กาต้มน้ำอัตโนมัติด้วยไมโครคอนโทรลเลอร์

Automatic boiler by microcontroller

โดย

นายวินัย ลาลุน

นายอธิคม จีนเสวก

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ประจำภาคเรียนที่ 2 ปีการศึกษา 2545

กาต้มน้ำอัตโนมัติด้วยไมโครคอนโทรลเลอร์

Automatic boiler by microcontroller

โดย

นายวินัย ลาลุน 43515931

นายอธิคม จีนเสวก 43515944

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ดอกเตอร์มนัส สังวรศิลป์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ประจำภาคเรียนที่ 2 ปีการศึกษา 2545

แบบฟอร์มรับรองความพร้อมในการสอบ

กาต้มน้ำอัตโนมัติด้วยไมโครคอนโทรลเลอร์

Automatic boiler by microcontroller

นายวินัย ลาลุน

นายอธิคม จีนเสวก

โครงการนี้ได้รับการตรวจสอบแล้วพร้อมที่จะทำการสอบได้

.....

(รศ.ดร.มนัส สังวรศิลป์)

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2545

กาต้มน้ำอัตโนมัติด้วยไมโครคอนโทรลเลอร์

นายวินัย ลาลุน
นายอธิคม จีนเสวก

อาจารย์ที่ปรึกษา
รศ.ดร.มนัส สังวรศิลป์
ปีการศึกษา 2545

บทคัดย่อ

รายงานเล่มนี้ได้นำเสนอการออกแบบเครื่องต้มน้ำร้อนอัตโนมัติด้วยไมโครคอนโทรลเลอร์ตระกูล MCS-51 โดยใช้โปรแกรม flash-x ออกแบบและวิเคราะห์ นอกจากนี้ส่วนสำคัญก็คือ ไอซีตรวจจับอุณหภูมิ DS1820 ซึ่งใช้สำหรับควบคุมอุณหภูมิได้ตามต้องการ โดยนำการแสดงผลออกทางหน้าจอ LCD และอีกส่วนหนึ่งของเครื่องต้มน้ำอัตโนมัติยังสามารถทำการตรวจวัดระดับของน้ำได้ด้วยกันถึง 3 ระดับ ให้เลือกใช้งานตามต้องการอีกด้วย และยังได้ทำการพัฒนาเครื่องต้อมน้ำนี้ให้สามารถทำการปั๊มน้ำเข้าได้โดยการควบคุมแบบอัตโนมัติ อีกทั้งยังสามารถใช้คอมพิวเตอร์เป็นตัวควบคุมการทำงานทั้งหมดของเครื่องต้มน้ำ โดยใช้ Program Visual Basic ทำการควบคุม และแสดงอุณหภูมิและขั้นตอนการทำงานผ่านหน้าจอคอมพิวเตอร์ ซึ่งสั่งการทำงานโดยใช้คีย์บอร์ดเป็นตัวควบคุม

Automatic boiler by microcontroller

Mr.Winai Lalun

Mr.Atikom Jeensavake

Assoe-Prof.Manas Sangworasil (Advisor)

Academic Year 2002

Abstract

This report presents the design of automatic boiler by microcontroller sensor MCS-51. This work uses flaxh-x for design and analysis.

Then the important IC DS1820 is used for sensoring temperatrue and control the temperature in requirement. The result is present on LCD monitor and the other part of this automatic boiler can the aamount of water in there levels that the user can choose. This design can be improved to obtair.The result to Automatics pump in the boiler and can used by Personal computer in control the boiler.This work uses Visual Basic program for control such as presented step of work and temperature on moniter.So that control by key board.

กิตติกรรมประกาศ

รายงานเล่มนี้สำเร็จลุล่วงไปได้ด้วยดีเพราะได้รับคำปรึกษา ข้อเสนอแนะ จากอาจารย์ที่ปรึกษา รศ.ดร.มนัส สังวรศิลป์ ขอบคุณห้องทำงาน ณ ตึกกรมหลวงนราธิวาส และขอบคุณรุ่นพี่ปริญญาโททุก ๆ ท่าน ที่ให้คำแนะนำ และเอื้อเฟื้ออุปกรณ์การทำงานด้วยดีมาโดยตลอด ไว้ ณ โอกาสนี้

วินัย ลาลุน

อธิคม จีนเสวก

สารบัญ

สารบัญ (ต่อ)

บทที่ 1
บทนำ

ความเป็นมาของโครงการ

ในปัจจุบันนี้ ไมโครคอนโทรลเลอร์ได้เข้ามามีบทบาทอย่างกว้างขวาง เนื่องจากในการทำงานของอุปกรณ์เครื่องใช้ไฟฟ้า และอิเล็กทรอนิกส์ต่าง ๆ ได้นำเอาเทคโนโลยีของ Microcontroller เข้ามาทำการควบคุม โดยได้มีการประยุกต์ใช้งานมาเป็นเวลาหลายสิบปีแล้ว ซึ่งส่วนใหญ่จะนำไปใช้ในการควบคุมระบบหรืออุปกรณ์ที่ต้องการความละเอียด และความเร็วในการทำงานสูง หรือในระบบที่มีความซับซ้อนมาก ทั้งนี้เนื่องจากตัวไมโครโปรเซสเซอร์ มีความสามารถในการรับข้อมูล ที่ถูกแปลงเป็นสัญญาณดิจิตอลมาประมวลผลตามลำดับคำสั่งของโปรแกรมอีกทั้งยังสามารถทำการเปลี่ยนแปลงแก้ไขระบบก็สามารถทำได้ง่าย เพียงแต่ปรับปรุงแก้ไขโปรแกรมเดิมหรือเปลี่ยนโปรแกรมใหม่ โดยไม่ต้องเปลี่ยนแปลงวงจร หรือ อุปกรณ์อื่น ๆ มากนัก ในที่นี้การออกแบบโปรแกรม Microcontroller ให้สามารถนำไปควบคุมการทำงานของ กาต้มน้ำร้อนอัตโนมัติได้ไม่ว่าจะเป็นการควบคุม อุณหภูมิ การแบ่งวัดระดับน้ำ และการ เปิด-ปิด ให้น้ำเข้าออกได้อัตโนมัติ อีกทั้งยังใช้ระบบคอมพิวเตอร์เข้ามาทำการควบคุมในส่วนของการทำงานทั้งหมด และยังให้แสดงผลของการปฏิบัติงานผ่านทางหน้าจอคอมพิวเตอร์อย่างเป็นขั้นตอน เพื่อที่จะนำไปพัฒนา และแก้ไขให้สมบูรณ์แบบมากขึ้นต่อไปในอนาคต

วัตถุประสงค์ของโครงการ

1. ศึกษาขั้นตอนการใช้งานอุปกรณ์ IC ควบคุมอุณหภูมิ เบอร์ DS1820
2. ศึกษาการเขียนโปรแกรมด้วย flash-x Microcontroller ควบคุมอุณหภูมิ
3. ศึกษาหลักการของการแบ่งวัดระดับน้ำรวมถึงการปั๊มน้ำเข้าด้วยการออกแบบและเขียนโปรแกรมให้สามารถควบคุมระดับน้ำได้ตามต้องการ
4. ศึกษาโปรแกรมในส่วนของ Visual Basic ให้สามาถสั่งการควบคุมการทำงานผ่านระบบคอมพิวเตอร์
5. ศึกษาหลักการของพอร์ตอนุกรมที่ใช้ทำหน้าที่เชื่อมต่อระหว่าง Microcontroller และ Computer

ขอบเขตของโครงการ

เป็นการออกแบบโปรแกรมควบคุมและตรวจจับอุณหภูมิของกาน้ำร้อนอัตโนมัติ โดยใช้ IC DS1820 เป็นตัวตรวจจับและควบคุมอุณหภูมิ โดยการทำการเขียนโปรแกรมด้วย flash-x Microcontroller เพื่อให้โปรแกรมที่เขียนขึ้นมาสามารถควบคุม กำหนด และตรวจจับอุณหภูมิได้ตามความเป็นจริง เมื่อเราสามารถควบคุมอุณหภูมิได้ตามต้องการแล้ว จากนั้นเราจะทำการแบ่งระดับน้ำออกเป็นระดับ ๆ เพื่อให้สะดวกในการใช้งาน เมื่อต้องการน้ำในปริมาณที่มากหรือน้อยตามต้องการได้ และเราก็จะเป็นต้องใช้การเขียนโปรแกรมที่มาเป็นตัวควบคุมระดับน้ำด้วยเช่นกัน นอกจากนี้ค่าของอุณหภูมิที่เราจะควบคุมแล้วนั้นเราจะทราบได้โดยหน้าจอแสดงผล LCD

ขีดความสามารถของโครงการ

1. สามาถทำการควบคุมอุณหภูมิของเครื่องต้มน้ำได้ตามต้องการ
2. สามารถแบ่งระดับน้ำที่ต้องการใช้งานในการต้มได้ 3 ระดับ ตามความประสงค์ที่จะใช้งาน
3. สามารถแสดงผลของอุณหภูมีที่ได้และระดับน้ำที่เราต้องการออกมายังจอแสดงผล(LCD)
4. สามารถปั๊มน้ำเข้าโดยอัตโนมัติตามระดับที่เราต้องการ
5. สามารถควบคุมด้วยคอมพิวเตอร์
6. สามารถแสดงผลออกทางหน้าจอคอมพิวเตอร์และใช้คีย์บอร์ดเป็นตัวควบคุม

ประโยชน์ที่ได้รับ

1. ได้รู้ถึงวิธีการใช้งานโปรแกรม Micorcontoller
2. ได้รู้ถึงหลักการออกแบบและเขียนโปรแกรมควบคุมอุณหภูมิ
3. ได้รู้ถึงปัญหาและขั้นตอนในการปฏิบัติงานกับอุปกรณ์ที่ใช้จริง
4. ได้รู้ถึงการใช้งานในส่วนของการแสดงผลทาง LCD
5. ได้รู้ถึงการออกแบบการแบ่งระดับน้ำโดยใช้เครื่องปั๊มน้ำเข้าโดยอัตโนมัติ
6. ได้รู้ถึงการออกแบบการเขียนโปรแกรม Visual Basic ให้สามารถเชื่อมต่อเข้ากับเครื่องคอมพิวเตอร์ได้
7. ได้รู้ถึงการสร้างและออกแบบวงจรพอร์ตอนุกรมเพื่อใช้ในการเชื่อมต่อระบบเข้าด้วยกัน

บทที่ 2
โครงสร้างและสถาปัตยกรรมของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ที่ใช้จะอ้างอิงถึงไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งมีหน่วยความจำภายในเป็นแบบแฟลช (flash memory) ของ Atmel Corporation มีเบอร์ขึ้นต้นด้วย AT89 เหตุผลที่ใช้ไมโครคอนโทรลเลอร์แบบนี้ในการเรียนรู้เพื่อใช้งานไมโครคอนโทรลเลอร์ ตระกูล MCS-51 มีด้วยกันหลายประการดังนี้

1. หน่วยความจำโปรแกรมภายในตัวไมโครคอนโทลเลอร์เป็นแบบแฟลช ทำให้สามารถลบและเขียนใหม่ได้นับพันครั้ง จึงสามารถใช้งานในรูปแบบของไมโครคอนโทรลเลอร์ชิปเดี่ยวไม่ต้องใช้หน่วยความจำภายนอก ส่งผลให้สามารถใช้งานพอร์ตอินพุตเอาต์พุตของไมโครคอนโทรลเลอร์ได้อย่างเต็มประสิทธิภาพ

2. ต้นทุนและเวลาในการพัฒนาระบบไมโครคอนโทรลเลอร์ลดลงอย่างมาก เนื่องจากไม่ต้องใช้เครื่องมือพัฒนาจำพวกอีมูลเตอร์และเครื่องโปรแกรมอีพรอม

3. บริษัทผู้ผลิตได้ทำการผลิตไมโครคอนโทรลเลอร์ตระกูลนี้ออกมาหลายเบอร์ และมีความสามารถแตกต่างกันไป ทำให้มีทางเลือกในการใช้งานสูง

4. ด้วยการใช้หน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ ทำให้สามารถป้องกันการคัดลอกข้อมูลของหน่วยความจำโปรแกรมได้เป็นอย่างดี

5. ในบางเบอร์ของไมโครคอนโทรลเลอร์ที่ผลิตโดย Atmel สามารถทำการโปรแกรมข้อมูลในหน่วยความจำโปรแกรมได้โดยที่ไม่ต้องถอดตัวไมโครคอนโทรลเลอร์ออกมาทำการโปรแกรมใหม่ หรือเรียกว่า การโปรแกรมในวงจร หรือ ในระบบ (In-system programming) ทำให้การซ่อมบำรุง ตลอดจนการปรับปรุงหรืออัปเกรดข้อมูลในหน่วยความจำโปรแกรมทำได้อย่างสะดวก ภายใต้งบประมาณที่ไม่สูงมากนัก

6. ชุดคำสั่งและสถาปัตยกรรมพื้นฐานเหมือนกับไมโครคอนโทรลเลอร์ MCS-51 ของผู้ผลิตอื่น ไม่ว่าจะเป็นอินเทล, ซีเมนส์ หรือ ดัลลัส

คุณสมบัติของไมโครคอนโทรลเลอร์ ตระกูล MCS-51 อนุกรม AT89xx

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

ในรูปที่ 1-1 เป็นโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx จะเห็นได้ว่าโครงสร้างของ AT89Cxx จะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐาน หากแต่แตกต่างกันเฉพาะหน่วยความจำโปรแกรมแบบแฟลชที่เพิ่มเติมเข้ามา หากเป็นไมโครคอนโทรลเลอร์ในอนุกรม 87xx หน่วยความจำโปรแกรมภายในจะเป็นแบบอีพรอม และบางเอร์สามารถโปรแกรมได้เพียงครั้งเดียว

สำหรับในรูปที่ 2.1 เป็นโครงสร้างพื้นฐานของอนุกรม AT89Sxx จะเห็นได้ว่า มีส่วนประกอบที่เพิ่มเติมแตกต่างจาก AT89Cxx อยู่หลายส่วน อาทิ วงจรเชื่อมต่ออนุกรมแบบ SPI ซึ่งในไมโครคอนโทรลเลอร์อนุกรมนี้ใช้ในการเขียนข้อมูลลงในหน่วยความจำโปรแกรมโดยไม่ต้องถอดตัวชิปออกไปจากระบบหรือเรียกว่าการโปรแกรมในวงจร ไทเมอร์/เคาน์เตอร์ขนาด 16 บิตที่เพิ่มเติมเข้ามาอีกหนึ่งตัวเป็นไทเมอร์ 2 และวงจรวอตช์ด็อกที่ใช้ในการตรวจสอบการทำงานผิดพลาดของซีพียู

ในตารางที่ 2.1 แสดงรายละเอียดบางส่วนของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แต่ละเบอร์ที่ Atmel ผลิตขึ้น และมีใช้งานอยู่ในปัจจุบัน

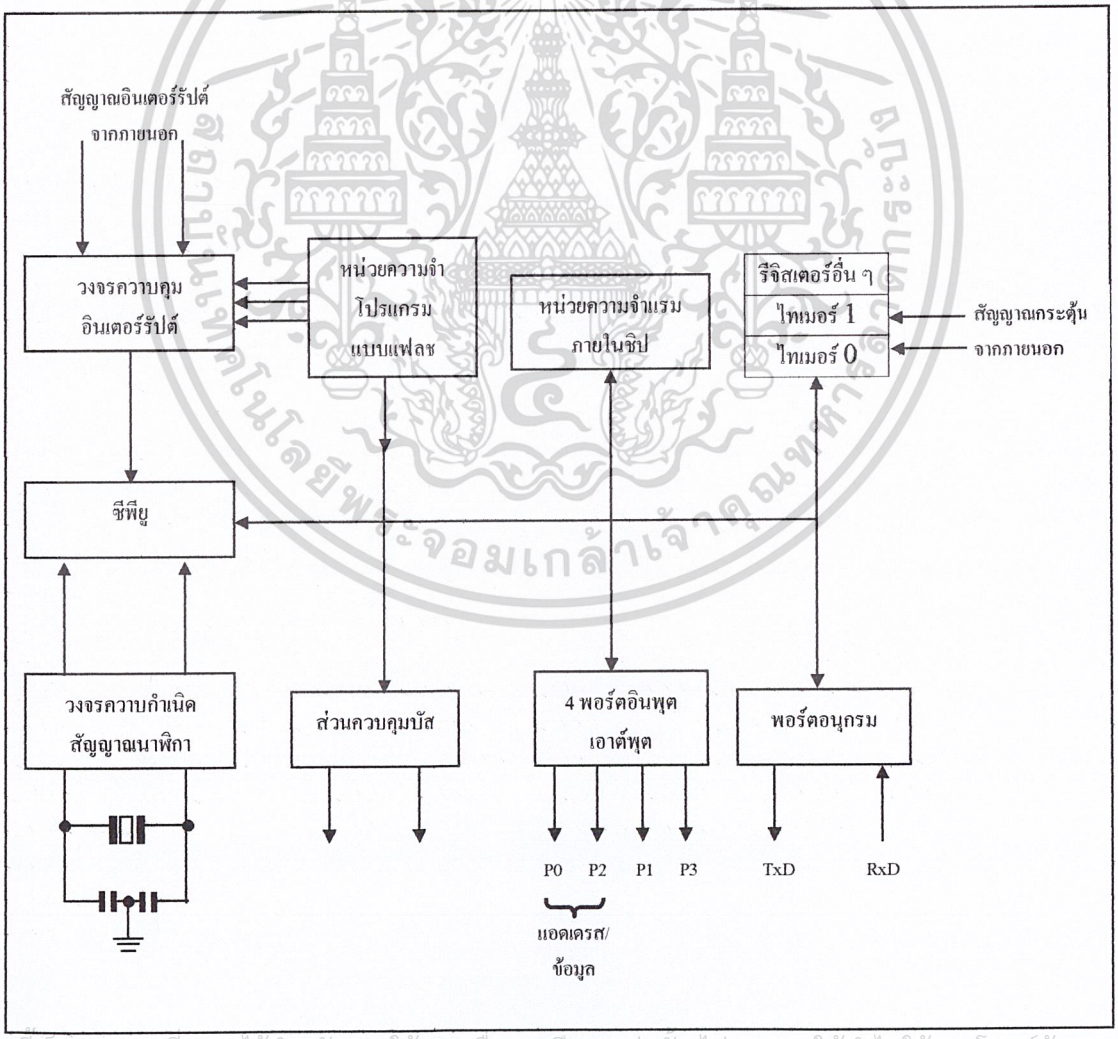
การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกัน ดังแสดงในรูปที่ 2.3 และ 2.4 โดยมีรายละเอียดขั้นต้น ดังนี้

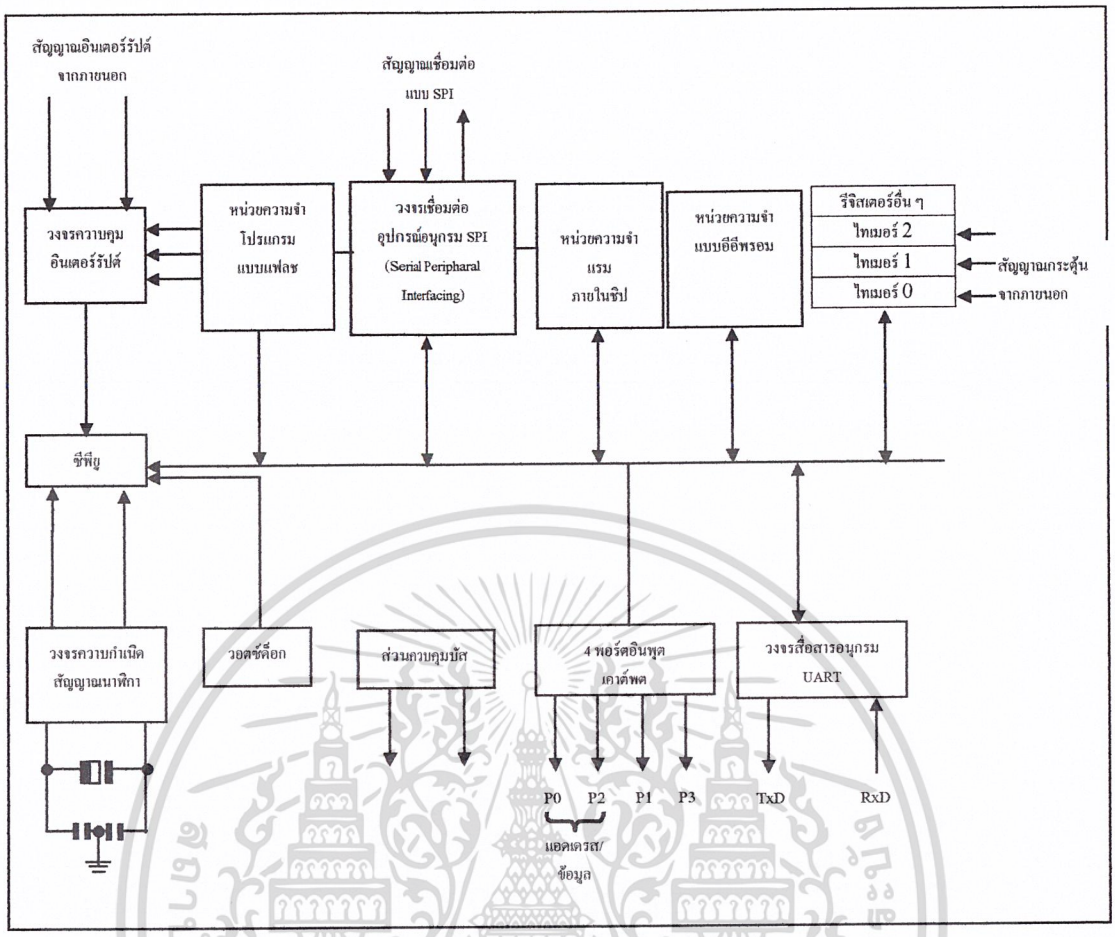
ขา Vcc ใช้สำหรับต่อไฟเลี้ยง +5V

ขา GND เป็นขากราวด์ สำหรับต่อกับกราวด์ของระบบ

ขาพอร์ต 0 (P0.0-P0.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูงสามาถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล(D0-D7) โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วยเพื่อสลับการทำงานเป็นได้ทั้งขาติดต่อแอดเดรสและขามูล



รูปที่ 2.1 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89Cxx



รูปที่ 2.2 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชในอนุกรม AT89Cxx

เบอร์ของไมโครคอนโทรลเลอร์	หน่วยความจำโปรแกรม	หน่วยความจำข้อมูล	จำนวนไทเมอร์/เคาน์เตอร์ 16 บิต
AT89C1051	แบบแฟลช ขนาด 1 กิโลไบต์	แรม 64 ไบต์	1
AT89C2051	แบบแฟลช ขนาด 2 กิโลไบต์	แรม 128 ไบต์	2
AT89C51	แบบแฟลช ขนาด 4 กิโลไบต์	แรม 128 ไบต์	2
AT89C52	แบบแฟลช ขนาด 8 กิโลไบต์	แรม 256 ไบต์	3
AT89C55	แบบแฟลช ขนาด 20 กิโลไบต์	แรม 256 ไบต์	3
AT89S8252	แบบแฟลช ขนาด 8 กิโลไบต์	แรม 256 ไบต์ อีอีพรอม 2 กิโลไบต์	3
AT89S53	แบบแฟลช ขนาด 12 กิโลไบต์	แรม 256 ไบต์	3

ตารางที่ 2.1 รายละเอียดโดยสรุปบางส่วนของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชที่ Atmel ผลิตขึ้น

ขาพอร์ต 1 (P1.0-P1.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล "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 (P2.0P2.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15)

ขาพอร์ต 3 (P3.0-P3.7) มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ต 3 ยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ดังมีรายละเอียดขั้นต้นต่อ ไปนี้

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

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

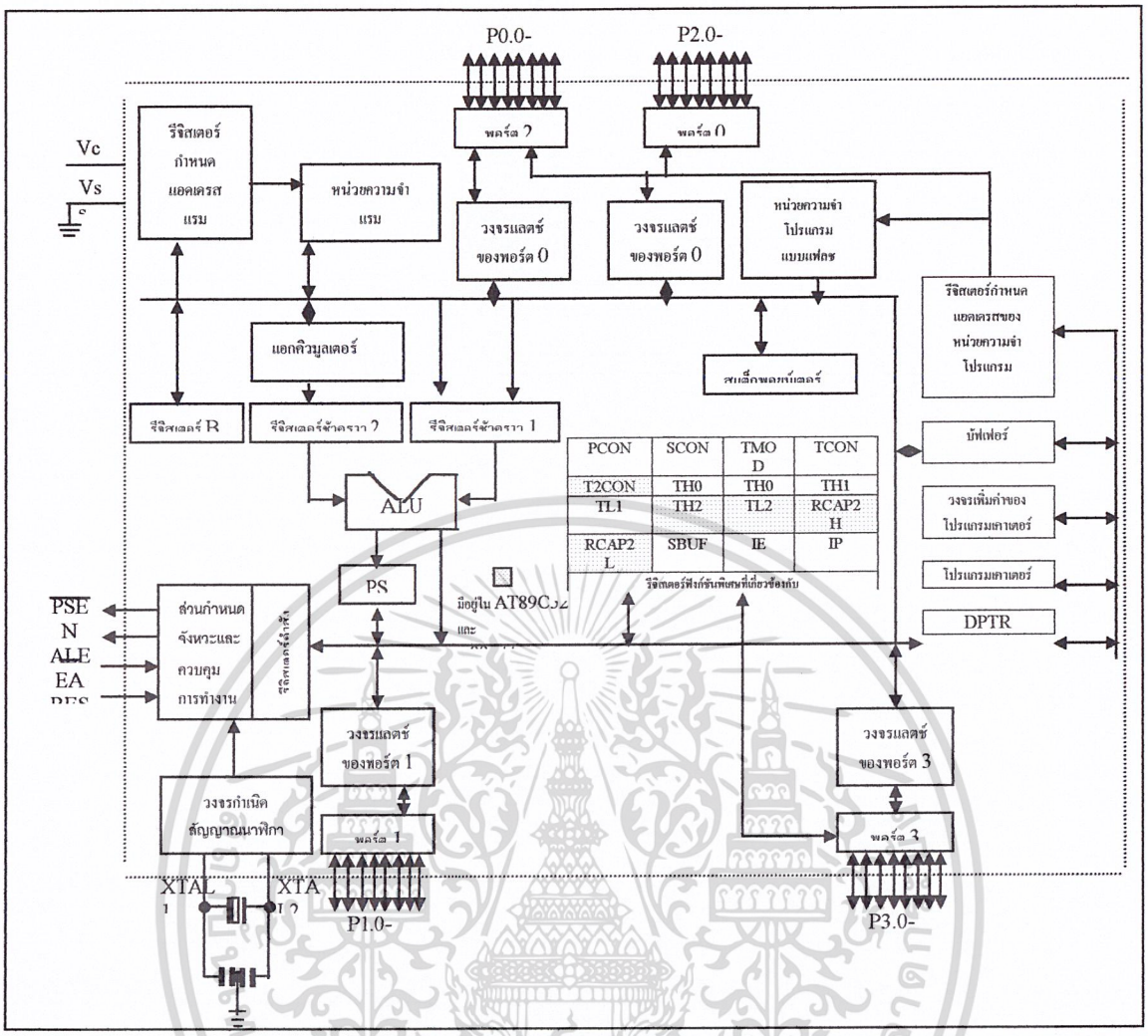
P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา $\overline{\text{INT1}}$

P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 0 หรือขา T0

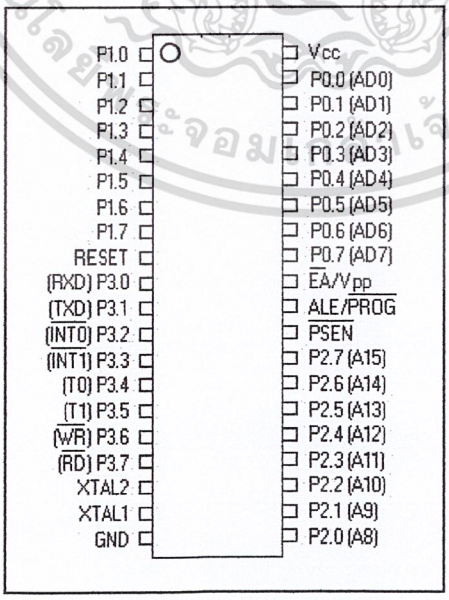
P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปต์จากภายนอกช่อง 1 หรือขา T1

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

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



รูปที่ 2.3 รายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชของ Atmel



รูปที่ 2.4 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89C5x

ขารีเซต (Reset) ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซตสานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชีน ไซเกิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างปกติ

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

ขา $\overline{\text{PSEN}}$ (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก ตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้งในแต่ละแมชีน แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอก ขานี้จะไม่มีการส่งสัญญาณใด ๆ ออกมา

ขา $\overline{\text{EA}}$/App (External Access enable/Programming voltage input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็นฯ “0” เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ถ้าหากขานี้เป็น “1” เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ ที่ขานี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูงสำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบปฟลชต้องการแรงดันสำหรับการโปรแกรม +12V

ขา XTAL1 และ XTAL2 เป็นขาสำหรับต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

โครงสร้างและการทำงานของพอร์ต

ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ตคือ พอร์ต 0 ถึง พอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทาง กล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับรับสัญญาณข้อมูลเข้าและเอาต์พุตสำหรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีวงแลตช์และวงจรขับตลอดจนบัฟเฟอร์อินพุต ดังแสดงให้เห็นในสถาปัตยกรรมรูปที่ 2.3

ที่พอร์ต 0 และพอร์ต 2 จะใช้งานเป็นพอร์ตอินพุตและเอาต์พุตสำหรับงานทั่วไป และใช้ในการติดต่อกับหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ตและพอร์ต 1 บางขานอกจากจะใช้เป็นขาพอร์ตอินพุตเอาต์พุตตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก ขึ้นอยู่กับว่าเป็นไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ใด ดังสรุปได้ในตารางที่ 2.2

ในรูปที่ 2.5 แสดงวงจรภายในของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดยในรูปที่ 2.5 (ก) เป็นวงจรของพอร์ต 0 วงจรแลตช์ของแต่ละบิตในแต่ละพอร์ตก็คือวงจรดีฟลิฟลอปนั่นเอง การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตช์สามารถกระทำได้อย่างอิสระด้วยสัญญาณที่แยกจากกัน นั่นคือ สัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูลจากวงจรแลตช์ ส่วนการเขียนข้อมูลมายังพอร์ตต้องส่งสัญญาณมายังขา CLK ของดีฟลิปฟลอปในขณะที่ข้อมูลจะผ่านมาทางขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิฟลอป

ที่พอร์ตนี้มีวงจรมัลติเพล็กซ์สำหรับกำหนดลักษณะการทำงานของพอร์ตว่า ต้องการใช้งานเป็นขาพอร์ตอินพุตเอาต์พุตปกติหรือใช้ในการติดต่อกับหน่วยความจำภายนอกไมโครคอนโทรลเลอร์

เนื่องจากที่ขาพอร์ต 0 ไม่มีวงจรพูลอัปภายใน หากมีการนำพอร์ต 0ไปใช้งานเป็นพอร์ตอินพุตจะต้องต่อตัวต้านทางพูลอัปภายนอกเข้าที่ขาพอร์ 0 ทุกขาด้วย

ในรูปที่ 2.5 (ข) เป็นวงจรของพอร์ต 1 ซึ่งมีลักษณะโดยทั่วไปคล้ายกับพอร์ต 0 หากแต่ไม่มีวงจรมัลติเพล็กซ์ เนื่องจากพอร์ตนี้จะไม่ใช้ในการติดต่อกับหน่วยความจำภายนอก แต่จะมีวงจรพูลอัปภายในที่แต่ละบิตของพอร์ตนี้แทน สำหรับรายละเอียดของวงจรพูลอัปแสดงในรูปที่ 2.6

ในรูปที่ 2.5 (ค) เป็นวงจรภายในของพอร์ต 2 จะคล้ายกับพอร์ต 0 มาก ต่างกันเพียงมีวงจรพูลอัปเพิ่มเติมเข้ามา ส่วนในรูปที่ 2.5 (ง) เป็นวงจรภายในของพอร์ต 3 จะเห็นได้ว่าคล้ายกับพอร์ต 1 มีการเพิ่มเติมวงจรบัฟเฟอร์และวงจรอินพุตเอาต์พุตเมื่อทำงานในฟังก์ชั่นพิเศษเข้ามา เนื่องจากพอร์ต 3 สามารถนำไปใช้งานในหน้าที่พิเศษได้ทุกขา

การใช้งานเป็นพอร์ตอินพุต

เนื่องจากพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเป็นได้ทั้งอินพุตและเอาต์พุต ดังนั้นจึงมีความจำเป็นอย่างยิ่งต้องทำความเข้าใจถึงการกำหนดลักษณะการทำงานให้แก่พอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในการกำหนดให้เป็นพอร์ตอินพุต ต้องเริ่มต้นด้วยการเขียนข้อมูล “1” มาที่แต่ละบิตของพอร์ตที่ต้องการใช้งานเป็นอินพุต เพื่อหยุดการทำงานของเฟตที่ใช้ในการขับสัญญาณเอาต์พุตของบิตนั้น ๆ ทำให้ขาสัญญาณของพอร์ตเชื่อมต่อเข้ากับวงจรพูลอัปภายในโดยตรงส่งผลให้ขาพอร์ตนั้นมีลอจิกเป็น “1” สามารถรับสัญญาณลอจิก “0” จากอุปกรณ์ภายนอกได้ง่าย สัญญาณข้อมูลจากอุปกรณ์ภายนอกจะถูกส่งเข้ามาแล้วเก็บไว้ในวงจรบัฟเฟอร์ภายในพอร์ต แล้วรอให้ซีพียูมาอ่านค่าเข้าไป เมื่อเป็นเช่นนี้ อุปกรณ์ภายนอกที่เชื่อมต่อกับพอร์ตอินพุตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชควรกำหนดให้ทำงานในสภาวะลอจิก “0” จะดีและสะดวกที่สุด (ซึ่งในปัจจุบันอุปกรณ์อินพุตที่เชื่อมต่อไมโครคอนโทรลเลอร์แทบทั้งหมดทำงานที่ลอจิก “0” แล้ว)

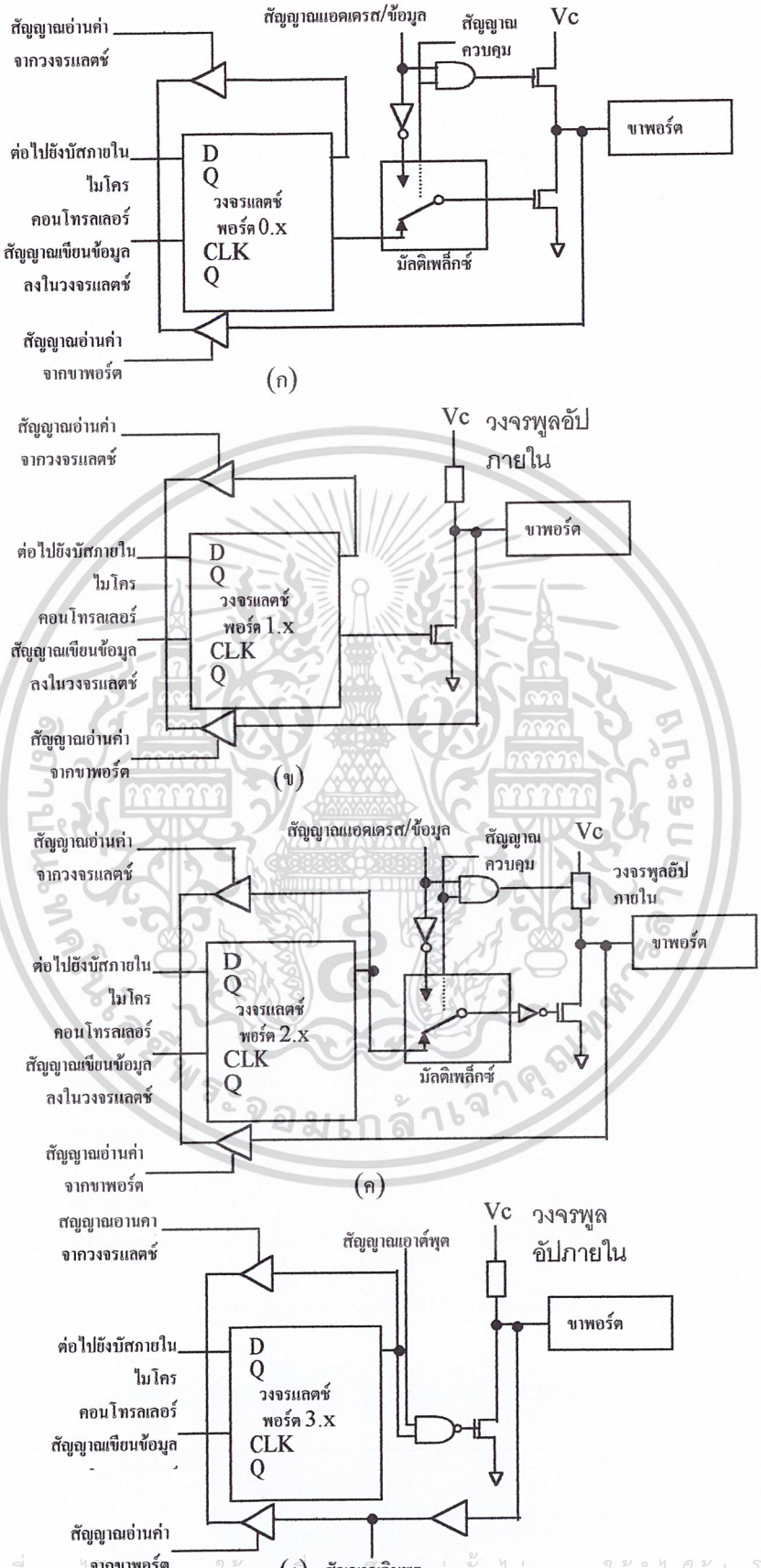
ขา	เบอร์ของไมโครคอนโทรลเลอร์	หน้าที่พิเศษ
P1.0	AT89C52/AT89Sxx	ขา T2 เป็นขาอินพุตนับค่าของไทเมอร์/เคาน์เตอร์ 2 และเป็นขา
P1.1	AT89C52/AT89Sxx	และควบคุมทิศทางของสัญญาณ
P1.4	AT89Sxx	ขา SS (Slave Select) เป็นขาเลือกการติดต่อในกรณีที่ไมโครคอนโทรลเลอร์เป็นอุปกรณ์สเลฟ ในระบบการติดต่อแบบ SPI
P1.5	AT89Sxx	ขา MOSI (Master data output, Slave data input) ใช้ในการติดต่อกับพอร์ต SPI
P1.6	AT89Sxx	ขา MISO (Master data input, Slave data output) ใช้ในการติดต่อกับพอร์ต SPI
P1.7	AT89Sxx	ขา SCK (Master clock output) เป็นขาสัญญาณนาฬิกาของการติดต่อกับพอร์ต SPI

ตารางที่ 2.2 หน้าที่พิเศษของพอร์ต 1 ในไมโครคอนโทลเลอร์ MCS-51 แบบแฟลชของ Atmel

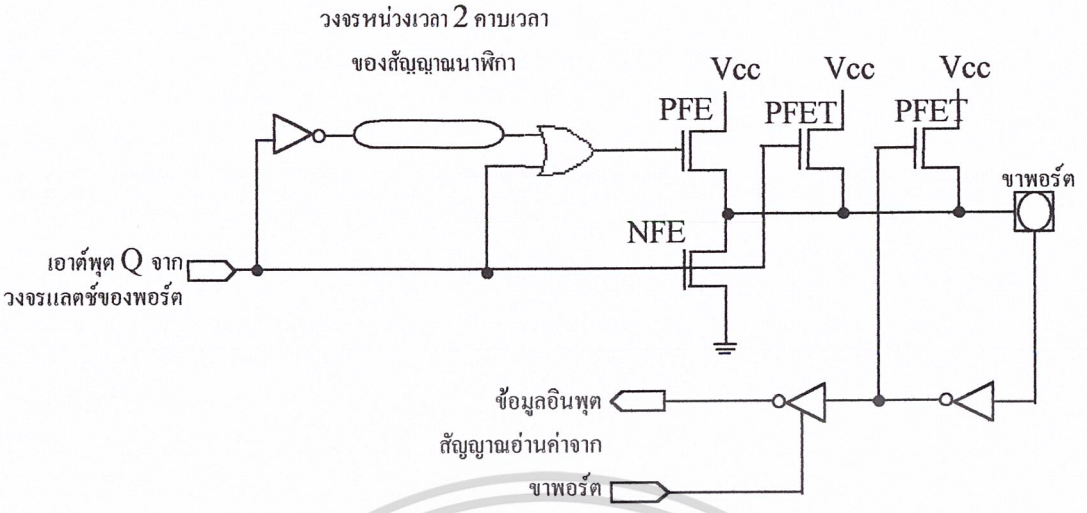
การใช้งานเป็นพอร์ตเอาต์พุต

โดยปกติแล้ว ขาพอร์ตจะกำหนดให้มีลักษณะเป็นเอาต์พุตอยู่แล้ว ดังนั้นจึงสามารถส่งข้อมูลออกไปได้อย่างง่ายดายและตรงไปตรงมา กล่าวคือ เมื่อต้องการส่งข้อมูล “0” ออกไปทางเอาต์พุตก็ให้เขียนข้อมูล “0” ขึ้น ไปยังวงจรแลตช์ ซึ่งก็จะส่งต่อไปขับเฟต ทำให้เฟตทำงาน ที่ขาพอร์ตที่กำหนดให้ทำงานก็จะเกิดลอจิก “0” ขึ้น ในทางตรงข้ามหากต้องการส่งข้อมูล “1” ออกไป ก็ให้เขียนข้อมูล “1” ไปยังวงจรแลตช์ วงจรขับก็จะหยุดทำงาน ทำให้ที่ขาพอร์ตเชื่อมต่อกับวงจรพูลอัปภายในเกิดเป็นลอจิก “1” ที่ขาพอร์ตนั้น ซึ่งจะคล้ายกับการกำหนดให้เป็นขาอินพุตมาก เพียงแต่แตกต่างกันที่กระบวนการในการเคลื่อนย้ายข้อมูล โดยถ้าเป็นอินพุตจะมีสัญญาณมาอ่านข้อมูลที่บัฟเฟอร์ แต่ถ้าเป็นเอาต์พุตจะไม่มีการอ่านข้อมูลที่บัฟเฟอร์แต่อย่างใด เว้นแต่ในกรณีที่ต้องการตรวจสอบข้อมูลที่ส่งออกมาทางเอาต์พุต

เมื่อใช้งานเป็นพอร์ตเอาต์พุต แต่ละขา (หรือแต่ละบิต) ของแต่ละพอร์ตมีความสามารถในการจ่ายกระแสหรือที่เรียกว่า กระแสซอร์ส (source current) ได้สูงสุด 10 mA และทุกขารวมกันในแต่ละพอร์ต (ทั้ง 8 บิต) สูงสุด 26 mA สำหรับพอร์ต 0 และ 15 mA สำหรับพอร์ต 1-3 ในกรณีที่ใช้งานทุกพอร์ตเอาต์พุตจะสามาถจ่ายกระแสได้รวมกันสูงสุด 71 mA ดังนั้นในการใช้งานเป็นพอร์ตเอาต์พุตเพื่อไม่ให้เกิดปัญหาเกี่ยวกับความสามารถในการจ่ายกระสจึงควรต่อวงจรบัฟเฟอร์ทางเาต์พุตเพื่อช่วยในการขับกระแสอีกทางหนึ่ง



รูปที่ 2.5 วงจรภายในของพอร์ตทุกพอร์ตในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช



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

รูปที่ 2.6 วงจรพูลอัปภายในพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

การอ่านค่าลอจิกจากพอร์ต

ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถอ่านค่าลอจิกจากพอร์ตได้ 2 ลักษณะ คือ อ่านจากขาพอร์ตโดยตรง และอ่านจากวงจรแลตช์ของแต่ละพอร์ต

ในการณีที่พอร์ตต่อกับขาเบสทรานซิสเตอร์ชนิด NPN และขาอิมิตเตอร์ของทรานซิสเตอร์ตัวนั้นต่อลงกราวด์ หากมีการส่งข้อมูล "1" ไปยังทรานซิสเตอร์ จะทำให้ทรานซิสเตอร์ทำงาน สถานะลอจิกที่ขาพอร์ตจะเป็น "0" เนื่องจากเมื่อทรานซิสเตอร์ทำงาน จะเสมือนว่าขาพอร์ตนั้นถูกต่อลงกราวด์ ทำให้หากอ่านค่าลอจิกที่ขาพอร์ตจะได้ผลตรงข้ามกับที่ส่งออกมา แต่ถ้าหากทำงานอ่านค่าลอจิกที่วงจรแลตช์ จะได้ค่าที่ตรงกับค่าที่ต้องการส่งจริงดังนั้น ในการอ่านค่าลอจิกจากพอร์ตจึงต้องเลือกวิธีการให้เหมาะสมกับอุปกรณ์ที่นำมาต่อด้วย

จังหวะการทำงานของไมโครคอนโทรลเลอร์ MCS-51

ในการใช้งานไมโครคอนโทรลเลอร์ MCS-51 จะต้องทำความเข้าใจถึงการทำงานของซีพียู และลำดับขั้นตอนการประมวลผลคำสั่ง ในการประมวลผลคำสั่งของซีพียูจะมีขั้นตอนหลัก ๆ 2 ขั้นตอนคือ **กระบวนการเฟตช์ (fetch)** *เป็นการเรียกคำสั่งออกจากหน่วยความจำโปรแกรมแล้วทำการแปลงรหัสคำสั่งนั้นเป็นภาษาเครื่องเพื่อเตรียมการประมวลผล* ขั้นตอนต่อมาคือ **กระบวนการเอ็กซิคิวต์ (execute)** *เป็นการกระทำตามคำสั่งที่กำหนดหรือตามที่เฟตช์ขึ้นมา* โดยกระบวนการก่อนหน้านี้เมื่อทำการเอ็กซิคิวต์คำสั่งเรียบร้อยแล้วก็จะไปเริ่มกระบวนการเฟตช์คำสั่งใหม่ต่อไป

เมื่อเริ่มจ่ายไฟให้แก่ไมโครคอนโทรลเลอร์จะเกิดการรีเซตในลักษณะที่เรียกว่า เพาเวอร์ออนรีเซต (poweron reset) ซีพียูเริ่มต้นการทำงานที่แอดเดรส 0000H ของหน่วยความจำโปรแกรม จังหวะการทำงานของซีพียูจะเป็นไปตามรูปแบบ โดยได้รับการกำหนดมาจากรอบการทำงานหรือแมชีนไซเกิล (machine cycle) ในรูปที่ 2.7 เป็นไดอะแกรมเวลาแสดงจังหวะการทำงานของไมโครคอนโทรลเลอร์ MCS-51 โดยใน 1 รอบการทำงานหรือแมชีนไซเกิลจะแบ่งย่อยออกเป็น 6 สเตต (state) กำหนดชื่อเป็น S1-S6 ในแต่ละสเตตมีค่าเวลาเท่ากับ 2 คาบเวลาของสัญญาณนาฬิกา ถ้าสัญญาณนาฬิกามีความถี่ 12 MHz จะมีคาบเวลาเท่ากับ 1 ms คาบเวลาทั้งสองภายในหนึ่งสเตตจะเรียกว่า เฟส 1 (phase 1) และเฟส 2 (phase 2)

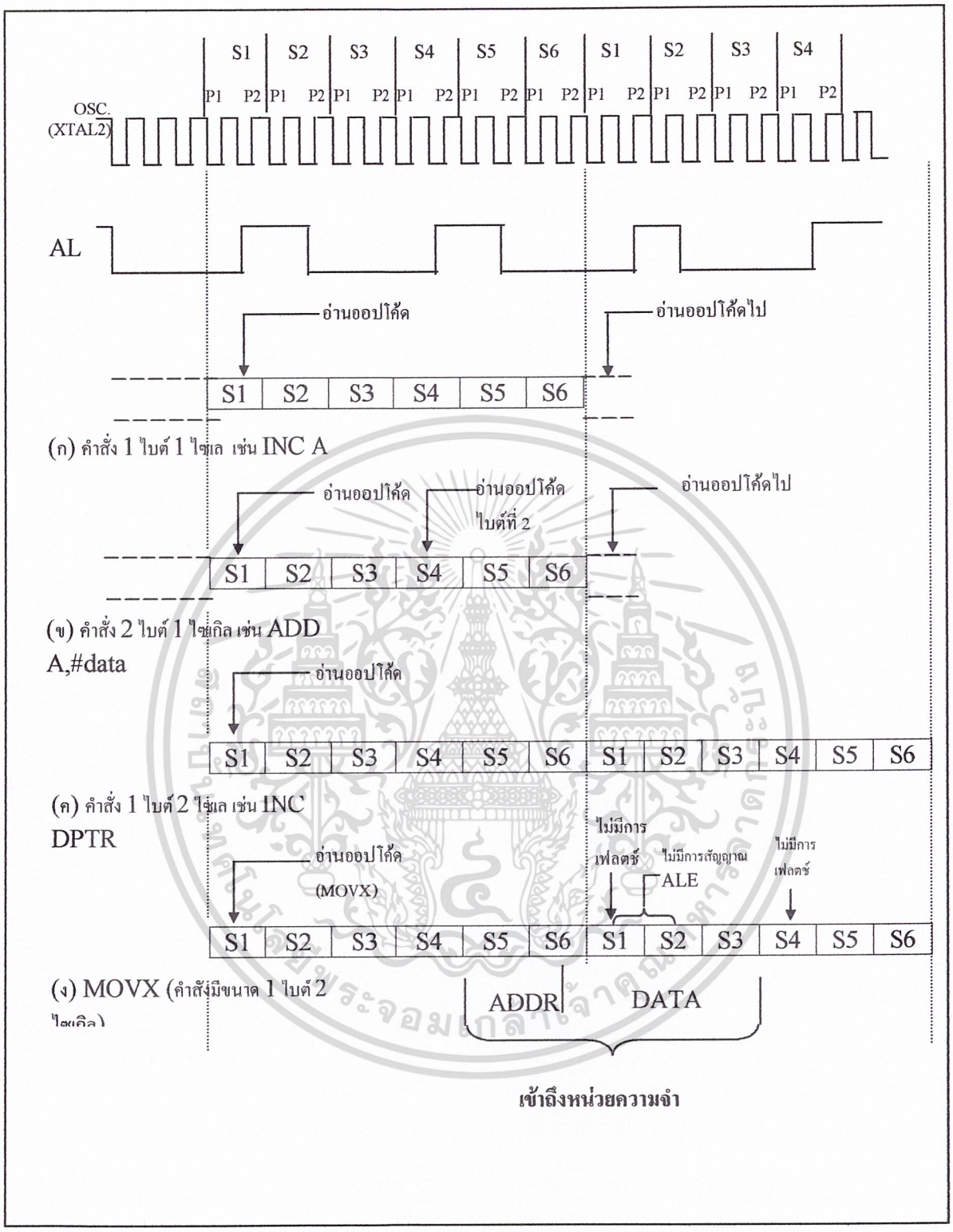
ในรูปที่ 2.7 (ก) และ(ข) จะเป็นการเอ็กซิคิวต์คำสั่งที่ใช้เวลา 1 ไซเกิล เริ่มต้นที่สเตต 1 จะเป็นการอ่านค่าออปโค้ด อันเป็นกระบวนการแลตช์ค่าของออปโค้ดส่งไปให้**รีจิสเตอร์คำสั่ง** *(Instruction Register : IR)* การเฟตช์ครั้งที่สองจะเกิดขึ้นที่สเตต 4 ภายในแมชีนไซเกิลเดียวกัน ในกรณีที่เป็นคำสั่งไบต์เดียว การเฟตช์ครั้งที่ 2 ภายในแมชีนไซเกิลเยวกันจะถูกตัดทิ้งไป ในคำสั่งที่มีใช้เวลา 1 ไซเกิลจะสิ้นสุดการทำงานลงในสเตต 6 ของแมชีนไซเกิลเดียวกัน

ในกรณีที่คำสั่งใช้เวลา 2 ไซเกิล การทำงานของคำสั่งนั้นจะสิ้นสุดลงในสเตต 6 ของแมชีนไซเกิลที่สองดังในไดอะแกรมรูปที่ 2.7 (ค) สำหรับในการกระทำคำสั่ง MOVX ซึ่งเป็นคำสั่งขนาด 1 ไบต์ 2 ไซเกิล จะไม่มีการเฟตช์เกิดขึ้นในไซเกิลที่สองของคำสั่ง MOVX นี้ เนื่องจากซีพียูจะไปทำการติดต่อกับหน่วยความจำภายนอกดังแสดงในไดอะแกรมรูปที่ 2.7 (ง) จะเห็นได้ว่าเวลาในการเอ็กซิคิวต์จะไม่ได้ขึ้นอยู่กับว่าทำการติดต่อหน่วยความจำโปรแกรมภายในหรือภายนอก

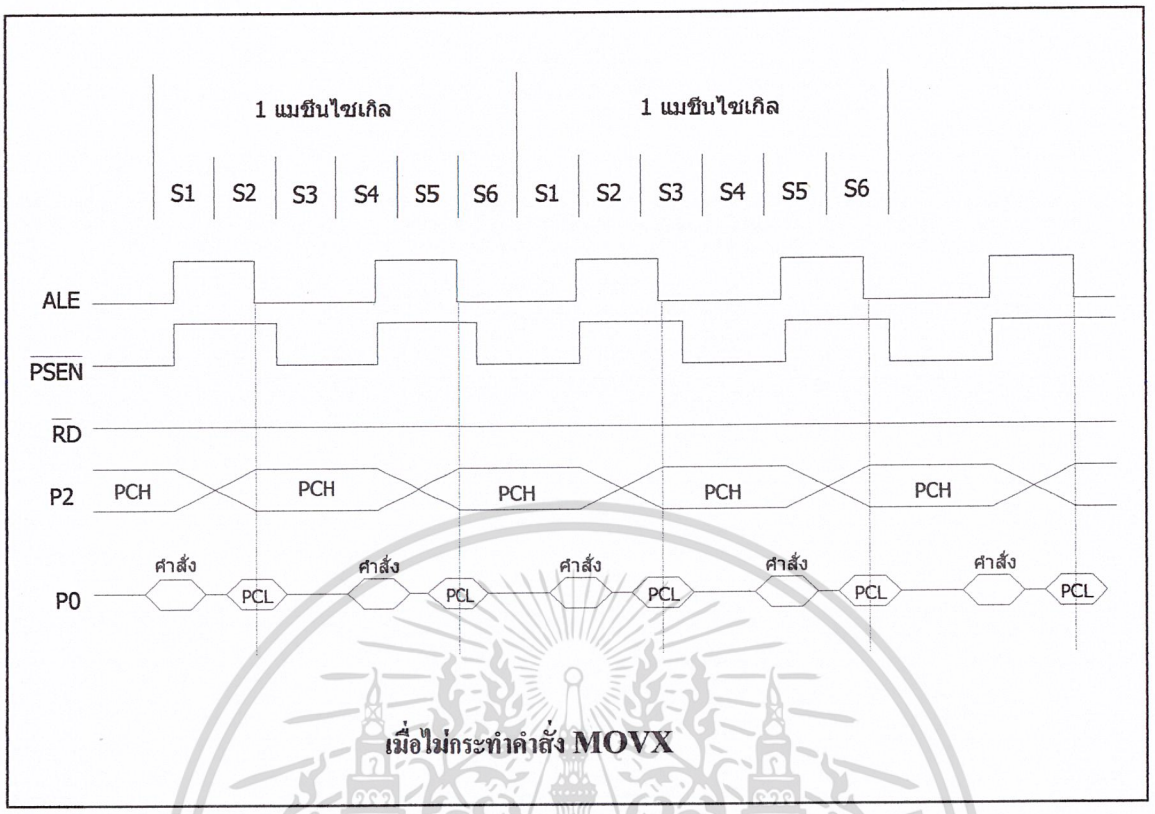
ในรูปที่ 2.8 แสดงสัญญาณและไดอะแกรมเวลาของการเข้าถึงหน่วยความจำโปรแกรมภายนอก โดยในรูปที่ 2.8 (ก) เป็นไดอะแกรมเวลาในขณะที่ยังไม่มีการกระทำคำสั่ง MOVX สัญญาณที่ขา ALE และ PSEN จะเกิดการแอกตีฟ 2 ครั้งภายในหนึ่งแมชีน ไซเกิล ในทุกครั้งที่ ALE เกิดการแอกตีฟที่พอร์ต 0 (P0) จะมีค่าของรีจิสเตอร์ PC ใน ไบต์ต่ำออกมา ใสขณะที่พอร์ต 2 (P2) ก็จะมีค่าของ PC ในไบต์สูงเพื่อชี้ไปยังแอดเดรสต่อไปที่ต้องไปดำเนินการ สำหรับขา PSEN ก็จะเกิดการแอกตีฟเมื่อมีการติดต่อกับหน่วยความจำโปรแกรมภายนอก ในกรณีที่กระทำคำสั่ง MOVX เพื่อเข้าถึงหน่วยความจำข้อมูลภายนอก ที่ขา PSEN จะไม่เกิดการแอกตีฟ 2 ครั้งภายใน 1 แมชีนไซเกิลเนื่องจากบัสแอดเดรสและบัสข้อมูลจะถูกใช้ในการติดต่อหน่วยความจำข้อมูลภายนอกแทน แต่สำหรับสัญญาณ ALE ยังแอกตีฟตามจังหวะการทำงานเหมือนเดิม

จากไดอะแกรมเวลาสามารถสรุปได้ว่า ในการทำงาน 1 รอบหรือ 1 แมชีนไซเกิล ซีพียูในไมโครคอนโทรลเลอร์ MCS-51 จะใช้เวลา 12 คาบเวลาของสัญญาณนาฬิกา นั่นคือ เวลาในการทำงาน 1 ไซเกิลมีค่าเท่ากับ 1 ms หรือมีความเร็วในการทำงานภายใน 1 MHz ในกรณีที่ใช้ความถี่สัญญาณนาฬิกา 12 MHz ดังนั้นถ้าต้องการทราบความเร็วของการทำงานภายในของไมโครคอนโทรลเลอร์ MCS-51 สามารถหาได้จาก ค่าความถี่สัญญาณนาฬิกาหารด้วย 12 และถ้าต้องการหาค่าเวลาของ 1 แมชีนไซเกิล สามารถทำได้โดยการหาส่วนกลับของความเร็ว ในการทำงานภายในของไมโครคอนโทรลเลอร์ MCS-51 สามารถสรุปเป็นสูตรทางคณิตศาสตร์ได้ดังนี้

ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์เท่ากับ
ความถี่ของสัญญาณนาฬิกา (ค่าของคริสตอลที่ต่ออยู่ที่ขา XTAL1 และ XTAL2)/12
เวลา 1 แมชีนไซเกิล = 1/ความเร็วในการทำงานภายในของไมโครคอนโทรลเลอร์



รูปที่ 2.7 ไซเคิลการทำงานของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช



รูปที่ 2.8 ไดอะแกรมแวลาแสดงการติดต่อและเข้าถึงหน่วยความจำโปรแกรมภายนอกของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

บทที่ 3
ระบบการสื่อสารข้อมูลอนุกรมแบบหนึ่งสาย (1-Wire ™ Serial Bus)

ระบบการสื่อสารข้อมูลแบบนี้ผู้ค้นคิดคือ **ดัลลัสเซมิคอนดักเตอร์** ดังนั้นในบางครั้งจึงเรียกระบบสื่อสารข้อมูลแบบนี้ว่า **ระบบสื่อสารข้อมูลดัลลัสหนึ่งสาย (The Dallas 1-Wire Bus)** ระบบสื่อสารข้อมูลแบบนี้เป็นระบบที่มีความชาญฉลาด และใช้จำนวนสายสัญญาณเพียง 1 เส้นเท่านั้น โดยไม่ต้องมีสายสัญญาณนาฬิกามาควบคุมจังหวะการถ่ายทอดข้อมูลเหมือนกับระบบสื่อสารข้อมูลอนุกรมในแบบอื่น ๆ เนื่องจากสายข้อมูลนั้นจะทำหน้าที่เสมือนหนึ่งเป็นสายสัญญาณนาฬิกาในตัว ส่วนค่าของข้อมูลจะพิจารณาจากลักษณะของรูปสัญญาณที่ปรากฏบนสายสัญญาณในแต่ละช่องของเวลาหรือต่อไปนี้จเขอเรียกว่า **ไทม์สล็อต (time-slot)** โดยคาบเวลาต่ำสุดและสูงสุดของสถานะต่าง ๆ ในการสื่อสารข้อมูลในแต่ละไทม์สล็อต มีการกำหนดขอบเขตไว้อย่างชัดเจนการถ่ายทอดข้อมูลจะเกิดขึ้นในแต่ละไทม์สล็อตนั้น รูปแบบการถ่ายทอดข้อมูลจะเป็นแบบอะซิงโครนัสในระดับบิต ไม่มีการกำหนดความยาวของข้อมูลเป็นระดับไบต์ ระบบสื่อสารแบบนี้เหมาะที่จะใช้ในการสื่อสารข้อมูลระหว่างไอซีบนแผงวงจรเดียวกัน หรือสร้างเป็นโครงข่ายสื่อสารแบบทวิสต์แพร์ก็ได้

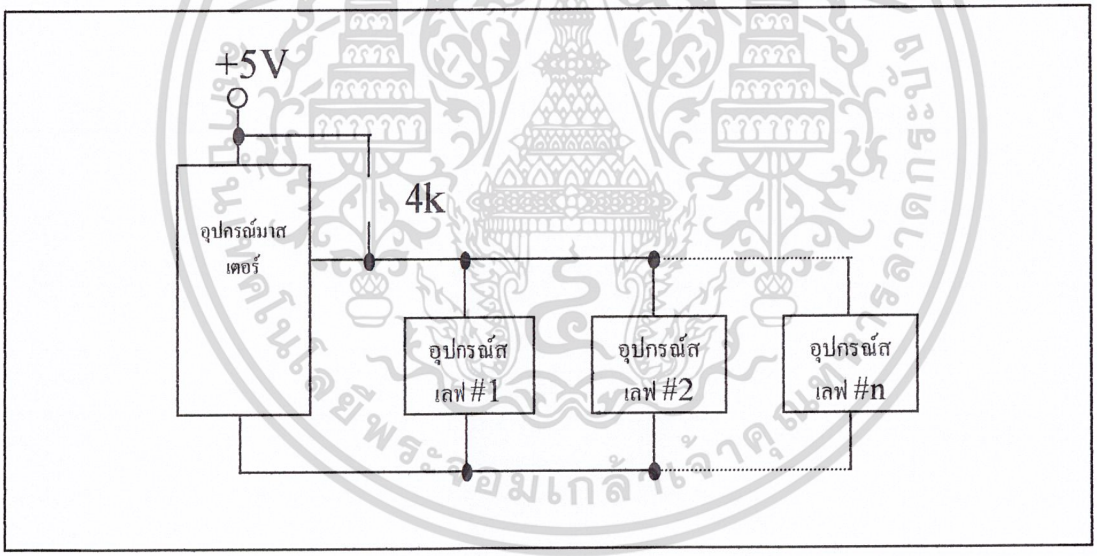
คุณสมบัติทางเทคนิคของระบบบัสหนึ่งสาย

สายสัญญาณบนระบบบัสแบบหนึ่งสายนี้จะเป็นสายสัญญาณแบบสองทิศทาง แต่ข้อมูลจะสามารถเดินทางได้ในทิศทางเดียวภายในช่วงเวลาหนึ่ง ๆนั่นคือ มีลักษณะคล้ายกับระบบสื่อสารแบบฮาล์ฟดูเพล็กซ์ (halfduplex) ตัวอย่างที่เห็นได้ชัดคือ การใช้งานวิทยุสื่อสารหรือวิทยุสมัครเล่น อุปหรณีบนระบบบัสต้องมีการระบุอย่างชัดเจนว่าตัวใดเป็นอุปกรณ์มาสเตอร์ ตัวใดเป็นอุปกรณ์สเลฟ โดยส่วนใหญ่อุปกรณ์มาสเตอร์คือไมโครคอนโทรลเลอร์ ส่วนอุปกรณ์สเลฟได้แก่ ไอซีตรวจจับอุณหภูมิ ไอซีหน่วยความจำแรม เป็นต้น อุปกรณ์มาสเตอร์จะเป็นตัวจัดเตรียมความพร้อมของสายสัญญาณและควาบคุมการถ่ายทอดข้อมูลบนสายสัญญาณ ข้อมูลทั้งหมดไม่ว่าจะเป็นข้อมูลควบคุมหรือข้อมูลใช้งานจะถูกส่งลงบนสายสัญญาณที่มีอยู่เพียงเส้นเดียวนี้ทั้งหมด ในระหว่างการทำงานอุปกรณ์มาสเตอร์และสเลฟสามารถเป็นได้ทั้งตัวส่งและตัวรับ ขึ้นอยู่กับเงื่อนไขของการทำงานในขณะนั้น ยกตัวอย่าง ถ้าหากมีการเขียนข้อมูลจากอุปรณ์มาสเตอร์ไปยังอุปกรณ์สเลฟ ตัวส่งคืออุปกรณ์มาสเตอร์ ตัวรับคืออุปกรณ์สเลฟ ในทางตรงกันข้าม หากเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ ตัวส่งจะกลายเป็นอุปกรณ์สเลฟ และตัวรับคืออุปกรณ์มาสตอร์ ในระบบบัส 1 ระบบต้องมีอุปกรณ์มารเตอร์เพียงตัวเดียวเท่านั้น

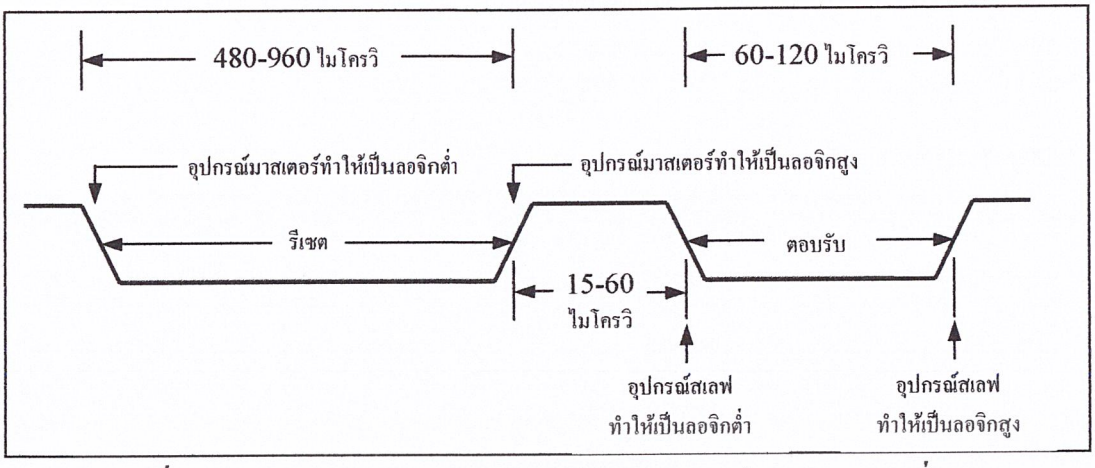
สายสัญญาณของระบบบัสนี้ต้องกำหนดสภาวะปกติไว้ที่ลอจิกสูง สามารถทำได้โดยการต่อต้านทานค่าประมาณ 4.7kΩ พูลอัฟกับไฟเลี้ยง +5V ดังนั้นอุปกรณ์ที่นำเข้ามาต่อบนระบบบัสนี้จึงต้องออกแบบให้ภาคเอาต์พุตที่ต้องต่อกับสายสัญญาณมีลักกษณะเป็นคอลเล็กเตอร์เปิดหรือเดรนเปิด ในรูปที่ P1 แสดงไดอะแกรมการสื่อสารข้อมูลอนุกรมแบบหนึ่งสายเบื้องต้น

คุณสมบัติของไทม์สล็อต

อุปกรณ์มาสเตอร์จะเป็นอุปกรณ์เพียงตัวเดียวบนระบบบัสหนึ่งสายนี้ที่สามารถทำการอินนิเชียลสายสัญญาณได้ โดยอุปกรณ์มานเตอร์จะกำเนิดจุดเริ่มต้นของไทม์สล็อตด้วยการทำให้สายสัญญาณเป็นลอจิกต่ำในช่วงเวลาหนึ่ง จากนั้นก็จะทำให้กลับมาเป็นลอจิกสูง ถ้าหากอุปกรณ์สเลฟต้องการส่งข้อมูลมายังอุปกรณ์มาสเตอร์ อุปกรณ์สเลฟจะเป็นตัวควบคุมสภาวะของสายสัญญาณต่อไป จนเสร็จสิ้นกระบวนการ แต่ถ้าหากอุปกรณ์มาสเตอร์ต้องการส่งข้อมูลก็จะสามารถดำเนินการต่อไปได้เลย



รูปที่ 3.1 การเชื่อมต่อบนระบบบัสหนึ่งสาย



รูปที่ 3.2 ไทม์สล็อตการรีเซตและการตอบรับของอุปกรณ์บนระบบบัสหนึ่งสาย

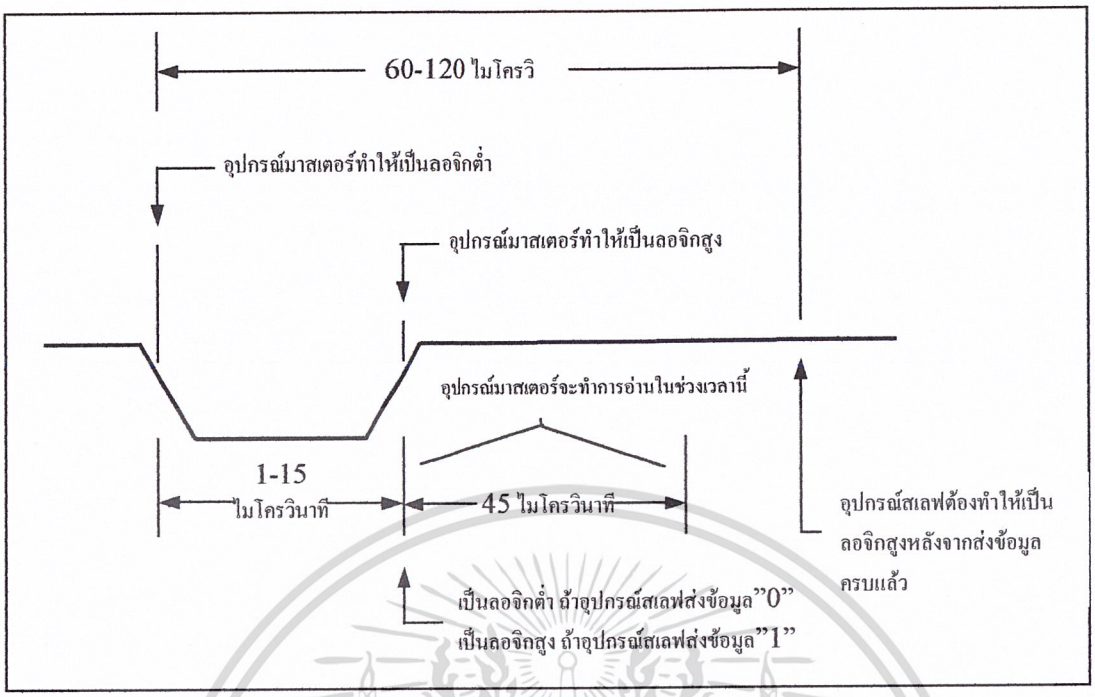
ฟังก์ชั่นของไทม์สล็อตที่กำหนดโดยอุปกรณ์มาสเตอร์มีด้วยกัน 4 ฟังก์ชั่นคือ **ไทม์สล็อตของการรีเซต (RESET), การอ่านข้อมูล (READ DATA), การเขียนข้อมูล "1" (WRITE ONE)** และ **การเขียนข้อมูล "0" (WRITE ZERO)** ไทม์สล็อตรีเซตใช้ในการเริ่มต้นติดต่อกับอุปกรณ์สเลฟ ในขณะที่ไทม์สล็อตการอ่านจะสำหรับอ่านข้อมูลที่ส่งมาจากอุปกรณ์สเลฟ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

ทางด้านอุปกรณ์สเลฟมีฟังก์ชั่นของไทม์สล็อตอยู่ทั้งสิ้น 3 ฟังก์ชั่นคือ **ไทม์สล็อตของการตอบสนอง (PRESENCE), การเขียนข้อมูล "1" (WRITE ONE)** และ **การเขียนข้อมูล "0" (WRITE ZERO)** ไทม์สล็อตของการตอบสนองใช้สำหรับตอบสนองการติดต่อจากอุปกรณ์มาสเตอร์ โดยอุปกรณ์สเลฟตัวที่ถูกเลือกจากอุปกรณ์มาสเตอร์จะต้องส่งสัญญาณตอบสนองลงบนสายสัญญาณเพื่อแจ้งให้อุปกรณ์มาสเตอร์ทราบว่า ขณะนี้สามารถติดต่อกันได้แล้ว ส่วนไทม์สล็อตการเขียนข้อมูล "1" และ "0" ใช้สำหรับส่งข้อมูลไปยังอุปกรณ์มาสเตอร์ผ่านสายสัญญาณของระบบ ซึ่งจะสัมพันธ์กับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์

การแยกแยะฟังก์ชั่นของแต่ละไทม์สล็อตจะใช้ความยาวของาบเวลาและลักษณะของรูปสัญญาณเป็นตัวกำหนด และทุกครั้งที่มีการเปลี่ยนแปลงฟังก์ชั่นต้องทำให้สายสัญญาณอยู่ในสภาวะว่างเสมอ ซึ่งก็คือการทำให้สายสัญญาณเป็นลอจิกสูงอย่างน้อยเป็นเวลา 1 ไมโครวินาที

ไทม์ส็อตการรีเซตและตอบสนอง

อุปกรณ์มาสเตอร์ทำให้เกิดการรีเซตบนสายสัญญาณเพื่อแจ้งแก่อุปกรณ์สเลฟ โดยการทำให้สัญญาณเป็นลอจิกต่ำต่อเนื่องอย่างน้อย 480 ไมโครวินาที และจะต้องทำให้สายสัญญาณกลับมาเป็นลอจิกสูงภายใน 480 ไมโครวินาทีหลังจากนั้น ถ้าหากมีอุปกรณ์สเลฟต่อยู่บนสายสัญญาณ จะมีการตอบสนองสัญญาณรีเซตนั้นด้วยสัญญาณตอบสนอง(PRESENCE) โดยการทำให้สายสัญญาณเป็นลอจิกต่ำต่อเนื่องนานประมาณ 60-240 ไมโครวินาที หลังจากสัญญาณรีเซตปรากฏประมาณ 15-60 ไมโครวินาที ในรูปที่ 3.2 แสดงไทม์สล็อตของการรีเซตและการตอบสนอง



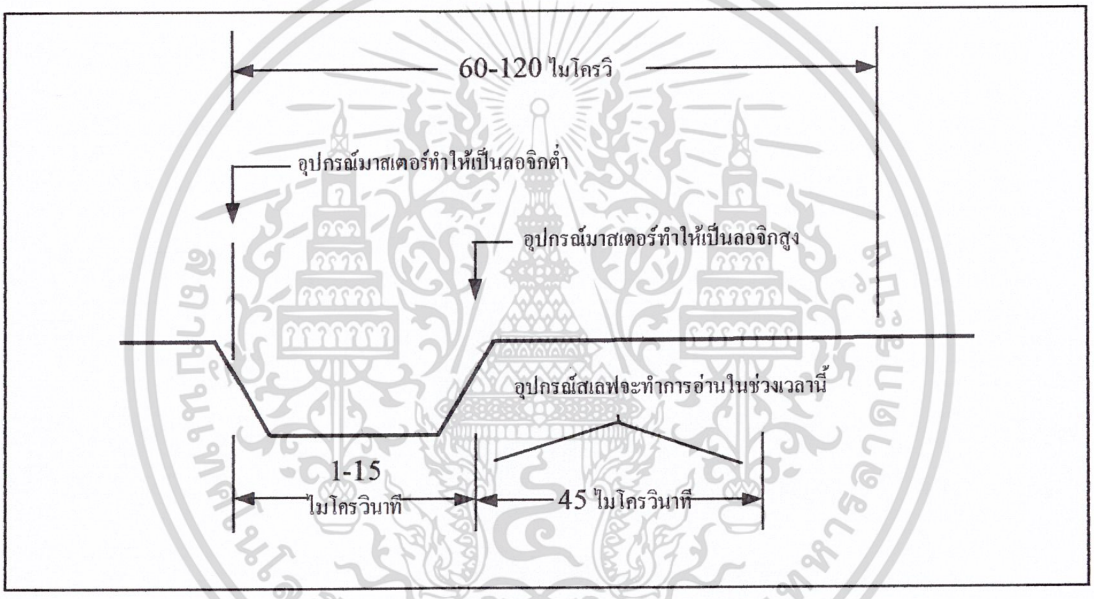
รูปที่ 3.3 ไทม์สล็อตการอ่านข้อมูลของอุปหรณ์มาสเตอร์ซึ่งตรงกับไทม์สล็อตการเขียนข้อมูลของอุปกรณ์สเลฟ

ไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์และการเขียนข้อมูลของอุปกรณ์สเลฟ

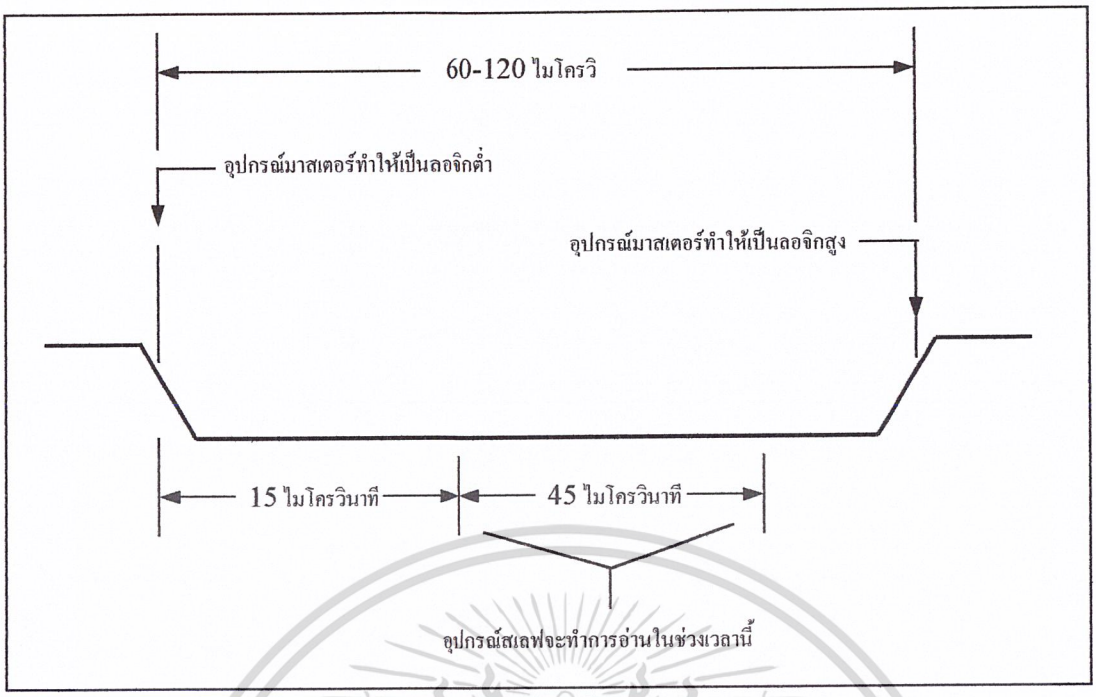
เมื่อต้องการอ่านข้อมูลจากอุปกรณ์สเลฟ อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 1-15 ไมโครวินาที จากนั้นต้องทำให้สถานะของสายกลับมาเป็นลอจิกสูง อุปกรณ์สเลฟจะส่งข้อมูลมาให้อุปกรณ์มาสเตอร์ โดยถ้าข้อมูลเป็น "0" อุปกรณ์สเลฟจะทำให้สายสัญญาณเป็นลอจิกต่ำนานประมาณ 45 ไมโครวินาทีแล้วทำให้สายสัญญาณกลับมาสู่สภาวะลอจิกสูงอีกครั้ง แต่ถ้าเป็นข้อมูล "1" อุปกรณ์สเลฟจะทำให้สายสัญญาณเป็นลอจิกสูงต่อเนื่องไปอีก 45 ไมโครวินาที รวมเวลาทั้งหมดในไทม์สล็อตนี้ประมาณ 60-120 ไมโครวินาที นั่นคือในไทม์สล็อตนี้จะต้องใช้เวลารวมไม่เกิน 120 ไมโครวินาที ในขณะที่อุปกรณ์มาสเตอร์จะใช้เวลาในการอ่านข้อมูลอยู่ระหว่าง 15 และ 60 ไมโครวินาทีหลังจากเริ่มต้นไทม์สล็อตนี้ ในรูปที่ 3.3 แสดงรูปสัญญาณของไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มานเตอร์ซึ่งก็จะมีลักษณะเหมือนกับการเขียนข้อมูลของอุปกรณ์สเลฟ และไทม์สล็อตทั้งสองจะเกิดขึ้นในช่วงเวลาเดียวกัน กล่าวคือ เมื่ออุปกรณ์มาสเตอร์อ่าน อุปกรณ์สเลฟก็ต้องทำการเขียน

ไทม์สล็อตการเขียนข้อมูลของอุปกรณ์มาสเตอร์

เมื่ออุปกรณ์มาสเตอร์ต้องการเขียนข้อมูล อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 1-15 ไมโครวินาที จากนั้นต้องทำให้สถานะของสายกลับมาเป็นลอจิกสูง แล้วดำเนินการเขียนข้อมูลได้ในทันที ถ้าข้อมูลที่ต้องการเขียนไปยังอุปกรณ์สเลฟเป็น “0” อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำนานประมาณ 45 ไมโครวินาที แล้วทำให้สายสัญญาณกลับมาสู่สภาวะลอจิกสูงอีกครั้ง แต่ถ้าต้องการเขียนข้อมูล “1” อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกสูงต่อเนื่องไปอีก 45 ไมโครวินาที รวมเวลาทั้งหมดในไทม์สล็อตนี้ประมาณ 60-120 ไมโครวินาที ในรูปที่ 3.4 แสดงรูปสัญญาณของไทม์สล็อตการเขียนข้อมูลของอุปกรณ์มาสเตอร์ซึ่งก็จะมีลักษณะเหมือนกับการอ่านข้อมูลของอุปกรณ์สเลฟ และไทม์สล็อตทั้งสองจะเกิดขึ้นในช่วงเวลาเดียวกัน กล่าวคือ เมื่ออุปกรณ์มาสเตอร์เขียน อุปกรณ์สเลฟก็ต้องทำการอ่านข้อมูล



รูปที่ 3.4 ไทม์สล็อตการเขียนข้อมูล “1” ของอุปกรณ์มาสเตอร์ ซึ่งตรงกับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์สเลฟ



รูปที่ 3.5 ไทม์สล็อตการเขียนข้อมูล "0" ของอุปกรณ์มาสเตอร์

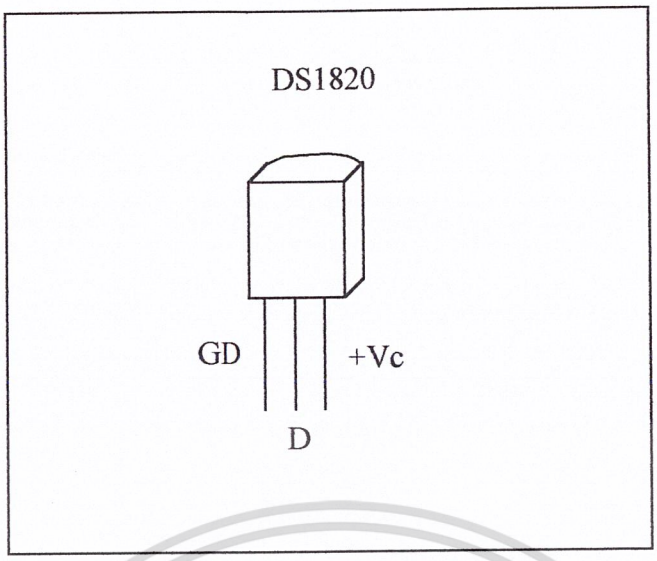
รูปแบบของการสื่อสารข้อมูลแบบหนึ่งสาย (1-Wire™ communication protocol)

ในการติดต่อสื่อสารข้อมูลในระบบบัสหนึ่งสายอุปกรณ์มาสเตอร์จะสามารถติดต่อกับอุปกรณ์สเลฟได้ครั้งละ 1 ตัวเท่านั้น ดังนั้นอุปกรณ์สเลฟแต่ละตัวต้องมีข้อมูลกำหนดแอดเดรสเฉพาะตัวโดยจะเก็บไว้ในหน่วยความจำรอมภายในอุปกรณ์สเลฟตัวนั้น ๆ โดยปกติอุปกรณ์สเลฟในระบบบัสหนึ่งสายของดัลลัสนี้จะมีหน่วยความจำขนาด 64 บิตหรือ 8 ไบต์ สำหรับเก็บข้อมูลต่างๆ ที่สำคัญของอุปกรณ์แต่ละตัว ซึ่งประกอบด้วย

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

ผู้ใช้งานสามารถอ่านข้อมูลประจำตัวของอุปกรณ์สเลฟได้ด้วยการใช้คำสั่งอ่านหน่วยความจำรอม (Read ROM) ในกรณีที่บนสายสัญญาณมีอุปกรณ์สเลฟเพียงตัวเดียวไม่จำเป็นต้องอ้างแอดเดรสในการติดต่อ

รูปแบบการติดต่อบนระบบบัสหนึ่งสายจะเริ่มต้นขึ้นเมื่ออุปกรณ์มาสเตอร์ทำการรีเซตและกำหนดแอดเดรสของอุปกรณ์ที่ทำการติดต่อ ถ้าหากมีอุปกรณ์สเลฟเพียงตัวเดียวสามารถข้ามขั้นตอนการติดต่อกับหน่วยความจำรอมในอุปกรณ์สเลฟได้ จะเรียกวิธีการดังกล่าว **การไม่ติดต่อหน่วยความจำรอม** หรือ **สคิปรอม (Skip ROM)** จากนั้นรอการตอบรับจากอุปกรณ์สเลฟ เมื่อการตอบรับสมบูรณ์ก็จะสามารถเริ่มต้นขั้นตอนการอ่านหรือเขียนข้อมูลได้ต่อไป

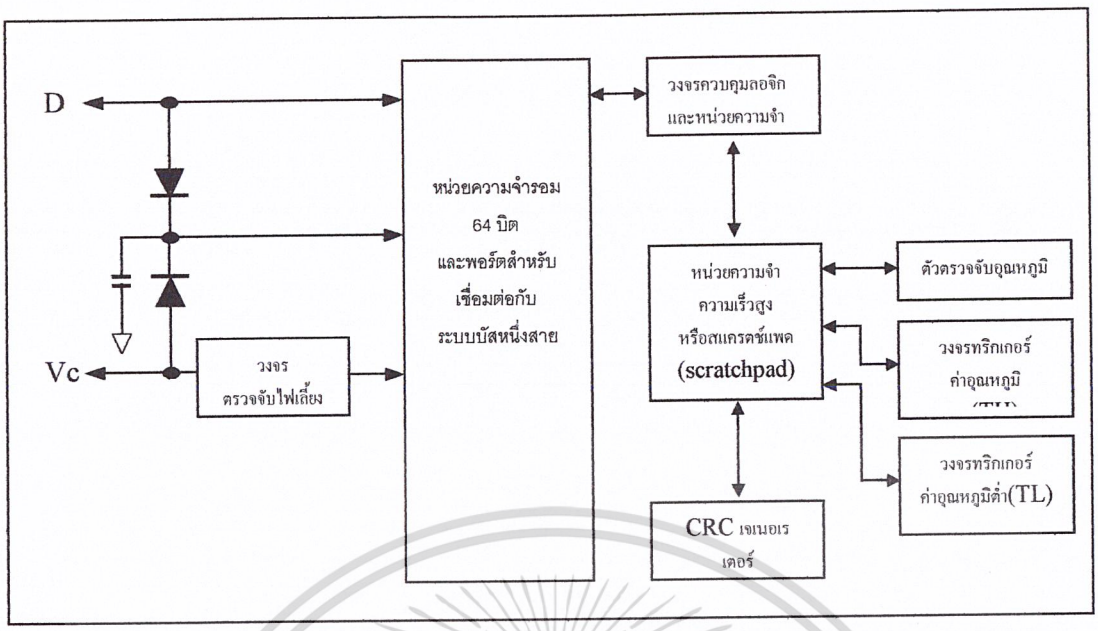


รูปที่ 3.6 ไอซีตรวจจับอุณหภูมิ DS1820

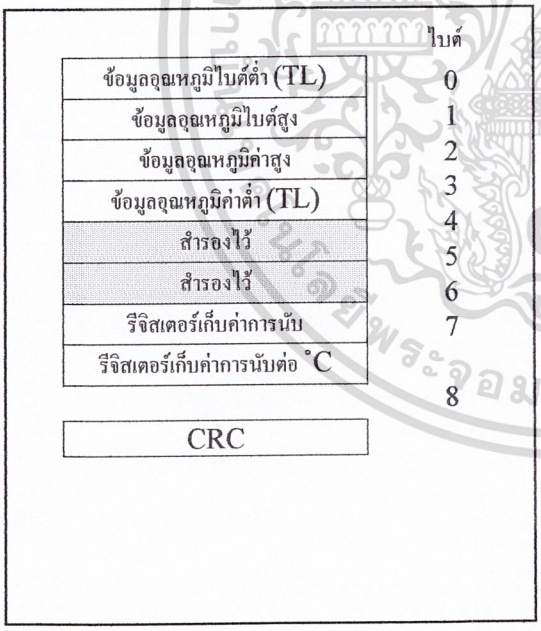
รูปแบบการติดต่อบนระบบบัสหนึ่งสายจะเริ่มต้นขึ้นเมื่ออุปกรณ์มาสเตอร์ทำการรีเซตและกำหนดแอดเดรสของอุปกรณ์ที่ทำการติดต่อกับระบบบัส, ขาต่อไฟเลี้ยงภายนอก และขากราวด์ ดังแสดงการจัดขาของไอซี DS1820 ในรูปที่ 3.6 และมีโครงสร้างการทำงานภายในแสดงในรูปที่ 3.7

หัวใจสำคัญของ DS1820 อยู่ที่ตัวตรวจจับอุณ๓มิและหน่วยความจำความเร็วสูงที่เรียกว่าสแครตช์แพด(scratchpad) ซึ่งมีขนาด 9 ไบต์ มีการจัดสรรหน่วยความจำส่วนนี้แสดงในรูปที่ 3.8

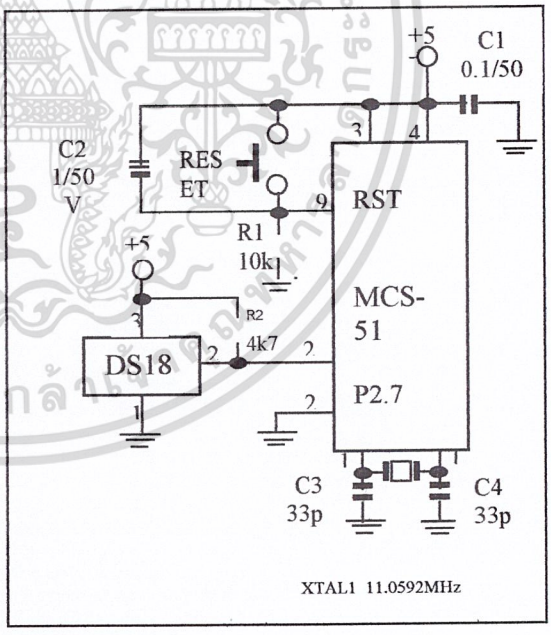
เมื่อวัดอุณ๓มิได้ก็จะนำค่าที่วัดได้นี้มาเก็บไว้ในสแครตช์แพดที่ไบต์ 0 และ 1 ทั้งนี้เนื่องจากไอซี DS1820 สามารถให้ข้อมูลของอุณหภูมิได้ละเอียดถึง 16 บิต เมื่อนำมาแปลงเป็นข้อมูลเลขฐานสิบจึงสามารถแสดงความละเอียดของค่าอุณหภูมิได้ถึง 0.5 องศาเซลเซียสและ 0.9 องศาฟาเรนไฮต์ โดยมีย่านวัดอุณหภูมิ —55 ถึง +125 องศาเซลเซียสหรือ —67 ถึง +257 องศาฟาเรนไฮต์ โดยค่าขององศาฟาเรนไฮต์ต้องใช้การแปลงหน่วยเข้ามาช่วยใช้เวลาในการแปลงค่าอุณหภูมิเป็นข้อมูลดิจิตอลประมาณ 200 มิลลิวินาที สามารถกำหนดขอบเขตของอุณหภูมิที่ทำการวัดได้ และให้แจ้งเตือนเมื่อค่าของอุณหภูมิสูงขึ้นหรือลดต่ำลงถึงค่าที่กำหนด โดยค่าอุณหภูมิที่กำหนดนี้จะเก็บไว้ที่สแครตช์แพดในไบต์ 2 และ 3



รูปที่ 3.7 โครงสร้างการทำงานภายในของไอซีตรวจจับอุณหภูมิ DS1820



รูปที่ 3.8 การจัดสรรพื้นที่ของสแครตช์แพดใน DS1820



รูปที่ 3.9 การเชื่อมต่อ DS1820 กับไมโครคอนโทรลเลอร์ MCS-51

คำสั่งเพื่อควบคุมการทำงานของ DS1820

ในการติดต่อกับไอซี DS1820 จะมีคำสั่งที่ต้องส่งให้แก่ DS1820 เพื่อกำหนดรูปแบบการทำงาน คำสั่งที่ใช้มากที่สุดมีด้วยกัน 3 คำสั่งดังนี้

1. **คำสั่งไม่ติดต่อกับหน่วยความจำรอมหรือสคิปรอม(Skip ROM)** เนื่องจากในการใช้งาน DS1820 โดยปกติแล้วจะมี DS1820 อยู่บนสายสัญญาณเพียงตัวเดียว จึงไม่จำเป็นต้องใช้ข้อมูลกำหนดแอดเดรส ดังนั้นจึงไม่ต้องติดต่อกับหน่วยความจำรอมเพื่ออ่านข้อมูล ข้อมูลของคำสั่งสคิปรอมที่ต้องส่งให้ DS1820 คือ **0CCH**

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

3. **คำสั่งอ่านข้อมูลจากสแครตช์แพด (Read Scratchpad)** มีค่าเท่ากับ **0BEH** เมื่อส่งคำสั่งนี้ DS1820 จะทยอยส่งข้อมูลค่าอุณหภูมิออกมาทั้งหมด 9 ไบต์

การเชื่อมต่อกับไมโครคอนโทรลเลอร์ MCS-51

แสดงวงจรการเชื่อมต่อในรูปที่ 3.9 ใช้ขาพอร์ตเพียง 1 ขาเท่านั้นสำหรับการเชื่อมต่อกับ DS1820 โดยต้องมีตัวต้านทานค่า 4.7kΩ ต่อพูลอัปกับไฟเลี้ยง +5V จากนั้นจึงทำการเขียนโปรแกรมเพื่อติดต่อกัน โดยใช้รูปแบบการติดต่อตามมาตรฐานระบบบัสหนึ่งสายของดัลลัส

คำอธิบายโปรแกรมอ่านค่าไอซีตรวจจับอุณหภูมิ DS1820

เป็นโปรแกรมที่ใช้สำหรับอ่านค่าจากไอซีวัดอุณหภูมิเบอร์ **DS1820** ซึ่งใช้การติดต่อแบบระบบบัสหนึ่งสาย โดยให้แสดงค่าอุณหภูมิที่อ่านได้บน **LCD** และพิจารณาเฉพาะอุณหภูมิที่อยู่ในช่วง **0-127** องศาเซลเซียส

เริ่มจากการส่งสถานะรีเซตด้วยโปรแกรมย่อย **DS1820_RST** จากนั้นให้รอการตอบรับจาก **DS1820** ด้วยโปรแกรมย่อย **DS1820_PRES** เมื่อมีการตอบรับแล้วให้ทำการส่งค่า **CCH** ไปยัง **DS1820** โดยผ่านโปรแกรมย่อย **DS1820_WR** เพื่อส่งคำสั่ง **Skip ROM** ให้ข้ามการตรวจสอบ **64-BIT LASERED ROM** และส่งค่า **44H** เป็นคำสั่งให้ **DS1820** ทำการแปลงค่าอุณหภูมิมาเก็บไว้ที่หน่วยความจำ **SCRATCHPAD** ภายใน

ในขั้นต่อมา ก็เริ่มส่งสถานะรีเซตใหม่ แล้วรอการตอบรับเช่นเดิม แต่จะทำการตรวจสอบสถานะบิต **BUSY** ว่าทำการแปลงค่าอุณหภูมิเสร็จแล้วหรือไม่ ถ้าไม่ก็ให้วนลูปแกรมรอจนกว่าจะเสร็จสิ้นการแปลงค่าจากนั้นให้ทำการส่งสถานะรีเซตใหม่ แล้วรอรับการตอบรับเช่นเดิม เมื่อมีการตอบรับแล้วให้ทำการส่งค่า **CCH** เพื่อทำการ **Skip ROM** และส่งค่า **BEH** เป็นคำสั่งเรียกอ่านค่าจาก

หน่วยความจำสแครตช์แพด จากนั้นให้อ่านค่าจาก **DS1820** โดยเรียกโปรแกรมย่อย **DS1820_RD** ซึ่งการอ่านค่าในไบต์แรกนั้น จะได้ค่าอุณหภูมิในช่วง **8** บิตล่างออกมา **(TEMPERATURE LSB)** และจะถูกเก็บอยู่ในรีจิสเตอร์ชื่อ **ONEWIRE_DATA** ให้ทำการเก็บค่านี้ไว้ที่รีจิสเตอร์ **TEMP** อีกตัวหนึ่งเพื่อนำมาใช้งานต่อไป จากนั้นให้ส่งสถานะรีเซตใหม่แล้วรอการตอบรับอีกครั้งหนึ่ง เป็นอันจบขั้นตอนการติดต่อกับ **DS1820**

นำข้อมูลในรีจิสเตอร์ **TEMP** ที่ได้มาทำการหมุนไปทางขวาโดยใช้แฟลกร่วมด้วย (ให้ทำการเคลียร์ค่าแฟลกทดก่อนใช้ด้วย) จะได้ค่าอุณหภูมิที่เป็นจำนวนเต็มพอดี โดยนำค่าที่ได้นี้มาแสดงบน **LCD** โดยค่าอุณหภูมิที่เป็นจำนวนเต็มให้เรียกผ่านโปรแกรมย่อย **HEX2LCD** เป็นเลยฐานสิบจำนวน **3** หลัก และดูค่าบิตล่างสุดของรีจิสเตอร์ **TEMP** ถ้ามีค่าเป็น **"1"** จะหมายถึงเพิ่มค่าอุณหภูมิอีก **0.5** องศาเซลเซียส ก็ให้เขียนค่า **".5"** ลงไป แต่ถ้าค่าเป็นศูนย์ ก็ให้เขียนค่า **".0"** ลงไปแทน อาจเขียนตัว "C" เพิ่มเติมเพื่อบอกว่าเป็นหน่วยองศาเซลเซียสก็ได้ เป็นอันสิ้นสุดการทำโปรแกรมใน 1 ครั้ง และทำเช่นนี้วนไปเรื่อย ๆ ได้

คำอธิบายโปรแกรมย่อยที่ใช้

HEX2LCD เป็นโปรแกรมย่อยที่ใช้แปลงค่ารีจิสเตอร์ มาแสดงเป็นเลขฐานสิบ 3 หลักที่โมดูล LCD โดยนำค่าจากรีจิสเตอร์ LCD_DATA มาหารด้วย 100 ก่อน แล้วนำค่าตัวส่วนมาแสดงเป็นเลขหลักแรกแต่ถ้าค่าที่ได้เป็นศูนย์ จะเขียนเป็นช่องว่างแทน ต่อมานำค่าเศษที่เหลือจากการหารครั้งแรกมาหารด้วย 10 แล้วนำค่าตัวส่วนมาแสดงบน LCD เป็นเลขหลักต่อมา และนำค่าเศษจากการหารครั้งที่สองมาแสดงเป็นเลขหลักสุดท้าย (การนำค่าเลขมาแสดงบน LCD ต้องบวกค่านั้นด้วย 30H ให้เป็นรหัสแอสกี้ก่อนจึงนำไปใช้ได้)

DS1820_RD เป็นโปรแกรมย่อยที่ใช้ในการอ่านค่าจาก DS1820 โดยในขั้นแรกจะทำการเคลียร์สัญญาณ ONEWIRE ให้เป็น "0" ก่อน จากนั้นให้หน่วงเวลาไปประมาณ 2µs แล้วเซตสัญญาณ ONEWIRE ให้เป็น "1" แล้วหน่วงเวลาไปประมาณ 4µs (ไม่เกินค่า Read Data Valid 15µs) จึงทำการอ่านค่าสัญญาณ ONEWIRE เก็บไว้ที่แฟลกทด แล้วหน่วงเวลาไป 75µs (ค่าไทม์สล็อตอยู่ระหว่าง 60-120µs) แล้วหมุนข้อมูลไปทางขวาโดยใช้แฟลกร่วมด้วย ทำเช่นนี้ 8 ครั้งจะได้ค่ามาเก็บไว้ในรีจิสเตอร์ ONEWIRE_DATA

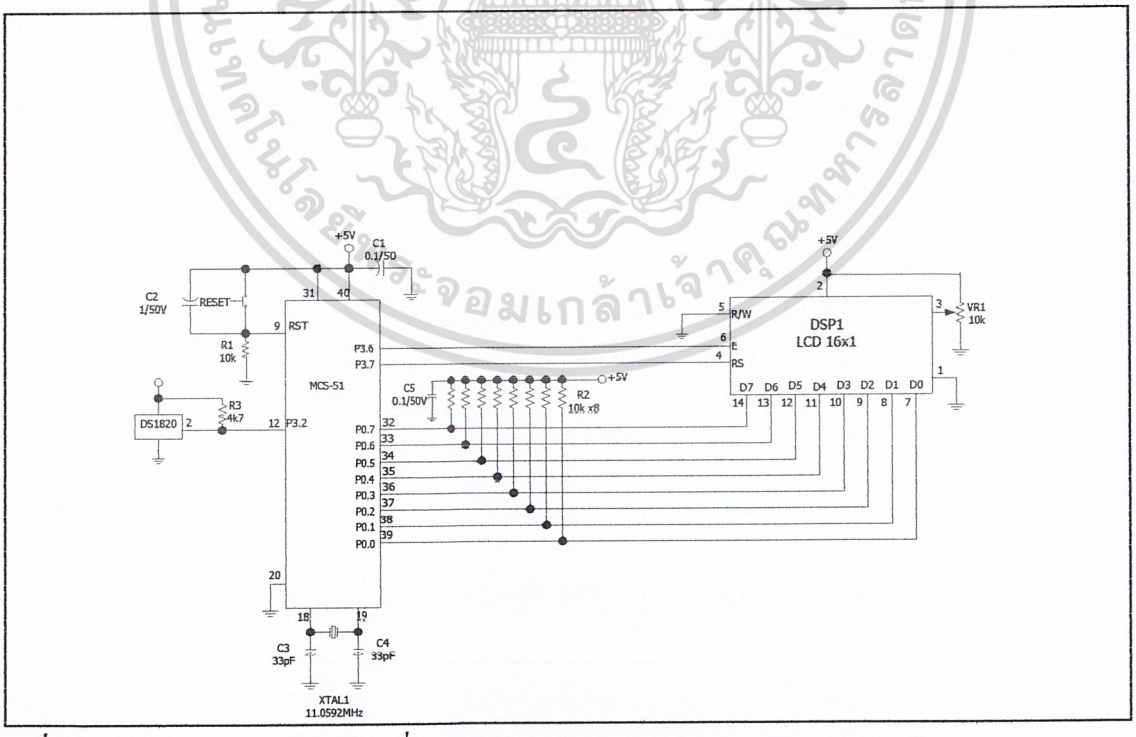
DS1820_WR เป็นโปรแกรมย่อยที่ใช้ในการเขียนค่าจากรีจิสเตร์ ONEWIRE_DATA ไปยัง DS1820 โดยขั้นแรกจะทำการหมุนข้อมูลไปทางขวาโดยใช้แฟลกทดร่วมด้วย จากนั้นให้ทำการตรวจสอบแฟลกทดว่าถูกเซตหรือไม่ ถ้าถูกเซต ก้ให้ทำการเขียน Time Slot ของลอจิก "1" คือทำการเคลียร์สัญญาณ ONEWIRE เป็นเวลา 4µs (อยู่ในช่วง Write 1 Low Time) แล้วทำการเซตสัญญาณ ONEWIRE เป็นเวลา 75µs (ไม่มีค่าไทม์สล็อตอยู่ระหว่าง 60-120µs) แต่ถ้าแฟลกทดไม่ถูกเซต ก็ให้ทำการเขียน Time Slot ของลอจิก "0" คือทำการเคลียร์สัญญาณ ONEWIRE เป็นเวลา 75µs (อยู่ใน

ช่วง Write 0 Low Time) แล้วทำการเซตสัญญาณ ONEWIRE เป็นเวลา 4μs แทน แล้วกลับไปหมุนข้อมูลใหม่ รวมเป็น 8 ครั้ง ก็จะเสร็จสิ้นการเขียนข้อมูล

DS1820_RST เป็นโปรแกรมย่อยที่ใช้สร้างสถานะรีเซต โดยจะทำการเคลียร์สัญญาณ ONEWIRE เป็นเวลา 1ms (อยู่ในช่วง Reset Time Low 480-4800μs) จากนั้นก็จะทำการเซตสัญญาณ ONEWIRE เป็นเวลาประมาร 16μs (พิจารณาจากค่า Presence Detect High 15-60μs) เพื่อหน่วงเวลารอ Presence Pulse จาก DS1820 ต่อไป

DS1820_PRES เป็นโปรแกรมย่อยที่ทำหน้าที่รอรับการตอบรับจาก DS1820 (Presence Pulse) และใช้ทำหน้าที่รอการสิ้นสุดกระบวนการแปลงค่าอุณหภูมิด้วย โดยจะทำการรวนตรวจสอบว่ามีการเคลียร์สัญญาณ ONEWIRE เป็น “0” หรือไม่ในช่วงเวลาประมาณ 8.85ms (การตอบรับปรกติจะสามารถตรวจสอบได้ในภายในช่วง Presence Detect High 15-60μs ในกรณีที่รอการแปลงค่าอุณหภูมิ จะต้องใช้เวลาช่วง Temperature Conversion Time 200μs-500μs) โดยถ้าเกินจากช่วงเวลานี้จะคืนค่าแฟลก BUSY เป็น “1” แต่ถ้าตรวจสอบได้ว่ามีการตอบรับ ก็จะรอจนกระทั่งสัญญาณ ONEWIRE เป็น “1” อีกครั้ง แล้วจะทำการหน่วงเวลาไปประมาณ 16μs ก่อนจะทำการเคลียร์แฟลก BUSY เป็น “0” เพื่อแสดงว่ามีการตอบรับกลับมา จึงจะกลับเข้าสู่การทำงานของโปรแกรมหลักได้

ONEWIRE_DELAY เป็นโปรแกรมย่อยสำหรับหน่วงเวลา เป็นเวลาประมาณ 75μs เพื่อใช้ในการเขียนไทม์สล็อตให้อยู่ในช่วงเวลาที่เหมาะสม



รูปที่ 3.10 โฟลวชาร์ตของโปรแกรมที่ 3.1โปรแกรมทดลองไอซีอราจจับอุณหภูมิ DS1820

บทที่ 4
รายละเอียดเกี่ยวกับโมดูล LCD

ในโมดูล LCD จะมีส่วนประกอบหลัก ๆ 3 ส่วน ดังนี้

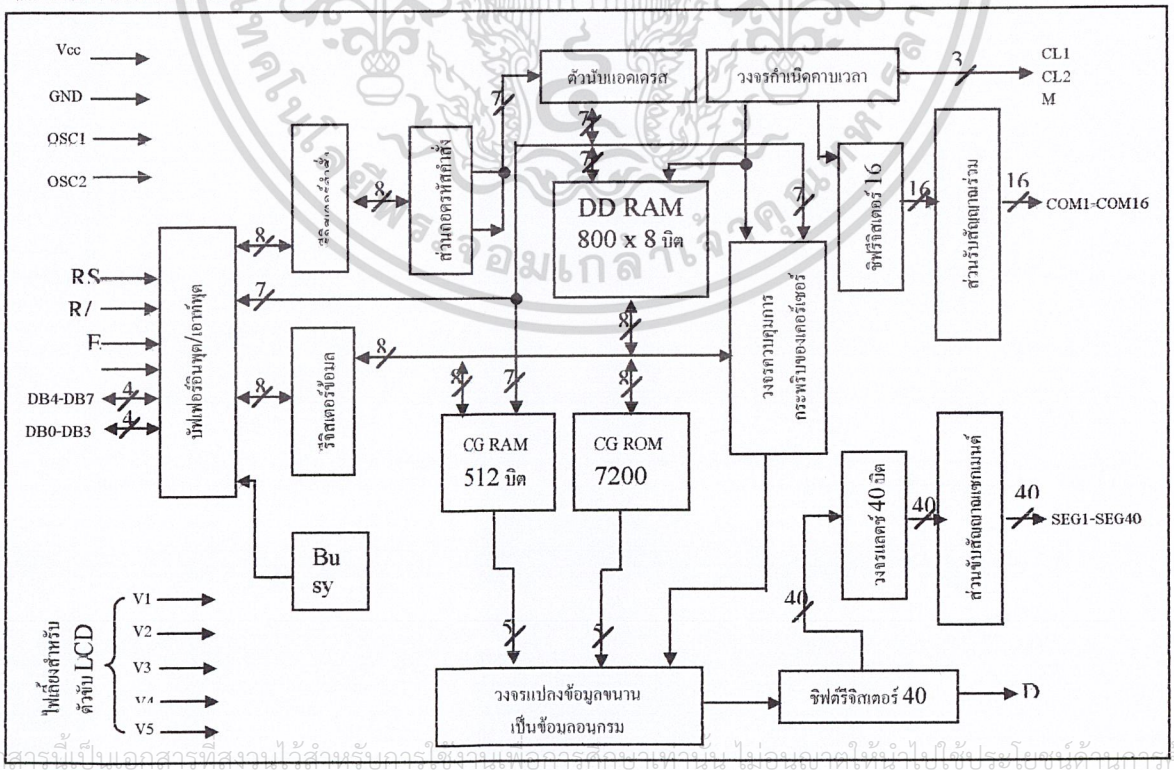
ตัวแสดงผล (display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

ตัวควบคุม (controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะชิปที่นิยมใช้คือ เบอร์ *HD44780* และ *HD61830* โดย HD44780 จะใช้ควบคุม LCD แบบอักขระส่วน HD61830 ใช้ควบคุม LCD แบบกราฟิก

ตัวขับ (driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ *HD44100H* และ *MSM5259* เป็นต้น

โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ในหนังสือจึ้ขอยกตัวอย่างโมดูล LCD แบบอักขระ เพราะสามารถเข้าใจได้ง่าย ให้รูปที่ 1 เป็นบล็อกไดอะแกรมภายในของชิปควบคุม LCD เบอร์ HD44780 ซึ่งใช้ในโมดูล LCD แบบอักขระ ประกอบด้วย



รูปที่ 4.1 ไดอะแกรมการทำงานของโมดูล LCD แบบอักขระ

บัฟเฟอร์อินพุตเอาต์พุต เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

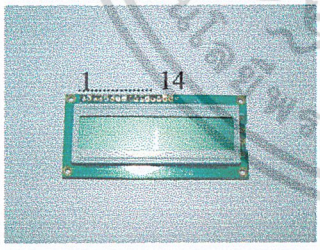
รีจิสเตอร์คำสั่ง (Instruction Register : IR) เป็นรีจิสเตอร์ใช้รับข้อมูลจากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

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

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

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

แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่าตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน



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

รูปที่ 4.2 รูปร่างและการจัดขาโมดูล LCD แบบอักขระ

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

ตารางที่ 4.1 แสดงความสัมพันธ์ในการทำงานของขา RS,R/W และ E ของโมดูล LCD แบบอักขระ

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

สำหรับโมดูล LCD ที่ยกมาใช้เป็นขนาด 16 ตัวอักษร 1 บรรทัด เนื่องจากราคาถูก หาง่าย และเป็นโมดูล LCD ที่มีโครงสร้างเป็นมาตรฐาน มีผู้ผลิตหลายราย และมีการระบุเบอร์แตกต่างกันออกไปจามผู้ผลิต อาทิ LM020L ของฮิตาชิ, DMC-16117A ของคอปเทร็กซ์ (Optex) เป็นต้น แต่อย่างไรก็ตามคอนโทรลเลอร์ที่ใช้คือเบอร์เดียวกันนั่นคือเบอร์ HD44750 ของฮิตาชิ

โมดูล LCD ขนาด 16 x 1 มีขาต่อใช้งานทั้งสิ้น 14 ขา มีการจัดขาดังในรูปที่ 2 สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

Vss (ขา 1) : ต่อกราวด์

V_{DD} (ขา 2) : ต่อไฟเลี้ยง +5 โวลต์

V_O (ขา 3) : เป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา 4) : เป็นขาอินพุตใช้ในการยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี้เป็น “0” ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

R/W (ขา 5) : เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับโมดูล LCD ถ้าเป็น “0” เป็นการกำหนดให้เขียนข้อมูล แต่ถ้าเป็น “1” จะเป็นการอ่านข้อมูล

E (ขา 6) : เป็นขาสำหรับรับสัญญาณพัลส์เอ็นเอเบิลโมดูล LCD ให้ทำงาน

D0-D7 (ขา 7-14) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต อนึ่งขา RS,R/W และ E จะใช้งานร่วมกันโดยมีความสัมพันธ์แสดงในตารางที่ 1

คำสั่งควบคุมโมดูล LCD

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

1. สั่งเคลียร์ตัวแสดงผล (clear display)

มีข้อมูลคำสั่งเป็น 01H เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่าง หรือ space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมเอ็กซิคิวต์คำสั่งนี้ จะทำการกำหนดแอดเดรสของ DDRAM เป็น 0 เคอร์เซอร์จะกลับไปอยู่ที่ตำแหน่งซ้ายมือสุดของจอแสดงผล แล้วเซตบิต I/D (ซึ่งจะกล่าวถึงภายหลัง) ให้เป็น “1”

2. คำสั่ง return home

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น “1” เป็นคำสั่งให้เคอร์เซอร์เคลื่อที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอแสดงผลไม่เปลี่ยนแปลง นั่นคือ ข้อมูลคำสั่งของคำสั่งนี้จะเป็น 02H หรือ 03H ก็ได้

3. คำสั่งเลือกโหมดการป้อนข้อมูล (Entry mode Set)

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

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

บิต I/D เป็นบิตที่ใช้กำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น "1" แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าเป็น "0"แอดเดรสจะลดลง

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

4. คำสั่งควบคุมการแสดงผล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

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

บิต C ใควบคุมการแสดงตัวเคอร์เซอร์บนจอแสดงผล ถ้าต้องการให้มีเคอร์เซอร์แสดงผลบนจอแสดงผล ต้องกำหนดให้บิตนี้เป็น "1" ถ้ากำหนดให้เป็น "0" จะเป็นการปอดเคอร์เซอร์ หรือไม่แสดงเคอร์เซอร์

บิต B ใช้ควบคุมการกะพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น "1" เคอร์เซอร์จะกะพริบ

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

5. คำสั่งควบคุมการเลื่อนเคอร์เซอร์และข้อมูลตัวอักษร

รายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

การควบคุมการเลื่อนเคอร์เซอรีและตัวอักษรบนจอแสดงผลขึ้นอยู่กับการกำหนดบิต S/C และ R/L ซึ่งสามารถสรุปได้ดังนี้

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

6. คำสั่งกำหนดฟังก์ชันการทำงาน

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	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” จุดที่น่าสังเกตคือ 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 ถ้าบิดนี้เป็น “0” จะเป็นการแสดงผลแบบ 5x7 จุด และถ้าเป็น “1” จะแสดงผลเป็นแบบ 5x10 จุด

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

7. คำสั่งเลือกแอดเดรสของ CGRAM

เมื่อต้องการกำหนดแอดเดรสของ CGRAM ต้องกำหนดให้บิต 7 เป็น “0” บิต 6เป็น “1” ส่วนอีก 6บิตที่เหลือจะแทนด้วยค่าแอดเดรสของ CGRAMจะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ก่อนที่จะอ่านหรือเขียนข้อมูลให้ 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 ทั้งนี้จำนวนแอดเดรสยังขึ้นกับการกำหนดสถานะที่บิต N ด้วย หากบิต N เป็น “0” แอดเดรสของ DDRAM จะอยู่ระหว่าง 80H-0CFH และถ้าบิต N เป็น “1” แอดเดรสของ DDRAM จะมี 2 ช่วงคือ 8CH-87H และ 0C0H-0C7H

9. คำสั่งอ่านแฟลก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบคำสั่งดังนี้

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 ว่าพร้อมจะรับข้อมูลอยู่หรือไม่ ถ้าหากบิต BF เป็น “0” แสดงว่าตัวควบคุม LCD พร้อมรับข้อมูลหรือคำสั่ง แต่ถ้าเป็น “1” แสดงว่า ขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการทำงานภายในหรือกำลังประมวลผลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง เมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา R/W เป็น “1” ด้วย แต่สัญญาณที่ RS ยังต้องเป็น “0” อยู่เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง นอกจากนี้ ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วย โดยบิต 0-บิต 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

การเขียนคำสั่งและข้อมูลให้แก่โมดูล LCD

ในการเขียนข้อมูลเพื่อควบคุมให้โมดูล LCD แสดงผลตามที่ผู้ใช้งานต้องการ ต้องส่งคำสั่ง (instruction) แล้วกำหนดโหมดการทำงานให้แก่โมดูล LCD ก่อน จากนั้นจึงค่อยส่งข้อมูล (data) ที่ต้องการแสดงผล เนื่องจากบัสข้อมูลของโมดูล LCD มี 8 เส้นคือ D0-D7 และใช้เป็นทางผ่านของทั้งคำสั่งและข้อมูล ดังนั้นในการส่งคำสั่งและข้อมูลจึงต้องอาศัยการกำหนดสัญญาณลอจิกที่ขา RS ถ้าหากที่ขา RS ได้ลอจิก “0” หมายความว่า ข้อมูลที่ป้อนให้แก่โมดูล LCD ขณะนั้นเป็นคำสั่ง ในทางตรงข้าม หากขา RS ได้รับลอจิก “1” ข้อมูลที่ป้อนให้ขณะนั้นเป็นข้อมูลที่ใช้ในการแสดงผล

เมื่อต้องการเขียนหรืออ่านข้อมูลใน CGRAM และ DDRAM เริ่มต้นต้องกำหนดแอดเดรสที่ต้องการอ่านหรือเขียนก่อน โดยใช้คำสั่งเลือกแอดเดรส จากนั้นกำหนดให้ขา RS เป็น “1” เพื่อแจ้งให้ตัวควบคุมภายในโมดูล LCD ทราบว่าข้อมูลที่ปรากฏต่อไปนี้เป็นข้อมูลปกติไม่ใช่คำสั่ง

ในกรณีที่ต้องการอ่านข้อมูลต้องกำหนดให้ขา R/W เป็น “1” ข้อมูลขนาด 8 บิต (หรือ 4 บิต) ก็จะปรากฏบนบัสข้อมูล โดยข้อมูลที่อ่านออกมาได้จะเป็นข้อมูลจากแอดเดรสของ CGRAM หรือ DDRAM ตามที่ต้องการ

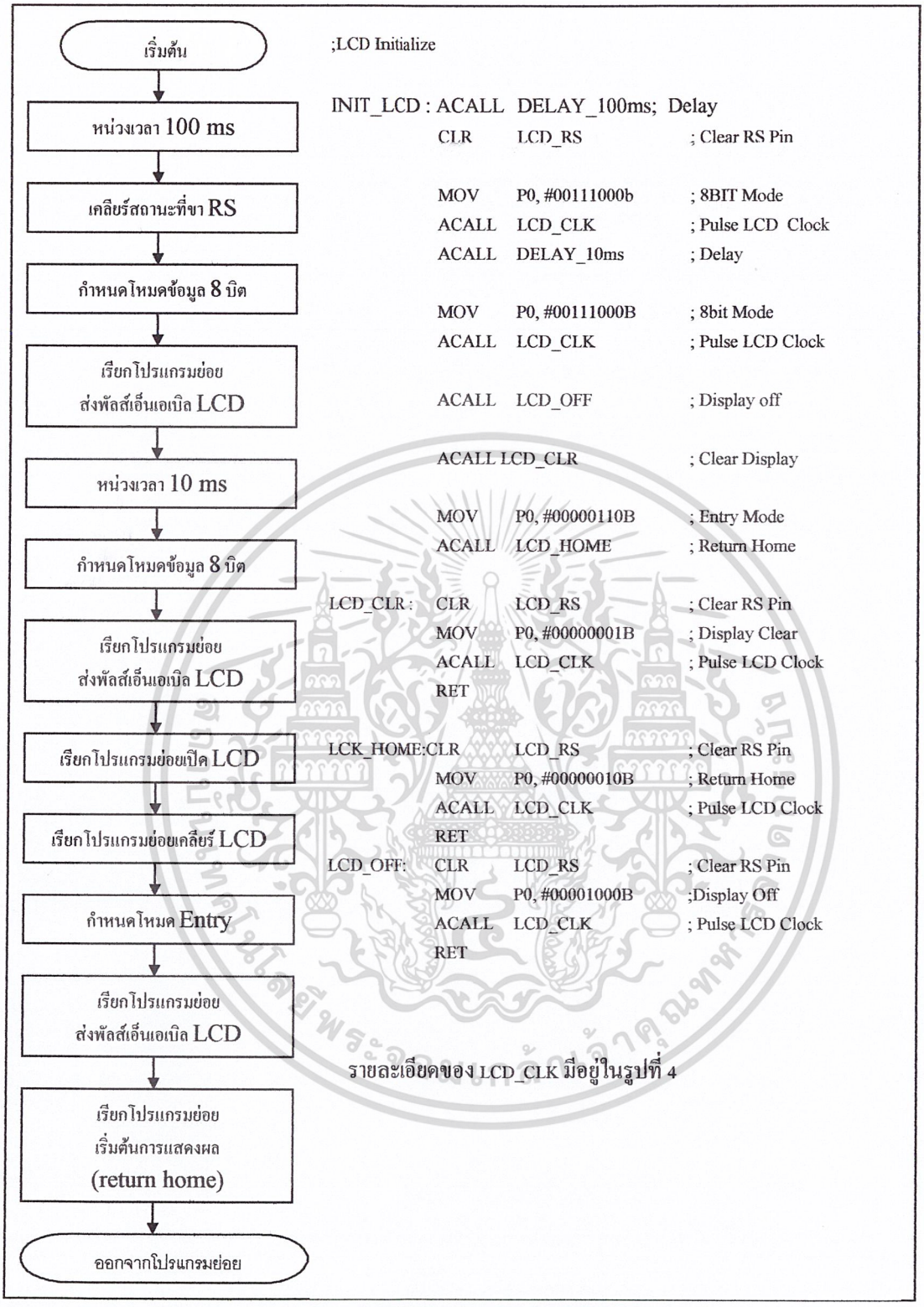
ในกรณีที่ต้องการเขียนข้อมูล เมื่อกำหนดแอดเดรสและป้อนลอจิก “1” ให้ขา RS แล้ว แล้วต้องกำหนดให้ขา R/W เป็น “0” ข้อมูลที่อยู่บนบัสข้อมูลจะถูกเขียนลงในรีจิสเตอร์ DR จากนั้นตึงถ่ายทอดลงใน DDRAM ต่อไป

จังหวะการทำงานของ LCD โมดูล

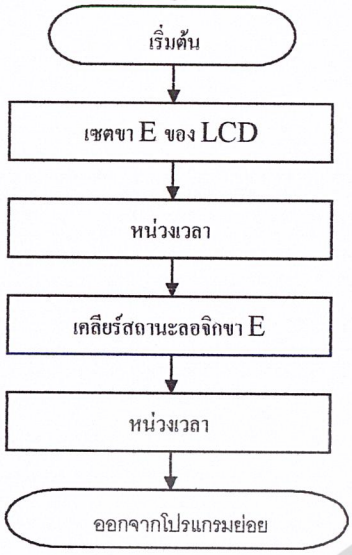
ในการติดต่อกับโมดูล LCD จะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูล เนื่องจากต้องรอให้คอนโทรลเลอร์ภายใน LCD โมดูล แปลความหมายของรหัสคำสั่งและทำงานตามคำสั่งให้เรียบร้อยก่อนจากนั้นจึงจะรับข้อมูลหรือดำเนินการต่อไป

ดังนั้น ในการใช้งานโมดูล LCD ต้องมีโปรแกรมเพื่อหน่วงเวลารอให้โมดูล LCD พร้อมทำงานด้วย โดยเมื่อเริ่มจ่ายไฟให้แก่โมดูล LCD ต้องรอประมาณ 10 มิลลิวินาที เพื่อให้โมดูล LCD ทำการเตรียมความพร้อมหรืออินิเชียล (initial) หลังจากนั้นก็จะกำหนดลอจิกให้แก่ขา RS ของโมดูล LCD แล้วต้องหน่วงเวลาอีกประมาณ 2 มิลลิวินาทีเพื่อให้คอนโทรลเลอร์ใน โมดูล LCD แปลความหมายของลอจิกที่ขา RS ว่า ข้อมูลต่อไปที่จะได้รับนั้นเป็นรหัสคำสั่งหรือเป็นข้อมูลที่ต้องการแสดงผล จากนั้นจะเป็นการส่งข้อมูลมารอที่บัสข้อมูล D0-D7 (กรณีทำงานในโหมด 8 บิต) ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ไปที่ขา E เพื่อเอ็นเอเบิลโมดูล LCD ให้รับข้อมูลจากบัสข้อมูลเข้าไป โดยพลส์ที่ป้อนที่ขา E ของโมดูล LCD ต้องเป็นพัลส์ขอบขาขึ้น จากนั้นทำการหน่วงเวลา 2 มิลลิวินาที

ทั้งหมดที่กล่าวมาคือขั้นตอนและจังหวะในการทำงาน 1 รอบของโมดูล LCD จะเห็นได้ว่ามีโปรแกรมย่อยที่สำคัญอยู่ 3 โปรแกรมย่อยคือ โปรแกรมอินิเชียล LCD , โปรแกรมหน่วงเวลา และโปรแกรมย่อยการส่งพัลส์เพื่อเอ็นเอเบิลโมดูล LCD โปรแกรมตัวอย่างของโปรแกรมย่อยทั้งสามพร้อมทั้งโฟลวชาร์ตแสดงไว้ในรูปที่ 3 และ รูปที่ 4



รูปที่ 4.3 โฟลวชาร์ตและตัวอย่างโปรแกรมย่อยการอินิเชียลโมดูล LCD



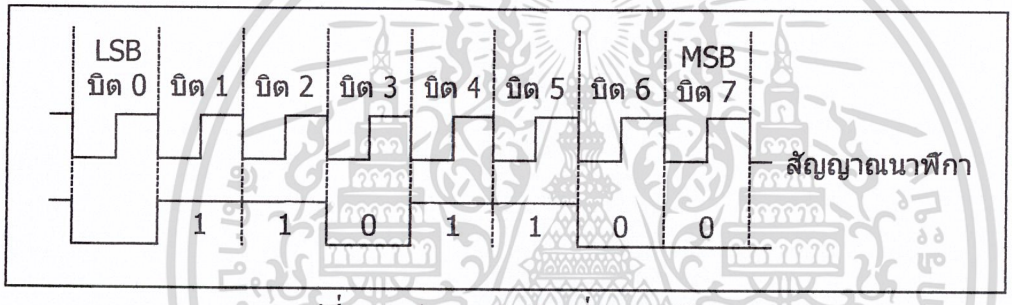

```
LCD_CLK:        SETB    LCD_EN          ; Pulse Clock to E pin
                ACALL   LCD_DELAY
                CLR     LCD_EN
                ACALL   LCD_DELAY
                RET
LCD_DELAY:      MOV     7, #002         ; Do 2 times
LCD_DELAY_1:    MOV     6, #0e6h        ; Each loop = 1 ms
LCD_DELAY_2:    NOP
                NOP
                DJNZ    R6, LCD_DELAY_2
                DJNZ    R7, LCD_DELAY_1
                RET
```

รูปที่ 4.4 โฟลวชาร์ตและตัวอย่างโปรแกรมย่อยส่งพัลส์เอ็นเอเบิลให้แก่โมดูล LCD

บทที่ 5
พอร์ตอนุกรมและโปรแกรม Visual Basic

การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมนั้นจะแบ่งออกได้เป็น 2 แบบคือการสื่อสารอนุกรมแบบซิงโครนัส และการสื่อสารอนุกรมแบบอะซิงโครนัส การสื่อสารแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วย ตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือคีย์บอร์ดของคอมพิวเตอร์ ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกา ส่วนสายอีกเส้นจะเป็นสายของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด 3 เส้นคือ สัญญาณนาฬิกา, ข้อมูล และกราวด์ รูปที่ 1 แสดงให้เห็นถึงไทมิ่งไดอะแกรมของการส่งข้อมูลแบบซิงโครนัส



รูปที่ 5.1 รูปแบบอย่างง่ายที่สุดของข้อมูลอนุกรม

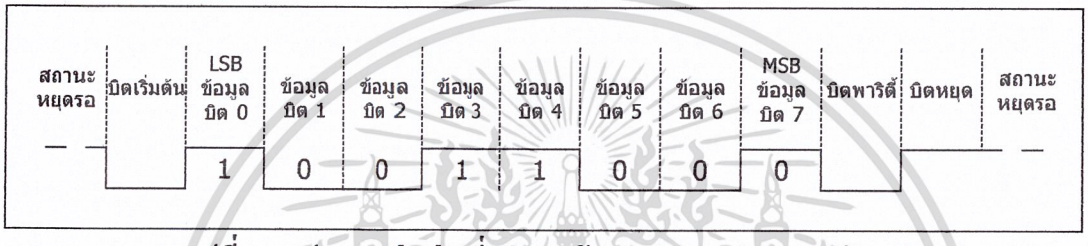
การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารแบบอะซิงโครนัส คือ การรับและส่งข้อมูลไปในสาย โดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยเหมือนกับการรับส่งข้อมูลแบบซิงโครนัส แต่จะใช้การกำหนดค่าสัญญาณนาฬิกาทั้งภาครับและภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณนาฬิกาที่ใช้ในการกำหนดค่าให้ภาครับและภาคส่งนี้ว่า **อัตราการถ่ายทอดข้อมูล** หรือ **บอดเรต (baudrate)** มีหน่วยเป็น *บิตต่อวินาที (bit per secone :bps)*

รูปแบบข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น (Start Bit) ซึ่งจะมีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรมจะมีขนาด 5,6,7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity Bit) จะมีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้าย (Stop Bit) จะมีขนาด 1,1.5 หรือ 2 บิต

รูปที่ 2 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส ซึ่งเมื่อไม่มีข้อมูลที่จะส่งขา DATA จะมีสถานะลอจิก "1" ซึ่งจะเรียกสถานะนี้ว่าสถานะหยุดรอ (waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต ซึ่งจะเรียกบิตนี้ว่าบบิตเริ่มต้น จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุด (LSB) ก่อน ซึ่งข้อมูลในไบต์ที่จะส่งอาจจะมีจำนวนบิต 5,6,7 หรือ 8 บิตก็ได้ จากนั้นจะตามด้วยบิตพาริตี้ ซึ่งใช้เพื่อตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือบิตปิดท้าย ซึ่งจะให้ขาดาต้ามีสถานะลอจิก 1 อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต,1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว



รูปที่ 5.2 รูปแบบอย่างง่ายที่สุดของข้อมูลอนุกรมแบบอะซิงโครนัส

อุปกรณ์พิเศษที่ได้รับการออกแบบมาสำหรับการรับและส่งข้อมูลแบบอะซิงโครนัสเรียกว่า Universal Asynchronous Receiver/Transmitter หรือ UART อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสคือ ค่าบอเรต ซึ่งก็คือค่าจำนวนบิตต่อวินาทีที่ใช้ในการรับและส่งข้อมูล บอดเรตมาตรฐานที่ใช้สำหรับพอร์ตอนุกรม RS-232 ได้แก่ 110, 150, 300 ,600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที และมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ซึ่งการรับส่งแบบอนุกรมโดยไม่ผ่านโมเด็มอาจจะสามารถกำหนดค่าบอดเรตได้สูงถึง 115200 บิต ต่อวินาที เนื่องจากบอดเรตคือจำนวนบิตของข้อมูลที่สามารถถ่ายทอดได้ภายใน 1 วินี ยกตัวอย่าง ข้อมูลอนุกรมถูกส่งในลักษณะ 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูลที่รับส่งนี้เท่ากับ 10 บิต ถ้าใช้บิดเรตในการส่งข้อมูลเท่ากับ 9600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที และถ้ามีการใช้พาริตี้ความเร็วในการรับส่งข้อมูลจะเหลือเป็น 872 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd), แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ การตรวจสอบพาริตี้เป็นการตรวจสอบจำนวนรวมของบิตที่เป็นลอจิก “1” ภายในข้อมูลที่ส่งไป 1 ไบต์ว่ามีจำนวนรวมเป็นเลขคู่หรือเลขคี่โดยต้องรวมบิตพาริตี้เข้าไปด้วย ยกตัวอย่างข้อมูลที่จะทำกาส่งมีขนาด 8 บิตและมีค่าเท่ากับ 99 ฐานสิบหก หรือ 10011001 ฐานสอง จะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก “1” จำนวน 4 ตัวซึ่งเป็นเลขคู่ ค่าในบิตพาริตี้ จะต้องมีลอจิกเป็น “0” แต่ถ้าค่าพาริตี้เป็นคี่ ค่าที่บิตพาริตี้จะต้องเป็น “1” เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริมีจำนวนบิตที่เป็น “1” มีจำนวนรวมกันเป็นเลขคี่ ในตารางที่ 1 แสดงตัวอย่างของบิตพาริตี้ในการรับส่งข้อมูลอนุกรม

ข้อมูล	บิตพาริตี้คู่	บิตพาริตี้คี่
00000000	0	1
00000001	1	0
00000010	1	0
00000011	0	1
00000100	1	0
11111110	0	1
11111111	1	0

ตารางที่ 1 แสดงบิตพาริตี้ของข้อมูล

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART ซึ่งทางภาครับจะต้องทำการกำหนดคุณสมบัติการตรวจสอบพาริตี้ให้ตรงกันว่าจะตรงกันว่าจะตรวจสอบพาริตี้คี่หรือพาริตี้คู่ จากนั้นภาครับของ UART จะทำการตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก “1” ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ นับเป็นการตรวจสอบความผิดพลาดที่เกิดขึ้นในการถ่ายทอดข้อมูลที่ง่ายที่สุด แต่จะเชื่อถือได้เมื่อมีบิตข้อมูลที่ทำการส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ UART เบอร์ 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่น XT ใช้ UART เบอร์ 8250 UART ชิปเหล่านี้มีระดับแรงดันเป็นแบบทีทีแอล (0 และ +5V) แต่เพื่อให้มีแรงดันเป็นไปตามมาตรฐาน RS-232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะทางไกลมากขึ้น ระดับแรงดันทีทีแอลจะถูกแปลงเป็นระดับแรงดันที่สูงขึ้น โดยลอจิก “0” มีระดับแรงดัน +3V ถึง +12V ในขณะที่ลอจิก “1” มีระดับแรงดัน –3V จนถึง –12V

มาตรฐานพอร์ตอนุกรมแบบ RS–232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS–232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS–232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลจากคอมพิวเตอร์ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้สื่อสารผ่านสายทาศัพท์ไปยังคอมพิวเตอร์อีกชุดซค่งอยู่ห่างไกลกัน โดยคณะกรรมการที่เรียกว่าสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association :EIA) ได้วางมาตรฐานที่มีชื่อเรียกกันว่า EIA RS–232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB–25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ –3 ถึง –12V แสดงว่ามีข้อมูล (Mark) และ +3 ถึง +12V แสดงว่าเป็นช่องว่าง (Space)

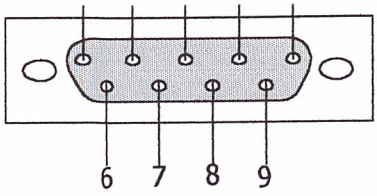
มาตรฐาน RS–232 ได้กำหนดรูปแบบของอุปกรณ์เชื่อมต่อข้มูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating : DEC) ไว้ว่า อุปกรณ์ DET จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์ DCE จะทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น โดยการรับส่งข้อมูลระหว่างอุปกรณ์ทั้งสอบจะกระทำผ่านมาตรฐาน RS-232

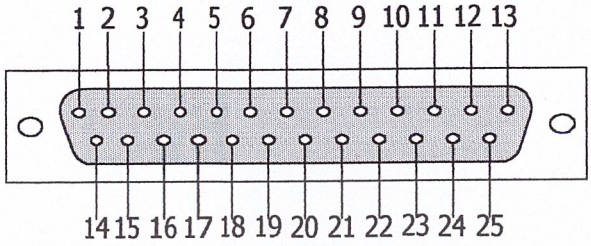
ข้อแตกต่างของอุปกรณ์ DTE และอุปกรณ์ DCE อย่างหนึ่งที่เห็นได้ชัดคือ คอนเน็กเตอร์ของ DTE จะเป็นตัวผู้ ส่วนคอนเน็กเตอร์ของ DCE จะเป็นตัวเมีย ซึ่งพอร์ตอนุกรมของคอมพิวเตอร์ที่ใช้กันอยู่ทั่วไปจะเป็นแบบ DTE ส่วนคอนเน็กเตอร์ที่อยู่ที่โมเด็มจะเป็นแบบ DCE

สำหรับการใช้งานบนคอมพิวเตอร์ พอร์ตอนุกรม RS–232 มักถูกใช้เชื่อมต่อกับโมเด็มหรือเมาส์ โดยสามารถรับส่งข้อมูลได้ที่ความยาวของายสัญญาณสูงสุดถึง 20 เมตร

คอนเน็กเตอร์สำหรับพอร์ต RS – 232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้หรือ DB-9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้นเช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่น ๆ ที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไป โดยแสดงรูปร่างและตำแหน่งขาในรูปที่ 3



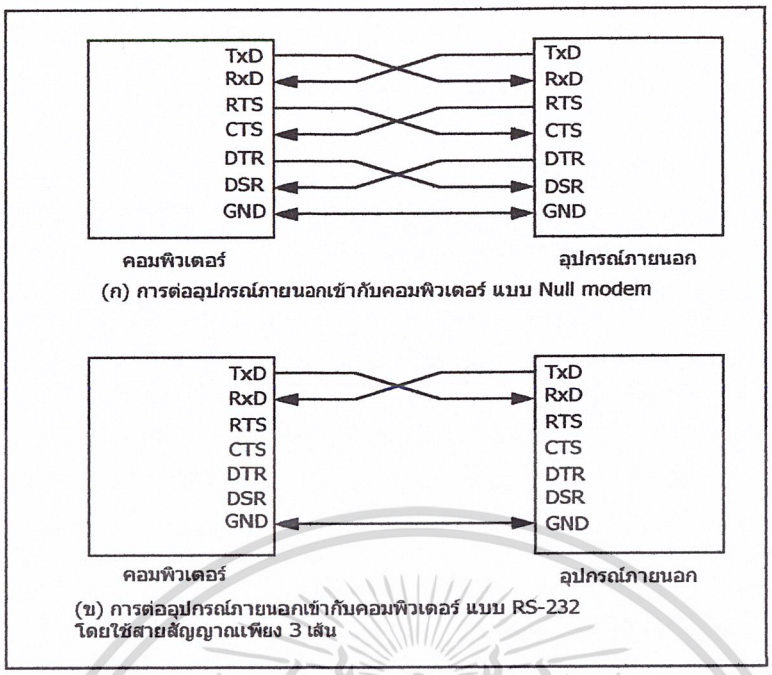


(ข) คอนเน็กเตอร์อนุกรม 25 ขาแรือแบบ DB-25 (มองจากด้านหลังคอมพิวเตอร์)

คอนเน็กเตอร์ DB-9	คอนเน็กเตอร์ DB-25	ชื่อของสายสัญญาณ	ชนิดของสายสัญญาณ
1	8	Data Carrier Detect : DCD	อินพุต
2	3	Received Data : RxD	อินพุต
3	2	Transmitted Data : TxD	เอาต์พุต
4	20	Data Terminal Ready : DTR	เอาต์พุต
5	7	Signal Ground : GND	-
6	6	Data Set Ready : DSR	อินพุต
7	4	Request To Send : RTS	เอาต์พุต
8	5	Clear To Send : CTS	อินพุต
9	22	Ring Indicator : RI	อินพุต

รูปที่5.3 การจัดขาของคอนเน็กเตอร์พอร์ตอนุกรมตามมาตรฐาน RS-232 ทั้งแบบ DB-9 และ DB-25

สำหรับการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกแสดงดังในรูปที่ 4 ลูกศรในรูปแสดงถึงทิศทางของข้อมูล ในรูปที่ 4 (ก) เป็นการเชื่อมต่อแบบ Null modem หรือการเชื่อมต่อโดยตรงโดยไม่ต้องผ่านโมเด็ม โดยมีการตรวจสอบหรือแฮนด์เช็กเต็มรูปแบบ ส่วนในรูปที่ 4 (ข) เป็นการเชื่อมต่อแบบ Null modem ในลักษณะที่ใช้สายสัญญาณเพียง 3 เส้นโดยเส้นหนึ่งสำหรับส่งข้อมูล อีกเส้นสำหรับรับข้อมูล และเส้นสุดท้ายเป็นกราวด์ สำหรับรายละเอียดหน้าที่การทำงานในแต่ละขาของพอร์ตอนุกรม RS-232 มีดังนี้



รูปที่ 5.4 การต่ออุปกรณ์ภายนอกกับพอร์ตอนุกรมของคอมพิวเตอร์ในลักษณะต่าง ๆ

- **Data Carrier Detect : DCD** หรืออาจเรียกว่า Carrier Detect : CD ขานี้จะแอกตีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก
- **Receive Data : RD หรือ RxD** ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสเตอร์บัฟเฟอร์
- **Transmited Data :Td หรือ TxD** ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลทที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป
- **Data Terminal Ready : DTR** เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่า ต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อต่อกับขา DSR ของคอมพิวเตอร์ถ้าใช้การเชื่อมต่อเป็นแบบ Null Midem ซึ่งใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกันและต้องต่อกับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์
- **Signal Ground :GND** ขากราวด์ของระบบ
- **Dta Set Ready : DSR** ขานี้จะใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR

- **Request To Send : RTS** เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบ Null modem 3 สาย จะต้องเชื่อมต่อขา RST ก็คือขา CTS ของตัวมันเองเข้าด้วยกัน เพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา
- **Clear To Send : CTS** ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าดพร้อมที่จะรับข้อมูลหรือไม่
- **Ring Indicator : RI** ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น

UART

UART มาจากคำว่า Universal Asynchronous Receiver Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัสนั่นเอง สำหรับการสื่อสารอนุกรมบนคอมพิวเตอร์แล้ว UART ถือว่าเป็นหัวใจสำคัญของการสื่อสารอนุกรม

หน้าที่หลักของ UART คือทำหน้าที่แปลงข้อมูลที่อยู่ในรูปแบบขนานจากคอมพิวเตอร์ให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัส แล้วส่งออกไป และทำหน้าที่แปลงสัญญาณอนุกรมแบบอะซิงโครนัส ที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่คอมพิวเตอร์ ซึ่งนอกจาก UART จะส่งข้อมูลไปยังคอมพิวเตอร์แล้ว ยังแจ้งข้อมูลอื่น ๆ ให้คอมพิวเตอร์รับทราบด้วย เช่น อัตราเร็วในการรับส่งข้อมูล (บอดเรต), รูปแบบการส่งข้อมูล, ความผิดพลาดที่เกิดขึ้นระหว่างการถ่ายทอดข้อมูล (ผิดพลาดจากพาริตี้,เฟรมข้อมูล,โอเวอร์รัน) เป็นต้น

ชนิดของ UART

ในเครื่องคอมพิวเตอร์ทั่วไปมี UART ที่ใช้งานกันอยู่ 2 เบอร์คือ 8250 ซึ่งเป็น UART มาตรฐานที่มีใช้กันมายาวนาน UART เบอร์นี้จะมีบัฟเฟอร์สำหรับรับและส่งข้อมูลตำแหน่งเดียวกัน ทำให้การรับและส่งข้อมูลถูกจำกัดความเร็วอยู่ที่ 57.6 กิโลบิตต่อวินาทีเท่านั้น แต่ UART เบอร์นี้ก็ถือว่าเป็นต้นแบบของ UART ที่ใช้ในคอมพิวเตอร์ โดยคอมพิวเตอร์ทุก ๆ รุ่นจะต้องสนับสนุนการทำงานตามรูปแบบของ UART เบอร์นี้

UART อีกเบอร์หนึ่งคือ 16450 มีความสามารถรับส่งข้อมูลได้ที่ความเร็ว 1152,200 บิตต่อวินาที และเพิ่มรีจิสเตอร์สำหรับพักข้อมูลสำหรับ UART นอกจากนั้นยังเพิ่มส่วนของชิฟต์รีจิสเตอร์แบบ FIFO (First In First Out) ขนาด 16 ไบต์เข้าไป ทำให้สามารถสนับสนุนความเร็วในการรับส่งข้อมูลที่ 256 กิโลบิตต่อวินาทีได้ โดยคอมพิวเตอร์ในปัจจุบันใช้ UART เบอร์นี้หรือใหม่กว่า เช่น เบอร์ TL16C750 ซึ่งมีรีจิสเตอร์แบบ FIFO ขนาด 64 ไบต์ ทำงานได้ที่ระดับแรงดัน +5V และ +3V มีโหมดประหยัดพลังงาน สามารถรับส่งข้อูลได้ที่ความเร็ว 1 เมกะบิตต่อวินาที เมื่อใช้สัญญาณนาฬิกา 16 MHz

วงจรภายในและรีจิสเตอร์ของพอร์ตอนุกรม RS-232

เครื่องคอมพิวเตอร์โดยทั่วไปสามารถต่อพอร์ตอนุกรม RS-232 สูงสุดได้ 4 พอร์ต ซึ่งจะมีชื่อเรียกเป็น COM1,COM2,COM3 และ COM4 ซึ่งพอร์ตอนุกรมแต่ละตัวต่างก็ใช้งาน UART ภายในคอมพิวเตอร์ในการติดต่อกับอุปกรณ์ภายนอกเช่นเดียวกัน

ในรูปที่ 5 แสดงไดอะแกรมการทำงานภายในของพอร์ตอนุกรม ซึ่งประกอบไปด้วย รีจีสเตอร์ขนาด 8 บิต 8 ตัว ที่ใช้งานร่วมกับ UART แอดเดรสของรีจิสเตอร์ภายในพอร์ตอนุกรมสามารถคำนวนได้จากค่ารีจิสเตอร์พื้นฐานของพอร์อนุกรม ยกตัวอย่าง พอร์ตอนุกรม COM 1 มีแอดเดรสอยู่ที่ 3F8H ตำแหน่งของรีจิสเตอร์ต่าง ๆ จะเป็นตำแหน่งที่บวกเข้าไปกับค่า 3F8H โดยรีจิสเตอร์ที่ใช้งานกับพอร์ตอนุกรมมีดังนี้

00H เป็นรีจิสเตอร์บัฟเฟอร์สำหรับเก็บข้อมูลที่รับเข้ามาหรือเตรียมข้อมูลก่อนที่จะส่งออกไป

01H รีจิสเตอร์อีนาเบิลการอินเตอร์รัปต์ ใช้ในการเซตโหมดการอินเตอร์รัปต์ของพอร์ตอนุกรม

02H รีจิสเตอร์แสดงโหมดการอินเตอร์รัปต์ ใช้เพื่อตรวจสอบโหมดของการอินเตอร์รัปต์เมื่อมีการอินเตอร์รัปต์เกิดขึ้น

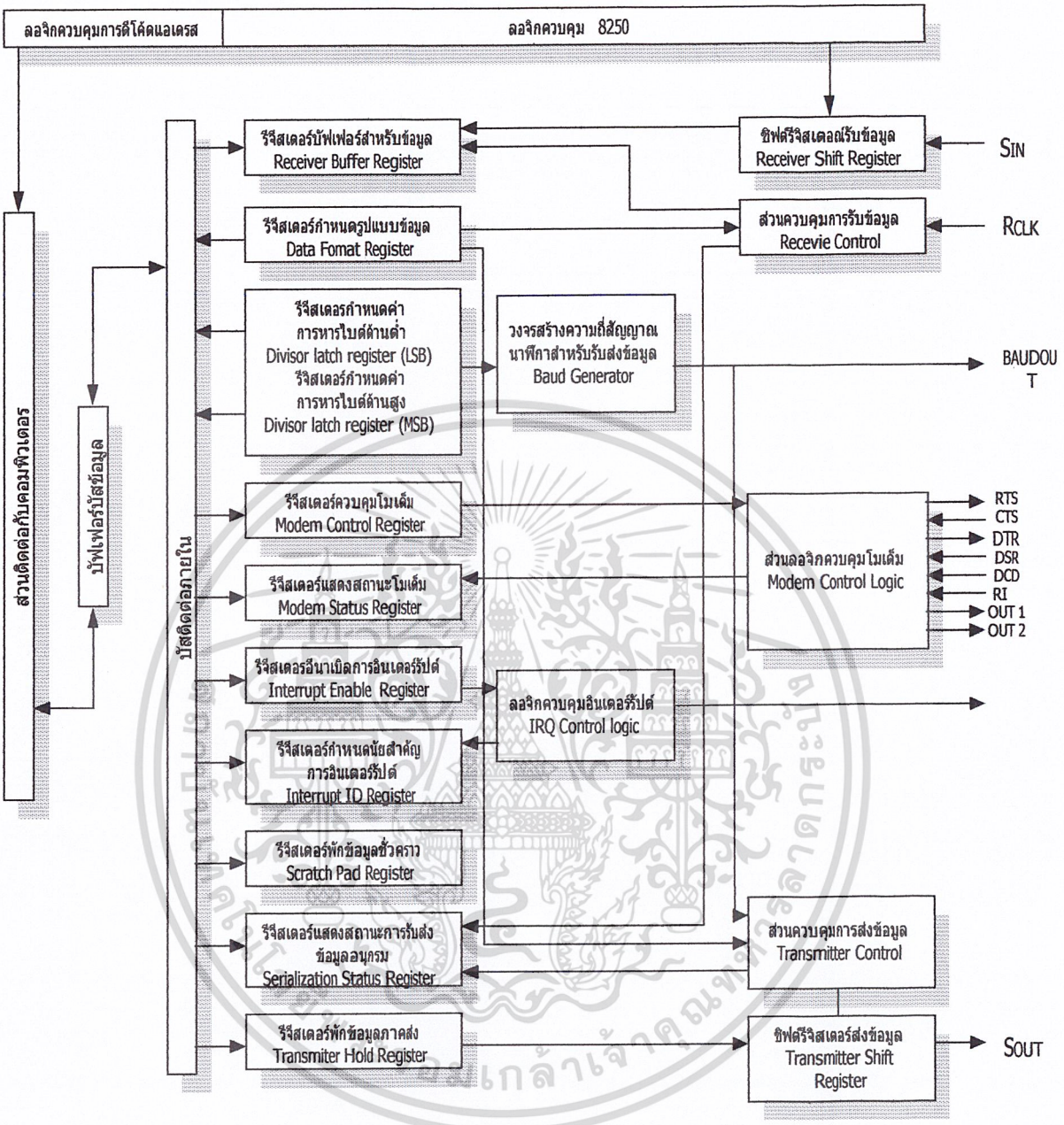
03H รีจิสเตอร์กำหนดรูปแบบของข้อมูล

04H รีจิสเตอร์ควบคุมโมเด็ม ใช้ตรวจสอบบิตสำหรับติดต่อกับโมเด็ม เช่น RTS หรือ DTR

05H รีจิสเตอร์แสดงสถานะการรับและการส่งข้อมูลแบบอนุกรม

06H รีจิสเตอร์แสดงสถานะของโมเด็ม ซึ่งจะแสดงสถานะของขาDCD,RI,DSRและ CTS

07H รีจิสเตอร์สำหรับการเก็บข้อมูลชั่วคราว



รูปที่ 5.5 ไดอะแกรมการทำงานภายในของพอร์ตอนุกรมของเครื่องคอมพิวเตอร์

รีจิสเตอร์ตำแหน่ง 00H : รีจิสเตอร์บัฟเฟอร์

เป็นรีจิสเตอร์สำหรับเก็บข้อมูลที่รับเข้ามาและข้อมูลที่จะส่งออกไป โดยการติดต่อกับรีจิสเตอร์นี้เพื่อเก็บข้อมูลที่ต้องการจะส่งจะต้องกำหนดให้บิต DLAB ในรีจิสเตอร์กำหนดรูปแบบข้อมูล (03H) จะต้องมีสถานะเป็น 0 ซึ่งการเขียนข้อมูลมายังแอดเดรสนี้ เป็นการส่งข้อมูลไปยังรีจิสเตอร์ส่งข้อมูลและข้อมูลจะถูกส่งออกไปแบบอนุกรม สำหรับการรับข้อมูล เมื่อข้อมูลที่รับเข้ามาเรียบร้อยและแปลงเป็นแบบขนานแล้ว ข้อมูลจะถูกส่งมายังรีจิสเตอร์เก็บข้อมูล หลังจากมีการอ่านรีจิสเตอร์นี้ออกไปรีจิสเตอร์นี้จะถูกเคลียร์ และเตรียมพร้อมสำหรับการรับข้อมูลในไบต์ต่อไป

รีจิสเตอร์ตำแหน่ง 01H : รีจิสเตอร์อีนาเบิลการอินเตอร์รัปต์

เป็นรีจิสเตอร์สำหรับการอีนาเบิลการอินเตอร์รัปต์ ซึ่งเป็นการกำหนดให้ UART สร้างสัญญาณอินเตอร์รัปต์ขึ้นมา ฟังก์ชันการทำงานในแต่ละบิตของรีจิสเตอร์นี้มีดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	0	SINP	ERBK	TBE	RxRD

บิต 4-7 บิตเหล่านี้ไม่ถูกใช้งาน กำหนดให้เท่ากับ “0”

SINP อีนาเบิลการอินเตอร์รัปต์เนื่องจากเกิดการเปลี่ยนสถานะที่ขาอินพุต CTS,DSR,DCD หรือขา RI
“1” อีนาเบิลการอินเตอร์รัปต์
“0” ไม่มีการใช้อินเตอร์รัปต์รูปแบบนี้หรือดิสเอเบิล

ERBK อีนาเบิลการอินเตอร์รัปต์เนื่องจากเกิดความผิดพลาดขึ้นด้วยสาเหตุจากพาริตี้,โอเวอร์รัน,เฟรมข้อมูล หรือการเบรกข้อมูล
“1” อีนาเบิลการอินเตอร์รัปต์
“0” ไม่มีการใช้อินเตอร์รัปต์รูปแบบนี้หรือดิสเอเบิล

TBE อีนาเบิลการอินเตอร์รัปต์เมื่อรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง
“1” อีนาเบิลการอินเตอร์รัปต์
“0”ไม่มีการใช้อินเตอร์รัปต์รูปแบบนี้หรือดิสเอเบิล

RxRD อีนาเบิลการอินเตอร์รัปต์เนื่องจากรีจิสเตอร์บัฟเฟอร์ได้รับข้อมูลเรียบร้อย
“1” อีนาเบิลการอินเตอร์รัปต์
“0”ไม่มีการใช้อินเตอร์รัปต์รูปแบบนี้หรือดิสเอเบิล

รีจิสเตอร์ตำแหน่ง 02H : รีจิสเตอร์แสดงโหมดและสถานะการอินเตอร์รัปต์

มีรายละเอียดของแต่ละบิตดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	0	0	ID1	ID0	PND

บิต 3-7 ไม่ได้ใช้งาน อ่านค่าได้เท่ากับ "0"

ID1,ID0 ใช้งานร่วมกันเพื่อแจ้งสาเหตุของการเกิดอินเตอร์รัปต์

"00" เกิดการอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงของขาอินพุตขึ้นการอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 4

"01" เกิดการอินเตอร์รัปต์เนื่องจากรีจิสเตอร์บัฟเฟอร์ส่งข้อมูลว่างขึ้นการอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 3

"10" เกิดการอินเตอร์รัปต์เนื่องจากข้อมูลถูกเก็บลงในรีจิสเรอร์บัฟเฟอร์สำหรับข้อมูลเรียบร้อยแล้ว การอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 2

"11" เกิดการอินเตอร์รัปต์เนื่องจากความผิดพลาดในการถ่ายทอดข้อมูลหรือเกิดการเบรก (break : เกิดการหยุดถ่ายทอดข้อมูลกระทันหัน) การอินเตอร์รัปต์แบบนี้มีนัยสำคัญเป็นอันดับ 1 หรือมีนัยสำคัญสูงสุด

PND ใช้แสดงสถานะของการเกิดอินเตอร์รัปต์

"1" แสดงว่าไม่มีการอินเตอร์รัปต์

"0" แสดงว่ามีการอินเตอร์รัปต์เกิดขึ้น

เมื่อมีการสร้างสัญญาณินเตอร์รัปต์ขึ้น จะต้องมีการเคลียร์ค่าก่อนที่จะให้เกิดอินเตอร์รัปต์ครั้งต่อไป โดยสามารถทำได้ดังนี้คือ

- ถ้าเกิดอินเตอร์รัปต์เนื่องจากการเปลี่ยนแปลงขาอินพุตจะต้องอ่านค่าจากรีจิสเตอร์แสดงสถานะของโมเด็ม (รีจิสเตอร์ตำแหน่ง 06H) เพื่อเคลียร์ค่าการอินเตอร์รัปต์
- ถ้าเกิดการอินเตอร์รัปต์เนื่องจากบัฟเฟอร์ส่งข้อมูลว่าง จะต้องเขียนข้อมูลไปยังรีจิสเตอร์บัฟเฟอร์ส่งข้อมูล (รีจิสเตอร์ตำแหน่ง 00H) หรืออ่านค่ารีจิสเตอร์แสดงสถานะอินเตอร์รัปต์ (รีจิสเตอร์ตำแหน่ง 02H) เพื่อเคลียร์ค่าการอินเตอร์รัปต์
- ถ้าเกิดอินเตอร์รัปต์เนื่องจากการเก็บข้อมูลลงในรีจิสเตอร์บัฟเฟอร์สำหรับข้อมูลเรียบร้อย จะต้องเคลียร์ค่าอินเตอร์รัปต์โดยการอ่านข้อมูลจากรีจิสเตอร์บัฟเฟอร์
- ถ้าเกิดอินเตอร์รัปต์เนื่องจากความผิดพลาดในการรับส่งข้อมูลหรือเกิดการเบรก จะต้องเคลียร์ค่าอินเตอร์รัปต์โดยการอ่านค่ารีจิสเตอร์แสดงสถานะการรับส่งข้อมูลแบบอนุกรม

รีจิสเตอร์ตำแหน่ง 03H : รีจิสเตอร์กำหนดรูปแบบของข้อมูล

มีรายละเอียดหน้าที่ของแต่ละบิตดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
DLAB	**BRK**	**PAR2**	**PAR1**	**PAR0**	**STOP**	**DAB1**	**DAB0**

DLAB ใช้ในการกำหนดหน้าที่การทำงานของรีจิสเตอร์บัฟเฟอร์ (00H)

"1" เป็นการเข้าสู่โหมดการหารค่าบอดเรต

"0" เป็นการเข้าถึงรีจิสเตอร์บัฟเฟอร์ (รีจิสเตอร์ตำแหน่ง 00H) และรีจิสเตอร์สำหรับอีนาเบิลการอินเตอร์รัปต์ (รีจิสเตอร์ตำแหน่ง 01H) เมื่อบิต DLAB เป็น "1" รีจิสเตอร์บัฟเฟอร์ (00H) และรีจิสเตอร์อีนาเบิลการอินเตอร์รัปต์ (01H) จะใช้สำหรับโหลดค่าการหารความถี่สำหรับกำหนดค่าบอดเรต โดยรีจิสเตอร์ 00H เก็บค่าตัวหารไบต์ต่ำ ส่วนรีจิสเตอร์ 01H ใช้เก็บค่าตัวหารไบต์สูง การหาค่าบอดเรตสามารถเขียนเป็นสมการได้ดังนี้

บอดเรต = 115200/ค่าตัวหาร 16 บิต

ค่าตัวเลข 115200 มาจากความถี่ของคริสตอบในวงจร UART ภายในเครื่องคอมพิวเตอร์ โดยคริสตอบที่ใช้มีความถี่ 1.8432 MHz วงจรภายใน UART จะทำการหารค่าความถี่นี้ด้วย 16 ทำให้ได้ค่าความถี่ 115200 Hz ออกมา

*ค่าตัวหาร 16 บิต = ข้อมูลในรีจิสเตอร์ 00H + (256 * ข้อมูลในรีจิสเตอร์ 01H)*

สมมติว่าต้องการค่าบอดเรตเท่ากับ 9600 ค่าตัวหารที่ใช้จะต้องมีค่าเท่ากับ 12 ซึ่งค่านี้จะต้องถูกโหลดลงในรีจิสเตอร์ 00H และโหลดค่า 0 ลงไปในรีจิสเตอร์ 01H ค่าตัวหารที่ทำให้เกิดค่าบอดเรตสูงสุดที่ 115200 บิตต่อวินาที คือ ค่า 0001 นั่นคือรีจิสเตอร์ 00H มีค่าเท่ากับ 1 และ รีจิสเตอร์ 01H มีค่าเท่ากับ 0

BRK ใช้ควบคุมการหยุดถ่ายทอดข้อมูล

"1" สามารถหยุดหรือเบรกได้

"0" ไม่มีการหยุดหรือเบรกได้

PAR2,PAR1,PAR0 ใช้เพื่อกำหนดบิตพาริตี้

"000" ไม่ใช้บิตพาริตี้

"001" กำหนดพารีตี้คี่

"011" กำหนดพารีตี้คู่

"101" มาร์ก (mark)

"111" ช่องว่าง (space)

STOP ใช้กำหนดจำนวนบิตปิดท้าย

"1" มีบิตปิดท้าย 2 บิต

"0" มีบิตปิดท้าย 1 บิต

DAB1,DAB0 ใช้ร่วมกันในการกำหนดจำนวนบิตของข้อมูลที่ต้องการถ่ายทอด

"00" จำนวนบิตข้อมูลเท่ากับ 5 บิต

"01" จำนวนบิตข้อมูลเท่ากับ 6 บิต

"10" จำนวนบิตข้อมูลเท่ากับ 7 บิต

"11" นำนวนบิตข้อมูลเท่ากับ 8 บิต

รีจีสเตอร์ตำแหน่ง 04H : รีจิสเตอร์ควบคุมโมเด็ม

มีรายละเอียดหน้าที่ของแต่ละบิตดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	0	0	LOOP	OUT2	OUT1	RTS	DTR

บิต 5- 7 ไม่มีการใช้งาน อ่านค่าได้เท่ากับ 0

LOOP "1" อีนาเบิลการส่งค่ากลับ

"0" ดิสเอเบิล

OUT1,OUT2 "1" อีนาเบิลการใช้งานภายใน

"0" ดีสเอเบิล

RTS ใช้ควบคุมการทำงานของขา RTS (Read To Send)

"1" อีนาเบิล

"0"ดิสเอเบิล

DTR ใช้ควบคุมการทำงานของขา DTR (Data Terminal Ready)

"1" อีนาเบิล

"0" ดิวเอเบิล

รีจีสเตอร์ตำแหน่ง 05H : รีจิสเตอร์แสดงสถานะการรับและส่งข้อมูลอนุกรมของ UART

ใช้งานร่วมกับรีจิสเตอร์แสดงโหมดและสถานะของการอินเตอร์รัปต์ (รีจิสเตอร์ตำแหน่ง 02H) เพื่อแสดงสาเหตุของการเกิดอินเตอร์รัปต์ มีรายละเอียดหน้าที่ของแต่ละบิตดังนี้

บิต7	**บิต6**	**บิต5**	**บิต4**	**บิต3**	**บิต2**	**บิต1**	**บิต0**
0	**TXE**	**TBE**	**BREK**	**FRME**	**PARE**	**OVFE**	**RxRD**

TXE (Transmitter Empty) "1" แสดงว่ารีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง

"0" แสดงว่ายังคงมีข้อมูล 1 ไบต์เก็บอยู่ในรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล

TBE (Transmitter Buffer Empty) "1" รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง

"0" ยังคงมีข้อมูล 1 ไบต์เก็บอยู่ในรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล

BREK (Break) "1" UART ตรวจพบการเบรก

"0" ไม่มีการเบรก

FRME (Frame Error) "1" UART ตรวจพบความผิดพลาดด้านแฟรมข้อมูล

"0" ไม่พบความผิดพลาดด้านเฟรมข้อมูล

PARE (Parity Errot) "1" UART ตรวจพบความผิดพลาดทางพาริตี้

"0" ไม่พบความผิดพลาดทางพาริตี้

OVRE (Overrun Error) "1" UART ตรวจพบความผิดพลาดแบบโอเวอร์รัน

"0" ไม่พบความผิดพลาดแบบโอเวอร์รัน

RxRD (Received Data Ready) "1" มีการรับข้อมูลเข้ามาเก็บไว้ในบัฟเฟอร์

"0"ไม่มีข้อมูล

รีจิสเตอร์ตำแหน่ง 06H : รีจีสเตอร์แสดงสถานะของโมเด็ม

ใช้เพื่อกำหนดสถานะสัญญาณอินพุต ของพอร์ตอนุกรม RS-232 ซึ่งได้แก่ สัญญาณ DCD, DSR, CTS และ RI สำหรับการเชื่อมต่อใช้งานแบบอเนกประสงค์ ดังมีรายละเอียดหน้าที่ของแต่ละบิตต่อไปนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
DCD	**RI**	**DSR**	**CTS**	**DDCD**	**DRI**	**DDSR**	**DCTS**

DCD ใช้แสดงสภาวะของขา DCD

"1" แสดงว่าที่ขา DCD เป็นลอจิก "1"

"0" แสดงว่าที่ขา DCD เป็นลอจิก "0"

RI ใช้แสดงสภาวะของขา RI

"1" แสดงว่าที่ขา RI เป็นลอจิก "1"

"0" แสดงว่าที่ขา RI เป็นลอจิก "0"

DSR ใช้แสดงสภาวะของขา DSR

"1" แสดงว่าที่ขา DSR เป็นลอจิก "1"

"0" แสดงว่าที่ขา DSR เป็นลอจิก "0"

DCTS (Delta Clear To Send) ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นของบิต CTS

"1" แสดงว่าบิต CTS (Slear To Send) เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DDSR (Delta Data Set Ready) ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นของบิต DSR

"1" แสดงว่าบิต DSR (Data Set Reaey) เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DRI (Ddlta Fing Indicator) ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นของบิต RI

"1" แสดงว่าบิต RI (Finging Indicator) เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DDCD (Delta Data Carrier Detect) ใช้แจ้งการเปลี่ยนแปลงที่เกิดขึ้นของบิต DDCD

"1" แสดงว่าบิต CTS (Clear To Send) เกิดการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DCTS (Delta Clear To Send) ใช้แสดงสภาวะของขา CTS

"1" แสดงว่าที่ขา CTS เป็นลอจิก "1"

"0" แสดงว่าที่ขา CTS เป็นลอจิก "0"

รีจิสเตอร์ตำแหน่ง 07H : รีจิสเตอร์สำหรับเก็บข้อมูลชั่วคราว

ทำหน้าที่เป็นหน่วยความจำแรมขนาด 1 ไบต์ การอ่านและเขียนข้อมูลที่รีจิสเตอร์ตัวนี้ไม่ส่งผลใด ๆ ต่อการใช้งาน UART

ลักษณะสัญญาณอินพุตและเอาต์พุตของพอร์ต RS-232

สัญญาณเอาต์พุตที่ใช้ควบคุม (RTS และ DTR) และสัญญาณแสดงสถานะอินพุต (CTS, DSR และ DCD) ของพอร์ตอนุกรม RS-232 จะถูกกลับสถานะภายในตัว UART ส่วนสัญญาณข้อมูลทั้งภาคส่งและรับจะไม่ถูกกลับสถานะ UART จะให้ระดับสัญญาณเอาต์พุตออกมาเป็นแบบทีทีแอลเท่านั้น ดังนั้นเมื่อสัญญาณถูกส่งออกมาจาก UART จึงต้องส่งเข้าสู่วงจรขับเพื่อปรับระดับแรงดันให้ได้ระดับสัญญาณเป็นไปตามมาตรฐาน RS-232 ก่อนส่งออกไปจากคอมพิวเตอร์สำหรับอุปกรณ์ต่อเชื่อมปลายทางก็จะต้องมีวงจรขับในลักษณะนี้เช่นเดียวกัน เพื่อให้ได้ระดับสัญญาณในระดับเดียวกัน แต่วงจรขับที่ใช้ทั้งภายในคอมพิวเตอร์และอุปกรณ์ต่อเชื่อมปลายทางนั้นจะถูกกลับสถานะ ดังแสดงเป็นบล็อกไดอะแกรมในรูปที่ 6

แอดเดรสของพอร์ตอนุกรม

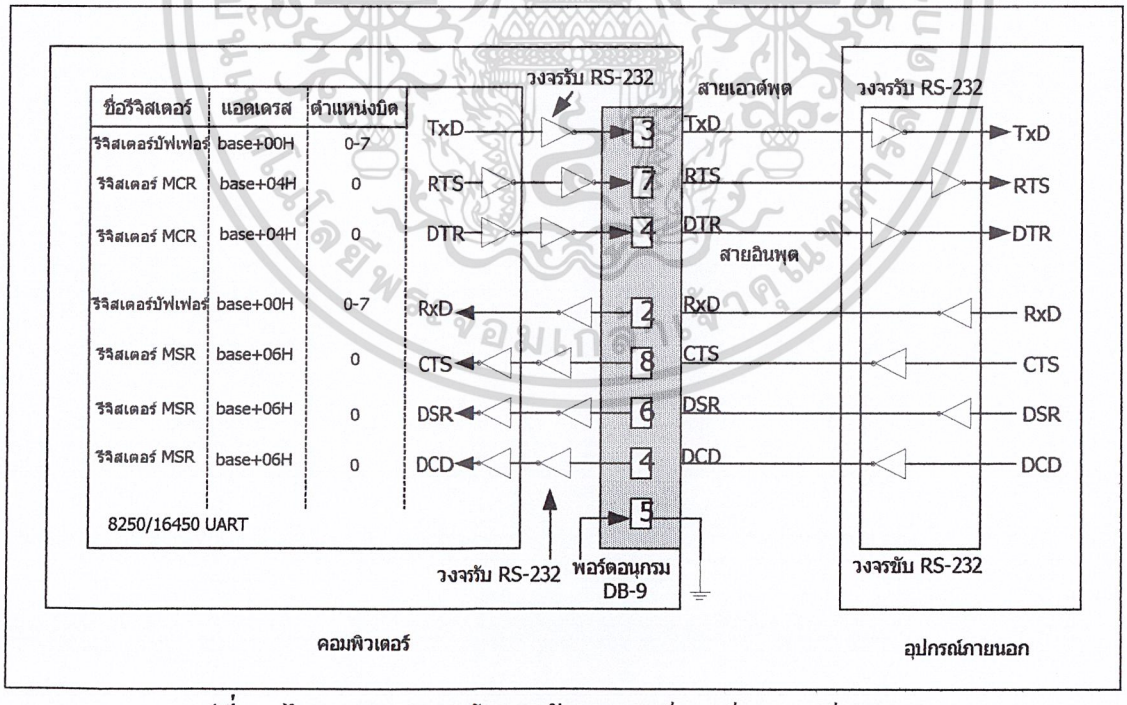
แอดเดรสพื้นฐานของพอร์ตอนุกรมมี 4 ตำแหน่งดังนี้คือ

COM1 : 3F8H

COM2 : 2F8H

COM3 : 3E8H

COM4 : 2E8H



รูปที่ 5.6 ไดอะแกรมแสดงโครงสร้างทางฮาร์ดแวร์ของพอร์ตอนุกรม

เมื่อเริ่มเปิดเครื่องเพื่อใช้งานคอมพิวเตอร์ ไบออสภายในคอมพิวเตอร์จะทำการตรวจสอบแอดเดรสของพอร์ตอนุกรมทั้งหด ถ้าไบออสตรวจพบแอดเดรสของพอร์ตอนุกรมไบออสจะนำแอดเดรสที่ตรวจพบไปเก็บไว้ในหน่วยความจำขนาด 2 ไบต์ สำหรับพอร์ตอนุกรม COM1 จะเก็บไว้ที่แอดเดรส 0000 : 0400H และ 0000 : 0401H ส่วนตำแหน่งอื่น ๆ มีรายละเอียดดังนี้

COM2 = 0000 : 0402H – 0000 : 0403H

COM3 = 0000 : 0404H – 0000 : 0405H

COM4 = 0000 : 0406H – 0000 : 0407H

นอกจากนี้ที่หน่วยความจำแอดเดรส 0000 : 0411H ยังใช้สำหรับแสดงจำนวนของพอร์ตอนุกรมที่มีใช้อยู่ในคอมพิวเตอร์อีกด้วยโดยมีรายละเอียดดังแสดงในตารางที่ 2

บิต3	บิต2	บิต1	จำนวนพอร์ต
0	0	0	ไม่มีพอร์ตอนุกรม
0	0	1	มีพอร์ตอนุกรม 1 พอร์ต
0	1	0	มีพอร์ตอนุกรม 2 พอร์ต
0	1	1	มีพอร์ตอนุกรม 3 พอร์ต
1	0	0	มีพอร์ตอนุกรม 4 พอร์ต

ตารางที่ 2 แสดงข้อมูลในแอดเดรส 0000 :0411H ที่ใช้แจ้งจำนวนพอร์ตอนุกรม

การกำหนดค่าเริ่มต้นให้กับพอร์ตอนุกรม

ก่อนการใช้งานพอร์ตอนุกรมนั้นจะต้องมีการกำหนดค่าเริ่มต้นให้กับตัวมันก่อน ซึ่งก็คือการกำหนดจำนวนบิตข้อมูลที่ต้องการส่ง, จำนวนบิตปิดท้าย, ชนิดของพาริตี้ที่ใช้ และบอดเรต

การกำหนดสามารถทำได้หลายวิธี วิธีแรกเป็นการกำหนดจากดอสพร้อมพ์ โดยใช้คำสั่ง MODE ซึ่งมีวิธีการใช้งานดังนี้

MODE COMm : baud = b, parity = p, data = d, stop = s, retry = r

หรือ MODE COMm : b, p, d, s, r

ตัวอย่าง MODE COM1 : 9600, n, 8,1 จะเป็นการกำหนดให้พอร์ตอนุกรม COM1 มีบอดเรตเท่ากับ 9,600 บิตต่อวินาที ไม่มีการตรวจสอบพาริตี้ รับส่งข้อมูลแบบ 8 บิต และมีบิตปิดท้าย 1 บิต

วิธีที่ 2 เป็นการกำหนดโดยใช้อินเตอร์รัปต์ของดอส ตำแหน่งที่ 14H ซึ่งการใช้งานจะต้องกำหนดค่าต่าง ๆ ลงในรีจิสเตอร์ด้วย โดยจะต้องกำหนดให้รีจิสเตอร์ AH มีค่าเท่ากับ 0 รีจิสเตอร์ DX เก็บค่าของพอร์ตอนุกรมที่ต้องการกำหนดค่าเริ่มต้น โดย

DX = 0 จะกำหนดให้กับพอร์ตอนุกรม COM1

DX = 1 จะกำหนดให้กับพอร์ตอนุกรม COM2

DX = 2 จะกำหนดให้กับพอร์ตอนุกรม COM3

DX = 3 จะกำหนดให้กับพอร์ตอนุกรม COM4

รีจิสเตอร์ AL ซึ่งมีขนาด 8 บิตใช้เก็บค่าเริ่มต้นต่าง ๆ มีรายละเอียดดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
BD2	BD1	BD0	PAR1	PAR0	STOP	DA1	DA0

BD2,BD1,BD0 ใช้สำหรับกำหนดค่าบอดเรต

"111" บอดเรตเท่ากับ 9,600 บิตต่อวินาที

"110" บอดเรตเท่ากับ 4,800 บิตต่อวินาที

"101" บอดเรตเท่ากับ 2,400 บิตต่อวินาที

"100" บอดเรตเท่ากับ 1,200 บิตต่อวินาที

"011" บอดเรตเท่ากับ 600 บิตต่อวินาที

"010" บอดเรตเท่ากับ 300 บิตต่อวินาที

"001" บอดเรตเท่ากับ 150 บิตต่อวินาที

"000" บอดเรตเท่ากับ 110 บิตต่อวินาที

ตัวอย่างโปรแกรมย่อยเทอร์โบปาสคาลสำหรับการกำหนดค่าให้กับพอร์ตอนุกรม โดยกำหนดค่าเริ่มต้นให้กับพอร์ตอนุกรม COM1 มีอัตราบอดเท่ากับ 9600 ไม่มีการตรวจสอบคาริตี้ บิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิตเขียนได้ดังนี้

```
procedue initial;
var regis : registers;
begin
        with regis do begin
                ah := 0;
                al := $0e3;                {11100011 }
                dx := 0;
                intr($14, regis);
        end;
end;
```

PAR1,PAR0 ใช้กำหนดค่าพาริตี้โดย

"00" หรือ "10" ไม่มีการตรวจสอบพาริตี้

"01" พาริตี้คู่

"11" พาริตี้คี่

STOP ใช้กำหนดจำนวนของบิตปิดท้าย

"1" มีบิตปิดท้ายเท่ากับ 2 บิต

"0" มีบิตปิดท้ายเท่ากับ 1 บิต

DA1,DA0 ใช้กำหนดความยาวของข้อมูลโดย

"10" ความยาวข้อมูลเท่ากับ 7 บิต

"11" ความยาวข้อมูลเท่ากับ 8 บิต

การรับและส่งข้อมูลแบบอนุกรม

มีหลากหลายวิธีในการรับและส่งข้อมูลแบบอนุกรมผ่านพอร์ตอนุกรม RS-232 เช่น ใช้คำสั่งพิมพ์ออกทางเครื่องพิมพ์ เรียกอินเตอร์รัปต์ของไบออสหรือของดอส การเขียนหรืออ่านไปยังแอดเดรสของพอร์ตโดยตรง วิธีสุดท้ายเป็นวิธีที่มีความยืดหยุ่นในการใช้งานที่สุด ยกตัวอย่าง ถ้าต้องการส่งข้อมูลไปยังพอร์ตอนุกรม COM1 สามารถเขียนข้อมูลโดยตรงไปที่รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (แอดเดรส 3F8H) โดยใช้คำสั่งภาษา QBASIC ง่าย ๆ ดังนี้

OUT &H3F8,X

ค่า X ในที่นี้หมายถึงข้อมูลที่ต้องการส่ง มีขนาด 8 บิต

สำหรับการอ่านข้อมูลจากพอร์ตอนุกรม จะเป็นการอ่านข้อมูลจากรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (แอดเดรส 3F8H เช่นเดียวกัน) ซึ่งสามารถเขียนโปรแกรมง่าย ๆ ได้ดังนี้

Y = INP (&H3F8)

ค่า Y ในที่นี้คือค่าที่อ่านได้จากรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล โดยมีขนาด 8 บิต

สำหรับการเขียนโปรแกรมด้วย TURBO PASCAL ก็สามารถใช้คำสั่ง

PORT [$3F8] = X

สำหรับการเขียนข้อมูลไปยังพอร์ตอนุกรมและ

Y = PORT [$3F8]

สำหรับการอ่านข้อมูลจากพอร์ตอนุกรม

แต่เมื่อใช้คำสั่งนี้ในขณะที่โปรแกรมทำงานผ่านระบบปฏิบัติการวินโดวส์ จะไม่สามารถใช้งานได้เนื่องจากระบบปฏิบัติการวินโดว์ ได้เข้าฝังตัวพอร์ตอนุกรมเข้าเป็นส่วนหนึ่งของระบบปฏิบัติการแล้ว ดังนั้นการเรียกใช้งานจึงจำเป็นต้องเรียกผ่านเครื่องมือที่ติดต่อผ่านระบบปฏิบัติการ เช่น การใช้คอนโทรล MSCOMM32.OCX ของโปรแกรม Visual BASIC

คอนโทรล MSComm

สำหรับการใช้งาน Visual BASIC ตั้งแต่เวอร์ชัน 2 เป็นต้นมา ใน Visual BASIC จะมีคัสตอมคอนโทรลสำหรับการสื่อสารอนุกรมผ่านทางพอร์ตอนุกรมของคอมพิวเตอร์มาให้โดยใน Visual BASIC เวอร์ชัน 2 และเวอร์ชัน 3 จะใช้ชื่อว่า **MSCOMM.VBX** ส่วนเวอร์ชัน 4 ใช้ชื่อว่า **MSCOMM16.OCX** สำหรับการทำงานกับระบบปฏิบัติการ 16 บิต และ **MSCO32.OCX** สำหรับการทำงานกับระบบปฏิบัติการ 32 บิต สำหรับใน Visual BASIC เวอร์ชัน 5 จะมีเพียง **MSCOMM32.OCX** เท่านั้นเพราะถูกออกแบบมาให้ใช้งานกับระบบปฏิบัติการ 32 บิต

MSComm จัดเตรียมทางเลือกเอาไว้ 2 ทางเพื่อความสะดวกในการสื่อสารข้อมูล ทางแรกคือการสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์ (event-driven communications) เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสนองแบบทันทีทันใด เช่น เมื่อตัวอักษรถูกส่งมาที่พอร์ตอนุกรมหรือเกิดการเปลี่ยนแปลงที่ขา Data Carier Detect (DCD) หรือขา Request To Send (RTS) เหตุการณ์ Oncomm ของ MSComm จะสามารถตรวจจับสัญญาณนั้นได้ทันที ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อคุณสมบัติ CommEvent ต่อไป ส่วนทางเลือกที่สองเป็นการคอยตรวจสอบค่าเหตุการณ์และความผิดพลาดที่เกิดขึ้นด้วยการดูค่าที่เปลี่ยนแปลงในคุณสมบัติ CommEvent หลังจากให้โปรแกรมทำงานในฟังก์ชั่นต่าง ๆ ไปเรียบร้อยแล้ว ซึ่งวิธีนี้ใช้งานได้ดีในกรณีที่โปรแกรมมีขนาดเล็ก

คอนโทรล MSComm 1 ตัวสามารถควบคุมการทำงานของพอร์ตอนุกรมได้ 1 พอร์ต ถ้าในโปรแกรมที่ใช้งานต้องการติดต่อกับพอร์ตอนุกรมมากกว่า 1 พอร์ตจะต้องใช้คอนโทรล MSComm มากกว่า 1 ตัวเพื่อควบคุมพอร์ตอนุกรมในแต่ละพอร์ต แอดเดรสของพอร์ตอนุกรมและแอดเดรสของการเกิดอินเตอร์รัปต์สามารถเปลี่ยนแปลงได้จากการแก้ไขค่าที่ Control Panel

ถึงแม้ว่า คอนโทรล MSComm จะมีคุณสมบัติ (property) มากมาย แต่สามารถทำความเข้าใจได้ไม่ยากดังนี้

CommPort

ใช้ในการกำหนดและอ่านค่าพอร์ตอนุกรมที่ติดต่ออยู่ (COM1, COM2, COM3, COM4)

รูปแบบการใช้งาน

Object.CommPort[= value]

โดย Value เป็นค่าของพอร์ตอนุกรม ชนิดของข้อมูลเป็น Integer ค่า Value สามารถกำหนดได้ในช่วง 1-16 (ค่าเริ่มต้นกำหนดไว้ที่ 1) เมื่อมีการกำหนดค่าแล้วทำการเปิดพอร์ตโดยใช้คุณสมบัติ PortOpen แต่ว่าพอร์ตนั้นไม่มีอยู่ในระบบ MSComm จะสร้างสัญญาณแสดงข้อผิดพลาด error 68 ขึ้นมา ซึ่งหมายถึง อุปกรณ์ตัวนี้ไม่มีอยู่ในระบบ ดังนั้นการเขียนโปรแกรมจึงจำเป็นต้องกำหนดตำแหน่งของพอร์ตอนุกรมก่อนที่ใช้คำสั่ง OpenPort

Setting

ใช้ในการกำหนดและอ่านค่าอัตราบอด,พาริตี้,จำนวนของบิตข้อมูล,จำนวนของบิตปิดท้าย

รูปแบบการใช้งาน

Object.Settings [= value]

ค่า Value มีชนิดข้อมูลเป็นแบบ String มีรูปแบบเป็น “BBBB,P,D,S” โดย BBBB เป็นค่าอัตราบอด, P เป็นค่าพาริตี้,D เป็นจำนวนของบิตข้อมูล และ S เป็นจำนวนของบิตปิดท้าย ปกติแล้วค่านี้ถูกกำหนดไว้เป็น “9600,N,8,1”

ค่าบอดเรตมาตรฐานที่ใช้กับ MSComm มีดังนี้

110 บิตต่อวินาที
300 บิตต่อวินาที
600 บิตต่อวินาที
1,200 บิตต่อวินาที
2,400 บิตต่อวินาที
9,600 บิตต่อวินาที **(ค่าปกติ)**
14,400 บิตต่อวินาที
19,200 บิตต่อวินาที
28,800 บิตต่อวินาที
38,400 บิตต่อวินาที **(สงวน)**
56,000 บิตต่อวินาที **(สงวน)**
128,000 บิตต่อวินาที **(สงวน)**
256,000 บิตต่อวินาที **(สงวน)**

สำหรับค่ามาตรฐานในการกำหนดค่าพาริตี้มีดังนี้

สัญลักษณ์	**รายละเอียด**
E	พาริตี้คู่ (Even)
M	ลอจิก "1" (MARK)
N	ไม่ใช้ (ค่าปกติ)
O	พาริตี้คี่ (Odd)
S	ลอจิก "0" (Space)

ค่าที่ใช้ในการกำหนดจำนวนบิตมี 5 ค่าคือ 4,5,6,7 และ **8 (เป็นค่าปกติ)**

ค่าที่ระบุจำนวนบิตปิดท้ายมี 3 ค่าคือ **1 (เป็นค่าปกติ)**, 1.5 และ 2

ตัวอย่างการใช้งานคำสั่ง Settings โดยจะเป็นการกำหนดค่าบอดเรตเท่ากับ 9600 ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และ บิตปิดท้าย 1 บิต สามารถเขียนโปรแกรมได้ดังนี้

MSComm1.Settings = "9600,N,8,1"

หมายเหตุ สาเหตุที่ค่าที่กำหนดจะต้องอยู่ภายในเครื่องหมายคำพูด "" เนื่องจาก ค่าที่กำหนดนี้อยู่ในรูปตัวแปร String

PortOpen

ใช้ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรม

รูปแบบการใช้งาน

Object.PortOpen [= value]

ค่า Value มีชนิดข้อมูลเป็นแบบบูลีนคือ True กับ False โดย True หมายถึงการเปิดพอร์ตอนุกรมและ False หมายถึงการปิดพอร์ตอนุกรม สำหรับการปิดพอร์ตนั้นจะมีการเคลียร์บัฟเฟอร์รับข้อมูลและบัฟเฟอร์ส่งข้อมูลด้วย คอนโทรล MSComm จะปิดพอร์ตอนุกรมโดยอัตโนมัติเมื่อออกจากโปรแกรม ก่อนที่จะใช้คุณสมบัติ PortOpen ต้องตรวจสอบให้แน่ใจก่อนว่าคุณสมบัติ CommPort นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ถูกต้องหรือไม่ มิเช่นนั้น MSComm จะแสดงข้อผิดพลาด Error 68 แจ้งแก่ผู้ใช้งาน หรือถ้าพอร์ตอนุกรมนั้นถูกเปิดเอาไว้แล้ว โปรแกรมก็จะแจ้งข้อผิดพลาดออกมาเช่นเดียวกัน

ถ้าคุณสมบัติ DTREnable หรือ RTSEnable ถูกกำหนดให้เป็น True ก่อนที่จะทำการเปิดพอร์ต ค่าคุณสมบัติของ DTREnable หรือ RTSEnable จะถูกเซตเป็น False หลังจากปิดพอร์ต แต่ถ้าเซตเป็น False หลังจากปิดโปรแกรมแล้ว ค่าที่กำหนดไว้จะเป็นค่าเดิม

ตัวอย่างการใช้คสั่งเปิดพอร์ต เพื่อติดต่อสื่อสารกับพอร์ตอนุกรม COM1 และมีบอดเรต 9,600 บิตต่อวินาที ไม่มีพาริตี้ จำนวนบิตข้อมูล 8 บิต และบิตปิดท้าย 1 บิต มีดังนี้

```
MSComm1.Settings = "9600,n,8,1"
MSComm1.CommPort = 1
MSComm1.PortOpen = True
```

Input

อ่านค่าและลบค่าขบวนข้อมูลจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งาน

Object.Input

คุณสมบัติ InputLen เป็นตัวกำหนดจำนวนของตัวอักษรที่จะอ่านโดยคุณสมบัติ Input การกำหนดค่าให้ InputLen เท่ากับ 0 เป็นการกำหนดให้คุณสมบัติ Input ทำการอ่านค่าข้อมูลเป็นบัฟเฟอร์รับข้อมูลทั้งหมด

คุณสมบัติ InputMode เป็นตัวกำหนดชนิดของข้อมูลที่คุณสมบัติ Input รับเข้ามา ถ้า InputMode ถูกกำหนดเป็น comInputModeText คุณสมบัติ Input จะส่งค่าข้อมูลกลับมาในรูปแบบของข้อความชนิดข้อมูลเป็นแบบ Variant ถ้า InputMode กำหนดเป็น comInputModeBinary คุณสมบัติ Input จะส่งข้อมูลกลับมาในรูปของไบนารี่และชนิดข้อมูลเป็นแบบ Variant

ตัวอย่างโปรแกรมแสดงให้เห็นถึงวิธีในการรับข้อมูลจากบัฟเฟอร์รับข้อมูลทั้งหมด

```
Private Sub Command1_Click()
Dim Instring as String
        MSComm1.UboytKeb = 0            , Retrieve all available data.
        If MSComm1.InBufferCount Then   , Check for data.
        InString = MSComm1.Input        , Read data.
        End If
End Sub
```

InBufferCount

ส่งค่าจำนวนของตัวอักษรที่อยู่ในบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

Object.InBufferCount [= balue]

คำสั่ง InBufferCount จะแสดงค่าจำนวนของตัวอักษร ซึ่งรับมาจากภายนอกและยังเก็บอยู่ในบัฟเฟอร์ภาครับ เพื่อให้ผู้ใช้งานอ่านค่าออกไป สำหรับการเคลียร์ค่าบัฟเฟอร์ภาครับทำได้โดยกำหนดให้ InBufferCount มีค่าเป็น 0

หมายเหตุ อย่าสับสนระหว่างคำสั่ง InBufferSize และ InBufferCount คำสั่ง InBufferSize นั่นใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ

InBufferSize

กำหนดและคืนค่าขนาดของบัฟเฟอร์ภาครับในหน่วยเป็นไบต์

รูปแบบการใช้งานคำสั่ง

Object.InBufferSize [= value]

คำสั่ง InBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาครับ โดยค่าเริ่มต้นถูกกำหนดไว้ที่ 1,024 ไบต์

หมายเหตุ การกำหนดค่าบัฟเฟอร์ภาครับขนาดใหญ่จะทำให้หน่วยความจำที่เหลือสำหรับการใช้งานส่วนอื่น ๆ จะเหลือน้อย อย่างไรก็ตามการกำหนดค่า บัฟเฟอร์ภาครับที่น้อยเกินไปจะทำให้เกิดการโอเวอร์โฟลวหรือข้อมูลล้นบัฟเฟอร์ เว้นแต่จะมีการใช้แฮนด์เช็ก ดังนั้นค่าปานกลางที่เหมาะสมก็คือค่า 1,024 ซึ่งเป็นค่าเริ่มต้นนั่นเอง แต่ถ้าโปรแกรมมีการเกิดโอเวอร์โฟลวแล้วจึงค่อนปรับเพิ่มค่าขนาดของบัฟเฟอร์ให้มีค่ามากขึ้น

InputLen

กำหนดค่าและคืนค่าจำนวนของตัวอักษรที่อ่านจากบัฟเฟอร์ภาครับ

รูปแบบการใช้งานคำสั่ง

Object.InputLen [= value]

ค่าเริ่มต้นของคุณสมบัติ InputLen มีค่าเท่ากับ “0” การกำหนดค่าเท่ากับ “0” จะทำให้คำสั่ง Input ของ MSComm อ่านค่าข้อมูลที่อยู่ภายในบัฟเฟอร์ภาครับทั้งหมด

ถ้าไม่มีข้อมูลอยู่ในบัฟเฟอร์ภาครับมากเท่ากับจำนวน InputLen คำสั่ง Input จะส่งค่าว่าง (“”) กลับออกมา ผู้ใช้งานสามารถตรวจสอบข้อมูลในบัฟเฟอร์ภาครับได้โดยใช้คุณสมบัติ InBufferCount โดยกำหนดให้มีข้อมูลอยู่ในบัฟเฟอร์ภาครับก่อนแล้วจึงค่อยอ่านข้อมูลจากบัฟเฟอร์ภาครับ

คุณสมบัตินี้มักใช้กับการอ่านค่าข้อมูลจากเครื่องมือหรือเครื่องจักรที่มีการกำหนดค่าขนาดความยาวของข้อมูลเอาไว้แล้ว

ตัวอย่างโปรแกรมการอ่านค่าตัวอักษรออกมา 10 ตัวอักษร

```
Private Command1_Click()
Dim CommData as String
MSComm1.InputLen = 10          , Specify a 10 character block of data.
CommData = MSComm1.Input    . Read data.
End Sub
```

InputMode

กำหนดค่าและคืนค่าชนิดของข้อมูลที่รับโดยคำสั่ง Input

รูปแบบการใช้งานคำสั่ง

Object.InputMode ใช้กำหนดว่าข้อมูลชนิดไหนที่รับเข้ามาผ่านคำสั่ง Input โดยข้อมูลจะเลือกได้ 2 ประเภทคือ

ComInputModeText สำหรับข้อมูลที่อยู่ในรูปข้อความตัวอักษรตามมาตรฐาน ANSI โดยจะต้องกำหนดค่าเป็น "0" และค่าเริ่มต้นของการรับค่าข้อมูลก็จะเป็นค่านี้

ComInputModeBinary สำหรับข้อมูลอื่น ๆ ซึ่งจะเก็บในรูปไบนารี่รวมกันอยู่เป็นไบต์ข้อมูล

ตัวอย่างการใช้งาน InputMode ต่อไปนี้จะทำการอ่านข้อมูล 10 ไบต์จากพอร์ตอนุกรมและเก็บข้อมูลไว้ในตัวแปรแบบอาเรย์ชนิดข้อมูลเป็นแบบไบต์

```
Private Sub Command1_Click()
Dim Buffer as Variant
Dim Arr() as Byte
MSComm1.CommPort = 1          ' Set and open port
MSComm1.PortOpen = True
MSComm1.InputMode = comInputModeBinary    ' Set InputMode to read binary data
Do Until MSComm1.InBofferCount < 10       ' Wait until 10 bttes are in the
Input buffer
        DoEvents
Loop
Buffer = MSComm1.Input                    ' Store binary data in buffer
Arr = Buffer                              ' Assign to byte array for processing
End Sub
```

Output

ใช้ในการส่งขบวนของข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล

รูปแบบการใช้งาน

Object.Output [= balue]

ค่า Value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ Output สามารถใช้ในการส่งข้อมูลตัวอักษรหรือข้อมูลไบนารี่ก็ได้ โดยการส่งข้อมูลเป็นรูปแบบตัวอักษรจะต้องกำหนดข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ String สำหรับการส่งข้อมูลไปนารี่จะต้องกำหนดชนิดของข้อมูลเป็นแบบ Variant และมีข้อมูลภายในเป็นแบบ Byte

ตัวอย่างโปรแกรมการส่งค่าที่ป้อนจากคีย์บอร์ดไปยังพอร์ตอนุกรมโดยใช้ Output

```
Private Sub Form_KeyPress (KeyAscii Ass Interger)
    Dim Buffer as Variant
    MSComm1.CommPort = 1              ' Use COM1
    MSComm1.PortOpen = True           ' Open port
    Buffer = Chr$ (DeyAscii)
    MSComm1.Output = Buffer           ' Send DATA
End Sub
```

OutBufferCount

คืนค่าจำนวนของข้อมูลตัวอักษรที่เก็บอยู่ในบัฟเฟอร์ภาคส่งและสามารถใช้คำสั่งนี้เพื่อเคลียร์บัฟเฟอร์ภาคส่งได้ด้วย

รูปแบบการใช้งานคำสั่ง

Object.OutBufferCount [= value]

ผู้ใช้งานสามารถเคลียร์บัฟเฟอร์ภาคส่งได้โดยการกำหนดค่า OutBufferCount เท่ากับ "0"

หมายเหตุ ระวังการสับสนระหว่างคุณสมบัติ OutBufferCount กับ OutBufferSize ซึ่ง OutBufferSize ใช้เพื่อกำหนดขนาดของบัฟเฟอร์ภาคส่ง

OutBufferSize

กำหนดค่าและคืนค่าขนาดของบัฟเฟอร์ภาคส่ง ชนิดตัวแปรเป็นแบบไบต์

รูปแบบการใช้งานคำสั่ง

Object.OutBufferSize [= object]

คุณสมบัติ OutBufferSize ใช้สำหรับกำหนดขนาดของบัฟเฟอร์ภาคส่ง โดยค่าปกติที่ใช้งานจะมีค่าเท่ากับ 512 ไบต์

หมายเหตุ การกำหนดค่าบัฟเฟอร์ภาคส่งที่มากเกินไปจะทำให้ มีหน่วยความจำเหลือให้ใช้งานน้อย แต่อย่างไรก็ตามถ้ากำหนดค่าน้อยเกินไป จะทำให้เกิดข้อมูลล้นบัฟเฟอร์ขึ้นได้ ยกเว้นจะมีการใช้ แฮนด์เช็ค วิธีการที่ถูกต้องในการกำหนดค่าคือ ทดลองใช้ค่าเริ่มต้นคือค่า 512 ไบต์ดูก่อน ถ้าโปรแกรมทำงานแล้วเกิดการล้นของข้อมูลค่อยเพิ่มค่าของ OutBufferSize ให้มากขึ้น

ParityReplace

กำหนดและคืนค่าตัวอักษรที่ไปวางแทนในตำแหน่งที่เกิดข้อผิดพลาดจากพาริตี้ฯ

รูปแบบการใช้งานคำสั่ง

Object.ParityReplace [= value]

บิตพาริตี้เป็นบิตที่ทางภาคส่งข้อมูลทำการส่งมาพร้อมกับข้อมูลเพื่อตรวจสอบข้อผิดพลาดของข้อมูล โดยเมื่อมีการใช้บิตพาริตี้ คอนโทรล MSCOM จะทำการบวกบิตทุกบิตที่มีค่าลอจิก “1” ในแต่ละไบต์ และทำการตรวจสอบผลลัพธ์ว่าบิตที่อ่านได้นั้นมีจำนวนลอจิก “1” เป็นเขคู่หรือคี่ และตรงกับค่าที่กำหนดไว้แต่ต้นหรือไม่ ถ้าค่าที่นำมาบวกแล้วมีพาริตี้ไม่ตรงแสดงว่าการับส่งข้อมูลผิดพลาด

การกำหนดค่า เริ่มต้นให้กับ ParityReplace นั้นกำหนดให้ใช้เครื่องหมาย (?) ไปวางไว้ที่ตำแหน่งที่เกิดพาริตี้ผิดพลาด ถ้ากำหนดค่า ParityRe;lace ให้เป็นค่าว่าง (“”) จะเป็นการยกเลิกการใช้งาน ParityReplace และไม่มีการป้อนข้อมูลแทนเมื่อตรวจพบข้อผิดพลาด

ParityReaplace ใช้ชนิดข้อมูลเป็นแบบสตริง แต่จะการกำหนด จะกำหนดได้เพียงไบต์เดียวเท่านั้น ซึ่งจะสามารถใช้ค่าใด ๆ ก็ได้ที่เป็นโค้ด ANSI มีค่าอยู่ระหว่าง 0-255

DTREnable

ใช้ในการกำหนดสถานะลอจิกของขา Data Terminal Teaey (DTR) โดยสัญญาณของขา DTR จะส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อแสดงว่าคอมพิวเตอร์พร้อมที่จะรับข้อมูลแล้ว ชนิดของข้อมูลเป็นแบบบูลีน

รูปแบบการใช้งาน

Object.DTREnable [= value]

ค่า Value เป็นค่าสถานะ True หรือ False เพื่อกำหนดลอจิกของขา DTR ให้เป็น “0” หรือ “1” โดย

True หมายถึง ให้ขา DTR มีลอจิก “1”

False หมายถึง ให้ขา DTR มีลอจิก “0” (เป็นค่าปกติ)

หมายเหตุ เมื่อขา DTR ถูกกำหนดสถานะให้เป็น True ที่ขา DTR จะมีสถานะลอจิก “1” เมื่อทำการเปิดพอร์ตและจะมีสถานะเป็น “0” เมื่อมีการปิดพอร์ต เมื่อขา DTR ถูกกำหนดสถานะเป็น False ที่ขา DTR จะมีสถานะลอจิก “0” ตลอดเวลาไม่ว่าจะใช้คำสั่งเปิดพอร์ตหรือปิดพอร์ต

สำหรับการใช้งานกับโมเด็ม การทำให้ขา DTR เป็นลอจิก “0” จะเป็นการวางหูโทรศัพท์หรือยกเลิกการติดต่อ

RTSEnable

ใช้เพื่อกำหนดสถานะลอจิกให้ขา equest To Send 9rts) โดยขา RTS จะเป็นสัญญาณที่ส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อร้องขอส่งข้อมูล ชนิดของข้อมูลเป็นแบบ Boolean

รูปแบบการใช้งาน

Object.RTSEnable [= value]

ค่า Value เป็นค่าสถานะ True หรือ False เพื่อกำหนดลอจิก “0” หรือ “1” ให้ขา RTS โดย

True หมายถึง ให้ขา RTS มีลอจิก “1”

False หมายถึง ให้ขา RST มีลอจิก “0” (เป็นค่าปกติ)

หมายเหตุ เมื่อขา RTSEnable ถูกกำหนดให้เป็น True ขา RTS จะมีสถานะลอจิก “1” เมื่อเปิดพอร์ตและมีสถานะลอจิก “0” เมื่อปิดพอร์ต

EOFEnable

เป็นการกำหนดให้ MSComm รอสัญญลักษณ์แสดงส่วนท้ายสุดของไฟล์ (End of file : EOF) ระหว่างการรับอินพุตเข้ามา ถ้าพบสัญญลักษณ์ EOF ภาคอินพุตจะหยุดรับข้อมูล และเหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงาน 86Il,[y9b CommEvent จะมีค่าเท่ากับ 7 หรือ ComEvEOF

รูปแบบการใช้งาน

Object.EOFEnable [= value]

โดย value เป็นค่าสถานะ True หรือ False เพื่ออีนาเบิลหรือดิสเอเบิลการทำงานของเหตุการณ์ OnComm เมื่อตรวจพบสัญลักษณ์ EOF โดย

True หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงานด้วย EOF

False หมายถึง เหตุการณ์ OnComm จะไม่ถูกกระตุ้นให้ทำงานด้วย EOF (เป็นค่าปกติ)

เมื่อ EOFEnable กำหนดให้เป็น False ส่วนควบคุมจะไม่มีการตวจสอบสัญลักษณ์ EOF

CTSHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Clear To Send (CTS) ได้ว่ามีสถานะลอจิก “0” หรือ “1” โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CTSHolding เป็น True ขา CTS จะมีสถานะลอจิกเป็น “1” ถ้าค่า CTSHolding เป็น False ขา CTS จะมีสถานะลอจิกเป็น “0”

รูปแบบการใช้งาน

Object.CTSHolding

เมื่อขา CTS เป็นลอจิก “0” (CTSHolding = False) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventCTSTO (Clear To Send Timeout) และกระตุ้นให้เกิดเหตุการณ์ OnComm

CDHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Data Carrier Detect (DCD) ได้ว่ามีสถานะลอจิกเป็น “1” หรือ “0” โดยค่าที่อานได้จะเป็นบูลีน True และ False ถ้าค่า CDHolding เป็น True ขา DCD จะมีสถานะลอจิก “1” ถ้าค่า CDHolding เป็น False ขา DCD จะมีสถานะลอจิก “0”

รูปแบบการใช้งาน

Object.CDHolding

เมื่อขา DCD มีลอจิก “1” (CDHolding = True) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventCDTO (Carrier Detect Timeout Error) และกระตุ้นให้เกิดเหตุการณ์ OnComm

DSRHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา DSR (DSR) ได้ว่ามีสถานะลอจิก “1” หรือ “0” โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า DSRHolding เป็น True ขา DSR จะมีสถานะลอจิก “1” ถ้าค่า DSRHolding เป็น False ขา DSR จะมีสถานะลอจิก “0”

รูปแบบการใช้งาน

Object.DSRHolding

เมื่อขา DSR เป็นลอจิก “1” (DSRHolding = True) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น ComEventDSRTO (Data Set Ready Timeout) และกระตุ้นให้เกิดเหตุการณ์ OnComm

Handshaking

กำหนดคุณสมบัติและคืนค่ารูปแบบแฮนด์เช็กทางฮาร์ดแวร์

รูปแบบการใช้งานคำสั่ง

Object.Handshaking [= blue]

ค่าตัวแปร Value ที่ใช้กำหนดได้ 4 รูปแบบด้วยกันคือ

1. comNone ค่าที่กำหนดคือ 0 เป็นการกำหนดให้ไม่มีการแฮนด์เช็ก (เป็นค่าเริ่มต้น)
2. comXOnXOff ค่าที่กำหนดคือ 1 เป็นการกำหนดให้ใช้แฮนด์เช็กแบบ XON/XOFF
3. comRTS ค่าที่กำหนดคือ 2 เป็นการกำหนดให้ใช้ขา RTS/CTS (Request To Send/Clear To Send)

4. comRTSXOnXOff ค่าที่กำหนดคือ 3 เป็นการกำหนดให้ใช้ทั้งแบบ Request To Send และ XON/XOFF

คุณสมบัติ Handshaking ใช้เพื่อกำหนดรูปแบบการสื่อสารภายใน ระหว่างที่ข้อมูลถูกส่งไปยังบัฟเฟอร์ภาครับ เมื่อข้อมูลตัวอักษรถูกส่งมาถึงพอร์ตอนุกรม อุปกรณ์สื่อสารข้อมูลจะทำการย้ายข้อมูลไปยังบัฟเฟอร์ภาครับ เพื่อที่จะให้โปรแกรมสามารถอ่านค่าไปใช้งานได้ ถ้าไม่มีบัฟเฟอร์ภาครับโปรรามที่ใช้งานจะต้องทำการอ่านค่าข้อมูลโดยตรงจากฮาร์ดแวร์ของพอร์ตอนุกรม ซึ่งผู้ใช้งานจะเกิดปัญหาข้อมูลสูญหายได้ เนื่องจากว่าการเปลี่ยนแปลงของข้อมูลที่ส่งเข้ามามีการเปลี่ยนแปลงอย่างรวดเร็ว

คุณสมบัติ Handshaking ช่วยให้ผู้ใช้งานแน่ใจได้ว่าข้อมูลที่ได้รับมานั้นไม่มีการสูญหายเมื่อบัฟเฟอร์ภาครับที่รับข้อมูลนั้นเกิดข้อมูลล้นหรือโอเวอร์โฟลว (overflow) โดยใช้วิธีการตรวจสอบความพร้อมของบัฟเฟอร์ว่าพร้อมรับข้อมูลหรือไม่ก่อนที่จะส่งข้อมูลมาให้

Break

ใช้ใจการเซตและเคลียร์ค่าสัญญาณ Break ชนิดของข้อมูลเป็นแบบ Boolean

รูปแบบการใช้งาน

Object.Break [= value]

โดย Value เป็นค่าบูลีน ถ้า Value = True หมายถึง การส่งสัญญาณ Break ออกไป ถ้า Value = False หมายถึงการเคลียร์สัญญาณ Break

เมื่อกำหนดให้สัญญาณ Break เป็น True จะเป็นการหยุดการส่งข้อมูลชั่วคราวจนกว่าจะมีการสั่งให้สัญญาณ Break เป็น False

ตัวอย่าง เป็นวิธีการส่งสัญญาณ Break ออกไปเป็นช่วงเวลาสั้น ๆ ที่ 1/10 ของวินาที

```
MSComm1.Break = True                    ' Set the Break condition.
Duration! = Timer +.1                   ' Set duration to 1/10 second.
Do Until Timer > Duration!              ' Wait for the duration to pass.
Dummy = DoEvents ()
Loop
MSComm1.Break = False                   ' Clear the Break condition.
```

เหตุการณ์ OnComm

เหตุการณ์ OnComm จะถูกสร้างขึ้นเมื่อค่าของคุณสมบัติ CommEvent มีการเปลี่ยนแปลงเพื่อแสดงผลการเปลี่ยนแปลงเหล่านั้นแบบทันทีทันใดหรือแสดงข้อผิดพลาดที่เกิดขึ้น ตัวอย่างโปรแกรมย่อย OnComm เพื่อนำเหตุการณ์ CommEvent มาแสดง

```
Private Sub MSComm_OnComm ()
        Select Case MSComm1.CommEvent
        ' Handle each event or error by placing
        ' code below each case statement
        ' Errors
                Case comEventBreak          ' A Break was received.
                Case commEventCDTO          ' CD (RLSD) Timeout.
                Case commEventCTSTO         ' CTS Timeout.
                Case commEventDSRTO         ' DSR Timeout.
                Case commEventFrame         ' Framing Error
                Case commEventOverrun       ' Data Lost.
                Case commEventRxOver        ' Receive buffer overflow.
                Case commEventRxParity      ' Parity Error.
Case commEventTxFul                         ' Transmit buffer full.
        ' Events
                Case comEvCD                ' Change in the CD line.
                Case comEvCTS               ' Change in the CTS line.
                Case comEvDSR               ' Change in the DSR line.
                Case comEvRing              ' Change in the Ring Indicator.
                Case comEvReceive           ' Received Rthreshold # of chars.
                Case comEvSend              ' Sthreshold number in the ' transmit buffer.
Case comEvEof                               'AnEOF charaterwas found in the input stream
        End Select
End Sub
```

การใช้ MSComm เพื่อการติดต่อฮาร์ดแวร์

จากรายละเอียดของ MSComm ที่กล่าวไปในตอนต้นจั้น จะเห็นได้ว่าวิธีการที่จะอ่านค่าหรือเขียนค่าไปยังขาสถานะและขาควบคุมของพอร์ตอนุกรมสามารถทำได้ง่ายดายมาก โดยใช้คณสั่งเหล่านี้

DTREnable	สำหรับสั่งให้ขา DTR มีลอจิก “0” หรือ “1”
RTSEnble	สำหรับสั่งให้ขา RST มีลอจิก “0” หรือ “1”
CTSHolding	สำหรับอ่านค่าสถานะจากขา CTS ว่ามีลอจิก “0” หรือ “1”
CDHolding	สำหรับอ่านค่าสถานะจากขา DCD ว่ามีลอจิก “0” หรือ “1”
DSRHolding	สำหรับอ่านค่าสถานะจากขา DSR ว่ามีลอจิก “0” หรือ “1”
Break	สำหรับการสั่งให้ขา Txd มีลอจิก “0” หรือ “1”

ระดับแรงดันที่ใช้งานสำหรับพอร์ตอนุกรม RS-232

มาตรฐานการสื่อสารข้อมูลของพอร์ตอนุกรม ได้ระบุช่วงระดับแรงดันสำหรับการทำงานของพอร์ตอนุกรมไว้ว่า ที่ลอจิก “0” จะมีระดับสัญญาณ +3 ถึง +15V ส่วนลอจิก “1” จะมีระดับสัญญาณ –3 ถึง –15V ระดับสัญญาณนั้ทำให้ไม่สามารพที่จะนำขาเอาต์พุตใด ๆ ต่อเข้ากับลอจิกเกตเพื่อใช้งานได้โดยตรง จะต้องผ่านวงจรเพื่อเปลี่ยนระดังแรงดันเสียก่อน โดยปกติจะใช้ไอซีจำพวก RS-232 transceiver ที่นิยมมากคือ MAX232 หรือ ICL232 ไอซีในกลุ่มนี้จะทำหน้าที่แปลงระดับแรงดันของ RS-232 ให้อยู่ในระดับทีทีแอล โดยลอจิก “0” ซึ่งเดิมมีระดับสัญญาณ +3 ถึง +15V จะถูกแปลงเป็น 0V ส่วนลอจิก “1” ซึ่งมีระดับสัญญาณ –3 ถึง –15V จะแปลงเป็น +5V ทั้งนี้เพื่อให้สามารถเชื่อมต่อกับอุปกรณ์ดิจิตอลอื่นที่ใช้ระดับแรงดันทีทีแอลได้

การเขียนซอฟต์แวร์เพื่อควบคุมขาเอาต์พุต

การติดต่อกับพอร์ตอนุกรมบนรบบปฏิบัติการวินโดวส์ จะต้องเพิ่มอุปกรณ์ทางซอฟต์แวร์สำหรับการติดต่อกับพอร์ตอนุกรม นั่นคือ MSComm ซึ่งกล่าวรายละเอียดไว้แล้วในตอนต้น

สามารถเขียนโปรแกรมด้วย Visual BASIC เพื่อส่งค่าออกไปยังขาเอาต์พุตต่าง ๆ ของพอร์ตอนุกรมได้ โดยใช้คำสั่งดังนี้

MSComm1.DTREnable = True	สำหรับการกำหนดให้ขา DTR มีลอจิก “1”
MSComm1.DTREnable = False	สำหรับการกำหนดให้ขา DTR มีลอจิก “0”
MSComm1.RTSEnable = True	สำหรับการกำหนดให้ขา RTS มีลอจิก “1”
MSComm1.RTSEnable = False	สำหรับการกำหนดให้ขา RTS มีลอจิก “0”
MSComm1.Break = True	‘ สำหรับการกำหนดให้ขา TxD มีลอจิก “1”
MSComm1.Break = False	‘ สำหรับการกำหนดให้ขา TxD มีลอจิก “0”

หมายเหตุ ก่อนที่จะใช้งานคำสั่งของคอนโทรล MSComm จะต้องทำการเปิดพอร์ตก่อนโดยเขียนโปรแกรมดังนี้

```
Private Sub Form_Load ()
MSComm1.PortOpen = True
End Sub
```

พร้อมกันนั้นจะต้องตรวจสอบพอร์ตที่ใช้งานให้ดีว่าพอร์ตอนุกรมที่ใช้นั้นถูกต้องหรือไม่ มิฉะนั้นโปรแกรมจะแสดงข้อผิดพลาดขึ้นมา

การอ่านค่าลอจิกจากพอร์ตอนุกรม RS-232

พอร์ตอนุกรมมีขาที่ทำหน้าที่อินพุตได้แก่ DSR, CTS, RI และ DCD โดยรีจิสเตอร์ที่ทำหน้าที่ควบคุมขาเหล่านี้คือรีจิสเตอร์แสดงสถานะโมเด็ม (MSR) มีแอดเดรสถัดจากรีจิสเตอร์หลักของพอร์ตอนุกรม 6 ตำแหน่ง สำหรับบิตต่าง ๆ บนรีจิสเตอร์มีรายละเอียดดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
DCD	RI	DSR	CTS	DDCD	DRI	DDSR	DCTS

4 บิตบนของรีจิสเตอร์จะแสดงสถานะการเปลี่ยนแปลงที่ขาอินพุตทั้ง 4 ขาของพอร์ตอนุกรมโดยตรง ส่วน 4 บิตล่าง จะมีสถานะเป็น “1” ก็ต่อเมื่อ 4 บิต บนมีการเปลี่ยนแปลงสถานะเมื่อเทียบกับการอ่านค่าครั้งก่อนหน้า

สำหรับการอ่านค่าจากขา DCD, CTS และ DSR โดโปรแกรม Visual BASIC จะใช้ MSComm ร่วมกับคำสั่ง CDHolding , CTSHolding และ DSRHolding ตามลำดับ ซึ่งค่าที่อ่านได้นั้นจะเป็นบูลีน มีค่าเป็น True หรือ False สามารถตรวจสอบผลจากขาอินพุตเหล่านี้ได้โดยใช้คำสั่ง IF THEN

เนื่องจากสัญญาณของ RS-232 ต้องมีระดับแรงดัน ±3 ถึง ±12V แต่สัญญาณอินพุตที่เกิดขึ้นเป็นระดับทีทีแอล ดังนั้นเมื่อเกิดสัญญาณอินพุตขึ้น จะต้องส่งผ่านวงจรขับเพื่อปรับระดับแรงดันให้เหมาะสมเสียก่อน

มารู้จักกับ Visual Basic

Visual Basic ใช้สำหรับสร้างแอพพลิเคชันเพื่อใช้งานอย่างรวดเร็ว เน้นส่วนติดต่อผู้ใช้ที่เป็นแบบกราฟิก และการสร้างแอพพลิเคชันแบบไคลเอ็นท์/เซิร์ฟเวอร์นอกจากนี้ยังสนับสนุนการสร้างแอพพลิเคชันสำหรับอินเตอร์เน็ต

Visual Basic มีอะไรดี

Visual Basic มีจุดเด่นที่แตกต่างจากเครื่องมือชุดอื่น ๆ คือ ความเรียบง่ายในการพัฒนาแอพพลิเคชัน ทำให้ลดเวลาในการพัฒนาแอพพลิเคชันลงมาก ซึ่งเป็นผลมาจาก Visual Basic สนับสนุนการพัฒนาแอพพลิเคชันแบบ Component ซึ่งก็คือการนำส่วนประกอบ (Component) ด้านซอฟต์แวร์ที่ได้สร้างและทดสอบเป็นอย่างดีแล้วนำมาประกอบกัน แล้วเขียนคำสั่งกำกับการทำงานให้เป็นแอพพลิเคชันที่ทำงานได้จริง

แอพพลิเคชันแบบต่าง ๆ ของ Visual Basic 6.0

เราทราบดีว่า Visual Basic 6.0 นั้นสร้างแอพลิเคชันได้หลายรูปแบบ และสร้างได้รวดเร็วเพราะมีขั้นตอนที่ไม่ยุ่งยากนัก แม้จะไม่เคยมีประสบการณ์ด้านการสร้างซอฟต์แวร์แอพพลิเคชันมาก่อนก็เรียนรู้ได้ไม่ยาก

เมื่อเราเปิด Visual Basic 6.0 ขึ้นมา เราจะพบไดอะล็อกบ็อกซ์ New Project ซึ่งแสดงประเภทของแอพพลิเคชันที่สร้างได้ซึ่งจะมีรายละเอียดของแอพพลิเคชันแต่ละประเภทดังนี้

Standard EXE	เป็นแอพพลิเคชันทั่วไปที่มีการใช้งานใน Windows เมื่อสร้าง แล้วจะได้ไฟล์ที่มีนามสกุลเป็น .EXE (เอ็กซ์คิวต์ไฟล์)
ActiveX EXE	เป็นการสร้างแอพพลิเคชันชนิดที่เรียกว่า out-of-process OLE server
ActiveX DLL	เป็นการสร้างแอพพลิเคชันชนิดที่เยรกว่า in-process OLE server
ActiveX Control	เป็นการสร้าง ActiveX Control เมื่อสร้างแล้วจะได้ไฟล์นามสกุลเป็น .OCX
VB Application Wizard	เป็นการสร้างวิซาร์ดเพื่อใช้งานร่วมกับแอพพลิเคชัน ซึ่งมักจะเป็นแอพพลิเคชันที่มีความซับซ้อน จึงต้องมีวิซาร์ดมาช่วยลดความยุ่งยากซับซ้อนในการใช้งาน
VB Wizard Manager	เป็นเครื่องมือที่ใช้สร้างวิซาร์ด
Data Project	เป็นการสร้างแอพพลิเคชันเพื่อให้ทำงานร่วมกับ Data Object

DHTML Application	เป็นการสร้างแอพพลิเคชันอินเตอร์เน็ตในฝั่งไคลเอ็นท์ โดยใช้ความสามารถของ Dynamic HTML ซึ่งจะรันบนบราวเซอร์ Intemet Explorer
IIS Application	เป็นการสร้างแอพพิลเคชันอินเตอร์เน็ตในฝั่งเซิร์ฟเวอร์ โดยใช้ความสามารถของอินเตอร์เน็ตเซิร์ฟเวอร์อย่าง IIS เช่น Active Server Pages
Addin	เป็นเครื่องมือที่ช่วยในการสร้าง Add-in
ActiveX Document DLL	ป็นการสร้างแอพพลิเคชันชนิดที่เรียกว่า in-process ActiveX Document
ActiveX Coucment EXE	เป็นการสร้างแอพพลิเคชันชนิดที่เรียกว่า out-of-process ActiveX Document

บันทึการสร้างแอพพลิเคชัน (Save)

เมื่อการทดสอบเสร็จสิ้น เราควรบันทึกว่าผลการแก้ไขครั้งล่าสุดเป็นอย่างไร เพื่อที่จะสามารถเรียกกลับมาแก้ไขได้อีก ซึ่งการบันทึกทำได้โดยเรียกเมนู File > Save Project และการบันทึกจะเริ่มจากบันทึกไฟล์ฟอร์มของทุกฟอร์มในโปรเจ็กต์ก่อนจากนั้นจะบันทึกไฟล์โปรเจ็กต์

การสร้างไฟล์ .EXE (Make)

เมื่อเราสร้างแอพพลิเคชันเสร็จแล้ว เราอาจจะต้องการนำแอพพลิเคชันที่สร้างขึ้นเรียกใช้งานได้เอง โดยไม่ต้องเรียกผ่าน Visual Basic หรือต้องการนำไปใช้งานในคอมพิวเตอร์เครื่องอื่น ๆ

เมื่อความต้องการข้างต้นเกิดขึ้นเราต้องทำการสร้างเอ็กซ์คิวต์ไฟล์ (ไฟล์ที่มีนามสกุลเป็น. EXE) โดยเลือกเมนู File > Make ชื่อโปรเจ็กต์ที่เราจะสร้าง เมื่อ Visual Basic ทำงานเรียบร้อย เราก็จะได้ไฟล์ .EXE เพื่อสามารถเรียกใช้งานได้ทันที

การคอมไพล์แบบ Native (Native Code Compiling)

ในอดีตมีคนค่อนขอดในความสามารถของ Visual Basic กันมากในเรื่องของความเร็วแอพพลิเคชัน เพราะ Visual Basic ใช้ตัวแปลภาษาที่เรียกว่า Interpreter คือต้องแปลคำสั่งใน Visual Basic ทุกครั้งที่จะต้องรันและแปลกันบรรทัดต่อบรรทัด ทำให้การทำงานช้ากว่าตัวแปลภาษที่เรียกว่า Compiler ซึ่งจะแปลภาษที่เราเขียนเป็นภาษาระดับต่ำที่ตอมพิวเตอร์พร้อมใช้งานได้ตลอดเวลา

ข้อบกพร่องดังกล่าวได้ถูกกำจัดไปตั้งแต่เวอร์ชัน 5.0 ซึ่งทำให้การแปลภาษาใน Visual Baic เลือกได้ว่า จะใช้แบบ Interpreter หรือ Compiler

การคอมไพล์แบบ Nateve Code ทำได้โดยการเลือกเมนู **Project > ชื่อโปรเจกต์ Properties** แล้วเลือกแท็บ Compile จะเห็นว่ามีการให้เลือกว่าจะเป็น P-Code หรือ Native Code เราจะเลือกคอมไพล์แบบ Native Code Compiling

การคอมไพล์แบบ P-Code หรือ Pseudo Code ก็คือการแปลภาษาโปรแกรมในแบบเดิม คือ Interpreter คือจะแปลให้ได้รหัส P-Code ซึ่งต้องแปลอีกครั้งเพื่อให้ได้ภาษาระดับต่ำที่เครื่องคอมพิวเตอร์เข้าใจ

รู้จักกับฟอร์ม

ฟอร์ม (Form) ก็คือ วินโดว์ที่ใช้แสดงผล ภายในฟอร์มสามารถมี ActeveX Control วางอยู่หรือมีฟอร์มอื่น ๆ อยู่ภายในก็ได้

จะเห็นว่าฟอร์มถือว่าเป็นรูปแบบพื้นฐานที่ Windows ใช้ในการติดต่อกับผู้ใช้งาน โดยมีการวางออบเจ็กต์ที่ทำหน้าที่ติดต่อกับผู้ใช้งานอยู่บนฟอร์ม

ฟอร์มก็ถือว่าเป็นออบเจ็กต์ด้วย เพราะฉะนั้นเราสามารถกำหนดพรอพเพอร์ตี้ หรือเรียกใช้เมธอดของฟอร์มได้

พรอพเพอร์ตี้ที่สำคัญของฟอร์ม

Name	เป็นชื่อของฟอร์มที่เราจะต้องกำหนดเพื่อแยกความแตกต่างของฟอร์มที่มีการใช้งาน (เพราะฉะนั้นในแอพพลิเคชันเดียวกันจะมีชื่อฟอร์มซ้ำกันไม่ได้)
Caption	เป็นข้อความที่แสดงที่ไตเติ้ลบาร์ของแต่ละฟอร์ม
Left, Top, Height Width	เป็นพรอพเพอร์ตี้ที่แสดงตำแหน่งของฟอร์มโดยระบุพิกัดมุมซ้ายบน (Left, Top) และกำหนดความกว้าง และความสูงฟอร์ม (Width, Height)
BorderStyle	เป็นพรอพเพอร์ตี้ที่กำหนดลักษณะขอบของฟอร์ม
BackColor	เป็นสีพื้นของฟอร์ม
ForeColor	เป็นสีตัวอักษรทีอยู่บนฟอร์ม
ControlBox	เป็นการกำหนดว่า ขณะรันจะแสดง Control Box หรือไม่
MinButton, MaxButton	เป็นการกำหนดว่า ฟอร์มจะมีปุ่มที่ย่อ (Minimize) หรือขยายฟอร์ม (Maximize) หรือไม่
Icon	เป็นการระบุไอคอนของฟอร์มเมื่อกดปุ่มย่อฟอร์ม
WindowState	เป็นสถานะของฟอร์มเมื่อเริ่มทำงาน

เมธอดที่สำคัญของฟอร์ม

Show	เป็นเมธอดที่เรียกฟอร์มขึ้นมาแสดงผล
Hide	เป็นเมธอดที่สั่งใช้ซ่อนฟอร์มที่กำลังแสดงผล (ยังไม่จบการทำงาน)
Unload	เป็นเมธอดที่สั่งจบการทำงานของฟอร์ม
CenterMe	เป็นเมธอดที่สั่งให้ฟอร์มแสดงผลอยู่ตรงกลางจอภาพพอดี
Print	เป็นเมธอดที่สั่งให้พิมพ์หน้าตาของฟอร์ม

Line เป็นเมธอดที่สั่งให้วาดเส้นลงนพื้นของฟอร์ม

อีเวนต์ที่สำคัญของฟอร์ม

Initialize จะเกิดขึ้นเมื่อฟอร์มถูกโหลดเข้ามาในหน่วยความจำ

Load จะเกิดขึ้นเมื่อฟอร์มถูกเรียกขึ้นมาใช้งาน (จะเกิดภายหลังอีเวนต์ Initalize)

Resize จะเกิดขึ้นเมื่อฟอร์มถูกปรับขนาดให้เปลี่ยนไป

Activate จะเกิดขึ้นเมื่อฟอร์มนั้นกลายเป็น active form (สำหรับ acteve form คือฟอร์มที่กำลังถูกใช้งานอยู่ ซึ่งอาจมีหลาย ๆ ฟอร์มถูกเปิดอยู่พร้อมกัน)

QueryUnload จะเกิดขึ้นเมื่อฟอร์มถูกปิด

Unload จะเกิดขึ้นเมื่อฟอร์มเลิกใช้งาน โดยจะเกิดภายหลังอีเวนต์ QueryUnload

Terminate จะเกิดขึ้นเมื่ออินสแตนซ์ของฟอร์มถูกกลบออกจากหน่วยความจำ จะเกิดภายหลังอีเวนต์ Unload

บทที่6
การออกแบบและการสร้าง

ในส่วนของการออกแบบและการสร้างได้แบ่งออกเป็น 2 ส่วนคือ

การออกแบบในส่วนฮาร์ดแวร์ เป็นการออกแบบและการสร้างบอร์ดทดลอง ซึ่งภายในบอร์ดประกอบไปด้วย ส่วนของการตรวจจับอุณหภูมิด้วย IC DS1820 ซึ่งภายในวงจรจะประกอบไปด้วยวงจรที่ทำหน้าที่ตรวจวัดอุณหภูมิ และรวมไปถึงตัวกาต้มน้ำที่ได้ทำการแบ่งระดับออกเป็น 3 ระดับอยู่ภายในและในส่วนสุดท้าย คือ ส่วนแสดงผลที่ประกอบไปด้วยหน้าจอ LCD และไฟ LED

การออกแบบในส่วนซอร์ฟแวร์ เป็นการออกแบบในการเขียนโปรแกรม Microcontroller MCS-51 ซึ่งจะเป็นตัวโปรแกรมที่ใช้สำหรับควบคุมการตัดของไฟและระดับของน้ำ โดยการตัดไฟนั้นจะขึ้นอยู่กับอุณหภูมิ

6.1 การออกแบบในส่วนฮาร์ดแวร์

1. วงจรจ่ายแรงดันคงที่

เป็นการใช้ไอซีเรกูเลเตอร์ เบอร์ LM7805 ทำให้ได้แรงดันคงที่ 5 VDC ซึ่งเป็นแรงดังหลักที่จะจ่ายให้กับอุปกรณ์ต่าง ๆ ในวงจร

2. ส่วนของการตรวจจับอุณหภูมิ

ในส่วนวงจรของภาคการตรวจวัดอุณหภูมิ จะใช้ IC DS1820 เป็น IC ชนิด 3 ขา คือ DQ เป็นขาต่อเชื่อมกับระบบบัส, ขาต่อไฟเลี้ยงภายนอก และขากราวด์โดยหัวใจสำคัญของ DS1820 อยู่ที่ตัวตรวจจับอุณหภูมิ และหน่วยความจำความเร็วสูง ที่เรียกว่า สแครตช์แพด (Scratchpad) มีขนาด 9 ไบต์

เมื่อวัดอุณหหภูมิได้ก็จะนำค่าที่วัดได้นี้มาเก็บไว้ในสแครตช์แพดที่บต์ 0 และ 1 ทั้งนี้เนื่องจากไอซี DS1820 สามารถให้ข้อมูลของอุณหภูมิได้ละเอียดถึง 16 บิต เมื่อนำมาแปลงเป็นข้อมูลเลขฐานสิบจึงสามารถแสดงความละเอียดของค่าอุณหภูมิได้ถึง 0.5 องศาเซลเซียสและ 0.9 องศาฟาเรนไฮต์ โดยมีย่านวัดอุณหภูมิ —55 ถึง +125 องศาเซลเซียสหรือ —67 ถึง +257 องศาฟาเรนไฮต์ต้องใช้การแปลงหน่วยเข้ามาช่วย ใช้เวลาในการแปลงค่าอุณหภูมิเป็นข้อมูลดิจิตอบประมารณ 200 มิลลิวินาที สามารถกำหนดขอบเขตของอุณหภูมิที่ทำการวัดได้ และให้แจ้งเตือนเมื่อค่าของอุณหภูมิสูงขึ้นหรือลดค่าต่ำลงถึงค่าที่กำหนด โดยค่าอุณหภูมิที่กำหนดนี้จะเก็บไว้ที่สแครตช์แพดในไบต์ 2 และ 3

3. การเชื่อมต่อกับไมโครคอนโทรลเลอร์ MCS-51

ใช้ขาพอร์ตเพียง 1 ขาเท่านั้นสำหรับการเชื่อมต่อกับ DS1820 โดยต้องมีตัวต้านทานค่า 4.7kΩ ต่อพูลอัปกับไฟเลี้ยง +5V จากนั้นจึงทำการเขียนโปรแกรมเพื่อติดต่อกัน โดยใช้รูปแบบการติดต่อตามมาตรฐานระบบบัสหนึ่งสายของดัลลัส

4. ในการเชื่อมต่อกับหน้าจอแสดงผล (LCD)

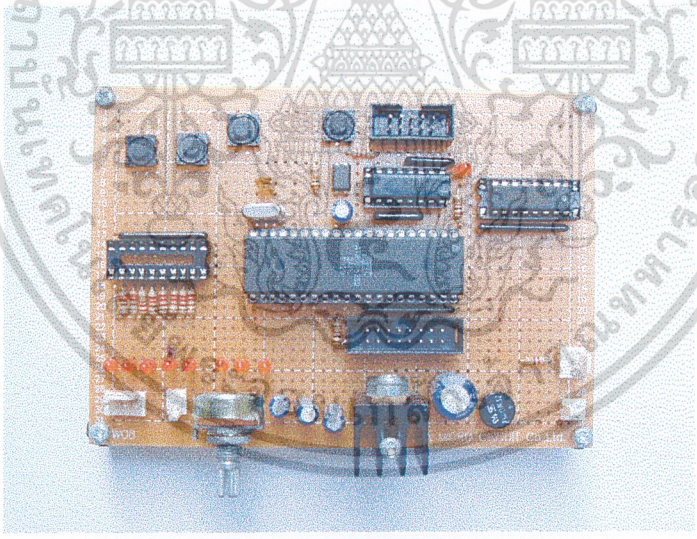
จะเป็นการเชื่อมต่อของ ไมโครคอนโทรลเลอร์ กับ LCD โดยเราจะใช้สายบัสขนาด 16 เส้น เป็นการการเชื่อมต่อระหว่าง MCS-51 กับ LCD

5. สวิตช์กดติดปล่อยดับ

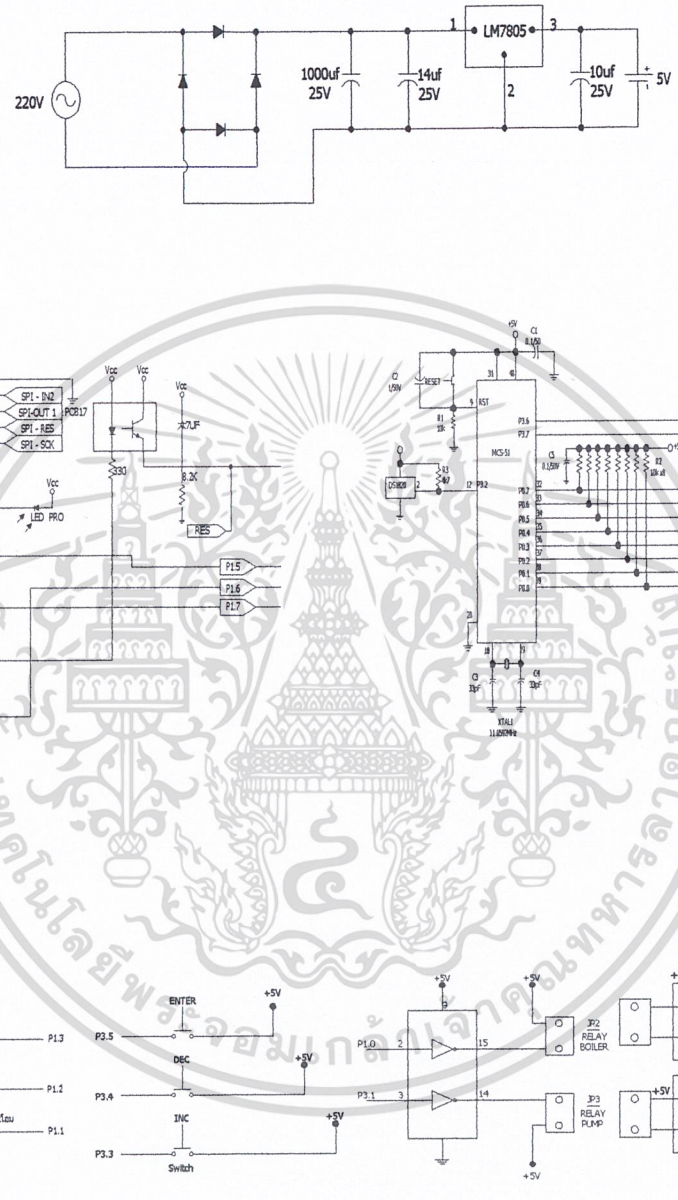
ทำหน้าที่เป็นอินพุท โดยในบอร์ดได้ใช้สวิตช์ไว้ทำการกดเพื่อให้ได้ ค่าระดับน้ำตามที่ต้องการและเมื่อกดเลือกอุณกภูมิที่เราต้องการใช้งานและยังใช้เป็นตัวกดให้ระบบทั้งหมดเริ่มทำงานอีกด้วย

6. ไดโอดเปล่งแสง (LED)

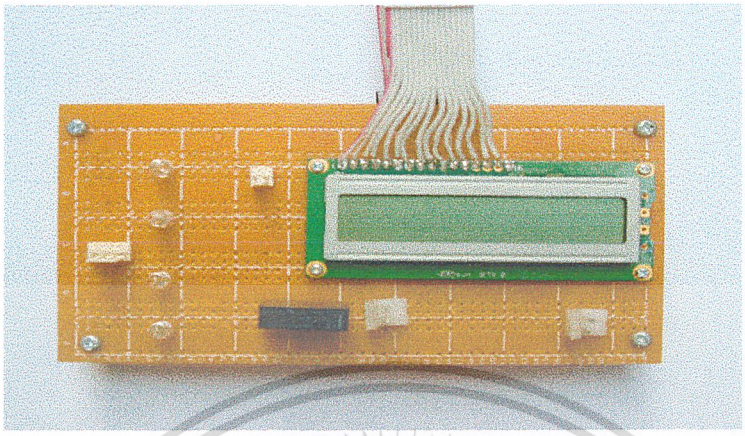
ใช้แสดงผลเมื่อกดสวิตช์ และไดโอดจะดับเมื่อมีกาตัดการทำงานเมื่อน้ำถึงจุดอุณภูมิที่เราต้องการ



รูป 6.1 แสดงการวางอุปกรณ์บนบอร์ดทดลอง



รูป 6.2 รูปแสดงวงจรรวมและการเชื่อมต่อ



รูป 6.3 แสดงในส่วนของจอแสดงผล

7. ส่วนอุปกรณ์กาต้มน้ำร้อน

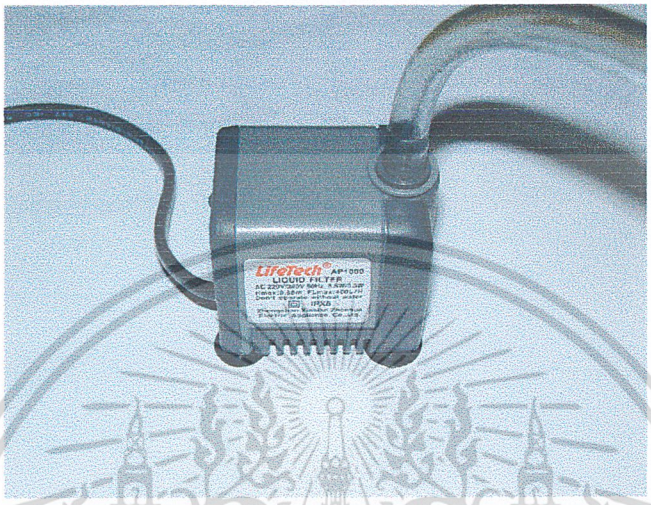
เราจะใช้กาต้มน้ำร้อนทั่วไปที่มีขายตามท้องตลาดโดยก่อนอื่นเราจะนำมาทำการดัดแปลง คือ เพิ่มให้กาต้มน้ำสามารถทำการแบ่งน้ำออกเป็นระดับ ได้ด้วยกัน 3 ระดับ

กระทำโดยใช้สายทองแดงทำในขนาดที่แตกต่างกันโดยจะมีระดับลึกสุดเราจะต่อให้เป็นไฟ DC 5 โวลต์ไว้ และอีก 3 ระดับจะเป็นกราวด์ไว้ก่อนโดยรอให้น้ำเป็นสื่อทางไฟฟ้าคือเมื่อระดับน้ำขึ้นมาถึงในแต่ละระดับจะปรากฏว่าช่องว่างระหว่างลายทองแดงมีค่าความต้านทานค่าหนึ่ง เรานำเอาค่าความต้านทานไปทำการขับทรานซิสเตอร์ให้ทำงานแล้วส่งต่อไปยังชุดควบคุมด้วยไมโครคอนโทรลเลอร์อีกทีหนึ่ง

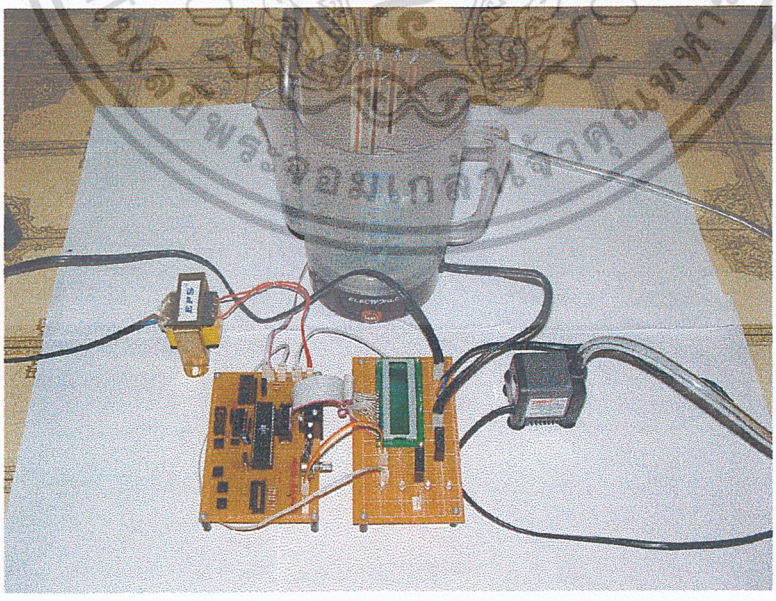
รูป 6.4 กาต้มน้ำที่ได้ถูกแบ่งระดับออกแล้ว

8. ส่วนอุปกรณ์ปั๊มน้ำเข้าอัตโนมัติ

เราใช้ปั๊มน้ำทั่วไปโดยที่ปั๊มน้ำจะทำหน้าที่ปั๊มน้ำจากแหล่งน้ำภายนอกเข้ามายังกาต้มน้ำโดย น้ำที่ปั๊มเข้ามานั้นจะขึ้นอยู่กับความต้องการของราโดยเราเป็นคนกำหนดจากการกดสวิตท์เลือกเอง



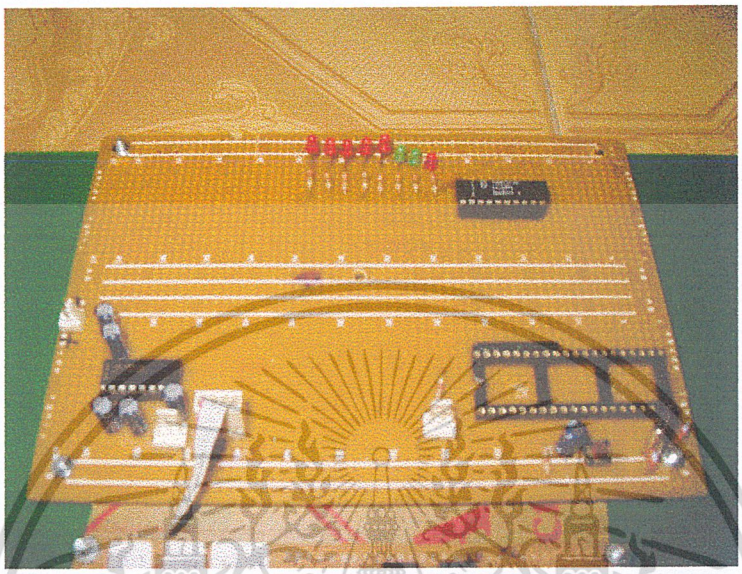
รูป 6.5 ปั๊มน้ำที่ได้ทำการตกแต่งพร้อมใช้งาน



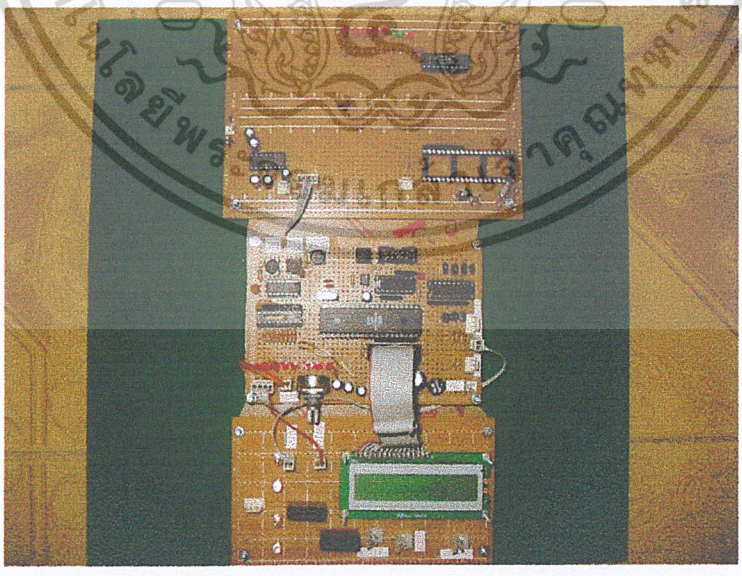
รูปที่ 6.6 ชุดอุปกรณ์ที่ต่อพร้อมใช้งานโดยสมบูรณ์

9.วงจรพอร์ตอนุกรม

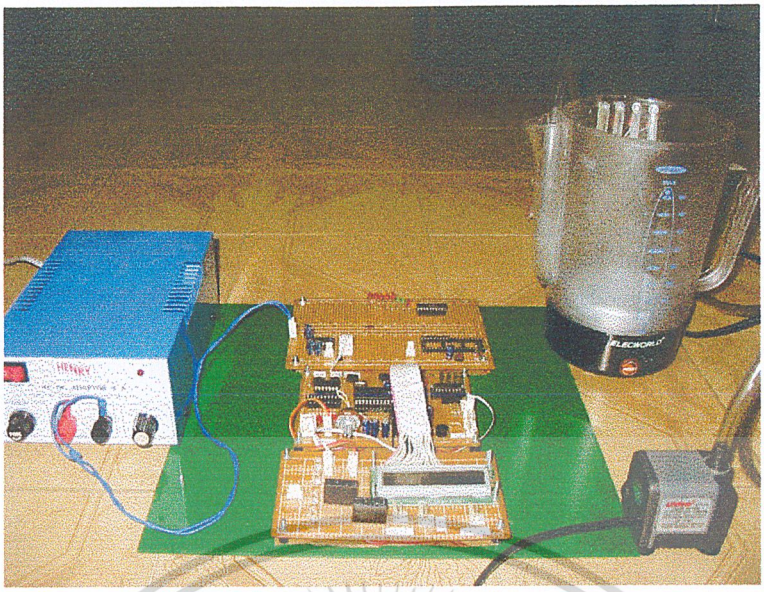
ทำหน้าที่ติดต่อระหว่างระบบในส่วนของไมโครคอนโทรลเลอร์และส่วนของคอมพิวเตอร์



รูปที่ 6.7 วงจรพอร์ตอนุกรม



รูปที่ 6.8 ทำการเชื่อมต่อพอร์ตอนุกรมกับวงจรควบคุม



รูปที่ 6.9 ชุดทดลองพร้อมใช้งานที่สามารถสั่งการด้วยระบบคอมพิวเตอร์

6.2 การออกแบบในส่วนซอร์ฟแวร์

เป็นการออกแบบการควบคุมตรวจจับอุณหภูมิของ DS1820 โดยใช้ Microcontroller MCS-51 (AT 895 8252) เป็นตัวควบคุมการทำงานของอุปกรณ์ทั้งหมดทั้งส่วนของการตั้งค่าอุณหภูมิตามต้องการ ส่วนของระดับน้ำที่ต้องการและการปั๊มน้ำเข้าอัตโนมัติ อีกทั้งในส่วนของโปรแกรม Visual Basic ที่ใช้สำหรับทำการสั่งการระหว่างเครื่องคอมพิวเตอร์กับระบบไมโครคอนโทรลเลอร์โดยที่เราจะใช้เครื่องคอมพิวเตอร์เป็นตัวสั่งการทำงานทั้งหมดและให้แสดงผลออกทางจอภาพซึ่งผ่านการควบคุมทางคีย์บอร์ด

คำอธิบายโปรแกรมอ่านค่าไอซีตรวจจับอุณหภูมิ DS1820

เป็นโปรแกรมที่ใช้สำหรับอ่านค่าจากไอซีวัดอุณหภูมิเบอร์ **DS1820** ซึ่งใช้การติดต่อแบบระบบบัสหนึ่งสาย โดยให้แสดงค่าอุณหภูมิที่อ่านได้บน **LCD** และพิจารณาเฉพาะอุณหภูมิที่อยู่ในช่วง **0-127** องศาเซลเซียส

เริ่มจากการส่งสถานะรีเซตด้วยโปรแกรมย่อย **DS1820_RST** จากนั้นให้รอการตอบรับจาก **DS1820** ด้วยโปรแกรมย่อย **DS1820_PRES** เมื่อมีการตอบรับแล้วให้ทำการส่งค่า **CCH** ไปยัง **DS1820** โดยผ่านโปรแกรมย่อย **DS1820_WR** เพื่อส่งคำสั่ง **Skip ROM** ให้ข้ามการตรวจสอบ **64-BIT LASERED ROM** และส่งค่า **44H** เป็นคำสั่งให้ **DS1820** ทำการแปลงค่าอุณหภูมิมาเก็บไว้ที่หน่วยความจำ **SCRATCHPAD** ภายใน

ในขั้นต่อมา ก็เริ่มส่งสถานะรีเซตใหม่ แล้วรอการตอบรับเช่นเดิม แต่จะทำการตรวจสอบสถานะบิต **BUSY** ว่าทำการแปลงค่าอุณหภูมิเสร็จแล้วหรือไม่ ถ้าไม่ก็ให้วนลูปรอจนกว่าจะเสร็จสิ้นการแปลงค่าจากนั้นให้ทำการส่งสถานะรีเซตใหม่ แล้วรอรับการตอบรับเช่นเดิม เมื่อมีการตอบรับแล้วให้ทำการส่งค่า **CCH** เพื่อทำการ **Skip ROM** และส่งค่า **BEH** เป็นคำสั่งเรียกอ่านค่าจากหน่วยความจำสแครตช์แพด จากนั้นให้อ่านค่าจาก **DS1820** โดยเรียกโปรแกรมย่อย **DS1820_RD** ซึ่งการอ่านค่าในไบต์แรกนั้น จะได้ค่าอุณหภูมิในช่วง 8 บิตล่างออกมา **(TEMPERATUR LSB)** และจะถูกเก็บอยู่ในรีจิสเตอร์ชื่อ **ONEWIRE_DATA** ให้ทำการเก็บค่านี้ไว้ที่รีจิสเตอร์ **TEMP** อีกตัวหนึ่งเพื่อนำมาใช้งานต่อไป จากนั้นให้ส่งสถานะรีเซตใหม่แล้วรอการตอบรับอีกครั้งหนึ่ง เป็นอันจบขั้นตอนการติดต่อกับ **DS1820**

นำข้อมูลในรีจิสเตอร์ **TEMP** ที่ได้มาทำการหมุนไปทางขวาโดยใช้แฟลกร่วมด้วย (ให้ทำการเคลียร์ค่าแฟลกทดก่อนใช้ด้วย) จะได้ค่าอุณหภูมิที่เป็นจำนวนเต็มพอดี โดยนำค่าที่ได้นี้มาแสดงบน **LCD** โดยค่าอุณหภูมิที่เป็นจำนวนเต็มให้เรียกผ่านโปรแกรมย่อย **HEX2LCD** เป็นเลยฐานสิบจำนวน 3 หลัก และดูค่าบิตล่างสุดของรีจิสเตอร์ **TEMP** ถ้ามีค่าเป็น **"1"** จะหมายถึงเพิ่มค่าอุณหภูมิอีก **0.5** องศาเซลเซียส ก็ให้เขียนค่า **".5"** ลงไป แต่ถ้าค่าเป็นศูนย์ ก็ให้เขียนค่า **".0"** ลงไปแทน อาจเขียนตัว "C" เพิ่มเติมเพื่อบอกว่าเป็นหน่วยองศาเซลเซียสก็ได้ เป็นอันสิ้นสุดการทำโปรแกรมใน 1 ครั้ง และทำเช่นนี้วนไปเรื่อย ๆ ได้

คำอธิบายโปรแกรมย่อยที่ใช้

HEX2LCD เป็นโปรแกรมย่อยที่ใช้แปลงค่ารีจิสเตอร์ มาแสดงเป็นเลขฐานสิบ 3 หลักที่โมดูล LCD โดยนำค่าจากรีจิสเตอร์ LCD_DATA มาหารด้วย 100 ก่อน แล้วนำค่าตัวส่วนมาแสดงเป็นเลขหลักแรกแต่ถ้าค่าที่ได้เป็นศูนย์ จะเขียนเป็นช่องว่างแทน ต่อมานำค่าเศษที่เหลือจากการหารครั้งแรก มาหารด้วย 10 แล้วนำค่าตัวส่วนมาแสดงบน LCD เป็นเลขหลักต่อมา และนำค่าเศษจากการหารครั้งที่สองมาแสดงเป็นเลขหลักสุดท้าย (การนำค่าเลขมาแสดงบน LCD ต้องบวกค่านั้นด้วย 30H ให้เป็นรหัสแอสกี้ก่อนจึงนำไปใช้ได้)

DS1820_RD เป็นโปรแกรมย่อยที่ใช้ในการอ่านค่าจาก DS1820 โดยในขั้นแรกจะทำการเคลียร์สัญญาณ ONEWIRE ให้เป็น "0" ก่อน จากนั้นให้หน่วงเวลาไปประมาณ 2µs แล้วเซตสัญญาณ ONEWIRE ให้เป็น "1" แล้วหน่วงเวลาไปประมาณ 4µs (ไม่เกินค่า Read Data Valid 15µs) จึงทำการอ่านค่าสัญญาณ ONEWIRE เก็บไว้ที่แฟลกทด แล้วหน่วงเวลาไป 75µs (ค่าไทม์สล็อตอยู่ระหว่าง 60-120µs) แล้วหมุนข้อมูลไปทางขวาโดยใช้แฟลกร่วมด้วย ทำเช่นนี้ 8 ครั้งจะได้ค่ามาเก็บไว้ในรีจิสเตอร์ ONEWIRE_DATA

DS1820_WR เป็นโปรแกรมย่อยที่ใช้ในการเขียนค่าจากรีจิสเตร์ ONEWIRE_DATA ไปยัง DS1820 โดยขั้นแรกจะทำการหมุนข้อมูลไปทางขวาโดยใช้แฟลกทดร่วมด้วย จากนั้นให้ทำการตรวจสอบแฟลกทดว่าถูกเซตหรือไม่ ถ้าถูกเซต ก้ให้ทำการเขียน Time Slot ของลอจิก “1” คือทำการเคลียร์สัญญาณ ONEWIRE เป็นเวลา 4μs (อยู่ในช่วง Write 1 Low Time) แล้วทำการเซตสัญญาณ ONEWIRE เป็นเวลา 75μs (ไม่ค่าไทม์สล็อตอยู่ระหว่าง 60-120μs) แต่ถ้าแฟลกทดไม่ถูกเซต ก็ให้ทำการเขียน Time Slot ของลอจิก “0” คือทำการเคลียร์สัญญาณ ONEWIRE เป็นเวลา 75μs (อยู่ในช่วง Write 0 Low Time) แล้วทำการเซตสัญญาณ ONEWIRE เป็นเวลา 4μs แทน แล้วกลับไปหมุนข้อมูลใหม่ รวมเป็น 8 ครั้ง ก็จะเสร็จสิ้นการเขียนข้อมูล

DS1820_RST เป็นโปรแกรมย่อยที่ใช้สร้างสถานะรีเซต โดยจะทำการเคลียร์สัญญาณ ONEWIRE เป็นเวลา 1ms (อยู่ในช่วง Reset Time Low 480-4800μs) จากนั้นก็จะทำการเซตสัญญาณ ONEWIRE เป็นเวลาประมาร 16μs (พิจารณาจากค่า Presence Detect High 15-60μs) เพื่อหน่วงเวลารอ Presence Pulse จาก DS1820 ต่อไป

Ds1820_PRES เป็นโปรแกรมย่อยที่ทำหน้าที่รอรับการตอบรับจาก DS1820 (Presence Pulse) และใช้ทำหน้าที่รอการสิ้นสุดกระบวนการแปลงค่าอุณหภูมิด้วย โดยจะทำการรวนตรวจสอบว่ามีการเคลียร์สัญญาณ ONEWIRE เป็น “0” หรือไม่ในช่วงเวลาประมาณ 8.85ms (การตอบรับปรกติจะสามารถตรวจสอบได้ในภายในช่วง Presence Detect High 15-60μs ในกรณีที่รอการแปลงค่าอุณหภูมิ จะต้องใช้เวลาช่วง Temperature Conversion Time 200μs-500μs) โดยถ้าเกินจากช่วงเวลานี้จะคืนค่าแฟลก BUSY เป็น “1” แต่ถ้าตรวจสอบได้ว่ามีการตอบรับ ก็จะรอจนกระทั้งสัญญาณ ONEWIRE เป็น “1” อีกครั้ง แล้วจะทำการหน่วงเวลาไปประมาณ 16μs ก่อนจะทำการเคลียร์แฟลก BUSY เป็น “0” เพื่อแสดงว่ามีการตอบรับกลับมา จึงจะกลับเข้าสู่การทำงานของโปรแกรมหลักได้

ONEWIRE_DELAY เป็นโปรแกรมย่อยสำหรับหน่วงเวลา เป็นเวลาประมาณ 75μs เพื่อใช้ในการเขียนไทม์สล็อตให้อยู่ในช่วงเวลาที่เหมาะสม

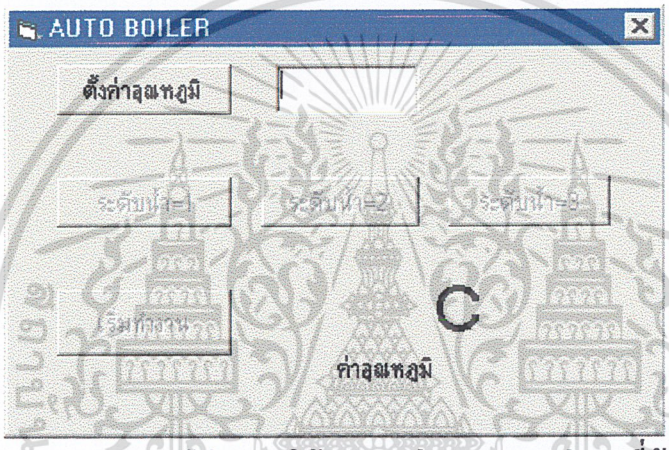
บทที่ 7

การทดลองและผลการทดลอง

การทดลองในส่วนของการปฏิบัติการของเครื่องทำความร้อนด้วยคอมพิวเตอร์มีรายละเอียดดังต่อไปนี้

7.1 กำหนดอุณหภูมิตามต้องการผ่านทางคีย์บอร์ด

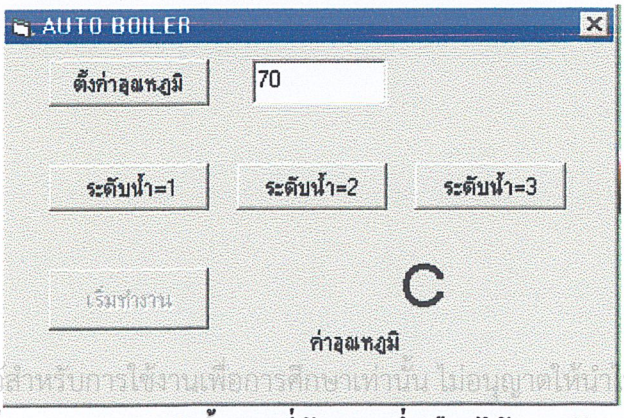
เมื่อเราเรียกโปรแกรมควบคุมการทำงานของระบบกาต้มน้ำอัตโนมัติขึ้นมา หน้าจอคอมพิวเตอร์จะปรากฏ Properties ซึ่งจะให้เราทำการป้อนอุณหภูมิตามที่เราต้องการเข้าไป แล้วกดสั่งการ



รูปที่ 7.1 Properties ของโปรแกรมให้กำหนดค่าของอุณหภูมิตามที่ต้องการ

7.2 กำหนดและตั้งค่าระดับน้ำที่ต้องการ

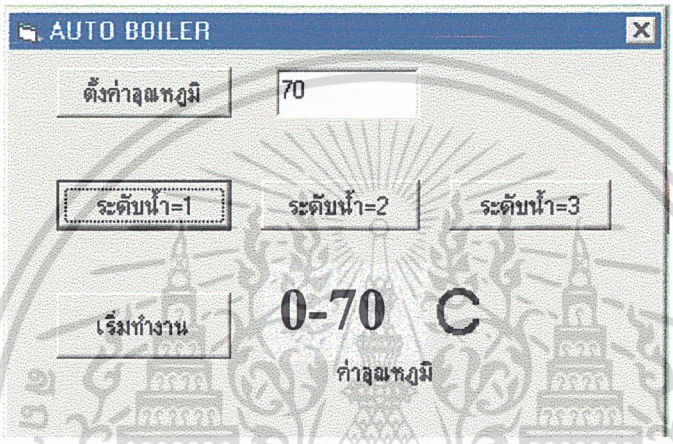
หลังจากทำการกำหนดอุณหภูมิเรียบร้อยแล้ว Program จะให้เราทำการเลือกระดับน้ำที่เราต้องการโดยใช้คีย์บอร์ดหรือเมาส์คลิก เป็นตัวเลือก ซึ่งจะสามารถทำการเลือกได้ 3 ระดับด้วยกัน เมื่อทำการเลือกแล้วก็กดสั่งการทำงาน



รูปที่ 7.2 กำหนดระดับน้ำตามที่ต้องการซึ่งเลือกได้ 3 ระดับ

7.3 สั่งการเพื่อเริ่มการทำงาน

หลังจากที่ได้ทำการกำหนดอุณหภูมิที่ต้องการและระดับน้ำที่ต้องการเรียบร้อยแล้ว Program จะให้เราทำการสั่งการอีกครั้งหนึ่งว่าจะดำเนินการทำงานแน่นอนเราจะเลือกคำสั่งเริ่มทำงานลงไป หลังจากเลือกคำสั่งนี้แล้ว ระบบทั้งหมดจะเริ่มปฏิบัติการ โดยจะทำการปั๊มน้ำอัตโนมัติจากเครื่องปั๊มน้ำให้ได้ตามระดับน้ำที่เราสั่งการ จากนั้นส่วนของระบบทำความร้อนก็จะเริ่มทำงานจนกระทั่งถึงจุดอุณหภูมิที่ได้สั่งการไว้เบื้องต้น ซึ่ง Properties จะปรากฏตัวเลขของอุณหภูมิที่เพิ่มขึ้นให้เห็นอย่างชัดเจน



รูปที่ 7.3 การสั่งการเสร็จสมบูรณ์แบบระบบเริ่มทำงาน

7.4 การทำงานที่เสร็จสมบูรณ์

หลังจากที่ความร้อนถึงจุดอุณภูมิตามที่ได้กำหนดไว้เป็นที่เรียบร้อยถึงเป็นการเสร็จสิ้นการทำงาน โดยที่กาต้มน้ำจะตัดการทำงานลงโดยอัตโนมัติ ซึ่งความร้อนที่ได้นี้ก็จะเป็นความร้อนจริงตามที่เราได้กำหนดไว้เบื้องต้น

บทที่ 8
บทสรุป

เมื่อทำการจ่ายไฟให้กับวงจรแล้วทำการต่ออุปกรณ์ทุกอย่างหใครบระบบและรูปแบบจากการที่เคยใช้การสั่งการผ่านสวิตช์แบบกด เราใช้การสั่งการผ่านเครื่องคอมพิวเตอร์โดยใช้คีย์บอร์ดเป็นตัวควบคุมการทำงานทั้งหมด โดยเมื่อเข้าไปในส่วนของโปรแกรม Visual Basic แล้วที่หน้าจอคอมพิวเตอร์จะปรากฏเป็นหน้าต่างการทำงานให้เราใส่อุณหภูมิตามที่เราต้องการตั้งแต่ 0-99 องศาเซลเซียส แล้วเราจึงสั่งเริ่มการทำงาน จากนั้นเราเลือกระดับน้ำที่ต้องการ โดยเราสามารถเลือกได้ 3 ระดับด้วยกันคือ ระดับ 1, ระดับ 2 และระดับ 3 เมื่อเราเลือกระดับน้ำและเริ่มการทำงาน แล้วระบบก็จะเริ่มการทำงานทุกอย่างโดยอัตโนมัติผลที่ได้คือน้ำและอุณหภูมิจะอยู่ในระดับตามที่เราต้องการ

ปัญหาและแนวทางแก้ไข

1. เนื่องจากในการเชื่อมต่อของระบบ Microcontroller กับ Computer เรายังต้องใช้สายในการเชื่อมต่อระหว่าง Port อยู่ทำให้ไม่สะดวกเท่าที่ควร เพราะสายต่ออาจจะทำให้กีดขวาง หรืออาจเสียหายจากปัจจัยภายนอกที่มากระทำต่อสายได้

2. ส่วนของการแบ่งและวัดระดับน้ำซึ่งยังใช้ระบบของใช้วงจรเป็นตัวแบ่งระดับน้ำอยู่อาจจะทำให้เกิดปัญหาการช็อตกันทางอุปกรณ์หรือลายทองแดงได้ซึ่งอาจทำให้เกิดความเสียหายให้ทั้งระบบก็เป็นได้

แก้ไขได้โดย

1. ออกแบบการเชื่อมต่อระหว่าง Microcontroller กับ Computer ให้เป็นแบบไร้สายเพื่อขจัดปัญหาการต้องใช้สายในการส่งผ่านข้อมูล

2. เปลี่ยนการแบ่งวัดระดับน้ำโดยใช้ตัว Sensor แทนซึ่งจะช่วยลดปัญหาการช็อตกันของวงจรและลายทองแดงลงได้ อีกทั้งยังสะดวกกว่าด้วย

แนวทางการพัฒนา

นำไปประยุกต์และพัฒนาให้เป็นอุปกรณ์ทำความร้อนแบบครบวงจรและไฮเทคโนโลยี ซึ่งง่ายต่อการควบคุมสืบต่อไป

บรรณานุกรม

ปรเมษฐ์ ประนยานันทน์, ปิยพงศ์ ผ่าวณิช, "คู่มือและการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51" กรุงเทพมหานคร : บริษัท เอช.เอ็น.กรุ๊ป จำกัด, 2536

วรพจน์ กรแก้ววัฒนกุล, ชัยวัฒน์ ลิ้มพรจิตรวิไล, "เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51" กรุงเทพมหานคร : บริษัทอินโนเวตีฟเอ็กเพอริเมนต์ จำกัด.

สัจจะ จรัสรุ่งรวีวัลย์, "Visaul Basic and Advanced" กรุงเทพมหานคร : สุทธาการพิมพ์ จำกัด, 2542

กฤษฎา ใจเย็น, อรรถพล บุญยะโภคา, ชัววัฒน์ ลิ้มพรจิตรวิไล, "เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม" กรุงเทพมหานคร : บริษัทอินโนเวดีฟ เอ็กซ์เพอริเมนต์ จำกัด.

DATA SHEET อุปกรณ์ต่าง ๆ ของวงจร, WWW.Atmel.com