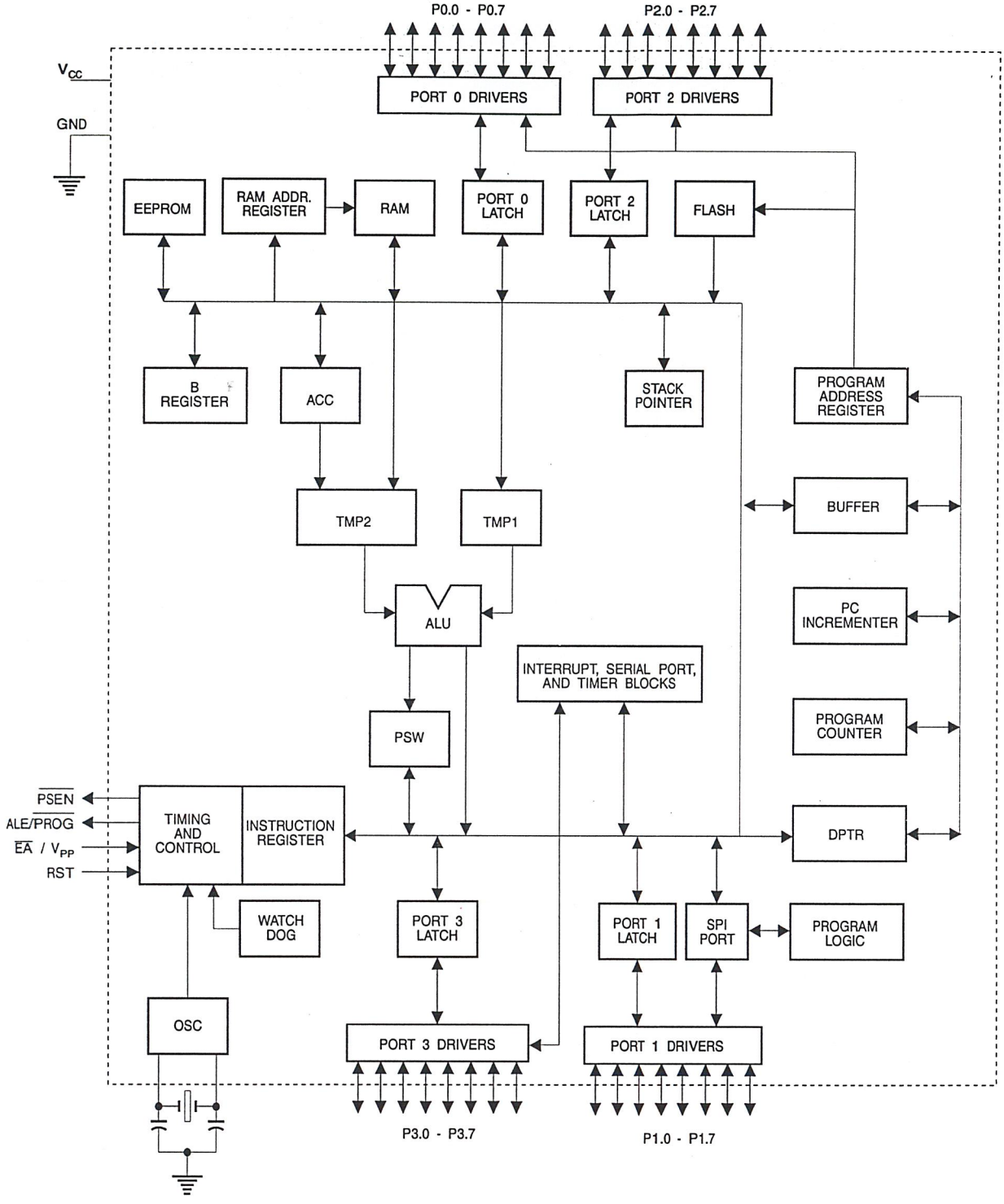
Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1

Port 1 is an 8-bit bi-directional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Block Diagram



Some Port 1 pins provide additional functions. P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively.

Pin Description

Furthermore, P1.4, P1.5, P1.6, and P1.7 can be configured as the SPI slave port select, data input/output and shift clock input/output pins as shown in the following table.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.4	$\overline{SS}$ (Slave port select input)
P1.5	MOSI (Master data output, slave data input pin for SPI channel)
P1.6	MISO (Master data input, slave data output pin for SPI channel)
P1.7	SCK (Master clock output, slave clock input pin for SPI channel)

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port 2

Port 2 is an 8-bit bi-directional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8 bit bi-directional I/O port with internal pullups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89S8252, as shown in the following table.

Port 3 also receives some control signals for Flash programming and verification.

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

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$

Address Latch Enable is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{PSEN}$

Program Store Enable is the read strobe to external program memory.

When the AT89S8252 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{EA}$/VPP

External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external pro-

gram memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions. This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming when 12-volt programming is selected.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

Table 1. AT89S8252 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000					SPCR 000001XX			0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000		SPSR 00XXXXXX						0AFH
0A0H	P2 11111111								0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111						WMCON 00000010		97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000	SPDR XXXXXXXX	PCON 0XXX0000	87H

Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 1.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Timer 2 Registers Control and status bits are contained in registers T2CON (shown in Table 2) and T2MOD (shown in Table 9) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16 bit capture mode or 16-bit auto-reload mode.

Table 2. T2CON—Timer/Counter 2 Control Register

T2CON Address = 0C8H — Reset Value = 0000 0000B

Bit Addressable

Bit	7	6	5	4	3	2	1	0
	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

Symbol	Function
TF2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
RCLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflows to be used for the receive clock.
TCLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
C/$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
CP/$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

Watchdog and Memory Control Register The WMCON register contains control bits for the Watchdog Timer (shown in Table 3). The EEMEN and EEMWE bits are used to select the 2K bytes on-chip EEPROM, and to enable byte-write. The DPS bit selects one of two DPTR registers available.

Table 3. WMCON—Watchdog and Memory Control Register

WMCON Address = 96H — Reset Value = 0000 0010B

Bit	PS2	PS1	PS0	EEMWE	EEMEN	DPS	WDTRST	WDTEN
	7	6	5	4	3	2	1	0

Symbol	Function
PS2 PS1 PS0	Prescaler Bits for the Watchdog Timer. When all three bits are set to "0", the watchdog timer has a nominal period of 16 ms. When all three bits are set to "1", the nominal period is 2048 ms.
EEMWE	EEPROM Data Memory Write Enable Bit. Set this bit to "1" before initiating byte write to on-chip EEPROM with the MOVX instruction. User software should set this bit to "0" after EEPROM write is completed.
EEMEN	Internal EEPROM Access Enable. When EEMEN = 1, the MOVX instruction with DPTR will access on-chip EEPROM instead of external data memory. When EEMEN = 0, MOVX with DPTR accesses external data memory.
DPS	Data Pointer Register Select. DPS = 0 selects the first bank of Data Pointer Register, DP0, and DPS = 1 selects the second bank, DP1
WDTRST RDY/$\overline{BSY}$	Watchdog Timer Reset and EEPROM Ready/$\overline{Busy}$ Flag. Each time this bit is set to "1" by user software, a pulse is generated to reset the watchdog timer. The WDTRST bit is then automatically reset to "0" in the next instruction cycle. The WDTRST bit is Write-Only. This bit also serves as the RDY/$\overline{BSY}$ flag in a Read-Only mode during EEPROM write. RDY/$\overline{BSY}$ = 1 means that the EEPROM is ready to be programmed. While programming operations are being executed, the RDY/$\overline{BSY}$ bit equals "0" and is automatically reset to "1" when programming is completed.
WDTEN	Watchdog Timer Enable Bit. WDTEN = 1 enables the watchdog timer and WDTEN = 0 disables the watchdog timer.

SPI Registers Control and status bits for the Serial Peripheral Interface are contained in registers SPCR (shown in Table 4) and SPSR (shown in Table 5). The SPI data bits are contained in the SPDR register. Writing the SPI data register during serial data transfer sets the Write Collision bit, WCOL, in the SPSR register. The SPDR is double buffered for writing and the values in SPDR are not changed by Reset.

Interrupt Registers The global interrupt enable bit and the individual interrupt enable bits are in the IE register. In addition, the individual interrupt enable bit for the SPI is in the SPCR register. Two priorities can be set for each of the six interrupt sources in the IP register.

Dual Data Pointer Registers To facilitate accessing both internal EEPROM and external data memory, two banks of 16 bit Data Pointer Registers are provided: DP0 at SFR address locations 82H-83H and DP1 at 84H-85H. Bit DPS = 0 in SFR WMCON selects DP0 and DPS = 1 selects DP1. The user should always initialize the DPS bit to the appropriate value before accessing the respective Data Pointer Register.

Power Off Flag The Power Off Flag (POF) is located at bit_4 (PCON.4) in the PCON SFR. POF is set to "1" during power up. It can be set and reset under software control and is not affected by RESET.

Table 4. SPCR—SPI Control Register

SPCR Address = D5H — Reset Value = 0000 01XXB

	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Bit	7	6	5	4	3	2	1	0

Symbol	Function
SPIE	SPI Interrupt Enable. This bit, in conjunction with the ES bit in the IE register, enables SPI interrupts: SPIE = 1 and ES = 1 enable SPI interrupts. SPIE = 0 disables SPI interrupts.
SPE	SPI Enable. SPI = 1 enables the SPI channel and connects $\overline{SS}$, MOSI, MISO and SCK to pins P1.4, P1.5, P1.6, and P1.7. SPI = 0 disables the SPI channel.
DORD	Data Order. DORD = 1 selects LSB first data transmission. DORD = 0 selects MSB first data transmission.
MSTR	Master/Slave Select. MSTR = 1 selects Master SPI mode. MSTR = 0 selects Slave SPI mode.
CPOL	Clock Polarity. When CPOL = 1, SCK is high when idle. When CPOL = 0, SCK of the master device is low when not transmitting. Please refer to figure on SPI Clock Phase and Polarity Control.
CPHA	Clock Phase. The CPHA bit together with the CPOL bit controls the clock and data relationship between master and slave. Please refer to figure on SPI Clock Phase and Polarity Control.
SPR0 SPR1	SPI Clock Rate Select. These two bits control the SCK rate of the device configured as master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the oscillator frequency, $F_{osc.}$, is as follows: SPR1 SPR0 SCK = $F_{osc.}$ divided by 0 0 4 0 1 16 1 0 64 1 1 128

Table 5. SPSR – SPI Status Register

SPSR Address = AAH — Reset Value = 00XX XXXXB

	SPIF	WCOL	–	–	–	–	–	–
Bit	7	6	5	4	3	2	1	0

Symbol	Function
SPIF	SPI Interrupt Flag. When a serial transfer is complete, the SPIF bit is set and an interrupt is generated if SPIE = 1 and ES = 1. The SPIF bit is cleared by reading the SPI status register with SPIF and WCOL bits set, and then accessing the SPI data register.
WCOL	Write Collision Flag. The WCOL bit is set if the SPI data register is written during a data transfer. During data transfer, the result of reading the SPDR register may be incorrect, and writing to it has no effect. The WCOL bit (and the SPIF bit) are cleared by reading the SPI status register with SPIF and WCOL set, and then accessing the SPI data register.

Table 6. SPDR – SPI Data Register

SPDR Address = 86H — Reset Value = unchanged

	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0
Bit	7	6	5	4	3	2	1	0

Data Memory – EEPROM and RAM

The AT89S8252 implements 2K bytes of on-chip EEPROM for data storage and 256 bytes of RAM. The upper 128 bytes of RAM occupy a parallel space to the Special Function Registers. That means the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the address mode used in the instruction specifies whether the CPU accesses the upper 128 bytes of RAM or the SFR space. Instructions that use direct addressing access SFR space.

For example, the following direct addressing instruction accesses the SFR at location 0A0H (which is P2).

```
MOV 0A0H, #data
```

Instructions that use indirect addressing access the upper 128 bytes of RAM. For example, the following indirect addressing instruction, where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

```
MOV @R0, #data
```

Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

The on-chip EEPROM data memory is selected by setting the EEMEN bit in the WMCON register at SFR address location 96H. The EEPROM address range is from 000H to 7FFH. The MOVX instructions are used to access the EEPROM. To access off-chip data memory with the MOVX instructions, the EEMEN bit needs to be set to "0".

The EEMWE bit in the WMCON register needs to be set to "1" before any byte location in the EEPROM can be written. User software should reset EEMWE bit to "0" if no further EEPROM write is required. EEPROM write cycles in the serial programming mode are self-timed and typically take 2.5 ms. The progress of EEPROM write can be monitored by reading the RDY/$\overline{BSY}$ bit (read-only) in SFR WMCON. RDY/$\overline{BSY}$ = 0 means programming is still in progress and RDY/$\overline{BSY}$ = 1 means EEPROM write cycle is completed and another write cycle can be initiated.

In addition, during EEPROM programming, an attempted read from the EEPROM will fetch the byte being written with the MSB complemented. Once the write cycle is completed, true data are valid at all bit locations.

Programmable Watchdog Timer

The programmable Watchdog Timer (WDT) operates from an independent oscillator. The prescaler bits, PS0, PS1 and PS2 in SFR WMCON are used to set the period of the Watchdog Timer from 16 ms to 2048 ms. The available timer periods are shown in the following table and the actual timer periods (at V_{CC} = 5V) are within ±30% of the nominal.

The WDT is disabled by Power-on Reset and during Power-down. It is enabled by setting the WDTEN bit in SFR WMCON (address = 96H). The WDT is reset by setting the WDTRST bit in WMCON. When the WDT times out without being reset or disabled, an internal RST pulse is generated to reset the CPU.

Table 7. Watchdog Timer Period Selection

WDT Prescaler Bits			Period (nominal)
PS2	PS1	PS0	
0	0	0	16 ms
0	0	1	32 ms
0	1	0	64 ms
0	1	1	128 ms
1	0	0	256 ms
1	0	1	512 ms
1	1	0	1024 ms
1	1	1	2048 ms

Timer 0 and 1

Timer 0 and Timer 1 in the AT89S8252 operate the same way as Timer 0 and Timer 1 in the AT89C51, AT89C52 and AT89C55. For further information, see the October 1995 Microcontroller Data Book, page 2-45, section titled, "Timer/Counters."

Timer 2

Timer 2 is a 16 bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/$\overline{T2}$ in the SFR T2CON (shown in Table 2). Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON, as shown in Table 8.

Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which

the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, the level should be held for at least one full machine cycle.

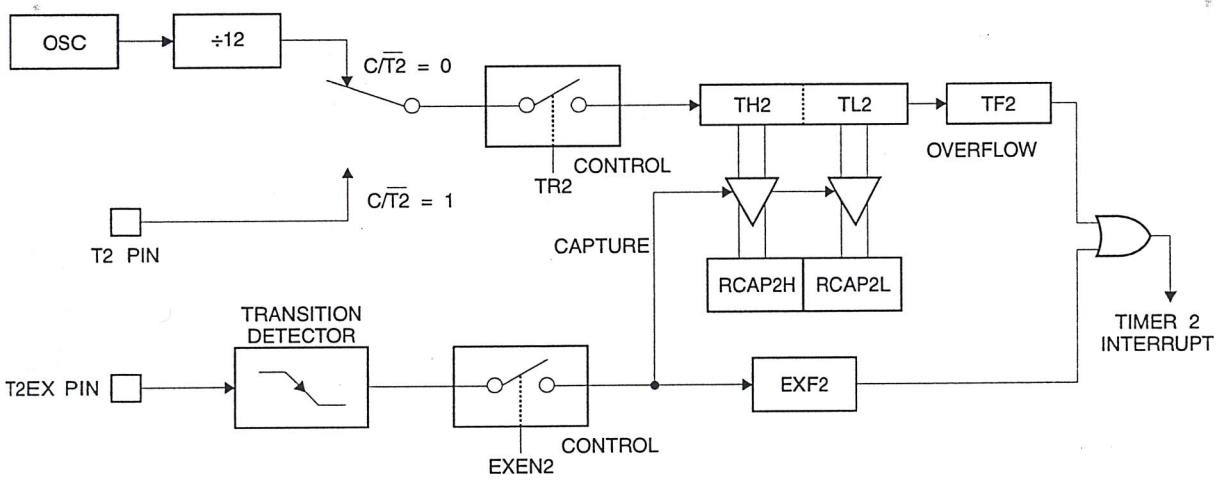
Table 8. Timer 2 Operating Modes

RCLK + TCLK	CP/$\overline{RL2}$	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(Off)

Capture Mode

In the capture mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16 bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 performs the same operation, but a 1-to-0 transition at external input T2EX also causes the current value in TH2 and TL2 to be captured into RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt. The capture mode is illustrated in Figure 1.

Figure 1. Timer 2 in Capture Mode



Auto-reload (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16 bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 9). Upon reset, the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

Figure 2 shows Timer 2 automatically counting up when DCEN = 0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16 bit value in RCAP2H and RCAP2L. The values in RCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16 bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Both the TF2 and EXF2 bits can generate an interrupt if enabled.

Setting the DCEN bit enables Timer 2 to count up or down, as shown in Figure 3. In this mode, the T2EX pin controls the direction of the count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit. This overflow also causes the 16 bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

A logic 0 at T2EX makes Timer 2 count down. The timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers.

The EXF2 bit toggles whenever Timer 2 overflows or underflows and can be used as a 17th bit of resolution. In this operating mode, EXF2 does not flag an interrupt.

Figure 2. Timer 2 in Auto Reload Mode (DCEN = 0)

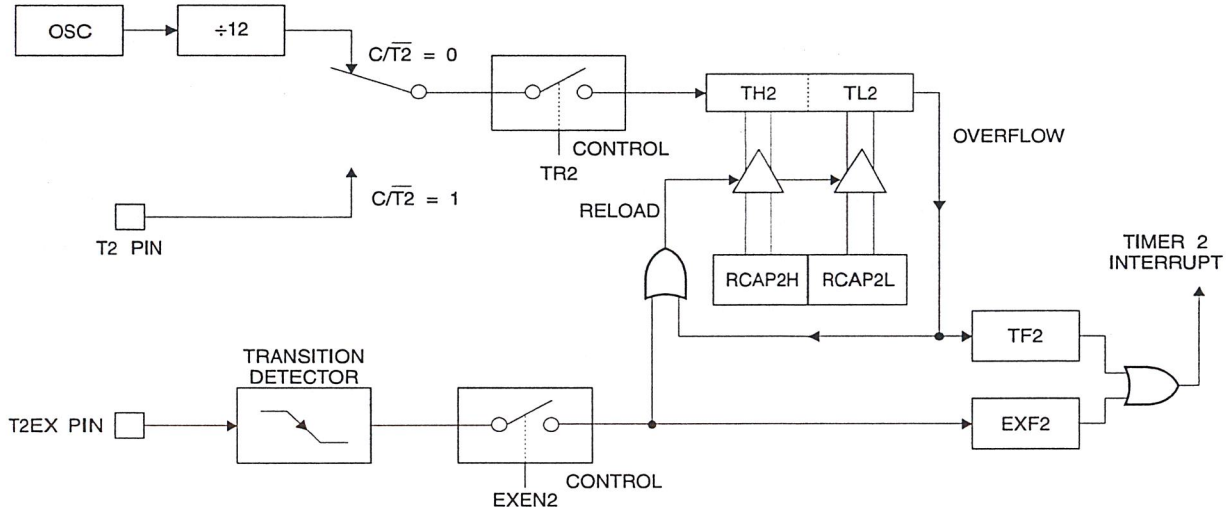


Table 9. T2MOD – Timer 2 Mode Control Register

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

Bit	7	6	5	4	3	2	1	0
	–	–	–	–	–	–	T2OE	DCEN

Symbol	Function
–	Not implemented, reserved for future use.
T2OE	Timer 2 Output Enable bit.
DCEN	When set, this bit allows Timer 2 to be configured as an up/down counter.

Figure 3. Timer 2 Auto Reload Mode (DCEN = 1)

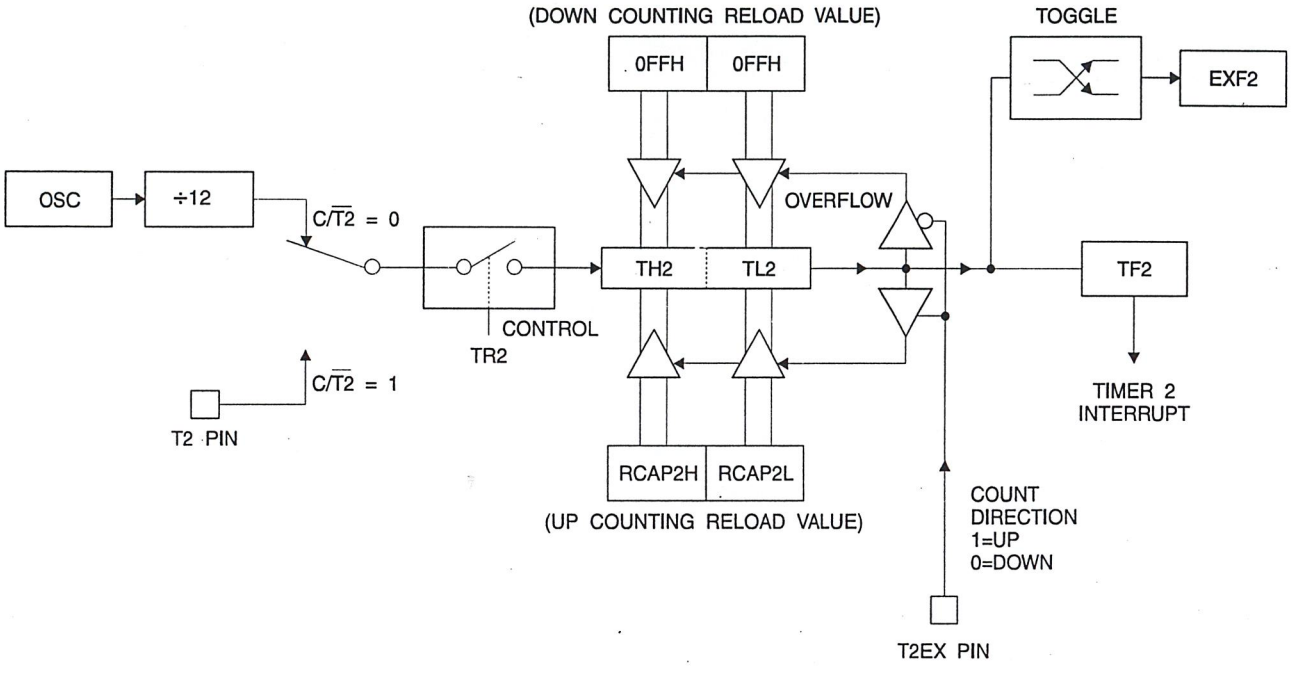
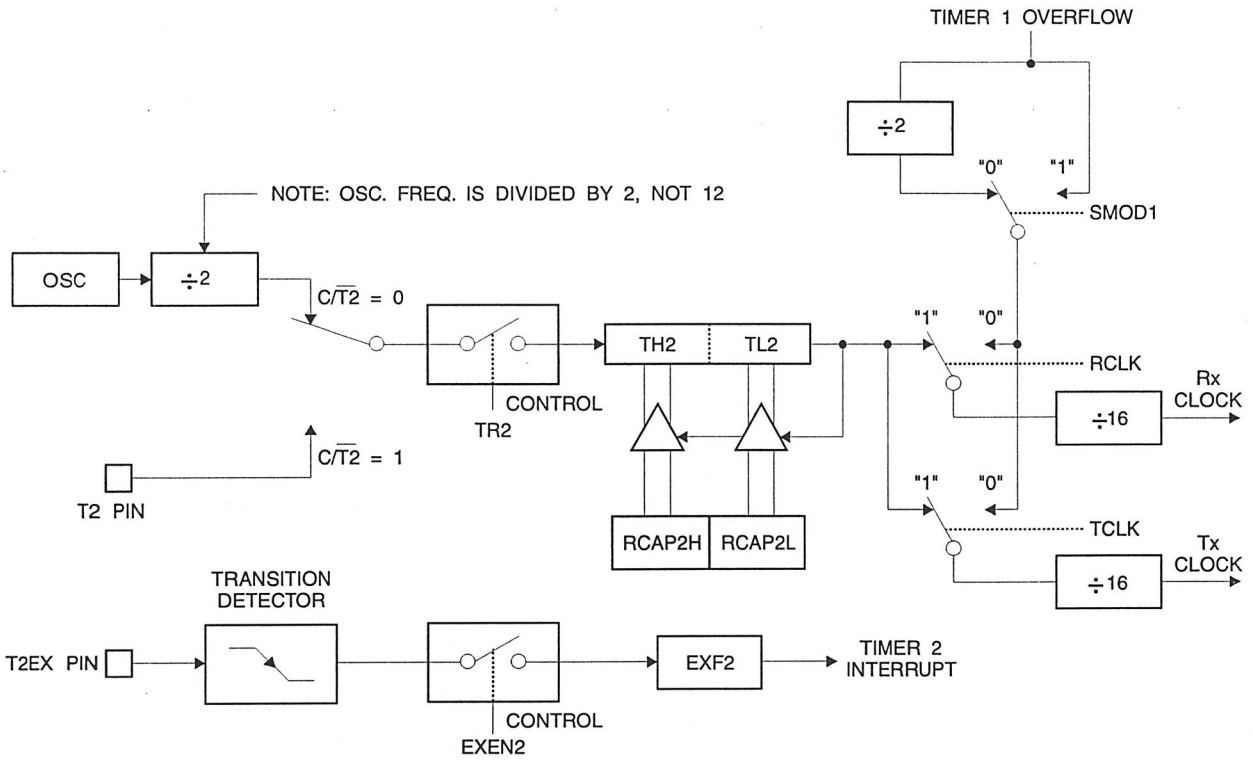


Figure 4. Timer 2 in Baud Rate Generator Mode



Baud Rate Generator

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 2). Note that the baud rates for transmit and receive can be different if Timer 2 is used for the receiver or transmitter and Timer 1 is used for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 4.

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16 bit value in registers RCAP2H and RCAP2L, which are preset by software.

The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate according to the following equation.

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either timer or counter operation. In most applications, it is configured for timer operation ($C/\overline{T2} = 0$). The timer operation is different for Timer 2 when it is used as a baud rate generator. Normally, as a timer, it increments every machine cycle (at 1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (at 1/2 the oscillator frequency). The baud rate formula is given below.

$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16 bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 4. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2 and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt.

Note that when Timer 2 is running (TR2 = 1) as a timer in the baud rate generator mode, TH2 or TL2 should not be read from or written to. Under these conditions, the Timer is incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read but should not be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 5. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed to input the external clock for Timer/Counter 2 or to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz at a 16 MHz operating frequency.

To configure the Timer/Counter 2 as a clock generator, bit $C/\overline{T2}$ (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

The clock-out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L), as shown in the following equation.

$$\text{Clock Out Frequency} = \frac{\text{Oscillator Frequency}}{4 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$

In the clock-out mode, Timer 2 rollovers will not generate an interrupt. This behavior is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and clock-out frequencies cannot be determined independently from one another since they both use RCAP2H and RCAP2L.

Figure 5. Timer 2 in Clock-out Mode

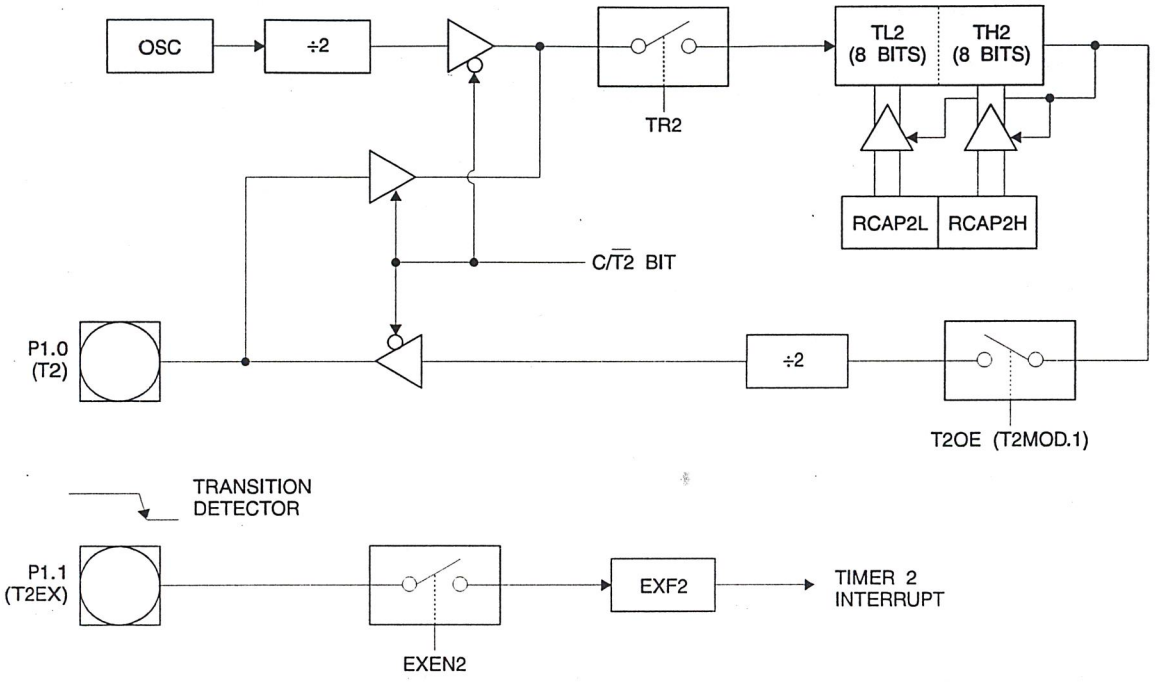
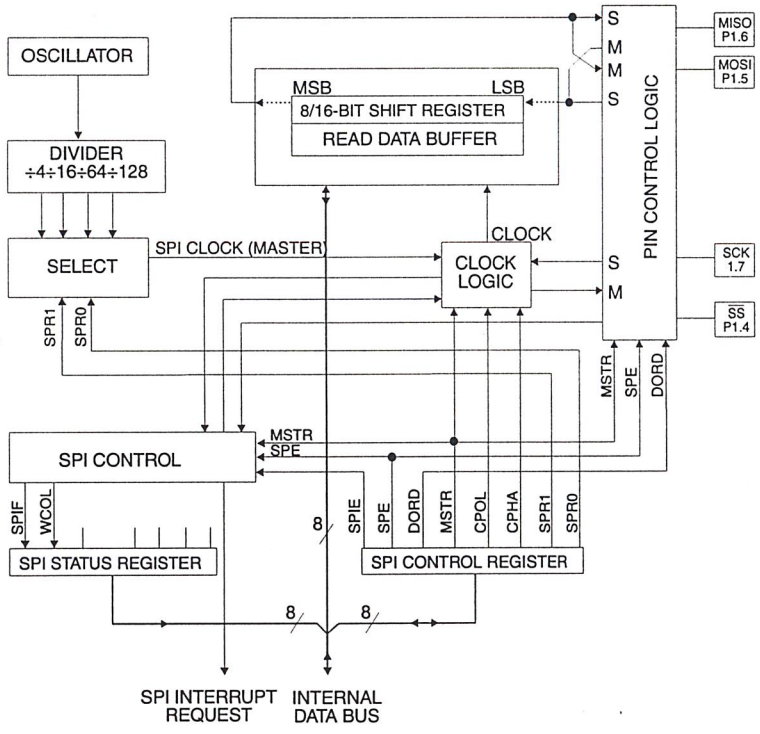


Figure 6. SPI Block Diagram



UART

The UART in the AT89S8252 operates the same way as the UART in the AT89C51, AT89C52 and AT89C55. For further information, see the October 1995 Microcontroller Data Book, page 2-49, section titled, "Serial Interface."

Serial Peripheral Interface

The serial peripheral interface (SPI) allows high-speed synchronous data transfer between the AT89S8252 and peripheral devices or between several AT89S8252 devices. The AT89S8252 SPI features include the following:

- Full-Duplex, 3-Wire Synchronous Data Transfer
- Master or Slave Operation
- 1.5 MHz Bit Frequency (max.)
- LSB First or MSB First Data Transfer
- Four Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wakeup from Idle Mode (Slave Mode Only)

The interconnection between master and slave CPUs with SPI is shown in the following figure. The SCK pin is the clock output in the master mode but is the clock input in the slave mode. Writing to the SPI data register of the master CPU starts the SPI clock generator, and the data written shifts out of the MOSI pin and into the MOSI pin of the slave CPU. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If both the SPI interrupt enable bit (SPIE) and the serial port interrupt enable bit (ES) are set, an interrupt is requested.

The Slave Select input, $\overline{SS}$/P1.4, is set low to select an individual SPI device as a slave. When $\overline{SS}$/P1.4 is set high, the SPI port is deactivated and the MOSI/P1.5 pin can be used as an input.

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 8 and Figure 9.

Figure 7. SPI Master-slave Interconnection

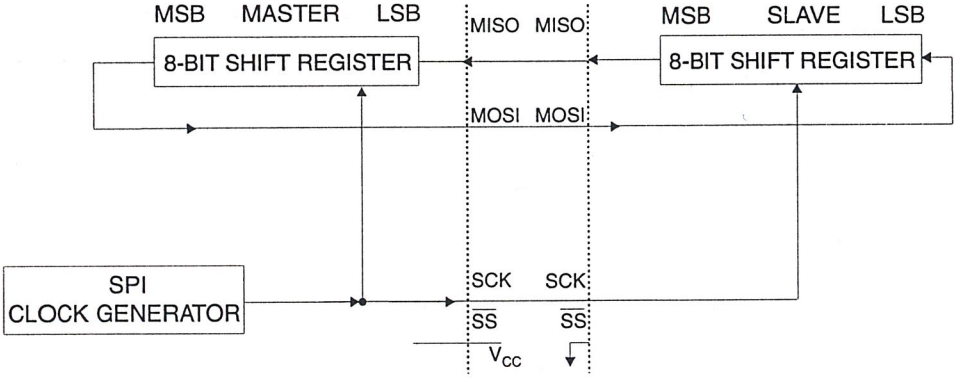
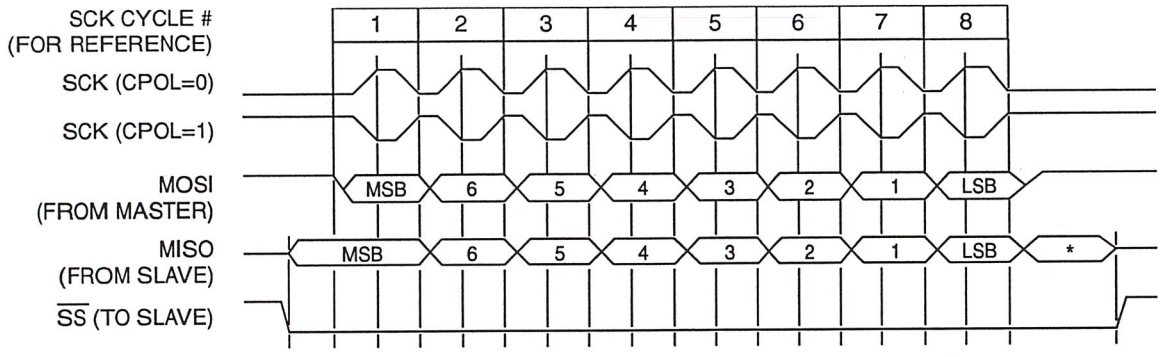
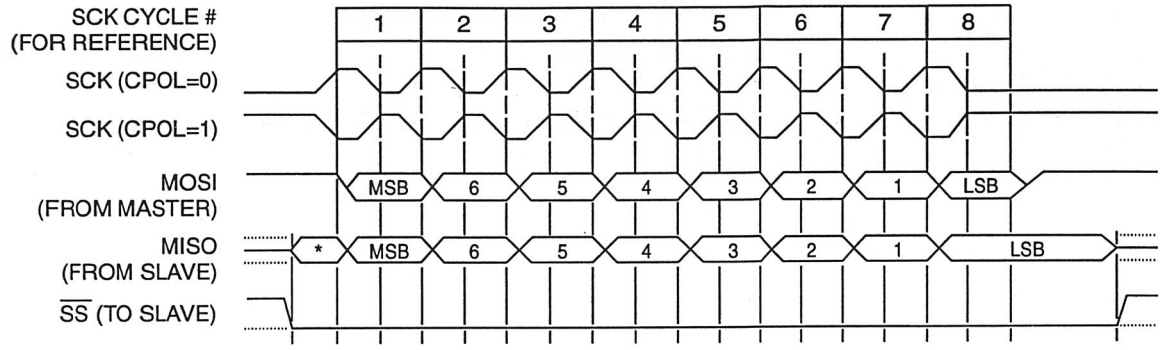


Figure 8. SPI transfer Format with CPHA = 0



*Not defined but normally MSB of character just received

Figure 9. SPI Transfer Format with CPHA = 1



*Not defined but normally LSB of previously transmitted character

Interrupts

The AT89S8252 has a total of six interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 10.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE. IE also contains a global disable bit, EA, which disables all interrupts at once.

Note that Table 10 shows that bit position IE.6 is unimplemented. In the AT89C51, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future AT89 products.

Timer 2 interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and that bit will have to be cleared in software.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag, TF2, is set at S2P2 and is polled in the same cycle in which the timer overflows.

Table 10. Interrupt Enable (IE) Register

(MSB)(LSB)

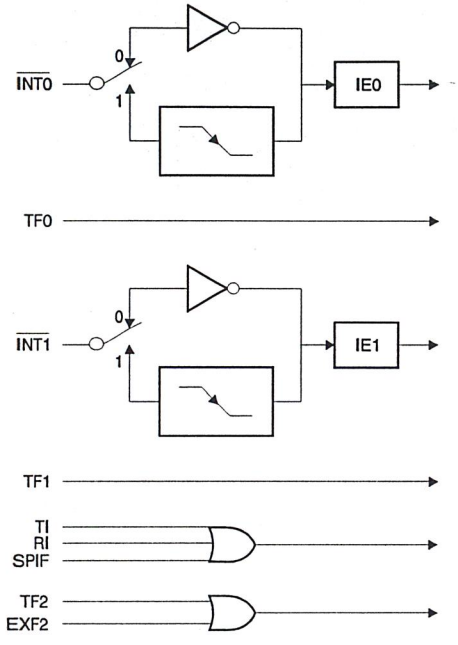
EA	–	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	Reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	SPI and UART interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.
User software should never write 1s to unimplemented bits, because they may be used in future AT89 products.		

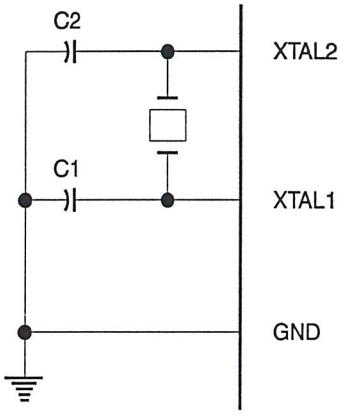
Figure 10. Interrupt Sources



Oscillator Characteristics

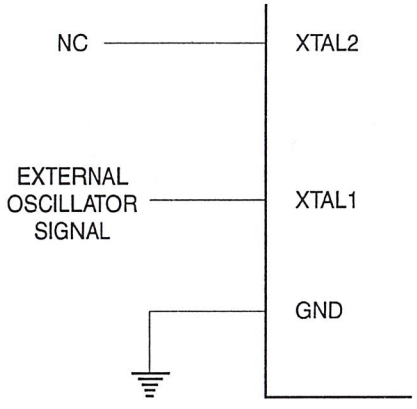
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 11. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 12. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Figure 11. Oscillator Connections



Note: Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 12. External Clock Drive Configuration



Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

Note that when idle mode is terminated by a hardware reset, the device normally resumes program execution from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when idle mode is terminated by a reset, the instruction following the one that invokes idle mode should not write to a port pin or to external memory.

Status of External Pins During Idle and Power-down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

Power-down Mode

In the power-down mode, the oscillator is stopped and the instruction that invokes power-down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power-down mode is terminated. Exit from power-down can be initiated either by a hardware reset or by an enabled external interrupt. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

To exit power-down via an interrupt, the external interrupt must be enabled as level sensitive before entering power-down. The interrupt service routine starts at 16 ms (nominal) after the enabled interrupt pin is activated.

Program Memory Lock Bits

The AT89S8252 has three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the following table.

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value and holds that value until reset is activated. The latched value of $\overline{EA}$ must agree with the current logic level at that pin in order for the device to function properly.

Once programmed, the lock bits can only be unprogrammed with the Chip Erase operations in either the parallel or serial modes.

Lock Bit Protection Modes[1][2]

Program Lock Bits				
	LB1	LB2	LB3	Protection Type
1	U	U	U	No internal memory lock feature.
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory. $\overline{EA}$ is sampled and latched on reset and further programming of the Flash memory (parallel or serial mode) is disabled.
3	P	P	U	Same as Mode 2, but parallel or serial verify are also disabled.
4	P	P	P	Same as Mode 3, but external execution is also disabled.

Notes: 1. U = Unprogrammed
2. P = Programmed

Programming the Flash and EEPROM

Atmel's AT89S8252 Flash Microcontroller offers 8K bytes of in-system reprogrammable Flash Code memory and 2K bytes of EEPROM Data memory.

The AT89S8252 is normally shipped with the on-chip Flash Code and EEPROM Data memory arrays in the erased state (i.e. contents = FFH) and ready to be programmed. This device supports a High-voltage (12V) Parallel programming mode and a Low-voltage (5V) Serial programming mode. The serial programming mode provides a convenient way to download the AT89S8252 inside the user's system. The parallel programming mode is compatible with conventional third party Flash or EPROM programmers.

The Code and Data memory arrays are mapped via separate address spaces in the serial programming mode. In the parallel programming mode, the two arrays occupy one contiguous address space: 0000H to 1FFFH for the Code array and 2000H to 27FFH for the Data array.

The Code and Data memory arrays on the AT89S8252 are programmed byte-by-byte in either programming mode. An auto-erase cycle is provided with the self-timed programming operation in the serial programming mode. There is no need to perform the Chip Erase operation to reprogram any memory location in the serial programming mode unless any of the lock bits have been programmed.

In the parallel programming mode, there is no auto-erase cycle. To reprogram any non-blank byte, the user needs to use the Chip Erase operation first to erase both arrays.

Parallel Programming Algorithm: To program and verify the AT89S8252 in the parallel programming mode, the following sequence is recommended:

1. Power-up sequence:
   Apply power between V_{CC} and GND pins.
   Set RST pin to "H".
   Apply a 3 MHz to 24 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
2. Set $\overline{PSEN}$ pin to "L"
   ALE pin to "H"
   $\overline{EA}$ pin to "H" and all other pins to "H".
3. Apply the appropriate combination of "H" or "L" logic levels to pins P2.6, P2.7, P3.6, P3.7 to select one of the programming operations shown in the Flash Programming Modes table.
4. Apply the desired byte address to pins P1.0 to P1.7 and P2.0 to P2.5.
   Apply data to pins P0.0 to P0.7 for Write Code operation.
5. Raise $\overline{EA}/V_{PP}$ to 12V to enable Flash programming, erase or verification.
6. Pulse ALE/$\overline{PROG}$ once to program a byte in the Code memory array, the Data memory array or the lock bits. The byte-write cycle is self-timed and typically takes 1.5 ms.
7. To verify the byte just programmed, bring pin P2.7 to "L" and read the programmed data at pins P0.0 to P0.7.
8. Repeat steps 3 through 7 changing the address and data for the entire 2K or 8K bytes array or until the end of the object file is reached.
9. Power-off sequence:
   Set XTAL1 to "L".
   Set RST and $\overline{EA}$ pins to "L".
   Turn V_{CC} power off.

In the parallel programming mode, there is no auto-erase cycle and to reprogram any non-blank byte, the user needs to use the Chip Erase operation first to erase both arrays.

$\overline{Data}$ Polling: The AT89S8252 features $\overline{DATA}$ Polling to indicate the end of a write cycle. During a write cycle in the parallel or serial programming mode, an attempted read of the last byte written will result in the complement of the written datum on P0.7 (parallel mode), and on the MSB of the serial output byte on MISO (serial mode). Once the write cycle has been completed, true data are valid on all outputs, and the next cycle may begin. $\overline{DATA}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{Busy}$: The progress of byte programming in the parallel programming mode can also be monitored by the RDY/$\overline{BSY}$ output signal. Pin P3.4 is pulled Low after ALE goes High during programming to indicate $\overline{BUSY}$. P3.4 is pulled High again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed Code or Data byte can be read back via the address and data lines for verification. The state of the lock bits can also be verified directly in the parallel programming mode. In the serial programming mode, the state of the lock bits can only be verified indirectly by observing that the lock bit features are enabled.

Chip Erase: Both Flash and EEPROM arrays are erased electrically at the same time. In the parallel programming mode, chip erase is initiated by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The Code and Data arrays are written with all "1"s in the Chip Erase operation.

In the serial programming mode, a chip erase operation is initiated by issuing the Chip Erase instruction. In this mode, chip erase is self-timed and takes about 16 ms.

During chip erase, a serial read from any address location will return 00H at the data outputs.

Serial Programming Fuse: A programmable fuse is available to disable Serial Programming if the user needs maximum system security. The Serial Programming Fuse can only be programmed or erased in the Parallel Programming Mode.

The AT89S8252 is shipped with the Serial Programming Mode enabled.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H and 031H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows:

(030H) = 1EH indicates manufactured by Atmel
(031H) = 72H indicates 89S8252

Programming Interface

Every code byte in the Flash and EEPROM arrays can be written, and the entire array can be erased, by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

All major programming vendors offer worldwide support for the Atmel microcontroller series. Please contact your local programming vendor for the appropriate software revision.

Serial Downloading

Both the Code and Data memory arrays can be programmed using the serial SPI bus while RST is pulled to V_{CC}. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After RST is set high, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

An auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the Chip Erase instruction unless any of the lock bits have been programmed. The Chip Erase operation turns the content of every memory location in both the Code and Data arrays into FFH.

The Code and Data memory arrays have separate address spaces:

0000H to 1FFFH for Code memory and 000H to 7FFH for Data memory.

Either an external system clock is supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The maximum serial clock (SCK) frequency should be less than 1/40 of the crystal frequency. With a 24 MHz oscillator clock, the maximum SCK frequency is 600 kHz.

Serial Programming Algorithm

To program and verify the AT89S8252 in the serial programming mode, the following sequence is recommended:

1. Power-up sequence:

   Apply power between VCC and GND pins.

   Set RST pin to "H".

   If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 24 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.
2. Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 40.
3. The Code or Data array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. The selected memory location is first automatically erased before new data is written. The write cycle is self-timed and typically takes less than 2.5 ms at 5V.
4. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/P1.6.
5. At the end of a programming session, RST can be set low to commence normal operation.

Power-off sequence (if needed):

Set XTAL1 to "L" (if a crystal is not used).

Set RST to "L".

Turn V_{CC} power off.

Serial Programming Instruction

The Instruction Set for Serial Programming follows a 3-byte protocol and is shown in the following table:

Instruction Set

Instruction	Input Format			Operation
	Byte 1	Byte 2	Byte 3	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	Enable serial programming interface after RST goes high.
Chip Erase	1010 1100	xxxx x100	xxxx xxxx	Chip erase both 8K & 2K memory arrays.
Read Code Memory	aaaa a001	low addr	xxxx xxxx	Read data from Code memory array at the selected address. The 5 MSBs of the first byte are the high order address bits. The low order address bits are in the second byte. Data are available at pin MISO during the third byte.
Write Code Memory	aaaa a010	low addr	data in	Write data to Code memory location at selected address. The address bits are the 5 MSBs of the first byte together with the second byte.
Read Data Memory	00aa a101	low addr	xxxx xxxx	Read data from Data memory array at selected address. Data are available at pin MISO during the third byte.
Write Data Memory	00aa a110	low addr	data in	Write data to Data memory location at selected address.
Write Lock Bits	1010 1100	x x111	xxxx xxxx	Write lock bits. Set LB1, LB2 or LB3 = "0" to program lock bits.

Note: 1. DATA polling is used to indicate the end of a write cycle which typically takes less than 2.5 ms at 5V.
2. "aaaaa" = high order address.
3. "x" = don't care.

Flash and EEPROM Parallel Programming Modes

Mode	RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.6	P2.7	P3.6	P3.7	Data I/O P0.7:0	Address P2.5:0 P1.7:0
Serial Prog. Modes	H	h(1)	h(1)	x						
Chip Erase	H	L	⌐⌐ (2)	12V	H	L	L	L	X	X
Write (10K bytes) Memory	H	L	⌐⌐	12V	L	H	H	H	DIN	ADDR
Read (10K bytes) Memory	H	L	H	12V	L	L	H	H	DOUT	ADDR
Write Lock Bits:	H	L	⌐⌐	12V	H	L	H	L	DIN	X
Bit - 1									P0.7 = 0	X
Bit - 2									P0.6 = 0	X
Bit - 3									P0.5 = 0	X
Read Lock Bits:	H	L	H	12V	H	H	L	L	DOUT	X
Bit - 1									@P0.2	X
Bit - 2									@P0.1	X
Bit - 3									@P0.0	X
Read Atmel Code	H	L	H	12V	L	L	L	L	DOUT	30H
Read Device Code	H	L	H	12V	L	L	L	L	DOUT	31H
Serial Prog. Enable	H	L	⌐⌐ (2)	12V	L	H	L	H	P0.0 = 0	X
Serial Prog. Disable	H	L	⌐⌐ (2)	12V	L	H	L	H	P0.0 = 1	X
Read Serial Prog. Fuse	H	L	H	12V	H	H	L	H	@P0.0	X

Notes:
1. "h" = weakly pulled "High" internally.
2. Chip Erase and Serial Programming Fuse require a 10 ms $\overline{PROG}$ pulse. Chip Erase needs to be performed first before reprogramming any byte with a content other than FFH.
3. P3.4 is pulled Low during programming to indicate RDY/$\overline{BSY}$.
4. "X" = don't care

Figure 13. Programming the Flash/EEPROM Memory

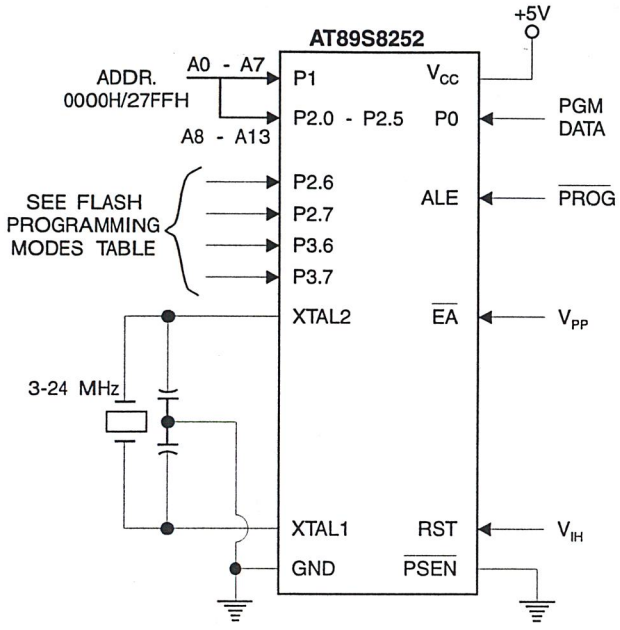


Figure 14. Verifying the Flash/EEPROM Memory

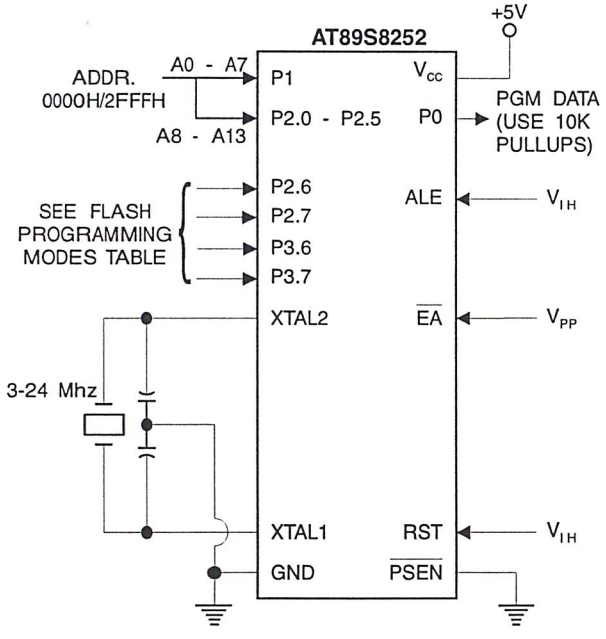
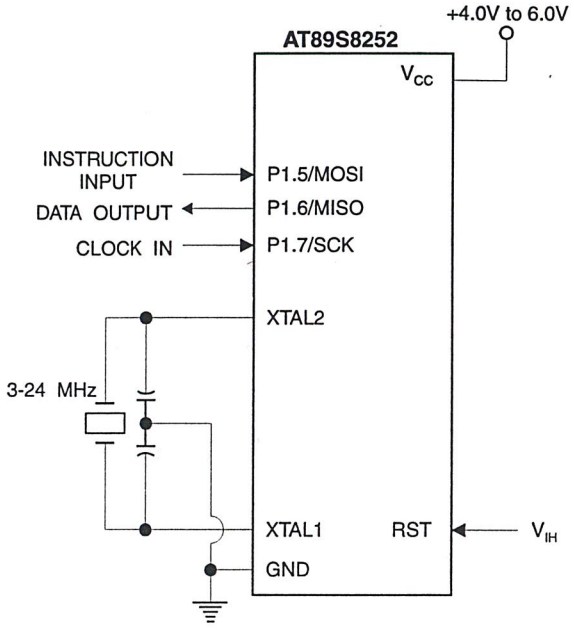


Figure 15. Flash/EEPROM Serial Downloading

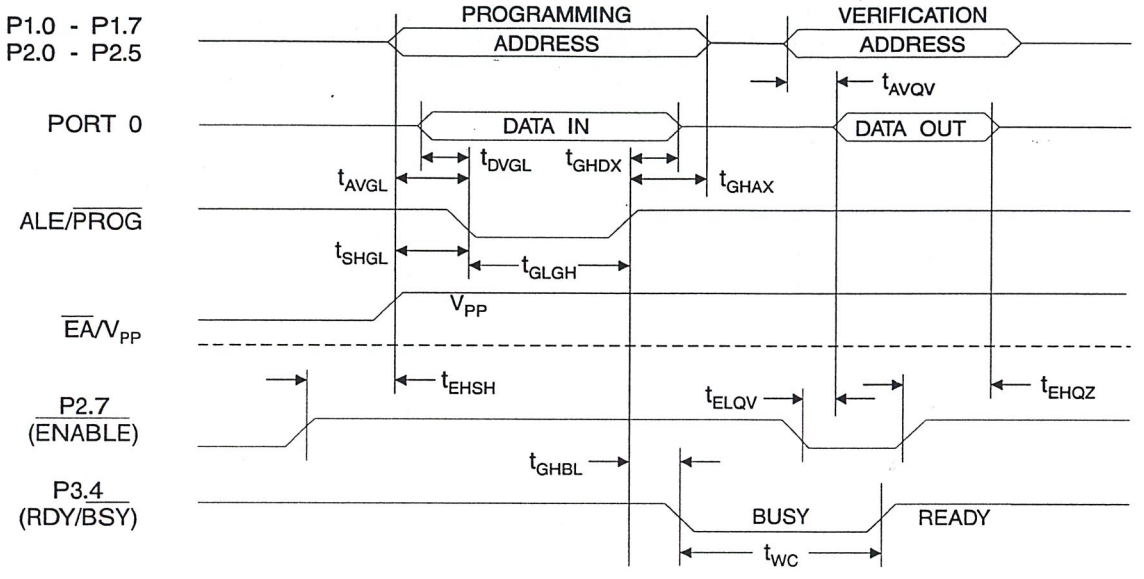


Flash Programming and Verification Characteristics – Parallel Mode

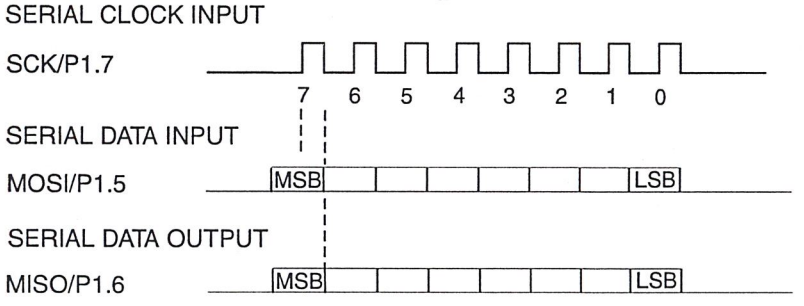
T_A = 0°C to 70°C, V_{CC} = 5.0V ± 10%

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Enable Voltage	11.5	12.5	V
I_{PP}	Programming Enable Current		1.0	mA
$1/t_{CLCL}$	Oscillator Frequency	3	24	MHz
t_{AVGL}	Address Setup to $\overline{\text{PROG}}$ Low	$48t_{CLCL}$		
t_{GHAX}	Address Hold after $\overline{\text{PROG}}$	$48t_{CLCL}$		
t_{DVGL}	Data Setup to $\overline{\text{PROG}}$ Low	$48t_{CLCL}$		
t_{GHDX}	Data Hold after $\overline{\text{PROG}}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{\text{ENABLE}}$) High to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} Setup to $\overline{\text{PROG}}$ Low	10		µs
t_{GLGH}	$\overline{\text{PROG}}$ Width	1	110	µs
t_{AVQV}	Address to Data Valid		$48t_{CLCL}$	
t_{ELQV}	$\overline{\text{ENABLE}}$ Low to Data Valid		$48t_{CLCL}$	
t_{EHQZ}	Data Float after $\overline{\text{ENABLE}}$	0	$48t_{CLCL}$	
t_{GHBL}	$\overline{\text{PROG}}$ High to $\overline{\text{BUSY}}$ Low		1.0	µs
t_{WC}	Byte Write Cycle Time		2.0	ms

Flash/EEPROM Programming and Verification Waveforms – Parallel Mode



Serial Downloading Waveforms



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	15.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

The values shown in this table are valid for T_A = -40°C to 85°C and V_{CC} = 5.0V ± 20%, unless otherwise noted.

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low-voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC} - 0.1	V
V_{IL1}	Input Low-voltage ($\overline{EA}$)		-0.5	0.2 V_{CC} - 0.3	V
V_{IH}	Input Hifh-voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input Hifh-voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
V_{OL}	Output Low-voltage (1) (Ports 1,2,3)	I_{OL} = 1.6 mA		0.5	V
V_{OL1}	Output Low-voltage (1) (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.5	V
V_{OH}	Output Hifh-voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
V_{OH1}	Output Hifh-voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ± 10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pull-down Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode (2)	V_{CC} = 6V		100	µA
		V_{CC} = 3V		40	µA

Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin: 10 mA
Maximum I_{OL} per 8-bit port:
Port 0: 26 mA
Ports 1, 2, 3: 15 mA
Maximum total I_{OL} for all output pins: 71 mA
If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

2. Minimum V_{CC} for Power-down is 2V

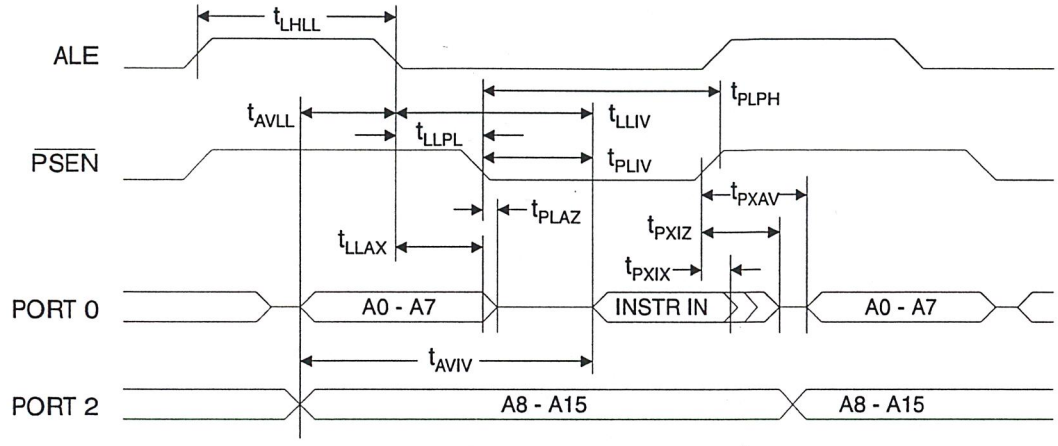
AC Characteristics

Under operating conditions, load capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; load capacitance for all other outputs = 80 pF.

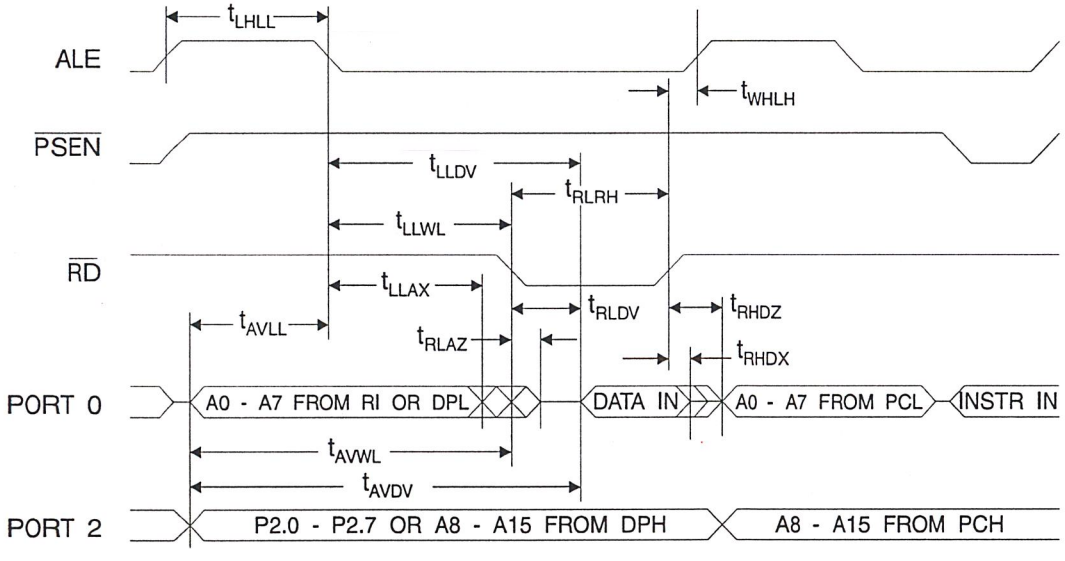
External Program and Data Memory Characteristics

Symbol	Parameter	Variable Oscillator		Units
		Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{LHLL}	ALE Pulse Width	$2t_{CLCL}$ - 40		ns
t_{AVLL}	Address Valid to ALE Low	t_{CLCL} - 13		ns
t_{LLAX}	Address Hold after ALE Low	t_{CLCL} - 20		ns
t_{LLIV}	ALE Low to Valid Instruction In		$4t_{CLCL}$ - 65	ns
t_{LLPL}	ALE Low to $\overline{PSEN}$ Low	t_{CLCL} - 13		ns
t_{PLPH}	$\overline{PSEN}$ Pulse Width	$3t_{CLCL}$ - 20		ns
t_{PLIV}	$\overline{PSEN}$ Low to Valid Instruction In		$3t_{CLCL}$ - 45	ns
t_{PXIX}	Input Instruction Hold after $\overline{PSEN}$	0		ns
t_{PXIZ}	Input Instruction Float after $\overline{PSEN}$		t_{CLCL} - 10	ns
t_{PXAV}	$\overline{PSEN}$ to Address Valid	t_{CLCL} - 8		ns
t_{AVIV}	Address to Valid Instruction In		$5t_{CLCL}$ - 55	ns
t_{PLAZ}	$\overline{PSEN}$ Low to Address Float		10	ns
t_{RLRH}	$\overline{RD}$ Pulse Width	$6t_{CLCL}$ - 100		ns
t_{WLWH}	$\overline{WR}$ Pulse Width	$6t_{CLCL}$ - 100		ns
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		$5t_{CLCL}$ - 90	ns
t_{RHDX}	Data Hold after $\overline{RD}$	0		ns
t_{RHDZ}	Data Float after $\overline{RD}$		$2t_{CLCL}$ - 28	ns
t_{LLDV}	ALE Low to Valid Data In		$8t_{CLCL}$ - 150	ns
t_{AVDV}	Address to Valid Data In		$9t_{CLCL}$ - 165	ns
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	$3t_{CLCL}$ - 50	$3t_{CLCL}$ + 50	ns
t_{AVWL}	Address to $\overline{RD}$ or $\overline{WR}$ Low	$4t_{CLCL}$ - 75		ns
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	t_{CLCL} - 20		ns
t_{QVWH}	Data Valid to $\overline{WR}$ High	$7t_{CLCL}$ - 120		ns
t_{WHQX}	Data Hold after $\overline{WR}$	t_{CLCL} - 20		ns
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0	ns
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	t_{CLCL} - 20	t_{CLCL} + 25	ns

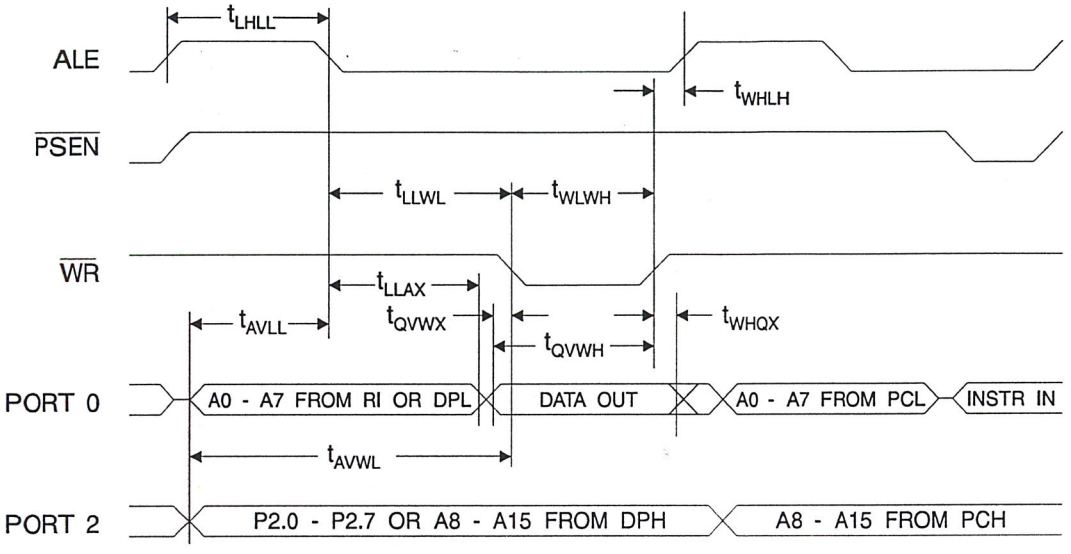
External Program Memory Read Cycle



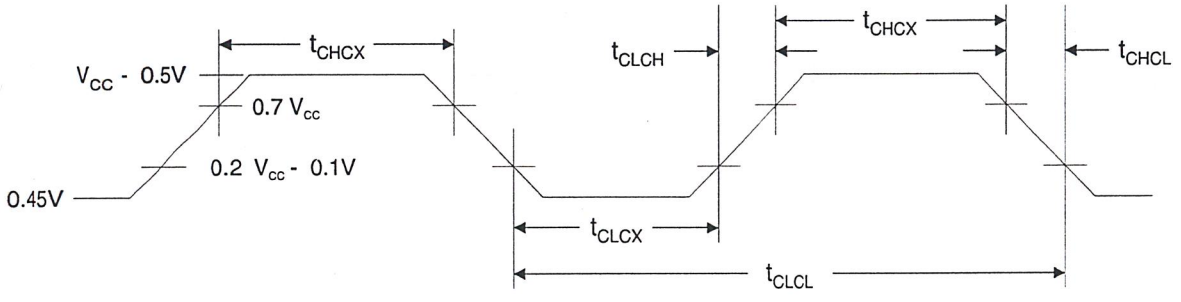
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

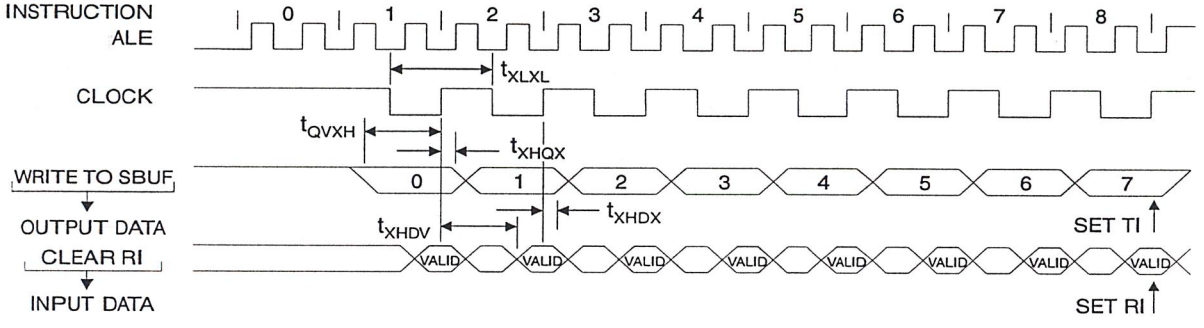
Symbol	Parameter	V_{CC} = 4.0V to 6.0V		Units
		Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{CLCL}	Clock Period	41.6		ns
t_{CHCX}	High Time	15		ns
t_{CLCX}	Low Time	15		ns
t_{CLCH}	Rise Time		20	ns
t_{CHCL}	Fall Time		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

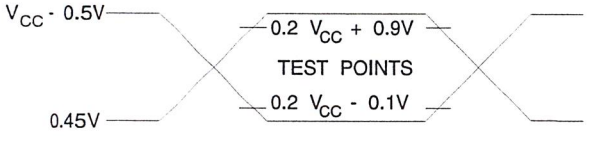
The values in this table are valid for V_{CC} = 4.0V to 6V and Load Capacitance = 80 pF.

Symbol	Parameter	Variable Oscillator		Units
		Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	$10t_{CLCL}$ - 133		ns
t_{XHQX}	Output Data Hold after Clock Rising Edge	$2t_{CLCL}$ - 117		ns
t_{XHDX}	Input Data Hold after Clock Rising Edge	0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		$10t_{CLCL}$ - 133	ns

Shift Register Mode Timing Waveforms

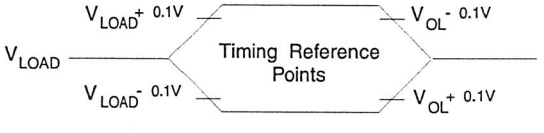


AC Testing Input/Output Waveforms[1]



Notes: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

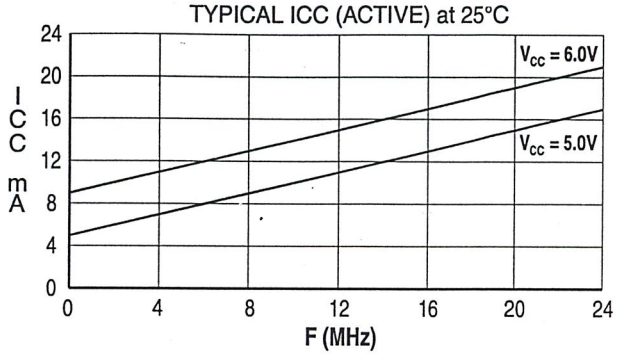
Float Waveforms[1]



Notes: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

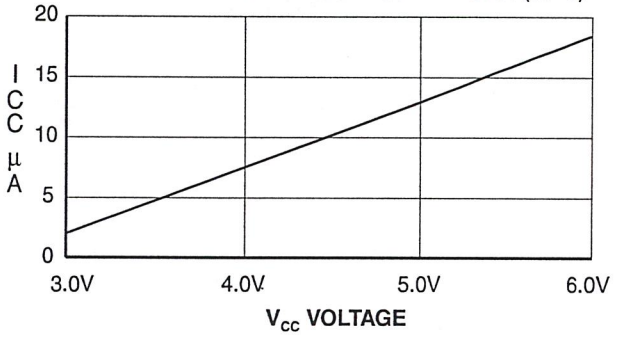
AT89S8252

TYPICAL ICC (ACTIVE) at 25°C



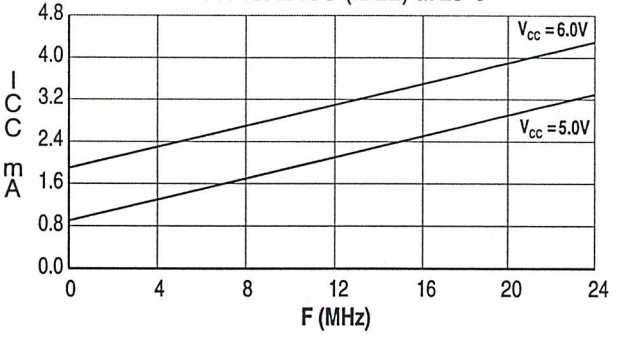
AT89S8252

TYPICAL ICC vs. VOLTAGE - POWER DOWN (85°C)



AT89S8252

TYPICAL ICC (IDLE) at 25°C



Notes: 1. XTAL1 tied to GND for Icc (power-down)
2. Lock bits programmed

Ordering Information

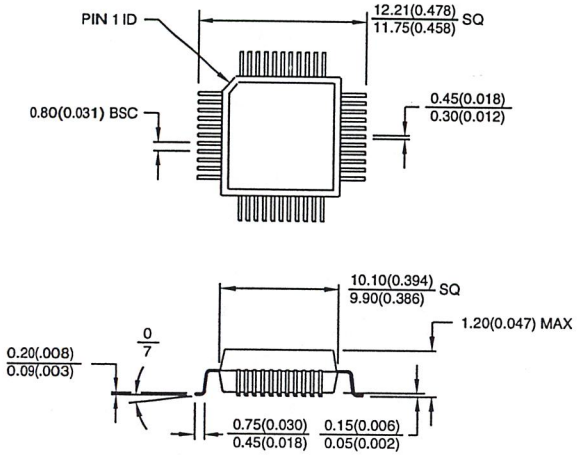
Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
24	4.0V to 6.0V	AT89S8252-24AC AT89S8252-24JC AT89S8252-24PC AT89S8252-24QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)
	4.0V to 6.0V	AT89S8252-24AI AT89S8252-24JI AT89S8252-24PI AT89S8252-24QI	44A 44J 40P6 44Q	Industrial (-40°C to 85°C)
33	4.5V to 5.5V	AT89S8252-33AC AT89S8252-33JC AT89S8252-33PC AT89S8252-33QC	44A 44J 40P6 44Q	Commercial (0°C to 70°C)

= Preliminary Information

Package Type	
44A	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
44J	44-lead, Plastic J-leaded Chip Carrier (PLCC)
40P6	40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
44Q	44-lead, Plastic Gull Wing Quad Flatpack (PQFP)

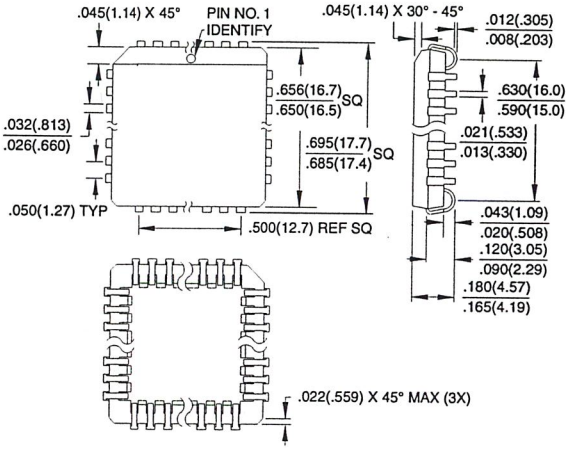
Packaging Information

44A, 44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flatpack (TQFP)
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-026 ACB

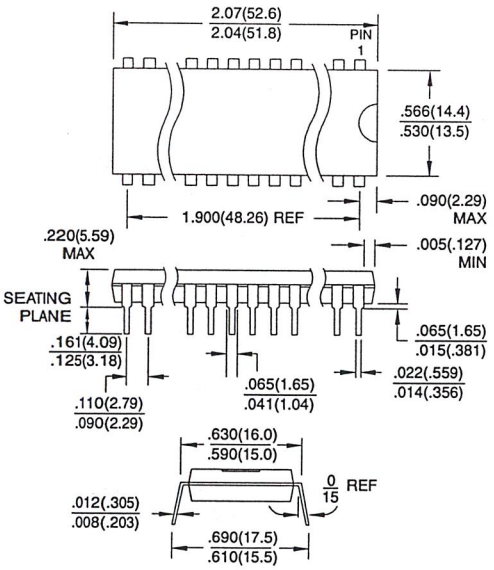


Controlling dimension: millimeters

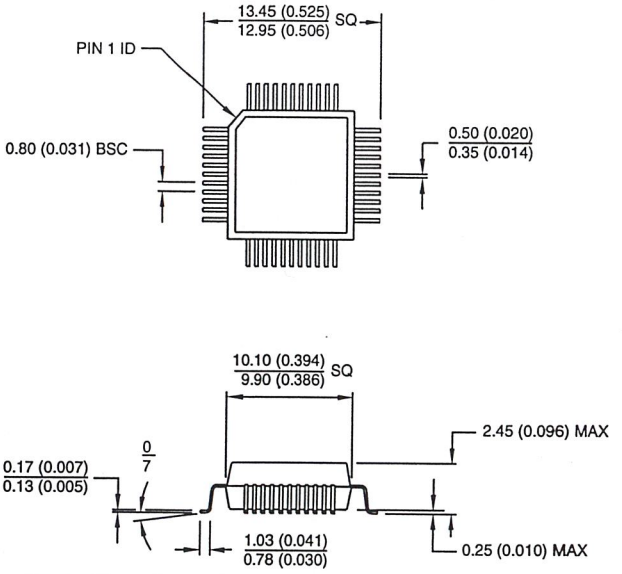
44J, 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-018 AC



40P6, 40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)



44Q, 44-lead, Plastic Quad Flat Package (PQFP)
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-022 AB



Controlling dimension: millimeters

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686-677
FAX (44) 1276-686-697

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

0401E–02/00/xM

DALLAS SEMICONDUCTOR

www.dalsemi.com

DS1307/DS1308
64 X 8 Serial Real Time Clock

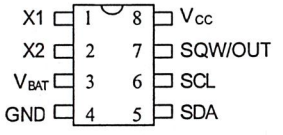
FEATURES

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

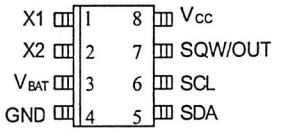
ORDERING INFORMATION

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

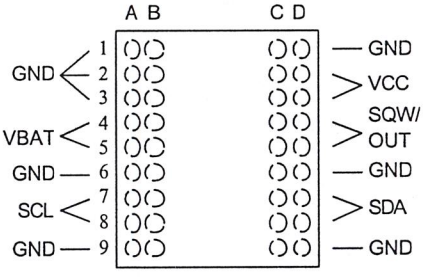
PIN ASSIGNMENT



DS1307 8-Pin DIP (300-mil)



DS1307Z 8-Pin SOIC (150-mil)



DS1308 36-Pin SMD BGA
(TOP VIEW)

PIN DESCRIPTION DS1307/DS1308

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

DS1308 PIN IDENTIFIER

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

DESCRIPTION

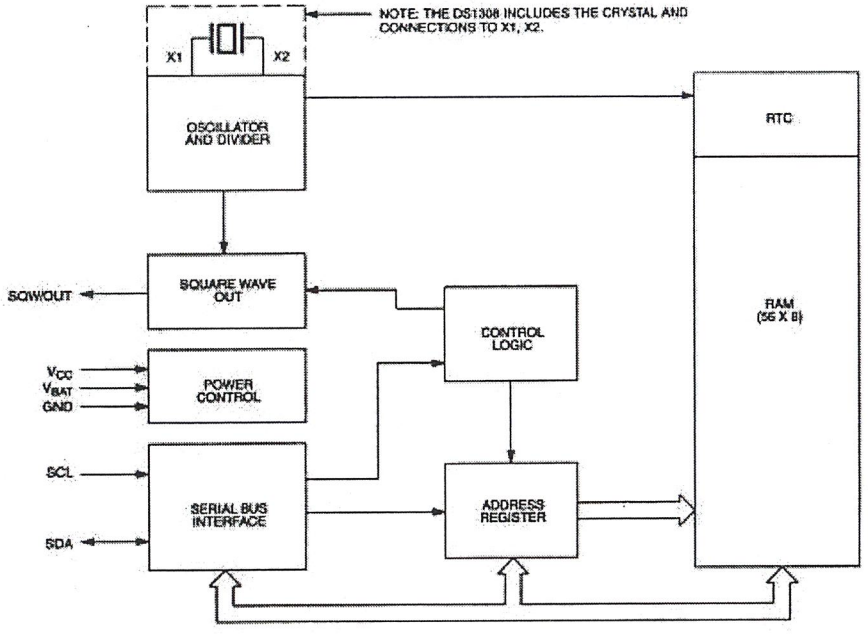
The DS1307 Serial Real Time Clock is a low power, full BCD clock/calendar plus 56 bytes of nonvolatile SRAM. Address and data are transferred serially via a 2-wire bi-directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date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

The DS1308 incorporates the DS1307 chip with a 32.768 kHz crystal in a surface mountable, 36-pin ball grid array package (BGA). The close proximity of the embedded crystal to the high impedance crystal input pins on the DS1307 minimizes capacitive loading and noise injection problems associated with many other oscillator designs. The total area required for installation is less than that of one United States dime: thus, minimizing PCB space required.

OPERATION

The DS1307/1308 operates as a slave device on the serial bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. When V_{CC} falls below 1.25 x V_{BAT} the device terminates an access in progress and resets the device address counter. Inputs to the device will not be recognized at this time to prevent erroneous data from being written to the device from an out of tolerance system. When V_{CC} falls below V_{BAT} the device switches into a low current battery backup mode. Upon power up, the device switches from battery to V_{CC} when V_{CC} is greater than V_{BAT} +0.2V and recognizes inputs when V_{CC} is greater than 1.25 x V_{BAT}. The block diagram in Figure 1 shows the main elements of the Serial Real Time Clock.

DS1307/DS1308 BLOCK DIAGRAM Figure 1



SIGNAL DESCRIPTIONS

V_{CC}, GND - DC power is provided to the device on these pins. V_{CC} is the +5 volt input. When 5 volts is applied within normal limits, the device is fully accessible and data can be written and read. When a 3-volt battery is connected to the device and V_{CC} is below 1.25 x V_{BAT}, reads and writes are inhibited. However, the Timekeeping function continues unaffected by the lower input voltage. As V_{CC} falls below V_{BAT} the RAM and timekeeper are switched over to the external power supply (nominal 3.0V DC) at V_{BAT}.

V_{BAT} - Battery input for any standard 3-volt lithium cell or other energy source. Battery voltage must be held between 2.0 and 3.5 volts for proper operation. The nominal write protect trip point voltage at which access to the real time clock and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A lithium battery with 48 mAhr or greater will back up the DS1307/DS1308 for more than 10 years in the absence of power at 25 degrees C.

SCL (Serial Clock Input) - SCL is used to synchronize data movement on the serial interface.

SDA (Serial Data Input/Output) - SDA is the input/output pin for the 2-wire serial interface. The SDA pin is open drain which requires an external pullup resistor.

SQW/OUT (Square Wave/ Output Driver) - When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square wave frequencies (1 Hz, 4 kHz, 8 kHz, 32 kHz). The SQW/OUT pin is open drain which requires an external pullup resistor.

NOTE: X1, X2 are not applicable for the DS1308 or DS1308N.

X1, X2 - Connections for a standard 32.768 kHz quartz crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (CL) of 12.5 pF.

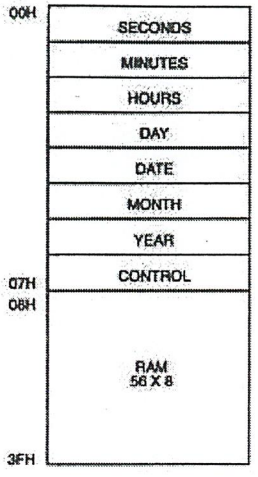
For more information on crystal selection and crystal layout considerations, please consult Application Note 58, "Crystal Considerations with Dallas Real Time Clocks." The DS1307 can also be driven by an external 32.768 kHz oscillator. In this configuration, the X1 pin is connected to the external oscillator signal and the X2 pin is floated.

Please review Application Note 95, "Interfacing the DS1307/DS1308 with a 8051-Compatible Microcontroller" for additional information.

RTC AND RAM ADDRESS MAP

The address map for the RTC and RAM registers of the DS1307/DS1308 is shown in Figure 2. The real time clock registers are located in address locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a multi-byte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to location 00h, the beginning of the clock space.

DS1307 ADDRESS MAP Figure 2



CLOCK AND CALENDAR

The time and calendar information is obtained by reading the appropriate register bytes. The real time clock registers are illustrated in Figure 3. The time and calendar are set or initialized by writing the appropriate register bytes. The contents of the time and calendar registers are in the Binary-Coded Decimal (BCD) format. Bit 7 of Register 0 is the Clock Halt (CH) bit. When this bit is set to a 1, the oscillator is disabled. When cleared to a 0, the oscillator is enabled.

Please note that the initial power on state of all registers is not defined. Therefore it is important to enable the oscillator (CH bit=0) during initial configuration.

The DS1307/DS1308 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10 hour bit (20-23 hours).

DS1307/DS1308 TIMEKEEPER REGISTERS Figure 3

	BIT7							BIT0	
00H	CH	10 SECONDS			SECONDS				00–59
	X	10 MINUTES			MINUTES				00–59
	X	12/24	10 HR / A/P	10 HR	HOURS				01–12 / 00–23
	X	X	X	X	X	DAY			1–7
	X	X	10 DATE		DATE				01–28/29 / 01–30 / 01–31
	X	X	X	10 MONTH	MONTH				01–12
	10 YEAR				YEAR				00–99
07H	OUT	X	X	SQWE	X	X	RS1	RS0	

CONTROL REGISTER

The DS1307/DS1308 Control Register is used to control the operation of the SQW/OUT pin.

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

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

SQWE (Square Wave Enable): This bit, when set to a logic 1, will enable the oscillator output. The frequency of the square wave output depends upon the value of the RS0 and RS1 bits.

RS (Rate Select): These bits control the frequency of the square wave output when the square wave output has been enabled. Table 1 lists the square wave frequencies that can be selected with the RS bits.

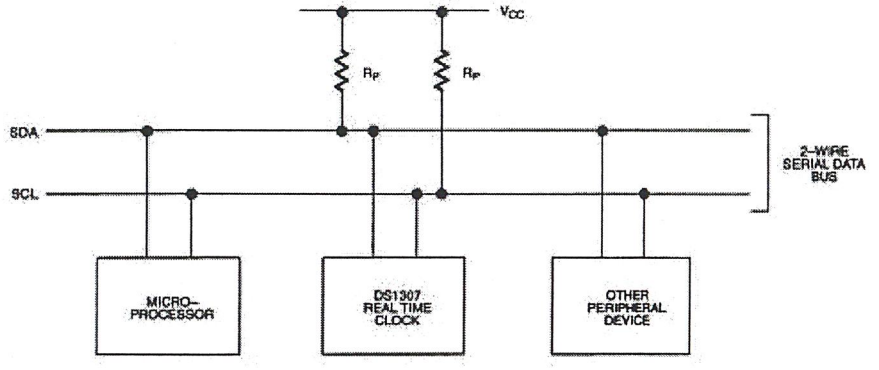
SQUAREWAVE OUTPUT FREQUENCY Table 1

RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1 Hz
0	1	4.096 kHz
1	0	8.192 kHz
1	1	32.768 kHz

2-WIRE SERIAL DATA BUS

The DS1307 supports a bi-directional 2-wire bus and data transmission protocol. A device that sends data onto the bus is defined as a transmitter and a device receiving data as a receiver. The device that controls the message is called a master. The devices that are controlled by the master are referred to as slaves. The bus must be controlled by a master device which generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions. The DS1307/DS1308 operates as a slave on the 2-wire bus. A typical bus configuration using this 2-wire protocol is show in Figure 4.

TYPICAL 2-WIRE BUS CONFIGURATION Figure 4



Figures 5, 6, and 7 detail how data is transferred on the 2-wire bus.

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

Accordingly, the following bus conditions have been defined:

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

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

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

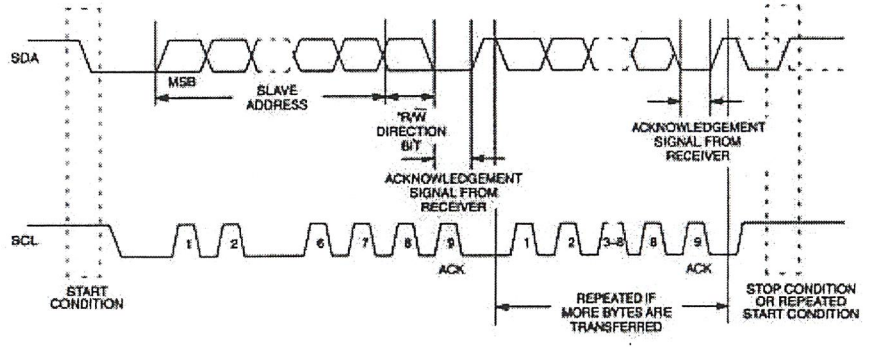
Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal. The data on the line must be changed during the LOW period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between START and STOP conditions is not limited, and is determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a ninth bit. Within the 2-wire bus specifications a regular mode (100 kHz clock rate) and a fast mode (400 kHz clock rate) are defined. The DS1307/DS1308 operates in the regular mode (100 kHz) only.

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

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

DATA TRANSFER ON 2-WIRE SERIAL BUS Figure 5



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

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

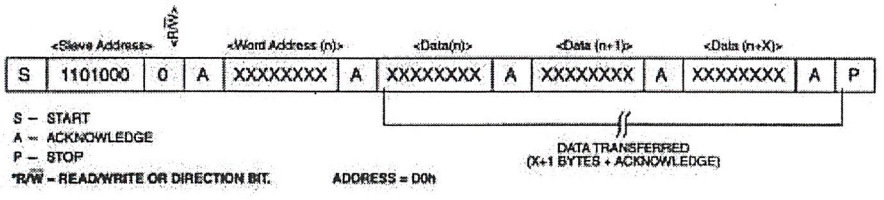
2. **Data transfer from a slave transmitter to a master receiver.** The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. This is followed by the slave transmitting a number of data bytes. The master returns an acknowledge bit after all received bytes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

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

The DS1307/DS1308 may operate in the following two modes:

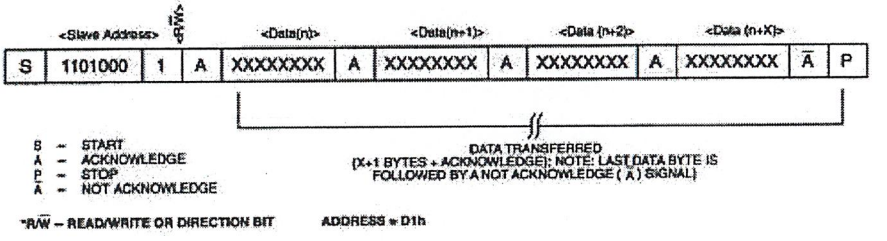
1. **Slave receiver mode (DS1307/DS1308 write mode):** Serial data and clock are received through SDA and SCL. After each byte is received an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and *direction bit (See Figure 6). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7 bit DS1307/DS1308 address, which is 1101000, followed by the *direction bit (R/$\overline{W}$) which, for a write, is a 0. After receiving and decoding the address byte the device outputs an acknowledge on the SDA line. After the DS1307/DS1308 acknowledges the slave address + write bit, the master transmits a register address to the DS1307/DS1308 This will set the register pointer on the DS1307/DS1308. The master will then begin transmitting each byte of data with the DS1307/DS1308 acknowledging each byte received. The master will generate a stop condition to terminate the data write.

DATA WRITE - SLAVE RECEIVER MODE Figure 6



2. **Slave transmitter mode (DS1307/DS1308 read mode):** The first byte is received and handled as in the slave receiver mode. However, in this mode, the *direction bit will indicate that the transfer direction is reversed. Serial data is transmitted on SDA by the DS1307/DS1308 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (See Figure 7). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7-bit DS1307/DS1308 address, which is 1101000, followed by the *direction bit ($R/\overline{W}$) which, for a read, is a 1. After receiving and decoding the address byte the device inputs an acknowledge on the SDA line. The DS1307/DS1308 then begins to transmit data starting with the register address pointed to by the register pointer. If the register pointer is not written to before the initiation of a read mode the first address that is read is the last one stored in the register pointer. The DS1307/DS1308 must receive a Not Acknowledge to end a read.

DATA READ - SLAVE TRANSMITTER MODE Figure 7



ABSOLUTE MAXIMUM RATINGS*

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

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

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

RECOMMENDED DC OPERATING CONDITIONS (0°C to 70°C)

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

DC ELECTRICAL CHARACTERISTICS (0°C to 70°C; V_{CC} =4.5V to 5.5V)

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

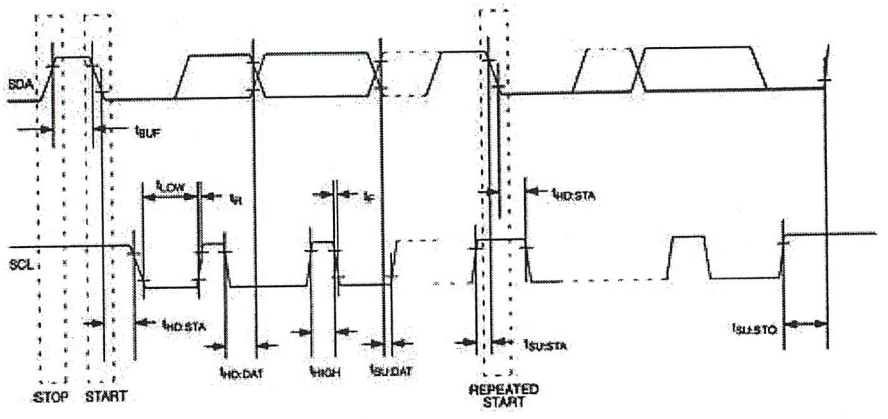
AC ELECTRICAL CHARACTERISTICS (0°C to 70°C; V_{CC} =4.5V to 5.5V)

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

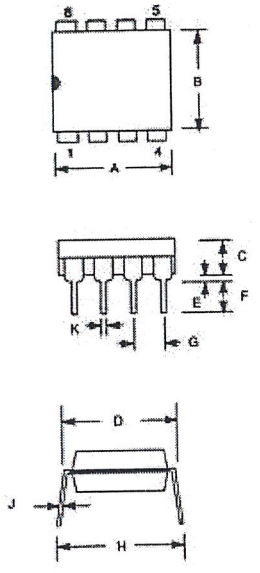
NOTES:

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

TIMING DIAGRAM Figure 8

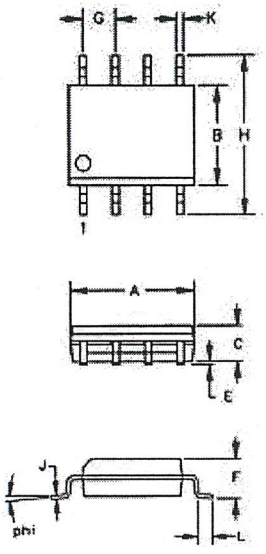


DS1307 64 X 8 SERIAL REAL TIME CLOCK
8-PIN DIP MECHANICAL DIMENSIONS



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

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

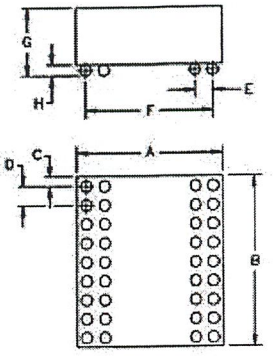


PKG	8-PIN (150 MIL)	
DIM	MIN	MAX
A IN.	0.188	0.196
MM	4.78	4.98
B IN.	0.150	0.158
MM	3.81	4.01
C IN.	0.048	0.062
MM	1.22	1.57
E IN.	0.004	0.010
MM	0.10	0.25
F IN.	0.053	0.069
MM	1.35	1.75
G IN.	0.050 BSC	
MM	1.27 BSC	
H IN.	0.230	0.244
MM	5.84	6.20
J IN.	0.007	0.011
MM	0.18	0.28
K IN.	0.012	0.020
MM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

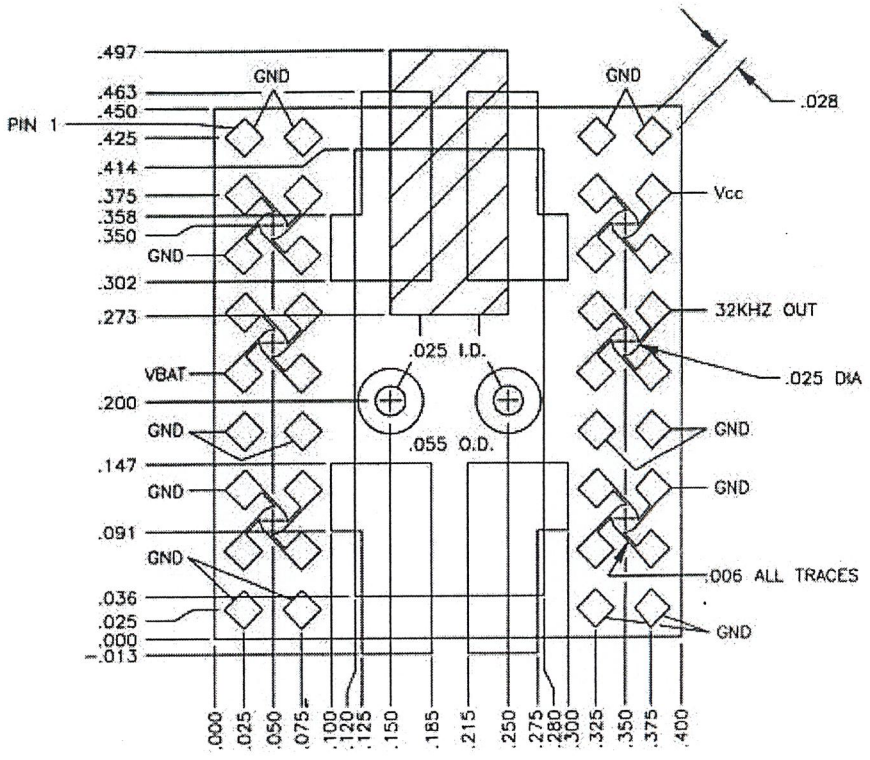
56-G2008-001

DS1308 MECHANICAL DIMENSIONS

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



DS1308 RECOMMENDED LAYOUT LAND PATTERN



DALLAS SEMICONDUCTOR

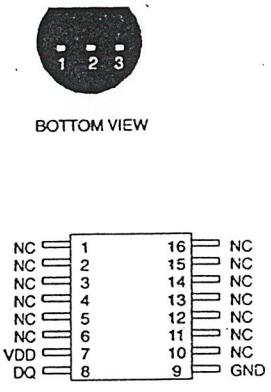
DS1820
1–Wire™ Digital Thermometer

FEATURES

- Unique 1–Wire™ interface requires only one port pin for communication
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line
- Zero standby power required
- Measures temperatures from –55°C to +125°C in 0.5°C increments. Fahrenheit equivalent is –67°F to +257°F in 0.9°F increments
- Temperature is read as a 9–bit digital value.
- Converts temperature to digital word in 200 ms (typ.)
- User–definable, nonvolatile temperature alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

PIN ASSIGNMENT

DS1820
PR35 PACKAGE
See Mech. Drawings Section



DS1820S
16–PIN SSOP
See Mech. Drawings Section

PIN DESCRIPTION

GND	– Ground
DQ	– Data In/Out
V_{DD}	– Optional V_{DD}
NC	– No Connect

DESCRIPTION

The DS1820 Digital Thermometer provides 9–bit temperature readings which indicate the temperature of the device.

Information is sent to/from the DS1820 over a 1–Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS1820. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

Because each DS1820 contains a unique silicon serial number, multiple DS1820s can exist on the same 1–Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and in process monitoring and control.

DETAILED PIN DESCRIPTION

PIN 16–PIN SSOP	PIN PR35	SYMBOL	DESCRIPTION
9	1	GND	**Ground.**
8	2	DQ	**Data Input/Output pin.** For 1–Wire operation: Open drain. (See "Parasite Power" section.)
7	3	V_{DD}	**Optional V_{DD} pin.** See "Parasite Power" section for details of connection.

DS1820S (16–pin SSOP): All pins not specified in this table are not to be connected.

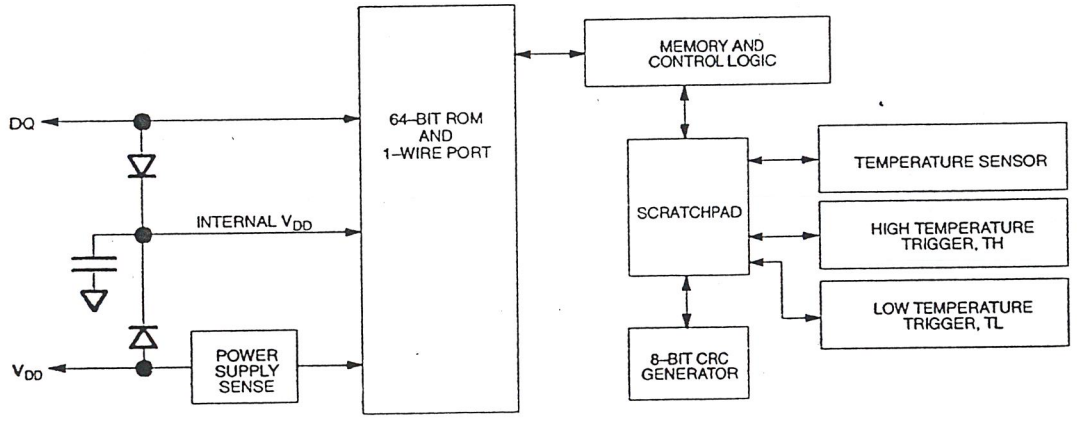
OVERVIEW

The block diagram of Figure 1 shows the major components of the DS1820. The DS1820 has three main data components: 1) 64–bit lasered ROM, 2) temperature sensor, and 3) nonvolatile temperature alarm triggers TH and TL. The device derives its power from the 1–Wire communication line by storing energy on an internal capacitor during periods of time when the signal line is high and continues to operate off this power source during the low times of the 1–Wire line until it returns high to replenish the parasite (capacitor) supply. As an alternative, the DS1820 may also be powered from an external 5 volts supply.

Communication to the DS1820 is via a 1–Wire port. With the 1–Wire port, the memory and control functions will not be available before the ROM function protocol has been established. The master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM, or 5) Alarm Search. These commands operate on the 64–bit lasered ROM portion of each device and can single out a specific device if many are present on the 1–Wire line as well as indicate to the Bus Master how many and what types of devices are present. After a ROM function sequence has been successfully executed, the memory and control functions are accessible and the master may then provide any one of the six memory and control function commands.

One control function command instructs the DS1820 to perform a temperature measurement. The result of this measurement will be placed in the DS1820's scratchpad memory, and may be read by issuing a memory function command which reads the contents of the scratchpad memory. The temperature alarm triggers TH and TL consist of one byte EEPROM each. If the alarm search command is not applied to the DS1820, these registers may be used as general purpose user memory. Writing TH and TL is done using a memory function command. Read access to these registers is through the scratchpad. All data is read and written least significant bit first.

DS1820 BLOCK DIAGRAM Figure 1



PARASITE POWER

The block diagram (Figure 1) shows the parasite powered circuitry. This circuitry "steals" power whenever the I/O or V_{DD} pins are high. I/O will provide sufficient power as long as the specified timing and voltage requirements are met (see the section titled "1–Wire Bus System"). The advantages of parasite power are two–fold: 1) by parasiting off this pin, no local power source is needed for remote sensing of temperature, and 2) the ROM may be read in absence of normal power.

In order for the DS1820 to be able to perform accurate temperature conversions, sufficient power must be provided over the I/O line when a temperature conversion is taking place. Since the operating current of the DS1820 is up to 1 mA, the I/O line will not have sufficient drive due to the 5K pull–up resistor. This problem is particularly acute if several DS1820's are on the same I/O and attempting to convert simultaneously.

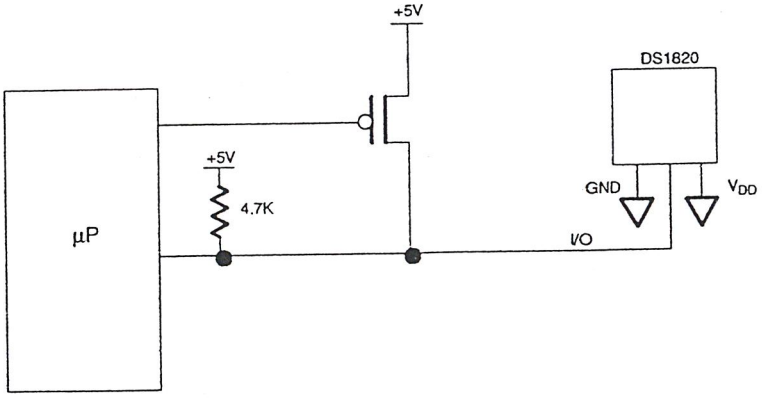
There are two ways to assure that the DS1820 has sufficient supply current during its active conversion cycle. The first is to provide a strong pull–up on the I/O line whenever temperature conversions or copies to the E^2 memory are taking place. This may be accomplished by using a MOSFET to pull the I/O line directly to the power supply as shown in Figure 2. The I/O line must be switched over to the strong pull–up within 10 µs maximum after issuing any protocol that involves copying to the E^2 memory or initiates temperature conversions. When using the parasite power mode, the V_{DD} pin must be tied to ground.

Another method of supplying current to the DS1820 is through the use of an external power supply tied to the V_{DD} pin, as shown in Figure 3. The advantage to this is that the strong pull–up is not required on the I/O line, and the bus master need not be tied up holding that line high during temperature conversions. This allows other data traffic on the 1–Wire bus during the conversion time. In addition, any number of DS1820's may be placed on the 1–Wire bus, and if they all use external power, they may all simultaneously perform temperature conversions by issuing the Skip ROM command and then issuing the Convert T command. Note that as long as the external power supply is active, the GND pin may not be floating.

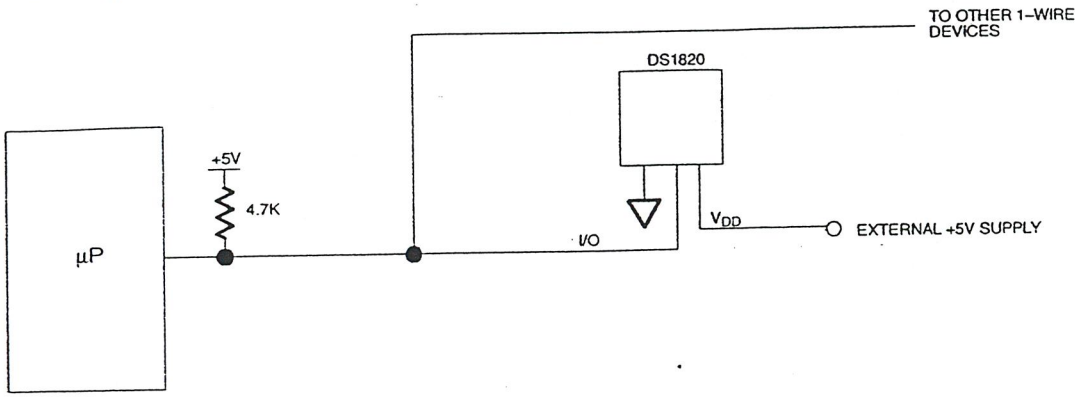
The use of parasite power is not recommended above 100°C, since it may not be able to sustain communications given the higher leakage currents the DS1820 exhibits at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that V_{DD} be applied to the DS1820.

For situations where the bus master does not know whether the DS1820's on the bus are parasite powered or supplied with external V_{DD}, a provision is made in the DS1820 to signal the power supply scheme used. The bus master can determine if any DS1820's are on the bus which require the strong pull–up by sending a Skip ROM protocol, then issuing the read power supply command. After this command is issued, the master then issues read time slots. The DS1820 will send back "0" on the 1–Wire bus if it is parasite powered; it will send back a "1" if it is powered from the V_{DD} pin. If the master receives a "0", it knows that it must supply the strong pull–up on the I/O line during temperature conversions. See "Memory Command Functions" section for more detail on this command protocol.

STRONG PULL–UP FOR SUPPLYING DS1820 DURING TEMPERATURE CONVERSION Figure 2



USING V_{DD} TO SUPPLY TEMPERATURE CONVERSION CURRENT Figure 3



OPERATION – MEASURING TEMPERATURE

The DS1820 measures temperature through the use of an on–board proprietary temperature measurement technique. A block diagram of the temperature measurement circuitry is shown in Figure 4.

The DS1820 measures temperature by counting the number of clock cycles that an oscillator with a low temperature coefficient goes througl. during a gate period determined by a high temperature coefficient oscillator. The counter is preset with a base count that corresponds to –55°C. If the counter reaches zero before the gate period is over, the temperature register, which is also preset to the –55°C value, is incremented, indicating that the temperature is higher than –55°C.

At the same time, the counter is then preset with a value determined by the slope accumulator circuitry. This circuitry is needed to compensate for the parabolic behavior of the oscillators over temperature. The counter is then clocked again until it reaches zero. If the gate period is still not finished, then this process repeats.

The slope accumulator is used to compensate for the non–linear behavior of the oscillators over temperature, yielding a high resolution temperature measurement. This is done by changing the number of counts necessary for the counter to go through for each incremental degree in temperature. To obtain the desired resolution, therefore, both the value of the counter and the number of counts per degree C (the value of the slope accumulator) at a given temperature must be known.

Internally, this calculation is done inside the DS1820 to provide 0.5°C resolution. The temperature reading is provided in a 16–bit, sign–extended two's complement reading. Table 1 describes the exact relationship of output data to measured temperature. The data is transmitted serially over the 1–Wire interface. The DS1820 can measure temperature over the range of –55°C to +125°C in 0.5°C increments. For Fahrenheit usage, a lookup table or conversion factor must be used.

Note that temperature is represented in the DS1820 in terms of a $^1/_2$°C LSB, yielding the following 9–bit format:

MSB								LSB
1	1	1	0	0	1	1	1	0

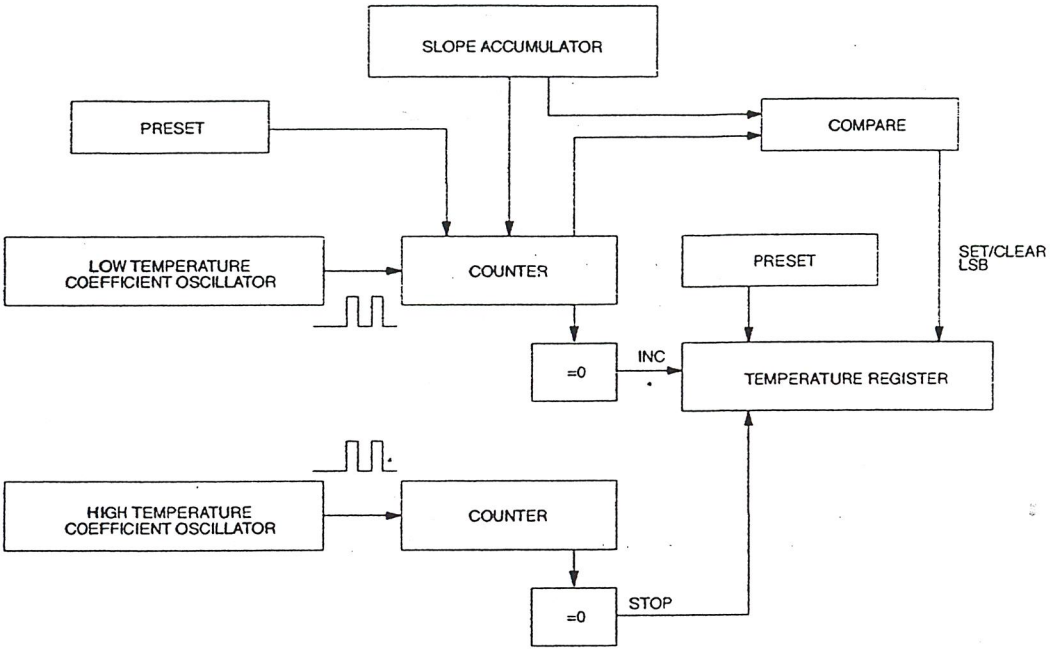
= –25°C

The most significant (sign) bit is duplicated into all of the bits in the upper MSB of the two–byte temperature register in memory. This "sign–extension" yields the 16–bit temperature readings as shown in Table 1.

Higher resolutions may be obtained by the following procedure. First, read the temperature, and truncate the 0.5°C bit (the LSB) from the read value. This value is TEMP_READ. The value left in the counter may then be read. This value is the count remaining (COUNT_REMAIN) after the gate period has ceased. The last value needed is the number of counts per degree C (COUNT_PER_C) at that temperature. The actual temperature may be then be calculated by the user using the following:

$$\text{TEMPERATURE} = \text{TEMP\_READ} - 0.25 + \frac{(\text{COUNT\_PER\_C} - \text{COUNT\_REMAIN})}{\text{COUNT\_PER\_C}}$$

TEMPERATURE MEASURING CIRCUITRY Figure 4



TEMPERATURE/DATA RELATIONSHIPS Table 1

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	00000000 11111010	00FA
+25°C	00000000 00110010	0032h
$+^1/_2$°C	00000000 00000001	0001h
+0°C	00000000 00000000	0000h
$-^1/_2$°C	11111111 11111111	FFFFh
–25°C	11111111 11001110	FFCEh
–55°C	11111111 10010010	FF92h

OPERATION – ALARM SIGNALING

After the DS1820 has performed a temperature conversion, the temperature value is compared to the trigger values stored in TH and TL. Since these registers are 8–bit only, the 0.5°C bit is ignored for comparison. The most significant bit of TH or TL directly corresponds to the sign bit of the 16–bit temperature register. If the result of a temperature measurement is higher than TH or lower than TL, an alarm flag inside the device is set. This flag is updated with every temperature measurement. As long as the alarm flag is set, the DS1820 will respond to the alarm search command. This allows many DS1820s to be connected in parallel doing simultaneous temperature measurements. If somewhere the temperature exceeds the limits, the alarming device(s) can be identified and read immediately without having to read non–alarming devices.

64–BIT LASERED ROM

Each DS1820 contains a unique ROM code that is 64–bits long. The first eight bits are a 1–Wire family code (DS1820 code is 10h). The next 48 bits are a unique serial number. The last eight bits are a CRC of the first 56 bits. (See Figure 5.) The 64–bit ROM and ROM Function Control section allow the DS1820 to operate as a 1–Wire device and follow the 1–Wire protocol detailed in the section "1–Wire Bus System". The functions required to control sections of the DS1820 are not accessible until the ROM function protocol has been satisfied. This protocol is described in the ROM function protocol flowchart (Figure 6). The 1–Wire bus master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM, or 5) Alarm Search. After a ROM functions sequence has been successfully executed, the functions specific to the DS1820 are accessible and the bus master may then provide and one of the six memory and control function commands.

CRC GENERATION

The DS1820 has an 8–bit CRC stored in the most significant byte of the 64–bit ROM. The bus master can compute a CRC value from the first 56–bits of the 64–bit ROM and compare it to the value stored within the DS1820 to determine if the ROM data has been received error–free by the bus master. The equivalent polynomial function of this CRC is:

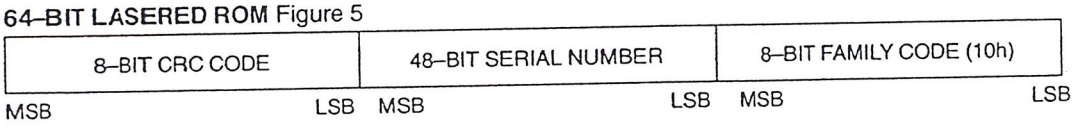
$$CRC = X^8 + X^5 + X^4 + 1$$

The DS1820 also generates an 8–bit CRC value using the same polynomial function shown above and provides this value to the bus master to validate the transfer of data bytes. In each case where a CRC is used for data transfer validation, the bus master must calculate a CRC value using the polynomial function given above and compare the calculated value to either the 8–bit CRC value stored in the 64–bit ROM portion of the DS1820 (for ROM reads) or the 8–bit CRC value computed within the DS1820 (which is read as a ninth byte when the scratchpad is read). The comparison of CRC values and decision to continue with an operation are determined entirely by the bus master. There is no circuitry inside the DS1820 that prevents a command sequence from proceeding if the CRC stored in or calculated by the DS1820 does not match the value generated by the bus master.

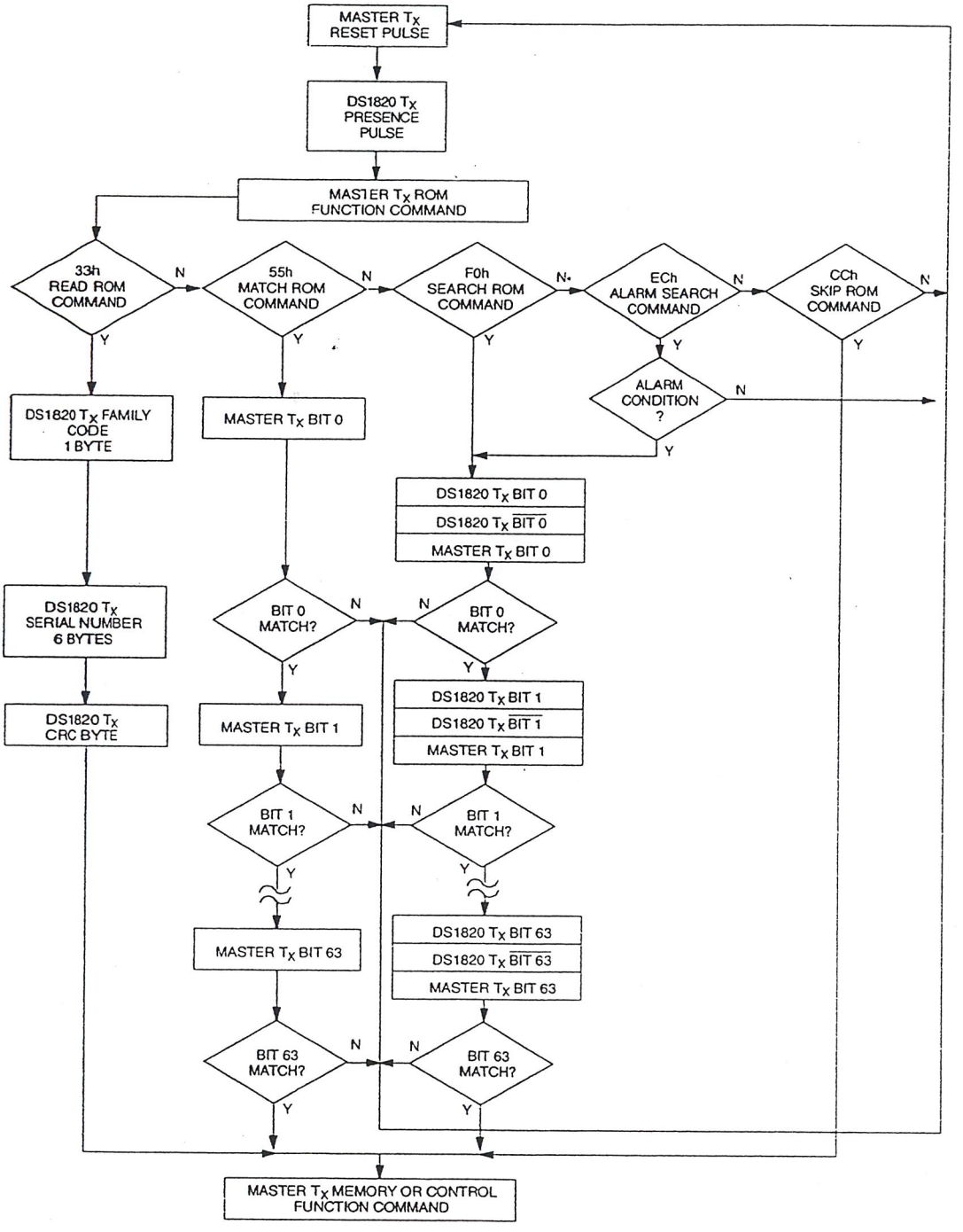
The 1–Wire CRC can be generated using a polynomial generator consisting of a shift register and XOR gates as shown in Figure 7. Additional information about the Dallas 1–Wire Cyclic Redundancy Check is available in Application Note 27 entitled "Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor Touch Memory Products".

The shift register bits are initialized to zero. Then starting with the least significant bit of the family code, one bit at a time is shifted in. After the 8th bit of the family code has been entered, then the serial number is entered. After the 48th bit of the serial number has been entered, the shift register contains the CRC value. Shifting in the eight bits of CRC should return the shift register to all zeros.

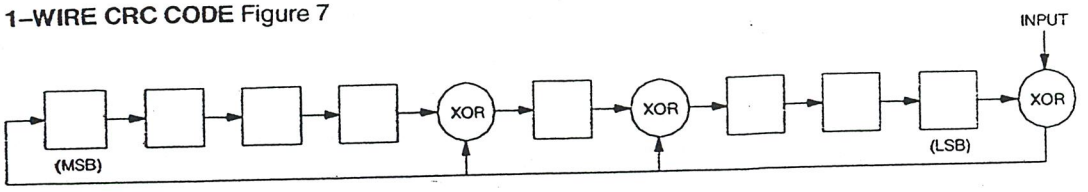
64–BIT LASERED ROM Figure 5



ROM FUNCTIONS FLOW CHART Figure 6



1–WIRE CRC CODE Figure 7



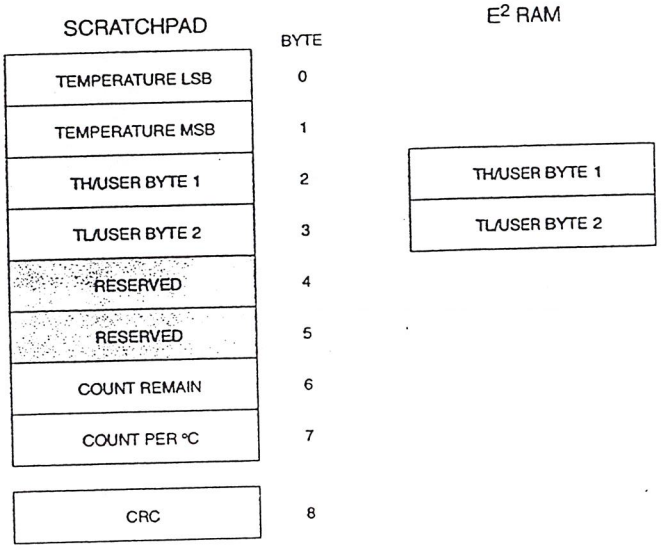
MEMORY

The DS1820's memory is organized as shown in Figure 8. The memory consists of a scratchpad RAM and a nonvolatile, electrically erasable (E^2) RAM, which stores the high and low temperature triggers TH and TL. The scratchpad helps insure data integrity when communicating over the 1–Wire bus. Data is first written to the scratchpad where it can be read back. After the data has been verified, a copy scratchpad command will transfer the data to the nonvolatile (E^2) RAM. This process insures data integrity when modifying the memory.

The scratchpad is organized as eight bytes of memory. The first two bytes contain the measured temperature information. The third and fourth bytes are volatile copies of TH and TL and are refreshed with every power–on reset. The next two bytes are not used; upon reading back, however, they will appear as all logic 1's. The seventh and eighth bytes are count registers, which may be used in obtaining higher temperature resolution (see "Operation–measuring Temperature" section).

There is a ninth byte which may be read with a Read Scratchpad command. This byte contains a cyclic redundancy check (CRC) byte which is the CRC over all of the eight previous bytes. This CRC is implemented in the fashion described in the section titled "CRC Generation".

DS1820 MEMORY MAP Figure 8



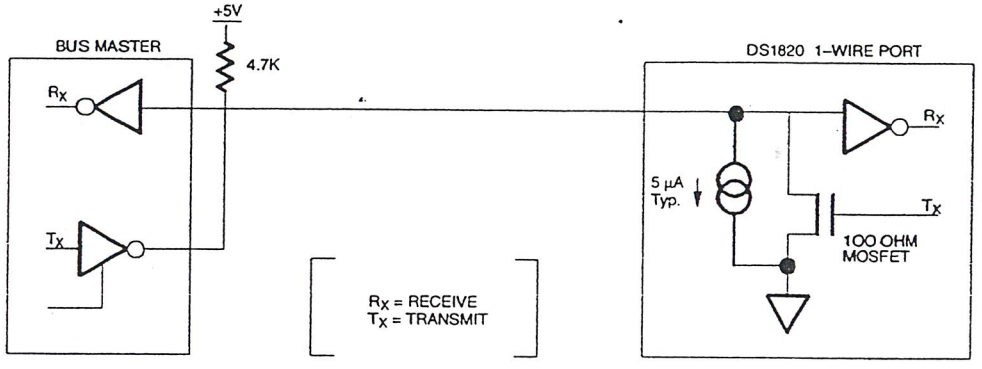
1–WIRE BUS SYSTEM

The 1–Wire bus is a system which has a single bus master and one or more slaves. The DS1820 behaves as a slave. The discussion of this bus system is broken down into three topics: hardware configuration, transaction sequence, and 1–Wire signaling (signal types and timing).

HARDWARE CONFIGURATION

The 1–Wire bus has only a single line by definition; it is important that each device on the bus be able to drive it at the appropriate time. To facilitate this, each device attached to the 1–Wire bus must have open drain or 3–state outputs. The 1–Wire port of the DS1820 (I/O pin) is open drain with an internal circuit equivalent to that shown in Figure 9. A multidrop bus consists of a 1–Wire bus with multiple slaves attached. The 1–Wire bus requires a pullup resistor of approximately 5KΩ.

HARDWARE CONFIGURATION Figure 9



The idle state for the 1–Wire bus is high. If for any reason a transaction needs to be suspended, the bus MUST be left in the idle state if the transaction is to resume. Infinite recovery time can occur between bits so long as the 1–Wire bus is in the inactive (high) state during the recovery period. If this does not occur and the bus is left low for more than 480 µs, all components on the bus will be reset.

TRANSACTION SEQUENCE

The protocol for accessing the DS1820 via the 1–Wire port is as follows:

- Initialization
- ROM Function Command
- Memory Function Command
- Transaction/Data

INITIALIZATION

All transactions on the 1–Wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s).

The presence pulse lets the bus master know that the DS1820 is on the bus and is ready to operate. For more details, see the "1–Wire Signaling" section.

ROM FUNCTION COMMANDS

Once the bus master has detected a presence, it can issue one of the five ROM function commands. All ROM function commands are 8–bits long. A list of these commands follows (refer to flowchart in Figure 6):

Read ROM [33h]

This command allows the bus master to read the DS1820's 8–bit family code, unique 48–bit serial number, and 8–bit CRC. This command can only be used if there is a single DS1820 on the bus. If more than one slave is present on the bus, a data collision will occur when all slaves try to transmit at the same time (open drain will produce a wired AND result).

Match ROM [55h]

The match ROM command, followed by a 64–bit ROM sequence, allows the bus master to address a specific DS1820 on a multidrop bus. Only the DS1820 that exactly matches the 64–bit ROM sequence will respond to the following memory function command. All slaves that do not match the 64–bit ROM sequence will wait for a reset pulse. This command can be used with a single or multiple devices on the bus.

Skip ROM [CCh]

This command can save time in a single drop bus system by allowing the bus master to access the memory functions without providing the 64–bit ROM code. If more than one slave is present on the bus and a read command is issued following the Skip ROM command, data collision will occur on the bus as multiple slaves transmit simultaneously (open drain pulldowns will produce a wired AND result).

Search ROM [F0h]

When a system is initially brought up, the bus master might not know the number of devices on the 1–Wire bus or their 64–bit ROM codes. The search ROM command allows the bus master to use a process of elimination to identify the 64–bit ROM codes of all slave devices on the bus.

Alarm Search [ECh]

The flowchart of this command is identical to the Search ROM command. However, the DS1820 will respond to this command only if an alarm condition has been encountered at the last temperature measurement. An alarm condition is defined as a temperature higher than TH or lower than TL. The alarm condition remains set as long as the DS1820 is powered up, or until another temperature measurement reveals a non–alarming value. For alarming, the trigger values stored in EEPROM are taken into account. If an alarm condition exists and the TH or TL settings are changed, another temperature conversion should be done to validate any alarm conditions.

Example of a ROM Search

The ROM search process is the repetition of a simple 3–step routine: read a bit, read the complement of the bit, then write the desired value of that bit. The bus master performs this simple, 3–step routine on each bit of the ROM. After one complete pass, the bus master knows the contents of the ROM in one device. The remaining number of devices and their ROM codes may be identified by additional passes.

The following example of the ROM search process assumes four different devices are connected to the same 1–Wire bus. The ROM data of the four devices is as shown:

ROM1	00110101...
ROM2	10101010...
ROM3	11110101...
ROM4	00010001...

The search process is as follows:

1. The bus master begins the initialization sequence by issuing a reset pulse. The slave devices respond by issuing simultaneous presence pulses.
2. The bus master will then issue the Search ROM command on the 1–Wire bus.
3. The bus master reads a bit from the 1–Wire bus. Each device will respond by placing the value of the first bit of their respective ROM data onto the 1–Wire bus. ROM1 and ROM4 will place a 0 onto the 1–Wire bus, i.e., pull it low. ROM2 and ROM3 will place a 1 onto the 1–Wire bus by allowing the line to stay high. The result is the logical AND of all devices on the line, therefore the bus master sees a 0. The bus master reads another bit. Since the Search ROM data command is being executed, all of the devices on the 1–Wire bus respond to this second read by placing the complement of the first bit of their respective ROM data onto the 1–Wire bus. ROM1 and ROM4 will place a 1 onto the 1–Wire, allowing the line to stay high. ROM2 and ROM3 will place a 0 onto the 1–Wire, thus it will be pulled low. The bus master again observes a 0 for the complement of the first ROM data bit. The bus master has determined that there are some devices on the 1–Wire bus that have a 0 in the first position and others that have a 1.

The data obtained from the two reads of the 3–step routine have the following interpretations:

00 There are still devices attached which have conflicting bits in this position.

01 All devices still coupled have a 0–bit in this bit position.

10 All devices still coupled have a 1–bit in this bit position.

11 There are no devices attached to the 1–Wire bus.

4. The bus master writes a 0. This deselects ROM2 and ROM3 for the remainder of this search pass, leaving only ROM1 and ROM4 connected to the 1–Wire bus.
5. The bus master performs two more reads and receives a 0 -bit followed by a 1–bit. This indicates that all devices still coupled to the bus have 0's as their second ROM data bit.
6. The bus master then writes a 0 to keep both ROM1 and ROM4 coupled.
7. The bus master executes two reads and receives two 0–bits. This indicates that both 1–bits and 0–bits exist as the third bit of the ROM data of the attached devices.
8. The bus master writes a 0–bit. This deselects ROM1 leaving ROM4 as the only device still connected.
9. The bus master reads the remainder of the ROM bits for ROM4 and continues to access the part if desired. This completes the first pass and uniquely identifies one part on the 1–Wire bus.
10. The bus master starts a new ROM search sequence by repeating steps 1 through 7.
11. The bus master writes a 1–bit. This decouples ROM4, leaving only ROM1 still coupled.
12. The bus master reads the remainder of the ROM bits for ROM1 and communicates to the underlying logic if desired. This completes the second ROM search pass, in which another of the ROMs was found.
13. The bus master starts a new ROM search by repeating steps 1 through 3.
14. The bus master writes a 1–bit. This deselects ROM1 and ROM4 for the remainder of this search pass, leaving only ROM2 and ROM3 coupled to the system.
15. The bus master executes two read time slots and receives two zeros.
16. The bus master writes a 0–bit. This decouples ROM3, and leaving only ROM2.
17. The bus master reads the remainder of the ROM bits for ROM2 and communicates to the underlying logic if desired. This completes the third ROM search pass, in which another of the ROMs was found.
18. The bus master starts a new ROM search by repeating steps 13 through 15.
19. The bus master writes a 1–bit. This decouples ROM2, leaving only ROM3.
20. The bus master reads the remainder of the ROM bits for ROM3 and communicates to the underlying logic if desired. This completes the fourth ROM search pass, in which another of the ROMs was found.

Note the following:

The bus master learns the unique ID number (ROM data pattern) of one 1–Wire device on each ROM Search operation. The time required to derive the part's unique ROM code is:

$$960\ \mu s + (8 + 3 \times 64)\ 61\ \mu s = 13.16\ ms$$

The bus master is therefore capable of identifying 75 different 1–Wire devices per second.

I/O SIGNALING

The DS1820 requires strict protocols to insure data integrity. The protocol consists of several types of signaling on one line: reset pulse, presence pulse, write 0, write 1, read 0, and read 1. All of these signals, with the exception of the presence pulse, are initiated by the bus master.

The initialization sequence required to begin any communication with the DS1820 is shown in Figure 11. A reset pulse followed by a presence pulse indicates the DS1820 is ready to send or receive data given the correct ROM command and memory function command.

The bus master transmits (TX) a reset pulse (a low signal for a minimum of 480 µs). The bus master then releases the line and goes into a receive mode (RX). The 1–Wire bus is pulled to a high state via the 5K pull–up resistor . After detecting the rising edge on the

I/O pin, the DS1820 waits 15–60 µs and then transmits the presence pulse (a low signal for 60–240 µs).

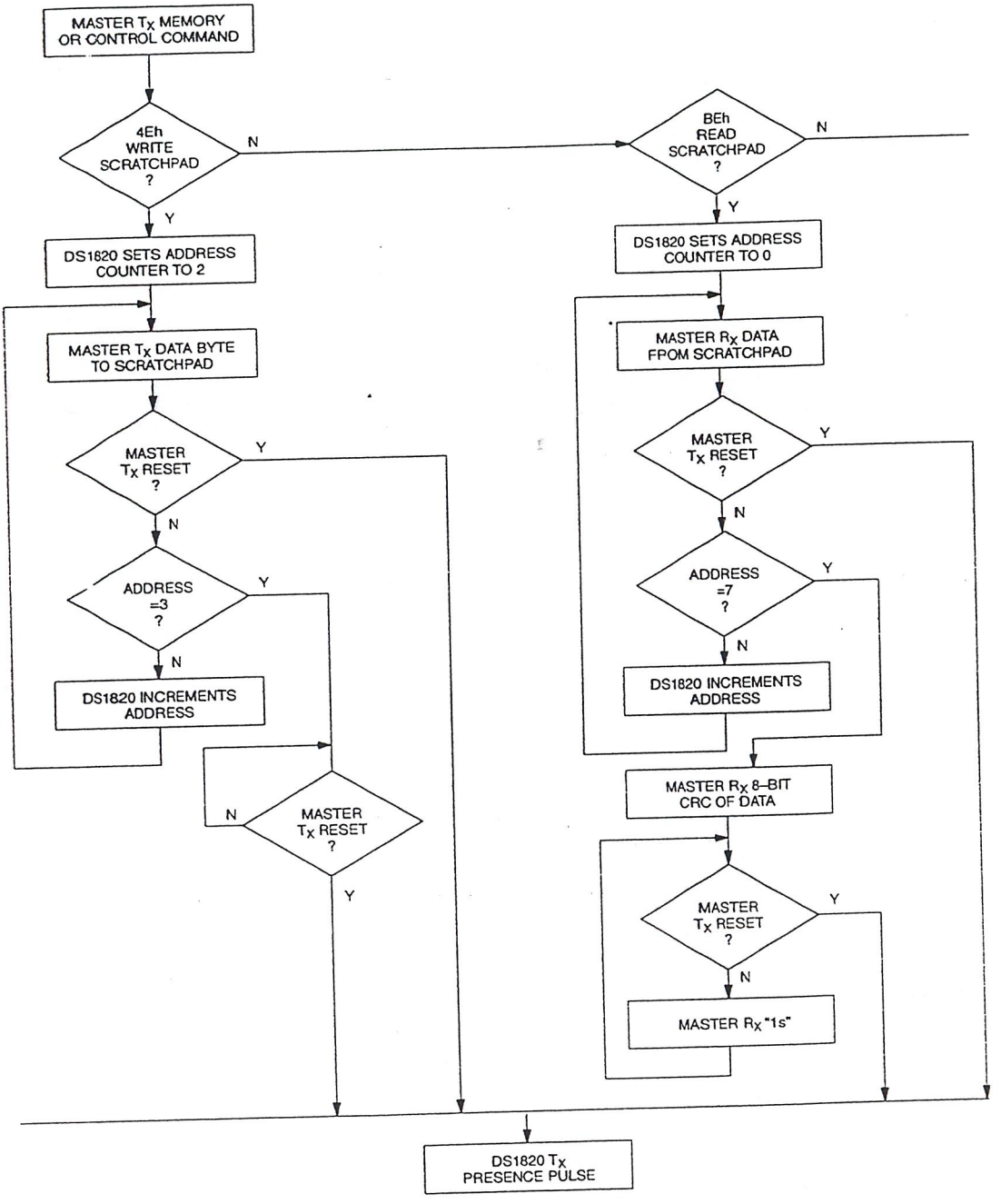
MEMORY COMMAND FUNCTIONS

The following command protocols are summarized in Table 2, and by the flowchart of Figure 10.

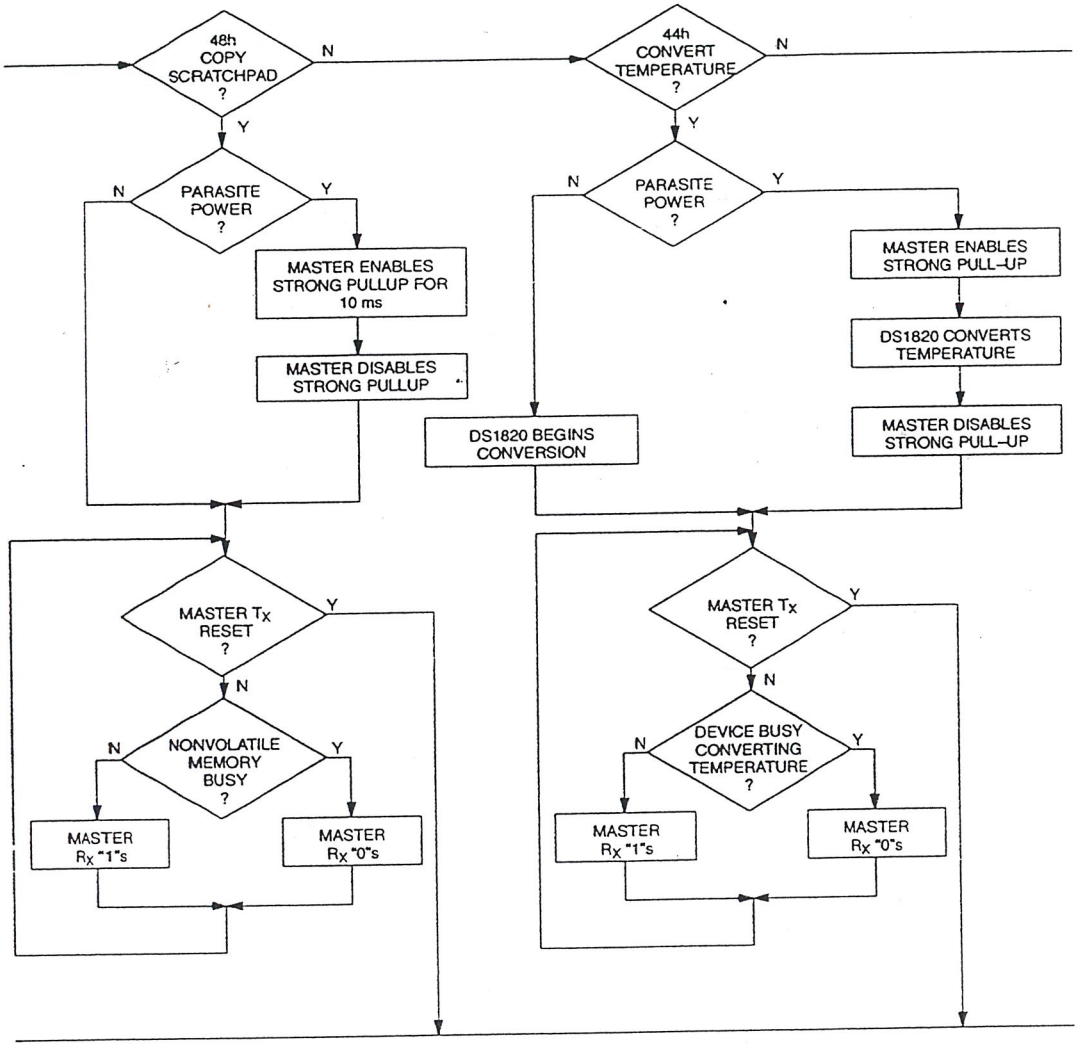
Write Scratchpad [4Eh]

This command writes to the scratchpad of the DS1820, starting at address 2. The next two bytes written will be saved in scratchpad memory, at address locations 2 and 3. Writing may be terminated at any point by issuing a reset.

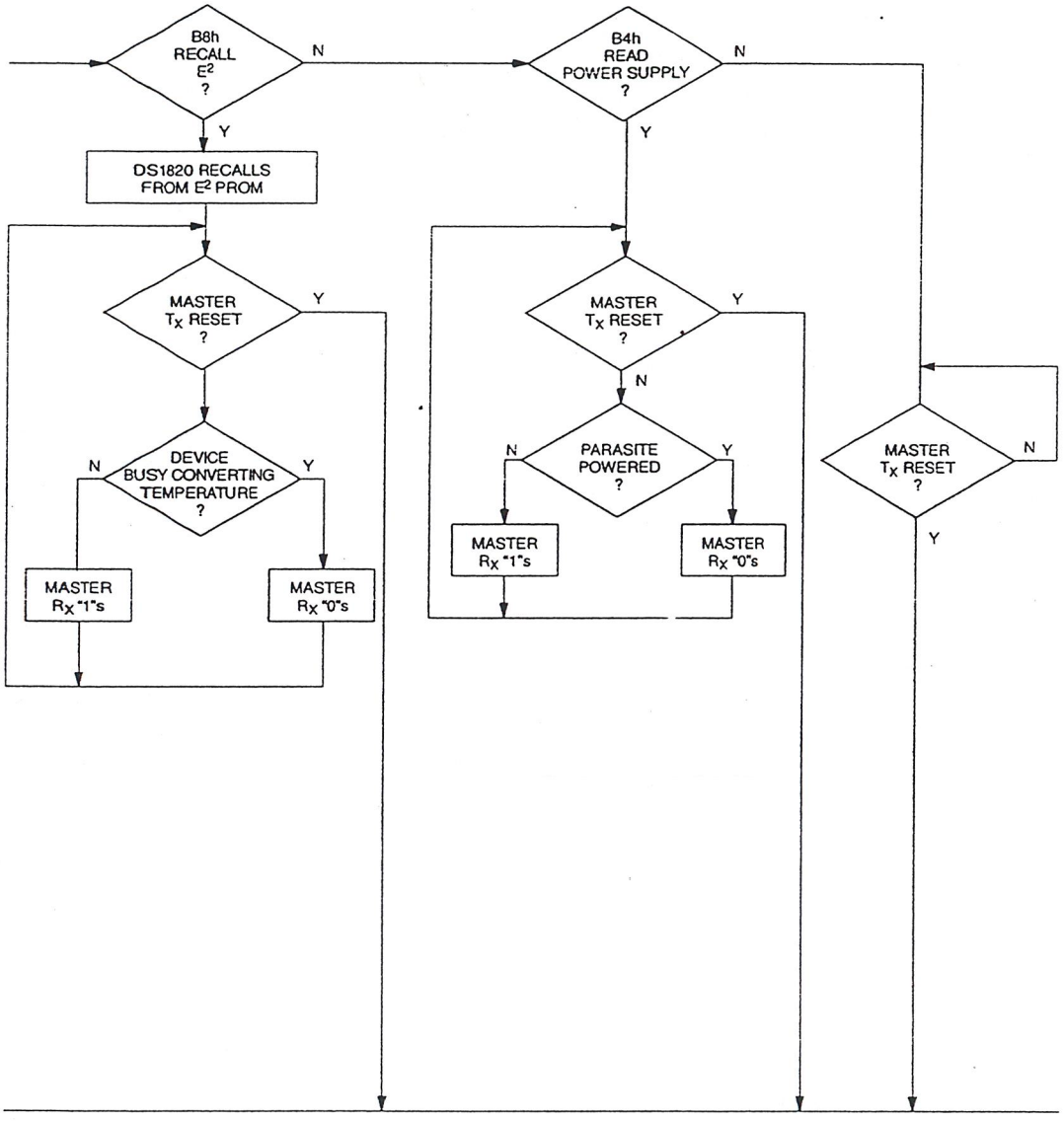
MEMORY FUNCTIONS FLOW CHART Figure 10



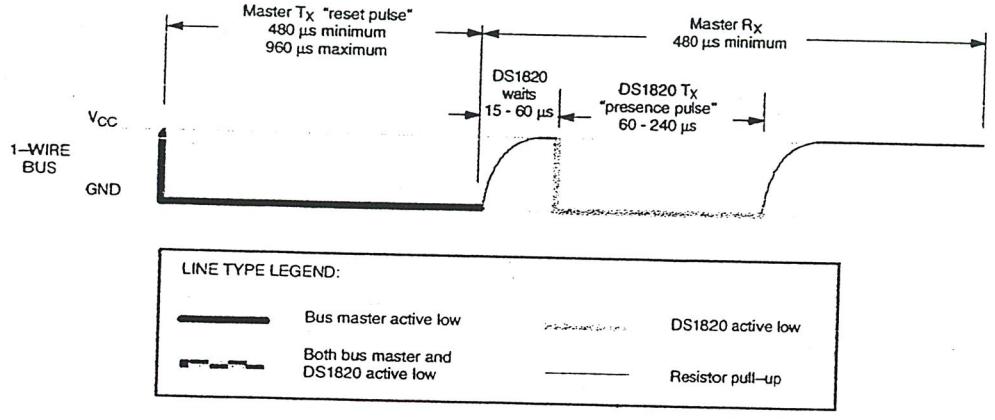
MEMORY FUNCTIONS FLOW CHART Figure 10 (cont'd)



MEMORY FUNCTIONS FLOW CHART Figure 10 (cont'd)



INITIALIZATION PROCEDURE "RESET AND PRESENCE PULSES" Figure 11



DS1820 COMMAND SET Table 2

INSTRUCTION	DESCRIPTION	PROTOCOL	1–WIRE BUS AFTER ISSUING PROTOCOL	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	<read temperature busy status>	1
MEMORY COMMANDS				
Read Scratchpad	Reads bytes from scratchpad and reads CRC byte.	BEh	<read data up to 9 bytes>	
Write Scratchpad	Writes bytes into scratchpad at addresses 2 and 3 (TH and TL temperature triggers).	4Eh	<write data into 2 bytes at addr. 2 and addr. 3>	
Copy Scratchpad	Copies scratchpad into nonvolatile memory (addresses 2 and 3 only).	48h	<read copy status>	2
Recall E^2	Recalls values stored in nonvolatile memory into scratchpad (temperature triggers).	B8h	<read temperature busy status>	
Read Power Supply	Signals the mode of DS1820 power supply to the master.	B4h	<read supply status>	

NOTES:

1. Temperature conversion takes up to 500 ms. After receiving the Convert T protocol, if the part does not receive power from the V_{DD} pin, the I/O line for the DS1820 must be held high for at least 500 ms to provide power during the conversion process. As such, no other activity may take place on the 1–Wire bus for at least this period after a Convert T command has been issued.

2. After receiving the Copy Scratchpad protocol, if the part does not receive power from the V_{DD} pin, the I/O line for the DS1820 must be held high for at least 10 ms to provide power during the copy process. As such, no other activity may take place on the 1–Wire bus for at least this period after a Copy Scratchpad command has been issued.

Read Scratchpad [BEh]
This command reads the contents of the scratchpad. Reading will commence at byte 0, and will continue through the scratchpad until the 9th (byte–8, CRC) byte is read. If not all locations are to be read, the master may issue a reset to terminate reading at any time.

Copy Scratchpad [48h]
This command copies the scratchpad into the E^2 memory of the DS1820, storing the temperature trigger bytes in nonvolatile memory. If the bus master issues read time slots following this command, the DS1820 will output "0" on the bus as long as it is busy copying the scratchpad to E^2; it will return a "1" when the copy process is complete. If parasite powered, the bus master has to enable a strong pull–up for at least 10 ms immediately after issuing this command.

Convert T [44h]
This command begins a temperature conversion. No further data is required. The temperature conversion will be performed and then the DS1820 will remain idle. If the bus master issues read time slots following this command, the DS1820 will output "0" on the bus as long as it is busy making a temperature conversion; it will return a "1" when the temperature conversion is complete. If parasite powered, the bus master has to enable a strong pullup for 500 ms immediately after issuing this command.

Recall E2 [B8h]
This command recalls the temperature trigger values stored in E^2 to the scratchpad. This recall operation happens automatically upon power–up to the DS1820 as well, so valid data is available in the scratchpad as soon as the device has power applied. With every read data time slot issued after this command has been sent, the device will output its temperature converter busy flag "0"=busy, "1"=ready.

Read Power Supply [B4h]
With every read data time slot issued after this command has been sent to the DS1820, the device will signal its power mode: "0"=parasite power, "1"=external power supply provided.

READ/WRITE TIME SLOTS
DS1820 data is read and written through the use of time slots to manipulate bits and a command word to specify the transaction.

Write Time Slots
A write time slot is initiated when the host pulls the data line from a high logic level to a low logic level. There are two types of write time slots: Write One time slots and Write Zero time slots. All write time slots must be a minimum of 60 µs in duration with a minimum of a one µs recovery time between individual write cycles.

The DS1820 samples the I/O line in a window of 15 µs to 60 µs after the I/O line falls. If the line is high, a Write One occurs. If the line is low, a Write Zero occurs (see Figure 12).

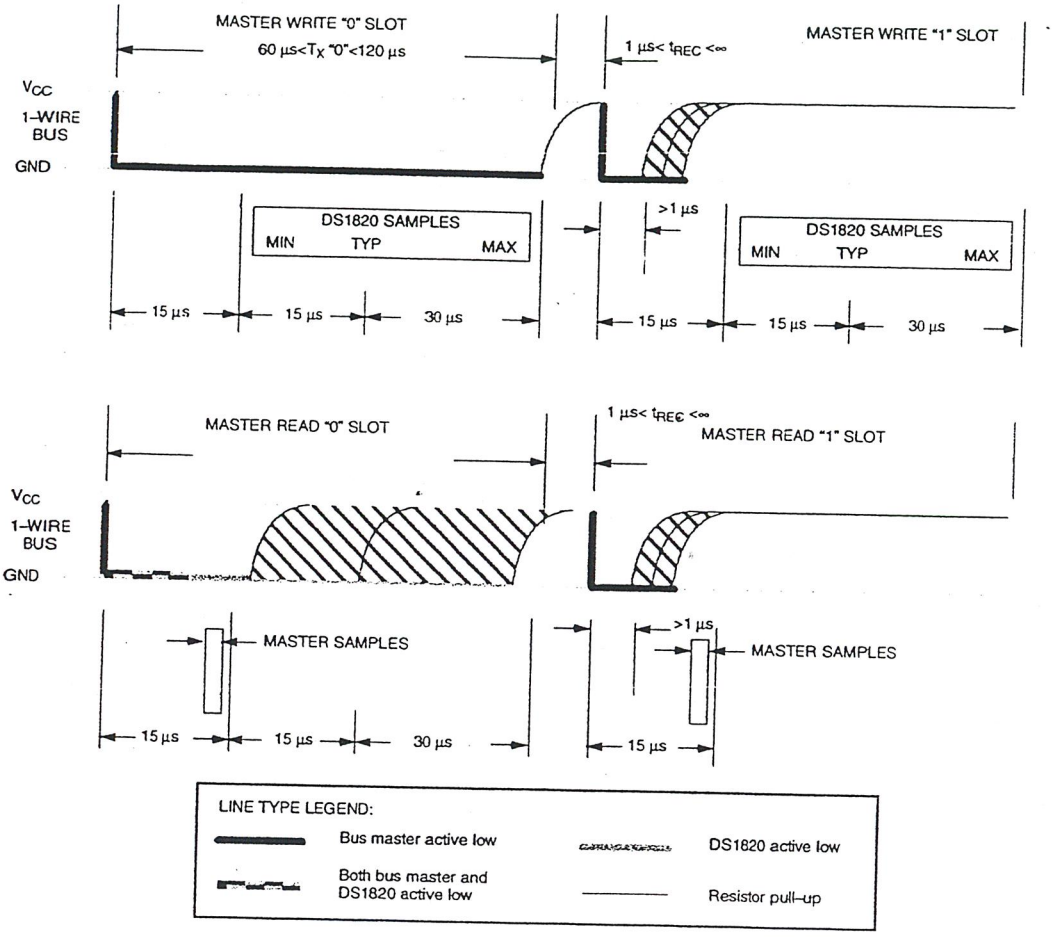
For the host to generate a Write One time slot, the data line must be pulled to a logic low level and then released, allowing the data line to pull up to a high level within 15 µs after the start of the write time slot.

For the host to generate a Write Zero time slot, the data line must be pulled to a logic low level and remain low for 60 µs.

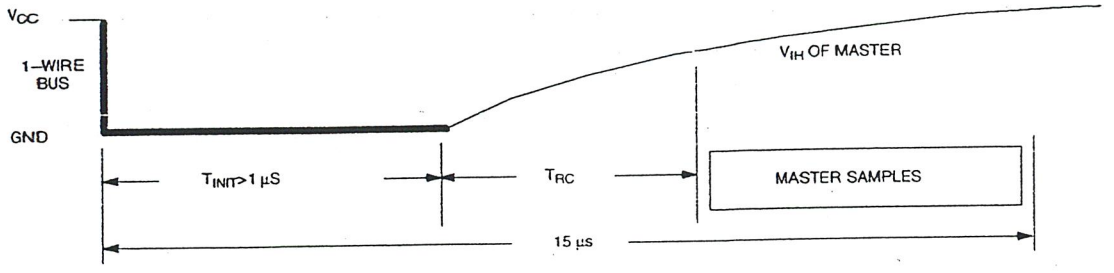
Read Time Slots
The host generates read time slots when data is to be read from the DS1820. A read time slot is initiated when the host pulls the data line from a logic high level to logic low level. The data line must remain at a low logic level for a minimum of one µs; output data from the DS1820 is valid for 15 µs after the falling edge of the read time slot. The host therefore must stop driving the I/O pin low in order to read its state 15 µs from the start of the read slot (see Figure 12). By the end of the read time slot, the I/O pin will pull back high via the external pull–up resistor. All read time slots must be a minimum of 60 µs in duration with a minimum of a one µs recovery time between individual read slots.

Figure 13 shows that the sum of T_{INIT}, T_{RC}, and T_{SAMPLE} must be less than 15 µs. Figure 14 shows that system timing margin is maximized by keeping T_{INIT} and T_{RC} as small as possible and by locating the master sample time towards the end of the 15 µs period.

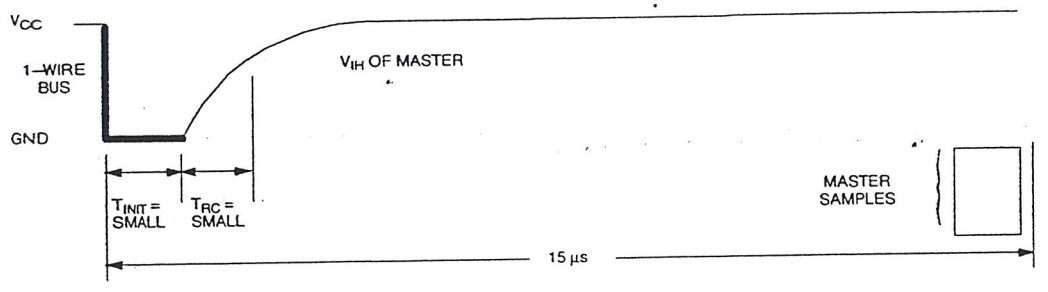
READ/WRITE TIMING DIAGRAM Figure 12

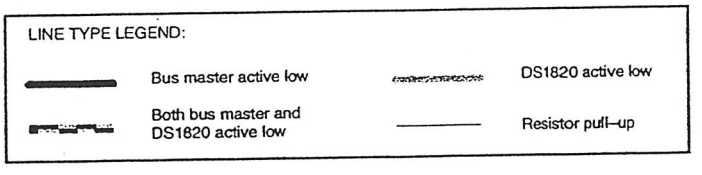


DETAILED MASTER READ "1" TIMING Figure 13



RECOMMENDED MASTER READ "1" TIMING Figure 14





Related Application Notes

The following Application Notes can be applied to the DS1820. These notes can be obtained from the Dallas Semiconductor "Application Note Book", via our website at http://www.dalsemi.com/, or through our faxback service at (214) 450–0441.

Application Note 27: "Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor Touch Memory Product"

Application Note 55: "Extending the Contact Range of Touch Memories"

Application Note 74: "Reading and Writing Touch Memories via Serial Interfaces"

Application Note 104: "Minimalist Temperature Control Demo"

Application Note 105: "High Resolution Temperature Measurement with Dallas Direct–to–Direct Temperature Sensors"

Application Note 106: "Complex MicroLANs"

Application Note 108: "MicroLAN – In the Long Run"

Sample 1–Wire subroutines that can be used in conjunction with AN74 can be downloaded from the website or our Anonymous FTP Site.

MEMORY FUNCTION EXAMPLE Table 3
Example: Bus Master initiates temperature conversion, then reads temperature (parasite power assumed).

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Reset pulse (480–960 µs).
RX	Presence	Presence pulse.
TX	55h	Issue "Match ROM" command.
TX	<64–bit ROM code>	Issue address for DS1820.
TX	44h	Issue "Convert T" command.
TX	<I/O LINE HIGH>	I/O line is held high for at least 500 ms by bus master to allow conversion to complete.
TX	Reset	Reset pulse.
RX	Presence	Presence pulse.
TX	55h	Issue "Match ROM" command.
TX	<64–bit ROM code>	Issue address for DS1820.
TX	BEh	Issue "Read Scratchpad" command.
RX	<9 data bytes>	Read entire scratchpad plus CRC; the master now recalculates the CRC of the eight data bytes received from the scratchpad, compares the CRC calculated and the CRC read. If they match, the master continues; if not, this read operation is repeated.
TX	Reset	Reset Pulse.
RX	Presence	Presence pulse, done.

MEMORY FUNCTION EXAMPLE Table 4

Example: Bus Master writes memory (parasite power and only one DS1820 assumed).

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Reset pulse.
RX	Presence	Presence pulse.
TX	CCh	Skip ROM command.
TX	4Eh	Write Scratchpad command.
TX	<2 data bytes>	Writes two bytes to scratchpad (TH and TL).
TX	Reset	Reset pulse.
RX	Presence	Presence pulse.
TX	CCh	Skip ROM command.
TX	BEh	Read Scratchpad command.
RX	<9 data bytes>	Read entire scratchpad plus CRC. The master now recalculates the CRC of the eight data bytes received from the scratchpad, compares the CRC and the two other bytes read back from the scratchpad. If data match, the master continues; if not, repeat the sequence.
TX	Reset	Reset pulse.
RX	Presence	Presence pulse.
TX	CCh	Skip ROM command.
TX	48h	Copy Scratchpad command; after issuing this command, the master must wait 6 ms for copy operation to complete.
TX	Reset	Reset pulse.
RX	Presence	Presence pulse, done.

MEMORY FUNCTION EXAMPLE Table 5
Example: Temperature conversion and interpolation (external power supply and only one DS1820 assumed).

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Reset pulse.
TR	Presence	Presence pulse.
TX	CCh	Skip ROM command.
TX	44h	Convert T command.
RX	<1 data byte>	Read busy flag eight times. The master continues reading one byte (or bit) after another until the data is FFh (all bits 1).
TX	Reset	Reset pulse.
RX	Presence	Presence pulse.
TX	CCh	Skip ROM command.
TX	BEh	Read Scratchpad command.
RX	<9 data bytes>	Read entire scratchpad plus CRC. The master now recalculates the CRC of the eight data bytes received from the scratchpad and compares both CRCs. If the CRCs match, the data is valid. The master saves the temperature value and stores the contents of the count register and count per °C register as COUNT_REMAIN and COUNT_PER_C, respectively.
TX	Reset	Reset pulse.
RX	Presence	Presence pulse, done.
–	–	CPU calculates temperature as described in the data sheet for higher resolution.

ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.5V to +7.0V
Operating Temperature	–55°C to +125°C
Storage Temperature	–55°C to +125°C
Soldering Temperature	260°C for 10 seconds

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

RECOMMENDED DC OPERATING CONDITIONS

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{DD}	I/O Functions	2.8	5.0	5.5	V	1, 2
		±1/2°C Accurate Temperature Conversions	4.3		5.5		
Data Pin	I/O		–0.5		+5.5	V	2
Logic 1	V_{IH}		2.0		V_{CC}+0.3	V	2, 3
Logic 0	V_{IL}		–0.3		+0.8	V	2, 4

DC ELECTRICAL CHARACTERISTICS

(–55°C to +125°C; V_{DD}=3.6V to 5.5V)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS	NOTES
Thermometer Error	t_{ERR}	–0°C to +70°C			±1/2	°C	
		–55°C to 0°C and +70°C to +125°C		See Typical Curve			1, 9, 10
Input Logic High	V_{IH}		2.2		5.5	V	2, 3
Input Logic Low	V_{IL}		–0.3		+0.8	V	2, 4
Sink Current	I_L	$V_{I/O}$=0.4V	–4.0			mA	2
Standby Current	I_Q			200	350	nA	8
Active Current	I_{DD}			1	1.5	mA	5, 6
Input Load Current	I_L			5		µA	7

AC ELECTRICAL CHARACTERISTICS:

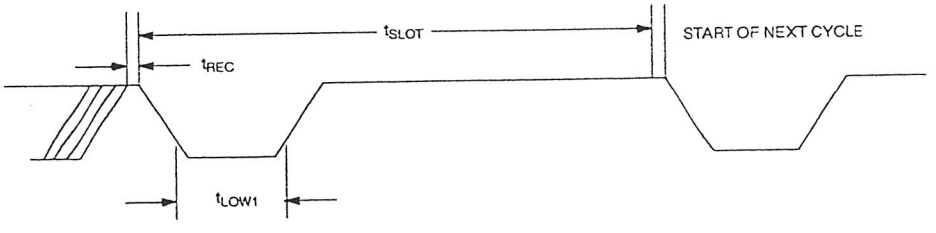
(–55°C to +125°C; V_{DD}=3.6V to 5.5V)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Temperature Conversion Time	t_{CONV}		200	500	ms	
Time Slot	t_{SLOT}	60		120	µs	
Recovery Time	t_{REC}	1			µs	
Write 0 Low Time	t_{LOW0}	60		120	µs	
Write 1 Low Time	t_{LOW1}	1		15	µs	
Read Data Valid	t_{RDV}			15	µs	
Reset Time High	t_{RSTH}	480			µs	
Reset Time Low	t_{RSTL}	480		4800	µs	
Presence Detect High	t_{PDHIGH}	15		60	µs	
Presence Detect Low	t_{PDLOW}	60		240	µs	
Capacitance	$C_{IN/OUT}$			25	pF	

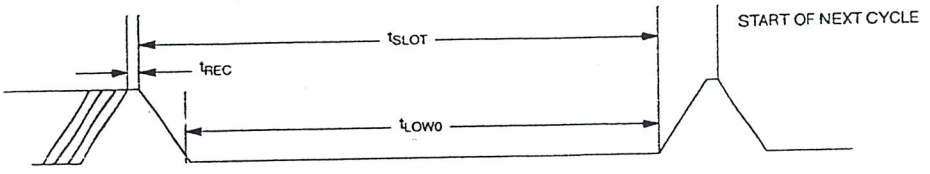
NOTES:

1. Temperature conversion will work with ±2°C accuracy down to V_{DD} = 3.4 volts.
2. All voltages are referenced to ground.
3. Logic one voltages are specified at a source current of 1 mA.
4. Logic zero voltages are specified at a sink current of 4 mA.
5. I_{DD} specified with V_{CC} at 5.0 volts.
6. Active current refers to either temperature conversion or writing to the E^2 memory. Writing to E^2 memory consumes approximately 200 µA for up to 10 ms.
7. Input load is to ground.
8. Standby current specified up to 70°C. Standby current typically is 5 µA at 125°C.
9. See Typical Curve for specification limits outside the 0°C to 70°C range. Thermometer error reflects sensor accuracy as tested during calibration.
10. Typical accuracy curve valid for 4.3V ≤ V_{DD} ≤ 5.5V.

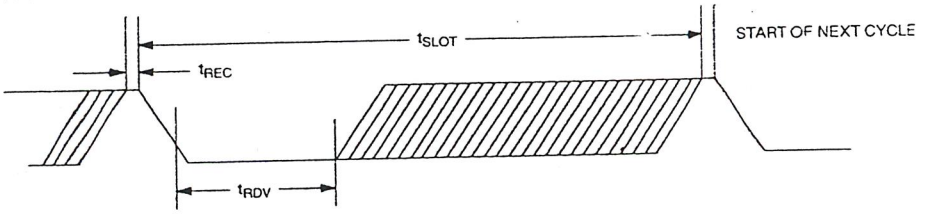
1–WIRE WRITE ONE TIME SLOT



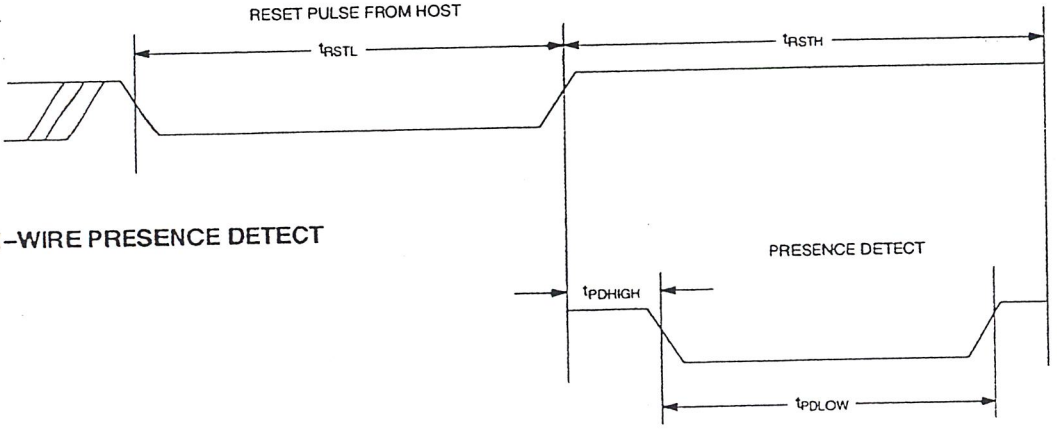
1-WIRE WRITE ZERO TIME SLOT



1-WIRE READ ZERO TIME SLOT



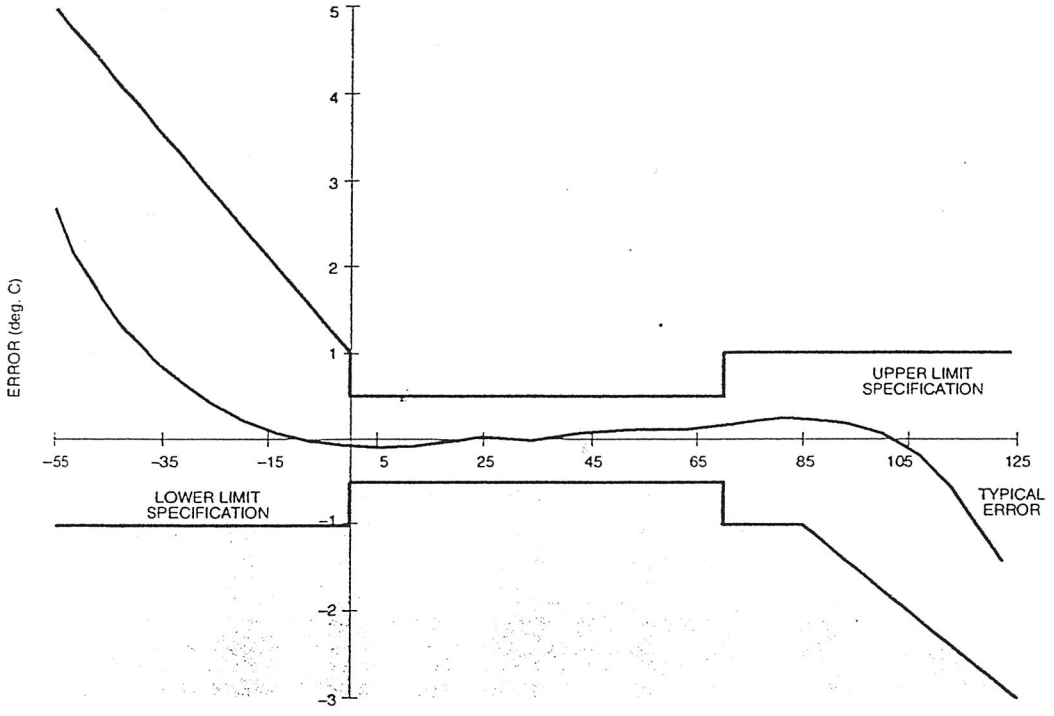
1-WIRE RESET PULSE



1-WIRE PRESENCE DETECT

TYPICAL PERFORMANCE CURVE

DS1820 DIGITAL TERMOMETER AND THERMOSTAT
TEMPERATURE READING ERROR



DALLAS SEMICONDUCTOR

www.dalsemi.com

DS275
Line-Powered RS-232 Transceiver Chip

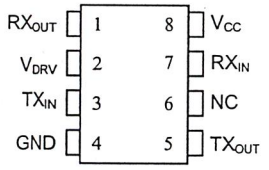
FEATURES

- Low-power serial transmitter/receiver for battery-backed systems
- Transmitter steals power from receive signal line to save power
- Ultra-low static current, even when connected to RS-232-E port
- Variable transmitter level from +5 to +12 volts
- Compatible with RS-232-E signals
- Available in 8-pin, 150 mil wide SOIC package (DS275S)
- Low-power CMOS

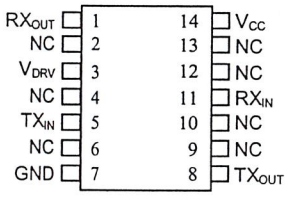
ORDERING INFORMATION

DS275	8-pin DIP
DS275S	8-pin SOIC
DS275E	14-pin TSSOP

PIN ASSIGNMENT



DS275 8-Pin DIP (300-mil)
DS275 8-Pin SOIC (150-mil)



DS275E 14-Pin TSSOP

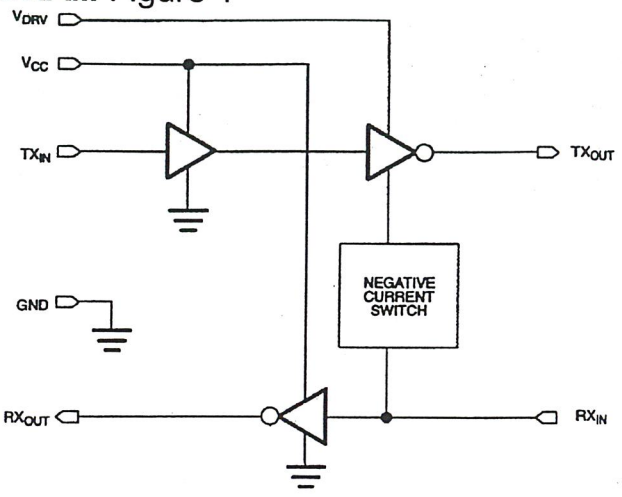
PIN DESCRIPTION

RX_{OUT}	- RS-232 Receiver Output
V_{DRV}	- Transmit driver +V
TX_{IN}	- RS-232 Driver Input
GND	- System Ground (0V)
TX_{OUT}	- RS-232 Driver Output
NC	- No Connection
RX_{IN}	- RS-232 Receive Input
V_{CC}	-System Logic Supply (+5V)

DESCRIPTION

The DS275 Line-Powered RS-232 Transceiver Chip is a CMOS device that provides a low-cost, very low-power interface to RS-232 serial ports. The receiver input translates RS-232 signal levels to common CMOS/TTL levels. The transmitter employs a unique circuit which steals current from the receive RS-232 signal when that signal is in a negative state (marking). Since most serial communication ports remain in a negative state statically, using the receive signal for negative power greatly reduces the DS275's static power consumption. This feature is especially important for battery-powered systems such as laptop computers, remote sensors, and portable medical instruments. During an actual communication session, the DS275's transmitter will use system power (5-12 volts) for positive transitions while still employing the receive signal for negative transitions.

DS275 BLOCK DIAGRAM Figure 1



OPERATION

Designed for the unique requirements of battery-backed systems, the DS275 provides a low-power half-duplex interface to an RS-232 serial port. Typically, a designer must use an RS-232 device which uses system power during both negative and positive transitions of the transmit signal to the RS-232 port. If the connector to the RS-232 port is left connected for an appreciable time after the communication session has ended, power will statically flow into that port, draining the battery capacity. The DS275 eliminates this static current drain by stealing current from the receive line (RXIN) of the RS-232 port when that line is at a negative level (marking). Since most asynchronous communication over an RS-232 connection typically remains in a marking state when data is not being sent, the DS275 will not consume system power in this condition. System power would only be used when positive-going transitions are needed on the transmit RS-232 output (TXOUT) when data is sent. However, since synchronous communication sessions typically exhibit a very low duty-cycle, overall system power consumption remains low.

RECEIVER SECTION

The RXIN pin is the receive input for an RS-232 signal whose levels can range from ±3 to ±15 volts. A negative data signal is called a mark while a positive data signal is called a space. These signals are inverted and then level-shifted to normal +5-volt CMOS/TTL logic levels. The logic output associated with RXIN is RXOUT which swings from +VCC to ground. Therefore, a mark on RXIN produces a logic 1 at RXOUT; a space produces a logic 0.

The input threshold of RXIN is typically around 1.8 volts with 500 millivolts of hysteresis to improve noise rejection. Therefore, an input positive-going signal must exceed 1.8 volts to cause RXOUT to switch states. A negative-going signal must now be lower than 1.3 volts (typically) to cause RXOUT to switch again. An open on RXIN is interpreted as a mark, producing a logic 1 at RXOUT.

TRANSMITTER SECTION

TXIN is the CMOS/TTL-compatible input for digital data from the user system. A logic 1 at TXIN produces a mark (negative data signal) at TXOUT while a logic 0 produces a space (positive data signal). As mentioned earlier, the transmitter section employs a unique driver design that uses the RXIN line for swinging to negative levels. The RXIN line must be in a marking or idle state to take advantage of this design; if RXIN is in a spacing state, TXOUT will only swing to ground. When TXOUT needs to transition to a positive level, it uses the VDRV power pin for this level. VDRV can be a voltage supply between 5 to 12

volts, and in many situations it can be tied directly to the +5 volt V_{CC} supply. *It is important to note that V_{DRV} must be greater than or equal to V_{CC} at all times.*

The voltage range on V_{DRV} permits the use of a 9-volt battery in order to provide a higher voltage level when TX_{OUT} is in a space state. When V_{CC} is shut off to the DS275 and V_{DRV} is still powered (as might happen in a battery-backed condition), only a small leakage current (about 50-100 nA) will be drawn. If TX_{OUT} is loaded during such a condition, V_{DRV} will draw current only if RX_{IN} is not in a negative state. During normal operation (V_{CC}=5 volts), V_{DRV} will draw less than 2 uA when TX_{OUT} is marking. Of course, when TX_{OUT} is spacing, V_{DRV} will draw substantially more current—about 3 mA, depending upon its voltage and the impedance that TX_{OUT} sees.

The TX_{OUT} output is slew rate-limited to less than 30 volts/us in accordance with RS-232 specifications. In the event TX_{OUT} should be inadvertently shorted to ground, internal current-limiting circuitry prevents damage, even if continuously shorted.

RS-232 COMPATIBILITY

The intent of the DS275 is not so much to meet all the requirements of the RS-232 specification as to offer a low-power solution that will work with most RS-232 ports with a connector length of less than 10 feet. As a prime example, the DS275 will not meet the RS-232 requirement that the signal levels be at least ±5 volts minimum when terminated by a 3 kΩ load and V_{DRV} = +5 volts. Typically a voltage of 4 volts will be present at TX_{OUT} when spacing. However, since most RS-232 receivers will correctly interpret any voltage over 2 volts as a space, there will be no problem transmitting data.

APPLICATIONS INFORMATION

The DS275 is designed as a low-cost, RS-232-E interface expressly tailored for the unique requirements of battery-operated handheld products. As shown in the electrical specifications, the DS275 draws exceptionally low operating and static current. During normal operation when data from the handheld system is sent from the TX_{OUT} output, the DS275 only draws significant V_{DRV} current when TX_{OUT} transitions positively (spacing). This current flows primarily into the RS-232 receiver's 3-7 kΩ load at the other end of the attaching cable. When TX_{OUT} is marking (a negative data signal), the V_{DRV} current falls dramatically since the negative voltage is provided by the transmit signal from the other end of the cable. This represents a large reduction in overall operating current, since typical RS-232 interface chips use charge-pump circuits to establish both positive and negative levels at the transmit driver output.

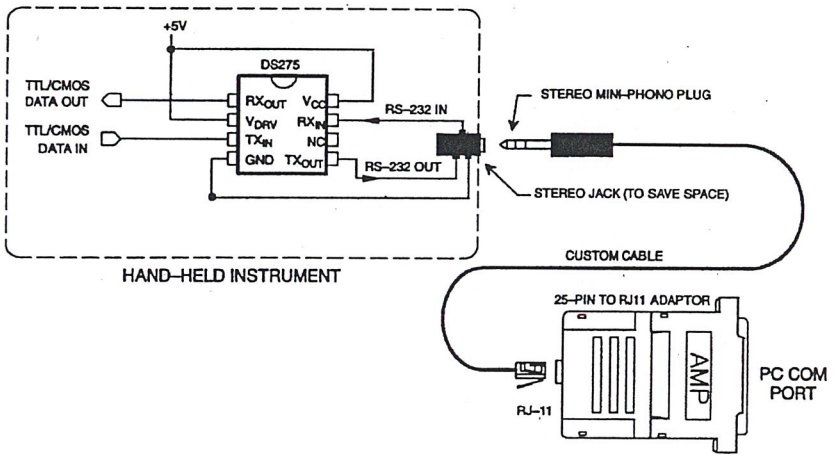
To obtain the lowest power consumption from the DS275, observe the following guidelines. First, to minimize V_{DRV} current when connected to an RS-232 port, always maintain TX_{IN} at a logic 1 when data is not being transmitted (idle state). This will force TX_{OUT} into the marking state, minimizing V_{DRV} current. Second, V_{DRV} current will drop to less than 100 nA when V_{CC} is grounded. Therefore, if V_{DRV} is tied directly to the system battery, the logic +5 volts can be turned off to achieve the lowest possible power state.

FULL-DUPLEX OPERATION

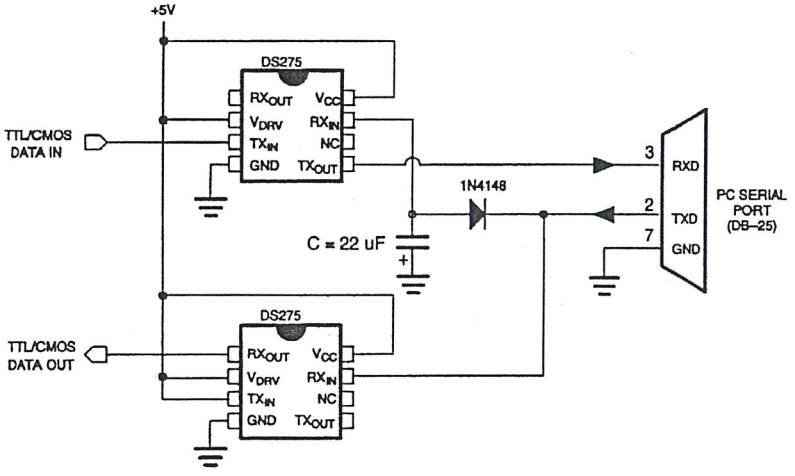
The DS275 is intended primarily for half-duplex operation; that is, RX_{IN} should remain idle in the marking state when transmitting data out TX_{OUT} and visa versa. However, the part can be operated full-duplex with most RS-232-E serial ports since signals swinging between 0 and +5V will usually be correctly interpreted by an RS-232-E receiver device. The 5-volt swing occurs when TX_{OUT} attempts to swing negative while RX_{IN} is at a positive voltage, which turns on an internal weak pulldown to ground for the TX_{OUT} driver's negative reference. So, transmit mark signals at TX_{OUT} may have voltage jumps from some negative value (corresponding to RX_{IN} marking) to approximately ground. One possible

problem that may occur in this case is if the receiver at the other end requires a negative voltage for recognizing a mark. In this situation, the full-duplex circuit shown in Figure 3 can be used as analternative. The 22 µF capacitor forms a negative-charge reservoir; consequently, when the TXD line is spacing (positive), TXOUT still has a negative source available for a time period determined by the capacitor and the load resistance at the other end (3-7 kΩ). This circuit was tested from 150-19,200 bps with error-free operation using a SN75154 Quad Line Receiver as the receiver for the TXOUT signal. Note that the SN75154 can have a marking input threshold below ground; hence there is the need for TXOUT to swing both positive and negative in full-duplex operation with this device.

HANDHELD RS-232-C APPLICATION USING A STEREO MINI-JACK Figure 2



FULL-DUPLEX CIRCUIT USING NEGATIVE-CHARGE STORAGE Figure 3



NOTE:

The capacitor stores negative charge whenever the TXD signal from the PC serial port is in a marking data state (a negative voltage that is typically -10 volts). The top DS275's TXOUT uses this negative charge reservoir when it is in a marking state. The capacitor will discharge to 0 volts when the TXD line is spacing (and TXOUT is still marking) at a time constant determined by its value and the value of the load resistance reflected back to TXOUT. However, when TXD is marking the capacitor will quickly charge back to -10 volts. Note that TXD remains in a marking state when idle, which improves the performance of this circuit.

ABSOLUTE MAXIMUM RATINGS*

VCC	-0.3 to +7.0 volts
VDRV	-0.3 to +13.0 volts

RX_{IN}	±15 volts
TX_{IN}	-0.3 to Vcc + 0.3 volts
TX_{OUT}	±15 volts
RX_{OUT}	-0.3 to Vcc + 0.3 volts
Storage Temperature	-55°C to +125°C
Operating Temperature	0°C to 70°C

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

RECOMMENDED DC OPERATING CONDITIONS

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Logic Supply	V_{CC}	4.5	5.0	5.5	V	1
Transmit Driver Supply	V_{DRV}	4.5	5-12	13.0	V	1
Logic 1 Input	V_{IH}	2.0		V_{CC}+0.3	V	2
Logic 0 Input	V_{IL}	-0.3		+0.8	V	
RS-232 Input Range (RX_{IN})	V_{RS}	-15		+15	V	
Dynamic Supply Current						
TX_{IN} = V_{CC}	I_{DRV1}		400	800	µA	
	I_{CC1}		40	100	µA	3
TX_{IN} = GND	I_{DRV1}		3.8	5.0	µA	
	I_{CC1}		40	100	µA	
Static Supply Current						
TX_{IN} = V_{CC}	I_{DRV2}		1.5	10.0	µA	
	I_{CC2}		10.0	15.0	µA	4
TX_{IN} = GND	I_{DRV2}		3.8	5.0	mA	
	I_{CC2}		10.0	20.0	µA	
Driver Leakage Current (V_{CC}=0V)	I_{DRV3}		0.05	1.0	µA	5

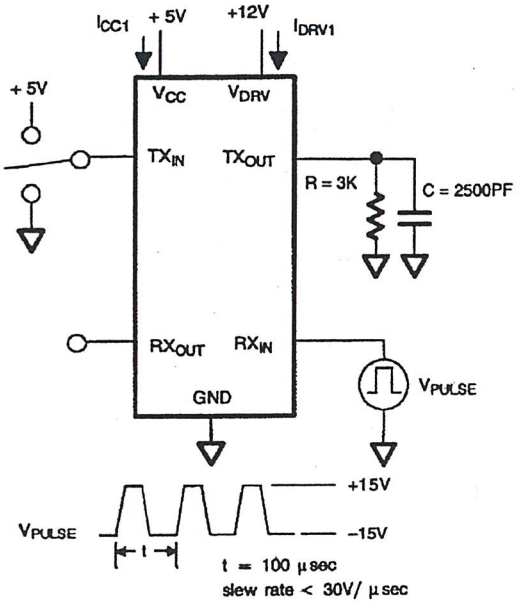
DC ELECTRICAL CHARACTERISTICS (0°C to 70°C; V_{CC} = V_{DRV} = 5V ± 10%)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
TX_{OUT} Level High	V_{OTXH}	3.5	4.0	5.0	V	6
TX_{OUT} Level Low	V_{OTXL}	-8.5	-9.0		V	7
TX_{OUT} Short Circuit Current	I_{SC}		+60	+85	mA	
TX_{OUT} Output Slew Rate	t_{SR}			30	V/µs	
Propagation Delay	t_{PD}		5		µs	8
RX_{IN} Input Threshold Low	V_{TL}	0.8	1.2	1.6	V	
RX_{IN} Input Threshold High	V_{TH}	1.6	2.0	2.4	V	
RX_{IN} Threshold Hysteresis	V_{HYS}	0.5	0.8		V	9
RX_{OUT} Output Current @ 2.4V	I_{OH}	-1.0			mA	
RX_{OUT} Output Current @ 0.4V	I_{OL}			3.2	mA	

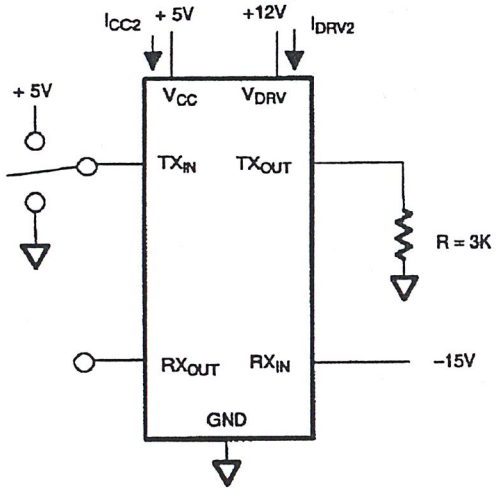
NOTES:

1. V_{DRV} must be greater than or equal to V_{CC}.
2. V_{CC} = V_{DRV} = 5V ± 10%.
3. See test circuit in Figure 4.
4. See test circuit in Figure 5.
5. See test circuit in Figure 6.
6. TX_{IN} = V_{IL} and TX_{OUT} loaded by 3 kΩ to ground.
7. TX_{IN} = V_{IH}, RX_{IN} = -10 volts and TX_{OUT} loaded by 3 kΩ to ground.
8. TX_{IN} to TX_{OUT} - see Figure 7.
9. V_{HYS} = V_{TH} - V_{TL}.

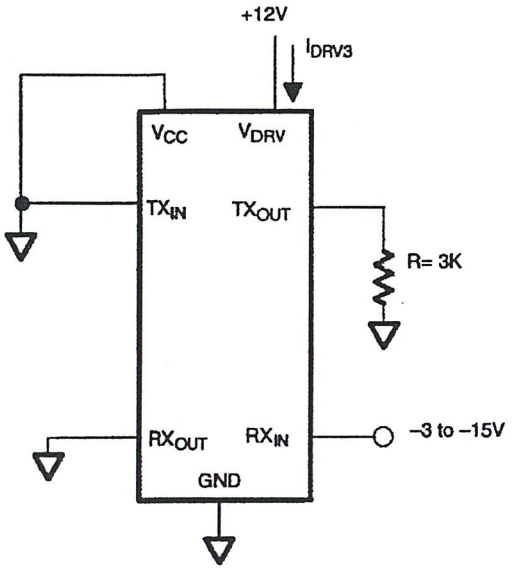
DYNAMIC OPERATING CURRENT TEST CIRCUIT Figure 4



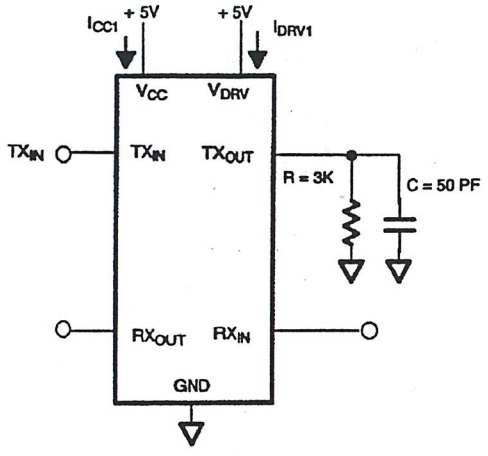
STATIC OPERATING CURRENT TEST CIRCUIT Figure 5

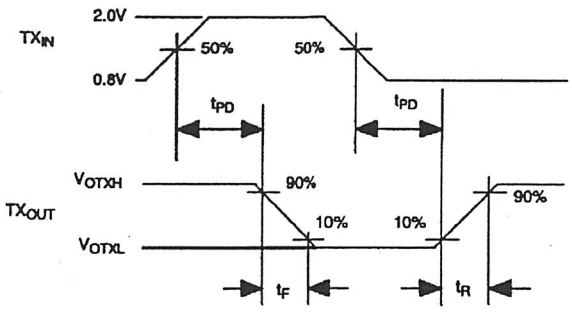


DRIVER LEAKAGE TEST CIRCUIT Figure 6



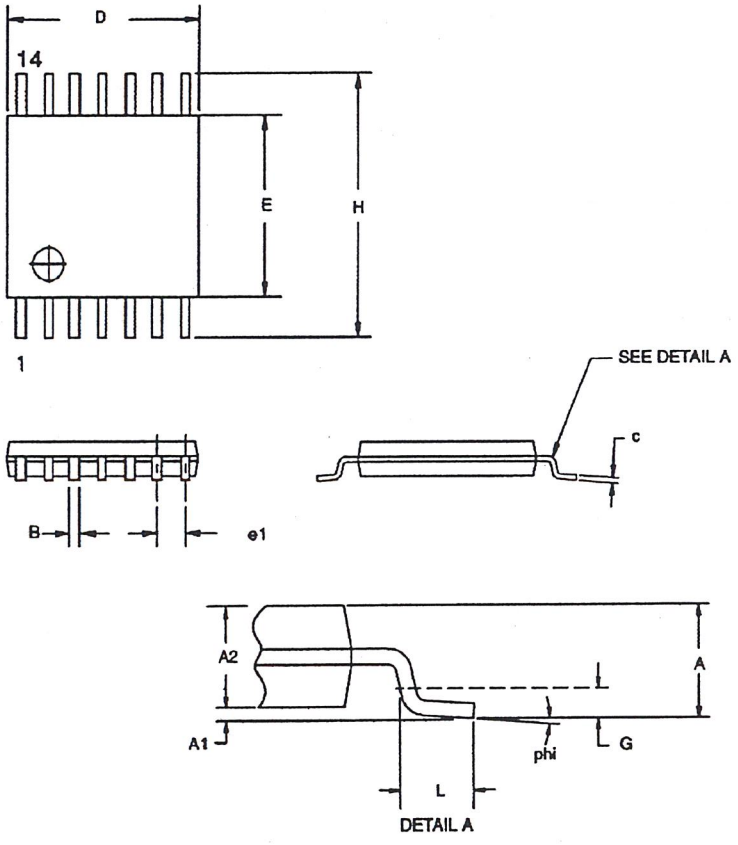
PROPAGATION DELAY TEST CIRCUIT Figure 7





$$t_{SR} = \frac{0.8(V_{OTXH} - V_{OTXL})}{t_F \text{ or } t_R}$$

DS275E 14-PIN TSSOP



DIM	14-PIN	
	MIN	MAX
A MM	-	1.10
A1 MM	0.05	-
A2 MM	0.75	1.05
B MM	0.18	0.30
C MM	0.09	0.18
D MM	4.90	5.10
E MM	4.40 NOM	
e1 MM	0.65 BSC	
G MM	0.25 REF	
H MM	6.25	6.55
L MM	0.50	0.70
phi	0°	8°

DALLAS SEMICONDUCTOR

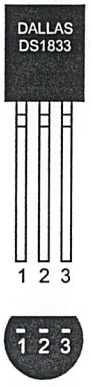
www.dalsemi.com

DS1833
5V EconoReset

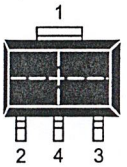
FEATURES

- Automatically restarts microprocessor after power failure
- Maintains active-high reset for 350 ms after V_{CC} returns to an in-tolerance condition
- Accurate 5%, 10% or 15% microprocessor 5V power supply monitoring
- Reduces need for discrete components
- Precision temperature-compensated voltage reference and voltage sensor
- Low-cost TO-92 package or surface mount SOT-223 package
- Internal 5k pull-up resistor
- Operating temperature of -40°C to +85°C

PIN ASSIGNMENT



TO-92 PACKAGE
See Mech.
Drawings Section



SOT-223 PACKAGE
See Mech.
Drawings Section

PIN DESCRIPTION

PIN 1	Ground
PIN 2	Reset
PIN 3	V_{CC}
PIN 4	Ground (SOT-223 only)

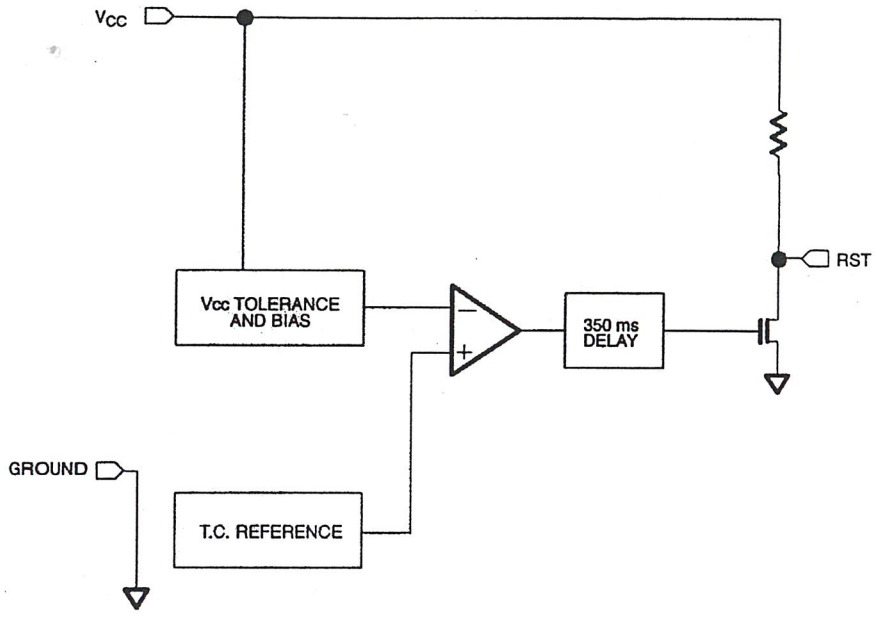
DESCRIPTION

The DS1833 EconoReset uses a precision temperature compensated reference and comparator circuit to monitor the status of the power supply (V_{CC}). When an out-of-tolerance condition is detected, an internal power fail signal is generated which forces reset to the active (high) state. When V_{CC} returns to an in-tolerance condition, the reset signal is kept in the active state for approximately 350 ms to allow the power supply and processor to stabilize.

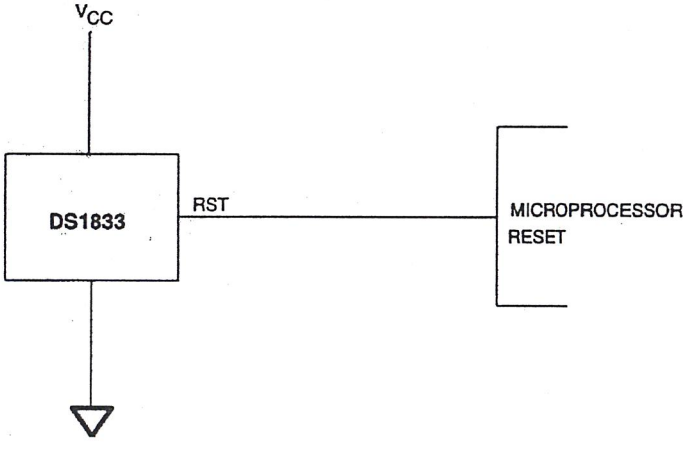
OPERATION - POWER MONITOR

The DS1833 provides the functions of detecting out-of-tolerance power supply conditions and warning a processor-based system of impending power failure. When V_{CC} is detected as out-of-tolerance, as defined by the tolerance of the part selected, the RST signal is asserted. On power-up, RST is kept active for approximately 350 ms after the power supply has reached the selected tolerance. This allows the power supply and microprocessor to stabilize before RST is released.

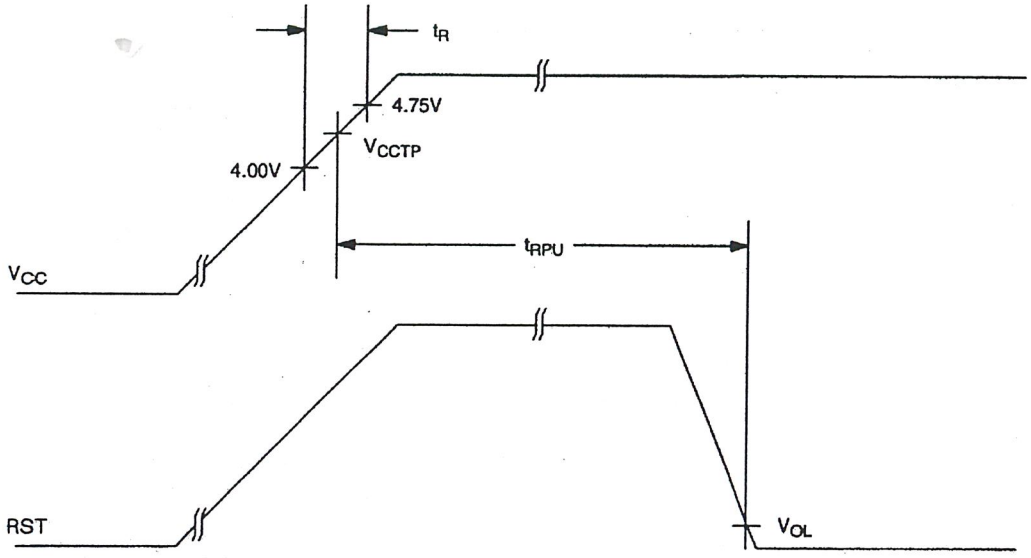
BLOCK DIAGRAM Figure 1



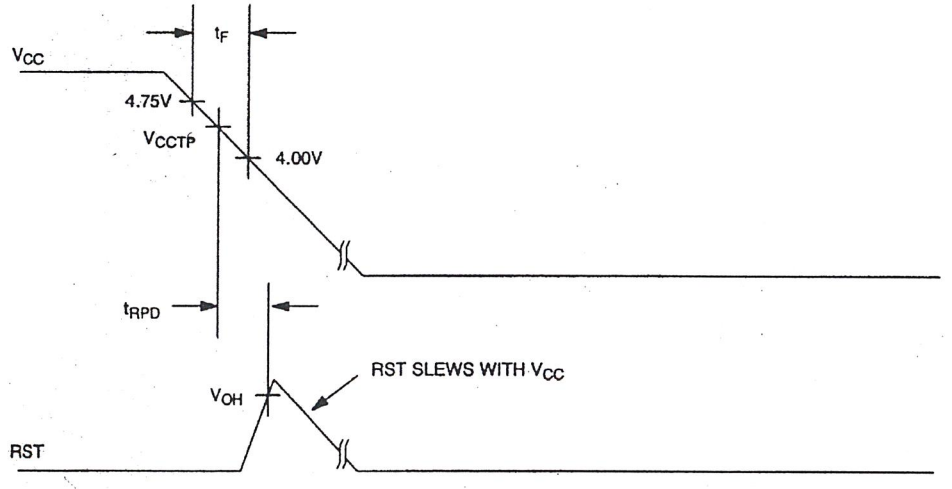
APPLICATION EXAMPLE Figure 2



POWER-UP Figure 3



POWER-DOWN Figure 4



ABSOLUTE MAXIMUM RATINGS*

Voltage on V_{CC} Pin Relative to Ground	-0.5V to +7.0V
Voltage on I/O Relative to Ground	-0.5V to V_{CC} +0.5V
Operating Temperature	-40°C to +85°C
Storage Temperature	-55°C to +125°C
Soldering Temperature	260°C for 10 seconds

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

RECOMMENDED DC OPERATING CONDITIONS (-40°C to +85°C)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{CC}	1.2	5.0	5.5	V	1

DC ELECTRICAL CHARACTERISTICS (-40°C to +85°C; V_{DD}=5V ± 10%)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Low Level @ RST	V_{OL}			0.4	V	1
Output Current @ 0.4V	I_{OL}	+8			mA	
Operating Current	I_{CC}		1.5	2	mA	
V_{CC} Trip Point 5%	V_{CCTP1}	4.5	4.625	4.74	V	1
V_{CC} Trip Point 10%	V_{CCTP2}	4.25	4.375	4.49	V	1
V_{CC} Trip Point 15%	V_{CCTP3}	4.0	4.125	4.24	V	1
Output Capacitance	C_{OUT}			10	pF	
Internal Pull-Up Resistor	R_P	3.75	5	6.25	kΩ	

AC ELECTRICAL CHARACTERISTICS (-40°C to +85°C; V_{CC}=5V ± 10%)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Reset Active Time	t_{RST}	250	350	450	ms	
V_{CC} Detect to RST	t_{RPD}			100	ns	
V_{CC} Slew Rate (4.75V - 4.00V)	t_F	300			µs	
V_{CC} Slew Rate (4.00V - 4.75V)	t_R	0			ns	
V_{CC} Detect to RST	t_{RPU}	250	350	450	ms	

NOTES:

1. All voltages are referenced to ground.

ECONORESET SELECTION GUIDE

		VCC TRIP POINT			PUSHBUTTON DETECT		
		MIN	TYP	MAX	MIN	TYP	MAX
5V	DS1233-15	4.0	4.125	4.24	2.4	-	3.3
	DS1233-10	4.25	4.375	4.49	2.4	-	3.3
	DS1233-5	4.5	4.625	4.75	2.4	-	3.3
	DS1233D-15	4.0	4.125	4.24	N/A		N/A
	DS1233D-10	4.25	4.375	4.49	N/A		N/A
	DS1233D-5	4.5	4.625	4.75	N/A		N/A
	DS1833-15	4.0	4.125	4.24	N/A		N/A
	DS1833-10	4.25	4.375	4.49	N/A		N/A
	DS1833-5	4.5	4.625	4.75	N/A		N/A
3.3V	DS1233A-15	2.64	2.72	2.80	1.8	-	3.0
	DS1233A-10	2.8	2.88	2.97	1.8	-	3.0