เครื่องเตือนภัยระบบควบคุมอุณหภูมิ

DETECTOR CONTROL TEMPERATURE SYSTEM

โดย

นายแก้วสุวรรณ แซ่เลี้ยง

นายวรวุฒิ โชติภูมิเวทย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

เครื่องเตือนภัยระบบควบคุมอุณหภูมิ

DETECTOR CONTROL TEMPERATURE SYSTEM

โดย

นายแก้วสุวรรณ แซ่เลี้ยง 43015763

นายวรวุฒิ โชติภูมิเวทย์ 43015783

อาจารย์ที่ปรึกษา

อ. มนชนก ศรีเสือขาม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

เครื่องเตือนภัยระบบควบคุมอุณหภูมิ

DETECTOR CONTROL TEMPERATURE SYSTEM

โดย
นายแก้วสุวรรณ แซ่เลี้ยง
นายวรวุฒิ โชติภูมิเวทย์

อาจารย์ที่ปรึกษา
อาจารย์ มนชนก ศรีเสือขาม

ปีการศึกษา 2545

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการศึกษา การสร้างเครื่องเตือนภัยระบบควบคุมอุณหภูมิ ซึ่งประกอบด้วย ไอซีวัดอุณหภูมิ DS1820 ซึ่งใช้การติดต่อออกแบบระบบบัสหนึ่งสายโดยจะส่งข้อมูลอุณหภูมิไปยังไมโครคอนโทรลเลอร์ ซึ่งไมโครคอนโทรลเลอร์จะทำการนำข้อมูลอุณหภูมิที่วัดได้ไปแสดงผลบนจอ LCD และไมโครคอนโทรลเลอร์จะทำการเปรียบเทียบค่าอุณหภูมิกับค่าที่ตั้งเตือนไว้แล้วจะทำการเตือนภัยเมื่ออุณหภูมิเท่ากับค่าที่เราตั้งค่าไว้ ซึ่งเครื่องจะทำการเตือนโดยเสียง

Abstract

This project refers to compose of IC measure temperature DS 1820 which use contraction design a bus system by send temperature data to micro controller which will lead measured temperature data to show output on LCD screen and microcontroller will compare temperature value with warn setted value then will warn when temperature equal setting value which this machine will warn by sound

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดีก็เพราะได้รับความเมตตาจาก อาจารย์ มนชนก ศรีเสือขาม อาจารย์ที่ปรึกษาโครงงานนี้ที่คอยให้ความช่วยเหลือ คำแนะนำ ชี้แนะ พร้อมกับให้ข้อมูลต่างๆ ซึ่งเป็นประโยชน์ต่อการทำโครงงานนี้เป็นอย่างยิ่ง

ขอขอบพระคุณคณาอาจารย์ทุกท่านที่ได้ให้คำชี้แนะ คำปรึกษาและประสิทธิ์ประสาทวิชาความรู้ให้กับผู้จัดทำ

ขอขอบพระคุณภาควิชาเทคนิคอุตสาหกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้เอื้อเฟื้อเรื่องเครื่องมือและอุปกรณ์ต่างๆ ในการทำโครงงานนี้ให้ได้ประสบผลสำเร็จลุล่วงไปด้วยดี

ขอขอบคุณเพื่อนๆพี่ๆ ทุกคนที่คอยให้คำปรึกษาและคำแนะนำต่างๆ ตลอดจนอุปกรณ์ในการทำงานมาโดยตลอด

ผู้จัดทำ
นายแก้วสุวรรณ แซ่เลี้ยง
นายวรวุฒิ โชติภูมิเวทย์

สารบัญ

หน้า

บทคัดย่อ

กิตติกรรมประกาศ

บทที่ 1 บทนำ 1

บทที่ 2 ทฤษฎี 3

2.1 ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช 3

2.1.1 คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์ตระกูล MCS-51 อนุกรม AT89xx 3

2.1.2 การจัดขาของ MCS-51 5

2.1.3 โครงสร้างและการทำงานของพอร์ต 8

2.1.4 หน่วยความจำโปรแกรม (Program memory) 13

2.1.5 หน่วยความจำของมูล (Data memory) 15

2.1.6 การเข้าถึงข้อมูล (Addressing) 24

2.1.7 ส่วนควบคุมพื้นฐานของไมโครคอนโทรลเลอร์ 25

2.2 ระบบสื่อสารข้อมูลอนุกรมแบบหนึ่งสาย (1 wire™ Serial Bus) 28

2.2.1 คุณสมบัติทางเทคนิคของระบบบัสหนึ่งสาย 29

2.2.2 คุณสมบัติของไทม์สล็อต 30

2.2.3 รูปแบบของการสื่อสารข้อมูลแบบหนึ่งสาย (1 wire™ comunication protocal) 33

2.3 ไอซีตรวจจับอุณหภูมิ DS1820 34

2.4 โมดูลแสดงผลแบบผลึกเหลว (LCD Module) 36

2.4.1 โครงสร้างภายในของตัวควบคุมโมดูล LCD 36

2.4.2 โมดูล LCD ขนาด 16 ตัวอักษร 1 บรรทัด (LCD 16 x 1) 37

2.4.3 คำสั่งควบคุม LCD 39

บทที่ 3 การออกแบบและการทดลอง 43

3.1 ฮาร์ดแวร์ (Hardware) 43

3.2 ซอฟท์แวร์ (Software) 48

สารบัญ (ต่อ)

บทที่ 4 การทดลองและผลทดลอง 49

4.1 การเชื่อมต่อไมโครคอนโทรลเลอร์กับคีย์แพด 49

4.2 การเชื่อมต่อโมดูล LCD 57

4.3 การทดลองการทำงานของเครื่องเตือนภัยระบบควบคุมอุณหภูมิ 67

บทที่ 5 สรุปผลการทดลอง 69

ภาคผนวก

หนังสืออ้างอิง

สารบัญรูปภาพ

หน้า

รูปที่ 1-1 บล็อคไดอะแกรมเครื่องเตือนระบบควบคุมอุณหภูมิ 2

รูปที่ 2-1 โครงสร้างพื้นฐานของ MCS-51 แบบแฟลชในอนุกรม AT89Cxx 4

รูปที่ 2-2 โครงสร้างพื้นฐานของ MCS-51 แบบแฟลชในอนุกรม AT89Sxx 5

รูปที่ 2-3 รายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช 5

รูปที่ 2-4 การจัดขามาตราฐานของไมโครคอนโทรลเลอร์ 6

รูปที่ 2-5 วงจรภายในของพอร์ตทุกพอร์ตใน MCS-51 แบบแฟลช 10

รูปที่ 2-6 วงจรพูลอัปภายในพอร์ตของ MCS-51 แบบแฟลช 11

รูปที่ 2-7 การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช 14

รูปที่ 2-8 การเชื่อมต่อหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์ MCS-51 15

รูปที่ 2-9 การเชื่อมต่อหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช 16

รูปที่ 2-10 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช 16

รูปที่ 2-11 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในส่วนล่างไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช 17

รูปที่ 2-12 โครงสร้างของหน่วยของมูลภายในส่วนบนของไมโครคอนโทรลเลอร์ MCS-51 18

รูปที่ 2-13 การจัดสรรพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR) 19

รูปที่ 2-14 การต่อออสซิลเลอร์แบบต่างๆ 26

รูปที่ 2-15 วงจรรีเซต 26

รูปที่ 2-16 วงจรรีเซตที่สมบูรณ์ 27

รูปที่ 2-17 วงจรสมบูรณ์ที่ออกแบบโดยใช้ MCS-51 ตัวเดียว 28

รูปที่ 2-18 การเชื่อมต่อบนระนาบบัสหนึ่งสาย 29

รูปที่ 2-19 ไทม์สล๊อตการรีเซตและการตอบรับของอุปกรณ์บนระบบบัสหนึ่งสาย 31

รูปที่ 2-20 ไทม์สล๊อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์ซึ่งตรงกับไทม์สล๊อตการเขียนข้อมูลของอุปกรณ์สเลฟ 32

สารบัญรูปภาพ (ต่อ)

หน้า

รูปที่ 2-21 ไทม์สล๊อตการเขียนข้อมูล "1" ของอุปกรณ์มาสเตอร์ซึ่งตรงกับไทม์สล๊อตการอ่านข้อมูลของอุปกรณ์สเลฟ 33

รูปที่ 2-22 การจัดขาของDS1820 34

รูปที่ 2-23 โครงสร้างการทำงานภายในของไอซีตรวจจับอุณหภูมิ DS1820 34

รูปที่ 2-24 การจัดสรรพื้นที่ของสแครตช์แพดใน DS1820 35

รูปที่ 2-25 ไดอะแกรมการทำงานของโมดูล LCD แบบอักขระ 36

รูปที่ 2-26 รูปร่างและการจัดขาโมดูล LCD แบบอักขระ 38

รูปที่ 3-1 วงจรสวิตช์แบบเมตริกซ์หรือคีย์แพด 43

รูปที่ 3-2 ลายวงจรคีย์แพด 44

รูปที่ 3-3 วงจรเครื่องเตือนระบบควบคุมอุณหภูมิ 45

รูปที่ 3-4 ลายวงจรเครื่องเตือนระบบควบคุมอุณหภูมิ 46

รูปที่ 3-5 วงจรแหล่งจ่ายไฟ 47

รูปที่ 3-6 ลายวงจรแหล่งจ่ายไฟ 47

รูปที่ 4-1 โฟลวชาร์ตโปรแกรมหลักการอ่านค่าคีย์แพดเพื่อนำไปแสดงผลที่ LED 7 ส่วน 49

รูปที่ 4-2 โฟลวชาร์ตโปรแกรมย่อยในการอ่านค่าคีย์แพด 50

รูปที่ 4-3 การอ่านค่าคีย์แพดของไมโครคอนโทรลเลอร์ MCS-51 51

รูปที่ 4-4 โฟลวชาร์ตแสดงข้อมูลบน LCD ขนาด 16 ตัวอักษร 1 บรรทัด 57

รูปที่ 4-5 วงจรการแสดงข้อความบนโมดูล LCD 16 ตัวอักษร 1 บรรทัด 58

สารบัญตาราง

หน้า

ตารางที่ 2-1 หน้าที่พิเศษของพอร์ต 1 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช 9

ตารางที่ 2-2 การเลือกแบงก์ของหน่วยความจำส่วนล่างเพื่อติดต่อกับรีจิสเตอร์ แบงก์ R0-R7 21

ตารางที่ 2-3 แสดงความสัมพันธ์ในการทำงานของขา RS, R/W และ E ของโมดูล LCD แบบอักขระ 39

ตารางที่ 2-4 แสดงการกำหนดบิต S/C และ R/L 41

บทที่ 1
บทนำ

แนวความคิดและที่มา

เนื่องจากในสภาวะปัจจุบันมีอันตรายส่วนใหญ่เกิดจากความร้อนเป็นจุดเริ่ม และความเสียหายที่เกิดขึ้น แต่ละครั้งก็มีมูลค่ามาก จึงได้มีการคิดค้นหาวิธีป้องกันขึ้น ซึ่งก็มีมากมายหลายวิธี ซึ่งการตรวจจับอุณหภูมิก็เป็นอีกหนึ่งวิธี ที่สามารถลดอันตรายที่จะเกิดขึ้นได้ ถ้าเรามีเครื่องที่สามารถเตือนภัยได้ก่อนเกิดอันตรายก็จะสามารถป้องกันอุบัติเหตุที่เกิดขึ้นได้ เช่น สารเคมีบางประเภทถ้ามีอุณหภูมิสูงกว่า 70°C จะเกิดปฏิกิริยาขึ้น ดังนั้นถ้าเราสามารถกำหนดอุณหภูมิการตรวจจับที่อุณหภูมิประมาณ 60°C ก็จะสามารถป้องกันอันตรายที่เกิดขึ้นได้

วัตถุประสงค์

1. นำความรู้ทางอิเล็กทรอนิกส์ที่ได้ศึกษามาใช้และประยุกต์ใช้งานจริงได้
2. สร้างอุปกรณ์เพื่อใช้ป้องกันภัยที่สามารถควบคุมได้ในชีวิตประจำวัน

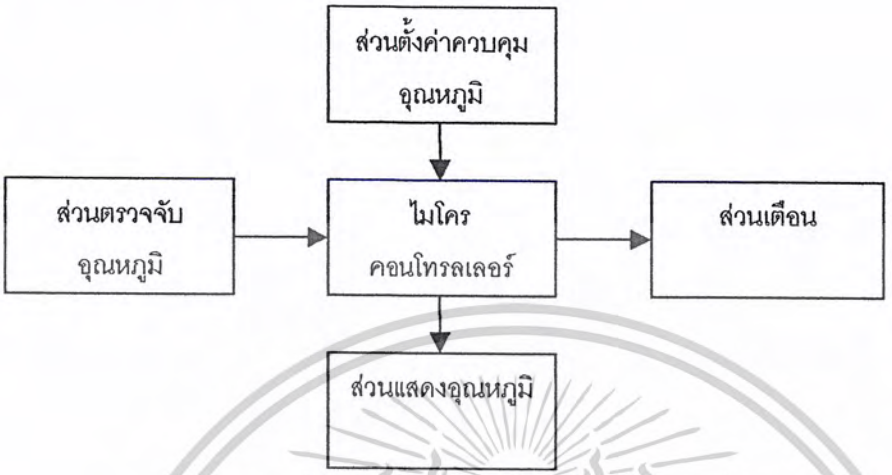
ขั้นตอนการดำเนินงาน

เริ่มต้นได้ทำการศึกษาค้นคว้าอุปกรณ์ที่เกี่ยวข้องกับการวัดอุณหภูมิและอุปกรณ์ที่เกี่ยวกับไมโครคอนโทรลเลอร์ การแสดงผลโดยใช้จอ LCD การเชื่อมต่อกับคีย์แพด

ศึกษาและออกแบบวงจรภาคต่างๆ ซึ่งประกอบไปด้วยภาควัดอุณหภูมิ ภาคกำหนดค่าอุณหภูมิที่ควบคุม และภาควงจรเตือน และทำการทดลองวงจรที่ได้ออกแบบ

เมื่อทดลองวงจรแล้วได้วงจรตามที่ต้องการ จึงนำภาคต่างๆมาทำงานร่วมกัน โดยใช้ไมโครคอนโทรลเลอร์ควบคุมแล้วทำการทดสอบการทำงาน แล้วทำการพัฒนาหรือแก้ไขเพื่อให้ได้ตามวัตถุประสงค์

รูปแบบของเครื่องเตือนภัยระบบควบคุมอุณหภูมิ



รูปที่ 1-1 บล็อกไดอะแกรมเครื่องเตือนภัยระบบควบคุมอุณหภูมิ

เครื่องเตือนภัยระบบควบคุมอุณหภูมิใช้ไมโครคอนโทรลเลอร์เป็นตัวประมวลผลซึ่งจะใช้การออกแบบระบบด้วยซอฟท์แวร์ (Software) โดยใช้ไอซีวัดอุณหภูมิ DS1820 เป็นตัววัดอุณหภูมิแล้วส่งค่าอุณหภูมิให้ไมโครคอนโทรลเลอร์ แล้วทำการแสดงค่าอุณหภูมิบนจอ LCD ในส่วนของการตั้งค่าควบคุมอุณหภูมิจะใช้คีย์แพดตั้งค่า โดยไมโครคอนโทรลเลอร์จะรับค่าจากคีย์แพดแล้วนำค่ามาเปรียบเทียบกันเพื่อควบคุมการเตือน

ประโยชน์ของโครงงาน

1. สามารถป้องกันอุบัติเหตุที่เกิดจากความร้อนตามอุณหภูมิแวดล้อมได้
2. ใช้แสดงและตรวจวัดอุณหภูมิได้
3. สามารถที่จะกำหนดค่าอุณหภูมิที่ต้องการทำการเตือนได้

คุณสมบัติของเครื่องเตือนภัยระบบควบคุมอุณหภูมิ

1. สามารถวัดค่าอุณหภูมิในหน่วยองศาเซลเซียลได้ ตั้งแต่ 0 - 99
2. สามารถกำหนดค่าอุณหภูมิที่ให้ทำการเตือนในหน่วยองศาเซลเซียลได้ตั้งแต่ 0 - 99

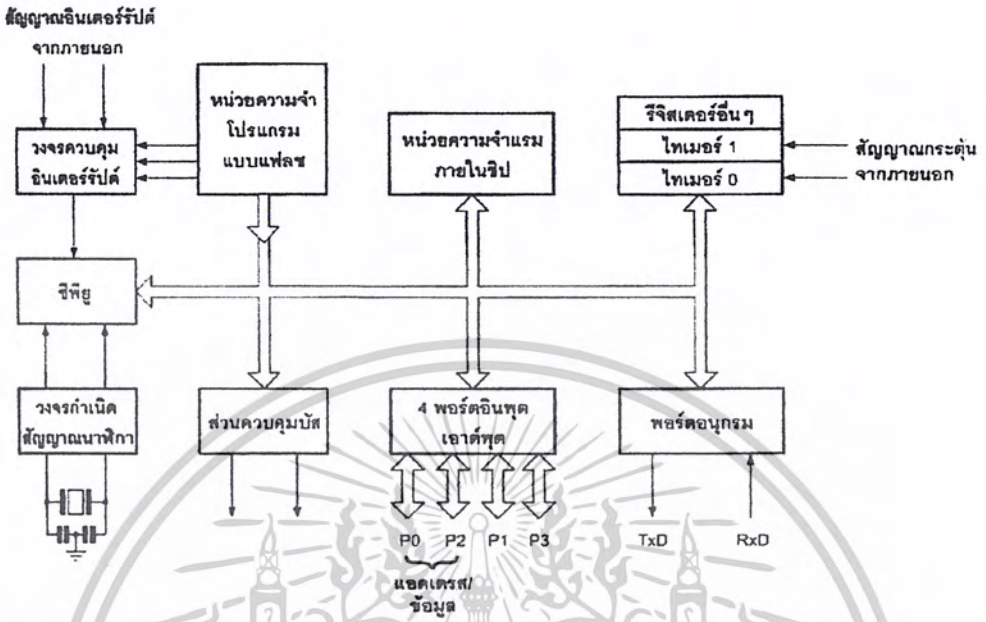
บทที่ 2
ทฤษฎี

2.1 ไมโครคอนโทรลเลอร์ MCS – 51 แบบแฟลช

2.1.1 คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์ตระกูล MCS-51 อนุกรมAT89xx

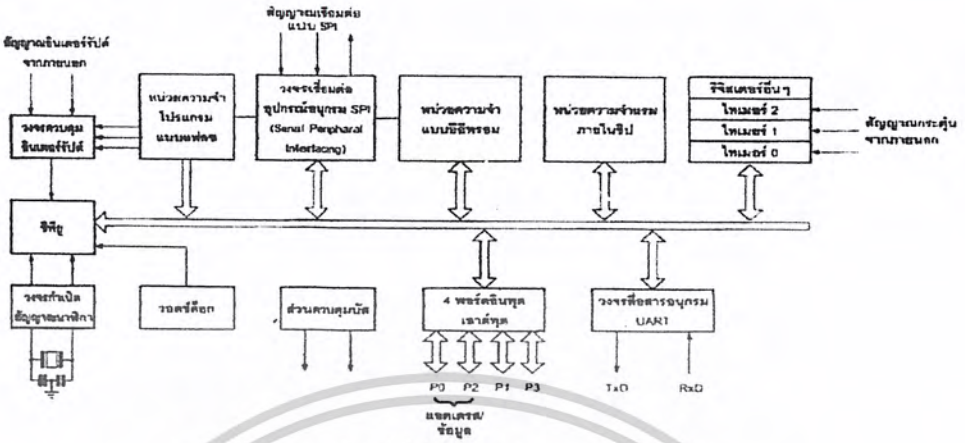
- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้พันครั้ง
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม ในบางเบอร์จะมีหน่วยความจำแบบอีอีพรอมเพิ่มเติม
- ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาต์พุต
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- ไทเมอร์ / เคาร์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัปต์ได้ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ในภายชิป
- มีวงจรสื่อสารอนุกรมแบบ SPI สำหรับในอนุกรม AT89Sxx
- มีวอตช์ด็อกไทเมอร์ในตัว สำหรับในอนุกรม AT89Sxx

ในรูปที่ 2-1 เป็นโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx จะเห็นได้ว่า โครงสร้างของ AT89Cxx จะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐานหากแต่แตกต่างกันเฉพาะหน่วยความจำโปรแกรมแบบแฟลชที่เพิ่มเติมเข้ามา หากเป็นไมโครคอนโทรลเลอร์ ในอนุกรม 87xx หน่วยความจำโปรแกรมภายในจะเป็นแบบอีพรอม และบางเบอร์สามารถโปรแกรมได้เพียงครั้งเดียว



รูปที่ 2-1 โครงสร้างพื้นของ MCS – 51 แบบแฟลชในอนุกรม AT89Cxx

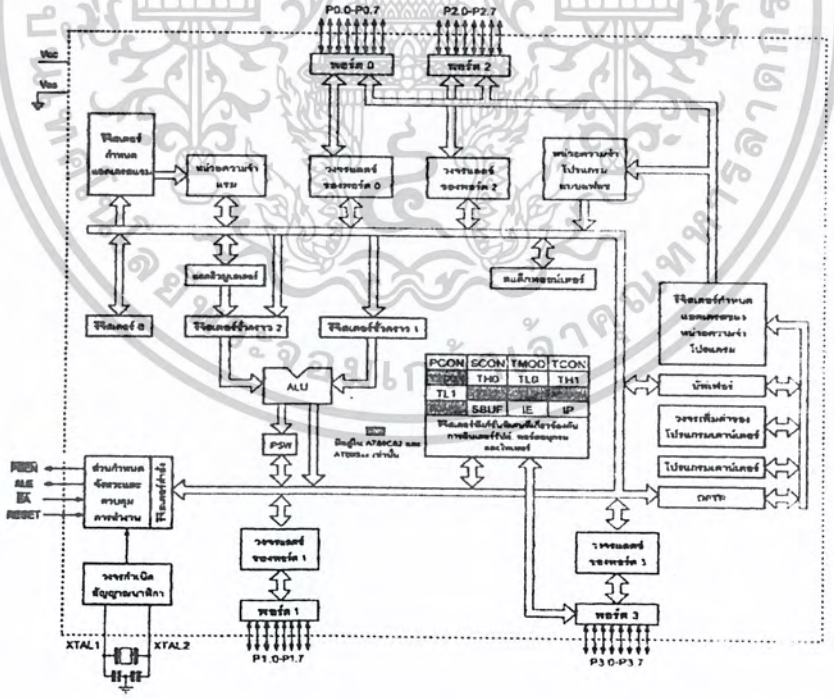
สำหรับในรูปที่ 2-2 เป็นโครงสร้างพื้นฐานของอนุกรม AT89Sxx จะเห็นได้ว่า มีส่วนประกอบที่เพิ่มเติมแตกต่างจาก AT89Cxx อยู่หลายส่วน อาทิ วงจรเชื่อมต่ออนุกรมแบบ SPI ซึ่งในไมโครคอนโทรลเลอร์ อนุกรมนี้ใช้ในการเขียนข้อมูลในหน่วยความจำโปรแกรมโดยไม่ต้องถอดตัวชิป ออกไปจากระบบหรือเรียกว่า การโปรแกรมในวงจร ไทเมอร์ / เคาน์เตอร์ขนาด 16 บิต ที่เพิ่มเติมเข้ามาอีกตัวหนึ่งตัวเป็น ไทเมอร์ 2 และวงจรวอชด็อกที่ใช้ในการตรวจสอบการทำงานผิดพลาดของซีพียู



รูปที่ 2-2 โครงสร้างพื้นของ MCS – 51 แบบแฟลชในอนุกรม AT89Cxx

2.1.2 การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกันดังแสดงในรูปที่ 2-3 และ รูปที่ 2-4



รูปที่ 2-3 รายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลช

รูปที่ 2-4 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์

โดยมีรายละเอียดขั้นต้น ดังนี้

ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5V

ขา GND เป็นขากราวด์ สำหรับต่อกับกราด์ของระบบ

ขาพอร์ต 0 (P0.0 - P0.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้ โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยใช้กระบวนการมัลติแพล็กซ์เข้าช่วย เพื่อสลับการทำงานให้เป็นได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล

ขาพอร์ต 1 (P1.0 - P1.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้ โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย นอกจากนั้นในอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทเมอร์ 2 และ P1.1 เป็นขาอินพุตทริกเกอร์ของไทเมอร์ 2 ในขณะที่ขา P1.4 ถึง P1.7 เป็นขาสำหรับเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ

ขาพอร์ต 2 (P2.0 - P2.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15)

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

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

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

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

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่องที่ 0 หรือขา T0

P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่องที่ 1 หรือขา T1

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

P3.7 ใช้เป็นขาสัญญาณ $\overline{\text{RD}}$ ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

ขารีเซต ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซต สถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชีนไซเกิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

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

ขา $\overline{\text{PSEN}}$ (ProgramStore Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ ต้องการข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้งในแต่ละแมชีนไซเกิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใดๆออกมา

ขา $\overline{\text{EA}}$ / Vpp (External Access enable / Programming voltage input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น "0" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่หากขานี้เป็น "1" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์นอกจากนี้ ที่ขานี้ยังใช้เป็นขาอินพุตสำหรับแรงดันไฟสูงสำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ MCS - 51 แบบแฟลชต้องการแรงดันสำหรับการโปรแกรม + 12 V

ขา XTAL1 และ XTAL2 เป็นขาสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนด จังหวะการทำงานของไมโครคอนโทรลเลอร์

2.1.3 โครงสร้างและการทำงานของพอร์ต

ไมโครคอนโทรลเลอร์ MCS - 51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ตคือ พอร์ต 0 ถึงพอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับสัญญาณข้อมูลเข้าและเอาต์พุตสำหรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของไมโครคอนโทรลเลอร์ MCS – 51 แบบแฟลชมีวงแลตช์และวงจรขับตลอดจนบัฟเฟอร์อินพุต ดังแสดงให้เห็นในสถาปัตยกรรมรูปที่ 2-3

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

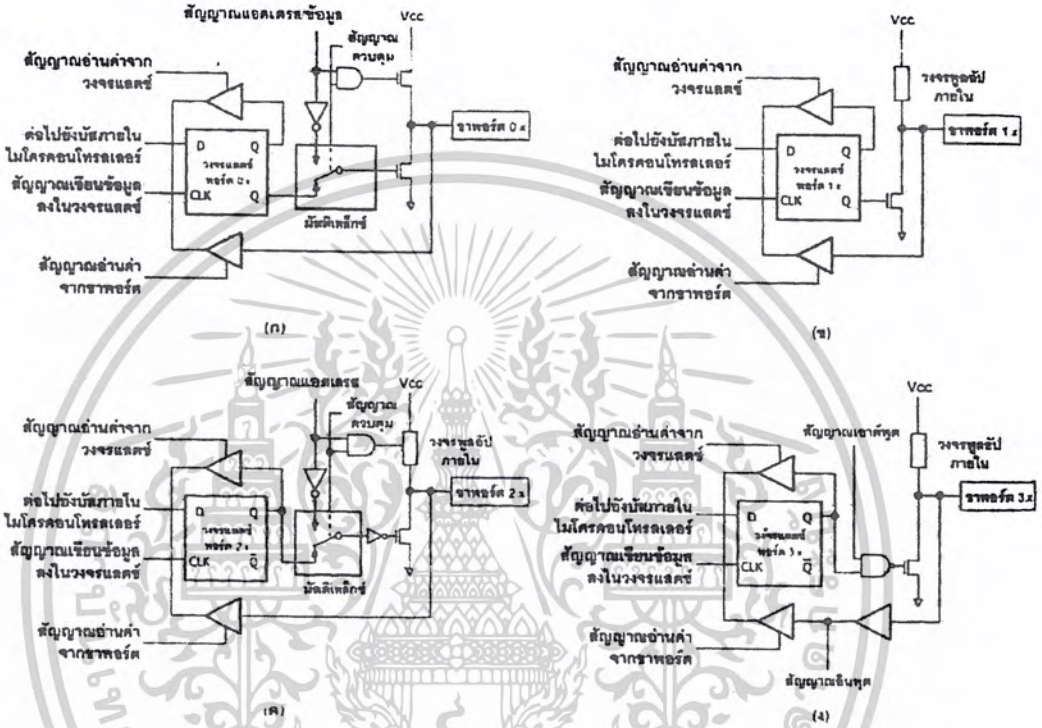
ขา	เบอร์ของ ไมโครคอนโทรลเลอร์	หน้าที่พิเศษ
P1.0	AT89C52 / AT89Sxx	ขา T2 เป็นขาอินพุตนับค่าของไทเมอร์ / เคาน์เตอร์ 2 และเป็นขาเอาต์พุตของการกำเนิดสัญญาณนาฬิกาโดยไทเมอร์ 2 (clock out)
P1.1	AT89C52 / AT89Sxx	ขา T2EX เป็นขาอินพุตทริกเกอร์สำหรับการแคปเจอร์ / รีโหลด และควบคุมทิศทางของสัญญาณ
P1.4	AT89Sxx	ขา $\overline{SS}$ (Slave Select) เป็นขาเลือกการติดต่อในกรณีที่ไมโครคอนโทรลเลอร์เป็นอุปกรณ์สเลฟ ในระบบติดต่อแบบ SPI
P1.5	AT89Sxx	ขา MOSI (Master data input , Slave data input) ใช้ในการติดต่อกับพอร์ต SPI
P1.6	AT98Sxx	ขา MISO (Master data input , Slave data output) ใช้ในการติดต่อกับพอร์ต SPI
P1.7	AT89Sxx	ขา SCK (Master clock output) เป็นขาสัญญาณนาฬิกาของการติดต่อกับพอร์ต SPI

ตารางที่ 2-1 หน้าที่พิเศษของพอร์ต 1 ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในรูปที่ 2-5 แสดงวงจรภายในของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชโดยในรูปที่ 2-5 (ก) เป็นวงจรของพอร์ต 0 วงจรแลตช์ของแต่ละบิต ในแต่ละพอร์ตก็คือ วงจรดีฟลิฟลอปนั่นเอง การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตช์สามารถกระทำได้อย่างอิสระต่อกันด้วยสัญญาณที่แยกจากกัน นั่นคือ สัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูลจากวงจรแลตช์ ส่วนการเขียนข้อมูลมายังพอร์ตต้องส่งสัญญาณมายังขา CLK ของดีฟลิฟลอปในขณะที่ข้อมูลจะส่งผ่านมาทางขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิฟลอป

ที่พอร์ตนี้มีวงจรมัลติเพล็กซ์สำหรับกำหนดลักษณะการทำงานของพอร์ตว่า ต้องการใช้งานเป็นขาพอร์ตอินพุตเอาต์พุตปกติหรือใช้ในการติดต่อกับหน่วยความจำภายนอกไมโครคอนโทรลเลอร์

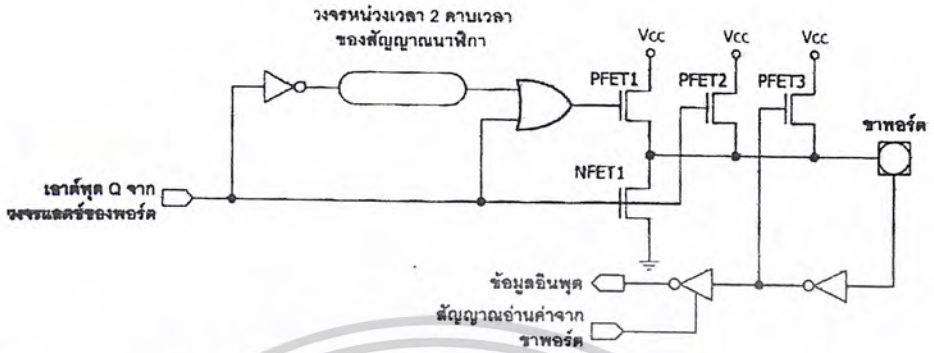
เนื่องจากที่ขาพอร์ต 0 ไม่มีวงจรพูลอัปภายใน หากมีการนำพอร์ต 0 ไปใช้งานเป็นพอร์ตอินพุตจะต้องต่อตัวต้านทานพูลอัปภายนอกเข้าที่ขาพอร์ต 0 ทุกขาด้วย



รูปที่ 2-5 วงจรภายในของพอร์ตทุกพอร์ตใน MCS – 51 แบบแฟลช

ในรูปที่ 2-5 (ข) เป็นวงจรของพอร์ต 1 ซึ่งมีลักษณะโดยทั่วไปคล้ายกับพอร์ต 0 หากแต่ไม่มีวงจรมัลติเพล็กซ์เนื่องจากพอร์ตนี้จะไม่ใช้ในการติดต่อกับหน่วยความจำภายนอก แต่จะมีวงจรพูลอัปภายในที่แต่ละบิตของพอร์ตนี้แทน สำหรับรายละเอียดของวงจรพูลอัปแสดงในรูปที่ 2-6

ในรูปที่ 2-5 (ค) เป็นวงจรภายในของพอร์ต 2 จะคล้ายกับพอร์ต 0 มาก ต่างกันเพียงมีวงจรพูลอัปเพิ่มเติมเข้ามา ส่วนในรูปที่ 2-5(ง) เป็นวงจรภายในของพอร์ต 3 จะเห็นได้ว่าคล้ายกับพอร์ต 1 มีการเพิ่มเติมวงจรบัฟเฟอร์ และวงจรอินพุตเอาต์พุตเมื่อทำงานในฟังชั่นพิเศษเข้ามา เนื่องจากพอร์ต 3 สามารถนำไปใช้งานในหน้าที่พิเศษได้ทุกขา



วงจรพูลอัปประกอบด้วยเฟตชนิดพีแชนเนล 3 ตัวคือ PFET1-PFET3 โดย NFET1 จะทำงานเมื่อได้รับลอจิก "1" จากขา Q และหยุดทำงานเมื่อได้รับลอจิก "0" วงจรพูลอัปจะเริ่มต้นทำงานเมื่อ NFET1 ได้รับลอจิก "1" PFET1 จะทำงานนานประมาณ 2 คาบเวลาของสัญญาณนาฬิกาภายใน หลังจากที่เกิดการเปลี่ยนแปลงจากลอจิก "0" เป็นลอจิก "1" ในขณะที่ PFET1 ทำงาน จะทำให้ PFET3 ทำงานตามไปด้วย ทำให้เกิดการพูลอัปขาพอร์ต

รูปที่ 2-6 วงจรพูลอัปภายในพอร์ตของ MCS – 51 แบบแฟลช

การใช้งานเป็นพอร์ตอินพุต

เนื่องจากพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ MCS – 51 แบบแฟลชสามารถเป็นได้ทั้งอินพุตและเอาท์พุต ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ตของไมโครคอนโทรลเลอร์ MCS – 51 แบบแฟลช

ในการกำหนดให้เป็นพอร์ตอินพุต ต้องเริ่มต้นด้วยการเขียนข้อมูล "1" มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานเป็นอินพุต เพื่อหยุดการทำงานของเฟตที่ใช้ในการขับสัญญาณเอาท์พุตของบิตนั้นๆ ทำให้ขาสัญญาณของพอร์ตเชื่อมต่อเข้ากับวงจรพูลอัปภายในโดยตรง ส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น "1" สามารถรับสัญญาณรับสัญญาณลอจิก "0" จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านค่าเข้าไป เมื่อเป็นเช่นนี้อุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ตอินพุตของไมโครคอนโทรลเลอร์MCS – 51 แบบแฟลชควรกำหนดให้ทำงานในสภาวะลอจิก "0" จะดีและสะดวกที่สุด (ซึ่งในปัจจุบันอุปกรณ์อินพุตที่เชื่อมต่อไมโครคอรโทรลเลอร์แทบทั้งหมดทำงานที่ลอจิก "0" แล้ว)

การใช้งานเป็นพอร์ตเอาต์พุต

โดยปกติแล้ว ขาพอร์ตจะกำหนดให้มีลักษณะเป็นเอาท์พุตอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูลออกไปได้อย่างง่ายดายและตรงไปตรงมากล่าวคือ เมื่อต้องการส่งข้อมูล "0" ออกไปทางเอาต์พุตก็ให้เขียนข้อมูล "0" ไปยังวงจรแลตช์ ซึ่งก็จะส่งต่อไปขับเฟต ทำให้เฟตทำงาน ที่ขาพอร์ตที่กำหนดให้ทำงานก็จะเกิดลอจิก "0" ขึ้น ในทางตรงข้ามหากต้องการส่งข้อมูล "1" ออกไป ก็ให้เขียนข้อมูล "1" ไปยังวงจรแลตช์ วงจรขับก็จะหยุดทำงาน ทำให้ที่ขาพอร์ตเชื่อมต่อกับวงจรพูลอัปภายในเกิดเป็นลอจิก "1" ที่ขาพอร์ตนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุตมาก เพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุตจะมีสัญญาณมาอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาต์พุตจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่อย่างใด เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาต์พุต

เมื่อใช้งานพอร์ตของไมโครคอนโทรลเลอร์ MCS – 51 แบบแฟลชเป็นพอร์ตเอาต์พุต แต่ขา (หรือแต่ละบิต) ของแต่ละพอร์ตมีความสามารถในการจ่ายกระแสหรือที่เรียกว่า กระแสซอร์ส (Source Current) ได้สูงสุด 10 mA และทุกขารวมกันในแต่ละพอร์ต (ทั้ง 8 บิต) สูงสุด 26 mA สำหรับพอร์ต 0 และ 15 mA สำหรับพอร์ต 1-3 ในกรณีที่ใช้งานทุกพอร์ตเอาท์พุตจะสามารถจ่ายกระแสได้รวมกันสูงสุด 71 mA ดังนั้นในการใช้งานเป็นพอร์ตเอาท์พุตเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถในการจ่ายกระแสจึงควรต่อวงจรบัฟเฟอร์ทางเอาท์พุตเพื่อช่วยในการขับกระแสอีกทางหนึ่ง

การอ่านค่าลอจิกจากพอร์ต

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถอ่านค่าลอจิกจากพอร์ตได้ 2 ลักษณะ คือ อ่านจากขาพอร์ตโดยตรง และอ่านจากวงจรแลตช์ของแต่ละพอร์ต

ในกรณีที่พอร์ตต่อกับขาเบสทรานซิสเตอร์ชนิด NPN และขาอิมิเตอร์ของทรานซิสเตอร์ตัวนั้นต่อลงกราวด์ หากมีการส่งข้อมูล "1" ไปยังทรานซิสเตอร์ จะทำให้ทรานซิสเตอร์ทำงาน สถานะลอจิกที่ขาพอร์ตจะเป็น "0" เนื่องจากเมื่อทรานซิสเตอร์ทำงานจะเสมือนว่าพอร์ตนั้นถูกต่อลงกราวด์ ทำให้หากอ่านค่าลอจิกที่ขาพอร์ตจะได้ผลตรงข้ามกับที่ส่งออกมา แต่ถ้าหากทำงานอ่านค่าลอจิกที่วงจรแลตช์ จะได้ค่าที่ตรงกับค่าที่ต้องการส่งจริง ดังนั้น ในการค่าลอจิกพอร์ตจึงต้องเลือกวิธีการให้เหมาะสมกับอุปกรณ์ที่นำมาต่อด้วย

2.1.4 หน่วยความจำโปรแกรม(Program memory)

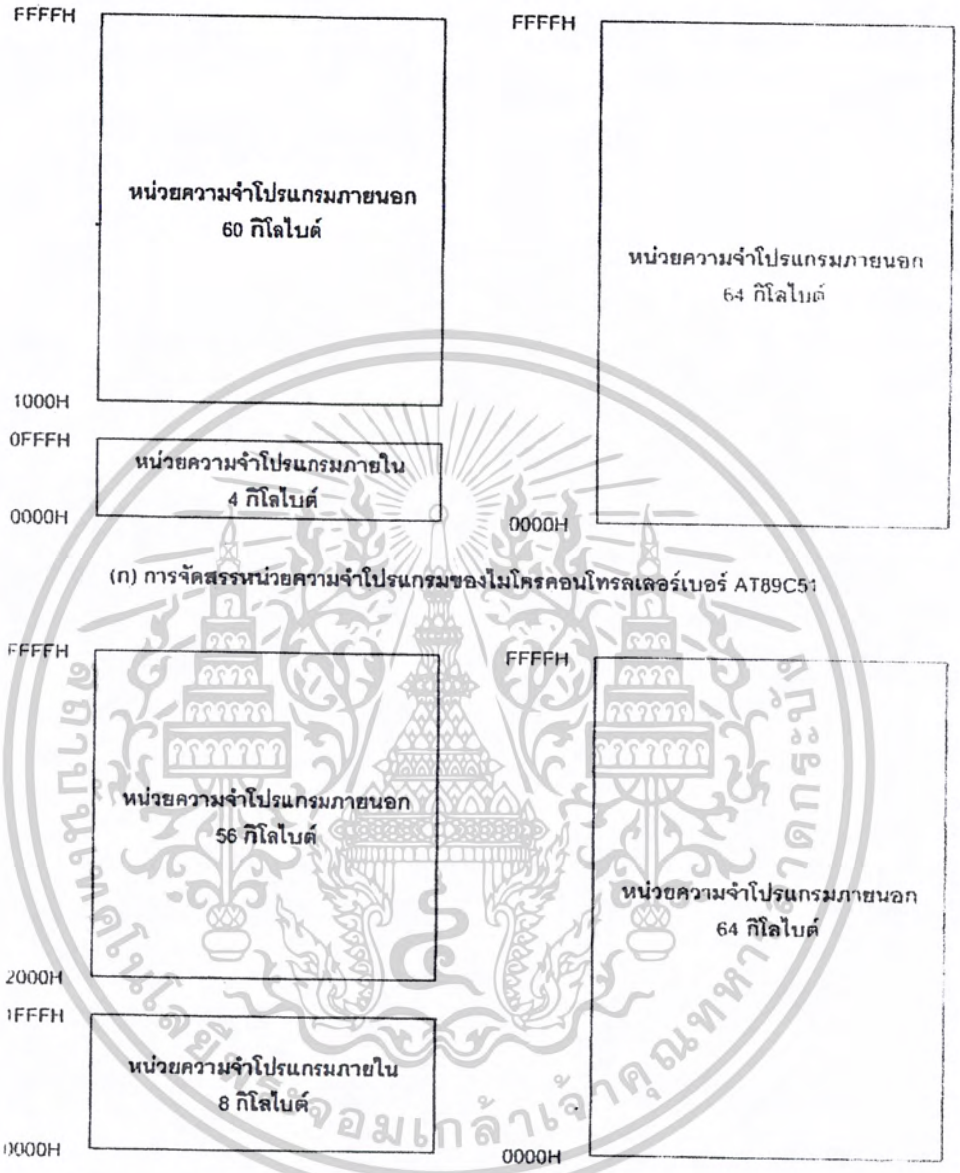
ในรูปที่ 2-7 แสดงการจัดหน่วยความจำโปรแกรมของไมโครคอนโทรเลอร์ MCS-51 แบบแฟลชในเบอร์ต่างๆที่นิยมใช้งานอันประกอบด้วยเบอร์ AT89C51 และ AT89C52จะเห็นได้ว่า ทั้งสองเบอร์สามารถติดต่อหน่วยความจำโปรแกรมได้สูงสุด 64 กิโลไบต์ โดยสามารถเลือกใช้หน่วยความจำโปรแกรมภายในอย่างเดียวหรือรวมกับภายนอกกรือเลือกใช้หน่วยความจำภายนอกอย่างเดียวก็ได้ ดังในรูปที่ 2-7(ก) โดยภายใน AT89C51 จะมีหน่วยความจำในโปรแกรมภายใน 4 กิโลไบต์ ในขณะที่ AT89ฉ52 จะมีขนาด 8 กิโลไบต์

ในกรณีที่ใช้หน่วยความจำภายในและภายนอกรวมกัน หากใช้ AT89C51 ก็จะสามารถติดต่อกับหน่วยความจำภายนอกได้ 60 กิโลไบต์ และถ้าใช้เบอร์ AT89C52 จะสามารถติดต่อหน่วยความจำโปรแกรมภายนอกได้ 56 กิโลไบต์

หน่วยความจำโปรแกรมใช้เก็บข้อมูลโปรแกรมควบคุมการทำงานของไมโครคอนโทรลเลอร์หรือที่เรียกว่า โปรแกรมมอนิเตอร์ (monitor program) หากใช้หน่วยความจำภายนอกมักจะบรรจุอยู่ในหน่วยความจำชนิดอีพรอม (EPROM : Erasable Read-Only Memory) ซึ่งสามารถทำการอ่านได้เพียงอย่างเดียว

หน่วยความจำโปรแกรมมีแอดเดรสเริ่มต้นที่ 0000H เมื่อซีพียูได้รับการรีเซตให้เริ่มต้นการทำงาน จะต้องมาเริ่มที่แอดเดรส 0000H นี้เสมอ อย่างไนก็ตาม ในพื้นที่ของหน่วยความจำโปรแกรมไม่ว่าจะใช้งานจากภายในหรือภายนอกก็ตาม ต้องมีการสงวนพื้ที่บางตำแหน่งเอาไว้สำหรับการอินเตอร์รัปต์ 6 ประเภทประเภทละ 8 ไบต์ ประกอบด้วย

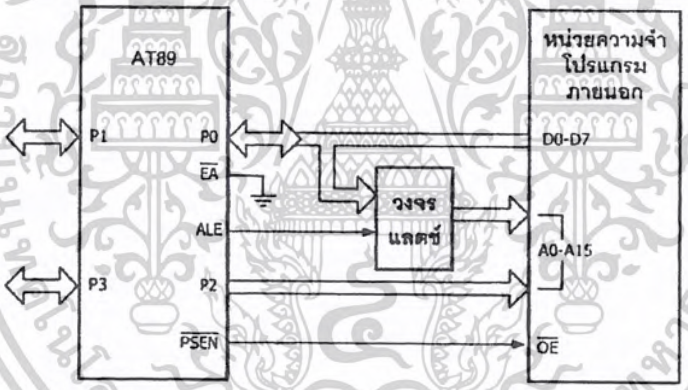
พื้นที่สำหรับบริการอินเตอร์รัปต์ 0 จากภายนอก กำหนดไว้ที่แอดเดรส 0003H
พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 0 กำหนดไว้ที่แอดเดรส 000BH
พื้นที่สำหรับบริการอินเตอร์รัปต์ 1 จากภายนอก กำหนดไว้ที่แอดเดรส 0013H
พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 1 กำหนดไว้ที่แอดเดรส 001BH
พื้นที่สำหรับบริการอินเตอร์รัปต์ของการสื่อสารอนุกรม กำหนดไว้ที่แอดเดรส 0023H
พื้นที่สำหรับบริการอินเตอร์รัปต์จากไทเมอร์ 2 กำหนดไว้ที่แอดเดรส 002BH



รูปที่ 2-7 การจัดสรรหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

กรณีที่ใช้ไมโครคอนโทรเลอร์ MCS-51 แบบแฟลชที่มีหน่วยความจำโปรแกรมภายใน แต่ต้องการติดต่อกับหน่วยความจำโปรแกรมภายนอกด้วย สามารถทำได้โดยต้องกำหนดแอดเดรสของหน่วยความจำโปรแกรมให้ต่อจากแอดเดรสสุดท้ายของหน่วยความจำโปรแกรมภายในของไมโครคอนโทรลเลอร์

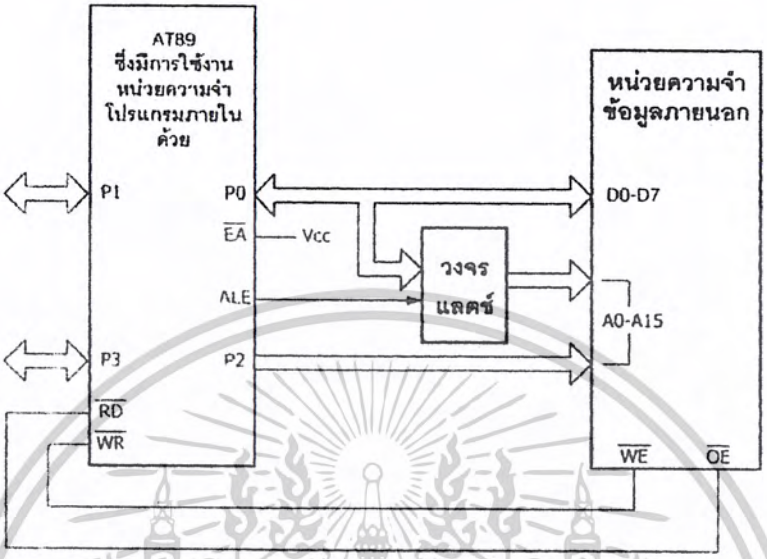
การต่อหน่วยความจำภายนอกแสดงในรูปที่ 2-8 ขาพอร์ต P0.0-P0.7 ใช้เป็นขาข้อมูล D0 -D7 และขาแอดเดรสไบต์ต่ำ โดยผ่านวงจรแลตช์ ซึ่งปกติใช้ไอซีเบอร์ 74HC573 และใช้สัญญาณ ALE และ PSEN ในการเลือกใช้งานขา P0.0- P0.7 เพื่อเป็นขาข้อมูลหรือขาแอดเดรส ในขณะที่ขา P2.0 – P2.7 ใช้ในการเชื่อมต่อกับขาแอดเดรสไบต์สูง A8-A15 ดังนั้นเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอก ไมโครคอนโทรลเลอร์จะเหลือเพียงขาพอร์ตเพียง 16 บิต คือขาพอร์ต P 1.0-P1.7 และ P3.0-P3.7



รูปที่ 2-8 การเชื่อมต่อหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์ MCS-51

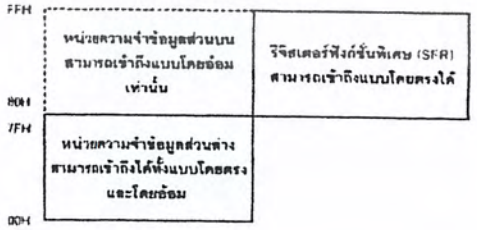
2.1.5หน่วยความจำข้อมูล (Data memory)

มีด้วยกัน 2 แบบคือ หน่วยความจำข้อมูลภายนอกและภายใน โดยไมโครคอนโทรลเลอร์ MCS- 51 แบบแฟลชในอนุกรม AT89 สามารถติดต่อกับหน่วยความจำข้อมูลภายนอกได้สูงสุด 64 กิโลไบต์ โดยการใช้คำสั่ง MOVX ในการติดต่อกับหน่วยความจำข้อมูลภายนอก การติดต่อกับหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช แสดงในรูปที่ 2-9 จะเห็นได้ว่า มีลักษณะคล้ายกับการติดต่อกับหน่วยความจำโปรแกรมภายนอก แตกต่างกันที่มีสัญญาณที่ใช้สำหรับการอ่านและเขียนหน่วยความจำข้อมูลภายนอก นั่นคือ ขา $\overline{RD}$ และ $\overline{WR}$



รูปที่ 2-9 การเชื่อมต่อหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์ MCS-51แบบแฟลช

สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89 ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในเป็นแบบแรม (RAM : Random Access Memory) โดยแต่ละเบอร์จะมีขนาดแตกต่างกันไป ในเบอร์ AT89C51 มีหน่วยความจำข้อมูลภายในขนาด 128 ไบต์ ในขณะที่เบอร์ AT89C52 มีขนาด 256 ไบต์ สำหรับการจัดสรรหน่วยความจำข้อมูลภายในแบ่งเป็น 3 ส่วนคือ หน่วยความจำข้อมูลส่วนล่าง (lower) , ส่วนบน(upper) และ รีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR : Special Funtion Register) แต่ละส่วนมีขนาด 128 ไบต์ ดังแสดงการจัดสรร ในรูปที่ 2-10

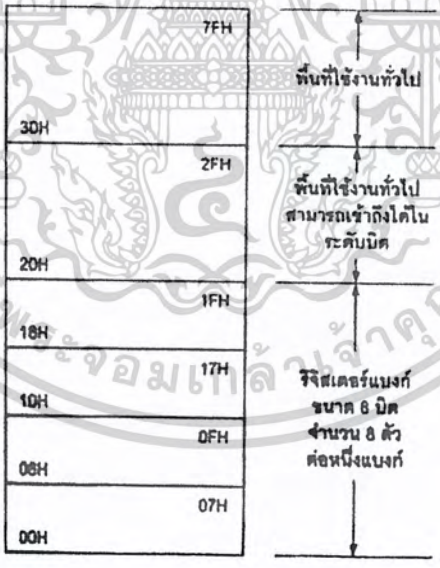


รูปที่ 2-10 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

จะเห็นได้ว่า หน่วยความจำข้อมูลส่วนบนและรีจิสเตอร์ฟังก์ชั่นพิเศษมีตำแหน่งทับซ้อนกัน แต่จะใช้การติดต่อที่แตกต่างกัน และในไมโครคอนโทรลเลอร์ MCS-51 บางเบอร์จะไม่มีหน่วยความจำข้อมูลส่วนบน

ขนาดของหน่วยความจำข้อมูลของไมโครคอนโทรลเลอร์MCS-51แบบแฟลชโดยแท้จริงแล้วมีแพียง 256 ไบต์ แต่ด้วยการจัดการเข้าถึงเข้าถึงที่แตกต่างกัน จึงดูเสมือนว่า ไมโครคอนโทรเลอร์ MCS-51 แบบแฟลชมีหน่วยความจำข้อมูลภายในสูงถึง 384 ไบต์ โดยหน่วยความจำข้อมูลส่วนล่างขนาด 128 ไบต์ มีแอดเดรสอยู่ที่ 00H-7FH สามารถเข้าถึงได้โดยตรงและโดยอ้อม สำหรับหน่วยความจำข้อมูลส่วนบนมีขนาด 128ไบต์เช่นกัน มีแอดเดรสอยู่ที่ 80H-FFH สามารถเข้าถึงแบบโดยอ้อมเท่านั้น ในขณะที่รีจิสเตอร์ SFR มีแอดเดรสอยู่ที่ 80H –FFH

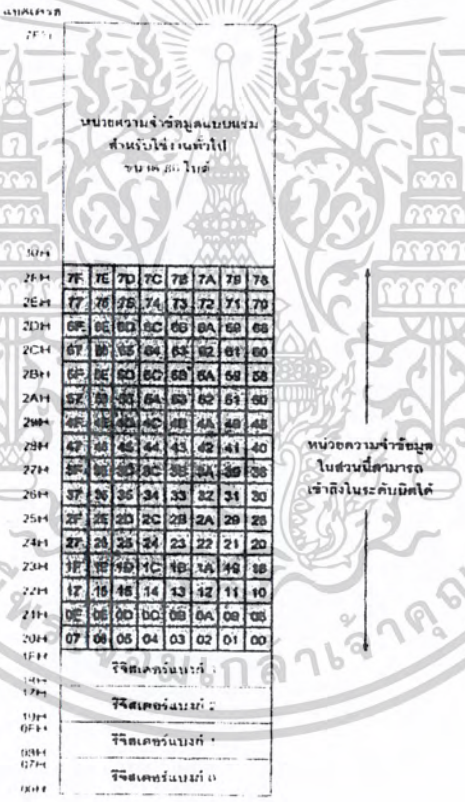
ในรูปที่ 2-11 แสดงการจัดสรรหน่วยความจำข้อมูลส่วนล่าง หน่วยความจำ 32 ไบต์ต่ำสุดที่แอดเดรส 00H-1FH แบ่งเป็น 4 กลุ่ม เรียกว่า 4 แบงก์ (bank) แต่ละแบงก์มีรีจิส 8 ตัวคือ R0-R7 การติดต่อกับหน่วยความจำในแบงก์ใดให้กำหนดที่รีจิสเตอร์ PSW (Program Status Word register)



รูปที่ 2-11 การจัดสรรพื้นที่ของหน่วยความจำข้อมูลภายในส่วนล่างไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

หน่วยความจำข้อมูล 16 ไบต์ถัดมาที่แอดเดรส 20H-2FH เป็นพื้นที่สำหรับใช้งานทั่วไป สามารถเข้าถึงได้ในระดับบิต (bit addressable) และหน่วยความจำข้อมูลที่เหลือ 80 ไบต์ จะต้องแบ่งส่วนหนึ่งสำรองไว้เป็นพื้นที่ของสแต็ก การเข้าถึงหน่วยความจำในส่วนนี้ต้องใช้การเข้าถึงในระดับไบต์

ในรูปที่ 2-12 แสดงโครงสร้างของหน่วยความจำข้อมูลส่วนบน ซึ่งมีลักษณะที่คล้ายกับหน่วยความจำข้อมูลส่วนล่าง หากแต่ใน 80ไบต์บนไม่จำเป็นต้องสำรองไว้สำหรับสแต็ก และต้องใช้การเข้าถึงในลักษณะโดยอ้อมเท่านั้น

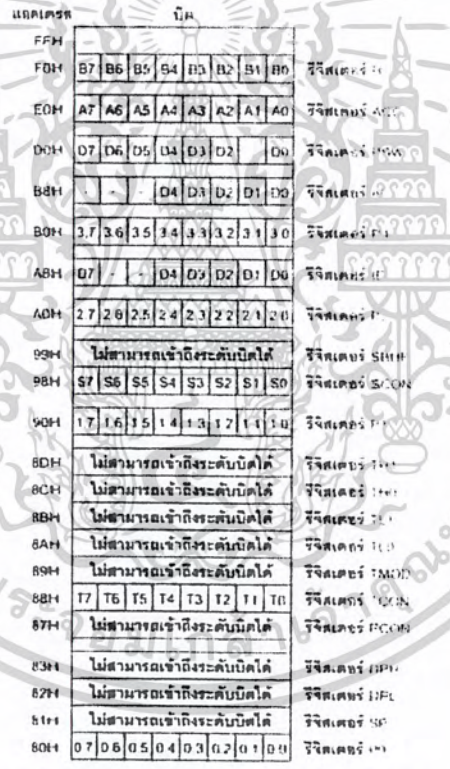


รูปที่ 2-12 โครงสร้างของหน่วยความจำข้อมูลภายในส่วนบนของไมโครคอนโทรลเลอร์ MCS-51

รีจิสเตอร์ฟังก์ชั่นพิเศษ

เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของไมโครคอนโทรเลอร์ MCS-51 มีด้วยกัน 22 ตัว สำหรับเบอร์ AT89C51 และ 28 ตัวในเบอร์ AT89C52 และอนุกรม AT89Sxx ทั้งนี้เนื่องจากใน AT89C52 และ AT89Sxx มีจำนวนไทเมอร์เคาร์เตอร์มากกว่า AT89C51

รีจิสเตอร์ SFR มีแอดเดรสอยู่ระหว่าง 80H-FFH ในพื้นที่ของหน่วยความจำข้อมูลส่วนบน สามารถเข้าถึงได้โดยตรง ในรูปที่ 2-13 แสดงการจัดสรรพื้นที่ของรีจิสเตอร์ SFR แต่ละตัวในหน่วยความข้อมูลส่วนบน สำหรับรายละเอียดเบื้องต้นของรีจิสเตอร์ SFRมีดังนี้



รูปที่ 2-13 การจัดสรรพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษ (SFR)

รีจิสเตอร์แสดงสถานะของโปรแกรม

เป็นรีจิสเตอร์ขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต จึงสามารกำหนดค่าในแต่ละบิตของรีจิสเตอร์ตัวนี้ได้อย่างอิสระ มีแอดเดรสอยู่ที่ D0H ทำหน้าที่เก็บสถานะของการทำงานของโปรแกรมในขณะนั้นจะเรียกสถานะต่างๆ ของโปรแกรมว่าแฟลก เมื่อซีพียูกระทำคำสั่งทางคณิตศาสตร์ และลอจิกแล้วเกิดการเปลี่ยนแปลงสถานะขึ้น ผลของการเปลี่ยนแปลงนั้นจะมาปรากฏที่บิตต่างๆ ของรีจิสเตอร์ PSW แสดงดังนี้

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

CY : แฟลกทด เป็น "1" เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์และลอจิก แล้วค่าของแอกคิวมูลเลเตอร์เกิน 255 หรือ FFH

AC : แฟลกทดเสริม เป็น "1" เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์แล้วทำให้เกิดการทดข้ามจากบิต 3 มายังบิต 4

F0 : แฟลกใช้งานทั่วไป เมื่อผู้เขียนโปรแกรมกำหนดค่าที่บิตนี้แล้ว ไม่ว่าจะกระทำคำสั่งใดๆ ที่บิตนี้จะไม่มีการเปลี่ยนแปลง

RS1 : บิตเลือกรีจิสเตอร์แบงก์ ใช้งานร่วมกับบิต RS0 เพื่อเลือกแบงก์ของ R0-R7

RS0 : บิตเลือกรีจิสเตอร์แบงก์ ใช้งานร่วมกับบิต RS1 เพื่อเลือกแบงก์ของ R0-R7

0V : บิตเกินเป็น "1" เมื่อมีการกระทำคำสั่งทางคณิตศาสตร์และลอจิกแล้ว ทำให้เกิดการทดข้ามจากบิต 6 มายังบิต 7 ของแอกคิวมูเลเตอร์ หรือแอกคิวมูเลเตอร์มีค่าเกิน 127 นอกจากนั้นยังใช้เป็นการแสดงค่าลบด้วย

- : บิตนี้ ผู้ใช้งานสามารถกำหนดใช้งานได้อย่างอิสระ

P : บิตพาริตี้ ใช้ในการตรวจสอบจำนวนค่า "1" ภายในแอกคิวมูเลเตอร์ ถ้าหากในแอกคิวมูเลเตอร์มีจำนวนบิตที่เป็น "1" รวมกันเป็นเลขคู่บิตนี้จะเป็น "0" ถ้ารวมกันเป็นเลขคี่ บิตนี้จะเป็น "1"

จะเห็นได้ว่า นอกจากรีจิสเตอร์ PSW ถูกใช้ในการเก็บสถานะของโปรแกรมแล้ว ที่บิต RS0 และ RS1 ยังใช้ในการเลือกแบงค์ของหน่วยความจำส่วนล่าง ซึ่งเป็นพื้นที่ของรีจิสเตอร์ R0-R7 ด้วย ดังมีรายละเอียดแสดงในตารางที่ 2-2 โดยปกติแล้วในการใช้งานรีจิสเตอร์ R0-R7 มักนิยมใช้แบงก์ 0 เป็นลำดับแรก หากไม่เพียงพอจึงเลือกในแบงค์อื่น มาใช้ แต่ต้องระมัดระวังในการกำหนดค่าและลำดับการติดต่อให้ดี มิเช่นอาจทำให้การเขียนโปรแกรมเกิดความสับสน ดังนั้นสำหรับผู้เริ่มต้นใช้งานไมโครคอนโทรลเลอร์ MCS-51 จึงควรเลือกใช้รีจิสเตอร์ R0-R7 ในแบงค์ 0 เพียงแบงก์เดียวให้ชำนาญเสียก่อน

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

ตารางที่ 2-2 การเลือกแบงก์ของหน่วยความจำส่วนล่างเพื่อติดต่อกับรีจิสเตอร์แบงก์ R0-R7

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

แอกคิวมูเลเตอร์

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ตำแหน่ง E0H เป็นรีจิสเตอร์ที่ใช้สำหรับเก็บข้อมูลหรือผลลัพธ์ที่ได้จากการทำงานของไมโครคอนโทรลเลอร์ โดยเฉพาะอย่างยิ่งในการคำนวนทางคณิตศาสตร์และลอจิก ก่อนที่จะส่งข้อมูลหรือผลลัพธ์ที่ได้ให้แก่ CPU เพื่อทำการประมวลผลต่อไป อาจเรียกสั้นๆว่า รีจิสเตอร์ A หรือ ACC รีจิสเตอร์นี้สามรถเข้าถึงระดับบิตได้

รีจิสเตอร์B

มีขนาด 8 บิต มีแอดเดรสอยู่ที่ F0H มีหน้าที่พิเศษคือ หากต้องการคูณหรือหารทางคณิตศาสตร์ ต้องำข้อมูลที่ต้องการหารหรือคูณมาเก็บไว้ในรีจิสเตอร์B แล้วจึงกระทำคำสั่งการคูณหรือหารกับค่าในรีจิสเตอร์ A ต่อไป

ในกรณีที่ไม่ได้มีความต้องการคูณหรือหารข้อมูล สามารถใช้รีจิสเตอร์B นี้ในการเก็บข้อมูลทั่วไปได้เหมือนกับรีจิสเตอร์ปกติ และสามารถเข้าถึงในระดับบิตได้เช่นเดียวกับรีจิสเตอร์ A

โปรแกรมเคาน์เตอร์

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

รีจิสเตอร์ PC มีความสำคัญมาก โดยเฉพาะอย่างยิ่งในการตรวจสอบการทำงานของโปรแกรม ดำเนินไปตามขั้นตอนตามที่กำหนดไว้หรือไม่

สแต็กพอยน์เตอร์

หรือรีจิสเตอร์ตัวชี้สแต็ก มีขนาด 8 บิต มีแอดเดรสอยู่ที่ 81H ใช้ในการเก็บค่าตำแหน่งของตัวชี้สแต็ก ซึ่งสามารถเปลี่ยนแปลงได้เมื่อซีพียูมีการกระโดไปทำงานที่โปรแกรมย่อย หรือกระโดดโปรแกรมย่อยกลับมายังโปรแกรมหลักเมื่อมีการรีเซตเกิดขึ้น ค่าของรีจิสเตอร์ SP จะเท่ากับ 07H ดังนั้นแอดเดรสแรกของพื้นที่ที่สำรองไว้ทำหน้าที่เป็นสแต็กจะเท่ากับ 08H

รีจิสเตอร์ชี้ข้อมูลหรือดาต้าพอยน์เตอร์

มีขนาด 16 บิต โดยแบ่งเป็นรีจิสเตอร์ชี้ข้อมูลไบต์สูง และรีจิสเตอร์ชี้ข้อมูลไบต์ต่ำ แต่ละตัวมีขนาด 8 บิต มีแอดเดรสอยู่ที่ 82H สำหรับ DPL และ 83H รีจิสเตอร์ DPTR นี้ใช้ในการเก็บค่าแอดเดรสของหน่วยความจำหรืออุปกรณ์ภายนอกที่ไมโครคอนโทรลเลอร์ต้องการติดต่อด้วย

รีจิสเตอร์พอร์ต

เป็นรีจิสเตอร์ขนาด 8 บิตที่ใช้ในการเก็บข้อมูลของแต่ละพอร์ตของไมโครคอนโทรเลอร์ MCS-51 มี 4 ตัวคือ รีจิสเตอร์พอร์ต 0 หรือ P0 มีแอดเดรสอยู่ที่ 80H, รีจิสเตอร์พอร์ต 1 หรือ P1 มีแอดเดรสอยู่ที่ 90H , รีจิสเตอร์พอร์ต 2 หรือ P2 มีแอดเดรสอยู่ที่ A0H และ รีจิสเตอร์พอร์ต 3 หรือ P3 มีแอดเดรสอยู่ที่ B0H รีจิสเตอร์สามารถเข้าถึงได้ในระดับบิต เมื่อต้องการอ่านหนังสือหรือเขียนข้อมูลออกไปยังพอร์ตของไมโครคอนโทรลเลอร์ จะต้องการกระทำผ่านรีจิสเตอร์นี้ทุกครั้ง

รีจิสเตอร์บัฟเฟอร์ข้อมูลอนุกรม

เป็นรีจิสเตอร์ขาด 8 บิต มีแอดเดรสอยู่ที่ 99H ใช้ในการเก็บข้อมูลที่ส่งออกหรือรับเข้าของวงจรสื่อสารอนุกรมที่อยู่ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดยภายในรีจิสเตอร์ SBUF นี้จะแบ่งออกเป็น 2 ส่วน คือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล และรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล เมื่อมีการเขียนข้อมูลมายังรีจอสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูลเพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือขา P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับส่งข้อมูลเพื่อส่งไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับส่งข้อมูลอนุกรมจากภายนอกนั้นจะผ่านทางขา RxD หรือ P3.0 ของไมโครคอนโทรลเลอร์ MCS-51

สำหรับรายละเอียดของรีจิสเตอร์ SBUF และวงจรสื่อสารอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชจะกล่าวถึงในบทที่ว่าด้วยเรื่องการสื่อสารผ่านพอร์ตอนุกรม

รีจิสเตอร์ไทเมอร์

เป็นรีจิสเตอร์ขนาด 16 บิต แบ่งเป็นไบต์สูงและไบต์ต่ำเช่นเดียวกับรีจิสเตอร์ DPTR รีจิสเตอร์ไทเมอร์ใช้ในการเก็บค่าของตัวนับหรือเคาน์เตอร์ ภายในไมโครคอนโทรลเลอร์ เพื่อใช้ในการสร้างฐานเวลา , จับเวลา หรือนับจำนวนพัลส์สัญญาณนาฬิกาภายใน บางทีเรียกตัวนี้ว่า รีจิสเตอร์ไทเมอร์ / เคาน์เตอร์

ในไมโครคอนโทรลเลอร์เบอร์ AT89C51 มีรีจิสเตอร์ไทเมอร์ / เคาน์เตอร์ 2 ตัว แบ่งเป็น T0 หรือ Timer 0 และ T1 หรือ Timer 1 ในรีจิสเตอร์ยังแบ่งเป็นรีจิสเตอร์ไทเมอร์ไบต์ต่ำ และรีจิสเตอร์ไทเมอร์ไบต์สูงเหมือนกัน โดยรีจิสเตอร์ TL0 มีแอดเดรสอยู่ที่ 8AH รีจิสเตอร์ TH0 มีแอดเดรสอยู่ที่ 8BHในขณะที่ TL1และ TH1 มีแอดเดรสอยู่ที่ 8CH และ 8DH สำหรับในเบอร์ AT89C52 และในอนุกรม AT89Sxx จะมีรีจิสเตอร์ไทเมอร์/เคาน์เตอร์ถึง 3 ตัว โดยมีรีจิสเตอร์ TL2 และ Th2 ซึ่งมีแอดเดรสอยู่ที่ 0CCH และ 0CDH เพิ่มเติมเข้ามา

รีจิสเตอร์แคปเจอร์

เป็นรีจิสเตอร์ขนาด16บิต มีเฉพาะ ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และ ในอนุกรม AT89Sxx เท่านั้น เนื่องจากต้องใช้ร่วมกับไทเมอร์ / เคาน์เตอร์ 2 โดยรีจิสเตอร์แคปเจอร์นี้มีชื่อเรียกอย่างย่อว่า รีจิสเตอร์ RCA51P2 ซึ่งแบ่งออกเป็นไบต์ต่ำ RCAP2L มีแอดเดรสอยู่ที่ 0CAH และไบต์สูงคือRCAP2H มีแอดเดรสอยู่ 0CBH

รีจิสเตอร์แคปเจอร์จะถูกใช้งานเมื่อกำหนดให้ไทเมอร์ 2 ทำงานในโหมดแคปเจอร์ ซึ่งเป็นโหมดที่กำหนดให้ไมโครคอนโทรลเลอร์ทำการตรวจจับการเปลี่ยนแปลงสถานะทางลอจิกที่ขา T2EX ทั้นี้เพื่อใช้ประ โยชน์ในโยชน์ในการวัดคาบเวลา ความถี่ และการเปลี่ยนแปลี่ยนของสัญญาณพัลส์ที่ขา T2EX

รีจิสเตอร์ควบคุม

รีจิสเตอร์ PCON เป็นรีจิสเตอร์ที่เกี่ยวข้องกับอัตราการรับส่งข้อมูลของวงจรสื่อสารอนุกรมและกำหนดการทำงานในโหมดประหยัดพลังงานของไมโครคอนโทรลเลอร์ MCS-51

รีจิสเตอร์ SCON เป็นรีจิสเตอร์ที่ใช้ในการการควบคุมการทำงานของวงจรสื่อสารอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

รีจิสเตอร์ TCON และ T2CON เป็นรีจิสเตอร์ที่ใช้ในการควบคุมการทำงานของไทเมอร์ / เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์MCS-51 แบบแฟลช โดย T2CON ใช้สำหรับไทเมอร์ /เคาน์เตอร์ 2 ของไมโครคอนโทรลเลอร์ MSC-51 แบบแฟลช AT89C52 และในอนุกรม AT89Sxx

รีจิสเตอร์ TMOD และ T2MOD เป็นรีจิสเตอร์ที่ใช้กำหนดโหมดหรือลักษณะในการทำงานของไทเมอร์ / เคาน์เตอร์ภายในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดย T2CON ใช้สำหรับไทเมอร์ / เคาน์เตอร์ 2 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ AT89C52 และในอนุกรม AT89Sxx

รีจิสเตอร์ IE และ IP เป็นแบบรีจิสเตอร์ที่เกี่ยวข้องกับการตอบสนองการอินเตอร์รัปต์ โดย IE เป็นรีจิสเตอร์สำหรับเอ็นเอเบิลหรือใช้ในการกำหนดลักษณะของการตอบสนองการอินเตอร์รัปต์ ในขณะที่ IP เป็นรีจิสเตอร์สำหรับกำหนดลำดับความสำคัญของการตอบสนองการอินเตอร์รัปต์ว่า จะใช้ซีพียูตอบสนองการเกิดอินเตอร์รัปต์ในลักษณะใดก่อนหรือหลัง

2.1.6 การเข้าถึงข้อมูล (addressing)

เป็นวิธีการที่ไมโครคอนโทรลเลอร์ใช้ในการระบุตำแหน่งเพื่อเข้าถึงข้อมูลที่ต้องดำเนินการ สำหรับไมโครคอนโทรลเลอร์ MCS-51 มีวิธีเข้าถึง 5 แบบคือ

1. แบบทันทีทันใด (immediate addressing mode)
2. แบบโดยตรง (direct addressing mode)
3. แบบผ่านรีจิสเตอร์ (register addressing mode)
4. แบบโดยอ้อม (indirect addressing mode)
5. แบบอินเด็กซ์หรือแบบโดยอ้อมผ่านค่าดัชนี (indexed addressing mode)

การเข้าถึงข้อมูลแบบทันทีทันใด

เป็นวิธีการเข้าถึงข้อมูลที่มีความซับซ้อนน้อยที่สุด สามารถทำความเข้าใจได้ง่าย เนื่องจากสามารถโอนย้ายหรือเข้าถึงข้อมูลของรีจิสเตอร์ได้ด้วยค่าของข้อมูลตรงๆ มีรูปแบบตัวอย่างดังนี้

คำสั่ง รีจิสเตอร์หรือตำแหน่งของหน่วยความจำ , # ข้อมูล (ฐานสอง,ฐานสิบ หรือ ฐานสิบหก)

เป็นการนำค่าของข้อมูลที่ตามหลังเครื่องหมาย # ไปดำเนินการกับรีจิสเตอร์หรือตำแหน่งของหน่วยความจำปลายทาง

การเข้าถึงข้อมูลแบบโดยตรง

เป็นการเข้าถึงข้อมูลโดยใช้หมายเลขตำแหน่งของหน่วยความจำภายในต้นทางเพื่อเรียกหรือดอนย้ายข้อมูลได้โดยตรงแล้วนำข้อมูลจากหน่วยความจำในต้นทางนั้นไปเก็บยังรีจิสเตอร์หรือหน่วยความจำภายในปลายทาง การเข้าถึงแบบนี้จะใช้ได้กับหน่วยความจำข้อมูลภายในของไมโครคอนโทรลเลอร์ MCS-51 เท่านั้น สามารถอ้างแอดเดรสได้ 256 ตำแหน่ง ตั้งแต่ 00H-FFH ซึ่งก็คือแอดเดรสของหน่วยความจำข้อมูลภายในนั่นเอง

การเข้าถึงข้อมูลแบบผ่านรีจิสเตอร์

การเข้าถึงข้อมูลแบบนี้จะคล้ายกับแบบโดยตรง ต่างกันตรงที่ข้อมูลต้นทางจะถูกเก็บไว้ในรีจิสเตอร์แบงก์ R0-R7 ส่วนรีจิสเตอร์ปลายทางต้องเป็นรีจิสเตอร์ A หรือแอกคิวมูเลเตอร์ มีรูปแบบตัวอย่างดังนี้

MOV A , Rn ; Rn เป็นรีจิสเตอร์แบงก์ใดๆ (R0-R7)

การเข้าถึงข้อมูลแบบโดยอ้อม

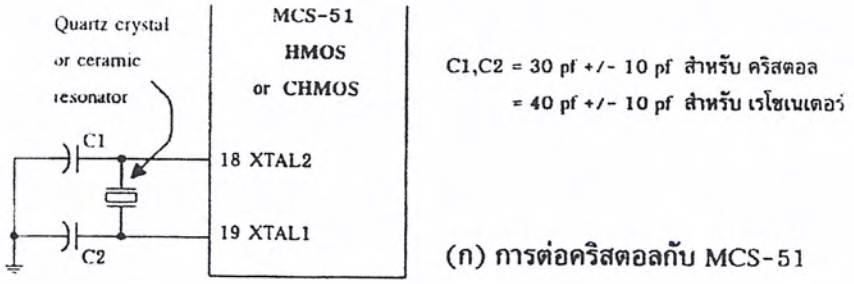
การเข้าถึงข้อมูลแบบนี้มีความซับซ้อนเพิ่มเติมจากแบบโดยตรง โดยค่าแอดเดรสของข้อมูลต้นทางจะถูกเก็บไว้ในรีจิสเตอร์ R0 หรือ R1 หรือ DPTR หรือใช้รีจิสเตอร์ R0,R1 และ DPTR เป็นตัวชี้ของหน่วยความจำแทน และต้องใส่สัญลักษณ์ @ หน้ารีจิสเตอร์ที่ที่ใช้เป็นตัวชี้แอดเดรสต้นทางด้วย

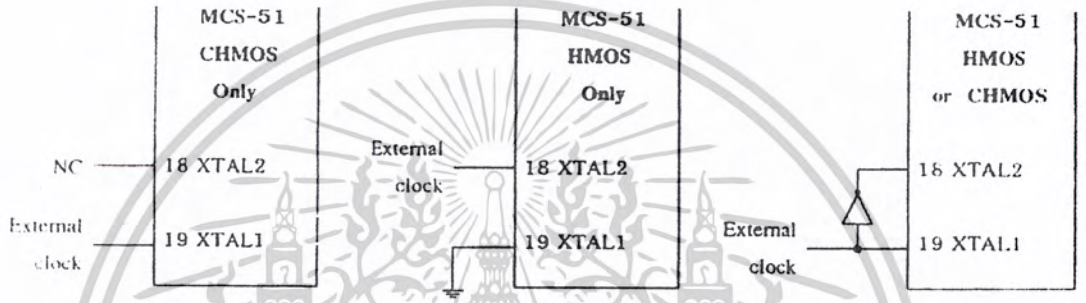
2.1.7 ส่วนควบคุมพื้นฐานของไมโครคอนโทรลเลอร์

การควบคุมพื้นฐานของไมโครคอนโทรลเลอร์ที่สำคัญจะมี 3-4 ส่วนดังนี้

1. **แหล่งจ่ายไฟ (Power supply)** เป็นส่วนสำคัญแรกสุดที่ต้องคิดถึงก่อนเสมอโดยปกติ MCS-51 ต้องการแหล่งจ่ายไฟ 5 โวลท์ + / -10% (แต่จะมีบางเบอร์ที่ใช้แหล่งจ่ายไฟที่ต่ำกว่า 5 โวลท์) การป้อนแหล่งจ่ายไฟจะทำผ่านขา 40 (Vcc) และขา 20 (Gnd)

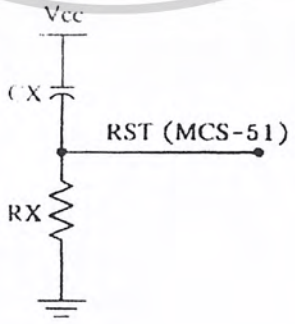
2. **สัญญาณนาฬิกา** จะเป็นตัวกำหนดจังหวะการทำงานของระบบ เนื่องจากภายใน MCS-51 มีวงจรออสซิลเลเตอร์อยู่แล้ว (On-chip oscillator) จึงง่ายต่อการใช้งานเพียงแต่ต่อคริสตอลหรือเรโซเนเตอร์ที่ขา 19 (XTAL1) และขา 18 (XTAL2) และต่อตัวเก็บประจุที่ขาทั้ง 2 ลงกราวน์ ดังรูป2-14 (ก) นอกจากการใช้ออสซิลเลเตอร์อยู่แล้วสร้างสัญญาณนาฬิกาแล้ว ยังสามารถนำสัญญาณนาฬิกาจากภายนอกต่อเข้าที่ขา XTAL1 ได้เช่นกัน แต่จะมีวิธีการต่อแตกต่างกันออกไปขึ้นอยู่กับโครงสร้างของ MCS-51 ดังรูป2-14 (ข)





รูปที่ 2-14 การต่อใช้ออสซิลเลเตอร์แบบต่างๆ

3. การรีเซต (Reset) สาเหตุที่ต้องรีเซตเพื่อเริ่มต้นการทำงาน โดยจะทำให้โปรแกรมเคาน์เตอร์เป็น 0000 หรืออาจกล่าวว่าทำให้ MCS-51 ไปที่จุดเริ่มต้น เงื่อนไขการรีเซตคือ ต้องให้สัญญาณลอจิก "1" ที่ขา 9 (RST) อย่างน้อยเป็นเวลา 2 วงรอบแมชชีน หรือ สัญญาณจากวงออสซิลเลเตอร์ 24 ลูก (ในขณะที่วงจรออสซิลเลเตอร์กำลังทำงานอยู่) โดยวงรอบแมชชีนสามารถคำนวณได้จากสัญญาณนาฬิกา ลองดู ตัวอย่างถ้าระบบ MCS-51 ใช้คริสตอล 12 MHz จะได้วงจรรีเซตดังนี้ จากความต้องการของขาRST = 24 คาบของสัญญาณออสซิลเลเตอร์



รูปที่ 2-15 วงจรรีเซ็ต

1 คาบของสัญญาณออสซิลเลเตอร์ = 1 / 12MHz = 83.3 uSec

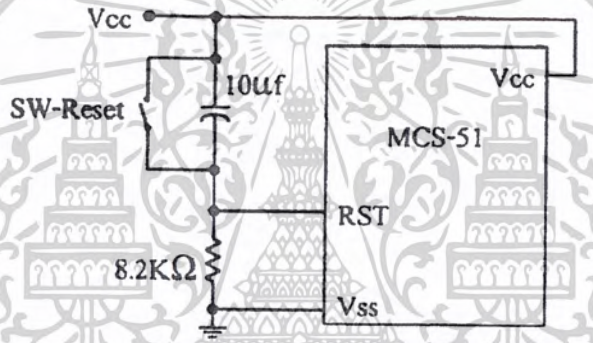
12 คาบ = 83.3 uSec*24 = 2 uSec

ดังนั้นจะได้เวลาในการรีเซตอย่างน้อยคือ 2 uSec วงจรรีเซตที่ออกแบบจะเป็นแบบ Power on reset กล่าวคือ เมื่อเริ่มต้นจ่ายแหล่งจ่ายให้กับวงจร MCS-51 รีเซต และเริ่มต้นทำงาน โดยวงจรจะใช้ RC time constant ซึ่งสามารถคำนวณได้ค่าดังนี้

CX = 10 uF

RX = 8.2Kilo ohm

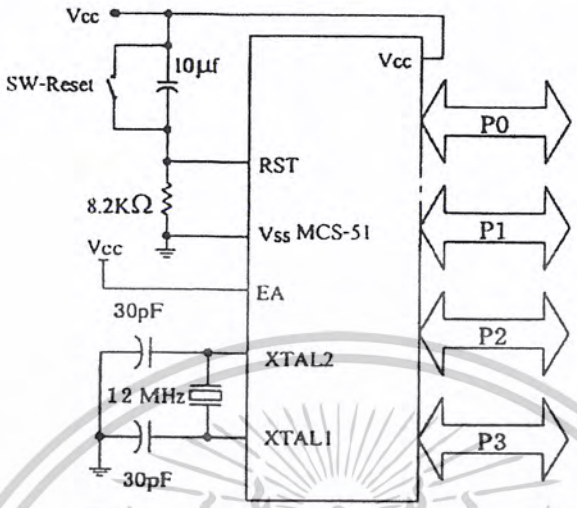
เมื่อนำมารวมกับการรีเซตด้วยสวิทช์ ได้เป็นวงจรรีเซตที่สมบูรณ์ดังนี้



รูปที่ 2-16 วงจรเซ็ตที่สมบูรณ์

4. การเลือกใช้หน่วยความจำโปรแกรมภายใน MCS-51 สามารถเลือกใช้หน่วยความจำโปรแกรมภายในหรือภายนอกได้ โดยใช้ขา $\overline{EA}$ กล่าวคือ $\overline{EA}$ ต่อกับ Vcc เมื่อต้องใช้หน่วยความจำเก็บโปรแกรมภายใน อาทิเบอร์ 8051, 8751, 8951 และ EA ต่อกับ GND เมื่อต้องการใช้หน่วยความจำเก็บโปรแกรมภายนอก (External access) CPU เช่นเบอร์ 8031, 8032

จากความต้องการพื้นฐานที่กล่าวถึง จะเพียงพอที่ทำให้ MCS-51 ใช้งานได้แล้ว (สำหรับเบอร์ที่มีหน่วยความจำโปรแกรมภายใน; 8951) ซึ่งจะมีพอร์ตใช้งาน 4 พอร์ต คือ พอร์ต 0-3 ซึ่งได้วงจรดังรูปที่ 2-17



รูปที่ 2-17 วงจรสมบูรณ์ที่ออกแบบโดยใช้ MCS – 51 ตัวเดียว

การออกแบบทางฮาร์ดแวร์สำหรับระบบควบคุมที่ใช้ไมโครคอนโทรลเลอร์ตัวเดียวนั้น ต้องเลือกไมโครคอนโทรลเลอร์ แบบที่มีหน่วยความจำสำหรับเก็บโปรแกรมอยู่ภายใน ซึ่งมีอยู่หลายเบอร์ เช่น 8051 , 8751 , 8951 หรือ 89C2051

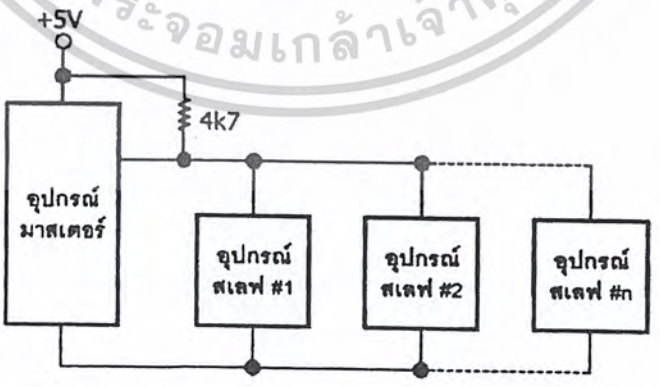
2.2 ระบบสื่อสารข้อมูลอนุกรมแบบหนึ่งสาย (1-wire™ Serial Bus)

ระบบการสื่อสารข้อมูลแบบนี้ผู้ค้นคิดคือ ดัลลัสเซมิคอนดักเตอร์ ดังนั้นในบางครั้งจึงเรียกระบบสื่อสารข้อมูลแบบนี้ว่า ระบบสื่อสารข้อมูลดัลลัสหนึ่งสาย (The Dallas 1-Wire Bus) ระบบสื่อสารข้อมูลแบบนี้เป็นระบบที่มีความชาญฉลาด และใช้สายสัญญาณเพียง 1 เส้นเท่านั้น โดยไม่ต้องมีสายสัญญาณนาฬิกามาควบคุมจังหวะการถ่ายทอดข้อมูลเหมือนกับระบบสื่อสารข้อมูลแบบอนุกรมแบบอื่นๆ เนื่องจากสายข้อมูลนั้นจะทำหน้าที่เสมือนหนึ่งนาฬิกาในตัว ส่วนค่าของข้อมูลจะพิจารณาจากลักษณะของรูปสัญญาณที่ปรากฏบนสายสัญญาณในแต่ละช่องของเวลาหรือต่อไปนี้จะขอเรียกว่า ไทม์สล็อต(time-slot) โดยคาบเวลาต่ำสุดและสูงสุดของสถานะต่างๆในการสื่อสารข้อมูลในแตละไทม์สล็อต มีการกำหนดขอบเขตไว้อย่างชัดเจน การถ่ายทอดข้อมูลจะเกิดขึ้นในแต่ละไทม์สล็อตนั้น รูปแบบการถ่ายทอดข้อมูลจะเป็นแบบอะซิงโครนัสในระดับบิต ไม่มีการกำหนดความยาวของข้อมูลเป็นระดับไบต์ ระบบสื่อสารแบบนี้เหมาะที่จะใช้ในการสื่อสารข้อมูลระหว่างไอซีบนแผงวงจรเดียวกัน หรือสร้างเป็นโครงข่ายสื่อสารแบบทวิสต์แพร์ก็ได้

2.2.1 คุณสมบัติทางเทคนิคของระบบบัสหนึ่งสาย

สายสัญญาณบนระบบบัสแบบหนึ่งสายนี้จะเป็นสายสัญญาณแบบสองทิศทาง แต่ข้อมูลจะสามารถเดินทางได้ในทิศทางเดียวในช่วงระยะเวลาหนึ่งๆนั่นคือ มีลักษณะคล้ายกับระบบสื่อสารแบบฮาล์ฟดูเพล็กซ์ (half-duplex) ตัวอย่างที่เห็นได้ชัดคือ การใช้งานวิทยุสื่อสารหรือวิทยุสมัครเล่น อุปกรณ์บนระบบบัสต้องมีการระบุอย่างชัดเจนว่าตัวใดเป็นอุปกรณ์ทาสเตอร์ ตัวใดเป็นอุปกรณ์สเลฟ โดยส่วนใหญ่เป็นอุปกรณ์มาสเตอร์คือไมโครคอนโทรลเลอร์ ส่วนอุปกรณ์สเลฟได้แก่ ไอซีตรวจจับอุณหภูมิ,ไอซีหน่วยความจำแรม เป็นต้นอุปกรณ์มาสเตอร์จะเป็นตัวจัดเตรียมพร้อมของสายสัญญาณและควบคุมการถ่ายทอดข้อมูลบนสายสัญญาณนั้น ข้อมูลทั้งหมดไม่ว่าจะเป็นข้อมูลควบคุมหรือข้อมูลใช้งานจะถูกส่งลงบนสายสัญญาณที่มีอยู่เพียงเส้นเดียวนี้ทั้งหมด ในระหว่างการทำงานอุปกรณ์มาสเตอร์และสเลฟสามารถเป็นได้ทั้งตัวรับและตัวส่ง ขึ้นอยู่กับเงื่อนไขของทำงานในขณะนั้น ยกตัวอย่าง ถ้าหากมีการเขียนข้อมูลจากอุปกรณ์มาสเตอร์ไปยังอุปกรณ์สเลฟ ตัวส่งคืออุปกรณ์มาสเตอร์ ตัวรับคืออุปกรณ์สเลฟ ในทางตรงกันข้าม หากเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ ตัวส่งจะกลายเป็นอุปกรณ์สเลฟ และตัวรับจะเป็นอุปกรณ์มาสเตอร์ในระบบบัส 1 ระบบต้องมีอุปกรณ์มาสเตอร์เพียงตัวเดียวเท่านั้น

สายสัญญาณของระบบบัสนี้ต้องกำหนดสภาวะปกติไว้ที่ลอจิกสูง สามารถทำได้โดยการต่อตัวต้านทานค่าประมาณ 4.7 kΩ พูลอัพกับไฟเลี้ยง +5v ดังนั้นอุปกรณ์ที่นำเข้ามาต่อบนระบบบัสนี้จึงต้องออกแบบให้ภาคเอาต์พุตต่อกับสายสัญญาณมีลักษณะเป็นคอลเล็กเตอร์เปิดหรือเดรนเปิด ในรูปที่ 2-18 แสดงไดอะแกรมการสื่อสารข้อมูลแบบหนึ่งสายเบื้องต้น



รูปที่ 2-18 การเชื่อมต่อบนระนาบบัสหนึ่งสาย

2.2.2 คุณสมบัติของไทม์สล็อต

อุปกรณ์มาสเตอร์จะเป็นอุปกรณ์เพียงตัวเดียวบนระบบบัสหนึ่งสายนี้ที่สามารถทำการอินนิเชียลสายสัญญาณได้ โดยอุปกรณ์มาสเตอร์จะกำเนิดจุดเริ่มต้นของไทม์สล็อตด้วยการทำให้สายสัญญาณเป็นลอจิกต่ำในช่วงเวลาหนึ่ง จากนั้นจะทำให้กลับมาเป็นลอจิกสูง ถ้าหากอุปกรณ์สเลฟต้องการส่งข้อมูลมายังอุปกรณ์มาสเตอร์ อุปกรณ์สเลฟจะเป็นตัวควบคุมสภาวะสายสัญญาณต่อไปจนเสร็จสิ้นกระบวนการ แต่ถ้าหากอุปกรณ์มาสเตอร์ต้องการส่งข้อมูลก็จะสามารถดำเนินการต่อไปได้เลย

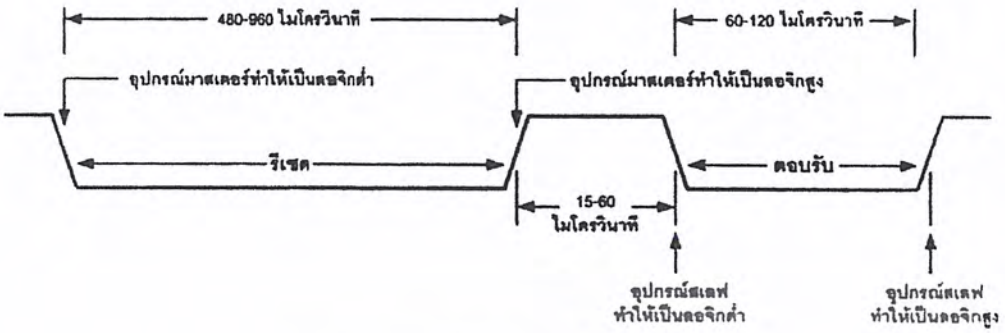
ฟังก์ชันของไทม์สล็อตที่มีการกำหนดโดยอุปกรณ์มาสเตอร์มีด้วยกัน 4 ฟังก์ชันคือ ไทม์สล็อตของการรีเซต (RESET), การอ่านข้อมูล (READ DATA), การเขียนข้อมูล "1"(WRITE ONE) และ การเขียนข้อมูล "0" (WRITE ZERO) ไทม์สล็อตรีเซตใช้ในการเริ่มต้นติดต่อกับอุปกรณ์สเลฟ ในขณะที่ไทม์สล็อตการอ่านจะสำหรับอ่านข้อมูลที่ส่งมาจากอุปกรณ์สเลฟ ส่วนไทม์สล็อตการเขียนข้อมูล "1" และ "0" ใช้สำหรับเขียนข้อมูลไปยังอุปกรณ์สเลฟผ่านสายสัญญาณของระบบ

ทางด้านอุปกรณ์สเลฟมีฟังก์ชันของไทม์สล็อตอยู่ทั้งสิ้น 3 ฟังก์ชันคือ ไทม์สล็อตของการตอบสนอง (PRESENSE) , การเขียนข้อมูล "1"(WRITE ONE) และ การเขียนข้อมูล "0" (WRITE ZERO) ไทม์สล็อตของการตอบสนองใช้สำหรับตอบสนองการติดต่อจากอุปกรณ์มาสเตอร์ โดยอุปกรณ์สเลฟตัวที่ถูกเลือกจากอุปกรณ์มาสเตอร์จะต้องส่งสัญญาณตอบสนองลงบนสายสัญญาณเพื่อแจ้งให้อุปกรณ์มาสเตอร์ทราบว่า ขณะนี้สามารถติดต่อกันได้แล้ว ส่วนไทม์สล็อตการเขียนข้อมูล "1", "0" ใช้สำหรับส่งข้อมูลไปยังอุปกรณ์มาสเตอร์ผ่านสายสัญญาณของระบบ ซึ่งจะสัมพันธ์กับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์

การแยกแยะฟังก์ชันของแต่ละไทม์สล็อตจะใช้ความยาวของคาบเวลาและลักษณะของรูปสัญญาณเป็นตัวกำหนด และทุกครั้งที่มีการเปลี่ยนแปลงฟังก์ชันต้องทำให้สายสัญญาณอยู่ในสภาวะว่างเสมอ ซึ่งก็คือการทำให้สายสัญญาณเป็นลอจิกสูงอย่างน้อยเป็นเวลา 1 ไมโครวินาที

ไทม์สล็อตการรีเซตและตอบสนอง

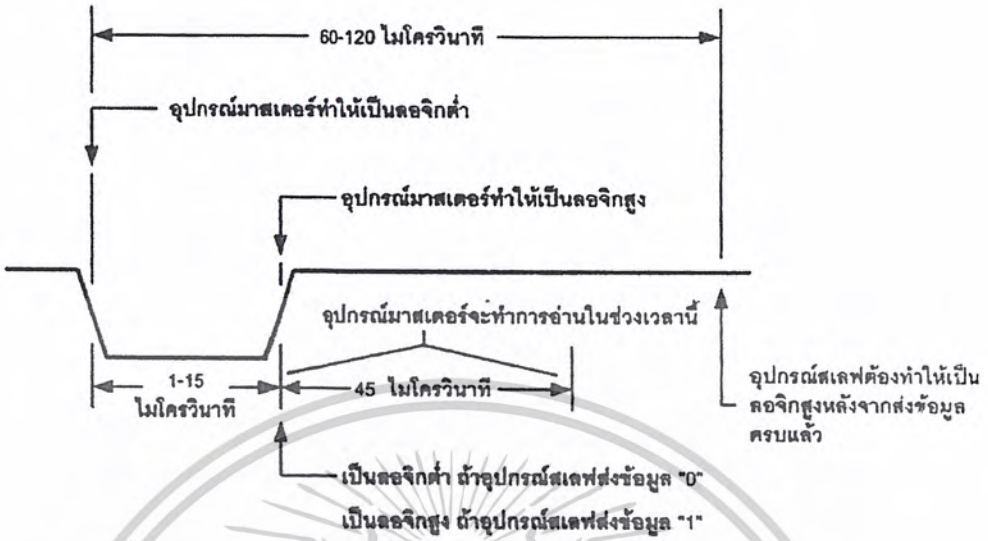
อุปกรณ์มาสเตอร์ทำให้เกิดการรีเซตบนสายสัญญาณเพื่อแจ้งแก่อุปกรณ์สเลฟ โดยการทำให้สายสัญญาณเป็นลอจิกต่ำต่อเนื่องอย่างน้อย 480 ไมโครวินาที และจะต้องทำให้สายสัญญาณกลับมาเป็นลอจิกสูงภายใน 480 ไมโครวินาทีหลังจากนั้น ถ้าหากมีอุปกรณ์สเลฟอยู่บนสายสัญญาณ จะมีการตอบสนองสัญญาณรีเซตนั้นด้วยสัญญาณตอบสนอง (PRESENCE) โดยการทำให้สายสัญญาณเป็นลอจิกต่ำต่อเนื่องประมาณ 60-240 ไมโครวินาที หลังจากสัญญาณนีเซตปรากฏประมาณ 15-60 ไมโครวินาที ในรูป 2-19 แสดงไทม์สล็อตของการรีเซตและตอบสนอง



รูปที่ 2-19 ไทม์สล็อตการรีเซตและการตอบรับของอุปกรณ์บนระบบบัสหนึ่งสาย

ไทม์สล็อตการอ่านของอุปกรณ์มาสเตอร์และการเขียนข้อมูลของอุปกรณ์สเลฟ

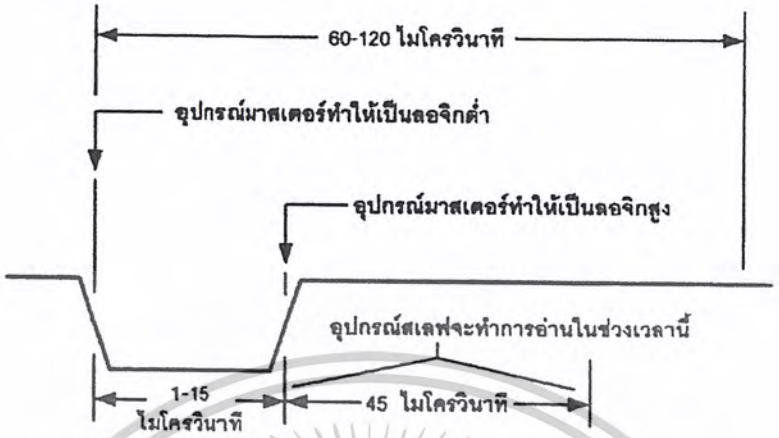
เมื่อต้องการอ่านข้อมูลจากอุปกรณ์สเลฟ อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 1-15 ไมโครวินาที จากนั้นต้องทำสถานะของสายให้กลับมาเป็นลอจิกสูง อุปกรณ์สเลฟจะส่งข้อมูลมาให้อุปกรณ์มาสเตอร์โดยถ้าข้อมูลเป็น "0" อุปกรณ์สเลฟจะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 45 ไมโครวินาที แล้วทำให้สายสัญญาณกลับสู่สภาวะลอจิกสูงอีกครั้ง แต่ถ้าเป็นข้อมูล "1" อุปกรณ์สเลฟจะทำให้สายสัญญาณเป็นลอจิกสูงต่อเนื่องไปอีก 45 ไมโครวินาที รวมเวลาทั้งหมดในไทม์สล็อตนี้ประมาณ 60-120 ไมโครวินาที นั่นคือในไทม์สล็อตนี้จะต้องใช้เวลาไม่เกิน 120 ไมโครวินาที ในขณะที่อุปกรณ์มาสเตอร์จะใช้เวลาในการอ่านข้อมูลอยู่ระหว่าง 15 และ 60 ไมโครวินาทีหลังจากเริ่มต้นไทม์สล็อตนี้ ในรูปที่ 2-20 แสดงรูปสัญญาณของไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์ก็จะมีลักษณะเหมือนกับการเขียนข้อมูลของอุปกรณ์สเลฟ และ ไทม์สล็อตทั้งสองจะเกิดขึ้นในช่วงเวลาเดียวกัน กล่าวคือเมื่ออุปกรณ์มาสเตอร์อ่าน อุปกรณ์สเลฟ ก็ต้องทำการเขียน



รูปที่ 2-20 ไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์ ซึ่งตรงกับไทม์สล็อตการเขียนข้อมูลของอุปกรณ์สเลฟ

ไทม์สล็อตการเขียนข้อมูลของอุปกรณ์มาสเตอร์

เมื่ออุปกรณ์มาสเตอร์ต้องการเขียนข้อมูล อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 1-15 ไมโครวินาที จากนั้นต้องทำให้สถานะของสายกลับมาเป็นลอจิกสูง แล้วดำเนินการเขียนข้อมูลได้ในทันทีถ้าข้อมูลที่ต้องการเขียนไปยังอุปกรณ์สเลฟเป็น "0" อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณต่ำนานประมาณ 45 ไมโครวินาที แล้วทำให้สายสัญญาณกลับมาสู่สภาวะลอจิกสูงอีกครั้ง แต่ถ้าต้องการเขียนข้อมูล "1" อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกสูง ต่อเนื่องไปอีก 45 ไมโครวินาที รวมเวลาทั้งหมดในไทม์สล็อตนี้ประมาณ 60-120 ไมโครวินาที ในรูปที่ 2-21 แสดงรูปสัญญาณของ ไทม์สล็อตการเขียนข้อมูลของอุปกรณ์มาสเตอร์ซึ่งก็จะมีลักษณะเหมือนกับการอ่านข้อมูลของอุปกรณ์สเลฟ และไทม์สล็อตทั้งสองจะเกิดขึ้นในช่วงเวลาเดียวกัน กล่าวคือ เมื่ออุปกรณ์มาสเตอร์เขียน อุปกรณ์สเลฟก็ต้องทำการอ่านข้อมูล



รูปที่ 2-21 ไทม์สล็อตการเขียนข้อมูล "1" ของอุปกรณ์มาสเตอร์ ซึ่งตรงกับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์สเลฟ

2.2.3 รูปแบบของการสื่อสารข้อมูลแบบหนึ่งสาย (1-wire™ communication protocal)

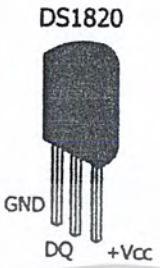
ในการติดต่อสื่อสารข้อมูลในระบบบัสหนึ่งสายอุปกรณ์มาสเตอร์จะสามารถติดต่อกับอุปกรณ์สเลฟได้ครั้งละ 1 ตัวเท่านั้น ดังนั้นอุปกรณ์สเลฟแต่ละตัวจะต้องมีข้อมูลกำหนดแอดเดรสเฉพาะตัว โดยจะเก็บไว้ในหน่วยความจำรอมภายในอุปกรณ์สเลฟตัวนั้นๆ โดยปกติอุปกรณ์สเลฟในระบบบัสหนึ่งสายของดัลลัสนี้จะมีหน่วยความจำขนาด 64 บิต หรือ 8 ไบต์ สำหรับเก็บข้อมูลต่างๆ ที่สำคัญของอุปกรณ์แต่ละตัว ซึ่งประกอบด้วย

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

ผู้ใช้งานสามารถอ่านข้อมูลประจำตัวของอุปกรณ์สเลฟด้วยการใช้คำสั่งอ่านหน่วยความจำรอม (Read ROM) ในกรณีที่บนสายสัญญาณมีอุปกรณ์สเลฟเพียงตัวเดียวไม่จำเป็นต้องอ้างแอดเดรสในการติดต่อ

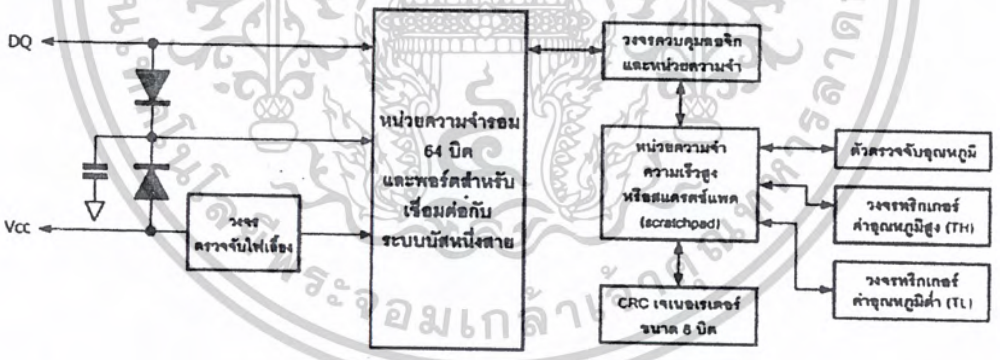
รูปแบบการติดต่อบนระบบบัสหนึ่งสายจะเริ่มต้นขึ้นเมื่ออุปกรณ์มาสเตอร์เริ่มทำการรีเซตและกำหนดแอดเดรสของอุปกรณ์ที่ทำการติดต่อ ถ้าหากมีอุปกรณ์สเลฟเพียงตัวเดียวสามารถข้ามขั้นตอนการติดต่อกับหน่วยความจำรอมในอุปกรณ์สเลฟได้ จะเรียกวิธีการดังกล่าวว่า การไม่ติดต่อหน่วยความจำรอม หรือ สคิปรอม (Skip ROM) จากนั้นรอการตอบรับจากอุปกรณ์สเลฟ เมื่อการตอบรับสมบูรณ์ก็จะสามารถเริ่มต้นขั้นตอนการอ่านหรือเขียนข้อมูลได้ต่อไป

2.3 ไอซีตรวจจับอุณหภูมิ DS1820



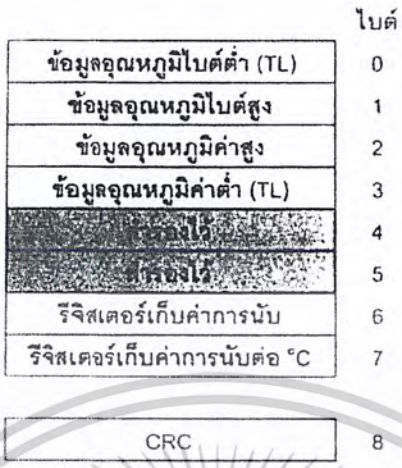
รูปที่ 2-22 การจัดขาของDS 1820

เป็นไอซีตรวจจับอุณหภูมิที่ใช้การติดต่อแบบระบบบัสหนึ่งสาย มีขาต่อใช้งานเพียง 3ขา คือ DQ ซึ่งเป็นขาเชื่อมต่อกับระบบบัส, ขาต่อไฟเลี้ยง และขากราวน์ ดังแสดงการจัดขาไอซี DS1820 ในรูปที่ 2-22 และมีโครงสร้างการทำงานภายในแสดงในรูปที่ 2-23



รูปที่ 2-23 โครงสร้างการทำงานภายในของไอซีตรวจจับอุณหภูมิ DS1820

หัวใจสำคัญของ DS1820 อยู่ที่ตัวตรวจจับอุณหภูมิและหน่วยความจำความเร็วสูงที่เรียกว่า สแครดช์แพด (scratchpad) ซึ่งมีขนาด 9 ไบต์ มีการจัดสรรหน่วยความจำส่วนนี้แสดงในรูป 2-24



รูปที่ 2-24 การจัดสรรพื้นที่ของสแครตช์แพดใน DS1820

เมื่อวัดอุณหภูมิได้ก็จะนำค่าที่วัดได้มาเก็บไว้ในสแครตช์แพดที่ไบต์ 0 และ 1 ทั้งนี้เนื่องจากไอซี DS1820 สามารถให้ข้อมูลของอุณหภูมิได้ละเอียดถึง 16 บิต เมื่อนำมาแปลงเป็นข้อมูลฐานสิบจึงสามารถแสดงความละเอียดของค่าอุณหภูมิได้ถึง 0.5°C และ 0.9°F โดยมีย่านวัดอุณหภูมิ –55 ถึง +125 °C หรือ –67 ถึง +257 °F โดยค่าของ °F ต้องใช้การแปลงหน่วยเข้ามาช่วยใช้เวลาในการแปลงค่าอุณหภูมิเป็นข้อมูลดิจิตอลประมาณ 200 มิลลิวินาที สามารถกำหนดขอบเขตของอุณหภูมิที่ทำการวัดได้ และให้แจ้งเตือนเมื่อค่าของอุณหภูมิสูงขึ้นหรือลดต่ำลงถึงค่าที่กำหนด โดยค่าอุณหภูมิที่กำหนดนี้จะเก็บไว้ที่สแครตช์แพดในไบต์ 2 และ 3

คำสั่งเพื่อควบคุมการทำงานของ DS1820

ในการติดต่อกับไอซี DS1820 จะต้องมีคำสั่งส่งให้แก่ DS1820 เพื่อกำหนดรูปแบบการทำงาน คำสั่งที่ใช้มากที่สุดมี 3 คำสั่งดังนี้

1. **คำสั่งไม่ติดต่อกับหน่วยความจำรอมหรือสคิปรอม (Skip ROM)** เนื่องจากในการใช้งาน DS1820 โดยปกติแล้วจะมี DS1820 อยู่บนสายสัญญาณเพียงตัวเดียว จึงไม่จำเป็นต้องใช้ข้อมูลกำหนดแอดเดรส ดังนั้น จึงไม่ต้องติดต่อกับหน่วยความจำรอมเพื่ออ่านข้อมูล ข้อมูลของคำสั่งสคิปรอมที่ต้องส่งให้คือ 0CCH

2. **คำสั่งแปลงอุณหภูมิ (Convert T)** มีค่าเท่ากับ 44H เมื่อส่งคำสั่งนี้ให้ DS1820 จะต้องทำการวนลูปรออย่างน้อย 200 มิลลิวินาที เพื่อให้ DS1820 ได้ใช้เวลาในการแปลงค่าอุณหภูมิเป็นข้อมูลดิจิตอลมาเก็บไว้ในสแครตช์แพด

3. คำสั่งอ่านข้อมูลจากสแครดช์แพด **(Read Scratchpad)** มีค่าเท่ากับ 0EBH เมื่อส่งคำสั่งนี้ไปยัง DS1820 จะทยอยส่งข้อมูลค่าอุณหภูมิออกมาทั้งหมด 9 ไบต์

2.4 โมดูลแสดงผลแบบผลึกเหลว (LCD module)

ในโมดูล LCD จะมีส่วนประกอบหลักอยู่ 3 ส่วน ดังนี้

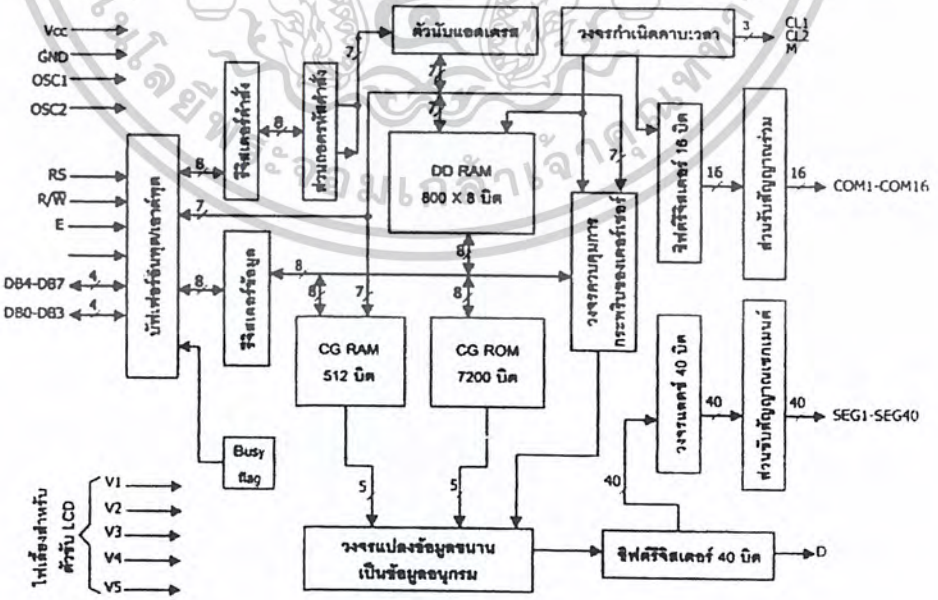
ตัวแสดงผล (display) ภายในมีผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

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

ตัวขับ (driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่ HD44100H และ MSM5259 เป็นต้น

2.4.1 โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ในหนังสือนี้ขอยกตัวอย่างโมดูล LCD แบบอักขระ เพราะสามารถเข้าใจได้ง่าย ให้รูปที่ 2-25 เป็นบล็อกไดอะแกรมภายในของชิปควบคุม LCD เบอร์ HD44780 ซึ่งใช้ในโมดูล LCD แบบอักขระ ประกอบด้วย



รูป 2-25 ไดอะแกรมการทำงานของโมดูล LCD แบบอักขระ

บัฟเฟอร์อินพุตเอาต์พุต เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจิสเตอร์คำสั่ง (Instruction Register:IR) เป็นรีจิสเตอร์ใช้รับข้อมูลคำสั่งจากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

รีจิสเตอร์ข้อมูล (Data Register:DR) เป็นรีจิสเตอร์ใช้รับข้อมูลจากอุปกรณ์ภายนอก เพื่อถ่ายทอดต่อไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผลหรือนำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

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

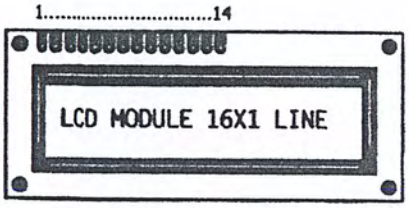
รอมเก็บตัวอักษร (Character Generator ROM : CGROM) เป็นหน่วยความจำรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปแสดงที่ตัวแสดงผลได้ ทีขนาด 7,200 บิต โดยอ่านความถูกต้องด้วยค่าของข้อมูลใน DDRAM

แรมเก็บตัวอักษร (Character Generator RAM : CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บตัวอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรใน CGROM ไม่เพียงพอ มีขนาด 512 บิต การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGRAM เอง

แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายในทราบว่าตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน

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

สำหรับโมดูล LCD ที่ยกมาใช้ในการเรียนรู้ในการทดลอง เป็นขนาด 16 ตัวอักษร 1 บรรทัด เนื่องจากราคาถูก หาง่าย เป็นโมดูล LCD ที่มีโครงสร้างเป็นมาตรฐาน มีผู้ผลิตหลายรายและมีการระบุเบอร์แตกต่างกันออกไปตามผู้ผลิต อาทิ LM020L ของฮิตาชิ, DMC-16117A ของคอปเทร็กซ์ (Optrex) เป็นจ้น แต่อย่างไรก็ตามคอนโทรลเลอร์ที่ใช้คือเบอร์เดียวกันนั่นคือเบอร์ HD44750 ของฮิตาชิ



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

รูปที่ 2-26 รูปร่างและจัดขาโมดูล LCD แบบอักขระ

โมดูล LCD ขนาด 16x1 มีขาต่อใช้งาน 14 ขา มีการจัดขาดังในรูปที่ 2-26 สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

V_{SS} (ขา1) : ต่อกราวด์

V_{DD} (ขา2) : ต่อไฟเลี้ยง +5 โวลต์

V_O (ขา3) : เป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา4) : เป็นขาอินพุตใช้ในการแยกชนิดข้อมูลที่ทำการประมวลผลในขณะนั้นว่า เป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี้เป็น " 0 " ข้อมูลที่ส่งมาจะเป็นคำสั่ง ถ้าขานี้เป็น "1" ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

R/W (ขา5) : เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับโมดูล LCD ถ้าเป็น "0" เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น "1" จะเป็นการอ่านข้อมูล

E (ขา6) : เป็นขาสำหรับรับสัญญาณพัลส์เอ็นเอเบิลโมดูล LCD ให้ทำงาน

D0-D7 (ขา7-14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต

อนึ่งขา RS , R/W และ E จะใช้งานร่วมกัน โดยมีความสัมพันธ์แสดงในตารางที่ 2-3

RS	R/W	E	การทำงาน
0	0	⎍	เขียนคำสั่ง
0	1	⎍⎍	อ่านสถานะของโมดูล LCD
1	0	⎍	เขียนข้อมูล
1	1	⎍⎍	อ่านข้อมูล

ตารางที่ 2-3 แสดงความสัมพันธ์ในการทำงานของขา RS , R/W และ E ของโมดูล LCD แบบอักขระ

2.4.3 คำสั่งควบคุมโมดูล LCD

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

1. คำสั่งเคลียร์ตัวแสดงผล (clear display)

มีข้อมูลคำสั่งเป็น 01H เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่าง หรือ space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมเอ็กซิคิวต์คำสั่งนี้ จะทำการกำหนดแอดเดรสของ DDRAM เป็น 0 เคอร์เซอร์จะกลับไปอยู่ที่ตำแหน่งซ้ายมือสุดของจอแสดงผลแล้วเซตบิต I/D (วิ่งจะกล่าวถึงภายหลัง)ให้เป็น "1"

2. คำสั่ง return home

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น "1" เป็นคำสั่งให้เคอร์เซอร์เคลื่อนที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอไม่เปลี่ยนแปลง นั่นคือ ข้อมูลคำสั่งขิงคำสั่งนี้จะเป็น 02H หรือ 03H ก็ได้

3. คำสั่งเลือกโหมดการป้อนข้อมูล (Entry mode set)

มีรายละเอียดของข้อมูลคำสั่งดังนี้

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

บิต S เป็นบิตที่ใช้ในการกำหนดลักษณะของการแสดงผล เมื่อมีการป้อนข้อมูล ถ้าหากบิต S เป็น "1" เมื่อเกิดข้อมูลใหม่บนจอแสดงผล ตัวเคอร์เซอร์อยู่กับที่ แต่ตัวอักษรข้อมูลเดิมจะถูกดันไปทางซ้าย แต่ถ้าหากบิตนี้เป็น "0" เมื่อเกิดข้อมูลใหม่เคอร์เซอร์จะเลื่อนไปทางขวามือ

บิต I/D เป็นบิตที่ใช้กำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว แอดเดรสของDDRAM เพิ่มขึ้นหรือลดลง หนึ่งแอดเดรส โดยถ้าบิตนี้เป็น "1" แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าเป็น "0" แอดเดรสจะลดลง

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

4. คำสั่งควบคุมตัวแสดงผล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

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

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

บิต B ใช้ควบคุมการกระพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น "1" เคอร์เซอร์จะกระพริบ

ดังนั้นตะมีข้อมูลคำสั่งไก้ตั้งแต่ 08H-0FH (8รูปแบบคำสั่ง) ที่ใช้บ่อยคือ 0CH เป็นการสั่งให้เปิดจอแสดงผล แต่ไม่แสดงเคอร์เซอร์ และ 0FH เป็นการสั่งให้เปิดจอแสดงผล แสดงเคอร์เซอร์ และสั่งให้เคอร์เซอร์กระพริบ

5. คำสั่งควบคุมการเลื่อนเคอร์เซอร์และข้อมูลตัวอักษร

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

การควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผล ขึ้นอยู่กับการกำหนดบิต S/C และ R/L ซึ่งสามารถสรุปได้ดังตาราง 2-4

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

ตาราง 2-4 แสดงการกำหนดบิต S/C และ R/L

6. คำสั่งกำหนดฟังก์ชันการทำงาน

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

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

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล ถ้าเป็น "0" จะแสดงผล 1 บรรทัดถ้าเป็น"1" จะแสดง 2 บรรทัด ในกรณีที่จอแสดงผลสามารถแสดงได้มากกว่า 2 บรรทัด และต้องการให้แสดงผลมากกว่า 2 บรรทัด ก็กำหนดบิต N นี้ให้เป็น "1" จุดที่น่าสังเกตุคือ โมดูล LCD แบบ 16 ตัวอักษร 1 บรรทัด แม้จะมีบรรทัดการแสดงผลเพียง 1 บรรทัด แต่จะต้องกำหนด N ให้เป็น "1" เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่องคือ 00H และ 40H

บิต F ใช้เลือกความละเอียดของตัวอักษรให้การแสดงผล ถ้าบิตนี้เป็น "0" จะเป็นการแสดงผลแบบ5x7 จุด และถ้าเป็น "1" จะแสดงผลแบบ 5x10 จุด

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

7. คำสั่งเลือกแอดเดรสของ CGRAM

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

8. คำสั่งเลือกแอดเดรสของ DDRAM

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

9. คำสั่งอ่านแฟลก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
BF	A	A	A	A	A	A	A
	แอดเดรสไบต์สูง			แอดเดรสไบต์ต่ำ			

เป็นคำสั่งที่ใช้อ่านแฟลก BUSY (BF) โดยแฟลกนี้จะเป็นตัวบอกสถานะของตัวควบคุม LCD ว่าพร้อมจะรับข้อมูลอยู่หรือไม่ถ้าหากบิต BF เป็น "0" แสดงว่าตัวควบคุม LCD พร้อมรับข้อมูลหรือคำสั่ง แต่ถ้าเป็น "1" แสดงว่าขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการทำงานภายในหรือกำลังประมวลผลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง เมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา R/W เป็น "1" ด้วย แต่สัญญาณที่ RS ยังต้องเป็น "0" อยู่เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง นอกจากนี้ยังใช้คำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วย โดยบิต 0-บิต 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

บทที่ 3

การออกแบบ

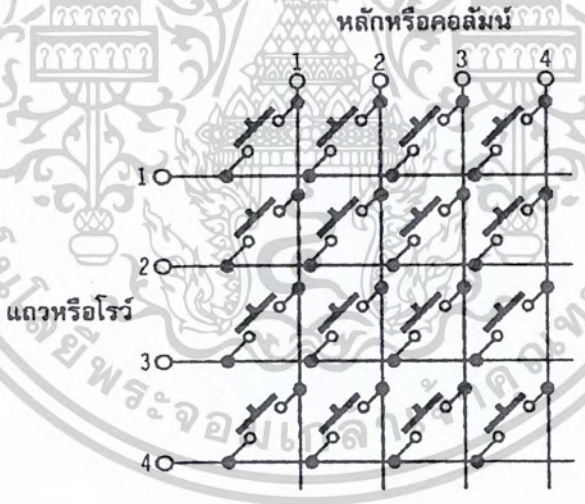
ในการออกแบบได้แยกการออกแบบออกเป็น 2 ส่วน คือ ทางด้านฮาร์ดแวร์และ ซอฟท์แวร์ โดยทางด้านฮาร์ดแวร์จะกล่าวถึง การสร้างบอร์ดกำหนดค่าควบคุมอุณหภูมิ บอร์ดแหล่งจ่ายไฟและบอร์ดไมโครคอนโทรลเลอร์ ในการประมวลผลและแสดงผล ส่วนทางด้านซอฟท์แวร์จะอธิบายเกี่ยวกับคำสั่งที่สำคัญ และแนวทางการเขียนโปรแกรม

3.1 ฮาร์ดแวร์ (Hardware)

ในการสร้างฮาร์ดแวร์เราแบ่งได้ออกเป็นส่วนต่างๆ 3 ส่วนดังนี้

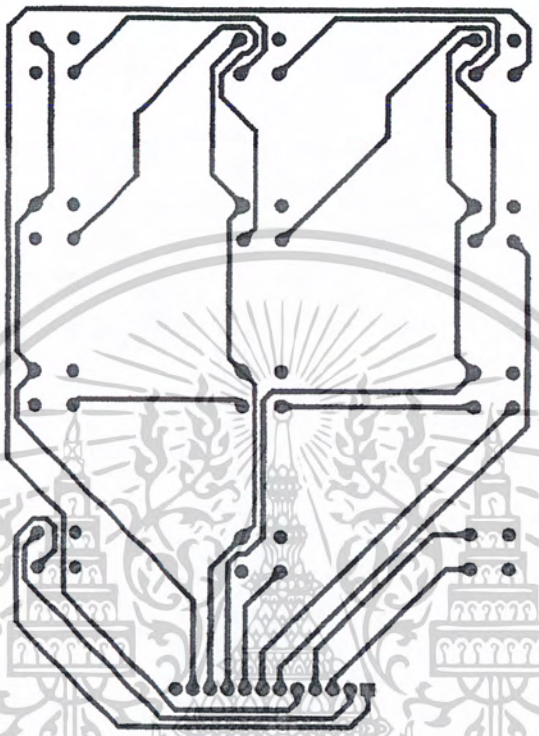
3.1.1 ส่วนกำหนดค่าควบคุมอุณหภูมิ

ในส่วนนี้เราจะใช้สวิตช์ทำการกำหนดค่าอุณหภูมิ โดยนำสวิตช์มาต่อแบบเมตริกซ์หรือคีย์แพดดังแสดงในรูปที่ 3-1



รูปที่ 3-1 วงจรสวิตช์แบบเมตริกซ์หรือคีย์แพด

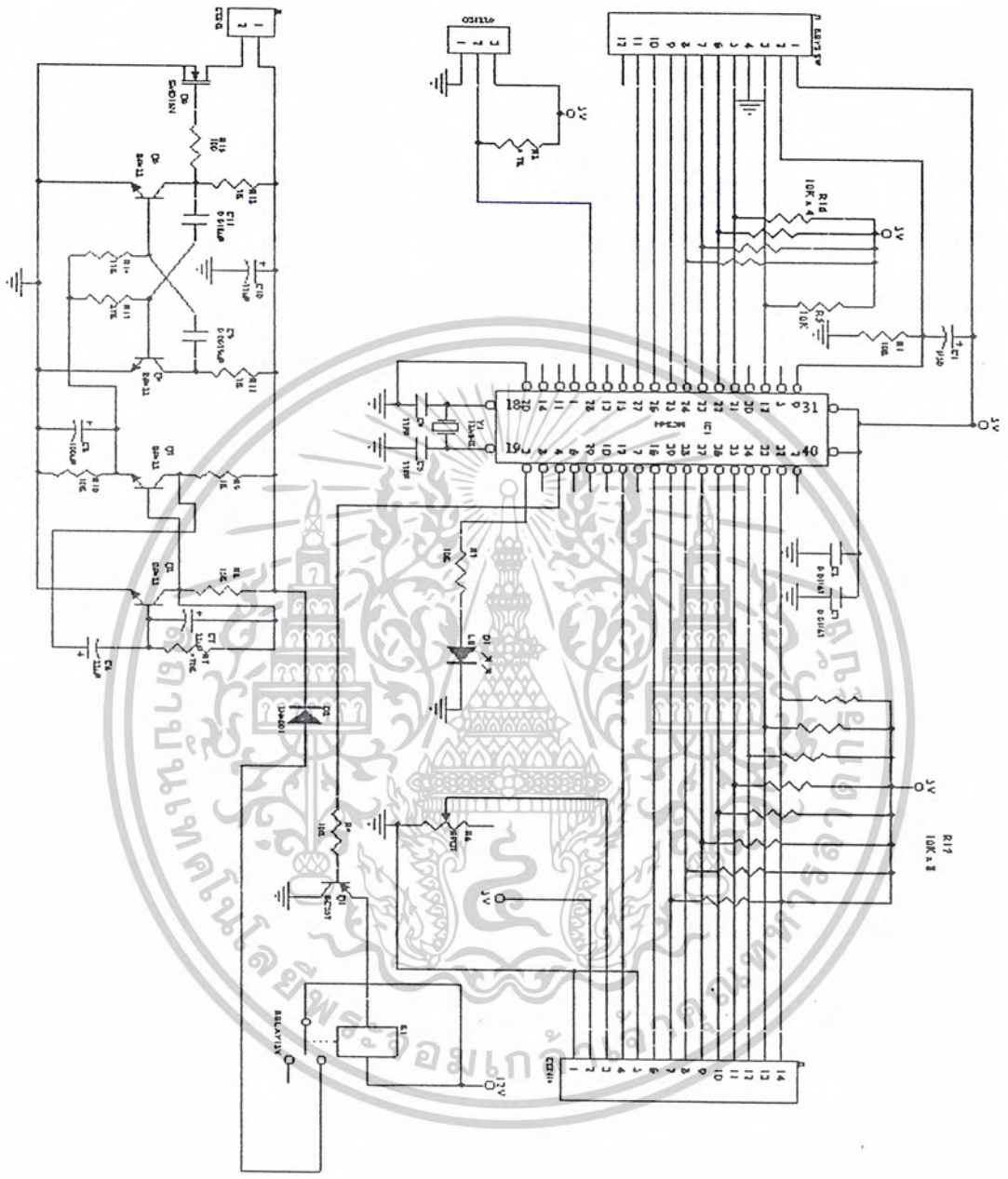
ในการทำบอร์ดส่วนกำหนดค่าควบคุมอุณหภูมิ จะใช้โปรแกรมโพรเทลออกแบบลายวงจร ดังแสดงในรูปที่ 3-2



รูปที่ 3-2 ลายวงจรคีย์แพด

3.1.2 ส่วนประมวลผลและแสดงผล

ในส่วนนี้จะเป็นส่วนหลักของโครงงาน โดยจะมีส่วนที่ทำการแสดงอุณหภูมิที่วัดได้ และส่วนของวงจรเตือน



รูปที่ 3-3 วงจรเครื่องเตือนภัยระบบควบคุมอุณหภูมิ

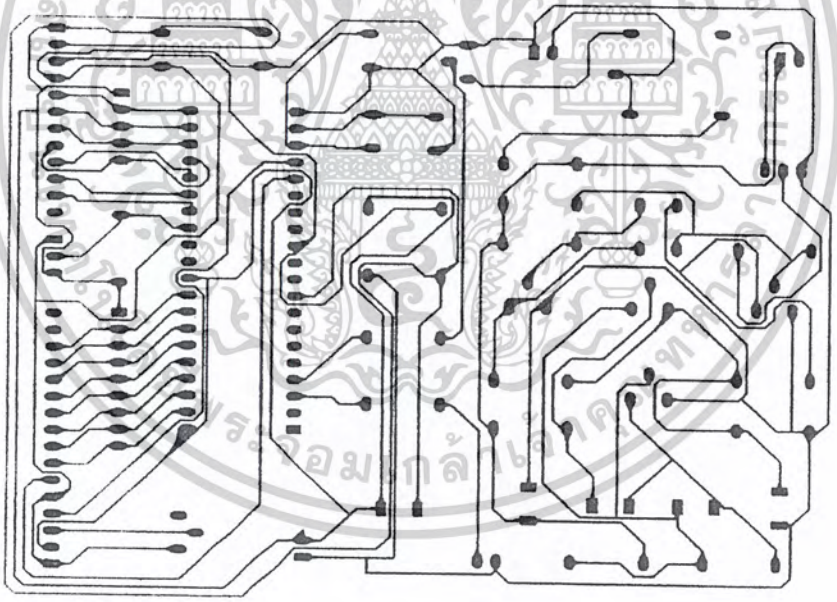
จากรูปที่ 3-3 ซึ่งแสดงวงจรเครื่องเตือนภัยระบบควบคุมอุณหภูมิ จะมีการรับค่าและแสดงผล คือ

พอร์ท 1 เราจะใช้งานพอร์ต 1.2 และพอร์ต 1.3 โดยพอร์ต 1.2 ใช้แสดงว่าเราได้ทำการตั้งค่าอุณหภูมิไว้เรียบร้อยแล้ว โดยจะใช้ LED ในการแสดงผลและพอร์ต 1.3 เป็นพอร์ตที่ใช้ในการควบคุมการเตือน

พอร์ต 2 จะใช้เชื่อมต่อกับคีย์แพดทั้ง 7 เส้น คือ สายของคอลัมน์ 3 สาย C0 – C2 และสายทางโรว์ 4 สาย คือ R0-R3 โดยที่ขาพอร์ต 2.0-2.3 จะต้องต่อตัวต้านทานพูลอัปไว้เพื่อกำหนดแสดงสภาวะ เริ่มต้นไม่มีการกดคีย์ พอร์ต 2.7 จะใช้ในการสื่อสารกับ DS1820 ซึ่งจะเป็นการสื่อสารข้อมูลแบบบัสหนึ่งสาย

พอร์ต 3 คือพอร์ตควบคุมโมดูล LCD ให้ ทำการแสดงค่าอุณหภูมิ

เมื่อใช้โปรแกรมโพรเทลออกแบบ จะได้วงจรดังแสดงในรูป 3-4



รูปที่ 3-4 ลายวงจรเครื่องเตือนภัยระบบควบคุมอุณหภูมิ

3.2 ซอฟ์แวร์ (Software)

เมื่อทำการติดต่อได้แล้วก็จะทำการเขียนโปรแกรมจากเครื่องคอมพิวเตอร์ เพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีชุดคำสั่งที่เหมือนกับไมโครคอนโทรลเลอร์MCS-51 ทุกประการ โดยสามารถแบ่งเป็นกลุ่มได้ดังนี้

3.2.1 กลุ่มคำสั่งการโอนย้ายข้อมูล เป็นกลุ่มคำสั่งที่ใช้ในการโอนย้ายข้อมูล ระหว่างรีจิสเตอร์ในหน่วยความจำข้อมูลด้วยกัน การโอนย้ายข้อมูลกับหน่วยความจำภายนอก การสลับข้อมูลและคำสั่งเก็บหรือเรียกข้อมูลออกจากสแต็ก

3.2.2 กลุ่มคำสั่งทางคณิตศาสตร์ เป็นกลุ่มคำสั่งที่ใช้ในการคำนวณทางคณิตศาสตร์ขึ้นในไมโครคอนโทรลเลอร์ MCS-51 การกระทำทางคณิตศาสตร์ต้องกระทำกับรีจิสเตอร์ A หรือแอคคิวมูเลเตอร์เป็นหลัก และผลลัพธ์ที่ได้จากการคำนวณ จะถูกเก็บไว้ในแอคคิวมูเลเตอร์เสมอ

3.2.3 กลุ่มคำสั่งทางลอจิก เป็นกลุ่มคำสั่งที่ใช้ในการประมวลผลทางตรรกหรือลอจิก ไม่ว่าจะเป็นการแอนด์ ออร์ และเอ็กคลูซีฟ-ออร์ ในไมโครคอนโทรลเลอร์ MCS-51 การกระทำทางลอจิก สามารถกระทำกับรีจิสเตอร์ A หน่วยความจำ และค่าข้อมูลผลลัพธ์ที่ได้จะถูกเก็บไว้ในรีจิสเตอร์หรือหน่วยความจำปลายทาง

3.2.4 กลุ่มคำสั่งจัดการข้อมูลระดับบิต เป็นกลุ่มคำสั่งที่ใช้ในการเปลี่ยนแปลงค่าของข้อมูลในระดับบิต ไม่ว่าจะเป็นการเคลียร์ค่า การเซ็ตค่า และการทำคอมพลีเมนต์หรือการกลับสถานะของข้อมูล

3.2.5 กลุ่มคำสั่งการกระโดด เป็นกลุ่มคำสั่งที่ใช้ในการเปลี่ยนแปลงตำแหน่งแอดเดรสการทำงานของ ซีพียู เนื่องจากในการทำงานบางครั้งซีพียู มีความจำเป็นต้องกระโดดข้ามไปทำงานที่แอดเดรสอื่นที่ไม่ได้อยู่ติดกัน กลุ่มคำสั่งในกลุ่มนี้จึงมีประโยชน์มาก คำสั่งการกระโดดมีหลายรูปแบบ มีทั้งแบบมีเงื่อนไขและไม่มีเงื่อนไข การกระโดดไปทำงานและออกจากโปรแกรมย่อย

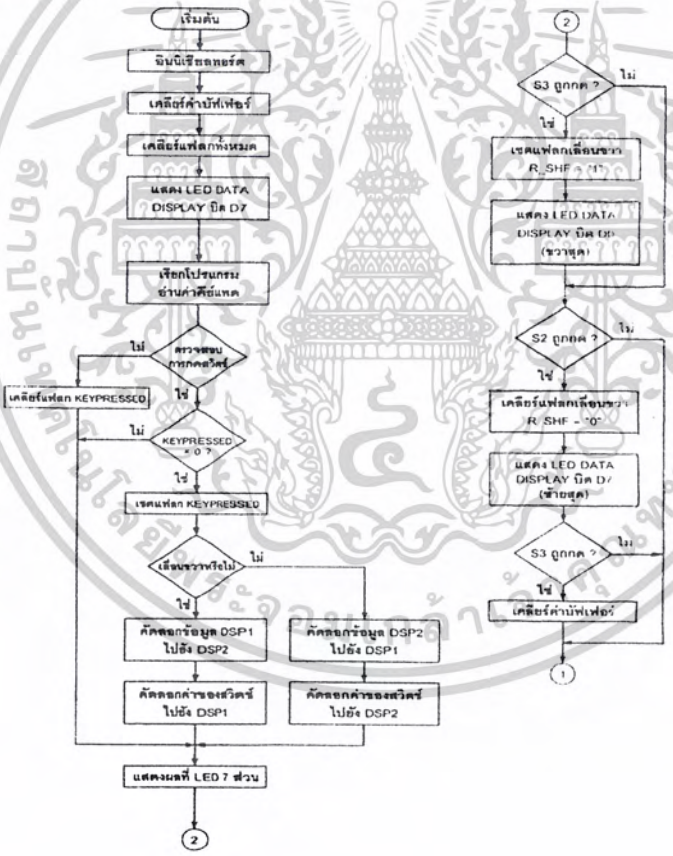
บทที่ 4

การทดลองและผลการทดลอง

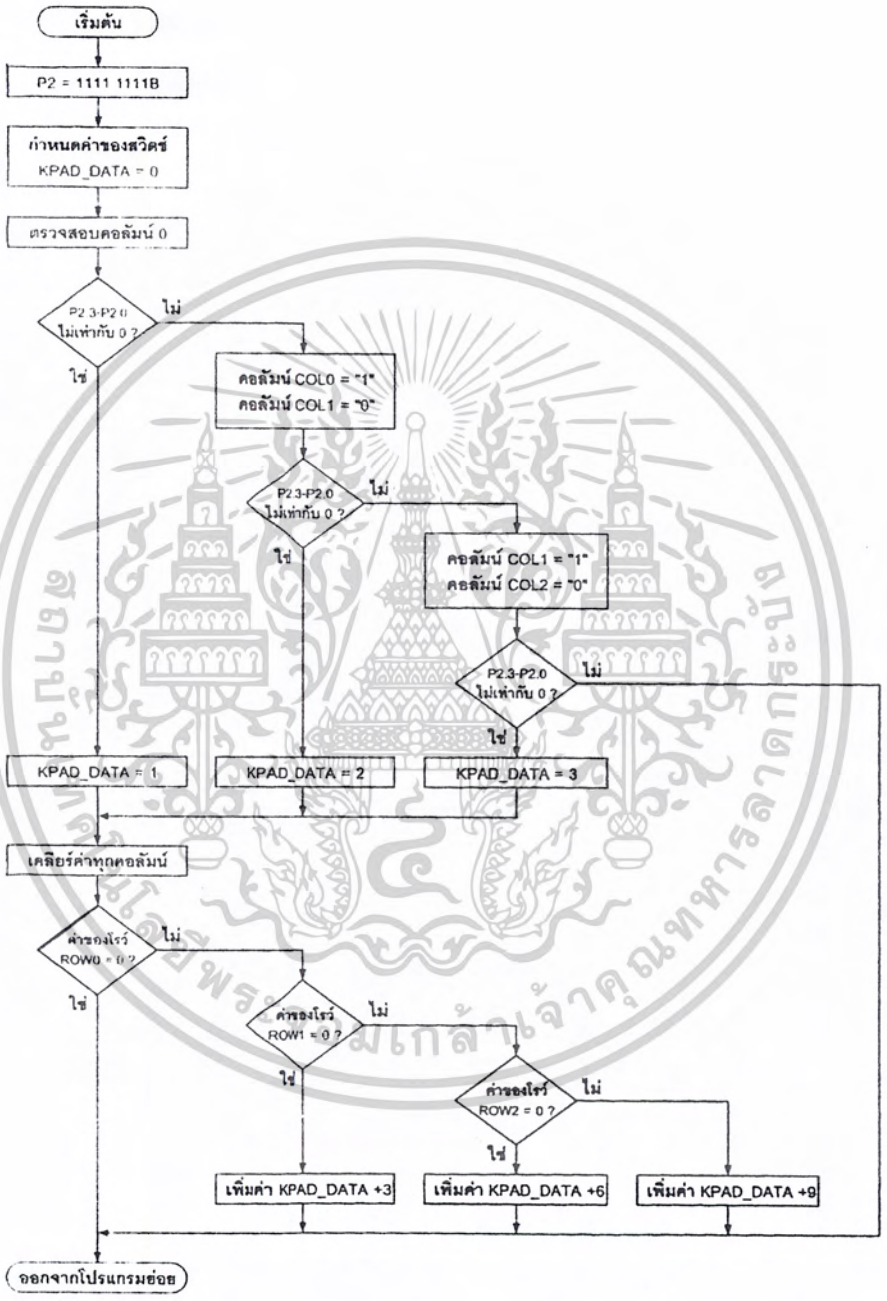
การทดลองเครื่องเตือนภัยระบบควบคุมอุณหภูมิ เราจะทำการทดลอง การเชื่อมต่อกับคีย์แพด การแสดงผลทางจอ LCD และการทำงานของเครื่องเตือนภัยระบบควบคุมอุณหภูมิ

4.1 การเชื่อมต่อไมโครคอนโทรลเลอร์กับคีย์แพด

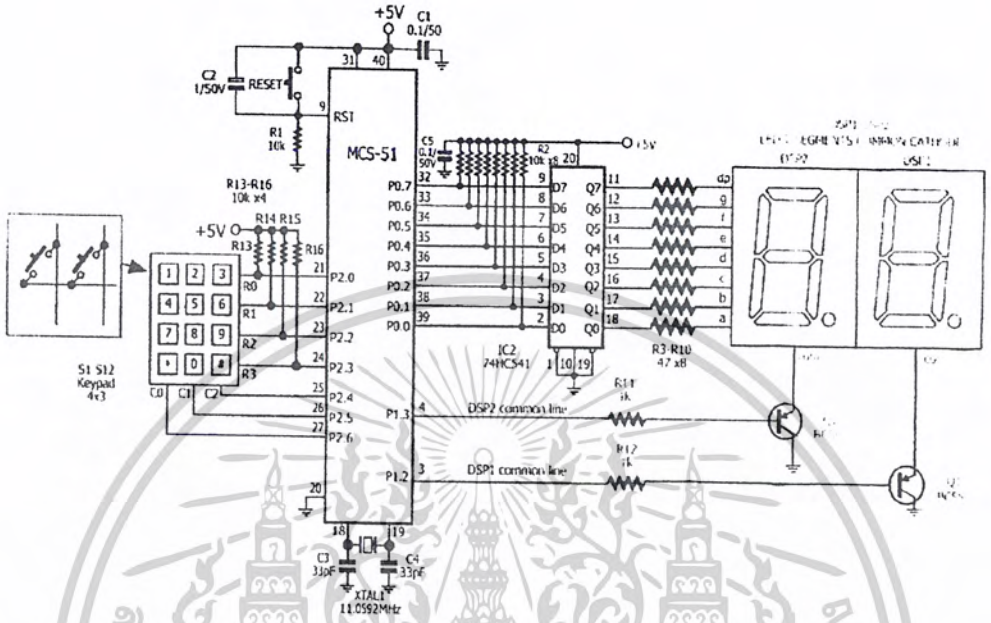
ศึกษาโฟลวชาร์ตตามรูปที่ 4-1 และ 4-2 ร่วมกับวงจรที่ใช้ในการทดลองรูปที่ 4-3 และโปรแกรมที่ 4-1



รูปที่ 4-1 โฟลวชาร์ตโปรแกรมหลักการอ่านค่าคีย์แพดเพื่อนำไปแสดงผลที่LED 7 ส่วน



ที่ 4-2 โฟลวชาร์ตโปรแกรมย่อยในการอ่านค่าคีย์แพด



รูปที่ 4-3 การอ่านค่าคีย์แพดของไมโครคอนโทรลเลอร์MCS-51

```
;-------------------------------------------------------
; Define Port&Pin Name
;-------------------------------------------------------
```

DSP1	BIT	P1.2
DSP2	BIT	P1.3
DRIVER_LE	BIT	P1.4
KPAD_ROW0	BIT	P2.0
KPAD_ROW1	BIT	P2.1
KPAD_ROW2	BIT	P2.2
KPAD_ROW3	BIT	P2.3
KPAD_COL2	BIT	P2.4
KPAD_COL1	BIT	P2.5

```
                KPAD_COL0       BIT         P2.6
;-------------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------------
                FLAG            EQU         02FH
                KEYPRESSED      BIT         FLAG.0
                R_SHF           BIT         FLAG.1
                DSP1_BUFFER     EQU         030H
                DSP2_BUFFER     EQU         031H
                KPAD_DATA       EQU         032H
;-------------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------------
                ORG             0000H
                MOV             P0 , #00000000B
                SETB            DSP1
                SETB            DSP2
                MOV             P2 , #11111111B
MAIN :          MOV             DSP1_BUFFER , #0
                MOV             DSP2_BUFFER , #0
                MOV             FLAG , #0
LOOP:           ACALL           GET_KPAD
                MOV             A , KPAD_DATA
                JZ              NEXT
                JB              KEYPRESSED , SHOW
                SETB            KEYPRESSED
LEFT_SHIFT:     MOV             DSP2_BUFFER , DSP1_BUFFER
                MOV             DSP1_BUFFER , KPAD_DATA
                AJMP            SHOW
NEXT:           CLR             KEYPRESSED
```

```
SHOW:           ACALL   SHOW_DSP
                CLR     R_SHF
                AJMP    LOOP
;-------------------------------------------------------------------
; Keypad Scan key Subroutine
;-------------------------------------------------------------------
GET_KPAD:       MOV     P2 , #0FFH
                MOV     KPAD_DATA , #0
CHK_COL0:       CLR     KPAD_COL0
                MOV     A , P2
                ANL     A , #00FH
                CJNE    A , #00FH , COL0_DETECT
                AJMP    CHK_COL1
COL0_DETECT:    MOV     KPAD_DATA , #01
                AJMP    GET_ROW
CHK_COL1:       SETB    KPAD_COL0
                CLR     KPAD_COL1
                MOV     A , P2
                ANL     A , #00FH
                CJNE    A , #00FH , COL1_DETECT
                AJMP    CHK_COL2
COL1_DETECT:    MOV     KPAD_DATA , #02
                AJMP    GET_ROW
CHK_COL2:       SETB    KPAD_COL1
                CLR     KPAD_COL2
                MOV     A , P2
                ANL     A , #00FH
                CJNE    A , #00FH , COL2_DETECT
                RET
COL2_DETECT:    MOV     KPAD_DATA , #03
```

```
GET_ROW:        CLR     KPAD_COL0
                CLR     KPAD_COL1
                CLR     KPAD_COL2
                JB      KPAD_ROW0 , CHK_ROW1
                RET
CHK_ROW1:       JB      KPAD_ROW1 , CHK_ROW2
                MOV     A , KPAD_DATA
                ADD     A , #3
                MOV     KPAD_DATA , A
                RET
CHK_ROW2:       JB      KPAD_ROW2 , CHK_ROW3
                MOV     A , KPAD_DATA
                ADD     A , #6
                MOV     KPAD_DATA , A
                RET
CHK_ROW3:       MOV     A , KPAD_DATA
                ADD     A , #9
                MOV     KPAD_DATA , A
                RET
;-------------------------------------------------------------------
; Show DSP Subroutine
;-------------------------------------------------------------------
SHOW_DSP:       MOV     R4 , #5
SCAN_DSP_LOOP:  MOV     A , DSP1_BUFFER
                MOV     DPTR , #DSP_BLANK
                MOVC    A , @A+DPTR
                MOV     P0 , A
                CLR     DSP1
                ACALL   DELAY_1ms
                SETB    DSP1
```

```
                MOV     A , DSP2_BUFFER
                MOV     DPTR , #DSP_BLANK
                MOVC    A , @A+DPTR
                MOV     P0 , A
                CLR     DSP2
                ACALL   DELAY_1ms
                SETB    DSP2
                DJNZ    R4 , SCAN_DSP_LOOP
                RET
;---------------------------------------------------------
; Dummy Delay time 1m,10ms,1s
;---------------------------------------------------------
DELAY_1ms:      MOV     R6 , #0E6H
DELAY_1ms_1:    NOP
                NOP
                DJNZ    R6 , DELAY_1ms_1
                RET
DELAY_10ms:     MOV     R7 , #010
DELAY_10ms_1:   MOV     R6 , #0E6H
DELAY_10ms_2:   NOP
                NOP
                DJNZ    R6 , DELAY_10ms_2
                DJNZ    R7 , DELAY_10ms_1
                RET
DELAY_1s:       MOV     R5 , #100
DELAY_1s_1:     ACALL   DELAY_10ms
                DJNZ    R5 , DELAY_1s_1
                RET
```

```
;-------------------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;-------------------------------------------------------------------------
;               Segment     .GFEDCBA
DSP_BLANK:                  DB          10000000B
DSP_NUM1:                   DB          00000110B
DSP_NUM2:                   DB          01011011B
DSP_NUM3:                   DB          01001111B
DSP_NUM4:                   DB          01100110B
DSP_NUM5:                   DB          01101101B
DSP_NUM6:                   DB          01111101B
DSP_NUM7:                   DB          00000111B
DSP_NUM8:                   DB          01111111B
DSP_NUM9:                   DB          01101111B
DSP_STAR:                   DB          01110110B
DSP_NUM0:                   DB          00111111B
DSP_HASH:                   DB          01100011B
```

โปรแกรมที่ 4-1 โปรแกรมทดลองการอ่านค่าจากคีย์แพดแล้วแสดงผลที่ LED ตัวเลขที่ LED ตัวเลข 7 ส่วน

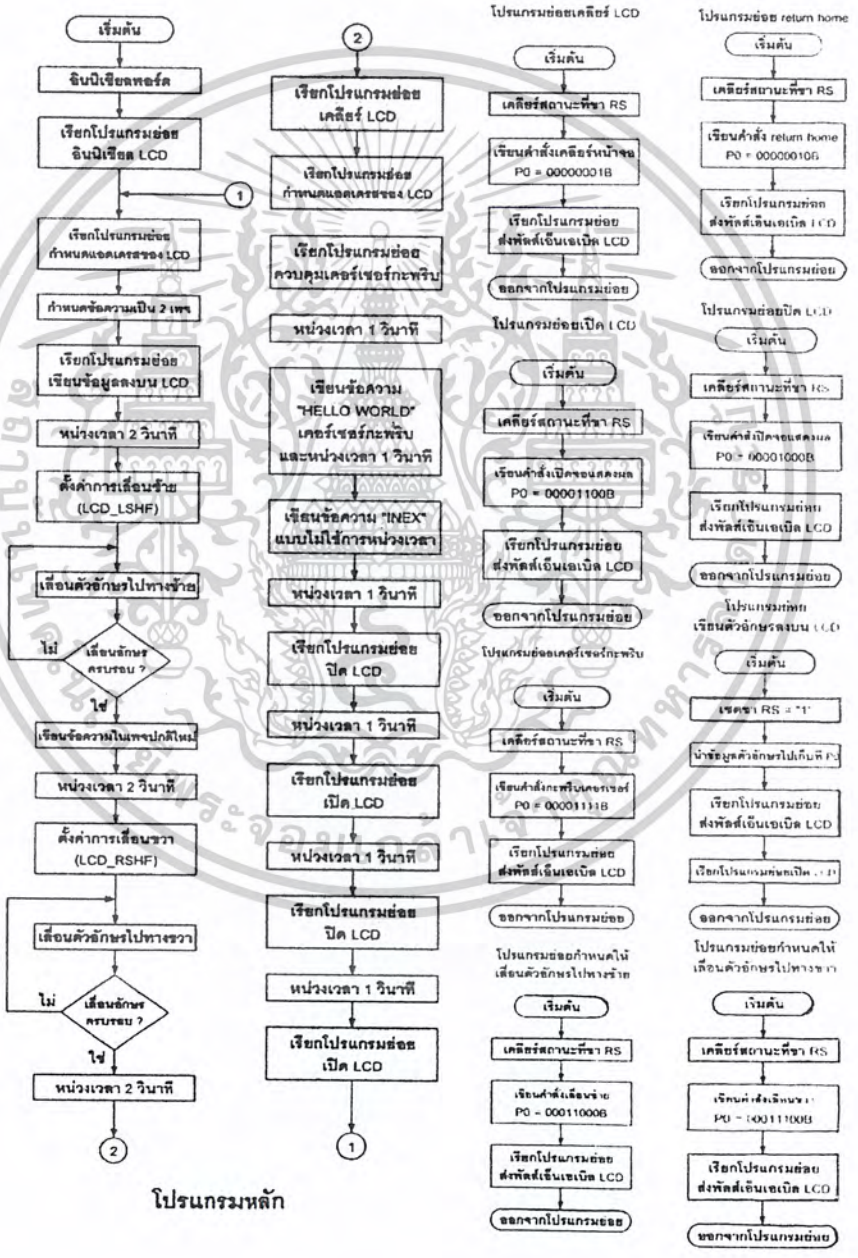
ผลการทดลอง

เมื่อรันโปรแกรมนี้ จุด dp ของ LED ตัวเลข 7 ส่วน ทั้ง 2 หลักจะติดสว่าง และเมื่อเริ่มกดสวิตช์ LED ตัวเลข 7 ส่วนหลักขวาสุดจะแสดงตัวเลขตามค่าของสวิตช์ที่กด ซึ่งก็คือเลข 0 และเมื่อกดสวิตช์ 1 ตัวเลข 0 จะมาแสดงยังซ้ายมือ จะเป็นเช่นนี้ไปเรื่อยๆ

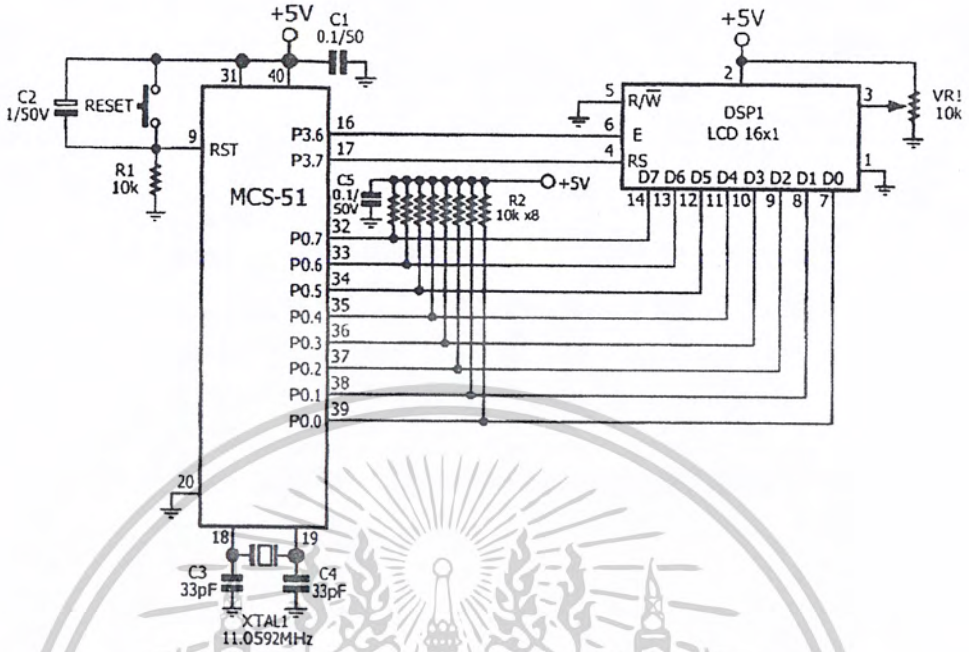
4.2 การเชื่อมต่อโมดูล LCD

การทดลองเพื่อแสดงข้อความบนโมดูล LCD 16 ตัวอักษร 1 บรรทัด

ศึกษาโฟลวชาร์ตในรูปที่ 4-4 กับวงจรที่ใช้ในการทดลองตามรูปที่ 4 -5 และโปรแกรมที่ 4-2



รูปที่ 4-4 โฟลวชาร์ตแสดงข้อมูลบน LCD ขนาด 16 ตัวอักษร 1 บรรทัด



รูปที่ 4-5 วงจรการแสดงข้อความบนโมดูล LCD 16 ตัวอักษร 1 บรรทัด

```
; Define Port&Pin Name
;-------------------------------------------------------------
LCD_EN                  BIT             P3.6
LCD_RS                  BIT             P3.7
;-------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------
LCD_ADDR                EQU             030H
LCD_DATA                EQU             031H
;-------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------
                        ORG             0000H
                        MOV             P0 , #00000000B
```

```
                CLR     LCD_EN
                CLR     LCD_RS
MAIN:           ACALL   INIT_LCD
LOOP:           MOV     LCD_ADDR , #000H
                ACALL   SET_ADDR_LCD
                MOV     DPTR , #TITLE_1
                ACALL   WRLINE_LCD
                MOV     DPTR , #TITLE_3
                ACALL   WRLINE_LCD
                MOV     LCD_ADDR , #040H
                ACALL   SET_ADDR_LCD
                MOV     DPTR , #TITLE_2
                ACALL   WRLINE_LCD
                MOV     DPTR , #TITLE_4
                ACALL   WRLINE_LCD
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                MOV     R4 , #8
LOOP_LCD_L_SHF: ACALL   LCD_LSHF
                ACALL   DELAY_100ms
                ACALL   DELAY_100ms
                DJNZ    R4 , LOOP_LCD_L_SHF
                MOV     LCD_ADDR , #000H
                ACALL   SET_ADDR_LCD
                MOV     DPTR , #TITLE_5
                ACALL   WRLINE_LCD
                MOV     LCD_ADDR , #040H
                ACALL   SET_ADDR_LCD
                MOV     DPTR , #TITLE_6
                ACALL   WRLINE_LCD
```

```
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                MOV     R4 , #8
LOOP_LCD_R_SHF :ACALL   LCD_RSHF
                ACALL   DELAY_100ms
                ACALL   DELAY_100ms
                DJNZ    R4 , LOOP_LCD_R_SHF
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                ACALL   LCD_CLR
                MOV     LCD_ADDR , #000H
                ACALL   SET_ADDR_LCD
                ACALL   LCD_BLINK
                ACALL   DELAY_1s
                MOV     LCD_DATA , # 'H'
                ACALL   WRCHAR_LCD
                ACALL   LCD_BLINK
                ACALL   DELAY_1s
                MOV     LCD_DATA , # 'e'
                ACALL   WRCHAR_LCD
                ACALL   LCD_BLINK
                ACALL   DELAY_1s
                MOV     LCD_DATA , # 'l'
                ACALL   WRCHAR_LCD
                ACALL   LCD_BLINK
                ACALL   DELAY_1s
                MOV     LCD_DATA , # 'l'
                ACALL   WRCHAR_LCD
                ACALL   LCD_BLINK
                ACALL   DELAY_1s
```

```
MOV     LCD_DATA , # 'o'
ACALL   WRCHAR_LCD
ACALL   LCD_BLINK
ACALL   DELAY_1s
MOV     LCD_DATA , # ' '
ACALL   WRCHAR_LCD
MOV     LCD_DATA , # 'W'
ACALL   WRCHAR_LCD
ACALL   LCD_BLINK
ACALL   DELAY_1s
MOV     LCD_DATA , # 'o'
ACALL   WRCHAR_LCD
MOV     LCD_ADDR , #040H
ACALL   SET_ADDR_LCD
ACALL   LCD_BLINK
ACALL   DELAY_1s
MOV     LCD_DATA , # 'r'
ACALL   WRCHAR_LCD
ACALL   LCD_BLINK
ACALL   DELAY_1s
MOV     LCD_DATA , # 'l'
ACALL   WRCHAR_LCD
ACALL   LCD_BLINK
ACALL   DELAY_1s
MOV     LCD_DATA , # 'd'
ACALL   WRCHAR_LCD
ACALL   LCD_BLINK
ACALL   DELAY_1s
MOV     LCD_DATA , # ' '
ACALL   WRCHAR_LCD
```

```
                MOV     LCD_DATA , # 'I'
                ACALL   WRCHAR_LCD
                MOV     LCD_DATA , # 'N'
                ACALL   WRCHAR_LCD
                MOV     LCD_DATA , # 'E'
                ACALL   WRCHAR_LCD
                MOV     LCD_DATA , # 'X'
                ACALL   WRCHAR_LCD
                ACALL   DELAY_1s
                ACALL   LCD_OFF
                ACALL   DELAY_1s
                ACALL   LCD_ON
                ACALL   DELAY_1s
                ACALL   LCD_OFF
                ACALL   DELAY_1s
                ACALL   LCD_ON
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                AJMP    LOOP
;---------------------------------------------------------------
; LCD Initialize
;---------------------------------------------------------------
INIT_LCD:       ACALL   DELAY_100ms
                CLR     LCD_RS
                MOV     P0 , #00111000B
                ACALL   LCD_CLK
                ACALL   DELAY_10ms
                MOV     P0 , #00111000B
                ACALL   LCD_CLK
                ACALL   LCD_OFF
```

```
                ACALL   LCD_CLR
                MOV     P0 , #00000110B
                ACALL   LCD_CLK
                ACALL   LCD_HOME
;---------------------------------------------------------------
; LCD Clear Display
;---------------------------------------------------------------
LCD_CLR:        CLR     LCD_RS
                MOV     P0 , #00000001B
                ACALL   LCD_CLK
                RET
;---------------------------------------------------------------
; LCD Return Home
;---------------------------------------------------------------
LCD_HOME:       CLR     LCD_RS
                MOV     P0,#00000010B
                ACALL   LCD_CLK
                RET
;---------------------------------------------------------------
; LCD Display Off
;---------------------------------------------------------------
LCD_OFF:        CLR     LCD_RS
                MOV     P0 , #00001000B
                ACALL   LCD_CLK
                RET
;---------------------------------------------------------------
; LCD Clk
;---------------------------------------------------------------
LCD_CLK:        SETB    LCD_EN
                ACALL   LCD_DELAY
```

```
                    CLR     LCD_EN
                    ACALL   LCD_DELAY
                    RET
;-------------------------------------------------------------
; LCD Display On
;-------------------------------------------------------------
LCD_ON:             CLR     LCD_RS
                    MOV     P0 , #00001100B
                    ACALL   LCD_CLK
                    RET
;-------------------------------------------------------------
; LCD Cursor On
;-------------------------------------------------------------
LCD_BLINK:          CLR     LCD_RS
                    MOV     P0 , #00001111B
                    ACALL   LCD_CLK
                    RET
;-------------------------------------------------------------
; LCD Left Shift Display
;-------------------------------------------------------------
LCD_LSHF:           CLR     LCD_RS
                    MOV     P0 , #00011000B
                    ACALL   LCD_CLK
                    RET
;-------------------------------------------------------------
; LCD Right Shift Display
;-------------------------------------------------------------
LCD_RSHF:           CLR     LCD_RS
                    MOV     P0 , #00011100B
```

```
                    ACALL       LCD_CLK
                    RET
;-------------------------------------------------------------------
; Set LCD Address
; I/P:          LCD_ADDR
;-------------------------------------------------------------------
SET_ADDR_LCD:       CLR         LCD_RS
                    MOV         A,LCD_ADDR
                    SETB        ACC.7
                    MOV         P0 , A
                    ACALL       LCD_CLK
                    RET
;-------------------------------------------------------------------
; Write Character to show LCD
; I/P:          LCD_DATA
;-------------------------------------------------------------------
WRCHAR_LCD:         SETB        LCD_RS
                    MOV         P0 , LCD_DATA
                    ACALL       LCD_CLK
                    ACALL       LCD_ON
                    RET
;-------------------------------------------------------------------
; Write Line of 8 Character from ROM
; I/P:          DPTR : Locate ROM Address
;-------------------------------------------------------------------
WRLINE_LCD:         MOV         R0 , #0
WRLINE_LCD_1:       SETB        LCD_RS
                    CLR         A
                    MOVC        A , @A+DPTR
                    MOV         P0 , A
```

```
                    ACALL     LCD_CLK
                    INC       DPTR
                    INC       R0
                    CJNE      R0 , #8 , WRLINE_LCD_1
                    ACALL     LCD_ON
                    RET
;-------------------------------------------------------------------
; Dummy Delay time LCD_DELAY, 10m, 100m, 1s
;-------------------------------------------------------------------
LCD_DELAY:          MOV       R7 , #002
LCD_DELAY_1:        MOV       R6 , #0E6H
LCD_DELAY_2:        NOP
                    NOP
                    DJNZ      R6 , LCD_DELAY_2
                    DJNZ      R7 , LCD_DELAY_1
                    RET
DELAY_10ms:         MOV       R7 , #010
DELAY_10ms_1:       MOV       R6 , #0E6H
DELAY_10ms_2:       NOP
                    NOP
                    DJNZ      R6 , DELAY_10ms_2
                    DJNZ      R7 , DELAY_10ms_1
                    RET
DELAY_100ms:        MOV       R7 , #100
DELAY_100ms_1:      MOV       R6 , #0E6H
DELAY_100ms_2:      NOP
                    NOP
                    DJNZ      R6 , DELAY_100ms_2
                    DJNZ      R7 , DELAY_100ms_1
                    RET
```

```
DELAY_1s:       MOV     R5 , #100
DELAY_1s_1:     ACALL   DELAY_10ms
                DJNZ    R5 , DELAY_1s_1
                RET
;---------------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;---------------------------------------------------------------------
;                       01234567
TITLE_1:        DB      'LCD Modu'
TITLE_2:        DB      'le Show.'
TITLE_3:        DB      'Test Run'
TITLE_4:        DB      'ning ...'
TITLE_5:        DB      'Write Ch'
TITLE_6:        DB      'aracter '
```

โปรแกรมที่ 4-2 โปรแกรมทดลองการแสดงข้อความบนโมดูล LCD 16 ตัวอักษร 1 บรรทัด

ผลการทดลอง

เมื่อรันโปรแกรมโมดูล LCD จะเริ่มต้นด้วยการแสดงข้อความ LCD Module Show. ตามด้วยการเลื่อนข้อความไปทางซ้ายและขวา ต่อด้วยการแสดงข้อความทีละตัวอักษร และปิดท้ายด้วยการทำให้จอแสดงผลเกิดการกะพริบ และวนกลับไปเริ่มต้นแสดงผลใหม่

4.3 การทดลองการทำงานของเครื่องเตือนภัยระบบควบคุมอุณหภูมิ

4.3.1 เปิดสวิตซ์ เริ่มต้นการทำงาน

4.3.2 สังเกตที่จอ LCD โดยจอ LCD จะแสดงอุณหภูมิที่วัดได้ขณะนั้น

4.3.3 กดปุ่ม * เพื่อทำการตั้งค่าควบคุมอุณหภูมิ แล้วหน้าจอจะกลับเข้าสู่ การแสดงอุณหภูมิปัจจุบัน

4.3.4 นำหัวแร้งเข้ามาใกล้ๆไอซีตรวจจับอุณหภูมิ DS1820 โดยสังเกตุที่จอโดยอุณหภูมิจะมีค่าเพิ่มขึ้นเรื่อยๆ

4.3.5 สังเกตการทำงานของเครื่องเตือนภัยระบบควบคุมอุณหภูมิ

4.3.6 นำหัวแร้งออกห่างจากไอซีตรวจจับอุณหภูมิ สังเกตุการเปลี่ยนแปลง

ผลการทดลอง

ในการทดลองเราวัดค่าอุณหภูมิได้ 32°C โดยเราจะทำการตั้งค่าอุณหภูมิที่ต้องการเตือนที่ 36°C เมื่อนำหัวแร้งมาไว้ใกล้ไอซีตรวจจับอุณหภูมิ DS1820 จนอุณหภูมิเพิ่มขึ้นจนถึง 36°C เครื่องจะส่งเสียงเตือน โดยเมื่ออุณหภูมิสูงกว่าค่าที่เราตั้งค่าไว้ เครื่องก็ยังส่งเสียงเตือนอยู่ และเมื่อเรานำหัวแร้งออกห่างจากไอซีตรวจจับอุณหภูมิ DS1820 อุณหภูมิที่วัดได้จะมีค่าลดลง โดยอุณหภูมิจะลดลงจนถึงค่าปกติ โดยที่เครื่องก็ยังทำการเตือนอยู่

บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

เราสามารถควบคุมการทำงานของเครื่องเตือนภัยระบบควบคุมอุณหภูมิ โดยทำการโหลดโปรแกรมซึ่งจะทำการเขียนบนคอมพิวเตอร์ โดยผ่านทางพอร์ตขนาน โดยโปรแกรมที่ควบคุมการทำงานของเครื่องเตือนภัยระบบควบคุมอุณหภูมิ ประกอบไปด้วยโปรแกรมส่วนต่างๆดังนี้

- โปรมแกรมการอ่านค่าจากคีย์แพด
- โปรแกรมการแสดงอุณหภูมิบนโมดูล LCD 16 ตัวอักษร 1 บรรทัด
- โปรแกรมการใช้งานไอซีตรวจจับอุณหภูมิ DS1820

ปัญหาในการทดลองและแนวทางการแก้ไข

ปัญหาที่พบในการทดลองการทำงานของเครื่องเตือนภัยระบบควบคุมอุณหภูมิ ประกอบไปด้วย

1. ในการวัดอุณหภูมิด้วยไอซี DS1820 เมื่อประกอบวงจรลงบนบอร์ด แล้วทำการวัดอุณหภูมิเปรียบเทียบกับกับ ดิจิตอลเทอร์โมมิเตอร์แล้วอุณหภูมิทำการวัดได้จะสูงกว่าอุณหภูมิจริงเล็กน้อย เนื่องจากตัวตรวจจับอุณหภูมิ DS1820 จะได้รับอุณหภูมิจากการทำงานของ Tranformer เราจะต้องทำการแยกบอร์ดของไอซีตรวจจับอุณหภมิ DS1820 ออกมาต่างหาก

2. ในการตั้งค่าอุณหภูมิควบคุม บางครั้งเมื่อทำการกดสวิตช์ตั้งค่า จะเกิดปัญหาคือการกดสวิตช์ซ้อนกันสองครั้ง เพราะฉะนั้นจะต้องทำการตรวจสอบให้แน่ใจก่อนว่า อุณหภูมิที่เราทำการตั้งค่านั้นถูกต้อง

แนวทางในการพัฒนา

เครื่องเตือนภัยระบบควบคุมอุณหภูมินี้ได้ออกแบบโดยใช้รีเลย์เป็นตัวตัดต่อเพื่อให้วงจรเตือนทำงาน เราสามารถนำประยุกต์ใช้งานได้หลากหลาย ใช้ควบคุมอุปกรณ์ไฟฟ้า เช่น แอร์ พัดลม และอื่นๆ ที่มีอุณหภูมิเข้ามาเกี่ยวข้อง

ภาคผนวก

ภาคผนวก ก

```
;-------------------------------------------------------------------------
; Define Port&Pin Name
;-------------------------------------------------------------------------
        ONEWIRE         BIT         P2.7
        KPAD_ROW0       BIT         P2.0
        KPAD_ROW1       BIT         P2.1
        KPAD_ROW2       BIT         P2.2
        KPAD_ROW3       BIT         P2.3
        KPAD_COL2       BIT         P2.4
        KPAD_COL1       BIT         P2.5
        KPAD_COL0       BIT         P2.6
        LIGHT           BIT         P1.2
        SOUND           BIT         P1.3
        SET_TEMP        BIT         P3.2
        LCD_EN          BIT         P3.6
        LCD_RS          BIT         P3.7
;-------------------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------------------
        FLAG            EQU         02FH
        BUSY            BIT         FLAG.0
        LCD_ADDR        EQU         030H
        LCD_DATA        EQU         031H
        LCD_PTR         EQU         032H
        KPAD_DATA       EQU         033H
        BUFFER          EQU         034H
        TEMP_1000       EQU         035H
        TEMP_100        EQU         036H
        TEMP_10         EQU         037H
        TEMP_1          EQU         038H
        TUBE_1000       EQU         039H
```

```
TUBE_100        EQU     040H
TUBE_10         EQU     041H
TUBE_1          EQU     042H
ONEWIRE_DATA    EQU     043H
TEMP            EQU     044H
;-------------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------------
                ORG     0000H
                AJMP    MAIN1
                ORG     0003H
                AJMP    TUBE
                ORG     0010H
MAIN1:          MOV     P0,#00000000B
                SETB    ONEWIRE
                MOV     IE,#081H
                MOV     TUBE_1000,#30H
                MOV     TUBE_100,#00H
                MOV     TUBE_10,#00H
                MOV     TUBE_1,#30H
MAIN:           ACALL   INIT_LCD
                CLR     LIGHT
                MOV     LCD_ADDR,#000H
                ACALL   SET_ADDR_LCD
                MOV     DPTR,#TITLE_1
                ACALL   WRLINE_LCD
                ACALL   DELAY_1s
                ACALL   DELAY_1s
                MOV     LCD_ADDR,#000H
                ACALL   SET_ADDR_LCD
```

```
                MOV     DPTR,#SCR_TEMP
                ACALL   WRLINE_LCD
LOOP:           ACALL   DELAY_1s
                ACALL   MAI
                ACALL   DS1820_RST
                ACALL   DS1820_PRES
                MOV     ONEWIRE_DATA,#0CCH
                ACALL   DS1820_WR
                MOV     ONEWIRE_DATA,#044H
                ACALL   DS1820_WR
                SETB    BUSY
PRES_CHK_LOOP:  ACALL   DS1820_RST
                ACALL   DS1820_PRES
                JB      BUSY,PRES_CHK_LOOP
                NOP
                NOP
                NOP
                NOP
                ACALL   DS1820_RST
                ACALL   DS1820_PRES
                MOV     ONEWIRE_DATA,#0CCH
                ACALL   DS1820_WR
                MOV     ONEWIRE_DATA,#0BEH
                ACALL   DS1820_WR
                ACALL   DS1820_RD
                MOV     TEMP,ONEWIRE_DATA
                ACALL   DS1820_RST
                ACALL   DS1820_PRES
                MOV     LCD_ADDR,#040H
                ACALL   SET_ADDR_LCD
                MOV     A,TEMP
                CLR     C
```

```
                RRC     A
                MOV     LCD_DATA,A
                ACALL   HEX2LCD
                MOV     LCD_ADDR,#044H
                ACALL   SET_ADDR_LCD
                MOV     A,TEMP
                JNB     ACC.0,WRITE_0C
                MOV     LCD_DATA,#'5'
                AJMP    WRITE_NEXT
WRITE_0C:       MOV     LCD_DATA,#'0'
WRITE_NEXT:     MOV     TEMP_1,LCD_DATA
                ACALL   WRCHAR_LCD
                ACALL   DELAY_1ms
                AJMP    LOOP
MAI:            MOV     A,TEMP_1000
                MOV     R0,TUBE_1000
                SUBB    A,R0
                CJNE    A,#00,MAI_1
                MOV     A,TEMP_100
                MOV     R0,TUBE_100
                SUBB    A,R0
                CJNE    A,#00,MAI_1
                MOV     A,TEMP_10
                MOV     R0,TUBE_10
                SUBB    A,R0
                CJNE    A,#00,MAI_1
                MOV     A,TEMP_1
                MOV     R0,TUBE_1
                SUBB    A,R0
                CJNE    A,#00,MAI_1
                CLR     SOUND
                ACALL   DELAY_1s
```

```
MAI_1:      RET
TUBE:       MOV     LCD_ADDR,#000H
            ACALL   SET_ADDR_LCD
            MOV     DPTR,#NEW_TEMP
            ACALL   WRLINE_LCD
            MOV     LCD_ADDR,#040H
            ACALL   SET_ADDR_LCD
MAI007:     POP     A
            CLR     A
            ACALL   WAIT_KEYPRESSED
            MOV     BUFFER,KPAD_DATA
            MOV     A,KPAD_DATA
            ADD     A,#30H
            CJNE    A,#'0',MAI007_1
            MOV     A,#'0'
            AJMP    MAI008
MAI007_1:   CJNE    A,#'1',MAI007
            MOV     A,#'1'
MAI008:     MOV     TUBE_1000,A
            MOV     LCD_DATA,TUBE_100
            ACALL   WRCHAR_LCD
            ACALL   LCD_BLINK
            ACALL   WAIT_KEY

            ACALL   WAIT_KEYPRESSED
            MOV     BUFFER+1,KPAD_DATA
            MOV     A,KPAD_DATA
            ADD     A,#30H
            MOV     TUBE_100,A
            MOV     LCD_DATA,TUBE_100
            ACALL   WRCHAR_LCD
            ACALL   LCD_BLINK
```

```
                ACALL   WAIT_KEY
                ACALL   WAIT_KEYPRESSED
                MOV     BUFFER+2,KPAD_DATA
                MOV     A,KPAD_DATA
                ADD     A,#30H
                MOV     TUBE_10,A
                MOV     LCD_DATA,TUBE_10
                ACALL   WRCHAR_LCD
                ACALL   LCD_BLINK
                ACALL   WAIT_KEY
LOOP_05:        MOV     LCD_ADDR,#044H
                ACALL   SET_ADDR_LCD
                ACALL   WAIT_KEYPRESSED
                MOV     BUFFER+3,KPAD_DATA
                MOV     A,KPAD_DATA
                ADD     A,#30H
                CJNE    A,#'0',LOOP_05_1
                AJMP    OPEN
LOOP_05_1       CJNE    A,#'5',LOOP_05
OPEN:           MOV     TUBE_1,A
                MOV     LCD_DATA,TUBE_1
                ACALL   WRCHAR_LCD
                ACALL   WAIT_KEY
                MOV     LCD_ADDR,#000H
                ACALL   SET_ADDR_LCD
                MOV     DPTR,#SCR_TEMP
                ACALL   WRLINE_LCD
                SETB    LIGHT
                PUSH    A
                RETI
```

```
;-------------------------------------------------------------------
; HEX Code to show LCD
; I/P:          LCD_DATA
;-------------------------------------------------------------------
HEX2LCD:        PUSH    ACC
                MOV     A,LCD_DATA
                MOV     B,#100
                DIV     AB
                ADD     A,#030H
                CJNE    A,#030H,HEX2_LCD_NX
                MOV      A,#'0'
HEX2_LCD_NX:    MOV     TEMP_1000,A
                MOV     LCD_DATA,TEMP_1000
                ACALL   WRCHAR_LCD
                MOV     A,B
                MOV     B,#10
                DIV     AB
                ADD     A,#030H
                MOV     TEMP_100,A
                MOV     LCD_DATA,TEMP_100
                ACALL   WRCHAR_LCD
                MOV     A,B
                ADD     A,#030H
                MOV     TEMP_10,A
                MOV     LCD_DATA,TEMP_10
                ACALL   WRCHAR_LCD
                POP     ACC
                RET
;-------------------------------------------------------------------
; DS1820 Data Read
;-------------------------------------------------------------------
DS1820_RD:      MOV     R4,#8
```

```
                    CLR     A
DS1820_RD_LOOP:     CLR     ONEWIRE
                    NOP
                    NOP
                    SETB    ONEWIRE
                    NOP
                    NOP
                    NOP
                    NOP
                    MOV     C,ONEWIRE
                    ACALL   ONEWIRE_DELAY
                    RRC     A
                    DJNZ    R4,DS1820_RD_LOOP
                    MOV     ONEWIRE_DATA,A
                    RET
;-------------------------------------------------------------
; DS1820 Data Write
;-------------------------------------------------------------
DS1820_WR:          MOV     R4,#8
                    MOV     A,ONEWIRE_DATA
DS1820_WR_LOOP:     RRC     A
                    JNC     DS1820_WR_L
                    CLR     ONEWIRE
                    NOP
                    NOP
                    NOP
                    NOP
                    SETB    ONEWIRE
                    ACALL   ONEWIRE_DELAY
                    AJMP    DS1820_WR_NX
DS1820_WR_L:        CLR     ONEWIRE
                    ACALL   ONEWIRE_DELAY
```

```
                SETB    ONEWIRE
                NOP
                NOP
                NOP
                NOP
DS1820_WR_NX:   DJNZ    R4,DS1820_WR_LOOP
                RET
;-------------------------------------------------------------------
; DS1820 Reset
;-------------------------------------------------------------------
DS1820_RST:     CLR     ONEWIRE
                ACALL   DELAY_1ms
                SETB    ONEWIRE
                MOV     R4,#8
                DJNZ    R4,$
                RET
;-------------------------------------------------------------------
; DS1820 Receive Presence Pulse
;-------------------------------------------------------------------
DS1820_PRES:    MOV     R4,#8
DS1820_PRES_1:  MOV     R3,#0
DS1820_PRES_2:  JNB     ONEWIRE,DS1820_PRES_3
                DJNZ    R3,DS1820_PRES_2
                DJNZ    R4,DS1820_PRES_1
                RET
DS1820_PRES_3:  JNB     ONEWIRE,$
                MOV     R4,#8
                DJNZ    R4,$
                CLR     BUSY
                RET
```

```
;-------------------------------------------------------------------
; Wait keypad depressed
;-------------------------------------------------------------------
WAIT_KEY:           MOV     A,P2
                    ANL     A,#00FH
                    CJNE    A,#00FH,WAIT_KEY
                    RET
;-------------------------------------------------------------------
; Wait keypad pressed 0-9 Only
;-------------------------------------------------------------------
WAIT_KEYPRESSED:    ACALL   GET_KPAD
                    MOV     A,KPAD_DATA
                    CJNE    A,#0,CHK_KEY_NEXT
                    AJMP    WAIT_KEYPRESSED
CHK_KEY_NEXT:       CJNE    A,#10,CHK_KEY_0
                    AJMP    WAIT_KEYPRESSED
CHK_KEY_0:          CJNE    A,#11,CHK_VALID_KEY
                    MOV     KPAD_DATA,#0
                    RET
CHK_VALID_KEY:      JNC     WAIT_KEYPRESSED
                    RET
;-------------------------------------------------------------------
; Keypad Scan key Subroutine
;-------------------------------------------------------------------
GET_KPAD:           MOV     P2,#0FFH
                    MOV     KPAD_DATA,#0
CHK_COL0:           CLR     KPAD_COL0
                    MOV     A,P2
                    ANL     A,#00FH
                    CJNE    A,#00FH,COL0_DETECT
                    AJMP    CHK_COL1
COL0_DETECT:        MOV     KPAD_DATA,#01
```

```
                AJMP    GET_ROW
CHK_COL1:       SETB    KPAD_COL0
                CLR     KPAD_COL1
                MOV     A,P2
                ANL     A,#00FH
                CJNE    A,#00FH,COL1_DETECT
                AJMP    CHK_COL2
COL1_DETECT:    MOV     KPAD_DATA,#02
                AJMP    GET_ROW
CHK_COL2:       SETB    KPAD_COL1
                CLR     KPAD_COL2
                MOV     A,P2
                ANL     A,#00FH
                CJNE    A,#00FH,COL2_DETECT
                RET
COL2_DETECT:    MOV     KPAD_DATA,#03
GET_ROW:        CLR     KPAD_COL0
                CLR     KPAD_COL1
                CLR     KPAD_COL2
                JB      KPAD_ROW0,CHK_ROW1
                RET
CHK_ROW1:       JB      KPAD_ROW1,CHK_ROW2
                MOV     A,KPAD_DATA
                ADD     A,#3
                MOV     KPAD_DATA,A
                RET
CHK_ROW2:       JB      KPAD_ROW2,CHK_ROW3
                MOV     A,KPAD_DATA
                ADD     A,#6
                MOV     KPAD_DATA,A
                RET
CHK_ROW3:       MOV     A,KPAD_DATA
```

```
                ADD     A,#9
                MOV     KPAD_DATA,A
                RET
;-------------------------------------------------------------------
; LCD Initialize
;-------------------------------------------------------------------
INIT_LCD:       ACALL   DELAY_100ms
                CLR     LCD_RS
                MOV     P0,#00111000B
                ACALL   LCD_CLK
                ACALL   DELAY_10ms
                MOV     P0,#00111000B
                ACALL   LCD_CLK
                ACALL   LCD_OFF
                ACALL   LCD_CLR
                MOV     P0,#00000110B
                ACALL   LCD_CLK
                ACALL   LCD_HOME
;-------------------------------------------------------------------
; LCD Clear Display
;-------------------------------------------------------------------
LCD_CLR:        CLR     LCD_RS
                MOV     P0,#00000001B
                ACALL   LCD_CLK
                RET
;-------------------------------------------------------------------
; LCD Return Home
;-------------------------------------------------------------------
LCD_HOME:       CLR     LCD_RS
                MOV     P0,#00000010B
                ACALL   LCD_CLK
                RET
```

```
;-----------------------------------------------------------------
; LCD Display Off
;-----------------------------------------------------------------
LCD_OFF:        CLR     LCD_RS
                MOV     P0,#00001000B
                ACALL   LCD_CLK
                RET
;-----------------------------------------------------------------
; LCD Clk
;-----------------------------------------------------------------
LCD_CLK:        SETB    LCD_EN
                ACALL   LCD_DELAY
                CLR     LCD_EN
                ACALL   LCD_DELAY
                RET
;-----------------------------------------------------------------
; LCD Display On
;-----------------------------------------------------------------
LCD_ON:         CLR     LCD_RS
                MOV     P0,#00001100B
                ACALL   LCD_CLK
                RET
;-----------------------------------------------------------------
; LCD Cursor On
;-----------------------------------------------------------------
LCD_BLINK:      CLR     LCD_RS
                MOV     P0,#00001111B
                ACALL   LCD_CLK
                RET
```

```
;-------------------------------------------------------------------
; Set LCD Address
; I/P:          LCD_ADDR
;-------------------------------------------------------------------
SET_ADDR_LCD:   CLR     LCD_RS
                MOV     A,LCD_ADDR
                SETB    ACC.7
                MOV     P0,A
                ACALL   LCD_CLK
                RET
;-------------------------------------------------------------------
; Write Character to show LCD
; I/P:          LCD_DATA
;-------------------------------------------------------------------
WRCHAR_LCD:     SETB    LCD_RS
                MOV     P0,LCD_DATA
                ACALL   LCD_CLK
                ACALL   LCD_ON
                RET
;-------------------------------------------------------------------
; Write Line of 16 Character from ROM
; I/P:          DPTR : Locate ROM Address
;-------------------------------------------------------------------
WRLINE_LCD:     MOV     R0,#0
WRLINE_LCD_1:   SETB    LCD_RS
                CLR     A
                MOVC    A,@A+DPTR
                MOV     P0,A
                ACALL   LCD_CLK
                INC     DPTR
                INC     R0
                CJNE    R0,#8,WRLINE_LCD_1
```

```
                    MOV     LCD_ADDR,#040H
                    ACALL   SET_ADDR_LCD
WRLINE_LCD_2:       SETB    LCD_RS
                    CLR     A
                    MOVC    A,@A+DPTR
        MOV         P0,A
                    ACALL   LCD_CLK
                    INC     DPTR
                    INC     R0
                    CJNE    R0,#16,WRLINE_LCD_2
                    ACALL   LCD_ON
                    RET
;-------------------------------------------------------------------
; Dummy Delay time ONEWIRE_DELAY, LCD_DELAY, 50u, 100u, 1m, 10m, 100m, 1s
;-------------------------------------------------------------------
ONEWIRE_DELAY:      MOV     R6,#012H
ONEWIRE_DELAY_1:    NOP
                    NOP
                    DJNZ    R6,ONEWIRE_DELAY_1
                    RET
LCD_DELAY:          MOV     R7,#002
LCD_DELAY_1:        MOV     R6,#0E6H
LCD_DELAY_2:        NOP
                    NOP
                    DJNZ    R6,LCD_DELAY_2
                    DJNZ    R7,LCD_DELAY_1
                    RET
DELAY_50us:         MOV     R6,#00CH
DELAY_50us_1:       NOP
                    NOP
                    DJNZ    R6,DELAY_50us_1
                    RET
```

```
DELAY_100us:      MOV     R6,#017H
DELAY_100us_1:    NOP
                  NOP
                  DJNZ    R6,DELAY_100us_1
                  RET
DELAY_1ms:        MOV     R6,#0E6H
DELAY_1ms_1:      NOP
                  NOP
                  DJNZ    R6,DELAY_1ms_1
                  RET
DELAY_10ms:       MOV     R7,#010
DELAY_10ms_1:     MOV     R6,#0E6H
DELAY_10ms_2:     NOP
                  NOP
                  DJNZ    R6,DELAY_10ms_2
                  DJNZ    R7,DELAY_10ms_1
                  RET
DELAY_100ms:      MOV     R7,#100
DELAY_100ms_1:    MOV     R6,#0E6H
DELAY_100ms_2:    NOP
                  NOP
                  DJNZ    R6,DELAY_100ms_2
                  DJNZ    R7,DELAY_100ms_1
                  RET
DELAY_1s:         MOV     R5,#100
DELAY_1s_1:       ACALL   DELAY_10ms
                  DJNZ    R5,DELAY_1s_1
                  RET
```

```
;-------------------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;-------------------------------------------------------------------------
TITLE_1:        DB      ' 1-Wire  DS1820 '
SCR_TEMP:       DB      ' Temp :   . ',0DFH,'C '
NEW_TEMP:       DB      'NEW_TEM:  . ',0DFH,'C '
```

ภาคผนวก ข

DALLAS SEMICONDUCTOR

www.dalsemi.com

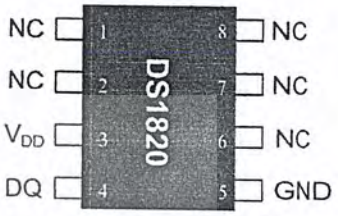
DS18S20
High Precision 1-Wire® Digital Thermometer

FEATURES

- Unique 1-wire interface requires only one port pin for communication
- Each device has a unique 64-bit serial code stored in an on-board ROM
- Multi-drop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line. Power supply range is 3.0V to 5.5V
- Measures temperatures from –55°C to +125°C (–67°F to +257°F)
- ±0.5°C accuracy from –10°C to +85°C
- 9-bit thermometer resolution
- Converts temperature in 750 ms (max.)
- User-definable nonvolatile 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

TO-92
(DS18S20)



8-pin 150-mil SOIC
(DS18S20Z)

PIN DESCRIPTION

GND - Ground
DQ - Data In/Out
V_{DD} - Power Supply Voltage
NC - No Connect

DESCRIPTION

The DS18S20 Digital Thermometer provides 9–bit centigrade temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18S20 communicates over a 1-wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. It has an operating temperature range of –55°C to +125°C and is accurate to ±0.5°C over the range of –10°C to +85°C. In addition, the DS18S20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18S20 has a unique 64-bit serial code, which allows multiple DS18S20s to function on the same 1–wire bus; thus, it is simple to use one microprocessor to control many DS18S20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment or machinery, and process monitoring and control systems.

DETAILED PIN DESCRIPTIONS Table 1

8-PIN SOIC*	TO-92	SYMBOL	DESCRIPTION
5	1	GND	**Ground.**
4	2	DQ	**Data Input/Output pin.** Open-drain 1-wire interface pin. Also provides power to the device when used in parasite power mode (see "Parasite Power" section.)
3	3	V_{DD}	**Optional V_{DD} pin.** V_{DD} must be grounded for operation in parasite power mode.

*All pins not specified in this table are "No Connect" pins.

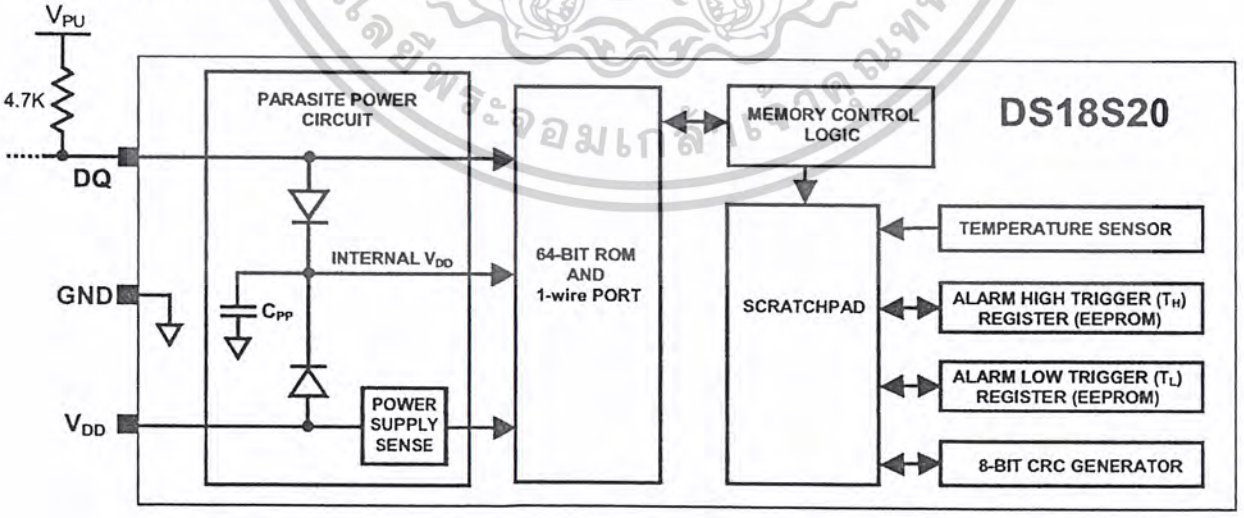
OVERVIEW

Figure 1 shows a block diagram of the DS18S20, and pin descriptions are given in Table 1. The 64-bit ROM stores the device's unique serial code. The scratchpad memory contains the 2-byte temperature register that stores the digital output from the temperature sensor. In addition, the scratchpad provides access to the 1-byte upper and lower alarm trigger registers (T_H and T_L). The T_H and T_L registers are nonvolatile (EEPROM), so they will retain data when the device is powered down.

The DS18S20 uses Dallas' exclusive 1-wire bus protocol that implements bus communication using one control signal. The control line requires a weak pullup resistor since all devices are linked to the bus via a 3-state or open-drain port (the DQ pin in the case of the DS18S20). In this bus system, the microprocessor (the master device) identifies and addresses devices on the bus using each device's unique 64-bit code. Because each device has a unique code, the number of devices that can be addressed on one bus is virtually unlimited. The 1-wire bus protocol, including detailed explanations of the commands and "time slots," is covered in the 1-WIRE BUS SYSTEM section of this datasheet.

Another feature of the DS18S20 is the ability to operate without an external power supply. Power is instead supplied through the 1-wire pullup resistor via the DQ pin when the bus is high. The high bus signal also charges an internal capacitor (C_{PP}), which then supplies power to the device when the bus is low. This method of deriving power from the 1-wire bus is referred to as "parasite power." As an alternative, the DS18S20 may also be powered by an external supply on V_{DD}.

DS18S20 BLOCK DIAGRAM Figure 1



OPERATION – MEASURING TEMPERATURE

The core functionality of the DS18S20 is its direct-to-digital temperature sensor. The temperature sensor output has 9-bit resolution, which corresponds to 0.5°C steps. The DS18S20 powers-up in a low-power idle state; to initiate a temperature measurement and A-to-D conversion, the master must issue a Convert T [44h] command. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18S20 returns to its idle state. If the DS18S20 is powered by an external supply, the master can issue "read time slots" (see the 1-WIRE BUS SYSTEM section) after the Convert T command and the DS18S20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. If the DS18S20 is powered with parasite power, this notification technique cannot be used since the bus must be pulled high by a strong pullup during the entire temperature conversion. The bus requirements for parasite power are explained in detail in the POWERING THE DS18S20 section of this datasheet.

The DS18S20 output data is calibrated in degrees centigrade; for Fahrenheit applications, a lookup table or conversion routine must be used. The temperature data is stored as a 16-bit sign-extended two's complement number in the temperature register (see Figure 2). The sign bits (S) indicate if the temperature is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. Table 2 gives examples of digital output data and the corresponding temperature reading.

Resolutions greater than 9 bits can be calculated using the data from the temperature, COUNT REMAIN and COUNT PER °C registers in the scratchpad. Note that the COUNT PER °C register is hard-wired to 16 (10h). After reading the scratchpad, the TEMP_READ value is obtained by truncating the 0.5°C bit (bit 0) from the temperature data (see Figure 2). The extended resolution temperature can then be calculated using the following equation:

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

Additional information about high-resolution temperature calculations can be found in Application Note 105: "High Resolution Temperature Measurement with Dallas Direct-to-Digital Temperature Sensors".

TEMPERATURE REGISTER FORMAT Figure 2

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}
	bit 15	**bit 14**	**bit 13**	**bit 12**	**bit 11**	**bit 10**	**bit 9**	**bit 8**
MS Byte	S	S	S	S	S	S	S	S

TEMPERATURE/DATA RELATIONSHIP Table 2

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+85.0°C*	0000 0000 1010 1010	00AAh
+25.0°C	0000 0000 0011 0010	0032h
+0.5°C	0000 0000 0000 0001	0001h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1111	FFFFh
-25.0°C	1111 1111 1100 1110	FFCEh
-55.0°C	1111 1111 1001 0010	FF92h

*The power-on reset value of the temperature register is +85°C

OPERATION – ALARM SIGNALING

After the DS18S20 performs a temperature conversion, the temperature value is compared to the user-defined two's complement alarm trigger values stored in the 1-byte T_H and T_L registers (see Figure 3). The sign bit (S) indicates if the value is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. The T_H and T_L registers are nonvolatile (EEPROM) so they will retain data when the device is powered down. T_H and T_L can be accessed through bytes 2 and 3 of the scratchpad as explained in the MEMORY section of this datasheet.

T_H AND T_L REGISTER FORMAT Figure 3

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
S	2^6	2^5	2^5	2^5	2^2	2^1	2^0

Only bits 8 through 1 of the temperature register are used in the T_H and T_L comparison since T_H and T_L are 8-bit registers. If the result of a temperature measurement is higher than T_H or lower than T_L, an alarm condition exists and an alarm flag is set inside the DS18S20. This flag is updated after every temperature measurement; therefore, if the alarm condition goes away, the flag will be turned off after the next temperature conversion.

The master device can check the alarm flag status of all DS18S20s on the bus by issuing an Alarm Search [ECh] command. Any DS18S20s with a set alarm flag will respond to the command, so the master can determine exactly which DS18S20s have experienced an alarm condition. If an alarm condition exists and the T_H or T_L settings have changed, another temperature conversion should be done to validate the alarm condition.

POWERING THE DS18S20

The DS18S20 can be powered by an external supply on the V_{DD} pin, or it can operate in "parasite power" mode, which allows the DS18S20 to function without a local external supply. Parasite power is very useful for applications that require remote temperature sensing or that are very space constrained. Figure 1 shows the DS18S20's parasite-power control circuitry, which "steals" power from the 1-wire bus via the DQ pin when the bus is high. The stolen charge powers the DS18S20 while the bus is high, and some of the charge is stored on the parasite power capacitor (C_{PP}) to provide power when the bus is low. When the DS18S20 is used in parasite power mode, the V_{DD} pin must be connected to ground.

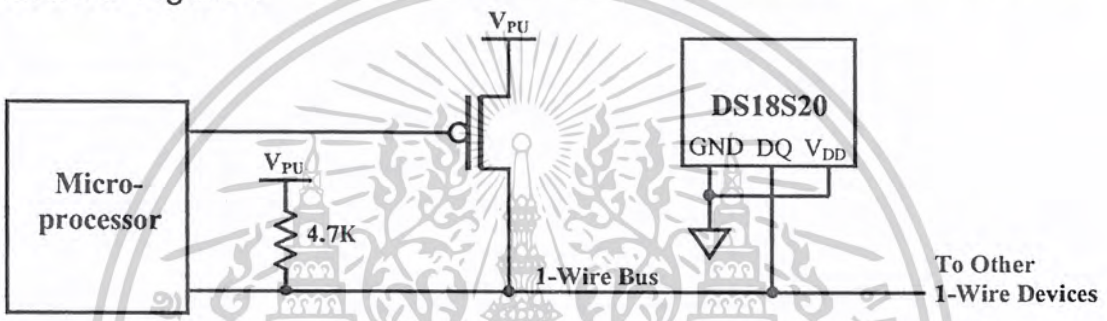
In parasite power mode, the 1-wire bus and C_{PP} can provide sufficient current to the DS18S20 for most operations as long as the specified timing and voltage requirements are met (refer to the DC ELECTRICAL CHARACTERISTICS and the AC ELECTRICAL CHARACTERISTICS sections of this data sheet). However, when the DS18S20 is performing temperature conversions or copying data from the scratchpad memory to EEPROM, the operating current can be as high as 1.5 mA. This current can cause an unacceptable voltage drop across the weak 1-wire pullup resistor and is more current than can be supplied by C_{PP}. To assure that the DS18S20 has sufficient supply current, it is necessary to provide a strong pullup on the 1-wire bus whenever temperature conversions are taking place or data is being copied from the scratchpad to EEPROM. This can be accomplished by using a MOSFET to pull the bus directly to the rail as shown in Figure 4. The 1-wire bus must be switched to the strong pullup within 10 µs (max) after a Convert T [44h] or Copy Scratchpad [48h] command is issued, and the bus must be held high by the pullup for the duration of the conversion (t_{conv}) or data transfer (t_{wr} = 10 ms). No other activity can take place on the 1-wire bus while the pullup is enabled.

The DS18S20 can also be powered by the conventional method of connecting an external power supply to the V_{DD} pin, as shown in Figure 5. The advantage of this method is that the MOSFET pullup is not required, and the 1–wire bus is free to carry other traffic during the temperature conversion time.

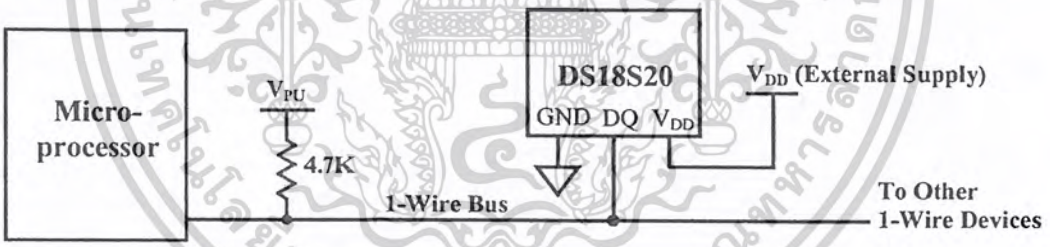
The use of parasite power is not recommended for temperatures above 100°C since the DS18S20 may not be able to sustain communications due to the higher leakage currents that can exist at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that the DS18S20 be powered by an external power supply.

In some situations the bus master may not know whether the DS18S20s on the bus are parasite powered or powered by external supplies. The master needs this information to determine if the strong bus pullup should be used during temperature conversions. To get this information, the master can issue a Skip ROM [CCh] command followed by a Read Power Supply [B4h] command followed by a "read time slot". During the read time slot, parasite powered DS18S20s will pull the bus low, and externally powered DS18S20s will let the bus remain high. If the bus is pulled low, the master knows that it must supply the strong pullup on the 1-wire bus during temperature conversions.

SUPPLYING THE PARASITE-POWERED DS18S20 DURING TEMPERATURE CONVERSIONS Figure 4



POWERING THE DS18S20 WITH AN EXTERNAL SUPPLY Figure 5



64-BIT LASERED ROM CODE

Each DS18S20 contains a unique 64–bit code (see Figure 6) stored in ROM. The least significant 8 bits of the ROM code contain the DS18S20's 1–wire family code: 10h. The next 48 bits contain a unique serial number. The most significant 8 bits contain a cyclic redundancy check (CRC) byte that is calculated from the first 56 bits of the ROM code. A detailed explanation of the CRC bits is provided in the CRC GENERATION section. The 64–bit ROM code and associated ROM function control logic allow the DS18S20 to operate as a 1–wire device using the protocol detailed in the 1-WIRE BUS SYSTEM section of this datasheet.

64-BIT LASERED ROM CODE Figure 6

8-BIT CRC		48-BIT SERIAL NUMBER		8-BIT FAMILY CODE (10h)	
MSB	LSB	MSB	LSB	MSB	LSB

MEMORY

The DS18S20's memory is organized as shown in Figure 7. The memory consists of an SRAM scratchpad with nonvolatile EEPROM storage for the high and low alarm trigger registers (T_H and T_L). Note that if the DS18S20 alarm function is not used, the T_H and T_L registers can serve as general-purpose memory. All memory commands are described in detail in the DS18S20 FUNCTION COMMANDS section.

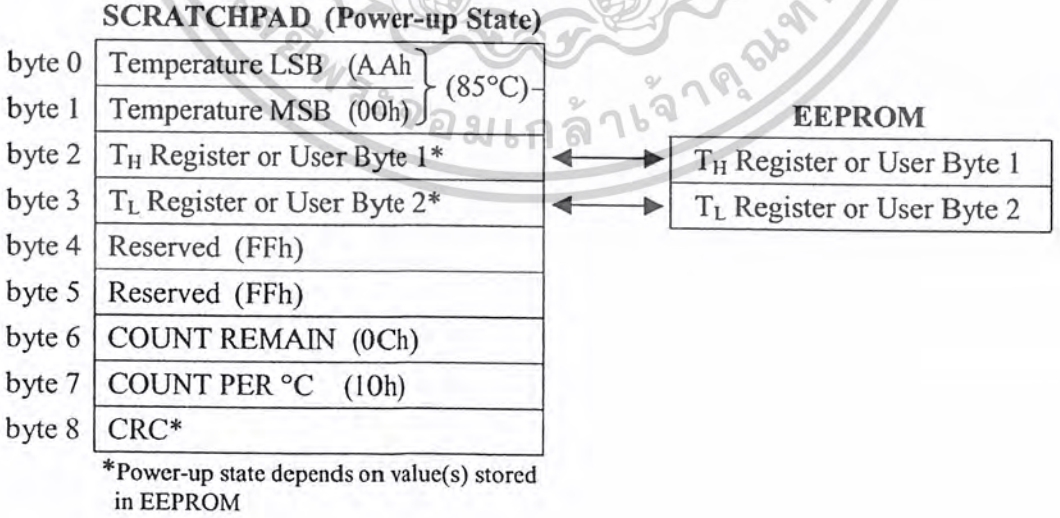
Byte 0 and byte 1 of the scratchpad contain the LSB and the MSB of the temperature register, respectively. These bytes are read-only. Bytes 2 and 3 provide access to T_H and T_L registers. Bytes 4 and 5 are reserved for internal use by the device and cannot be overwritten; these bytes will return all 1s when read. Bytes 6 and 7 contain the COUNT REMAIN and COUNT PER °C registers, which can be used to calculate extended resolution results as explained in the OPERATION – MEASURING TEMPERATURE section.

Byte 8 of the scratchpad is read-only and contains the cyclic redundancy check (CRC) code for bytes 0 through 7 of the scratchpad. The DS18S20 generates this CRC using the method described in the CRC GENERATION section.

Data is written to bytes 2 and 3 of the scratchpad using the Write Scratchpad [4Eh] command; the data must be transmitted to the DS18S20 starting with the least significant bit of byte 2. To verify data integrity, the scratchpad can be read (using the Read Scratchpad [BEh] command) after the data is written. When reading the scratchpad, data is transferred over the 1-wire bus starting with the least significant bit of byte 0. To transfer the T_H and T_L data from the scratchpad to EEPROM, the master must issue the Copy Scratchpad [48h] command.

Data in the EEPROM registers is retained when the device is powered down; at power-up the EEPROM data is reloaded into the corresponding scratchpad locations. Data can also be reloaded from EEPROM to the scratchpad at any time using the Recall E^2 [B8h] command. The master can issue "read time slots" (see the 1-WIRE BUS SYSTEM section) following the Recall E^2 command and the DS18S20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done.

DS18S20 MEMORY MAP cáÖ éÉ¤



CRC GENERATION

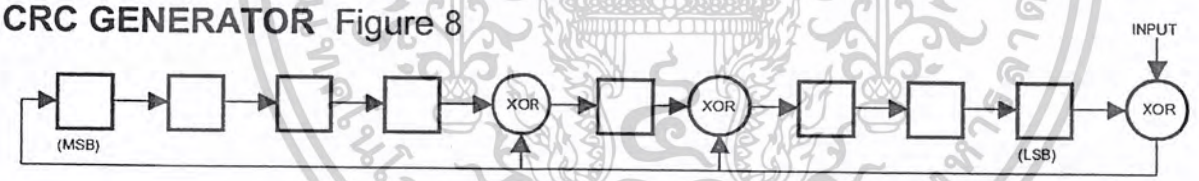
CRC bytes are provided as part of the DS18S20's 64-bit ROM code and in the 9th byte of the scratchpad memory. The ROM code CRC is calculated from the first 56 bits of the ROM code and is contained in the most significant byte of the ROM. The scratchpad CRC is calculated from the data stored in the scratchpad, and therefore it changes when the data in the scratchpad changes. The CRCs provide the bus master with a method of data validation when data is read from the DS18S20. To verify that data has been read correctly, the bus master must re-calculate the CRC from the received data and then compare this value to either the ROM code CRC (for ROM reads) or to the scratchpad CRC (for scratchpad reads). If the calculated CRC matches the read CRC, the data has been received error free. The comparison of CRC values and the decision to continue with an operation are determined entirely by the bus master. There is no circuitry inside the DS18S20 that prevents a command sequence from proceeding if the DS18S20 CRC (ROM or scratchpad) does not match the value generated by the bus master.

The equivalent polynomial function of the CRC (ROM or scratchpad) is:

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

The bus master can re-calculate the CRC and compare it to the CRC values from the DS18S20 using the polynomial generator shown in Figure 8. This circuit consists of a shift register and XOR gates, and the shift register bits are initialized to 0. Starting with the least significant bit of the ROM code or the least significant bit of byte 0 in the scratchpad, one bit at a time should shifted into the shift register. After shifting in the 56th bit from the ROM or the most significant bit of byte 7 from the scratchpad, the polynomial generator will contain the re-calculated CRC. Next, the 8-bit ROM code or scratchpad CRC from the DS18S20 must be shifted into the circuit. At this point, if the re-calculated CRC was correct, the shift register will contain all 0s. 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."

CRC GENERATOR Figure 8



1-WIRE BUS SYSTEM

The 1-wire bus system uses a single bus master to control one or more slave devices. The DS18S20 is always a slave. When there is only one slave on the bus, the system is referred to as a "single-drop" system; the system is "multi-drop" if there are multiple slaves on the bus.

All data and commands are transmitted least significant bit first over the 1-wire bus.

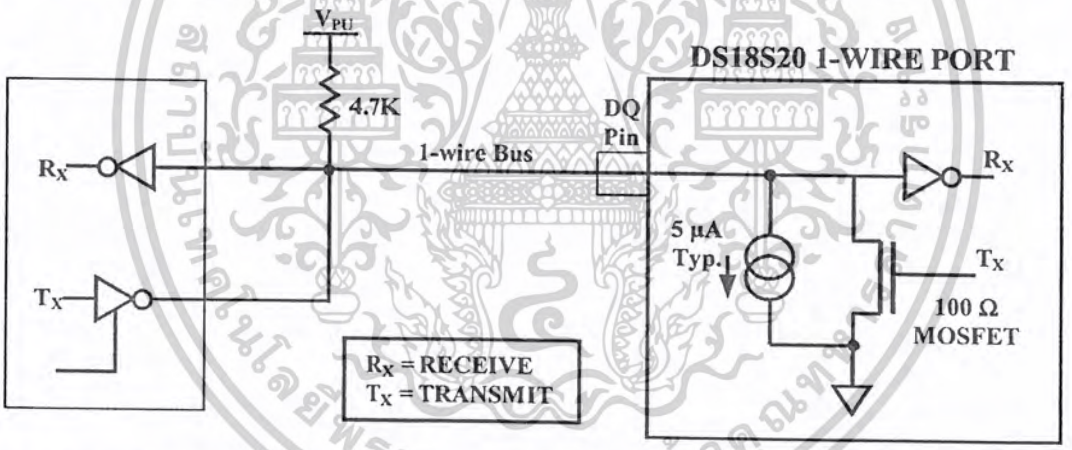
The following discussion of the 1-wire bus system is broken down into three topics: hardware configuration, transaction sequence, and 1-wire signaling (signal types and timing).

HARDWARE CONFIGURATION

The 1-wire bus has by definition only a single data line. Each device (master or slave) interfaces to the data line via an open drain or 3–state port. This allows each device to "release" the data line when the device is not transmitting data so the bus is available for use by another device. The 1-wire port of the DS18S20 (the DQ pin) is open drain with an internal circuit equivalent to that shown in Figure 9.

The 1-wire bus requires an external pullup resistor of approximately 5 kΩ; thus, 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 the bus is held low for more than 480 µs, all components on the bus will be reset.

HARDWARE CONFIGURATION Figure=V=



TRANSACTION SEQUENCE

The transaction sequence for accessing the DS18S20 is as follows:

Step 1. Initialization

Step 2. ROM Command (followed by any required data exchange)

Step 3. DS18S20 Function Command (followed by any required data exchange)

It is very important to follow this sequence every time the DS18S20 is accessed, as the DS18S20 will not respond if any steps in the sequence are missing or out of order. Exceptions to this rule are the Search ROM [F0h] and Alarm Search [ECh] commands. After issuing either of these ROM commands, the master must return to Step 1 in the sequence.

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 slave devices (such as the DS18S20) are on the bus and are ready to operate. Timing for the reset and presence pulses is detailed in the 1-WIRE SIGNALING section.

ROM COMMANDS

After the bus master has detected a presence pulse, it can issue a ROM command. These commands operate on the unique 64–bit ROM codes of each slave device and allow the master to single out a specific device if many are present on the 1-wire bus. These commands also allow the master to determine how many and what types of devices are present on the bus or if any device has experienced an alarm condition. There are five ROM commands, and each command is 8 bits long. The master device must issue an appropriate ROM command before issuing a DS18S20 function command. A flowchart for operation of the ROM commands is shown in Figure 14.

SEARCH ROM [F0h]

When a system is initially powered up, the master must identify the ROM codes of all slave devices on the bus, which allows the master to determine the number of slaves and their device types. The master learns the ROM codes through a process of elimination that requires the master to perform a Search ROM cycle (i.e., Search ROM command followed by data exchange) as many times as necessary to identify all of the slave devices. If there is only one slave on the bus, the simpler Read ROM command (see below) can be used in place of the Search ROM process. For a detailed explanation of the Search ROM procedure, refer to the iButton Book of Standards at www.ibutton.com/ibuttons/standard.pdf. After every Search ROM cycle, the bus master must return to Step 1 (Initialization) in the transaction sequence.

READ ROM [33h]

This command can only be used when there is one slave on the bus. It allows the bus master to read the slave's 64-bit ROM code without using the Search ROM procedure. If this command is used when there is more than one slave present on the bus, a data collision will occur when all the slaves attempt to respond at the same time.

MATCH ROM [55h]

The match ROM command followed by a 64–bit ROM code sequence allows the bus master to address a specific slave device on a multi-drop or single-drop bus. Only the slave that exactly matches the 64–bit ROM code sequence will respond to the function command issued by the master; all other slaves on the bus will wait for a reset pulse.

SKIP ROM [CCh]

The master can use this command to address all devices on the bus simultaneously without sending out any ROM code information. For example, the master can make all DS18S20s on the bus perform simultaneous temperature conversions by issuing a Skip ROM command followed by a Convert T [44h] command. Note, however, that the Skip ROM command can only be followed by the Read Scratchpad [BEh] command when there is one slave on the bus. This sequence saves time by allowing the master to read from the device without sending its 64–bit ROM code. This sequence will cause a data collision on the bus if there is more than one slave since multiple devices will attempt to transmit data simultaneously.

ALARM SEARCH [ECh]

The operation of this command is identical to the operation of the Search ROM command except that only slaves with a set alarm flag will respond. This command allows the master device to determine if any DS18S20s experienced an alarm condition during the most recent temperature conversion. After

every Alarm Search cycle (i.e., Alarm Search command followed by data exchange), the bus master must return to Step 1 (Initialization) in the transaction sequence. Refer to the OPERATION – ALARM SIGNALING section for an explanation of alarm flag operation.

DS18S20 FUNCTION COMMANDS

After the bus master has used a ROM command to address the DS18S20 with which it wishes to communicate, the master can issue one of the DS18S20 function commands. These commands allow the master to write to and read from the DS18S20's scratchpad memory, initiate temperature conversions and determine the power supply mode. The DS18S20 function commands, which are described below, are summarized in Table 4 and illustrated by the flowchart in Figure 15.

CONVERT T [44h]

This command initiates a single temperature conversion. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18S20 returns to its low-power idle state. If the device is being used in parasite power mode, within 10 µs (max) after this command is issued the master must enable a strong pullup on the 1-wire bus for the duration of the conversion (t_{conv}) as described in the POWERING THE DS18S20 section. If the DS18S20 is powered by an external supply, the master can issue read time slots after the Convert T command and the DS18S20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. In parasite power mode this notification technique cannot be used since the bus is pulled high by the strong pullup during the conversion.

WRITE SCRATCHPAD [4Eh]

This command allows the master to write 2 bytes of data to the DS18S20's scratchpad. The first byte is written into the T_H register (byte 2 of the scratchpad), and the second byte is written into the T_L register (byte 3 of the scratchpad). Data must be transmitted least significant bit first. Both bytes MUST be written before the master issues a reset, or the data may be corrupted.

READ SCRATCHPAD [BEh]

This command allows the master to read the contents of the scratchpad. The data transfer starts with the least significant bit of byte 0 and continues through the scratchpad until the 9th byte (byte 8 – CRC) is read. The master may issue a reset to terminate reading at any time if only part of the scratchpad data is needed.

COPY SCRATCHPAD [48h]

This command copies the contents of the scratchpad T_H and T_L registers (bytes 2 and 3) to EEPROM. If the device is being used in parasite power mode, within 10 µs (max) after this command is issued the master must enable a strong pullup on the 1-wire bus for at least 10 ms as described in the POWERING THE DS18S20 section.

RECALL E^2 [B8h]

This command recalls the alarm trigger values (T_H and T_L) from EEPROM and places the data in bytes 2 and 3, respectively, in the scratchpad memory. The master device can issue read time slots following the Recall E^2 command and the DS18S20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done. The recall operation happens automatically at power-up, so valid data is available in the scratchpad as soon as power is applied to the device.

READ POWER SUPPLY [B4h]

The master device issues this command followed by a read time slot to determine if any DS18S20s on the bus are using parasite power. During the read time slot, parasite powered DS18S20s will pull the bus low, and externally powered DS18S20s will let the bus remain high. Refer to the POWERING THE DS18S20 section for usage information for this command.

DS18S20 FUNCTION COMMAND SET Table 4

Command	Description	Protocol	1-Wire Bus Activity After Command is Issued	Notes
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18S20 transmits conversion status to master (not applicable for parasite-powered DS18S20s).	1
MEMORY COMMANDS				
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18S20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2 and 3 (T_H and T_L).	4Eh	Master transmits 2 data bytes to DS18S20.	3
Copy Scratchpad	Copies T_H and T_L data from the scratchpad to EEPROM.	48h	None	1
Recall E^2	Recalls T_H and T_L data from EEPROM to the scratchpad.	B8h	DS18S20 transmits recall status to master.	
Read Power Supply	Signals DS18S20 power supply mode to the master.	B4h	DS18S20 transmits supply status to master.	

NOTES:

1. For parasite-powered DS18S20s, the master must enable a strong pullup on the 1-wire bus during temperature conversions and copies from the scratchpad to EEPROM. No other bus activity may take place during this time.
2. The master can interrupt the transmission of data at any time by issuing a reset.
3. Both bytes must be written before a reset is issued.

1-WIRE SIGNALING

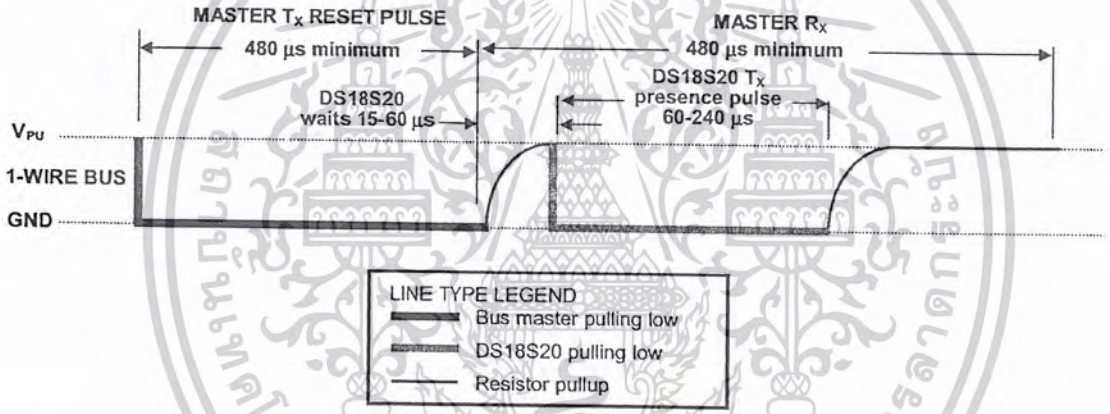
The DS18S20 uses a strict 1-wire communication protocol to insure data integrity. Several signal types are defined by this protocol: reset pulse, presence pulse, write 0, write 1, read 0, and read 1. All of these signals, with the exception of the presence pulse, are initiated by the bus master.

INITIALIZATION PROCEDURE: RESET AND PRESENCE PULSES

All communication with the DS18S20 begins with an initialization sequence that consists of a reset pulse from the master followed by a presence pulse from the DS18S20. This is illustrated in Figure 10. When the DS18S20 sends the presence pulse in response to the reset, it is indicating to the master that it is on the bus and ready to operate.

During the initialization sequence the bus master transmits (T_X) the reset pulse by pulling the 1-wire bus low for a minimum of 480 µs. The bus master then releases the bus and goes into receive mode (R_X). When the bus is released, the 5k pullup resistor pulls the 1-wire bus high. When the DS18S20 detects this rising edge, it waits 15–60 µs and then transmits a presence pulse by pulling the 1-wire bus low for 60–240 µs.

INITIALIZATION TIMING Figure 10



READ/WRITE TIME SLOTS

The bus master writes data to the DS18S20 during write time slots and reads data from the DS18S20 during read time slots. One bit of data is transmitted over the 1-wire bus per time slot.

WRITE TIME SLOTS

There are two types of write time slots: "Write 1" time slots and "Write 0" time slots. The bus master uses a Write 1 time slot to write a logic 1 to the DS18S20 and a Write 0 time slot to write a logic 0 to the DS18S20. All write time slots must be a minimum of 60 µs in duration with a minimum of a 1 µs recovery time between individual write slots. Both types of write time slots are initiated by the master pulling the 1-wire bus low (see Figure 11).

To generate a Write 1 time slot, after pulling the 1-wire bus low, the bus master must release the 1-wire bus within 15 µs. When the bus is released, the 5k pullup resistor will pull the bus high. To generate a Write 0 time slot, after pulling the 1-wire bus low, the bus master must continue to hold the bus low for the duration of the time slot (at least 60 µs). The DS18S20 samples the 1-wire bus during a window that lasts from 15 µs to 60 µs after the master initiates the write time slot. If the bus is high during the sampling window, a 1 is written to the DS18S20. If the line is low, a 0 is written to the DS18S20.

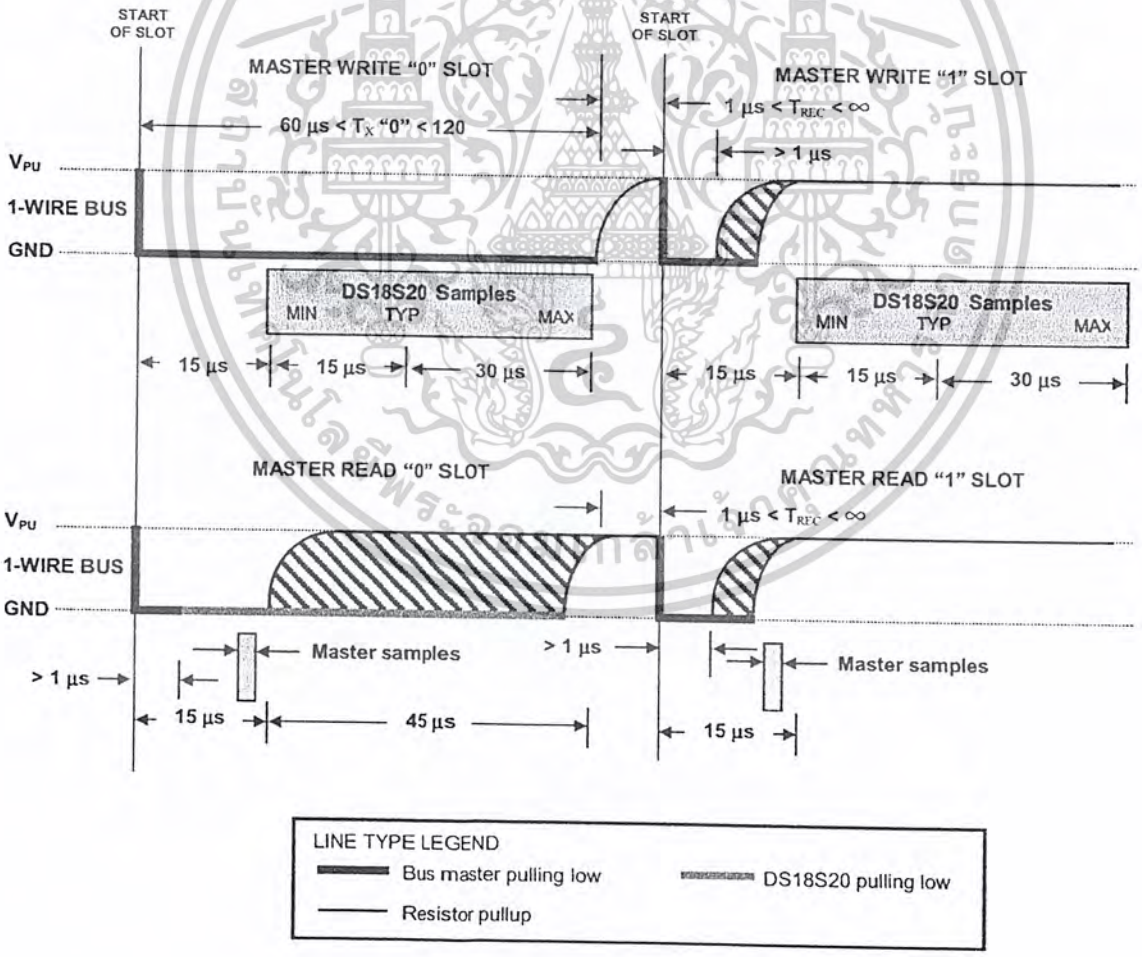
READ TIME SLOTS

The DS18S20 can only transmit data to the master when the master issues read time slots. Therefore, the master must generate read time slots immediately after issuing a Read Scratchpad [BEh] or Read Power Supply [B4h] command, so that the DS18S20 can provide the requested data. In addition, the master can generate read time slots after issuing Convert T [44h] or Recall E^2 [B8h] commands to find out the status of the operation as explained in the DS18S20 FUNCTION COMMAND section.

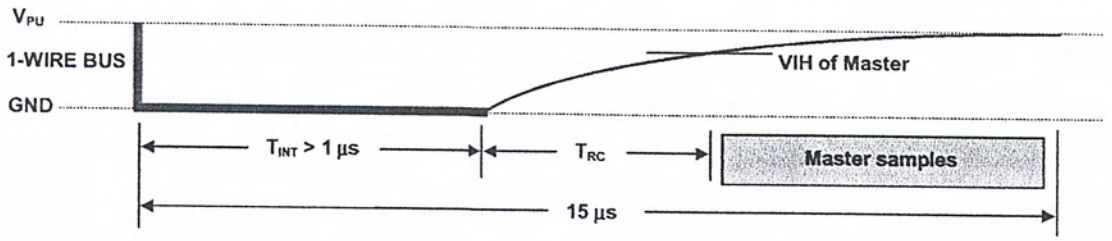
All read time slots must be a minimum of 60 µs in duration with a minimum of a 1 µs recovery time between slots. A read time slot is initiated by the master device pulling the 1-wire bus low for a minimum of 1 µs and then releasing the bus (see Figure 11). After the master initiates the read time slot, the DS18S20 will begin transmitting a 1 or 0 on bus. The DS18S20 transmits a 1 by leaving the bus high and transmits a 0 by pulling the bus low. When transmitting a 0, the DS18S20 will release the bus by the end of the time slot, and the bus will be pulled back to its high idle state by the pullup resister. Output data from the DS18S20 is valid for 15 µs after the falling edge that initiated the read time slot. Therefore, the master must release the bus and then sample the bus state within 15 µs from the start of the slot.

Figure 12 illustrates that the sum of T_{INIT}, T_{RC}, and T_{SAMPLE} must be less than 15 µs for a read time slot. Figure 13 shows that system timing margin is maximized by keeping T_{INIT} and T_{RC} as short as possible and by locating the master sample time during read time slots towards the end of the 15 µs period.

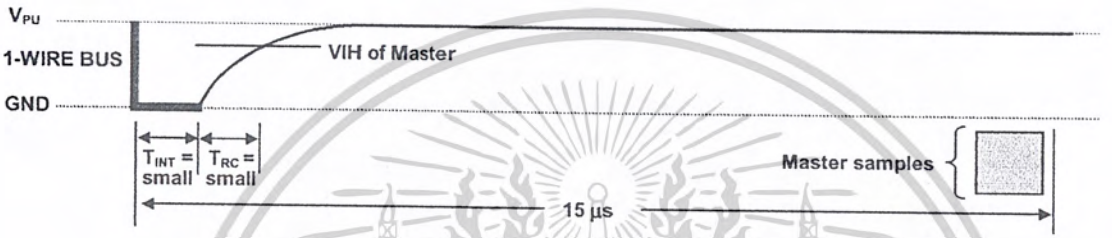
READ/WRITE TIME SLOT TIMING DIAGRAM Figure 11



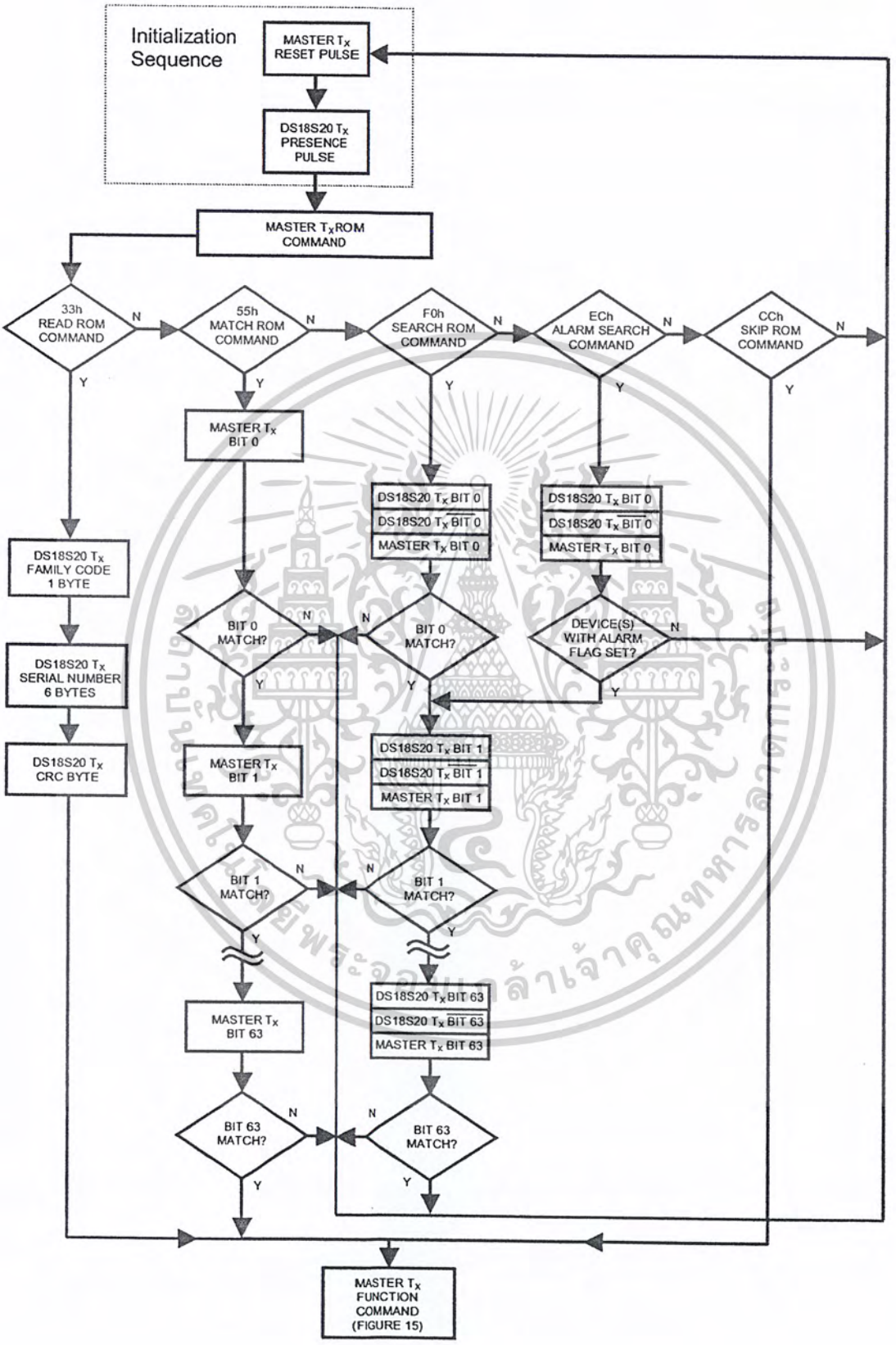
DETAILED MASTER READ 1 TIMING Figure 12



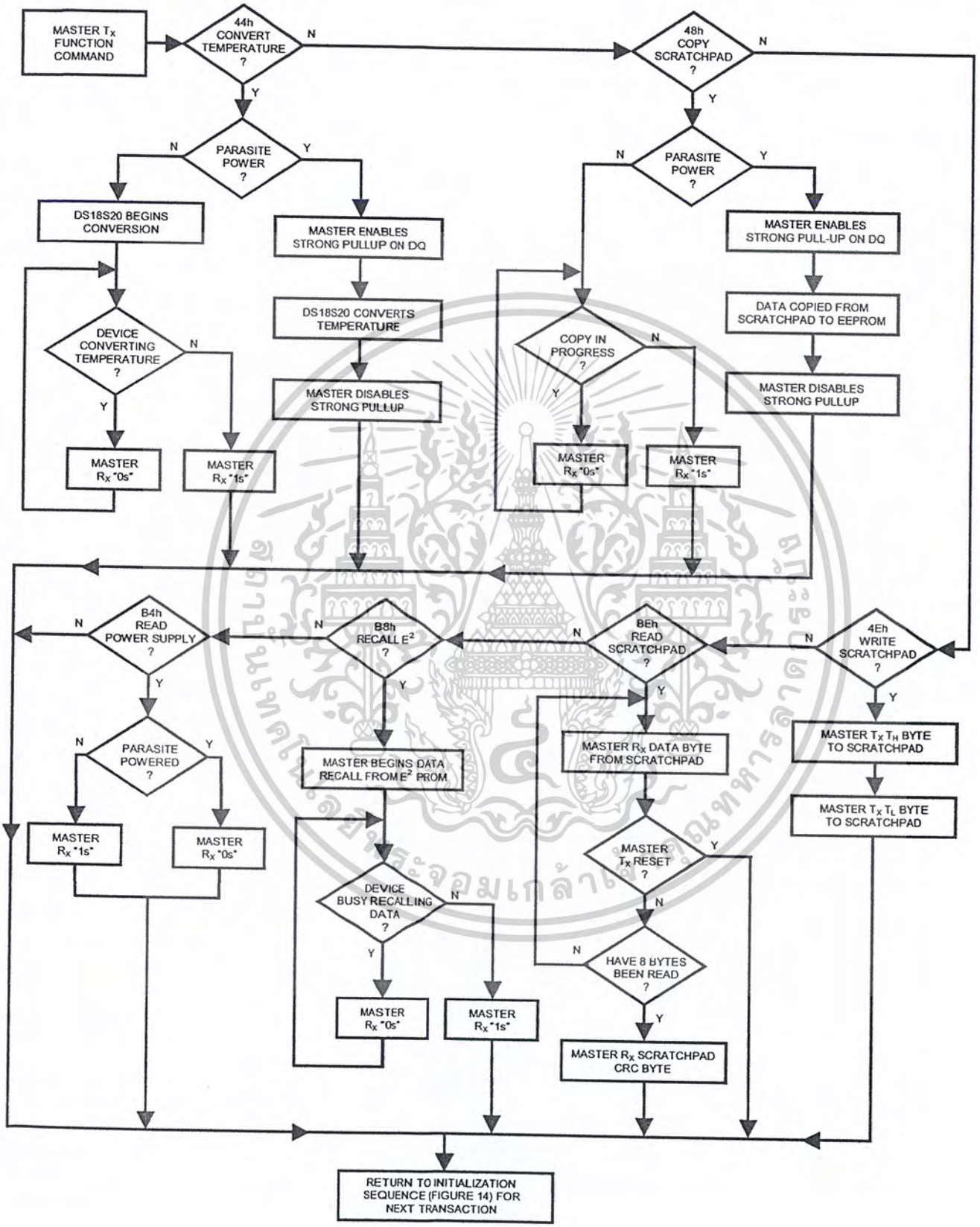
RECOMMENDED MASTER READ 1 TIMING Figure 13



ROM COMMANDS FLOW CHART Figure 14



DS18S20 FUNCTION COMMANDS FLOW CHART Figure 15



DS18S20 OPERATION EXAMPLE 1

In this example there are multiple DS18S20s on the bus and they are using parasite power. The bus master initiates a temperature conversion in a specific DS18S20 and then reads its scratchpad and recalculates the CRC to verify the data.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20s respond with presence pulse.
TX	55h	Master issues Match ROM command.
TX	64-bit ROM code	Master sends DS18S20 ROM code.
TX	44h	Master issues Convert T command.
TX	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{conv}).
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20s respond with presence pulse.
TX	55h	Master issues Match ROM command.
TX	64-bit ROM code	Master sends DS18S20 ROM code.
TX	BEh	Master issues Read Scratchpad command.
RX	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.

DS18S20 OPERATION EXAMPLE 2

In this example there is only one DS18S20 on the bus and it is using parasite power. The master writes to the T_H and T_L registers in the DS18S20 scratchpad and then reads the scratchpad and recalculates the CRC to verify the data. The master then copies the scratchpad contents to EEPROM.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	4Eh	Master issues Write Scratchpad command.
TX	2 data bytes	Master sends two data bytes to scratchpad (T_H and T_L)
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	BEh	Master issues Read Scratchpad command.
RX	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	48h	Master issues Copy Scratchpad command.
TX	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10 ms while copy operation is in progress.

DS18S20 OPERATION EXAMPLE 3

In this example there is only one DS18S20 on the bus and it is using parasite power. The bus master initiates a temperature conversion then reads the DS18S20 scratchpad and calculates a higher resolution result using the data from the temperature, COUNT REMAIN and COUNT PER °C registers.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
TX	Reset	Master issues reset pulse.
TR	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	44h	Master issues Convert T command.
TX	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{conv}).
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
TX	CCh	Master issues Skip ROM command.
TX	BEh	Master issues Read Scratchpad command.
RX	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated. The master also calculates the TEMP_READ value and stores the contents of the COUNT REMAIN and COUNT PER °C registers.
TX	Reset	Master issues reset pulse.
RX	Presence	DS18S20 responds with presence pulse.
-	-	CPU calculates extended resolution temperature using the equation in the OPERATION - MEASURING TEMPERATURE section of this datasheet.

RELATED APPLICATION NOTES

The following Application Notes can be applied to the DS18S20. These notes can be obtained from the Dallas Semiconductor "Application Note Book," via the Dallas 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-Digital 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 Dallas website or anonymous FTP Site.

ABSOLUTE MAXIMUM RATINGS*

Voltage on any pin relative to ground	–0.5V to +6.0V
Operating temperature	–55°C to +125°C
Storage temperature	–55°C to +125°C
Soldering temperature	See J-STD-020A Specification

*These are stress ratings 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.

DC ELECTRICAL CHARACTERISTICS (-55°C to +125°C; V_{DD}=3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{DD}	Local Power	+3.0		+5.5	V	1
Pullup Supply Voltage	V_{PU}	Parasite Power	+3.0		+5.5	V	1,2
		Local Power	+3.0		V_{DD}		
Thermometer Error	t_{ERR}	-10°C to +85°C			±0.5	°C	3
		-55°C to +125°C			±2		
Input Logic Low	V_{IL}		-0.3		+0.8	V	1,4,5
Input Logic High	V_{IH}	Local Power	+2.2		The lower of 5.5 or V_{DD} + 0.3	V	1, 6
		Parasite Power	+3.0				
Sink Current	I_L	$V_{I/O}$=0.4V	4.0			mA	1
Standby Current	I_{DDS}			750	1000	nA	7,8
Active Current	I_{DD}	V_{DD}=5V		1	1.5	mA	9
DQ Input Current	I_{DQ}			5		µA	10
Drift				±0.2		°C	11

NOTES:

1. All voltages are referenced to ground.
2. The Pullup Supply Voltage specification assumes that the pullup device is ideal, and therefore the high level of the pullup is equal to V_{PU}. In order to meet the V_{IH} spec of the DS18S20, the actual supply rail for the strong pullup transistor must include margin for the voltage drop across the transistor when it is turned on; thus: $V_{PU\_ACTUAL} = V_{PU\_IDEAL} + V_{TRANSISTOR}$.
3. See typical performance curve in Figure 16
4. Logic low voltages are specified at a sink current of 4 mA.
5. To guarantee a presence pulse under low voltage parasite power conditions, V_{ILMAX} may have to be reduced to as low as 0.5V.
6. Logic high voltages are specified at a source current of 1 mA.
7. Standby current specified up to 70°C. Standby current typically is 3 µA at 125°C.
8. To minimize I_{DDS}, DQ should be within the following ranges: GND ≤ DQ ≤ GND + 0.3V or V_{DD} – 0.3V ≤ DQ ≤ V_{DD}.
9. Active current refers to supply current during active temperature conversions or EEPROM writes.
10. DQ line is high ("hi-Z" state).
11. Drift data is based on a 1000 hour stress test at 125°C with V_{DD} = 5.5V.

AC ELECTRICAL CHARACTERISTICS: NV MEMORY

(-55°C to +100°C; V_{DD}=3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS
NV Write Cycle Time	t_{WR}			2	10	ms
EEPROM Writes	N_{EEWR}	-55°C to +55°C	50k			writes
EEPROM Data Retention	t_{EEDR}	-55°C to +55°C	10			years

AC ELECTRICAL CHARACTERISTICS

(-55°C to +125°C; V_{DD}=3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITION	MIN	TYP	MAX	UNITS	NOTES
Temperature Conversion Time	t_{CONV}				750	ms	1
Time to Strong Pullup On	t_{SPON}	Start Convert T Command Issued			10	µs	
Time Slot	t_{SLOT}		60		120	µs	1
Recovery Time	t_{REC}		1			µs	1
Write 0 Low Time	t_{LOW0}		60		120	µs	1
Write 1 Low Time	t_{LOW1}		1		15	µs	1
Read Data Valid	t_{RDV}				15	µs	1
Reset Time High	t_{RSTH}		480			µs	1
Reset Time Low	t_{RSTL}		480			µs	1,2
Presence Detect High	t_{PDHIGH}		15		60	µs	1
Presence Detect Low	t_{PDLOW}		60		240	µs	1
Capacitance	$C_{IN/OUT}$				25	pF	

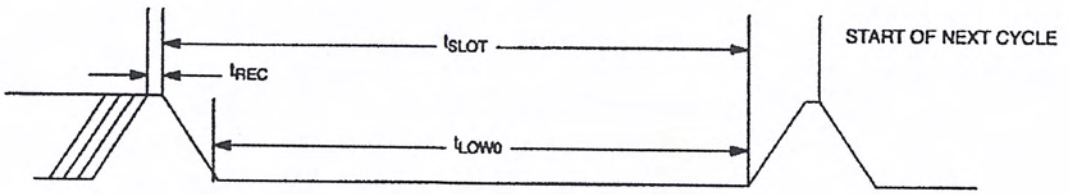
NOTES:

1. Refer to timing diagrams in Figure 17.
2. Under parasite power, if t_{RSTL} > 960 µs, a power on reset may occur.

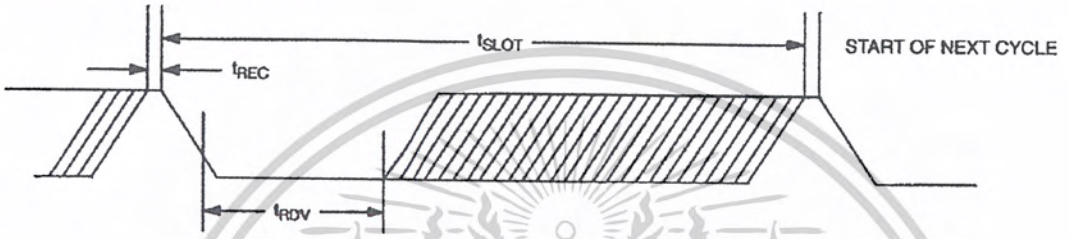
TYPICAL PERFORMANCE CURVE Figure 16

TIMING DIAGRAMS Figure 17

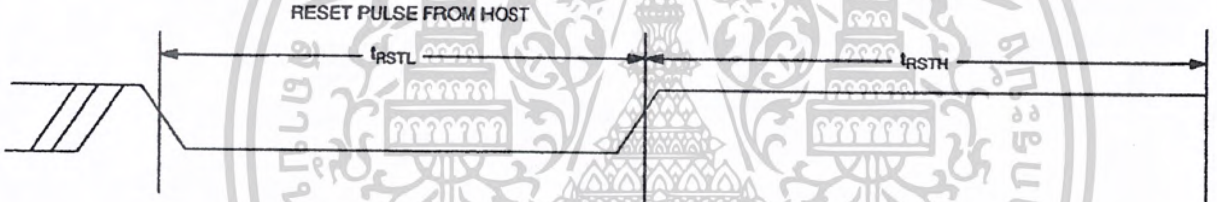
1–WIRE WRITE ZERO TIME SLOT



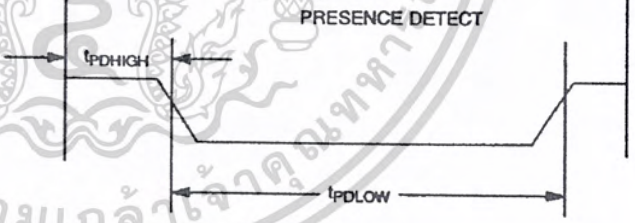
1–WIRE READ ZERO TIME SLOT



1–WIRE RESET PULSE



1–WIRE PRESENCE DETECT



ภาคผนวก ค

Features

- Compatible with MCS-51™ Products
- 4K Bytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

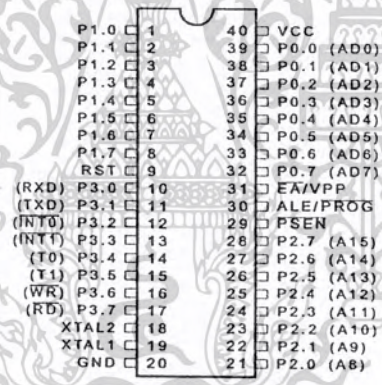
Description

The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash Programmable and Erasable Read Only Memory (PEROM). The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

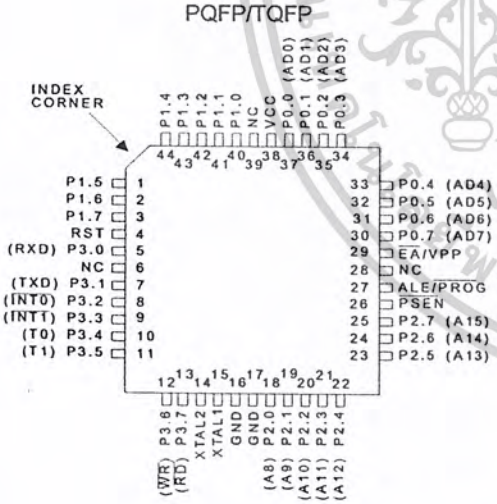
(continued)

Pin Configurations

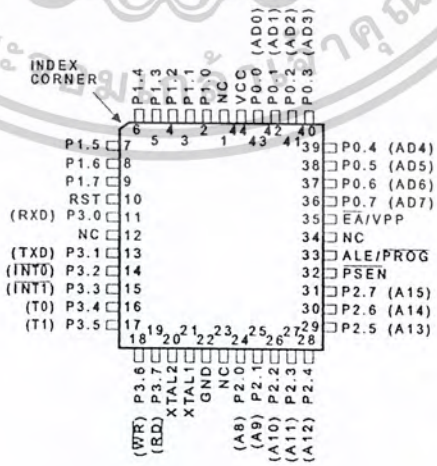
PDIP



PQFP/TQFP



PLCC

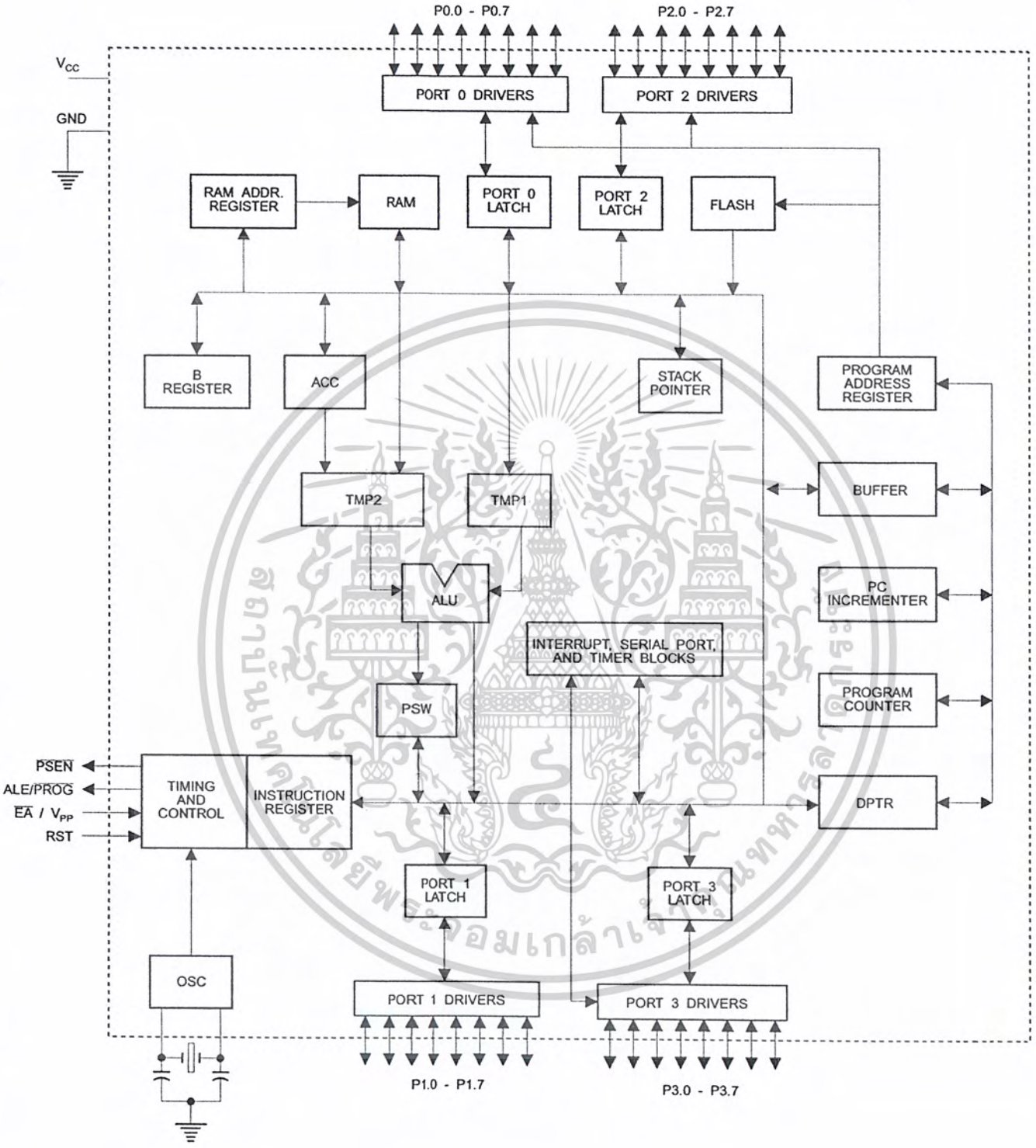


8-Bit Microcontroller with 4K Bytes Flash

AT89C51

0265F-A–12/97

Block Diagram



The AT89C51 provides the following standard features: 4K bytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT89C51 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 hardware reset.

Pin Description

V_{CC}
Supply voltage.

GND
Ground.

Port 0
Port 0 is an 8-bit open drain bidirectional 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 may 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 bidirectional 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.

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

Port 2
Port 2 is an 8-bit bidirectional 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 it 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 bidirectional 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 AT89C51 as listed below:

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)

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

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 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 AT89C51 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.

$\overline{EA}$/V_{PP}
External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external program 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, for parts that require 12-volt V_{PP}.

XTAL1
Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2
Output from the inverting oscillator amplifier.

Oscillator Characteristics

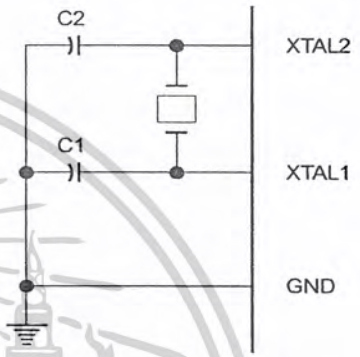
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. 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.

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.

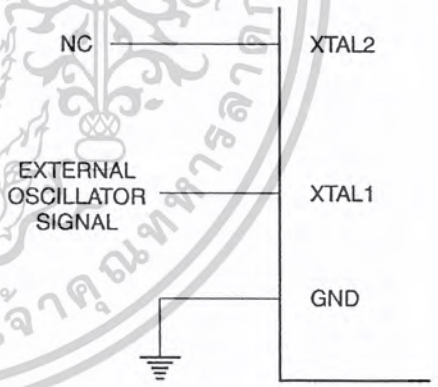
It should be noted that when idle is terminated by a hard ware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port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

Figure 1. Oscillator Connections



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

Figure 2. External Clock Drive Configuration



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. The only exit from power down is a hardware reset. 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.

Program Memory Lock Bits

On the chip are three lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

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. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

Lock Bit Protection Modes

Program Lock Bits				Protection Type
	LB1	LB2	LB3	
1	U	U	U	No program lock features.
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 is disabled.
3	P	P	U	Same as mode 2, also verify is disabled.
4	P	P	P	Same as mode 3, also external execution is disabled.

Programming the Flash

The AT89C51 is normally shipped with the on-chip Flash memory array in the erased state (that is, contents = FFH) and ready to be programmed. The programming interface accepts either a high-voltage (12-volt) or a low-voltage (V_{CC}) program enable signal. The low voltage programming mode provides a convenient way to program the AT89C51 inside the user's system, while the high-voltage programming mode is compatible with conventional third party Flash or EPROM programmers.

The AT89C51 is shipped with either the high-voltage or low-voltage programming mode enabled. The respective top-side marking and device signature codes are listed in the following table.

	V_{PP} = 12V	V_{PP} = 5V
Top-Side Mark	AT89C51 xxxx yyww	AT89C51 xxxx-5 yyww
Signature	(030H)=1EH (031H)=51H (032H)=FFH	(030H)=1EH (031H)=51H (032H)=05H

The AT89C51 code memory array is programmed byte-by-byte in either programming mode. *To program any non-blank byte in the on-chip Flash Memory, the entire memory must be erased using the Chip Erase Mode.*

Programming Algorithm: Before programming the AT89C51, the address, data and control signals should be set up according to the Flash programming mode table and Figures 3 and 4. To program the AT89C51, take the following steps.

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}/V_{PP}$ to 12V for the high-voltage programming mode.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 1.5 ms. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

Data Polling: The AT89C51 features $\overline{Data}$ Polling to indicate the end of a write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written datum on PO.7. 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 can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.4 is pulled low after ALE goes high during programming to indicate 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 data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

Chip Erase: The entire Flash array is erased electrically by using the proper combination of control signals and by holding ALE/$\overline{PROG}$ low for 10 ms. The code array is written with all "1"s. The chip erase operation must be executed before the code memory can be re-programmed.

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H,

031H, and 032H, 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) = 51H indicates 89C51
(032H) = FFH indicates 12V programming
(032H) = 05H indicates 5V programming

Programming Interface

Every code byte in the Flash array 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.

Flash Programming Modes

Mode		RST	$\overline{PSEN}$	ALE/$\overline{PROG}$	$\overline{EA}$/V_{PP}	P2.6	P2.7	P3.6	P3.7
Write Code Data		H	L		H/12V	L	H	H	H
Read Code Data		H	L	H	H	L	L	H	H
Write Lock	Bit - 1	H	L		H/12V	H	H	H	H
	Bit - 2	H	L		H/12V	H	H	L	L
	Bit - 3	H	L		H/12V	H	L	H	L
Chip Erase		H	L	 (1)	H/12V	H	L	L	L
Read Signature Byte		H	L	H	H	L	L	L	L

Note: 1. Chip Erase requires a 10-ms PROG pulse.

Figure 3. Programming the Flash

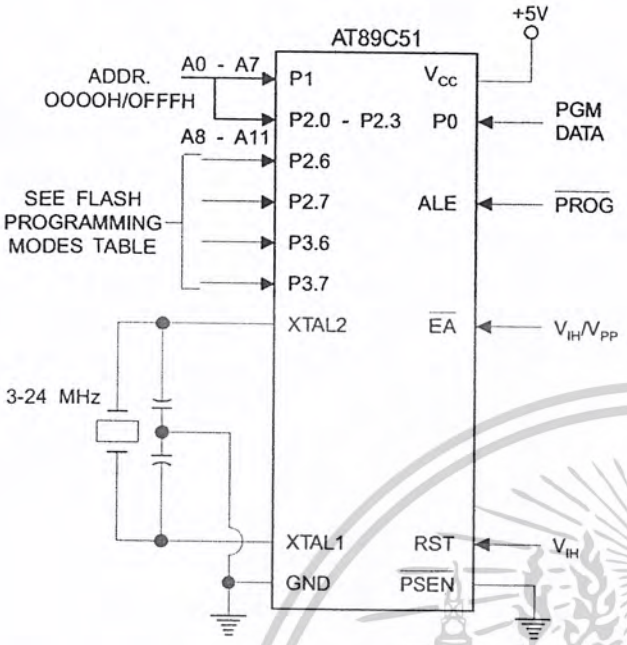
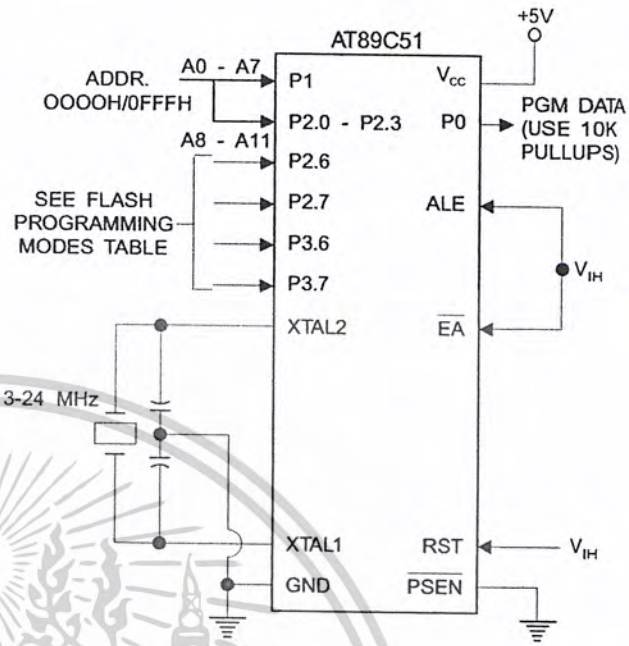


Figure 4. Verifying the Flash



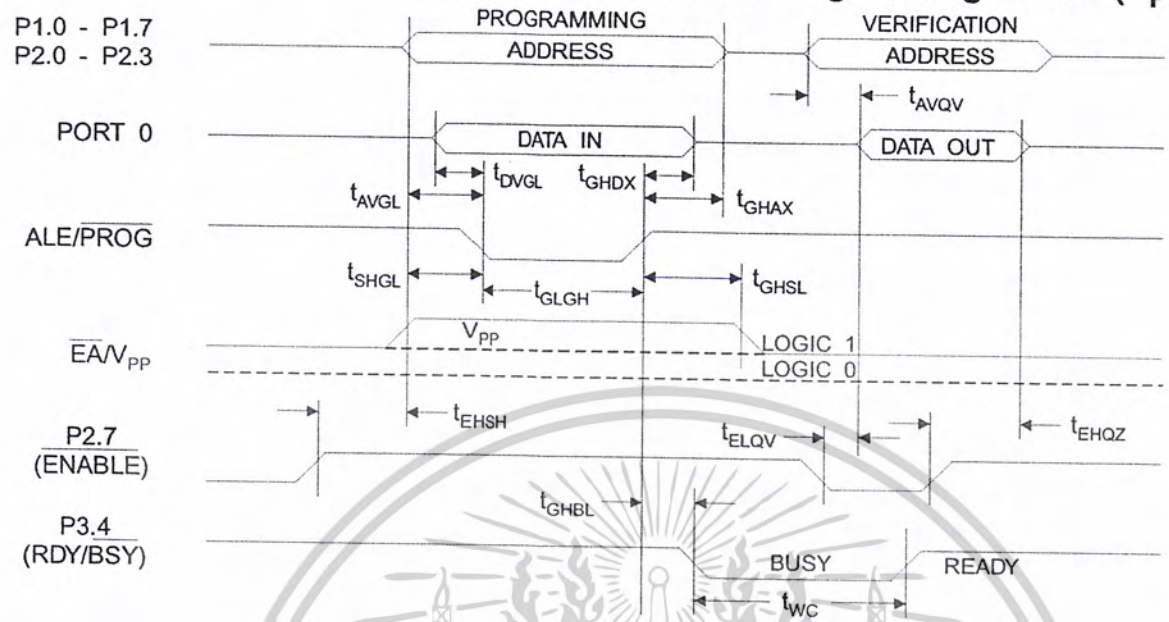
Flash Programming and Verification Characteristics

T_A = 0°C to 70°C, V_{CC} = 5.0 ± 10%

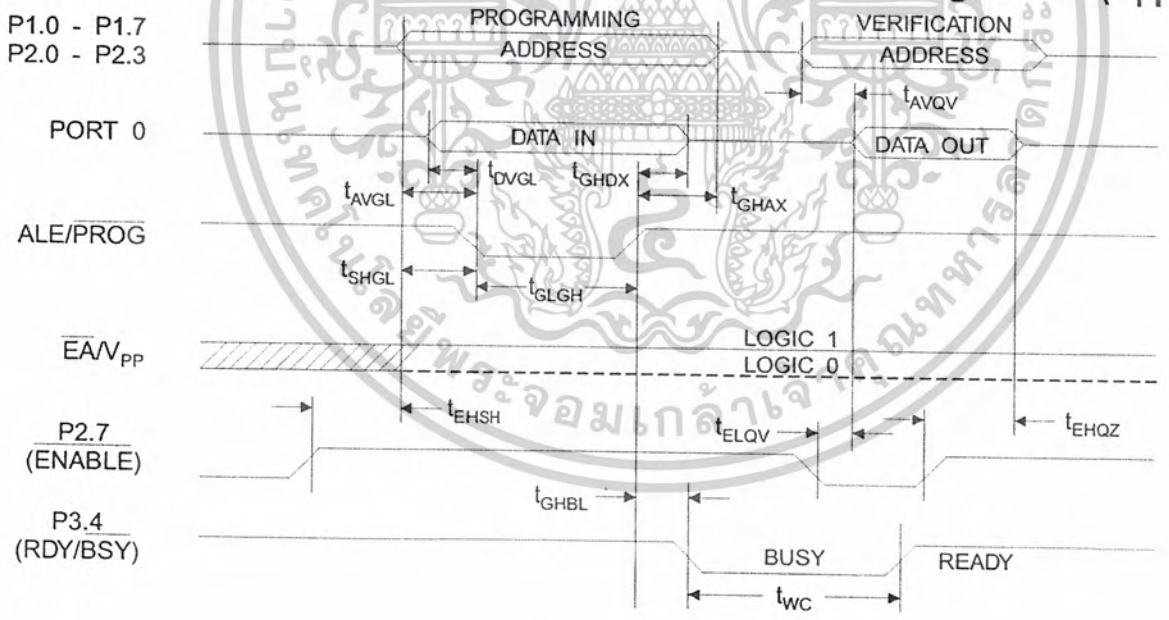
Symbol	Parameter	Min	Max	Units
V_{PP}(1)	Programming Enable Voltage	11.5	12.5	V
I_{PP}(1)	Programming Enable Current		1.0	mA
$1/t_{CLCL}$	Oscillator Frequency	3	24	MHz
t_{AVGL}	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHAX}	Address Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}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

Note: 1. Only used in 12-volt programming mode.

Flash Programming and Verification Waveforms - High Voltage Mode (V_{PP} = 12V)



Flash Programming and Verification Waveforms - Low Voltage Mode (V_{PP} = 5V)



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

T_A = -40°C to 85°C, 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 High Voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input High 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		0.45	V
V_{OL1}	Output Low Voltage[1] (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
V_{OH}	Output High 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 High 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, VCC = 5V ± 10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pulldown 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		20	mA
		Idle Mode, 12 MHz		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.

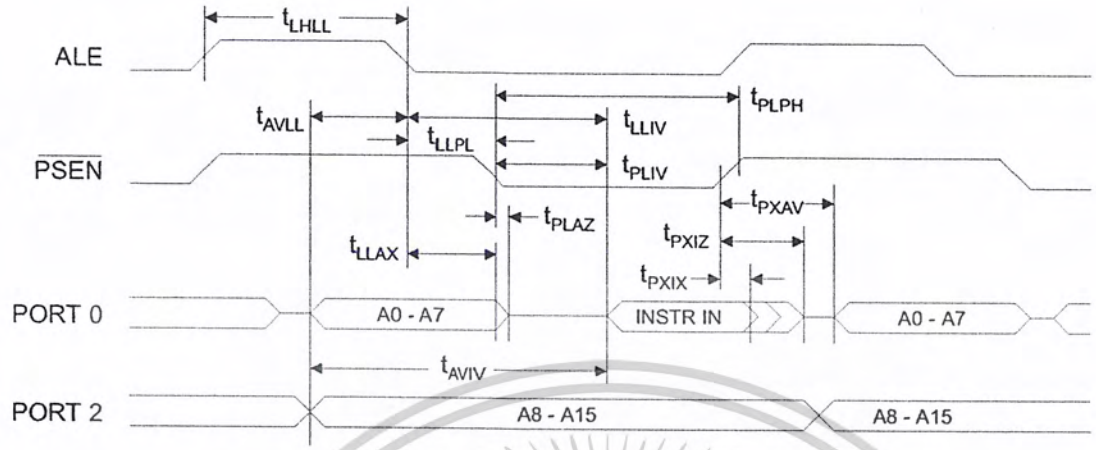
AC Characteristics

(Under Operating Conditions; Load Capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; Load Capacitance for all other outputs = 80 pF)

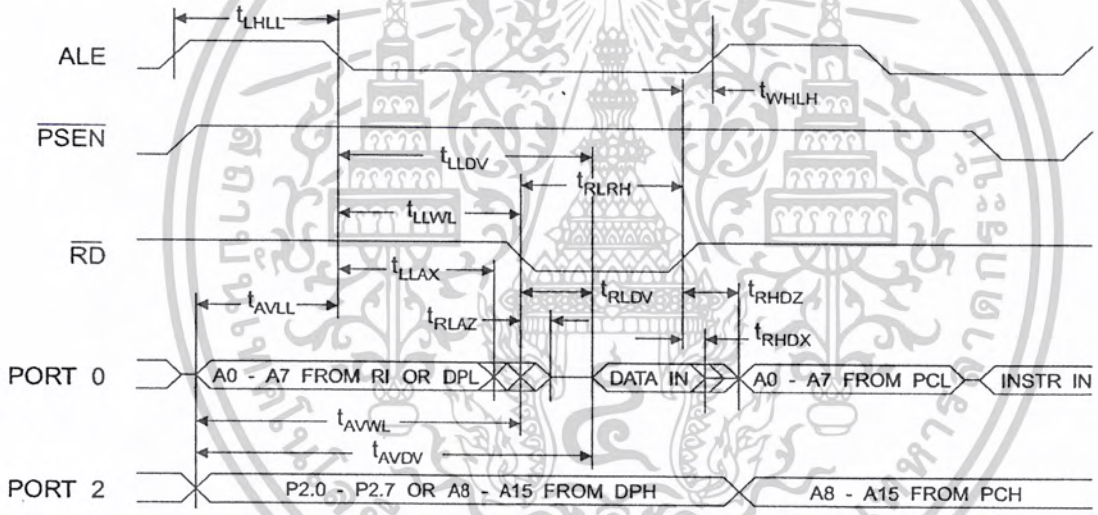
External Program and Data Memory Characteristics

Symbol	Parameter	12 MHz Oscillator		16 to 24 MHz Oscillator		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency			0	24	MHz
t_{LHLL}	ALE Pulse Width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	Address Valid to ALE Low	43		$t_{CLCL}-13$		ns
t_{LLAX}	Address Hold After ALE Low	48		$t_{CLCL}-20$		ns
t_{LLIV}	ALE Low to Valid Instruction In		233		$4t_{CLCL}-65$	ns
t_{LLPL}	ALE Low to $\overline{PSEN}$ Low	43		$t_{CLCL}-13$		ns
t_{PLPH}	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}-20$		ns
t_{PLIV}	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}-45$	ns
t_{PXIX}	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
t_{PXIZ}	Input Instruction Float After $\overline{PSEN}$		59		$t_{CLCL}-10$	ns
t_{PXAV}	$\overline{PSEN}$ to Address Valid	75		$t_{CLCL}-8$		ns
t_{AVIV}	Address to Valid Instruction In		312		$5t_{CLCL}-55$	ns
t_{PLAZ}	$\overline{PSEN}$ Low to Address Float		10		10	ns
t_{RLRH}	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}-90$	ns
t_{RHDX}	Data Hold After $\overline{RD}$	0		0		ns
t_{RHDZ}	Data Float After $\overline{RD}$		97		$2t_{CLCL}-28$	ns
t_{LLDV}	ALE Low to Valid Data In		517		$8t_{CLCL}-150$	ns
t_{AVDV}	Address to Valid Data In		585		$9t_{CLCL}-165$	ns
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}-75$		ns
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	23		$t_{CLCL}-20$		ns
t_{QVWH}	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}-120$		ns
t_{WHQX}	Data Hold After $\overline{WR}$	33		$t_{CLCL}-20$		ns
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0		0	ns
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	$t_{CLCL}-20$	$t_{CLCL}+25$	ns

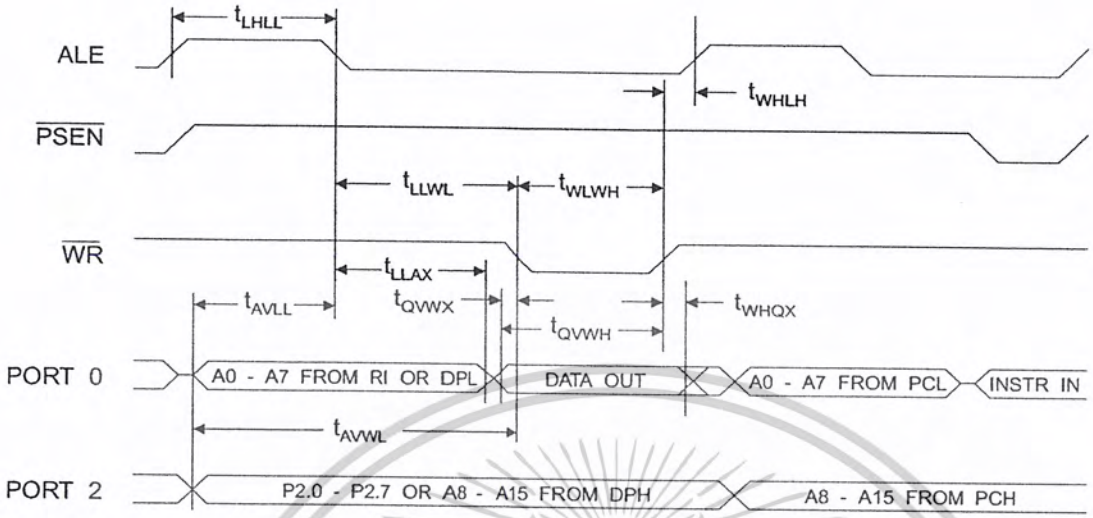
External Program Memory Read Cycle



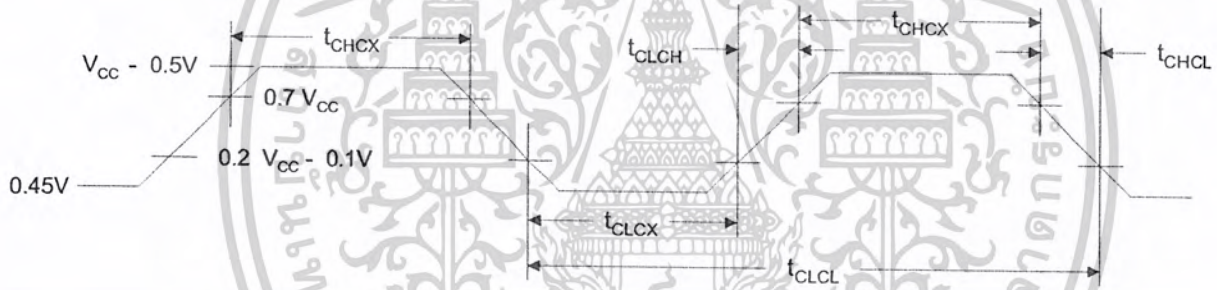
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

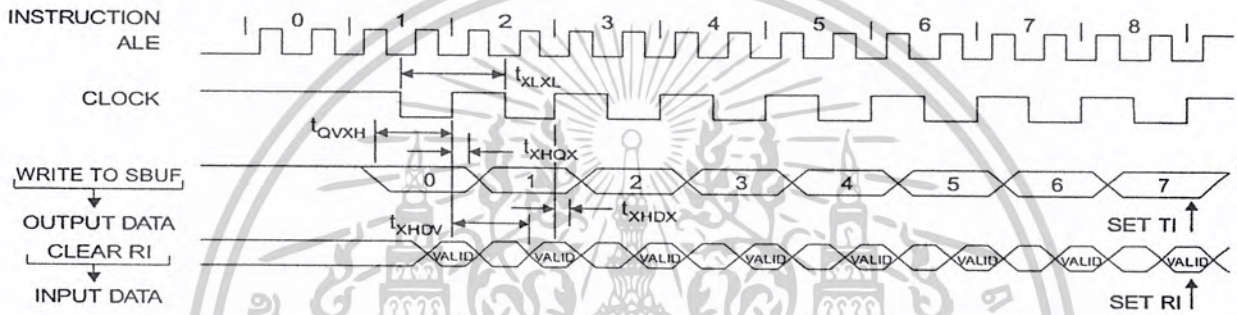
Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	24	MHz
t_{CLCL}	Clock Period	41.6		ns
t_{CHCX}	High Time	15		ns
t_{CLCX}	Low Time	15		ns
t_{CLCH}	Rise Time		20	ns
t_{CHCL}	Fall Time		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

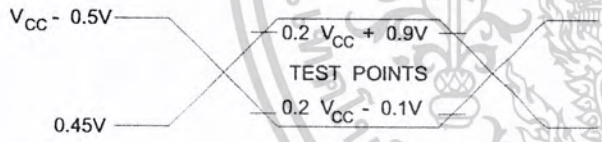
(V_{CC} = 5.0 V ± 20%; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-117		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		$10t_{CLCL}$-133	ns

Shift Register Mode Timing Waveforms

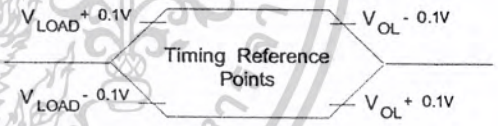


AC Testing Input/Output Waveforms(1)



Note: 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)



Note: 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 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	5V ± 20%	AT89C51-12AC	44A	Commercial (0°C to 70°C)
		AT89C51-12JC	44J	
		AT89C51-12PC	40P6	
		AT89C51-12QC	44Q	
		AT89C51-12AI	44A	Industrial (-40°C to 85°C)
		AT89C51-12JI	44J	
		AT89C51-12PI	40P6	
		AT89C51-12QI	44Q	
		AT89C51-12AA	44A	Automotive (-40°C to 105°C)
		AT89C51-12JA	44J	
		AT89C51-12PA	40P6	
		AT89C51-12QA	44Q	
16	5V ± 20%	AT89C51-16AC	44A	Commercial (0°C to 70°C)
		AT89C51-16JC	44J	
		AT89C51-16PC	40P6	
		AT89C51-16QC	44Q	
		AT89C51-16AI	44A	Industrial (-40°C to 85°C)
		AT89C51-16JI	44J	
		AT89C51-16PI	40P6	
		AT89C51-16QI	44Q	
		AT89C51-16AA	44A	Automotive (-40°C to 105°C)
		AT89C51-16JA	44J	
		AT89C51-16PA	40P6	
		AT89C51-16QA	44Q	
20	5V ± 20%	AT89C51-20AC	44A	Commercial (0°C to 70°C)
		AT89C51-20JC	44J	
		AT89C51-20PC	40P6	
		AT89C51-20QC	44Q	
		AT89C51-20AI	44A	Industrial (-40°C to 85°C)
		AT89C51-20JI	44J	
		AT89C51-20PI	40P6	
		AT89C51-20QI	44Q	

หนังสืออ้างอิง

ชัยวัฒน์ ลิ้มพรจิตรวิไล และ วรพจน์ กรแก้ววัฒนกุล , เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 : บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด

เจน สงสมพันธุ์ , เทคโนโลยีอิเล็กทรอนิกส์ 1 . พิมพ์ครั้งที่ 13 : สถาบันอิเล็กทรอนิกส์กรุงเทพรังสิต , 2538