ระบบควบคุมสภาพแวดล้อมในเรือนเพาะชำ

SYSTEM CONTROL SURROUNDING FOR PLANTING

โดย

นายสารทิศ สุดเขต

นางสาวอภิรดี พันธ์ทิพย์จตุพร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ระบบควบคุมสภาพแวดล้อมในเรือนเพาะชำ

SYSTEM CONTROL SURROUNDING FOR PLANTING

โดย

นายสารทิศ สุดเขต รหัสประจำตัว 41014457

นางสาวอภิรดี พันธ์ทิพย์จตุพร รหัสประจำตัว 41014518

อาจารย์ที่ปรึกษา

อาจารย์พลศาสตร์ เลิศประเสริฐ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

โครงงาน ปีการศึกษา 2544

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมสภาพแวดล้อมในเรือนเพาะชำ

SYSTEM CONTROL SURROUNDING FOR PLANTING

ผู้จัดทำ

1. นายสารทิศ สุดเขต รหัสประจำตัว 41014457
2. นางสาวอภิรดี พันธ์ทิพย์จตุพร รหัสประจำตัว 41014518

..... อาจารย์ที่ปรึกษา

(อาจารย์พลศาสตร์ เลิศประเสริฐ)

โครงงาน ระบบควบคุมสภาพแวดล้อมในเรือนเพาะชำ

SYSTEM CONTROL SURROUNDING FOR PLANTING

ชื่อนักศึกษา นายสารทิศ สุดเขต รหัสประจำตัว 41014457

นางสาวอภิรดี พันธ์ทิพย์จตุพร รหัสประจำตัว 41014518

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....อาจารย์ที่ปรึกษา

(อาจารย์พลศาสตร์ เลิศประเสริฐ)

กิตติกรรมประกาศ

โครงงานระบบควบคุมสภาพแวดล้อมในเรือนเพาะชำชิ้นนี้สำเร็จลุล่วงไปด้วยดี เนื่องจากได้รับความช่วยเหลืออย่างดียิ่งจาก อาจารย์พลศาสตร์ เลิศประเสริฐ อาจารย์ที่ปรึกษาโครงงานที่กรุณาให้คำปรึกษา เอื้ออำนวยอุปกรณ์ ตลอดจนแก้ไขข้อบกพร่องจนโครงงานชิ้นนี้เสร็จสมบูรณ์ ผู้จัดทำขอกราบขอบพระคุณอย่างสูงไว้ ณ ที่นี้

ขอกราบขอบพระคุณคณาจารย์ในภาควิชาอิเล็กทรอนิกส์ทุกท่าน ที่ได้ถ่ายทอดความรู้ตลอดจนให้ความเอื้อเฟื้อทางด้านอุปกรณ์และให้ข้อเสนอแนะในสิ่งที่ดี

นอกจากนี้ผู้จัดทำขอขอบคุณพี่ๆและเพื่อนๆ ในภาควิชาอิเล็กทรอนิกส์ที่ได้กรุณาเอื้อเฟื้อคำแนะนำและอุปกรณ์ตลอดจนแนวคิดที่ดี จนโครงงานชิ้นนี้สำเร็จและมีอุปสรรคอยู่บ้างผู้จัดทำถือเป็นประสบการณ์ที่ดียิ่งในการที่จะนำไปปรับปรุงแก้ไขและใช้ในโครงงานชิ้นต่อไป

.....

นายสารทิศ สุดเขต

.....

น.ส.อภิรดี พันธ์ทิพย์จตุพร

ระบบควบคุมสภาพแวดล้อมในเรือนเพาะชำ

นายสารทิศ สุดเขต รหัส 41014457
นางสาวอภิรดี พันธ์ทิพย์จตุพร รหัส 41014518
อ. พลศาสตร์ เลิศประเสริฐ อาจารย์ที่ปรึกษา
ปีการศึกษาที่ 2544

บทคัดย่อ

การเกษตรกรรมเป็นสิ่งที่อยู่คู่คนไทยมาเป็นเวลานาน นับตั้งแต่อดีตจวบจนถึงปัจจุบัน รวมถึงเป็นสิ่งที่สร้างรายได้ให้กับประเทศจำนวนมากในการเป็นสินค้าส่งออกที่สำคัญประเภทหนึ่ง โครงงานเรื่องระบบควบคุมสภาพแวดล้อมภายในเรือนเพาะชำนี้ มุ่งเน้นที่จะสร้างแนวคิดในการประกอบเกษตรกรรมแผนใหม่ โดยใช้เทคโนโลยีมาช่วยในการประกอบการเกษตร เพื่อเพิ่มความสะดวกสบาย และที่สำคัญคือการช่วยเพิ่มผลผลิต โครงงานชิ้นนี้จะทดลองสร้างสภาพแวดล้อมที่เหมาะสมกับการทำการเกษตรโดยให้ความสำคัญกับสองเรื่องคือ การควบคุมอุณหภูมิให้เหมาะสมกับการทำการเกษตรโดยอัตโนมัติ และการควบคุมการให้อาหารพืชโดยอัตโนมัติเช่นกัน โดยต้องการควบคุมให้งบประมาณการสร้างเรือนเพาะชำนั้นอยู่ในวงเงินที่จำกัด เพื่อคำนึงถึงความคุ้มค่าในการลงทุนด้วย การควบคุมนั้นจะใช้ไมโครโปรเซสเซอร์ เบอร์ 89S8252 ในการควบคุมทั้งในเรื่องอุณหภูมิและการให้อาหาร และมีการแสดงผลการทำงานรวมไปถึงการควบคุมการทำงานโดยคอมพิวเตอร์

System control surrounding for planting

Saratit Sudkhet 41014457

Aphiradi Phanthipchatuporn 41014518

Polasat Lertprasert Advisor

2001

Abstract

The majority of the Thai population is in agriculture is play a very essential role in export and earn income for the country.Therefore new technologies are necessary for the agricultural sector to make quality and mass productions.

This project has an idea for new agriculture which is comfortable and highly qualified for mass product. The project is focused at temperature control and automatic feeding by microcontroller number 89s8252.Control and display system are performed by computer.

สารบัญ

สารบัญรูปและตาราง

บทที่ 1
บทนำ

โครงงาน การควบคุมสภาพแวดล้อมในเรือนเพาะชำ นี้ได้เกิดขึ้นโดยมีวัตถุประสงค์ในการทดลองแนวคิดการทำเกษตรแผนใหม่ โดยต้องการให้ผลผลิตของการทำการเกษตรมีค่าเพิ่มขึ้น ในขณะที่ความสะดวกสบายในการทำการเกษตรก็มีค่าเพิ่มขึ้นด้วย นอกจากนี้การทำโครงงานชิ้นนี้ยังสามารถทำให้ผู้จัดทำโครงงานได้รับความรู้ทางด้านอิเลคทรอนิกส์มากมายและหลากหลายด้าน เช่น ทางด้านการใช้งานไมโครโปรเซสเซอร์ทั้งในส่วนของการออกแบบและการเขียนโปรแกรมด้วยภาษา แอสเซมบลี รวมทั้งความรู้ในการออกแบบวงจรพาวเวอร์อิเลคทรอนิกส์ การใช้งานวงจร Analog to Digital Conversion ฯลฯ

การทำโครงงานชิ้นนี้ได้ออกแบบแบบจำลองเพื่อควบคุมสภาพแวดล้อมที่เหมาะกับการเพาะปลูกของพืชแต่ละชนิด โดยควบคุมอุณหภูมิ และควบคุมการรดน้ำใช้เป็นตามเวลา และปริมาณที่กำหนด มีการป้อนอินพุทและแสดงผลทางคอมพิวเตอร์ การควบคุมอุณหภูมิของเรือนเพาะชำนั้น เพื่อต้นทุนที่มีค่าต่ำเหมาะกับการนำไปใช้ในการทำการเกษตรจริงๆจึงใช้พัดลมและสปริงเกิล ในการลดอุณหภูมิ โดยเมื่ออุณหภูมิมีค่ามากกว่าอุณหภูมิที่เราต้องการควบคุมไมโครโปรเซสเซอร์ จะสั่งให้สปริงเกิลพ่นละอองน้ำที่เป็นฝอยออกมาเพื่อให้อุณหภูมิภายในเรือนเพาะชำลดลง และในขณะเดียวกันไมโครโปรเซสเซอร์ก็จะสั่งให้พัดลมเปิดเพื่อดูดความร้อนและความชื้นภายในเรือนเพาะชำออกมา โดยมีวัตถุประสงค์เพื่อทำให้การลดอุณหภูมิของเรือนเพาะชามีการลดลงอย่างรวดเร็วมากยิ่งขึ้น และยังเป็นการช่วยลดความชื้นภายในเรือนเพาะชำให้ลดลงอีกด้วย

<u>การออกแบบวงจร</u> มีขั้นตอนดังนี้

1. การออกแบบวงจร Sensor อุณหภูมิ โดยใช้ Sensor อุณหภูมิเบอร์ LM 335 โดยจะมีการเปลี่ยนแปลงค่าแรงดัน 10mV / K โดยเพื่อความสะดวกในการใช้งาน เราจึงออกแบบให้วงจรมีค่าแรงดันเปลี่ยนแปลงตามอุณหภูมิในหน่วยองศาเซลเซียสอย่างเป็นเชิงเส้น โดยที่อุณหภูมิเท่ากับ 30 องศาเซลเซียส แรงดันเอาท์พุทจะมีค่าเท่ากับ 3 โวลท์ และเมื่ออุณหภูมิเปลี่ยนแปลงเป็น 40 และ 50 องศาเซลเซียส แรงดันเอาท์พุทจะเปลี่ยนแปลงเป็น 4 โวลท์และ 5 โวลท์ ตามลำดับ

2. การออกแบบวงจร Digital to Analog Conversion นั้นมีความจำเป็นเมื่อต้องการให้ค่าแรงดันที่เกิดจากอุณหภูมิที่ทำการวัดได้ นำมาเปรียบเทียบกับแรงดันที่เกิดจากอุณหภูมิอ้างอิงในไมโครคอนโทรลเลอร์ จึงจำเป็นต้องแปลงแรงดันนั้นให้เป็นสัญญาณ Digital และป้อนเข้าสู่ไมโครคอนโทรลเลอร์ต่อไป

การออกแบบวงจร Digital to Analog Conversion นั้นจะออกแบบให้วงจร Digital to Analog Conversion ทำงานแบบ Free running คือเมื่อมีการเปลี่ยนแปลงค่าของแรงดันอินพุทวงจรจะทำการเปลี่ยนสัญญาณที่เปลี่ยนแปลงนั้นให้เป็นสัญญาณ Digital ทันทีโดยไม่ต้องรอสัญญาณจาก ไมโครคอนโทรลเลอร์ โดยการทำงานแบบ Free running นั้น ต้องมีการใส่ Switch ที่ควบคุมการเริ่มต้นการเปลี่ยนแปลงค่าเป็น Digital โดยเราจะแทน Switch ด้วย Relay ที่ควบคุมด้วยกระแสไบอัสของทรานซิสเตอร์ เมื่อต้องการเริ่มต้นการแปลงค่าเราจะมีการสั่งให้ไมโครคอนโทรลเลอร์จ่าย Pulse ขนาด 5 โวลต์มาให้ทรานซิสเตอร์ตัวนี้ส่งผลให้ทรานซิสเตอร์ On และ Relay เกิดการตัดเสมือนเป็น Switch ที่เริ่มต้น การเปลี่ยนแปลงค่าเป็น Digital

3. การออกแบบ Switch ควบคุมการทำงานของปั๊มน้ำและพัดลม นั้นจะใช้ Relay แทนการทำงานของ Switch โดยจะมีทรานซิสเตอร์ควบคุมการทำงานของ Relay โดยเมื่อต้องการให้ Relay On เราต้องสั่งให้ไมโครคอนโทรลเลอร์จ่าย อินพุทขนาด 5 โวลท์ จะทำให้ทรานซิสเตอร์ On และทำให้ Relay ทำงาน

4. การออกแบบส่วนแสดงผลด้วยจอ LCDที่ใช้ในการแสดงผลขนาดเพียง 16 ตัวอักษร1 บรรทัด ติดอยู่ที่กล่องควบคุม

5. การป้อนอินพุทแสดงผลทางคอมพิวเตอร์ผ่านทางส่วนติดต่อผู้ใช้งาน ที่เขียนด้วยโปรแกรมภาษา Basic ผ่านทางพอร์ตอนุกรม ของเครื่องคอมพิวเตอร์ PC โดยใช้คอนโทรล MSComm ที่มีอยู่ในโปรแกรม Visual Basic 6

6. การออกแบบวงจรไฟเลี้ยง เราต้องทำการออกแบบวงจรไฟเลี้ยง ขนาด 5 โวลท์ +15 โวลท์ โดยใช้ Power IC เบอร์ LM7805 และเบอร์ LM7815 ตามลำดับ

บทที่ 2
ทฤษฎีพื้นฐาน

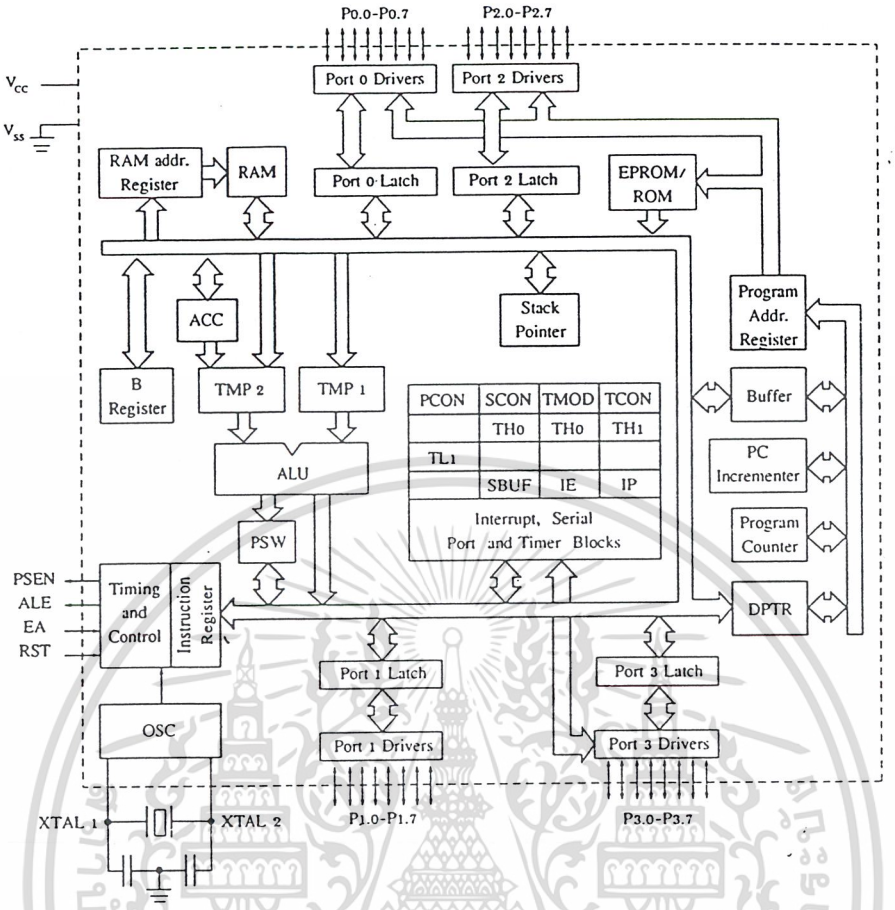
2.1 ไมโครคอนโทรลเลอร์ MCS-51

2.1.1โครงสร้างของไมโครคอนโทรลเลอร์ MCS-51

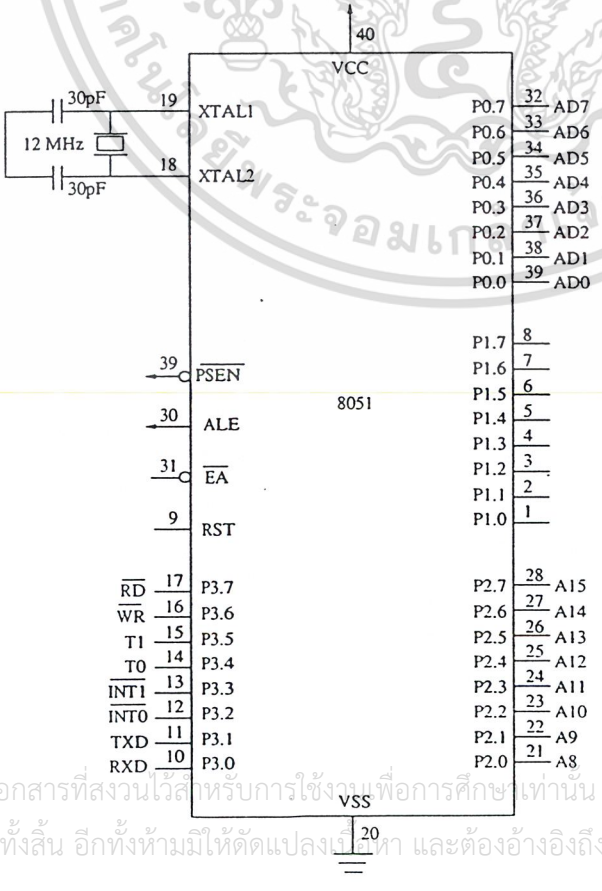
ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีด้วยกันหลายเบอร์ขึ้นกับโครงสร้างภายในโดยคุณสมบัติที่สำคัญของMCS-51มีดังนี้

1. มีหน่วยความจำ ROM 4K byte
2. มีหน่วยความจำ RAM 128 byte
3. มีพอร์ต I/O ขนาด 8 bits 4 พอร์ต
4. มี Timer 16 bit 2ตัว
5. สามารถ interrupt ได้ 5 แหล่ง
6. มีวงจรออสซิสเลเตอร์และวงจรนาฬิกาบนชิป
7. มีพอร์ตอนุกรมที่สามารถรับส่งข้อมูลแบบ Full duplex ความเร็วสูง
8. อ้างหน่วยความจำโปรแกรมภายนอกได้ 64K
9. อ้างหน่วยความจำข้อมูลภายนอกได้ 64K
10. สามารถประมวลผลทีละบิตได้
12. สามารถอ้างหน่วยความจำแบบบิต ได้ 210 ตำแหน่ง
13. หนึ่งวัฏจักรคำสั่ง(MachineCycle)กินเวลา 1 ไมโครวินาทีขณะทำงานด้วย clock12 MHz

โดยในโครงงานนี้ใช้ไมโครคอลโทรลเลอร์เบอร์ 89S8252 เป็นไมโครคอนโทรลเลอร์เบอร์ที่มี EEPROM อยู่ภายใน 2 Kbyte สำหรับเขียนและอ่านข้อมูล และมี FLASH MEMORY ภายในอีก 8 Kbyte สำหรับเก็บโปรแกรมที่ใช้ควบคุมการทำงาน



รูปที่2.1โครงสร้างภายในของ MCS-51



รูปที่2.2 แสดงขาของ 8051

2.1.2 ตำแหน่งขาของ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51ทุกเบอร์มีตำแหน่งขาพื้นฐานเหมือนกันคือมีโครงสร้างไอซีเป็นแบบ DIP มีขาทั้งหมด 40 ขา โดยจะใช้เป็นขาอินพุท,เอาท์พุท,ขาสัญญาณควบคุม,ขาตำแหน่งหน่วยความจำและขาข้อมูลดังรูปที่2.2โดยแบ่งเป็นขาต่างๆดังนี้

1.พอร์ต 0 (port 0) ได้แก่ขาที่ 32-39 ของMCS-51 สามารถใช้เป็นอินพุท,เอาท์พุท,ใช้ติดต่อกับหน่วยความจำทั้งข้อมูลและโปรแกรมภายนอกชิปและใช้เป็นขาAddress Busและ Data Bus อีกด้วย

2.พอร์ต 1 (port 1)ได้แก่ที่1-8 เป็นพอร์ต 8 บิต สามารถอ้างได้ทีละบิต คือP1.1, P1.2, P1.3, P1.4, P1.5, P1.6, P1.7 และ P1.8 ใช้งานเป็นพอร์ตอินพุทและเอาท์พุททั่วไปได้

3.พอร์ต 2 (port 2) ได้แก่ขาที่21-28 จะใช้งาน2หน้าที่ คือเป็นพอร์ท 8 บิต และใช้เป็นขาแอดแดรส 8 bitsในการอ้างหน่วยความจำภายนอก

4.พอร์ต 3 (port 3) ได้แก่ขาที่ 10-17 จะใช้งาน 2 หน้าที่คือ เป็นพอร์ตอินพุท,เอาท์พุท และใช้เป็นขาควบคุมดังตารางที่2.1

Bit	ชื่อ	หน้าที่พิเศษ
P3.0	RXD	ใช้รับข้อมูลทางพอร์ตอนุกรม
P3.1	TXD	ช้าส่งข้อมูลทางพอร์ตอนุกรม
P3.2	$INT_{0/}$	อินเทอร์รัพท์ภายนอกหมายเลข0
P3.3	$INT_{1/}$	อินเทอร์รัพท์ภายนอกหมายเลข1
P3.4	T_0	ตัวจับเวลา/ตัวนับตัวที่0
P3.5	T_1	ตัวจับเวลา/ตัวนับตัวที่1
P3.6	WR/	สัญญาณเขียนข้อมูลหน่วยความจำภายนอก
P3.7	RD/	สัญญาณอ่านข้อมูลหน่วยความจำภายนอก

ตารางที่2.1 แสดงบิตและหน้าที่ของพอร์ต 3

5. PSEN/ (Program Store Enable) เป็นขาที่ส่งสัญญาณออกโดยขาที่29 ขานี้จะแอคทีฟเมื่อ MCS-51 ต้องการอ่านcodeโปรแกรมภายนอก โดยปกติถ้าหน่วยความจำภายนอกเป็น EPROM ขา PSEN/ จะต่อกับขา output Enable(OE/) ของ EPROM

6. ALE (Address Latch Enable) เนื่องจากพอร์ต 0 สามารใช้เป็นขาอ้างตำแหน่ง และขาข้อมูล MCS-51 จะมีขาALEคือขาที่ 30 ขานี้จะใช้ Multiplex สัญญาณ Address Bus ของพอร์ต 0 โดยในการใช้งานของ MCS-51 นั้นจะต้องมีอุปกรณ์มาต่อกับพอร์ท0 ทำหน้าที่Latch สัญญาณ

Address Bus เมื่อ MCS-51 ต้องการติดต่อกับหน่วยความจำภายนอก MCS-51 จะส่งสัญญาณALE มา Latch อุปกรณ์ภายนอก ให้เก็บค่า Address Bus ของพอร์ต 0 ไว้เพื่อใช้พอร์ต 0 เป็น Data bus

7. EA/ (External Access) คือขาที่31 ถ้าขานี้เป็นlogic "1" จะใช้กับเบอร์8051/8052 เพื่อบอกว่าให้อ่านโปรแกรมจากหน่วยความจำโปรแกรมภายใน แต่ถ้าเป็นlogic "0" MCS-51 จะอ่านโปรแกรมจากหน่วยความจำโปรแกรมภายนอกและขา PSEN จะแอคทีฟ

8. RST (reset) คือขาที่9 ใช้ในการรีเซต MCS-51โดยต้องให้ขา9นี้เป็นลอจิก"1" อย่างน้อย 2 Machine Cycle จึงจะรีเซตระบบได้

2.1.3 ความถี่สัญญาณนาฬิกาใน MCS-51

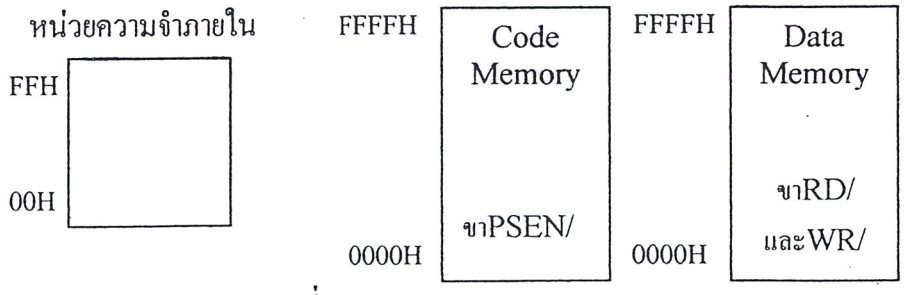
การทำงานของไมโครคอนโทรลเลอร์จำเป็นต้องพึ่งสัญญาณนาฬิกาเป็นตัวกำหนดจังหวะการทำงานภายในทั้งหมดและไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีวงจรออสซิเลเตอร์บนชิป(on chip oscillator)สามารถใช้เป็นตัวสร้างสัญญาณนาฬิกาให้กับ CPU ได้โดยเพียงแค่ต่อ Cystal เข้าที่ขา18-19 โดยปกติมักจะใช้Cystal ความถี่ 12 MHz กับตัวเก็บประจุหรืออาจใช้สัญญาณนาฬิกาจาก TLL Clock Source ต่อกับ XTAL1 และ XTAL2

2.1.4 Power Connection ใน MCS-51

จะใช้แหล่งจ่ายไฟ 5V ต่อเข้ากับขา Vcc(ขา40) ส่วนขา Vss(ขา20) ต่อลงกราวด์

2.1.5 โครงสร้างหน่วยความจำใน MCS-51

หน่วยความจำของMCS-51 มี 2ชนิด คือหน่วยความจำที่ใช้เก็บโปรแกรม(ROM) กับหน่วยความจำที่เก็บข้อมูลในการประมวลผล(RAM) MCS-51จะมีหน่วยความจำข้อมูลและหน่วยจำโปรแกรมแยกจากกัน เบอร์8051,8052 จะมีหน่วยความจำโปรแกรมภายในชิป และMCS-51 ทุกเบอร์สามารถอ้างหน่วยความจำโปรแกรมภายนอกได้มากที่สุด 64K และอ้างหน่วยความจำข้อมูลภายนอกได้มากที่สุด 64K โดยหน่วยความจำ RAM ภายในจะประกอบไปด้วยพื้นที่ใช้งานทั่วไป,รีจีสเตอร์แบงค์,พื้นที่ใช้งานระดับบิต และรีจีสเตอร์ฟังค์ชันพิเศษ เราอาจเขียนไดอะแกรมของหน่วยความจำได้ดังนี้

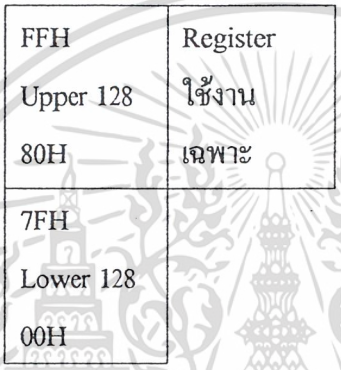


รูปที่2.3 การจัดหน่วยความจำของ MCS-51

5.1 หน่วยความจำข้อมูล (RAM)

1)ในชิปหน่วยความจำข้อมูลภายในชิป ถูกแบ่งออกเป็น 3 ย่อยดังนี้

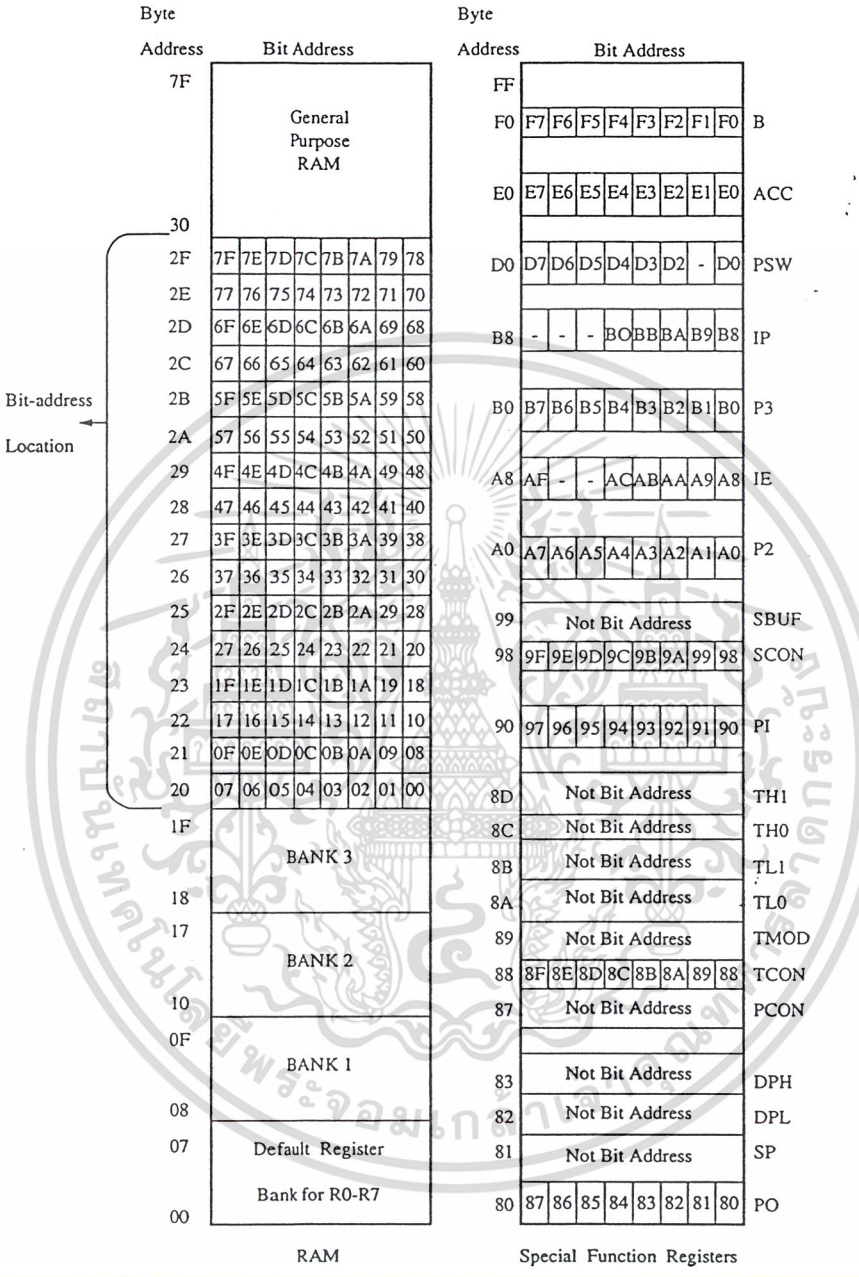
- หน่วยความจำข้อมูลทั่วไปบริเวณ 128 ไบต์ล่าง (lower 128)
- หน่วยความจำข้อมูลทั่วไปบริเวณ 128 ไบต์บน (upper 128)
- หน่วยความจำข้อมูลที่ใช้เป็นรีจีสเตอร์ใช้งานเฉพาะ (STR)



รูปที่2.4 แผนภาพแสดงหน่วยความจำข้อมูลภายในชิปทั้ง 3ส่วน

ค่าตำแหน่งของหน่วยความจำข้อมูลภายใน MCS-51 มีขนาด 8 บิต ทำให้สมารถติดต่อกับหน่วยความจำข้อมูลภายในได้เพียง 256 byte (00H-FFH) แต่จากโครงสร้างจะเห็นว่ามีหน่วยความจำข้อมูลภายในรวมทั้งสิ้น 384 byte เนื่องจากว่าหน่วยความจำข้อมูลภายในบริเวณ128 ไบต์บน(7FH-0FFH)จะเข้าถึงโดยทางอ้อมเท่านั้น แต่สำหรับหน่วยความจำข้อมูลภายในที่ใช้เป็นรีจีสเตอร์ใช้งานเฉพาะ(7FH-0FFH เช่นเดียวกัน)จะเข้าถึงได้โดยทางตรงเท่านั้น ดังนั้นหน่วยความจำข้อมูลภายในทั้งสองบริเวณมีตำแหน่งซ้ำกันได้

หน่วยความจำข้อมูลภายในบริเวณ 128 ไบต์ล่าง ตำแหน่ง 00F-1FH รวม 32 ไบต์จะถูกกำหนดให้เป็นกลุ่มรีจีสเตอร์ใช้งานทั่วไป 4 กลุ่ม กลุ่มละ 8 ตัวคือ R0-R7 โดยจะถูกเลือกใช้งานเพียงกลุ่มเดียวในขณะใดขณะหนึ่ง และกลุ่มที่อยู่ถัดจากกลุ่มรีจีสเตอร์ใช้งานทั่วไปทั้ง 4 กลุ่ม ตั้งแต่บริเวณ 20H-2FH รวม 16 ไบต์มีลักษณะโครงสร้างพิเศษกว่าบริเวณอื่นคือแต่ละบิตของหน่วยความจำบริเวณนี้มีหมายเลขตำแหน่งกำหนดไว้แน่นอนเพื่อนำมาใช้เป็นข้อมูลขนาดหนึ่งบิตในการทำงานของกลุ่มคำสั่งประมวลผลแบบบูลีนได้หรือเพื่อมาใช้เป็นบิตบอกสถานะของระบบตามต้องการของผู้ใช้งาน



รูปที่2.5 ตำแหน่งของหน่วยความจำทั้งแบบไบต์และบิต

หน่วยความจำข้อมูลที่ใช้เป็นรีจีสเตอร์ใช้งานพิเศษเริ่มตั้งแต่ตำแหน่ง 80H-0FFH โดยแต่ละตัวมีไว้เพื่อควบคุมการทำงานและรายงานผลการทำงานของวงจรภายในชิปที่ทำหน้าที่เฉพาะต่างๆ

2) หน่วยความจำข้อมูลภายนอกชิป ซึ่งมีขนาดได้สูงสุด 64 กิโลไบต์แต่อาจใช้ไม่ครบทั้งหมดหรือใช้เพียงบางส่วนขึ้นอยู่กับระบบและการออกแบบ หน่วยความจำข้อมูลภายนอกชิปจำ

เป็นต้องมีสัญญาณควมคุมการอ่านและการเขียนข้อมูลคือสัญญาณ RD และ WR ตามลำดับ โดยจะถูกส่งผ่านทางขา P3.7 และ P3.6 ตามลำดับ

5.2 หน่วยความจำโปรแกรม

ในกรณีของ MCS-51 ผู้ใช้สามารถเลือกได้ว่าจะให้ CPU ทำงานจากโปรแกรมที่เก็บไว้ที่หน่วยความาจำโปรแกรมภายในชิปหรือนอกชิปก็ได้ โดยการควบคุมจากขา EA/ (External Access) ให้ต่อลงVcc หรือ Gnd

ต่อขา EA/ กับ Vcc และเลือกใช้ชิปที่มีหน่วยความจำโปรแกรมเป็น EPROM หรือ ROM อยู่ภายในและ CPU จะเฟตช์คำสั่งในโปรแกรมตั้งแต่ตำแหน่ง 0000H ถึงตำแหน่งตามขนาดของหน่วยความาจำโปรแกรมภายในจากหน่วยความาจำโปรแกรมภายใน ส่วนคำสั่งโปรแกรมตำแหน่งถัดมาจะถูกเฟตช์จากหน่วยความาจำโปรแกรมภายนอกชิป ตัวอย่างเช่น เลือกใช้ชิปที่หน่วยความจำโปรแกรมภายในชิป 4Kbyte CPU จะเริ่มเฟตช์คำสั่งในโปรแกรมตั้งแต่จำแหน่ง 0000H — 0FFFH จากหน่วยความจำโปรแกรมภายในชิป และจะเฟตช์คำสั่งในโปรแกรมตำแหน่ง 1000H — 0FFFFH จากหน่วยความจำโปรแกรมภายนอกชิป

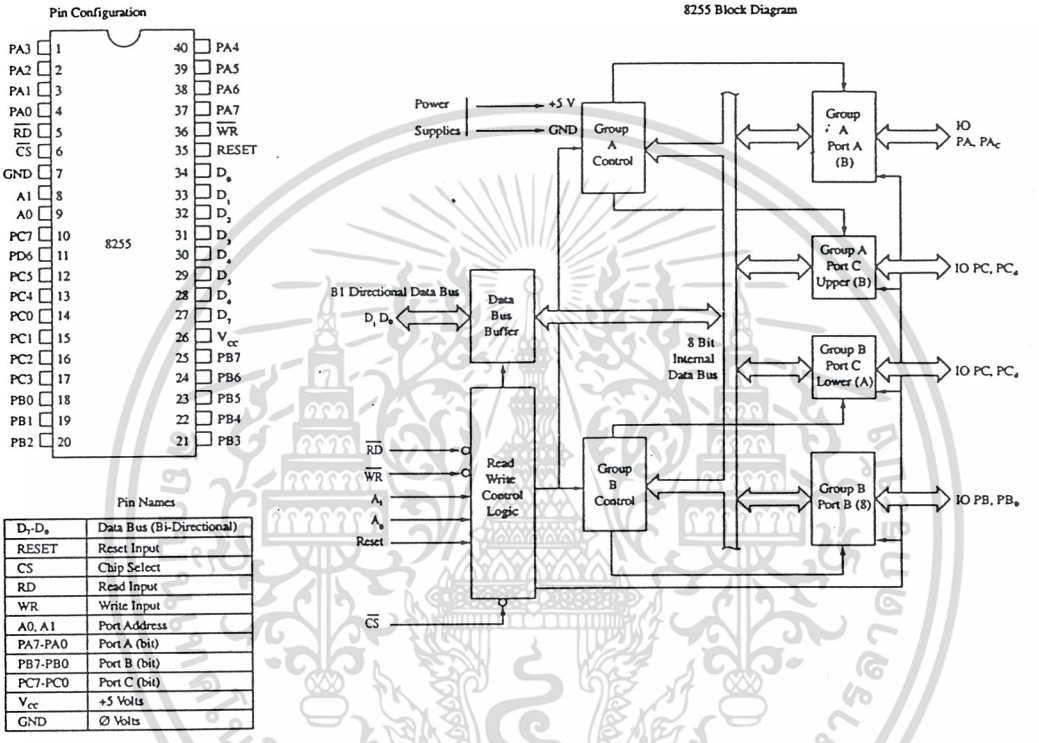
ต่อขา EA/ กับ Gnd CPU จะเฟตช์คำสั่งกับโปรแกรมที่เก็บไว้ในหน่วยความจำโปรแกรมภายนอกชิป (read strobe) คือสัญญาณ PSEN/ จะต่อกับขา RD/ ของหน่วยความจำโปรแกรมภายนอกชิป สัญญาณ PSEN/ จะไม่ถูกใช้งานถ้าเมื่อ MCS-51 ทำงานจากโปรแกรมที่อยู่ในหน่วยความจำโปรแกรมภายในชิป

หน่วยความจำโปรแกรมจะใช้แอดเดรสในการติดต่อขนาด 16 บิตเสมอ ถึงแม้ขนาดของโปรแกรมจะยาวไม่ถึง 64 Kbtye โดยค่าแอดเดรสจะจะส่งผ่านพอร์ต 0 และพอร์ต 2

เมื่อเริ่มจ่ายพลังงานCPUจะเริ่มเฟตช์(fetch)คำสั่งที่หน่วยความจำโปรแกรมของMCS-51 ตำแหน่ง0000Hโดยส่วนหนึ่งจะถูกกำหนดไว้ใช้เป็นโปรแกรมการบริการอินเทอร์รัพท์ แต่ละชนิดโดยถูกกำหนดตำแหน่งที่แน่นอนในหน่วยความจำโปรแกรมดังรูปสัญญาณอินเทอร์รัพท์ที่เกิดขึ้นจะทำให้ CPU ย้ายการทำงานไปทำงานในตำแหน่งการบริการอินเตอร์รัปต์ตามชนิดของสัญญาณอินเทอร์รัพท์ที่เกิดขึ้น

2.2 ขยายพอร์ตโดยใช้ 8255

IC เบอร์ 8255 เป็นIC 40 ขาที่มีพอร์ทแบบขนานสามารถโปรแกรมให้ทำงานเป็นพอร์ทอินพุทหรือพอร์ทเอาท์พุทได้



รูปที่2.6 แสดงโครงสร้างของ IC 8255

2.2.1 ขาของ 8255

D0-D7 เป็นขาข้อมูลอินพุทและเอาท์พุท จึงเป็นส่วนที่ต่อเข้ากับระบบบัสข้อมูลของไมโครคอนโทรลเลอร์

CS/ เป็นขาอินพุทโดยรับสัญญาณจากภายนอกเพื่อเลือกชิป 8255เมื่อขานี้เป็นลอจิก 0 8255 จะต่อเข้ากับระบบบัสของไมโครคอนโทรลเลอร์

RD/ ขาสัญญาณการอ่านที่ส่งมาจาก CPU เพื่ออ่านข้อมูลจากบัสในขณะที่เป็นอินพุทพอร์ต จะมาพร้อมกับสัญญาณ CS/

WR / ขาสัญญาณการอ่านที่ส่งมาจาก CPU เมื่อต้องการต้องการเขียนข้อมูลลงบนพอร์ตที่กำหนด จะมาพร้อมกับสัญญาณ CS/

A0-A1 ขาแอดเดรส ลอจิกทั้งของทั้งสองขาสัญญาณนี้ถอดรหัสได้ 4 ค่ารีจีสเตอร์ภายในที่เชื่อมต่อกับพอร์ตอินพุทและเอาท์พุทของ 8255 เพื่อเลือกใช้พอร์ท A,B,C

RESET ขารีเซต เป็นขาสัญญาณที่ส่งมาจากภายนอกเพื่อทำการรีเซต 8255 เพื่อเคลียร์สถานะต่างๆของ 8255 เมื่อทำการรีเซต 8255 จะกลับสู่โหมดรีเซต

PA0-PA7 เป็นสายสัญญาณที่พอร์ตของ 8255 ชื่อพอร์ต A

PB0-PB7 เป็นสายสัญญาณที่พอร์ตของ 8255 ชื่อพอร์ต B

PC0-PC7 เป็นสายสัญญาณที่พอร์ตของ 8255 ชื่อพอร์ต C

2.2.2 โหมดการทำงานของ8255 การที่จะให้ 8255 จะทำงานต้องให้ขา CS/ แอดทีฟจากนั้นเลือกพอร์ตที่ติดต่อโดย A0-A1 รีจีสเตอร์แต่ละตัวใน 8255 จะได้รับการกำหนดค่าควบคู่กับสัญญาณ RD และ WR โดยการทำงานของ 8255 จะทำงานได้ 3 โหมดคือ

1. Mode 0 : Basic I/O
2. Mode 1 : Strobed I/O
3. Mode 2 : Bidirectional Bus

การทำงานของ 8255 แต่ละโหมดจะเลือกโดยการป้อนคอนโทรลเวิร์ดในโปรแกรมให้กับ 8255

บิตที่	กลุ่ม	ความหมาย
D0	B	พอร์ต C ล่าง 1 = อินพุท 0 = เอาท์พุท
D1	B	พอร์ต B 1 = อินพุท 0 = เอาท์พุท
D2	B	เลือกโหมด 1 = โหมด 0 0 = โหมด 1
D3	A	พอร์ต C บน 1 = อินพุท 0 = เอาท์พุท
D4	A	พอร์ต A 0 = อินพุท 0 = เอาท์พุท
D5 D6	A	เลือกโหมด 00 = โหมด 0 01 = โหมด 1 0X= โหมด 2
D7		โหมดแอคทีฟ 1 = แอคทีฟ

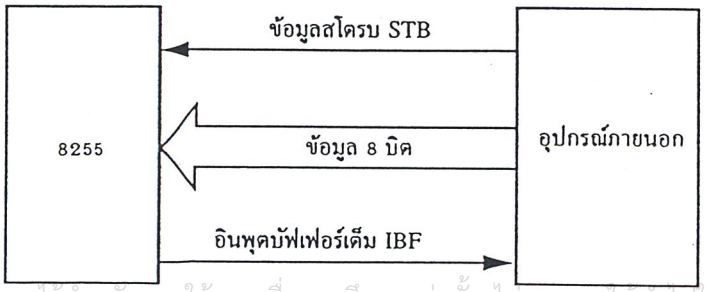
ตารางที่ 2.2 แสดงความหมายของบิตควบคุม

1.Mode 0 เป็นการทำงานแบบอินพุทเอาท์พุทพื้นฐานโดยคอนโทรลเวิร์ดที่ทำให้ พอร์ต A,BและC ทำงานป็นอินพุทเอาท์พุทเป็นตามตารงที่2.3

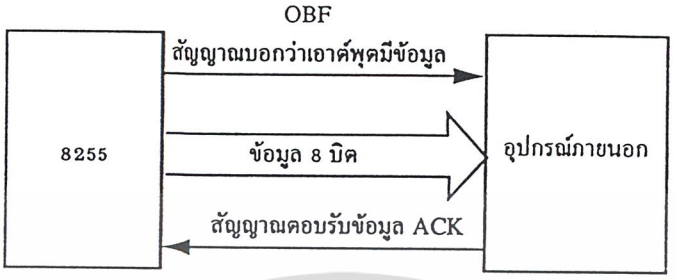
Contlol word	PortA	PortC บน	PortB	Port Cล่าง
80H	Output	Output	Output	Output
81H	Output	Output	Output	Input
82H	Output	Output	Input	Output
83H	Output	Output	Input	Input
88H	Output	Input	Output	Output
89H	Output	Input	Output	Input
8AH	Output	Input	Input	Output
8BH	Output	Input	Input	Input
90H	Input	Output	Output	Output
91H	Input	Output	Output	Input
92H	Input	Output	Input	Output
93H	Input	Output	Input	Input
98H	Input	Input	Output	Output
99H	Input	Input	Output	Input
9AH	Input	Input	Input	Output
9BH	Input	Input	Input	Input

ตารางที่2.3 แสดงหน้าที่ของพอร์ตต่างๆตามControl Word

2. Mode 1 ประกอบด้วยพอร์ตขนาด 8 บิต 2 พอร์ต คือ พอร์ตA และพอร์ตB ซึ่งสามารถโปรแกรมเป็นอินพุทและเอาท์พุทได้ส่วนพอร์ต C บนและล่างจะใช้ทำ Handshaking ของพอร์ตA และ พอร์ต B



รูปที่ 2.7 แสดงการทำงานแบบโหมด 1 อินพุท



รูปที่ 2.8 แสดงการทำงานโหมด 1 เอาท์พุท

3. Mode 2 การทำงานในโหมดนี้จะใช้พอร์ท 8บิตของพอร์ต A เป็นแบบ 2 ทิศทางทั้งอินพุทและเอาท์พุทโดยมีพอร์ต C เป็นสัญญาณบอกสถานะและควบคุมพอร์ต A โดยมีการทำ Handshaking ดังเช่นโหมด 1

2.3 โมดูล LCD

2.3.1 ส่วนประกอบภายในโมดูล LCD มี 3 ส่วน ดังนี้

1. ตัวแสดงผล(Display) ภายในผลึกเหลวที่สามารถจะแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

2. ตัวควบคุม(Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น การลบจอภาพ การแสดงตัวอักษร หรือการเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปควบคุมโดยเฉพาะ ชิปที่นิยมใช้คือ เบอร์ HD44780 และ HD61830 โดย HD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD แบบกราฟิก

3. ตัวขับ(Driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผล แสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่เบอร์ HD44100H และ MSM5259 เป็นต้น

2.3.2 โครงสร้างภายในของตัวควบคุมโมดูล LCD

การใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน

- บัฟเฟอร์อินพุทและเอาท์พุท เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ภายนอก เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

- รีจิสเตอร์คำสั่ง (Instrument Register : IR) เป็นรีจิสเตอร์ใช้รับข้อมูลคำสั่ง จากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

- รีจิสเตอร์ข้อมูล (Data Register : DR) เป็นรีจิสเตอร์ที่ใช้รับข้อมูลจากอุปกรณ์ภายนอก เพื่อถ่ายทอดต่อไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผล หรือนำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

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

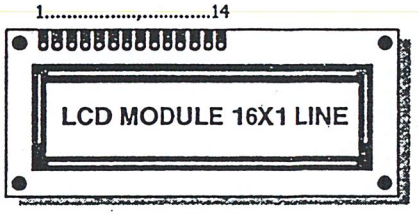
- รอมเก็บตัวอักษร (Character General ROM : CGROM) เป็นหน่วยความจำรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปที่ตัวแสดงผลได้ มีขนาด 7200 บิต โดยจะถูกอ่านด้วยค่าของข้อมูลใน DDRAM

- แรมเก็บตัวอักษร (Character Generater : CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บอักษรที่มีการสร้างเพิ่มเติมขึ้นมาใหม่ ในกรณีที่ตัวอักษรใน CGROM ไม่เพียงพอ มีขนาด 512 บิต การเขียนและการอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGRAM เอง

- แฟลก BUSY เป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่า ตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน

2.3.3 โมดูล LCD ขนาด 16 ตัวอักษร 1 บรรทัด

สำหรับโมดูล LCD ที่ยกมาใช้ในโครงงานชิ้นนี้เป็นขนาด 16 ตัวอักษร 1 บรรทัด เนื่องจากราคาถูก หาง่าย และเป็นโมดูล LCD ที่มีโครงสร้างเป็นมาตรฐาน



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

รูปที่ 2.9 รูปร่างและการจัดขาโมดูล LCD

โมดูล LCD ขนาด 16 x 1 มีขาต่อใช้งานทั้งสิ้น 14 ขา มีการจัดขาดังรูป สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

Vss (ขา 1) ต่อกราวด์

Vdd (ขา 2) ต่อไฟเลี้ยง + 5 โวลต์

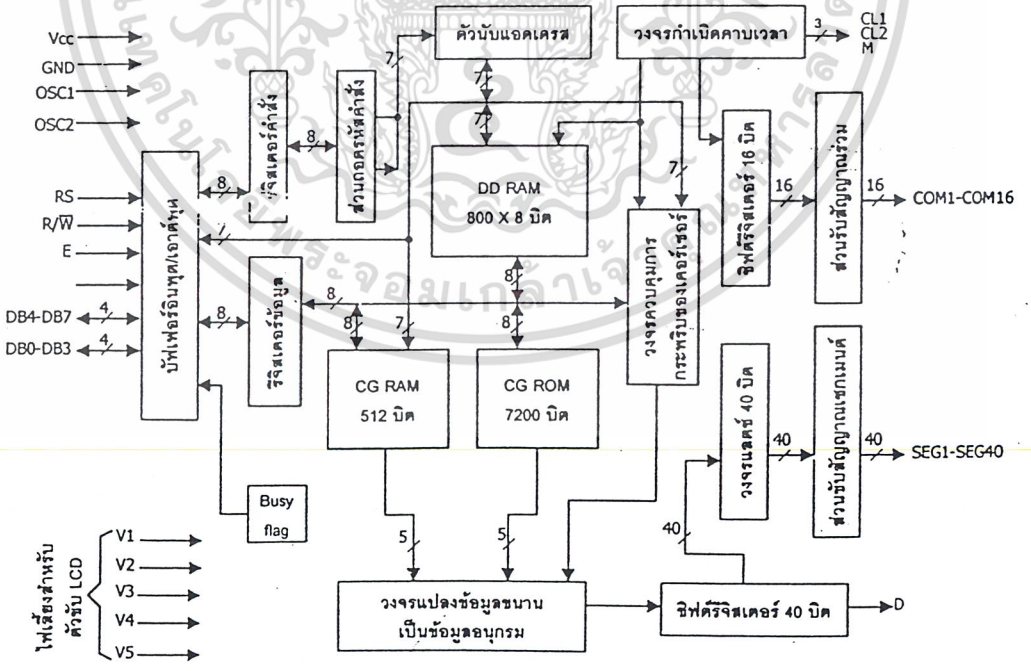
Vo (ขา 3) เป็นขาอินพุทรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา 4) เป็นขาอินพุท ใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้น ว่าเป็นคำสั่งสำหรับรีจิสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจิสเตอร์ DR โดยถ้าขานี้เป็น '0' ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น "1" ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

R/W (ขา 5) เป็นขาที่ใช้เลือกการอ่านหรือเขียนข้อมูลกับ LCD ถ้าเป็น "0" เป็นการกำหนดให้เขียนข้อมูล ถ้าเป็น "1" จะเป็นการอ่านข้อมูล

E (ขา 6) เป็นขาอีนาเบิล LCD ให้ทำงาน

D0-D7 (ขา 7 — ขา 14) เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอกขนาด 8 บิต



รูปที่ 2.10 ไดอะแกรมการทำงานของโมดูลแบบอักขระ

2.3.4 คำสั่งควบคุมโมดูล LCD

การเขียนคำสั่งลงในตัวควบคุม แน่นอนว่าต้องกำหนดให้ขา RS และ ขา R/W เป็น '0' แล้วเขียนคำสั่งตามไป โดยคำสั่งควบคุม LCD

ที่สำคัญมี 10 คำสั่งคือ

1. คำสั่งเคลียร์ตัวแสดงผล (Clear Display)

มีข้อมูลคำสั่งเป็น 01H เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่างหรือ space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมเอ็กซีคิวต์คำสั่งนี้ จะทำการกำหนดแอดเดรสของ DDRAM เป็น 0 เคอร์เซอร์จะกลับไปอยู่ที่ตำแหน่งซ้ายสุดของจอแสดงผลแล้วเซตบิต I/D ให้เป็น "1"

2. คำสั่ง return home

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น "1" เป็นคำสั่งที่ให้เคอร์เซอร์เคลื่อนไปตำแหน่งที่อยู่ซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอจะไม่เปลี่ยนแปลง

3. คำสั่งเลือกโหมดการป้อนข้อมูล (Entry mode Set)

มีรายละเอียดของข้อมูลคำสั่งดังนี้

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

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

บิต I/D เป็นบิตที่ใช้ในการกำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้วทำให้แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น "1" แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าเป็น '0' ค่าแอดเดรสจะลดลง

4. คำสั่งควบคุมการแสดงผล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

บิต D ใช้ควบคุมการเปิดปิดของจอแสดงผล โดยถ้าบิตนี้เป็น "1" จะเป็นการเปิดจอแสดงผล แต่ถ้าเป็น '0' จะเป็นการปิดจอแสดงผล

บิต C ใช้ควบคุมการแสดงตัวเคอร์เซอร์บนจอแสดงผล โดยถ้าบิตนี้เป็น “1” จะมีเคอร์เซอร์บนจอแสดงผล แต่ถ้าเป็น ‘0’ จะเป็นการไม่แสดงเคอร์เซอร์

บิต B ใช้ควบคุมการกระพริบของเคอร์เซอร์ ถ้าเป็น “1” เคอร์เซอร์จะกระพริบ

5. คำสั่งควบคุมการเลื่อนเคอร์เซอร์และข้อมูลตัวอักษร

มีรายละเอียดรูปแบบของคำสั่งดังนี้

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

การควบคุมการเลื่อนเคอร์เซอร์ของตัวอักษรบนจอแสดงผลขึ้นอยู่กับการกำหนดบิต S/C และ R/L ซึ่งสามารถสรุปได้ดังนี้

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

6. คำสั่งกำหนดฟังก์ชันการทำงาน

มีรายละเอียดรูปแบบคำสั่งดังนี้

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

บิต DL ใช้กำหนดจำนวนบิตที่ใช้ติดต่อส่งผ่านข้อมูล โดยถ้าบิตนี้เป็น “1” จะเป็นการติดต่อแบบ 8 บิต แต่ถ้าเป็น ‘0’ จะเป็นการติดต่อแบบ 4 บิต

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล โดยถ้าบิตนี้เป็น “0” จะเป็นการแสดงผล 1 บรรทัด แต่ถ้าเป็น ‘1’ จะเป็นการแสดงผลแบบ 2 บรรทัด

บิต F ใช้เลือกความละเอียดของตัวอักษรให้การแสดงผล โดยถ้าบิตนี้เป็น “0” จะเป็นการแสดงผลแบบ 5x7 จุด แต่ถ้าเป็น ‘1’ จะเป็นการแสดงผลแบบ 5x10 จุด

7. คำสั่งเลือกแอดเดรสของ CGRAM

เมื่อต้องการกำหนดแอดเดรสของ CGRAM ต้องกำหนดให้บิต 7 เป็น “0” บิต 6 เป็น “1” ส่วนอีก 6 บิตที่เหลือจะแสดงด้วยค่าแอดเดรสของ CGRAM จะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ก่อนที่จะอ่านหรือเขียนข้อมูลให้ CGRAM โดย แอดเดรสของ CGRAM อยู่ระหว่าง 00H — 3FH

8. คำสั่งเลือกแอดเดรสของ DDRAM

ใช้ในการเลือกแอดเดรสของ DDRAM ก่อนที่จะทำการอ่านหรือเขียนข้อมูล โดย บิต 7 ต้องเป็น ”1” และข้อมูลอีก 7 บิตที่เหลือจะเป็นค่าแอดเดรสของ DDRAM ซึ่งแอดเดรสของ DDRAM จะอยู่ระหว่าง 8CH — 0FFH ทั้งนี้จำนวนแอดเดรสนั้นยังขึ้นกับการกำหนดสถานะที่บิต N ด้วย โดยถ้าบิตนี้เป็น “0” แอดเดรสของ DDRAM จะอยู่ระหว่าง 80H-0CFH แต่ถ้าเป็น ‘1’ แอดเดรส ของ DDRAM จะมี 2 ช่วงคือ 8CH-87H และ 0C0H-0C7H

9.คำสั่งอ่านแฟลก BUSY และแอดเดรส

บิตที่ 7	บิตที่ 6	บิตที่ 5	บิตที่ 4	บิตที่ 3	บิตที่ 2	บิตที่ 1	บิตที่ 0
BF	A	A	A	A	A	A	A

เป็นคำสั่งที่ใช้อ่านค่าแฟลก BUSY (BF) โดยแฟลกนี้จะเป็นตัวบอกสถานะของตัวควบคุม LCD ว่าพร้อมที่จะรับข้อมูลหรือไม่ถ้าหากบิต BF เป็น “0” แสดงว่าตัวควบคุม LCD พร้อมรับข้อมูลหรือคำสั่ง แต่ถ้าเป็น “1” แสดงว่าขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการภายในยังไม่พร้อมในการรับข้อมูลหรือคำสั่ง

2.3.5 จังหวะการทำงานของ LCD โมดูล

ในการติดต่อกับโมดูล LCD จะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูล เนื่องจากต้องรอให้คอนโทรลเลอร์ภายใน LCD โมดูลแปลความหมายของรหัสคำสั่งนั้นและทำงานตามคำสั่งให้เรียบร้อย จากนั้นจึงจะรับข้อมูลหรือคำเนินการต่อไป

ดังนั้นการใช้งาน LCD โมดูลต้องมีโปรแกรมเพื่อหน่วงเวลารอให้โมดูล LCD พร้อมทำงานด้วย โดยเมื่อเริ่มจ่ายไฟให้แก่โมดูล LCD ต้องรอประมาณ 10 มิลลิวินาที เพื่อให้โมดูล LCD ทำการเตรียมความพร้อม หลังจากนั้นก็จะกำหนดลอจิกให้แก่ขาของโมดูล LCD แล้วต้องหน่วงเวลาอีกประมาณ 2 มิลลิวินาที เพื่อให้คอนโทรลเลอร์ใน LCD โมดูล แปลความหมายของลอจิกที่ขา RS ว่าข้อมูลต่อไปที่จะได้รับนั้นเป็นรหัสคำสั่ง หรือเป็นข้อมูลที่ต้องการแสดงผล จากนั้นจะเป็นการส่งข้อมูลมารอที่บัสข้อมูล D0 — D7 ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ไปที่

ขา E เพื่ออีนาเบิลโมดูล LCD สำรับข้อมูลจากบัสข้อมูลเข้าไป โดยต้องเป็นพัลส์ขอบขาขึ้นจากนั้นทำการหน่วงเวลาอีก 2 มิลลิวินาที

ทั้งหมดที่กล่าวมาคือขั้นตอนและจังหวะในการทำงาน 1 รอบของโมดูลLCD จะเห็นได้ว่ามีโปรแกรมย่อยที่สำคัญอยู่ 3 โปรแกรมคือ โปรแกรมอินนิเชียล LCD , โปรแกรมหน่วงเวลา และ โปรแกรมย่อยการส่งพัลส์เพื่อ อีนาเบิล(Enable)โมดูล LCD

2.4 <u>การใช้งานไอซีสร้างฐานเวลาจริง</u> (Real time clock)

ในโครงงานชิ้นนี้ เลือกใช้ DS 1307 โดยผู้ผลิตคือ ดัลลัสเซมิคอนดักเตอร์ (Dallas semiconductor) ทีหน้าที่สร้างฐานเวลาจริงให้แก่ระบบไมโครคอนโทรลเลอร์ โดย DS 1307 จะให้ข้อมูลเกี่ยวกับฐานเวลาจริงทั้งหมด ไม่ว่าจะเป็นค่าของเวลาที่ละเอียดถึงหลักวินาที, นาที, ชั่วโมง, วันที่(date), วันในสัปดาห์ (day), เดือน และปี โดยสามารถปรับวันเดือนปี ให้ตรงตามปฏิทินได้อย่างถูกต้อง รวมถึงการกำหนดวันในปี อธิกสุรทินด้วย

2.4.1 คุณสมบัติทางเทคนิค

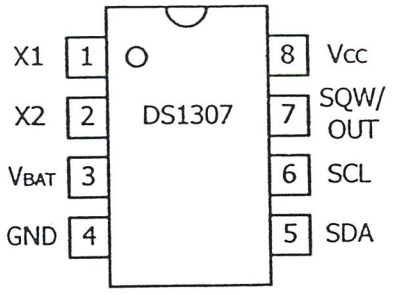
- เป็นไอซีรีลไทม์คล็อกให้ข้อมูลตั้งแต่วินาทีจนถึงปี รวมถึงการกำหนดวันในปีอธิกสุรทินด้วยสามารถให้ข้อมูลเวลาได้อย่างเที่ยงตรงถึงปีคริสตศักราช 2100
- มีหน่วยความจำนอนโวลาไทล์แรม 56 ไบต์อยู่ภายใน สามารถใช้เก็บข้อมูลทั่วไปได้
- ใช้การเชื่อมต่อแบบระบบบัส I^2C
- มีวงจรตรวจจับไฟเลี้ยงต่ำหรือหายไปอย่างอัตโนมัติ และสามารถรักษาข้อมูลเวลาได แม้ไม่มีไฟเลี้ยงไอซี

2.4.2 รายละเอียดขาต่อใช้งานของ DS 1307

ในรูปที่ 2.11 แสดงการจัดขาของ DS 1307 แต่ละขามีหน้าที่และการใช้งานดังนี้

- V_{CC}, GND (ขา 8, 4) ต่อกับไฟเลี้ยง +5V

- V_{BAT} (ขา 3) ใช้ต่อกับแบตเตอรี่ 3V เพื่อรักษาการทำงานของวงจรสร้างฐานเวลาของ DS 1307 ให้คงอยู่ต่อไป แม้ว่าไม่มีไฟเลี้ยงจ่ายให้แก่ DS1307 ชนิดของแบตเตอรี่ที่เหมาะสมคือแบตเตอรี่แบบลิเธียม ซึ่งมีความจุ 40 mAhr หรือมากกว่า จะสามารถรักษาข้อมูลได้นาน 10 ปีที่อุณหภูมิ 25 องศาเซลเซียส



รูปที่ **2.11** แสดงการจัดขาของ DS 1307

- SDA, SCL (ขา 5 และ 6) เป็นขาสำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์บนระบบบัส I^2C

- SQW/OUT (ขา 7) ที่ขานี้จะมีสัญญาณรูปสี่เหลี่ยมส่งออกมา โดยสามารถเลือกความถี่ได้ 1Hz, 4.096kHz, 8.192kHz, 32kHz ในการใช้งานต้องมีการต่อตัวต้านทาน 1k พูลอัปที่ขานี้ด้วย

- X1, X2 (ขา 1 และ 2) ใช้ต่อกับคริสตอลความถี่มาตรฐาน 32.768kHz เพื่อใช้เป็นฐานเวลาในการสร้างค่าเวลาจริง ในการใช้งานต้องต่อคริสตอล เข้ากับขาทั้งสองนี้และที่แต่ละขาต้องต่อตัวเก็บประจุค่าต่ำๆ ประมาณ 15 pF คร่อมกับขากราวด์ด้วย

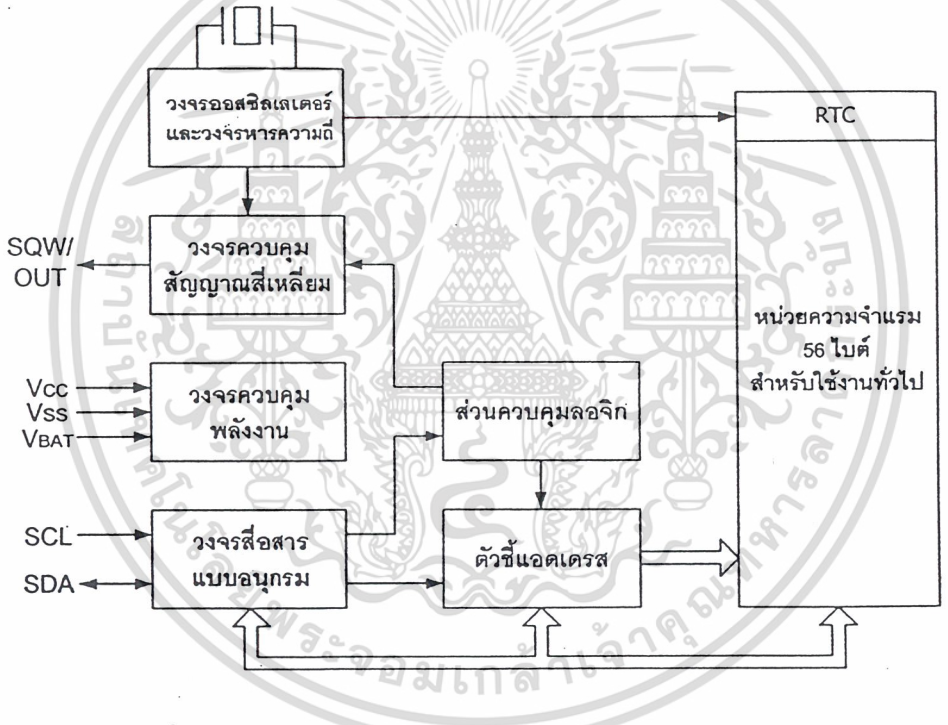
2.4.3 การทำงานของ DS 1307

ไอซี DS1307 จัดการเชื่อมต่อในระบบบัส I^2C โดยจะทำงานเป็นอุปกรณ์สเลฟเสมอ ดังนั้นการติดต่อเพื่อการใช้งานจึงต้องกำหนดรูปแบบตามที่กำหนดไว้ในการติดต่อแบบ I^2C ในรูปที่แสดงส่วนประกอบหลักที่สำคัญและไดอะแกรมการทำงานของ DS1307 วงจรออสซิลเลเตอร์ถือเป็นหัวใจหลักของไอซี เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลเวลาจริง ในขณะที่ DS1307 ทำงาน ที่ขา SQW/OUT จะมีสัญญาณพัลส์สี่เหลี่ยมส่งออกมาตลอดเวลาในกรณีที่มีการ อีนาเบิลวงจรกำเนิดสัญญาณพัลส์ที่รีจิสเตอร์ควบคุม ค่าความถี่ของสัญญาณนี้สามารถเลือกได้ 4 ค่าคือ 1Hz, 4.096kHz, 8.192kHz และ 32 kHz พร้อมกันนั้นก็จะมีการเก็บค่าของเวลาในหน่วยความจำนอนโวลาไทล์แรม ซึ่งมีขนาดความจำรวม 64 ไบต์ แต่จัดสรรให้ใช้เก็บข้อมูลเวลา 8 ไบต์ และเป็นหน่วยความจำสำหรับเก็บข้อมูลทั่วไปสำหรับผู้ใช้งานอีก 56 ไบต์

วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสภาวะของไฟเลี้ยงไอซี หากไฟเลี้ยงต่ำกว่า 1.25 x V_{BAT} ก็จะควบคุมให้ DS1307 หยุดการทำงานรีเซตค่าตัวนับแอดเดรสภายในทำให้ไม่สามารถติดต่อกับ DS1307 ได้ ดังนั้นการใช้งาน DS1307 ต้องระมัดระวังอย่าให้ไฟเลี้ยงตกต่ำกว่า 1.25 x V_{BAT} หรือประมาณ 3.75 โวลท์ ในกรณีที่ใช้ V_{BAT} เท่ากับ 3 โวลท์ ถ้าหากไฟเลี้ยงมีค่าต่ำกว่า V_{BAT} ไอซี DS1307 จะเข้าสู่โหมดสำรองข้อมูลกระแสต่ำทันที จะไม่มีการส่งสัญญาณพัลส์

ออกมาที่ขา SQW/OUT แต่วงจรสร้างฐานเวลายังคงทำงานเพื่อให้ค่าของข้อมูลเวลาเดินไปอย่างไม่ผิดพลาด เมื่อมีไฟเลี้ยงปรากฏขึ้นอีกครั้ง DS1307 ก็จะสามารถให้ค่าของเวลาที่เป็นจริงแก่ผู้ใช้งานได้ต่อไป

วงจรสื่อสารอนุกรมภายใน DS1307 ได้รับการกำหนดให้ทำงานตามรูปแบบของบัส I^2C เป็นช่องทางการสื่อสารระหว่าง DS1307 กับอุปกรณ์มาสเตอร์ ผู้ใช้งานสามารถเข้าถึงหน่วยความจำที่ใช้เก็บค่าเวลาและหน่วยความจำใช้งานทั่วไปได้โดยการเขียนข้อมูลตามรูปแบบที่กำหนดในระบบ I^2C



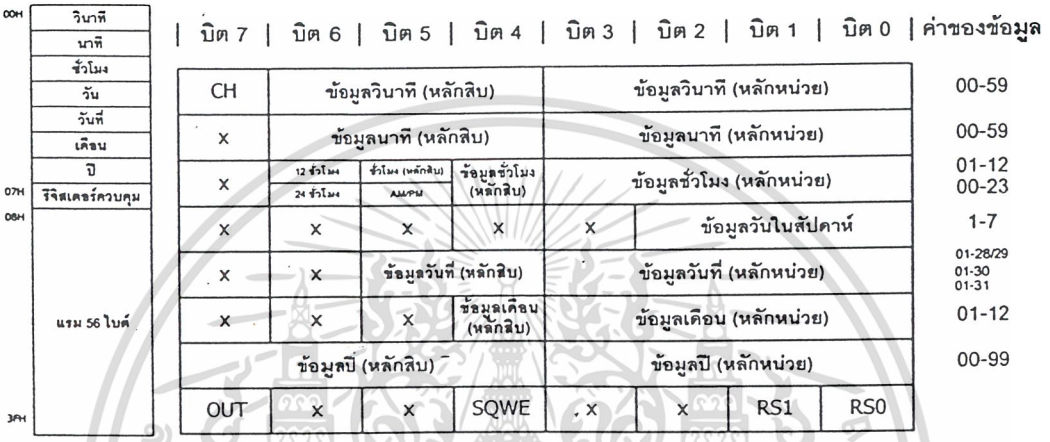
รูปที่ 2.12 โครงสร้างภายในของ IC Real time clock เบอร์ DS1307.

2.4.4 การจัดสรรหน่วยความจำใน DS1307

ในรูปที่ 12 แสดงการจักสรรพื้นที่ของหน่วยความจำภายใน DS1307 พื้นที่ 7 ไบต์แรกตั้งแต่แอดเดรส 00H-06H เป็นพื้นที่ของรีจิสเตอร์ค่าเวลาที่ใช้ในการเก็บข้อมูลเกี่ยวกับเวลา ไบต์ต่อมาที่แอดเดรส 07H เป็นที่พื้นที่ของรีจิสเตอร์ควบคุมการทำงานของ DS1307 ในรูปที่ แสดงราบละเอียดของรีจิสเตอร์ค่าเวลาและรีจิสเตอร์ควบคุมของ DS1307

ด้วยการจัดสรรพื้นที่แบบนี้ทำให้ผู้ใช้งานสามารถเรียกข้อมูลออกมาได้ตามที่ต้องการ โดยไม่จำเป็นต้องอ่านออกมาทั้งหมดก็ได้ ค่าของเวลาทั้งหมดจะอยู่ในรูปของเลขฐานสิบ สำหรับการแสดงเวลาในรูปของชั่วโมง สามารถเลือกได้ว่าต้องการแบบ 12 ชั่วโมง หรือ 24 ชั่วโมง โดย

กำหนดบิตที่ 6 ของแอดเดรส 02H และเมื่อเลือกแบบ 12 ชั่วโมง ที่บิตที่ 5 ในแอดเดรสเดียวกันจะใช้ในการแสดงค่า AM/PM โดยถ้าบิตนี้เป็น “1” หมายถึง ค่าชั่วโมงในขณะนี้เป็นช่วงหลังเที่ยงวัน ในกรณีที่เป็นแบบ 24 ชั่วโมง บิตนี้จะใช้ในการแสดงค่า 2 ของหลักสิบในหน่วยชั่วโมง



รูปที่ 2.13 แสดงการจัดสรรพื้นที่ของหน่วยความจำภายใน DS1307

2.4.5 รีจิสเตอร์ควบคุม

มีแอดเดรสที่ 07H มีรายละเอียดของแต่ละบิตดังนี้

- OUT (Outputcontrol) : ใช้ในการควบคุมระดับลอจิกที่ขา SWQ/OUT ในกรณีที่ดิสเอเบิลการกำเนิดสัญญาณสี่เหลี่ยม โดยถ้าบิตนี้เป็น “1” ที่ขา SWQ/OUT ก็จะเป็น “1” ถ้าบิตนี้เป็น “0” ที่ขา SWQ/OUT ก็จะเป็น “0”

- SQWE (Square Wave Enable) : ใช้ในการอินาเบิลวงจรกำเนิดสัญญาณสี่เหลี่ยมที่ขา SQW/OUT ถ้าต้องการให้มีสัญญาณสี่เหลี่ยมออกมาให้กำหนดบิตรี้เป็น “1”

- RS0 , RS1 (Rate Select) : ใช้ในการเลือกความถี่ของสัญญาณสี่เหลี่ยม ที่ออกจากของ SQW/OUT มีรายละเอียดดังต่อไปนี้

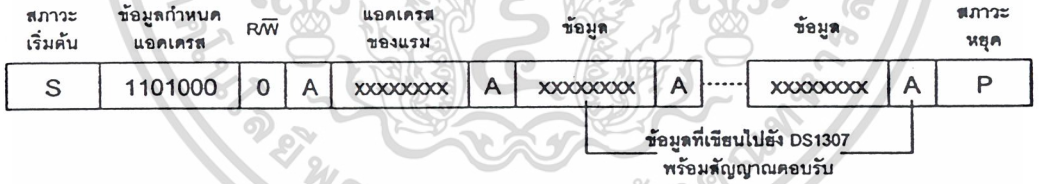
RS1	RS0	ค่าความถี่ของสัญญาณสี่เหลี่ยม
0	0	1 Hz
0	1	4.096Hz
1	0	8.192Hz
1	1	32.768Hz

2.4.6 โหมดการทำงานของ DS1307

มีด้วยกัน 2 โหมดคือโหมดเขียนและอ่านข้อมูล ในการใช้งาน DS1307 ตามปกติ จะใช้งานเฉพาะโหมดอ่านข้อมูลเท่านั้น เนื่องจากไมโครคอนโทรลเลอร์จะติดต่อกับ DS1307 เพื่ออ่านข้อมูลของเวลาไปใช้งาน โหมดการเขียนข้อมูลจะถูกใช้งานก็ต่อเมื่อต้องการตั้งค่าเวลาใหม่และต้องเขียนข้อมูลลงในหน่วยความจำใช้งานทั่วไป อย่างไรก็ตามเมื่อเริ่มต้นติดต่อกับ DS1307 จำเป็นอย่างยิ่งที่จะต้องเข้าสู่โหมดการเขียนข้อมูลก่อนเพื่อกำหนดแอดเดรสที่ต้องการอ่านข้อมูล จากนั้นจึงเปลี่ยนโหมดการทำงานมาเป็นการอ่านข้อมูลต่อไป

- โหมดการเขียนข้อมูล

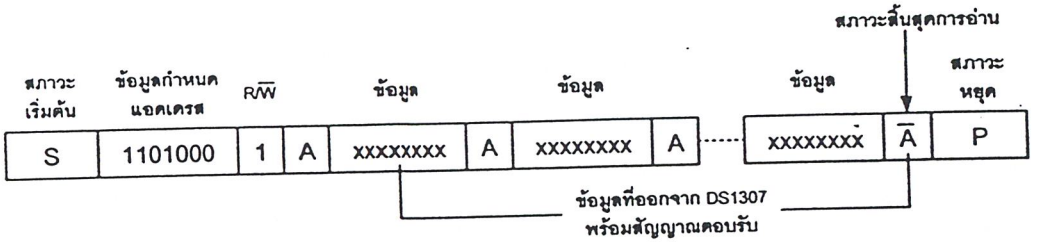
มีรูปแบบดังรูปที่ 2.13 เริ่มต้นเมื่อไมโครคอนโทรลเลอร์ ทำการกำหนดสภาวะเริ่มต้น (START:S) จากนั้นส่งข้อมูลกำหนดแอดเดรส 1101000 ตามด้วยข้อมูลเลือกการเขียนนั่นคือค่า 0 จากนั้นจะรอการตอบรับจาก DS1307 ขั้นตอนต่อมาคือ ส่งข้อมูลเพื่อเลือกแอดเดรสที่ต้องการเขียนจากนั้นรอการตอบรับจาก DS1307 เมื่อมีการตอบรับมาเรียบร้อย ก็เริ่มทะยอยเขียนข้อมูลลงไปครั้งละแอดเดรส หลังจากเขียนข้อมูลในแต่ละแอดเดรส จะต้องหยุดการตอบรับจาก DS1307 ทุกครั้งจึงสามารถเขียนข้อมูลต่อไปได้ เมื่อเขียนเรียบร้อยแล้วก็ให้ส่งสภาวะหยุด (STOP:P) เป็นอันสิ้นสุดกระบวนการเขียนข้อมูล



รูปที่ 2.14 แสดงรูปแบบของข้อมูลสำหรับติดต่อกับ DS1307 ในโหมดการเขียนข้อมูล

- โหมดการอ่านข้อมูล

มีรูปแบบดังรูปที่ 2.14 เริ่มต้นการทำงานเหมือนกับโหมดการเขียนข้อมูลคือ ไมโครคอนโทรลเลอร์กำหนดสภาวะเริ่มต้นแล้วส่งข้อมูลกำหนดแอดเดรสตามด้วยข้อมูลเลือกการอ่านซึ่งเท่ากับ "1" จากนั้นรอคอยการตอบรับจาก DS1307 เมื่อตอบรับเรียบร้อย DS1307 จะทยอยส่งข้อมูลออกมาให้ไมโครคอนโทรลเลอร์คราวละ 1 แอดเดรสหรือ 1 ไบต์ โดยแอดเดรสที่เลือกอ่านข้อมูลจะต้องมีการกำหนดล่วงหน้าด้วยโหมดการเขียนข้อมูล วิธีการง่ายๆ คือ เข้าสู่โหมดการเขียนข้อมูล ให้ทำการสร้างสภาวะเริ่มต้นและส่งข้อมูลกำหนดแอดเดรสใหม่อีกครั้ง ตามด้วยเลือโหมดการอ่านข้อมูล ข้อมูลที่ออกมาจาก DS1307 ก็จะเป็นข้อมูลจากแอดเดรสที่กำหนดไว้ก่อนนี้



รูปที่ **2.15** แสดงรูปแบบของข้อมูลสำหรับการติดต่อกับ DS1307 ในโหมดการอ่านข้อมูล

2.5 การสื่อสารข้อมูลแบบอนุกรม

การเคลื่อนย้ายข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์ต่อพ่วงนั้นมีอยู่ด้วยกัน 2 ทางเลือกคือ การรับส่งแบบขนานและอนุกรม โดยการส่งข้อมูลแบบอนุกรมนั้นสามารถส่งข้อมูลคราวละ 4 หรือ 8 บิตจึงทำให้การรับส่งข้อมูลทำได้ที่ความเร็วสูง โดยการส่งข้อมูลแบบอนุกรมเป็นการรับส่งข้อมูลครั้งละ 1 บิต สามารถส่งข้อมูลคราวละหลายๆบิตได้ แต่ต้องมีการตกลงกันระหว่างตัวส่งและตัวรับว่า จะรับส่งข้อมูลคราวละกี่บิต ตัวรับจะต้องรับข้อมูลมาทุกบิตเสียก่อนจะทำการประมวลผล ดังนั้นการสื่อสารแบบอนุกรมจะทำได้ช้ากว่าแบบขนาน แต่ในด้านของจำนวนสายสัญญาณนั้นจะน้อยกว่าและยาวกว่า

2.5.1 มาตรฐาน EIA RS-232

การเชื่อมต่อระหว่างพอร์ตใช้เส้นสัญญาณอนุกรมมาตรฐาน 3 แบบ คือ มาตรฐาน EIA RS-422, ระบบวงรอบกระแส, และมาตรฐาน EIA RS-232 ซึ่งเป็นระบบสัญญาณอนุกรมที่ใช้มากที่สุด โดยจะใช้ในหน่วยแสดงผล เครื่องพิมพ์ โมเดม และอุปกรณ์อื่นๆ ซึ่งมีความยาวของสายไม่เกิน 50 ฟุต

มาตรฐาน EIA RS-232 ได้กำหนดให้ค่าสัญญาณไฟฟ้าที่มีระดับศักดาไฟฟ้าเท่ากับ 3 โวลต์ หรือสูงกว่าจะมีค่าทางตรรกะเป็น 1 และกำหนดสัญญาณไฟฟ้าที่มีระดับศักดาไฟฟ้าเท่ากับ –3 โวลต์ หรือต่ำกว่าจะมีค่าทางตรรกะเป็น 0 RS-232 จะใช้สัญญาณ 1 เส้นสำหรับส่งข้อมูลและอีกหนึ่งเส้นสำหรับรับข้อมูล โดยสัญญาณในสายสัญญาณแต่ละเส้นถูกอ้างอิงเทียบกับกราวด์ และ RS-232 ยังได้กำหนดสัญญาณตอบรับเพื่อใช้ในการควบคุมการรับ/ส่งข้อมูลด้วย

มาตรฐาน EIA RS-232 สามารถส่งสัญญาณได้ไกลสุด 50 เมตร ด้วยอัตรา 9600 บอด ถ้าต้องการส่งได้ไกลกว่านั้นจะต้องส่งข้อมูลด้วยอัตราที่ช้ากว่านี้ หรือในทางกลับกันถ้าต้องการส่งสัญญาณใกล้กว่านี้จะส่งข้อมูลด้วยอัตราที่เร็วขึ้น

ในตารางที่ 2.4 แสดงตัวเชื่อมแบบ DB-25 มี 25 ขาซึ่งยังไม่เป็นรูปแบบที่มาตรฐานเนื่องจากกินเนื้อที่มากในการติดตั้ง ดังนั้นจึงได้มีการเชื่อมต่อแบบใหม่เป็นแบบ DB-9 มี 9 ขามาใช้แทนซึ่งเป็นที่นิยมใช้ในเครื่อง Computer PC รุ่นใหม่

DB-9	DB-25	ชื่อสายสัญญาณ	ชนิดของสายสัญญาณ
1	8	Data carier detect : DCD	Input
2	3	Received Data : RxD	Input
3	2	Transmitted Data : TxD	Output
4	20	Data Terminal Ready : DTR	Output
5	7	Signal Ground : GND	-
6	6	Data Set Ready : DSR	Input
7	4	Request To Send	Output
8	5	Clear To Send : CTS	Input
9	22	Ring Indicator :RI	Input

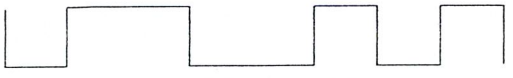
ตารางที่ 2.4 ขาตัวเชื่อมต่อที่ใช้กับพอร์ตอนุกรม RS-232 แบบ DB-9 และ DB-25

2.5.2 การสื่อสารพอร์ตอนุกรมของไมโครคอนโทรลเลอร์

การส่งข้อมูลแบบอนุกรมมีโครงสร้างที่สำคัญหลายอย่างเช่นความเร็วในการส่งข้อมูลที่เรียกว่า Baud rate มีหน่วยเป็น บิต/วินาที เช่นความเร็ว 1200, 2400, 4800 หรือ 9600 เป็น หากใช้ความเร็ว 2400 บิต/วินาทีจะใช้เวลาในการส่ง 1 บิต เท่ากับ 1/2400 วินาที ซึ่งสามารถคำนวณเวลาการส่งข้อมูลได้ รูปแบบของสัญญาณการรับส่งข้อมูลแบบอนุกรมเป็นดังรูป

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	0	0	1	1	0

ข้อมูล 8 บิต ที่ต้องการส่ง



D0 D1 D2 D3 D4 D5 D6 D7

สัญญาณที่ส่งออก โดยบิต 0 ออกมาก่อน

รูปที่ 2.16 รูปแบบของสัญญาณการส่งข้อมูลแบบอนุกรม

ภายในไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีพอร์ตการสื่อสารแบบอนุกรม (Serial Port) ที่สามารถรับและส่งข้อมูลแบบ Full Duplex อยู่ 1 พอร์ต การรับส่งข้อมูลแบบ Full Duplex หมายถึงว่าพอร์อนุกรมสามารถรับและส่งข้อมูลในเวลาเดียวกัน การควบคุมการทำงานของพอร์ตอนุกรมทางด้านส่งและรับข้อมูล ทำโดยการกำหนดค่าให้กับรีจิสเตอร์ควบคุมการทำงานของพอร์อนุกรมชื่อ SCON

2.5.2.1 รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม SCON (Serial Port Control Register)

รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม SCON เป็นรีจิสเตอร์เฉพาะที่ทำหน้าที่ควบคุมโหมดการทำงานของพอร์ตอนุกรม และเป็นที่เก็บข้อมูลบิตที่ 9 ของการรับและส่งข้อมูล (บิต TB8 และ RB8) และมีแฟลกของการร้องขออินเตอร์รัพต์ของพอร์ตอนุกรมรวมอยู่ด้วย บิตต่าง ๆ แสดงดังรูปที่ ??? การควบคุมการทำงานเราจะกำหนดบิตต่างๆ ในรีจิสเตอร์ตัวนี้ด้วยคำสั่งการโอนย้ายข้อมูลหรือใช้คำสั่งการเซตหรือเคลียบิตก็ได้เนื่องจากรีจิสเตอร์ SCON สามารถอ้างในระดับบิตได้

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

รูปที่ 2.17 บิตต่างๆ ของรีจิสเตอร์ SCON

ความหมายของบิตต่างๆ ในรีจิสเตอร์ SCON เป็นดังนี้

- SM0 และ SM1: เป็นบิตกำหนดโหมดการทำงานของพอร์ตอนุกรมซึ่งมี 4 โหมดดังนี้

SM0	SM1	Mode	การทำงาน	อัตราการรับส่ง
0	0	0	Shift register	$F_{OSC}/12$
0	1	1	8 bit UART	Variable
1	0	2	9 bit UART	$f_{OSC}/32$ หรือ $f_{OSC}/64$
1	1	3	9 bit UART	Variable

รูปที่ 2.18 การเลือกโหมดการทำงานของพอร์ตอนุกรม

- SM2: เป็นบิตที่ควบคุมให้ทำงานในลักษณะการเชื่อมต่อไมโครคอนโทรลเลอร์หลายตัวเข้าด้วยกัน สำหรับการใช้งานในโหมด 2 หรือ โหมด 3 เป็นดังนี้

SM2 = 1 จะทำให้แฟลกอินเตอร์รัพต์ทางด้านรับ (RI) ไม่ถูกเซตเมื่อรับข้อมูลเข้ามาแล้วมีค่าบิตที่ 9 เป็น 0 (อยู่ในบิต RB8) สำหรับการทำงานในโหมด 1 ถ้าเซต SM2 = 1 แฟลกอินเตอร์

รัพท์ทางด้านรับ (แฟลก RI) จะไม่ถูกเซตหากข้อมูลที่รับเข้ามาไม่มี STOP BIT การใช้งานในโหมด 0 ต้องกำหนด SM2 = 0

- REN: เซตหรือรีเซตด้วยซอฟท์แวร์เป็นตัวควบคุมการรับข้อมูลของพอร์ตอนุกรมดังนี้

1 = ให้มีการรับข้อมูล

0 = ไม่ให้มีการรับข้อมูล

- TB8: เป็นข้อมูลบิตที่ 9 ที่ต้องการส่งในโหมด 2 และ 3 สามารถเซตและเคลียร์ด้วยซอฟต์แวร์

- RB8: เป็นบิตเก็บข้อมูลที่รับเข้ามาบิตที่ 9 ในโหมด 2 และ 3 สำหรับการทำงานในโหมด 1 หากกำหนดให้บิต SM2 = 0 บิต R8 จะเป็นค่าของ STOP BIT ที่รับเข้ามาสำหรับโหมด 0 ไม่มีการใช้ RB8

- TI: แฟลของการอินเตอร์รัพท์ด้านส่งข้อมูลแฟลกนี้จะถูกเซตด้านฮาร์ดแวร์เมื่อจบการส่งข้อมูลบิตที่ 8 ในโหมด 0 หรือเมื่อเริ่มต้นส่ง STOP BIT ในโหมด 1, 2 หรือ 3 เราต้องเคลียแฟลกนี้ด้วยซอฟต์แวร์ เมื่อจบโปรแกรมการตอบสนองการอินเตอร์รัพต์ของการส่งข้อมูลแล้ว

- RI: แฟลกอินเตอร์รัพต์ด้านการรับข้อมูล ถูกเซตด้วยฮาร์ดแวร์เมื่อข้อมูลบิตที่ 8 ในโหมด 0 ถูกรับเข้ามาในครึ่งเวลาแรกในโหมด 1, 2 หรือ 3 เราต้องเคลียร์แฟลกนี้ด้วยซอฟต์แวร์เมื่อจบโปรแกรมตอบสนองการอินเตอร์รัพต์ของการรับข้อมูลแล้ว

2.5.2.2 การเลือกโหมดการทำงาน

การเลือกโหมดการทำงานของพอร์ตการทำงานของพอร์ตอนุกรม สามารถกำหนดโหมดต่างๆ ได้ดังรูป

MODE	รีจิสเตอร์ SCON	บิต SM2 Variation
0	10H	Single processor
1	50H	Environment
2	90H	(SM2 = 0)
3	D0H	
0	NA	
1	70H	Multiprocessor
2	B0H	Environment
3	F0H	(SM2 = 1)

รูปที่ 2.19 การกำหนดค่าให้กับรีจิสเตอร์ SCON เพื่อเลือกโหมดการทำงาน

การส่งและรับข้อมูลของพอร์ตอนุกรมจะมีรีจิสเตอร์ที่ทำหน้าที่รับและส่งข้อมูลอยู่ 1 ตัว คือรีจิสเตอร์ SBUF การส่งข้อมูลออกไปยังพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ทำโดยการใส่ข้อมูลลงไปในรีจิสเตอร์ SBUF การอ่านข้อมูลจากภายนอกจะที่รับเข้ามาทางพอร์ตอนุกรมจะอ่านจากรีจิสเตอร์ SBUF เช่นกัน วงจรด้านรับจะมีบัฟเฟอร์ขนาด 1 ไบต์ที่ทำหน้าที่เก็บข้อมูลที่รับเข้ามาประกอบอยู่ภายใน (การมีบัฟเฟอร์รับข้อมูลทำให้สามารถรับข้อมูลไบต์ที่ 2 เข้ามาได้ทันทีหลังจากข้อมูลไบต์แรกเข้ามาแล้วแม้ยังไม่ถูกอ่านออกไป แต่ถ้าข้อมูลไบต์แรกยังไม่ถูกอ่านก่อนที่ข้อมูลไบต์ที่ 2 จะเข้ามาครบ ข้อมูลไบต์ที่ 2 จะถูกยกเลิก)

การทำงานของพอร์ตอนุกรมของไมโครคอนโทรลเลอร์สามารถแบ่งออกเป็น 4 โหมด คือ โหมด 0, 1, 2 และ 3 แต่ละโหมดมีการทำงานดังนี้

Mode 0

ข้อมูลขนาด 8 บิตอนุกรมรับเข้ามาทางขา RXD และข้อมูล 8 บิตที่ส่งออกแบบอนุกรมทางขา TXD การรับและส่งข้อมูลจะเริ่มจากบิตต่ำ (LSB) ก่อน อัตราการรับส่งข้อมูล (Baud rate) จะเป็น 1/12 เท่าของสัญญาณนาฬิกา

Mode 1

ใช้ในการรับและส่งข้อมูลแบบ 10 บิต ซึ่งข้อมูลอนุกรม 10 บิตเข้ามาทางขา RXD และข้อมูลขนาด 10 บิตส่งออกแบบอนุกรมทางขา TXD โดยข้อมูล 10 บิต ประกอบด้วย Start bit (ค่า 0), 8 บิตข้อมูล (การรับการส่งข้อมูลจะเริ่มจากบิตต่ำก่อน) และ 1 Stop bit (ค่า 1) ด้านรับข้อมูลจะนำค่า Stop bit ที่รับเข้ามาไปเก็บในบิต RB8 ที่อยู่ในรีจิสเตอร์ SCON อัตราการรับและการส่งข้อมูลในโหมดนี้สามารถกำหนดได้ตามต้องการ

Mode 2

ใช้การรับและส่งข้อมูลขนาด 11 บิต ข้อมูลแบบอนุกรมรับเข้ามาทาง RXD และส่งออกไปทาง TXD ซึ่งข้อมูลขนาด 11 บิต ประกอบไปด้วย 1 Start bit (ค่า 0), 8 บิตข้อมูล (การรับข้อมูลเริ่มจากบิตก่อน), 8 บิตข้อมูลที่กำหนดค่าได้ และ 1 Stop bit (ค่า 1) สำหรับด้านส่งข้อมูลบิตที่ 9 จะกำหนดไว้ใน TB8 ที่อยู่ในรีจิสเตอร์ SCON ซึ่งเราสามารถกำหนดเป็น 1 หรือ 0 ก็ได้ ในการใช้งานเราอาจใช้บิตข้อมูลบิตที่ 9 เป็นบิตตรวจสอบ (Parity) ก็ได้โดยการนำค่าในแฟลก P มากำหนดให้กับ TB8 เมื่อข้อมูล 8 มาจากแอคคิวมูเลเตอร์ซึ่งจะทำให้เราได้การตรวจสอบการส่งข้อมูลแบบพาริตี้คู่ ในกรณีการรับข้อมูลจะนำข้อมูลบิตที่ 9 ที่รับเข้ามาไปเก็บไว้ในบิต RB8 ที่อยู่ในรีจิสเตอร์ SCON ส่วน Stop bit จะไม่มีการนำมาเก็บ อัตราการรับและส่งข้อมูลในโหมดนี้จะเลือกใช้ความเร็วได้ 2 ค่าคือ 1/32 และ 1/64 เท่าของสัญญาณนาฬิกา

Mode 3

การทำงานในโหมด 3 นี้ จะมีลักษณะคล้ายกับโหมด 2 แต่ในโหมด 3 เราสามารถกำหนดอัตราการรับและส่งข้อมูลได้ตามต้องการ

การทำงานของพอร์ตอนุกรมทั้ง 4 โหมด การส่งข้อมูลจะเริ่มต้นเมื่อมีการกำหนดข้อมูลให้กับรีจิสเตอร์ SBUF ซึ่งการกำหนดค่าให้กับรีจิสเตอร์ SBUF จะใช้คำสั่งการโอนย้ายข้อมูลเช่น MOV SBUF,#15H หรือ MOV SBUF,@R1 ก็ได้ การรับข้อมูลในโหมด 0 จะเริ่มต้นรับข้อมูลเมื่อค่าของบิต RI = 0 และ REN = 1 ส่วนในโหมดอื่นๆ การรับข้อมูลจะเริ่มต้นเมื่อกำหนดบิต REN = 1 และมี Start bit เข้ามาที่ขา RXD

2.5.2.3 การกำหนดอัตราการรับส่ง (Baud rate Generator)

อัตราการรับส่งข้อมูลแบบอนุกรมเป็นความเร็วในหน่วย บิต/วินาที ซึ่งเราสามารถกำหนดอัตรการรับส่งที่แตกต่างกันได้ในโหมด 1 และ โหมด 3 สำหรับอัตรารับส่งในโหมด 0 จะคงที่ส่วนในโหมด 2 เลือกได้ 2 อัตราความเร็ว

- **อัตรารับส่งในโหมด 0** จะใช้สัญญาณนาฬิกาของระบบไมโครคอนโทรลเลอร์เป็นฐาน คือ อัตราการรับส่งโหมด 0 = ความถี่สัญญาณนาฬิกาของไมโครคอนโทรลเลอร์/12

- **อัตรารับส่งในโหมด 2** เลือกได้ 2 อัตราความเร็วขึ้นอยู่กับบิต SMOD ที่อยู่ในรีจิสเตอร์ PCON เป็นตัวเลือก

ถ้าบิต SMOD = 0 (เป็นค่าหลังจากรีเซต) อัตรารับส่งโหมด 2 = 1/64 เท่าของสัญญาณนาฬิกา

ถ้าบิต SMOD = 1 อัตรารับส่งโหมด 2 = 1/32 เท่าของสัญญาณนาฬิกา

- **อัตราการรับและส่งข้อมูลในโหมด 1 และ โหมด 3** จะกำหนดได้ตามต้องการ โดยใช้อัตราการเกิดโอเวอร์โฟลวของ Timer1 หรือ Timer2 เป็นตัวกำหนด อัตราการรับและการส่งข้อมูลสามารถกำหนดได้จาก Timer1 หรือ Timer2 หรือจากทั้ง 2 ตัว (ตัวหนึ่งสำหรับส่งอีกตัวหนึ่งสำหรับรับข้อมูล) ความเร็วในการรับและส่งข้อมูลอาจกำหนดไม่เท่ากันก็ได้

2.5.2.4 การรับส่งข้อมูลในโหมดต่างๆ

- การรับส่งข้อมูลในโหมด 0

การรับส่งข้อมูลในโหมด 0 เป็นการทำงานในลักษณะของ Shift Register ข้อมูลอนุกรมรับเข้ามาทางขา RXD และส่งข้อมูลออกทางขา TXD จะเป็นแบบ 8 บิต โดยบิตต่ำจะรับหรือส่งก่อน อัตราการรับหรือการส่งในโหมดนี้คือ 1/12 เท่าของสัญญาณนาฬิกา รูป แสดงไดอะแกรมของการรับส่งข้อมูลของพอร์ตอนุกรมในโหมด 0 ซึ่งมีไดอะแกรมประกอบอยู่ด้วย

การส่งข้อมูลจะเริ่มต้นหลังจากมีการกำหนดค่าให้กับรีจิสเตอร์ SBUF ซึ่งสัญญาณการส่งข้อมูลให้กับรีจิสเตอร์ SBUF (write to SBUF) ในช่วงเวลา S6P2 จะทำการโหลดค่า 1 ลงไปในตำแน่งของบิตที่ 9 ของ Shift Register และเป็นตัวเริ่มต้นการส่งข้อมูลของ Tx Conrol box การทำงานดังกล่าวจะใช้เวลา 1 cycle แล้วจึงเริ่มต้นทำการส่งข้อมูล ซึ่งมีสัญญาณ SEND แอคทีฟ สัญญาณ SEND จะควบคุมให้ข้อมูลใน Shift Register ถูกเลื่อนออกไปที่ขา P3.0 และไปควบคุมสัญญาณ SHIFT CLOCK ส่งออกไปที่ขา P3.1 สัญญาณที่ออกจาก SHIFT CLOCK จะเป็น 0 ในช่วงเวลา S3, S4, S5 ในทุกๆ Machine Cycle และเป็น 1 ในช่วงเวลา S6, S1, S2 สำหรับช่วงเวลา S6P2 ของทุกๆ Machine Cycle ที่มีสัญญาณ SEND แอคทีฟค่าอยู่ใน Shift Register ของการส่งจะถูกเลื่อนทางขวา 1 ตำแหน่ง

เมื่อข้อมูลถูกเลื่อนออกไปทางขวาจะทำให้ข้อมูลทางซ้ายเป็น 0 เมื่อข้อมูลบิตสูงสุดไปอยู่ในตำแหน่งเอาท์พุทของ Shift Register แล้วจะทำให้ค่า 1 ซึ่งได้โหลดเข้าไปในบิตที่ 9 ตอนเริ่มต้นอยู่ทางซ้ายมือของบิตที่ 8 (MSB) และข้อมูลบิตทางซ้ายมือต่อมาทั้งหมดเป็น 0 จากเงื่อนไขนี้ทำให้ Tx CONTROL BOX ทำการส่งข้อมูลตัวสุดท้ายออกไป แล้วจึงเลิกส่งสัญญาณ SEND และเซตบิต TI = 1 (อินเตอร์รัพต์การส่งข้อมูล) ซึ่งการทำงานทั้งสองจะเกิดขึ้นในช่วง S1P1 ของแมชชีนไซเกิลที่ 10 หลังจากการเขียนข้อมูลลงใน SBUF ด้วยสัญญาณ write to SBUF แล้ว

การรับข้อมูลจะเริ่มต้นโดยการเซตบิต REN = 1 และ RI = 1 (REN และ RI อยู่ในรีจิสเตอร์ SCON) โดยในช่วงเวลา S6P2 ของ แมชชีนไซเกิลต่อมา RX CONTROL UNIT จะเขียนข้อมูล 11111110 ลงไปใน shift register ด้านรับ และในเฟสต่อไปจะเริ่มต้นส่งสัญญาณ RECEIVE

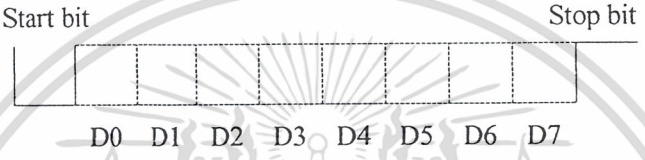
สัญญาณ RECIVE จะทำให้ SHIFT CLOCK ส่งออกไปที่ขา P3.1 ซึ่งสัญญาณ SHIFT CLOCK จะเปลี่ยนแปลงในช่วง S3P1 และ S6P1 ของทุกแมชชีนไซเกิล ในช่วง S2P6 ของทุกแมชชีนไซเกิลที่สัญญาณ RECEIVE แอคทีฟอยู่ จะทำให้ข้อมูลใน shift register เลื่อนซ้ายไป 1 ตำแหน่ง โดยที่ข้อมูลที่เข้ามาทางบิตขวามือ คือค่าที่อ่านจากขา P3.0 ในช่วงเวลา S5P2 ของแมชชีนไซเกิลเดียวกัน

เมื่อมีการเลื่อนข้อมูลใน Shift Registerไปจนกระทั่งบิตที่อยู่ทางขวามือสุดที่ได้กำหนดในตอนเริ่มต้นถูกเลื่อนไปอยู่ซ้ายมือสุด จะควบคุมให้ RX CONTROL BOX ทำการเลื่อนข้อมูลอีก 1 ครั้งเป็นครั้งสุดท้ายแล้วนำข้อมูลไปเก็บไว้ในรีจิสเตอร์ SBUF ซึ่งเป็นการสิ้นสุดการรับข้อมูล 1 ไบต์ การเริ่มต้นการรับข้อมูลที่เกิดขึ้นเมื่อมีการเขียนการควบคุมไปที่รีจิสเตอร์ SCON เพื่อเริ่มรับข้อมูลจะทำให้ RI ถูกรีเซตด้วย จนเวลาผ่านไปถึงในช่วง S1P1 ของแมชชีนไซเกิลที่ 10

สัญญาณ RECEIVE จะถูกเคลียร์ ซึ่งแสดงว่าได้รับข้อมูลเข้ามาครบ 8 บิตแล้ว และ แฟลก RI จะถูกเซต แฟลก RI นี้เป็นแฟลก อินเตอร์รัพต์ของการรับข้อมูลของพอร์ตอนุกรม

- การรับส่งข้อมูลในโหมด 1

การส่งข้อมูลจำนวน 10 บิตจะส่งออกไปที่ขา TXD และรับข้อมูลเข้ามาทางขา RXD ข้อมูล 10 บิตประกอบด้วย 1 Start bit (ค่าลอจิก 0) ข้อมูลจำนวน 8 บิต (LSB ส่งหรือรับก่อน) และ 1 Stop bit (ค่าลอจิก 1)



รูปที่ 2.20 สัญญาณของการส่งข้อมูลของโหมด 1

ในกรณีของการรับข้อมูล Stop bit จะถูกนำเข้าไปในบิต RB8 ที่อยู่ในรีจิสเตอร์ SCON ความเร็วในการรับส่งข้อมูลจะกำหนดโดยอัตราการเกิด Overflow ของ Timer1 หรือ Timer2 หรือกำหนดจากทั้ง 2 ตัว (ตัวหนึ่งสำหรับด้านส่งและอีกตัวหนึ่งสำหรับด้านรับ)

การส่งข้อมูลจะเริ่มจากการกำหนดค่าลงในรีจิสเตอร์ข้อมูล SBUF ด้านส่งจะมีสัญญาณ write to SBUF ควบคุมให้มีการใส่ค่า 1 ลงในตำแหน่งของบิตที่ 9 ของ Shift Register ด้านส่ง และเริ่มทำการส่งข้อมูลของ Tx CONTROL UNIT การส่งข้อมูลออกจะเริ่มต้นในเวลา S1P1 ของแมชชีนไซเกิลและบิตต่อไปตามจังหวะของตัวกำหนดอัตราส่งที่มาจาก Overflow ของ Timer1 หรือ Timer2 ที่หารด้วย 16

การส่งข้อมูลจะเริ่มต้นด้วยการส่งสัญญาณ SEND ให้แอคทีฟ เมื่อเริ่มต้นการส่งข้อมูลบิตแรกออกไปที่ TXD และบิตของข้อมูลจะถูกส่งตามออกไปโดยการทำงานของ Shift Register ซึ่งจะทำการเลื่อนข้อมูลออกไป 1 ตำแหน่งตามจังหวะของอัตราที่ส่งเข้ามายัง Tx CLOCK เมื่อบิตข้อมูลทั้งหมดถูกเลื่อนไปทางด้านขวา จะมี 0 เข้ามาแทนที่ทางด้านซ้าย จนกระทั่งบิตข้อมูลสูงสุด (MSB) ออกไปอยู่ที่เอาท์พุทของ Shift Register จะทำให้บิต 1 ที่ได้โหลดเข้าไปตอนเริ่มต้นในตำแหน่งที่ 9 ซึ่งอยู่ทางซ้ายมือของ MSB และมีบิตต่างๆที่อยู่ทางซ้ายของบิตที่ 9 เป็น 0 ทั้งหมด เมื่อเกิดเงื่อนไขดังกล่าวจะทำให้ TX CONTROL UNIT ทำการเลื่อนข้อมูลออกไปทางขวาอีก 1 ครั้ง เป็นครั้งสุดท้าย แล้วเลิกส่งสัญญาณ SEND และเซตแฟลก TI = 1 เวลาการส่งข้อมูลจะจบในช่วง CLOCK ลูกที่ 10 ของสัญญาณการส่งข้อมูล หลังจากมีสัญญาณ write to SBUF

การรับข้อมูลจะเริ่มต้นเมื่อสัญญาณที่ขา RXD เปลี่ยนจาก 1 เป็น 0 โดยซีพียู จะมีการตรวจสอบด้วยความเร็ว 16 เท่าของอัตราเร็วในการรับข้อมูล เมื่อซีพียูตรวจสอบได้ว่ามีการส่งข้อมูลเข้ามาจะทำการรีเซตวงจรหาร 16 ทันทีเพื่อเริ่มต้นใหม่ให้สัมพันธ์กับข้อมูลที่เข้ามาและค่า 1FFH จะถูกกำหนดให้กับ Shift Register ด้านรับ

- **การรับส่งข้อมูลในโหมด 2 และโหมด 3** การทำงานในโหมด 2 และ โหมด 3 จะมีบิตที่รับส่งข้อมูลทั้งหมด 11 บิตซึ่งประกอบไปด้วย 1 START BIT (ค่า 0) บิตข้อมูล 8 บิต (บิตต่ำก่อน) 1 บิตข้อมูลที่ 9 ที่กำหนดค่าได้และ 1 STOP BIT

รูปที่ 2.21 สัญญาณการส่งข้อมูลโหมด 2 และโหมด 3

การกำหนดข้อมูลบิตที่ 9 ของการส่งข้อมูลกำหนดโดยการเซตหรือเคลียร์บิต TB8 ที่อยู่ในรีจิสเตอร์ SCON สำหรับการรับข้อมูลบิตที่ 9 จะถูกนำไปเก็บใน RB8 ซึ่งอยู่ในรีจิสเตอร์ SCON เช่นกัน อัตราการรับส่งข้อมูลในโหมด 2 สามารถโปรแกรมเพื่อเลือก 1/32 หรือ 1/64 เท่าของสัญญาณนาฬิกาได้ ส่วนในโหมด 3 สามารถกำหนดอัตราการรับและส่งข้อมูลได้ตามต้องการโดยใช้ Timer1 หรือ Timer2 ซึ่งใช้ TCLK และ RCLK ที่อยู่ใน T2CON เป็นตัวกำหนดอัตราการรับส่งข้อมูล

การส่งข้อมูลเริ่มจากคำสั่งกำหนดค่าให้กับรีจิสเตอร์ SBUF ซึ่งจะทำให้สัญญาณ write to SBUF ควบคุมให้บิตข้อมูลใน TB8 ไปกำหนดค่าให้กับบิตที่ 9 ของ Shift Register และควบคุมให้ TX CONTROL UNIT ทำการส่งข้อมูลออกไปในช่วงเวลา S1P1 ของแมชชีนไซเกิลที่ต่อจากการเริ่มทำงานใหม่ของวงจรหาร 16 (ทำให้เกิดการตรงกันของวงจรหาร 16 ซึ่งไม่ขึ้นกับเวลาของสัญญาณ write to SBUF)

การเริ่มต้นส่งข้อมูลเริ่มต้นโดยการทำให้สัญญาณ SEND แอคทีฟ ซึ่งจะส่งค่า START BIT ออกไปที่ TDX และตามด้วยข้อมูลบิตต่ำไปบิตสูงตาทจังหวะของบิตกำหนดการส่งข้อมูลตามด้วยบิตที่ 9 ที่อยู่ใน 1TB8 และจบด้วย STOP BIT (ค่า 1) หลังจากที่บิตต่างๆ เลื่อนไปทางขวาแล้วจะมี 0 เข้ามาทางซ้าย ดังนั้นเมื่อค่าของ TB8 อยู่ในตำแหน่งเอาท์พุทของ Shift Register แล้วถัดมาทางซ้ายจะเป็น STOP BIT และถัดมาจะมีค่าเป็น 0 ทั้งหมด ซึ่งเงื่อนไขนี้จะเป็นตัวกำหนด

ให้ Shift Register ทำการเลื่อนไปทางขวาอีกครั้งหนึ่งและเป็นครั้งสุดท้ายแล้วจึงเลิกแอคทีฟสัญญาณ SEND และเซตแฟลก TI ซึ่งการทำงานทั้งหมดจะจบสิ้นภายหลังจากมีสัญญาณ write to SBUF เกิดขึ้นแล้วมีสัญญาณคล็อกเข้ามาที่ TXCLK จำนวน 11 ลูก

การรับข้อมูล เริ่มต้นจากการเปลี่ยนแปลงของสัญญาณที่ขา RXD จาก 1ไป 0 (START BIT) ซึ่งวงจรด้านรับจะมีการตรวจสอบด้วยความเร็ว 16 เท่าของอัตราการรับข้อมูลเมื่อตรวจพบการเปลี่ยนแปลงสัญญาณจะทำให้เคาท์เตอร์ของวงจรหาร 16 ถูกรัเซต เพื่อเริ่มใหม่ และมีการกำหนดค่า 1FFH ให้กับ Shift Register ด้านรับ และในช่วงของการนับ 7, 8 และ 9 ของเคาท์เตอร์วงจรหาร 16 จะมีการตรวจสอบสัญญาณที่ขา RXD อีกซึ่งจะใช้ค่าที่อ่านได้จาก 2 ใน 3 ครั้งเป็นข้อมูล หากค่าที่อ่านได้ใน START BIT ไม่เป็น 0 วงจรรับข้อมูลจะทำการรีเซตและรอตรวจสอบบิตที่เริ่มต้นเปลี่ยนจาก 1 เป็น 0 ใหม่ หาก START BIT ถูกต้อง จะทำการเลื่อนข้อมูลเข้าไปเก็บใน Shift Register โดยบิตของข้อมูล เข้ามาทางขวามือและบิตที่มีค่า 1 ซึ่งกำหนดในตอนเริ่มต้นถูกเลื่อนออกไปทางซ้ายมือ เมื่อ Start Bit ถูกเลื่อนออกไปทางซ้ายมือสุด ในโหมด 2 และโหมด 3 Shift Register มีขนาด 9 บิต) จะเป็นตัวกำหนดให้ RX CONTROL BOX ทำการเลื่อนข้อมูลใน Shift Register อีกครั้งเป็นครั้งสุดท้าย แล้วกำหนดค่าใน Shift Register ลงใน SBUF และ RB8 และเซตแฟลก RI ซึ่งการกำหนดค่าลงใน SBUF และ RB8 และการเซต R1 จะเกิดขึ้นเมื่อมีเงื่อนไข 2 อย่างคือ

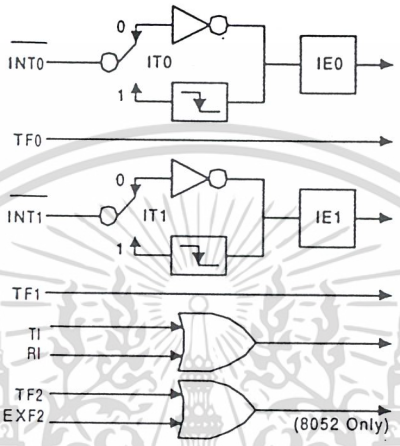
1) RI = 0 (จะต้องเคลียร์ RI หลังจากทำโปรแกรมอ่านข้อมูลไปแล้ว) และ
2) SM2 = 0 หรือข้อมูลบิตที่ 9 เป็น 1

ถ้าหากเงื่อนไขทั้ง 2 ไม่เป็นจริง เฟรมข้อมูลที่รับเข้ามาจะถูกยกเลิกและ RI จะไม่ถูกเซต ถ้าหากเงื่อนไขทั้ง 2 เป็นจริง ข้อมูลบิตที่ 9 จะถูกนำไปเก็บไว้ใน RB8 และข้อมูล 8 บิตแรกจะถูกนำไปเก็บไว้ใน SBUF และในช่วงเวลาของสัญญาณ CLOCK ลูกถัดไป วงจรด้านรับจะเริ่มตรวจสอบ START BIT เข้ามาต่อไป

2.5.2.5 การใช้งานอินเตอร์รัพท์

การติดต่อสื่อสารกันระหว่างไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอกสามารถทำได้ 2 ลักษณะคือ ใช้วิธีการโพลลิ่ง (Polling) ซึ่งไมโครคอนโทรลเลอร์ต้องคอยตรวจสอบอุปกรณ์ภายนอกอยู่ตลอดเวลาว่ามีข้อมูลที่ต้องการติดต่อกับไมโครคอนโทรเลอร์หรือไม่ ซึ่งวิธีนี้จะทำให้ไมโครคอนโทรลเลอร์ต้องเสียเวลาไปกับการตรวจสอบนานมาก หากมีอุปกรณ์จำนวนมากต่ออยู่กับไมโครคอนโทรลเลอร์ การติดต่อแบบที่สองเป็นการใช้การออนเตอร์รัพท์ซึ่งเป็นวิธีการที่ไมโครคอนโทรลเลอร์ไม่ต้องคอยตรวจสอบการอุปกรณ์ภายนอกอยู่ตลอดเวลา แต่จะให้อุปกรณ์ภาย

นอกส่งสัญญาณอนเตอร์รัพท์เข้ามาให้ไมโครคอนโทรลเลอร์ เมื่ออุปกรณ์ภายนอกต้องการติดต่อกับไมโครคอนโทรลเลอร์ เมื่อไมโครคอนโทรลเลอร์ได้รับสัญญาณอินเตอร์รัพต์แล้วจึงทำการส่งข้อมูลกับอุปกรณ์ภายนอกนั้นๆ ต่อไป



รูปที่ 2.22 แหล่งกำเนิดสัญญาณอินเตอร์รัพต์

แหล่งกำเนิดสัญญาณอินเตอร์รัพต์ของ 8051 มี 5 แหล่ง และ 8052 มี 6 แหล่ง สัญญาณร้องขออินเตอร์รัพต์ต่างๆ แสดงดังรูปที่

แหล่งกำเนิดสัญญาณอินเตอร์รัพต์แต่ละแหล่งจะมาอุปกรณีต่าง ๆ ดังนี้

สัญญาณร้องขออินเตอร์รัพต์	ความหมาย
INT0\	สัญญานร้องขออินเตอร์รัพต์จากขา P3.2
INT1\	สัญญาณร้องขออินเตอร์รัพต์จากขา P3.3
TF0	สัญญาณร้องขอการอินเตอร์รัพต์จาก Timer0 ที่อยู่ในตัวไมโครโปรเซสเซอร์
TF1	สัญญาณร้องขอการอินเตอร์รัพต์จาก Timer1 ที่อยู่ในตัวไมโครคอนโทรลเลอร์
RI,TI	สัญญาณร้องขอการอินเตอร์รัพต์จากพอร์ตอนุกรมที่อยู่ในตัวไมโครคอนโทรลเลอร์
TF2,EXF2	สัญญาณร้องขอการอินเตอร์รัพต์จาก Timer2

- การควบคุมอินเตอร์รัพต์

สัญญาณต่างๆ ของการร้องขออินเตอร์รัพต์ทั้งหมด เราสามารถเซตหรือเคลียร์ได้ด้วยซอฟต์แวร์ ดังนั้นการร้องขออินเตอร์รัพต์จึงสามารถสร้างหรือยกเลิกได้ด้วยซอฟต์แวร์ แหล่งกำเนิดสัญญาณอินเตอร์รัพต์ทั้งหมด เราสามารถควบคุมให้ทำการร้องขออินเตอร์รัพต์หรือไม่ก็ได้ โดยการเซตหรือเคลียร์บิตต่างๆ ที่อยู่ในรีจิสเตอร์ IE (Interupt Enable) ภายในรีจิสเตอร์ IE มีบิต EA ที่ทำหน้าที่ควบคุมให้ไมโครคอนโทรลเลอร์ตอบรับการร้องขออินเตอร์รัพต์ หรือไม่ตอบรับทั้งหมด หากเรากำหนดให้บิต EA เป็น 0 ไมโครคอนโทรลเลอร์จะไม่ตอบรับการร้องขออินเตอร์รัพต์ทั้งหมด

IE : INTERRRUPT ENABLE REGISTER (Bit addressable)

IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA IE.7 (Disable all interrupt) หากกำหนดให้บิตนี้มีค่าเป็น 0 จะไม่มีการตอบรับการเตอร์รัพต์ทั้งหมดหากต้องการให้มีการตอบรับอินเตอร์รัพต์ จะต้องเซตให้ EA เป็น 1 และเซตบิตควบคุมแต่ละอินเตอร์รัพต์ที่ต้องการใช้ในแต่ละอินเตอร์รัพต์ให้เป็น 1 ด้วย โดยคำสั่งเซต หรือเคลียร์บิต

\- IE.6 ไม่ใช้งาน

ET2 IE.5 ควบคุมให้ Timer2 หรือ Capture ส่งอินเตอร์รัพต์หรือไม่ (1 ส่งอินเตอร์รัพต์)

ES IE.4 ควบคุมให้พอร์ตอนุกรมส่งอินเตอร์รัพต์หรือไม่ (1 ส่งอินเตอร์รัพต์

ET1 IE.3 ควบคุมให้ Timer1 ส่งอินเตอร์รัพต์เมื่อถึงค่าที่ตั้งไว้หรือไม่ (1 ส่งอินเตอร์รัพต์)

EX1 IE.2 ควบคุมให้รับอินเตอร์รัพต์จากภายนอก INT1\ หรือไม่ (1 ตอบรับอินเตอร์รัพต์)

ET0 IE.1 ควบคุมให้ Timer0 ส่งอินเตอร์รัพต์เมื่อถึงค่าที่ตั้งไว้หรือไม่ (1 ส่งอินเตอร์รัพต์)

EX0 IE.0 ควบคุมให้ตอบรับอินเตอร์รัพต์จากภายนอก INT0\ หรือไม่ (1 ส่งอินเตอร์รัพต์)

หากต้องการใช้สัญญาณอินเตอร์รัพต์จากตำแหน่งเกิดใด จะต้องเซตบิตควบคุมให้เป็น 1 และต้องเซตให้ EA เป็น 1 ด้วย

- การจัดลำดับความสำคัญของการอินเตอร์รัพต์

การจัดลำดับความสำคัญของการอินเตอร์รัพต์ของไมโครคอนโทรลเลอร์ จัดได้ 2 ระดับแตกต่างกันโดยการเซตค่าในรีจิสเตอร์ควบคุมการจัดลำดับความสำตัญของการอินเตอร์รัพต์ (IP : Interrupt Piority Register) ในกรณีที่ไม่มีการจัดลำดับความสำคัญของการอินเตอร์รัพท์หรือจัดให้

มีความสำคัญในระดับเดียวกัน ไมโครคอนโทรลเลอร์จะจัดให้มีความสำคัญของสัญญาณอินเตอร์รัพต์ที่อยู่ในระดับเดียวกัน (เพื่อแก้ปัญหาการขออินเตอร์รัพต์ในระดับเดียวกันที่เกิดขึ้นพร้อมกัน) ตามลำดับความสำคัญจากสูงไปต่ำดังแสดงในตารางที่ 2.5

IE0	**External Interrupt 0**
TF0	Timer 0
IE1	External Interrupt 1
TF1	Timer 1
RI หรือ TI	Serial Port
TF2 หรือ EXF2	Timer2

ตารางที่ 2.5 ลำดับความสำคัญของการอินเตอร์รัพต์

การกำหนดความสำคัญของการอินเตอร์รัพต์กำหนดได้จากบิตต่างๆ ของรีจิสเตอร์ IP โดยบิตที่มีค่าเป็น 1 จะมีลำดับความสำคัญสูงกว่า

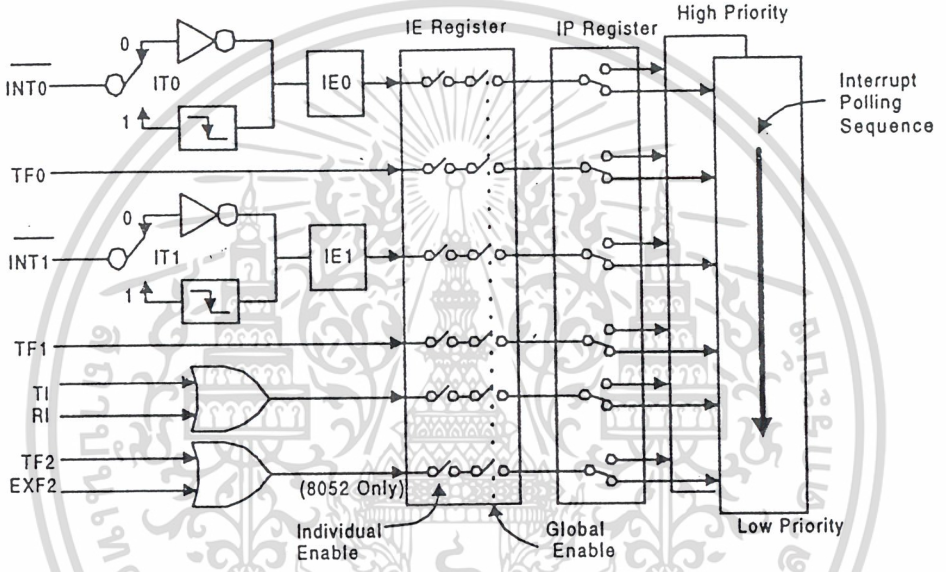
- \- IP.7 ไม่ใช้
- \- IP.6 ไม่ใช้
- PT2 IP.5 กำหนดความสำคัญของ Timer 2
- PS IP.4 กำหนดลำดับความสำคัญของพอร์ตอนุกรม
- PT1 IP.3 กำหนดลำดับความสำคัญของ Timer 1
- PX1 IP.2 กำหนดลำดับความสำคัญของสัญญาณ INT1\ (External Interrupt 1)
- PT0 IP.1 กำหนดลำดับความสำคัญของ Timer 0
- PX0 IP.0 กำหนดลำดับความสำคัญของสัญญาณ INT0\

บิตใดมีค่าเป็น 1 แสดงว่ามีความสำคัญสูง

เมื่อมีการกำหนดลำดับความสำคัญของการอินเตอร์รัพต์แตกต่างกันแล้ว หากมีการร้องขออินเตอร์รัพต์เกิดขึ้นพร้อมกันจากแหล่งกำเนิด 2 แหล่งกำเนิดที่มีความสำคัญแตกต่างกัน ไมโครคอนโทรลเลอร์จะตอบรับการร้องขออินเตอร์รัพต์จากแหล่งกำเนิดสัญญาณอินเตอร์รัพต์ที่มีระดับความสำคัญสูงกว่า สัญญาณการร้องขออินเทอร์รัพต์ที่มีความสำคัญสูงกว่าสามารถอินเทอร์

รัพต์ซ้อนในขณะที่ไมโครคอนโทรลเลอร์กำลังทำโปรแกรมตอบสนองการอินเตอร์รัพต์ของสัญญาณที่มีความสำคัญต่ำกว่าได้

วงจรควบคุมการอินเตอร์รัพต์ของ MCS-51 ทำโดยการควบคุมการตัดต่อสวิตช์ต่างๆ ดังแสดงในรูป ซึ่งควบคุมการตัดต่อสวิตช์ด้วยซอฟต์แวร์จากการเซตหรือเคลียร์บิตของรีจิสเตอร์ IE และ IP



รูปที่ 2.23 ระบบควบคุมการอินเตอร์รัพต์

หากต้องการให้ไมโครคอนโทรลเลอร์ตอบรับการตอบสนองอินเตอร์รัพต์ จะต้องมีการกำหนดการทำงานตามขั้นตอนต่อไปนี้

1.เซตบิต EA (Enable all) ในรีจิสเตอร์ IE.7 ให้เป็น 1

2.เซตบิตควบคุมอินเตอร์รัพต์ของแต่ละแหล่งกำเนิดที่ต้องการใช้งานอินเตอร์รัพต์ในรีจิสเตอร์ IE ให้มีค่าเป็น 1

3.กำหนดความสำคัญของการอินเตอร์รัพต์ (หากต้องการ) โดยกำหนดในรัจิสเตอร์ IP

4.เขียนโปรแกรมการตอบสนองการร้องขออินเตอร์รัพต์ของแต่ละแหล่งกำเนิด ลงในตำแหน่งต่างๆ ตาม Vector Address ในตารางที่ 2.5

Interrupt Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI&TI	0023H
TF2 & EXF2	002BH

ตารางที่ 2.6 Vector Adderss ของการตอบสนองอินเตอร์รัพต์

จากตารางที่ 2.6 แสดงค่าตำแหน่งต่างๆ ของโปรแกรมตอบสนองการอินเตอร์รัพต์ของแต่ละแหล่งกำเนิดซึ่งจะต้องนำตำแหน่งเริ่มต้นของโปรแกรมตอบสนองการอินเตอร์รัพต์บรรจุไว้ หากมีการใช้อินเทอร์รัพต์ทั้งหมดจะพบว่าความยาวของโปรแกรมตอบสนองอินเตอร์รัพต์ของแต่ละโปรแกรมจะมีความยาวเพียง 8 ไบต์เท่านั้น หากโปรแกรมมีความยาวมากกว่านี้ ต้องนำไปเขียนโปรแกรมย่อยไว้ภายนอก แล้วใช้วิธีการเรียกโปรแกรมย่อยมาทำงาน

- การอินเตอร์รัพต์จากพอร์ตอนุกรม

โครงงานชิ้นนี้มีการติดต่อกับผู้ใช้ผ่านคอมพิวเตอร์ผ่าน Serial Port โดยใช้โปรแกรม Visual Basic เป็น Interface Programming ดังนั้นจึงมีความจำเป็นต้องใช้สัญญาณอินเตอร์รัพต์จาก Serial Port โดยมีรายละเอียดดังนี้

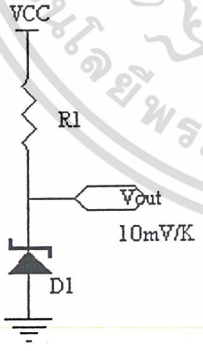
แหล่งกำเนิดสัญญาณอินเตอร์รัพต์จากพอร์ตอนุกรมมีสัญญาณอินเตอร์รัพต์เกิดจาก 2 แหล่งตือจาก RI (ด้านรับ) และ TI (ด้านส่ง) เป็นแหล่งกำเนิดที่มาจากบัฟเฟอร์ทางด้านรับและส่งข้อมูลของพอร์ตอนุกรมที่อยู่ในไมโครคอรโทรลเลอร์ สัญญาณทั้งสองจะถูกนำมา OR กันเพื่อรวมเป็นสัญญาณเดียว ทำให้มีสัญญาณร้องขออินเตอร์รัพต์ที่เกิดขึ้นได้ทั้งในกรณีของการรับและการส่งข้อมูลของพอร์ตอนุกรม โปรแกรมย่อยของการตอบสนองอินเตอร์รัพต์ของพอร์ตอนุกรมอยู่ที่ 23H ตำแหน่งเดียว ดังนั้นในส่วนของโปรแกรมตอบสนองการอินเตอร์รัพต์ของพอร์ตอนุกรม จะต้องมีการตรวจสอบก่อนว่าการขออินเตอร์รัพต์เกิดขึ้นจาก RI หรือ TI โดยการใช้คำสั่งการตรวจสอบบิต RI หรือ TI ที่อยู่ในรีจิสเตอร์ SCON เมื่อทราบที่มาของการร้องขออินเตอร์รัพต์แล้วต้องทำการเคลียร์บิต RI หรือ TI ด้วยซอฟต์แวร์ก่อนที่จะจบโปรแกรมการตอบสนองการอินเตอร์รัพต์ ตัวอย่างการใช้งานจะอยู่ในเรื่องการรับส่งข้อมูลของพอร์ตอนุกรม

บทที่ 3
หลักการออกแบบฮาร์ดแวร์

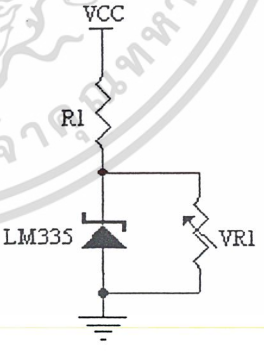
3.1 การออกแบบวงจรตรวจวัดอุณหภูมิ

3.1.1 วงจรตรวจวัดอุณหภูมิด้วย LM335

เซ็นเซอร์ตรวจวัดอุณหภูมิ LM335 มีลักษณะทำงานคล้าย ZENER DIODE โดยแรงดันพังทลายหรือแรงดันเอาท์พุทจะแปรผันตามอุณหภูมิสัมบูรณ์ในหน่วยองศาเคลวินโดยมีค่าเท่ากับ 10 mV / 1K และคุณสมบัติที่สำคัญของ LM335 คือให้แรงดันเอาท์พุทเป็นเชิงเส้น ซึ่งถ้านำเอาท์พุทมาพลอตกราฟระหว่างแรงดันกับอุณหภูมิตลอดย่านการทำงานของ LM335 เส้นกราฟไปตัดแกนที่อุณหภูมิศูนย์องศาสัมบูรณ์ ดังนั้นจึงต้องมีการปรับแต่งอุณหภูมิให้ถูกต้องโดยการต่อความต้านทานปรับค่าได้คร่อม LM335 ดังรูป



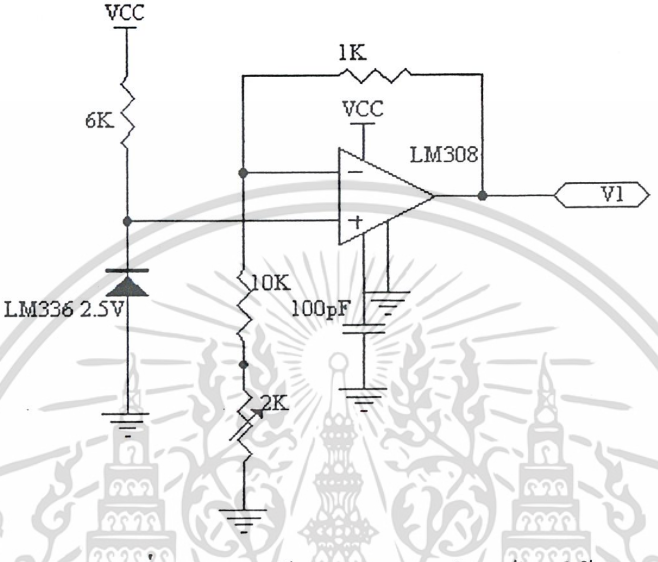
รูปที่ 3.1 วงจรพื้นฐานการใช้งานของLM35



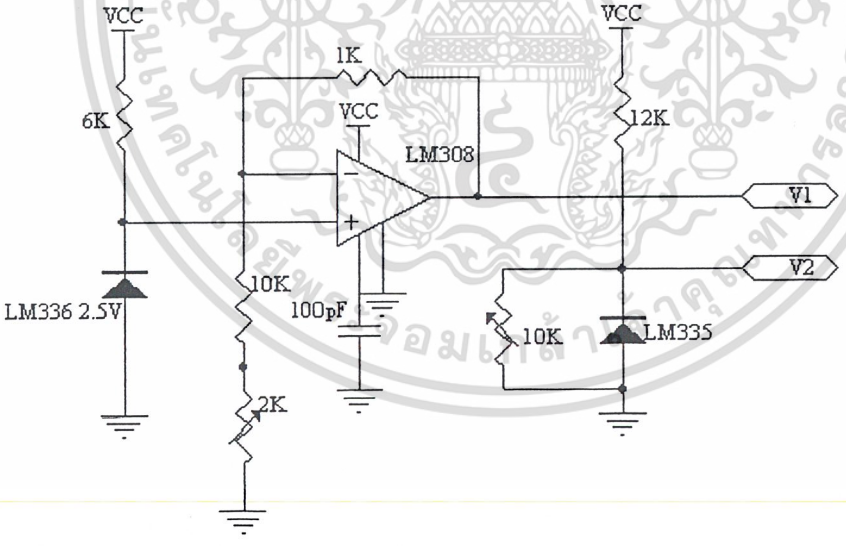
รูปที่ 3.2 วงจรปรับแต่งความถูกโดยใช้ความต้านทานปรับค่าได้

ในโครงงานนี้ต้องทำการควบคุมการแปรผันตามอุณหภูมิของแรงดันเอาท์พุทนั้นให้แปรผันตามอุณหภูมิในหน่วยองศาเซลเซียสแต่เนื่องจาก LM335 นั้นจะมีแรงดันแปรผันตามอุณหภูมิเป็นเคลวิน และอุณหภูมิในหน่วยองศาเคลวินต่างจากอุณหภูมิในหน่วยองศาเซลเซียสอยู่ 273องศา ดังนั้นเพื่อการแปลงหน่วยอุณหภูมิออกมาในหน่วยองศาเซลเซียสที่คุ้นเคยกับคนไทยนั้นจึงต้องทำ

การสร้างแรงดันขนาด 2.73 โวลต์คงที่ขึ้นมาโดยใช้ IC Op-Amp เบอร์ LM308 สร้างแรงดันขนาด 2.73 โวลต์ เพื่อนำมาหาผลต่างกับแรงดันเอาท์พุทจาก LM335 ที่แปรผันตามอุณหภูมิในหน่วยองศาเคลวิน ผลที่ได้ออกมาคือแรงดันที่แปรผันตามอุณหภูมิองศาเซลเซียส



รูปที่ 3.3 วงจรสร้างแรงดัน 2.73โวลต์โดยใช้ LM308

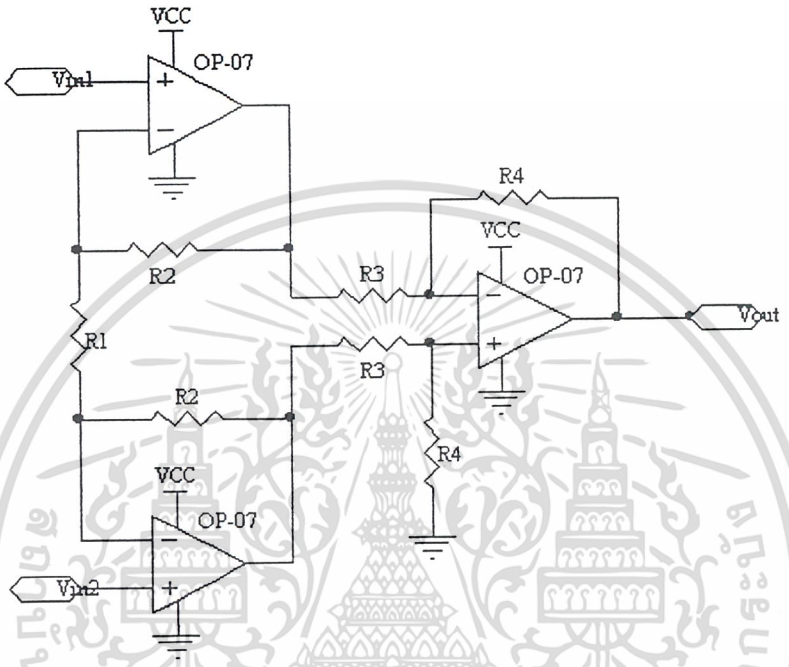


รูปที่ 3.4 วงจรตรวจวัดอุณหภูมิที่แปรผันตามอุณหภูมิในหน่วยองศาเซลเซียส

ดังนั้นก่อนทำการวัดอุณหภูมิจึงต้องปรับค่าความต้านทานเพื่อให้แรงดันเอาท์พุทที่ออกมาแปรผันตามอุณหภูมิแบบเชิงเส้นเพื่อความถูกต้องแม้นยำ ตัวอย่างเช่นที่อุณหภูมิห้อง 30 องศาเซลเซียส เราจึงต้องทำการปรับค่าความต้านทานปรับค่าได้จนได้ค่าแรงดันเอาท์พุทเป็น3.03โวลต์ ทำให้การวัดอุณหภูมิตลอดทั้งย่านถูกต้องและแม้นยำ

3.1.2 การออกแบบวงจร Instrumentation Ampliflier

- **ทฤษฎีของวงจร Instrumentation Amplifier** เป็นวงจรขยายที่มีคุณสมบัติดีกว่า Difference amplifier คือ มีค่าอินพุทอิมพีแดนซ์ที่สูงขึ้น และสามารถปรับค่าอัตราขยายได้ง่ายขึ้น



รูปที่ 3.5 วงจร Instrumentation amplifier

โดยค่าอัตราขยายของวงจรมีค่า

$$V_o / (V_{in2} - V_{in1}) = (R4/R3)(1+(2R2/R1)) \quad \text{.....(1)}$$

-หลักการออกแบบ เมื่อเราได้แรงดันเอาท์พุทออกที่แปรผันตามอุณหภูมิในหน่วยองศาเซลเซียสต้องนำมาขยายเนื่องจากเอาท์พุทที่ได้มีขนาดเล็กเกินไปไม่สามารถนำไปใช้ในวงจร A/D ต่อไปได้ เราเลือกใช้วงจร Instrumentation Ampliflier ในการทำการขยายเพราะเป็นวงจรขยายที่มีประสิทธิภาพสูงมีอัตราขยายที่ค่อนข้างคงที่ และเนื่องจากในวงจร A/D เราทำการป้อนไฟเลี้ยงขนาด 5 โวลต์เราจึงต้องออกแบบอัตราขยายของวงจรเป็น 10 เท่า ทำให้ได้ค่าเอาท์พุทที่มีค่าตั้งแต่ 0 – 5 โวลต์เพื่อที่จะสามารถวัดอุณหภูมิได้ตั้งแต่ค่า 0 ถึง 50 องศาเซลเซียส

ดังนั้นจากสมการที่ (1) แทนค่าอัตราขยายเท่ากับ 10 เท่า ได้ค่า R1, R2, R3, R4 ได้ดังนี้

$R2 = 2R1$, $R4 = 2R3$ แทนค่า $R1 = R3 = 10K$ ได้ค่า $R2 = R4 = 20K$

3.2 การออกแบบในส่วนควบคุม

3.2.1 การออกแบบวงจร Analog to Digital Conversion

การที่เราจะนำอุณหภูมิที่วัดได้ในรูปของแรงดันไปใช้งานร่วมกับไมโครคอนโทรลเลอร์นั้น ต้องทำการแปลงค่าแรงดันในรูปของสัญญาณอนาลอกให้กลายเป็นแรงดันในรูปสัญญาณดิจิตอล เพื่อที่จะสามารถนำไปใช้งานในการวิเคราะห์หรือเปรียบเทียบกันอุณหภูมิอ้างอิงได้

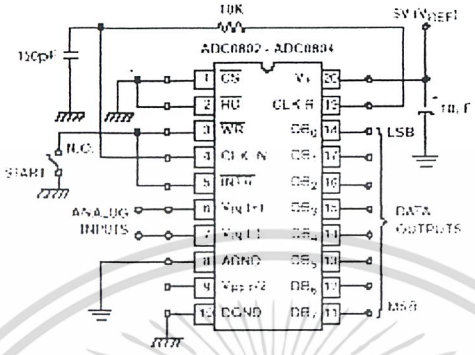
- **ทฤษฎีการ Sampling** การแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล ต้องใช้เวลาช่วงหนึ่งในการจัดการ (Aperture time) ซึ่งขึ้นอยู่กับหลายปัจจัย เช่น ความละเอียดของการแปลงสัญญาณ เทคนิคของการแปลงสัญญาณและความเร็วในการแปลงสัญญาณของอุปกรณ์อื่น

วงจร Sample and hold จะทำการสุ่มสัญญาณอินพุท และนำสัญญาณนั้นมาเก็บไว้ในช่วงเวลาหนึ่งได้ ซึ่งส่วนใหญ่จะใช้การประจุแรงดันไว้ในตัวเก็บประจุแรงดันไว้ในตัวเก็บประจุที่รั่วไหลต่ำ โดยสัญญาณอนาลอกจะถูกสุ่มเป็นช่วงๆการสุ่มจะเป็นการตัดต่อสัญญาณอนาลอกในช่วงเวลาสั้นๆด้วยสวิตช์ที่ทำงานด้วยความเร็วสูง ผลที่ได้จะเป็นสัญญาณที่มอดูเลทระหว่างสัญญาณพัลส์กับสัญญาณอนาลอกและสัญญาณอนาลอกคงค่า(Hold)ไว้จนกว่าค่าสัญญาณใหม่จะถูกสุ่มเข้ามา โดยอัตราการสุ่มจะต้องเหมาะสมที่จะไม่ทำให้ข้อมูลเสียหายเมื่อสัญญาณนั้นถูกดีมอดูเลทกลับตามเดิม นั้นคือสัญญาณอนาลอกก่อนทำการมอดูเลทมีความถี่ f_C อัตราของการsampling:f_C จะต้องมีค่าน้อยกว่า $2f_C$

- **หลักการออกแบบ** เราเลือกใช้วงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล ขนาด 8 บิท ซึ่งเราจะได้ความละเอียดสูงสุด 5 / 256 = 0.01953V ซึ่งการนำอุณหภูมิที่ทำการวัดได้ไปใช้ในการเปรียบเทียบ ไม่ต้องอาศัยความละเอียดมากนักเพราะเราแค่ทำการเปรียบเทียบว่าอุณหภูมิที่เราทำการวัดได้มีค่ามากหรือน้อยกว่าอุณหภูมิที่เราทำการกำหนด ดังนั้นการใช้ A/D ขนาด 8 บิท จึงเพียงพอต่อการใช้งาน

ในโครงงานนี้เลือกใช้วงจร Analog to Digital Conversion ที่ต่อแบบ Free running โดยการต่อวงจร A/D แบบนี้เมื่ออุณหภูมิเกิดการเปลี่ยนแปลงทำให้แรงดันอินพุทของวงจร A/D เปลี่ยนวงจร A/D นี้ก็จะทำการแปลงค่าของสัญญาณอนาลอกเป็นสัญญาณดิจดตอลทันที โดยไม่ต้องรอสัญญาณจากไมโครโปรเซสเซอร์ โดยเราจะทำการต่อขา Read ลงกราวด์ และทำการเชื่อมขา WR และขา INTR เข้าด้วยกัน เมื่อแรงดัน อินพุทเกิดการเปลี่ยนแปลงสัญญาณจากขา INTR จะถูกส่งออกมาเมื่อขา WR ได้รับสัญญาณ ดังกล่าวก็จะทำการแปลงค่าจาก Analog เป็น Digital ทันที แต่ส่วนประกอบที่สำคัญอีกส่วนก็คือ ขา WR และขา INTR นั้นจะต้องถูกเชื่อมต่อกับ Switch ซึ่งทุกครั้งที่เราเริ่มต้นทำการแปลงค่าจาก Analog เป็น Digital เราต้องทำการ Trigg Switch

ลงกราวด์เพื่อเป็นสัญญาณบอกว่าจะเริ่มทำการเปลี่ยนแปลงค่าแล้ว ซึ่ง Switch ดังกล่าวนี้เราสามารถใช้วงจร Relay ช่วยได้ดังที่จะได้กล่าวต่อไป



รูปที่ 3.6 แสดงการต่อวงจร Analog to Digital Converter แบบ Free running

3.2.2 ออกแบบส่วนไมโครคอนโทรลเลอร์

ในโครงงานนี้ใช้บอร์ดสำเร็จรูปของศิลารุ่น V-S82 ซึ่งประกอบด้วยส่วนต่างๆทางฮาร์ดดังนี้

1. **Microcontroller** เบอร์ AT89S8252 มีFlash Memory ขนาด 8 Kbyte, EEPROM ขนาด 2 Kbyte และมี Internal Ram 256 Byte

2. **Real time clock** DS1307 เป็นชิพนาฬิกา เพื่ออ่าน-เขียน ค่าของเวลา-วัน-เดือน-ปีพร้อม RAM 56 Byte พร้อม RAM ขนาด 56 Byte สำหรับใช้งานอิสระ

3. **RESET**ใช้ DS1833 เป็น POWER UP/DOWN RESET ให้กับระบบ

4. **Serial** RS232 ใช้ชิพ DS275, RS485 ใช้ชิพ 75176 ควบคุมทิศทางด้วย T0

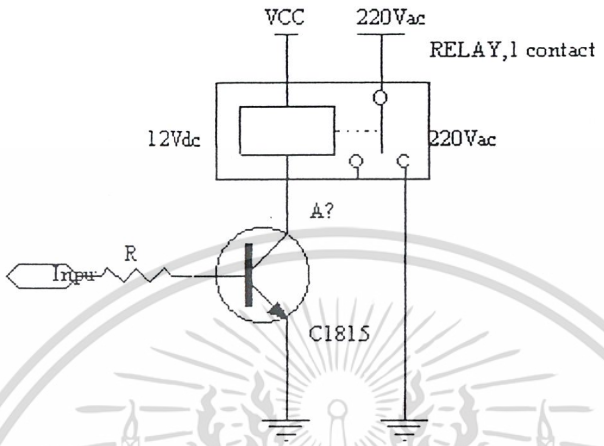
5. **USER PORT** สำหรับใช้เป็น I/O ต่างๆ

- LCD PORT เป็นส่วนแสดงผลข้อมูล
- 8255 PORT สำหรับใช้งานในส่วน I/O ของระบบ โดยไม่ได้ต่อใช้งานในลักษณะ Memmory Map แต่จะต่ออยู่ในรูป I/O โดย DATA=PORT0, /WR=P3.6, /RD=P3.7, A0=P2.0, A1=P2.1 Address port A=0000H, port B=0100H, port C=0200H, port Control=0300H

3.3 <u>การออกแบบวงจรควบคุมสวิตช์</u>

การออกแบบวงจรที่ใช้ Relay แทน Switch นั้นใช้วิธีการออกแบบดังรูป โดยจะมีการควบคุมการทำงานของ Relay ด้วยไมโครโปรเซสเซอร์ โดยปรกติจะมีการป้อนอินพุทให้กับทรานซิสเตอร์ขนาด 5 โวลต์ จากไมโครโปรเซสเซอร์ส่งผลกับสถานะของทรานซิสเตอร์ จะกลาย

เป็น On เนื่องจากค่า Vbe ของทรานซิสเตอร์มีค่ามากกว่า 0.7 โวลต์ ทำให้มีกระแสไหลผ่านขดลวดของ Relay



รูปที่ 3.7 วงจรควบคุมสวิตช์ด้วย RELAY

- หลักการออกแบบ จ่ายไฟหรือการปิดเปิดสวิตช์ด้วย RELAY โดย RELAY ทำงานที่แรงดัน 12 Vdc และที่กระแส 33.3 mA ใช้ทรานซิสเตอร์ในการขับกระแสให้ RELAY ทำงาน เนื่องจากขดลวดของ RELAY ทำงานที่แรงดัน 12 Vdc และ Vcc มีค่าเท่ากับ 15 V ดังนั้นแรงดันอิ่มตัวของทรานซิสเตอร์ที่ใช้ต้องไม่เกิน 15-12=3V ทรานซิสเตอร์2SC1815 มีค่าแรงดันอิ่มตัว Vce(sat)= 0.25 V ดังนั้นจึงใช้ได้ และ RELAY ทำงานที่กระแส 33.3 mA เพราะฉะนั้นกระแสคอลเลคเตอร์ (Ic) ของทรานซิสเตอร์ต้องมีค่ามากกว่า 33.3 mA ดังนั้นต้องใช้ค่า Rbias เพื่อให้ได้กระแสคอลเลคเตอร์ตามที่ต้องการ ซึ่งคำนวณได้ดังนี้

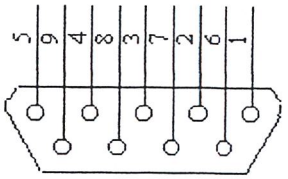
$$Ib = (Vin - Vbe) / Rbias$$

$$Ic = \beta Ib$$

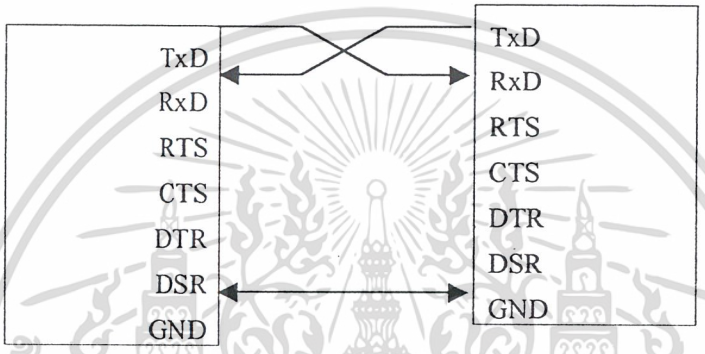
ดังนั้นจะคำนวณค่า Rbias

3.4 ออกแบบส่วนเชื่อมต่อกับผู้ใช้ (Interface)

ส่วนเชื่อมต่อสามารถรับอินพุทและแสดงผลทางคอมพิวเตอร์ซึ่งเขียนโดยโปรแกรมภาษา Visual Basic6โดยส่งข้อมูลเชื่อมต่อกับไมโครคอนโทรลเลอร์ ผ่านทางพอร์ตอนุกรมของ Computer โดยใช้มาตรฐานพอร์ตอนุกรมแบบ RS-232 ซึ่งใช้คอนเน็กเตอร์ตัวผู้แบบ DB-9 ดังรูป



รูปที่ 3.8 คอนเน็กเตอร์ 9 ขาแบบ DB-9



รูปที่ 3.9 การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ RS-232

มาตรฐาน RS-232 มีระดับแรงดันของสัญญาณตั้งแต่ +3 ถึง +12 V และ –3 ถึง –12 V ดังนั้นจึงต้องมี IC เบอร์ DS275 เพื่อแปลงระดับแรงดันเป็น logic 1 และ 0 เนื่องจากต้องเชื่อมต่อกับไมโครคอนโทรลเลอร์

3.5 การออกแบบโครงสร้างเรือนเพาะชำ

การออกแบบโครงสร้างเรือนเพาะชำนั้นแบ่งออกเป็นการออกแบบ 2 ส่วนใหญ่ๆ คือ การออกแบบในส่วนของการควบคุมอุณหภูมิและการออกแบบในส่วนของการรดน้ำ โดยโครงของเรือนเพาะชำนั้นนั้นถูกคลุมด้วยพลาสติกเกษตร แบบกันแสง UV ด้วย โดยปิดเรือนเพาะชำทั้งหมดยกเว้น บริเวณด้านหลังของเรือนเพาะชำที่ต้องคลุมด้วย Slan และใยสังเคราะห์และบริเวณด้านหน้าที่มีพัดลมติดอยู่

3.5.1 การออกแบบโครงสร้างของการลดอุณหภูมิ

การออกแบบในส่วนของการควบคุมอุณภูมินั้นจะใช้สปริงเกิลและพัดลมในการควบคุมอุณหภูมิ ในส่วนของพัดลมจะติดบริเวณด้านหน้าของเรือนเพาะชำ โดยเป็นพัดลม AC หันหน้าออกข้างนอกเรือนเพาะชำเพื่อดูดอากาศภายในเรือนเพาะชำออกมา โดยด้านหลังของเรือนเพาะชำนั้นคลุมด้วย Slan ขนาด 50 เปอร์เซ็นต์ 2 ชั้นรวมทั้งมีใยสังเคราะห์อยู่ตรงกลาง โดย จะมี สปริงเกิล 2 ตัวคอยทำน้ำที่ฉีดน้ำให้ Slan และใยสังเคราะห์ โดยให้ทั้ง Slan และใยสังเคราะห์นั้นชุ่มน้ำ

ตลอดเวลา ขณะที่พัดลมดูดอากาศภายในเรือนเพาะชำออก อากาศที่อยู่ภายนอกก็จะไหลเข้าไปแทนที่ อากาศภายในเรือนเพาะชำโดยผ่านทางใยสังเคราะห์เพียงทิศทางเดียวเนื่องจาก ด้านอื่นๆ ของเรือนเพาะชำถูกปิดด้วยพลาสติกเกษตรทั้งหมด โดยอากาศภายนอกเมื่อไหนเข้าผ่านใยสังเคราะห์ที่ชุ่มน้ำก็จะทำให้อุณหภูมิของอากาศลดลง

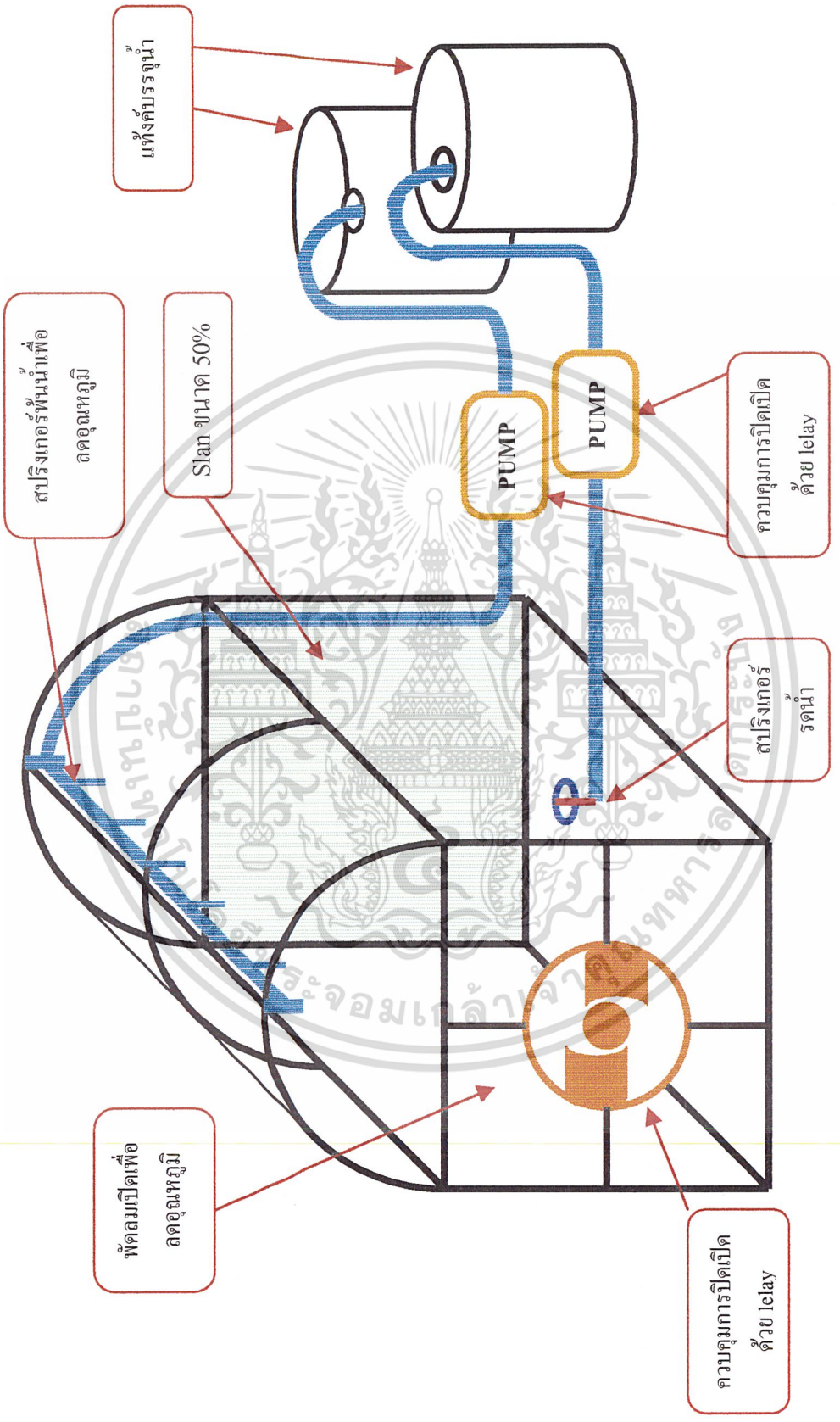
ส่วนสปริงเกิลแบบหมอกนั้นจะใช้ทั้งหมด 4 ตัว โดยสปริงเกิล 2 ตัวจะฉีดภายในเรือนเพาะชำเพื่อลดอุณหภูมิภายในเรือนเพาะชำโดยตรง และสปริงเกิลอีก2 ตัวจะติดอยู่บริเวณด้านหลังของเรือเพาะชำเพื่อฉีด Slan และใยสังเคราะห์เพื่อลดอุณหภูมิของอากาศภายนอกที่ไหลผ่านใยสังเคราะห์เข้ามา

ถ้าไมโครคอนโทรลเลอร์นั้นสั่งให้มีการลดอุณหภูมิปั๊มน้ำที่ควบคุม สปริงเกิลก็จะถูกเปิดในขณะเดียวกันที่พัดลมถูกเปิด สปริงเกิล 2 ตัวที่อยู่ในเรือนเพาะชำก็จะฉีดน้ำออกมาเพื่อลดอุณหภูมิ และพัดลมก็จะดูดละอองน้ำออกไปภายนอก และอากาศภายนอกก็จะเคลื่อนที่ผ่านใยสังเคราะห์ ที่ชุ่มน้ำจากสปริงเกิล อีก 2 ตัวบริเวณด้านหลังของเรือนเพาะชำมาแทนที่ เมื่อลดอุณหภูมิถึงอุณหภูมิที่เราต้องการแล้ว ไมโครคอนโทรลเลอร์ก็จะสั่งให้ปั๊มน้ำและพัดลม หยุดทำงานทันที และอุปกรณ์ทั้งสองจะทำงานอีกครั้งเมื่ออุณหภูมิสูงขึ้นอีก

3.5.2 การออกแบบในส่วนของการรดน้ำอัตโนมัติ

ในส่วนของระบบการรดน้ำอัตโนมัตินั้น การออกแบบจะใช้ปั๊มน้ำอีกตัวหนึ่ง จะถูกสั่งโดยไมโครคอนโทรลเลอร์ให้ทำงาน โดยการเปรียบเทียบเวลาที่เราต้องการรดน้ำกับเวลาจริงในขณะนั้นที่ควบคุมด้วย IC แสดงฐานเวลาจริง (Real Time Clock) เมื่อถึงเวลาที่ต้องการรดน้ำและปั๊มน้ำก็จะถูกเปิดตามระยะเวลาที่ผู้ใช้ต้องการอินพุทเข้าไป และจะถูกปิดเมื่อปริมาณการรดน้ำนั้นเทากับปริมาณที่ผู้ใช้อินพุทไว้เช่นกัน

การออกแบบในส่วนการรดน้ำนี้จะทำเป็นแบบจำลองโดย ใช้สปริงเกิลที่ใช้สำหรับรดน้ำ 1 ตัวเป็นตัวที่ใช้ในการแสดงผล โดยสปริงเกิลตัวนี้จะใช้ในการรดน้ำทั้งเรือนเพาะชำ โดยในระยะเวลา 20 วินาทีสปริงเกิลตัวนี้สามารถรดน้ำได้ m^3 เพราะฉะนั้น ในระยะเวลา 20 วินาที, 40 วินาที, 50 วินาที ทีเราสามารถ อินพุทเข้าไปได้เราจึงสามารถรดน้ำได้ในปริมาตร ตามลำดับ



รูปที่ 3.10 รูปแสดงโครงสร้างเรือนเพาะชำ

บทที่ 4
โครงสร้างทางซอฟท์แวร์

โครงงานระบบควบคุมสภาพแวดล้อมในเรือนเพาะชำนี้ในการควบคุมอุณหภูมิให้เหมาะสมต่อพืชแต่ละชนิดนั้นโดยเฉพาะพืชในเขตร้อน ได้ใช้ไมโครคอนโทรลเลอร์ในการควบคุมการการทำงานของฮาร์ดแวร์(การปิดเปิดของสวิตช์พัดลมและเครื่องสูบน้ำ) โดยรับอินพุทจากวงจรตรวจวัดอุณหภูมิ โดยสามารถป้อนค่าอุณหภูมิที่ต้องการควบคุมปริมาณน้ำและเวงาในการรดน้ำได้ผ่านทางคอมพิวเตอร์ที่มีส่วน Interface เขียนด้วยโปรแกรม Visual Basic 6 ทั้งยังสามารถแสดงผลและเก็บข้อมูลได้อีกด้วยการเขียนโปรแกรมควบคุมนั้นอาศัยชุดคำสั่งของไมโคคอนโทรลเลอร์ตระกูล MCS-51

4.1 วัตถุประสงค์ของโปรแกรม

- ส่วนควบคุม **Hardware**

ทำการรับอินพุทมาจากวงจรตรวจวัดอุณหภูมิที่แปลงจาก Analog to Digital แล้วผ่านทางพอร์ตของ IC เบอร์ 8255 และให้เอาท์พุทผ่านทางพอร์ตของ IC เบอร์ 8255 ไปควบคุมการปิดเปิดชองสวิตช์ของพัดลมและเครื่องสูบน้ำเพื่อรดน้ำและควบคุมอุณหภูมิให้เหมาะสมกับพืชชนิดนั้นๆ

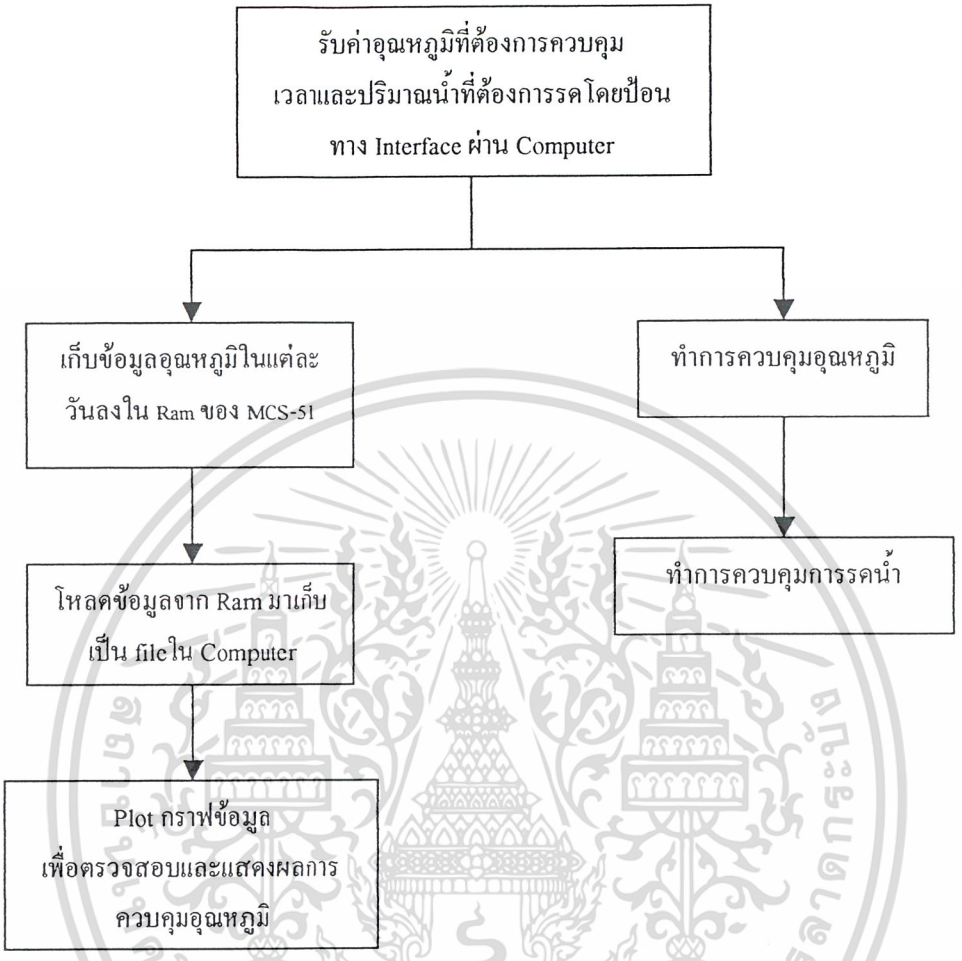
- ส่วน ติดต่อผู้ใช้งาน

โปรแกรมในส่วนนี้ทำให้ผู้ใช้งานสามารถป้อนอินพุทผ่านทาง Computer ซึ่งโปรแกรมนี้เขียนจากโปรแกรมภาษา Visual Basic 6 โดยส่งข้อมูลเชื่อมต่อกับไมโครคอนโทรลเลอร์ ผ่านทางพอร์ตอนุกรมของ Computer

4.2 ขั้นตอนการทำงานของโปรแกรม

4.2.1 ขั้นตอนโดยรวมทั้งหมดของโปรแกรม

1. รับข้อมูลจากส่วนติดต่อ
2. ส่งค่าไปที่ไมโครคอนโทรลเลอร์ผ่านทางพอร์ตอนุกรม
3. ทำการควบคุมอุณหภูมิและควบคุมการรดน้ำ
4. ส่งค่าอุณหภูมิในแต่วันไปเก็บเป้นไฟล์ข้อมูล



รูปที่ 4.1 แผนภาพแสดงการทำงานของโปรแกรมทั้งระบบ

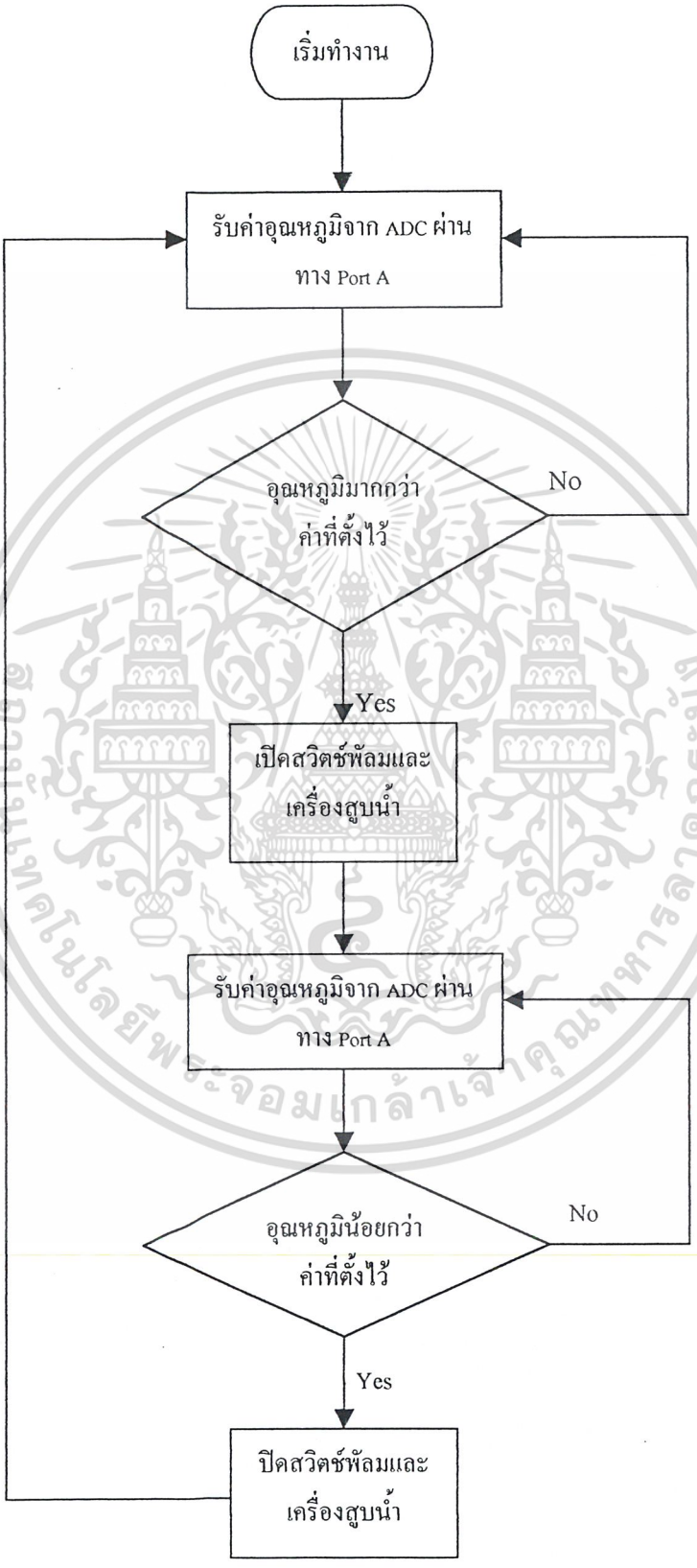
4.2.2 ส่วนควบคุมอุณหภูมิ

1. รับสัญญาณดิจิตอล 8 บิตที่แปลงมาจากวงจรAnolog to Digotal Conversion ซึ่งรับมาจากวงจรตรวจวัดอุณหภูมิ โดยรับมาทางพอร์ต A ของ IC เบอร์ 8255

2. ทำการเปรียบเทียบค่าของอุณหภูมิที่ตั้งตามชนิดของพืชไว้กับอุณหภูมิที่วัดได้จากวงจรตรวจวัดอุณหภูมิ หากค่าอุณหภูมิที่วัดได้น้อยกว่าค่าอุณหภูมิที่ตั้งไว้ก็จะทำการเปรียบเทียบใหม่อีกครั้ง

3. จนกระทั่งอุณหภูมิที่วัดได้มีค่ามากกว่าอุณหภูมิที่ตั้งไว้ ไมโครคอนโทรลเลอร์จะส่งเอาท์พุทผ่านทางพอร์ตC ของICเบอร์8255เพื่อไปเปิดสวิตช์พัดลมและเครื่องสูบน้ำเพื่อลดอุณหภูมิลงมาจนอุณหภูมิต่ำกว่าค่าที่ตั้งไว้ ไมโครคอนโทรลเลอร์จะเอาท์พุทผ่านทางพอร์ต C ของ IC เบอร์ 8255 เพื่อทำหารปิดสวิตช์พัดลมและเครื่องสูบน้ำ โดยที่

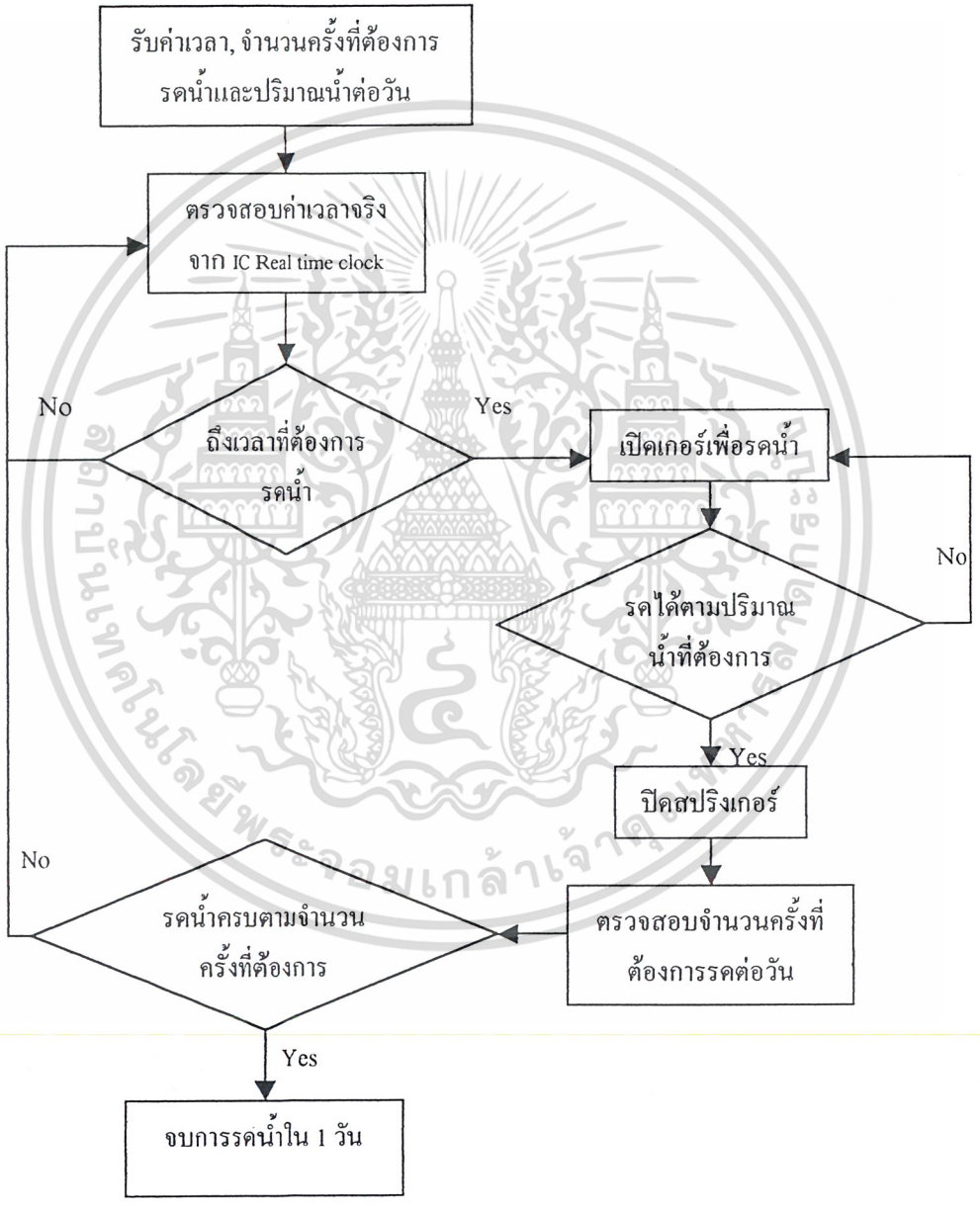
4. ทำเช่นนี้เรื่อยๆเพื่อรักษาระดับอุณหภูมิให้เหมาะสมกับพืชที่ทำการเพาะปลูก



รูปที่ 4.2 แผนภาพแสดงการทำงานของโปรแกรมควบคุมอุณหภูมิ

4.2.3 ส่วนควบคุมการรดน้ำ

1. รับค่าเวลาเวลาและปริมาณน้ำที่ต้องการรด
2. เปรียบเทียบเวลาที่ต้องรดกับค่าเวลาจริงจาก IC Real time clock
3. เมื่อถึงเวลาที่ต้องการรด ไมโครคอนโทรลเลอร์สั่งให้เปิดสปริงเกอร์เพื่อรดน้ำ



รูปที่ 4.3 แผนภาพแสดงการทำงานของโปรแกรมควบคุมการรดน้ำ

4.2.4 ส่วนติดต่อกับผู้ใช้

เป็นส่วนที่ใช้ป้อนอินพุทต่างทางคอมพิวเตอร์ซึ่งเขียนด้วยโปรแกรมภาษา Visual Basic 6 ซึ่งเป็นการสร้างระบบปฏิบัติการ Windows เป็นการพัฒนามาจากภาษา Basic โดยการออกแบบโปรแกรมจะเป็นลักาณะ Visualize ใช้ในการกำหนดตำแหน่งของ Object ลงบนจอภาพเพื่อติดต่อกับผู้ใช้โดยตรง Object เหล่านี้จะเปลี่ยนไปตามเหตุการณ์ต่างๆที่เกิดขึ้น ในการกำหนดขั้นตอนการทำงานให้กับ Object ภายใต้เหตุการณ์ใดๆ จะใช้ภาษา Basic เข้ามาช่วยในการเขียนโปรแกรม

การติดต่อกับ ไมโครคอนโทรลเลอร์ นั้นต้องติดต่อผ่านพอร์ตอนุกรมของคอมพิวเตอร์. อาศัยคอนโทรลของ Visual Basic คือ MSComm control ซึ่งมีประโยชน์ในการสร้าง Application ด้านการสื่อสาร

-คอนโทรล MSComm

คอนโทรล MSComm เป็นคอนโทรลที่ช่วยในการติดต่อกับพอร์ตอนุกรม (Serial Port) ให้สามารถทำการรับส่งข้อมูลผ่านพอร์ตอนุกรม โดยจะมีการตอบสนองเหตุการณ์แบบ even drive คือ คอนโทรลจะทำการตรวจสอบเหตุการณ์ต่างๆที่เกิดขึ้นกับพอร์ตอนุกรมโดยอัติโนมัติ

1. การกำหนดคุณสมบัติของ MSComm ทั่วไป

CommPort	หมายเลขพอร์ตอนุกรมที่ต้องการใช้ com1, com2, com3 com4
Setting	กำหนดค่าอัตราการอ่าน Buad rate, Parity bit, จำนวนบิต, บิตปิดท้าย
PortOpen	กำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรม

2. การกำหนดคุณสมบัติเกี่ยวกับ Buffer ข้อมูล

InputLen	กำหนดค่าและคืนค่าจำนวนของตัวอักษรที่อ่านจาก Buffer ภาครับ
InBufferCount	ส่งค่าจำนวนของตัวอักษรที่อยู่ใน Buffer ภาครับ
InBufferSize	กำหนดขนาดของ Buffer ภาครับเป็นจำนวนไบต์
Input	อ่านค่าและลบค่าข้อมูลจาก Buffer ภาครับ
InputMode	กำหนดค่าและคืนค่าชนิดของข้อมูลที่รับโดยคำสั่ง Input
Output	ใช้ในการส่งขบวนของข้อมูลไปยัง Buffer ส่งข้อมูล
RThreshold	กำหนดจำนวนตัวอักษรที่จะรับเข้าก่อนที่ MSComm กำหนดคุณสมบัติ CommEventมีค่าเท่ากับComEvReceiveและมีการเรียกเหตุการณ์ OnComm
SThreshold	จำนวนอักษรที่น้อยที่สุดที่ที่ถูกจัดเก็บในBufferด้านส่งออกก่อนที่ MSComm จะกำหนดคุณสมบัติ CommEvent มีค่าเท่ากับ comEvSend และมีการเรียกเหตุการณ์ OnComm

คอนโทรล MSComm จะมีการเรียก Procedure เหตุการณ์ OnComm ทุกครั้งที่เกิดข้อผิดพลาด หรือมีการสื่อสารเกิดขึ้น โดยค่าตัวเลขจำนวนเต็ม (Integer) แสดงถึงข้อผิดพลาดหรือเหตุการณ์ที่มีการสื่อสารดังกล่าว ก็จะเก็บไว้ในคุณสมบัติ CommEvent เสมอ ดังนั้นหากต้องการตรวจสอบข้อผิดพลาดหรือเหตุการณ์ที่มีการสื่อสารภาย Procedure OnComm ก็จะใช้ค่าตัวเลขในคุณสมบัติ CommEvent ในตรวจสอบเสมอ

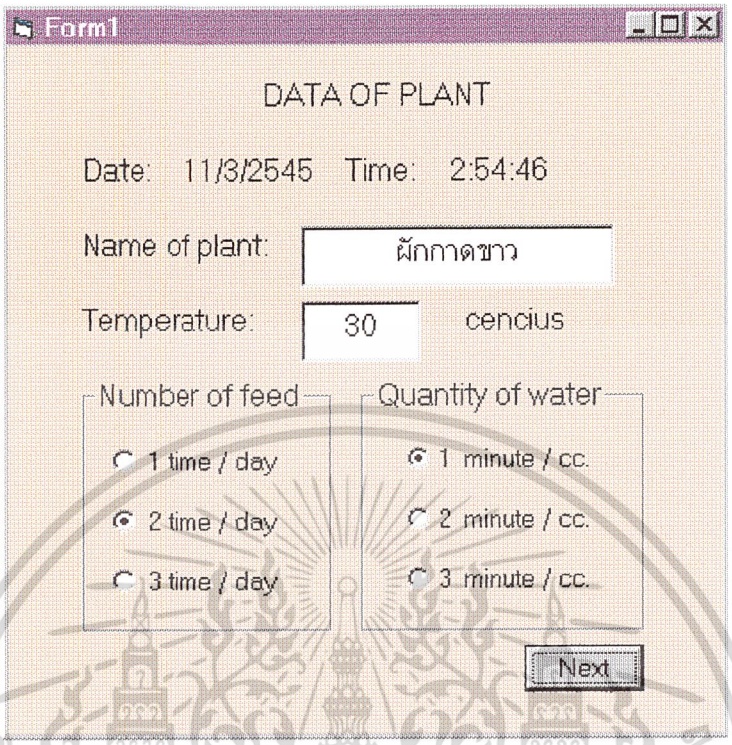
- หลักการทำงาน

การรับส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์และคอนโทรล MSComm ในแต่ละครั้งจะมีรหัส (Code) เพื่อให้ทั้ง ไมโครคอนโทรลเลอร์คอนโทรล MSComm รู้ว่าตอนนี้ต้องการรับส่งข้อมูลใด โดยมีการส่งรหัสออกไปก่อนแล้วตามด้วยข้อมูล

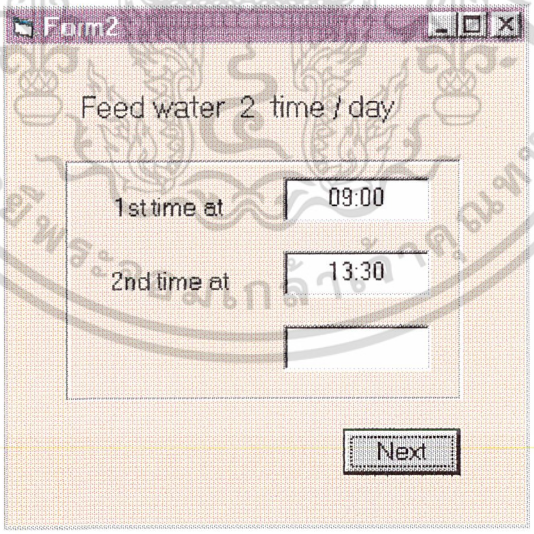
รหัส PC	การทำงานของ PC	รหัส MCS-51	การทำงานไมโครคอนโทรลเลอร์
41H(A)	บอก MCS-51ว่าจะส่งค่า วัน/เดือน/ปี จริงๆ เพื่อไปกำหนดค่าเวลาเริ่มต้นให้ RTC	61H(a)	บอก PC ว่าพร้อมจะรับค่า วัน/เดือน/ปี
42H(B)	บอก MCS-51ว่าจะส่งค่าเวลา จริงๆเพื่อไปกำหนดค่าเวลาเริ่มต้นให้ RTC	62H(b)	บอก PC ว่าพร้อมจะรับค่าเวลา
43H(C)	บอก MCS-51ว่าจะส่งค่าอุณหภูมิที่ต้องการควบคุม	63H(c)	บอก PC ว่าพร้อมจะรับค่าอุณหภูมิที่ต้องการควบคุม
44H(D)	บอก MCS-51ว่าจะส่งค่าจำนวนครั้งที่ต้องการรดน้ำ	64H(d)	บอก PC ว่าพร้อมจะรับค่าจำนวนครั้งที่ต้องการรดน้ำ
45H(E)	บอก MCS-51ว่าจะส่งค่าปริมาณน้ำที่ต้องการรด	65H(e)	บอก PC ว่าพร้อมจะรับค่าปริมาณน้ำที่ต้องการรด
46H(F)	บอก MCS-51ว่าจะส่งค่าเวลาที่ต้องการรดน้ำ	66H(f)	บอก PC ว่าพร้อมจะรับค่าเวลาที่ต้องการรดน้ำ
4CH(L)	บอก MCS-51ว่าจะโหลดค่าอุณหภูมิที่เก็บไว้ใน Ram	6CH(l)	บอก PC ว่าพร้อมจะโหลดค่าอุณหภูมิที่เก็บไว้ใน Ram

ตารางที่ 4.1 แสดงรหัสการรับส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์และคอมพิวเตอร์

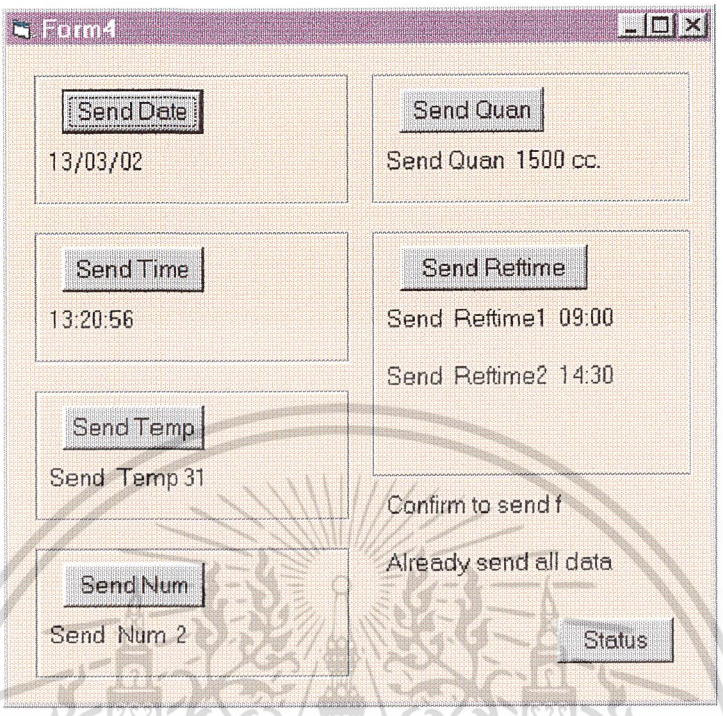
การรับส่งข้อมูลนั้นจะส่งเป็นรหัส ASCII เนื่องจากว่าเลือกการรับส่งข้อมูลในโหมด Text ดังนั้นในโปรแกรมภาษา Assembly ที่ใช้ควบคุมไมโครคอนโทรลเลอร์นั้นการส่งค่าออกมาจะต้องแปลงเป็นรหัส ASCII ของค่านั้นก่อน



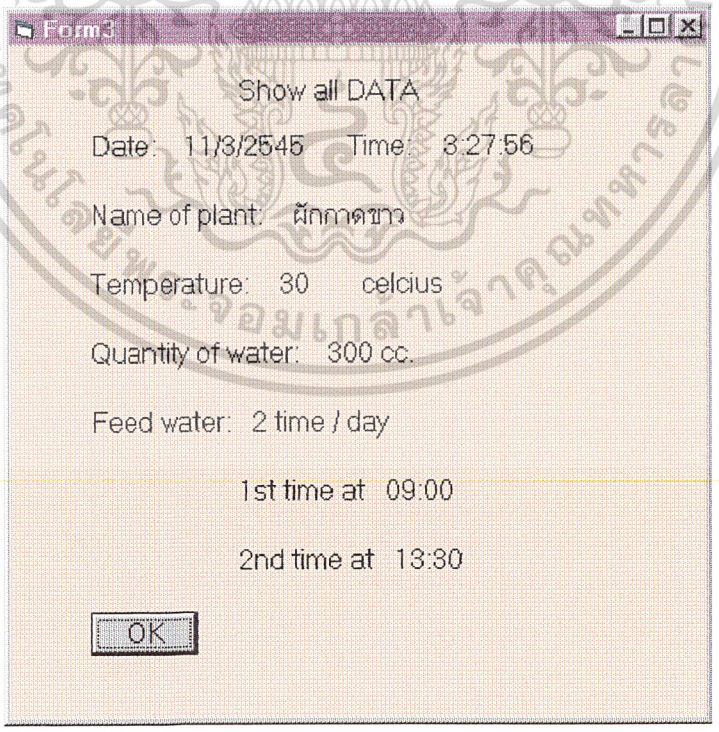
A. ป้อนข้อมูลพืชที่ต้องการควบคุม



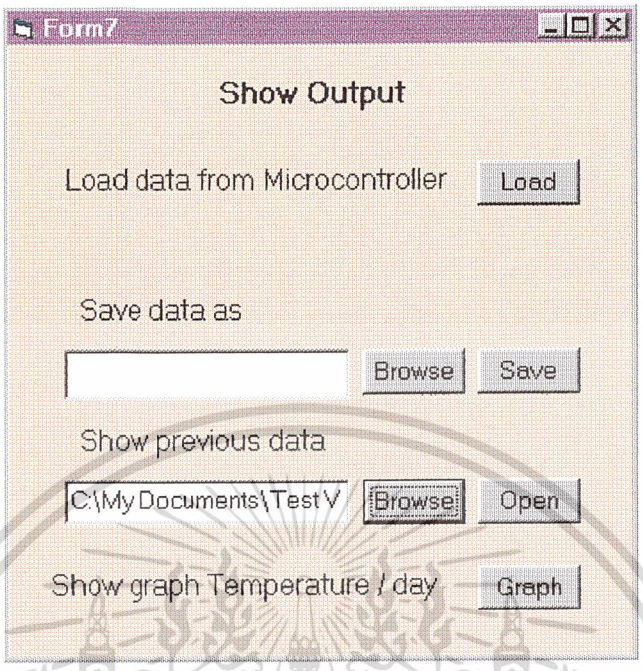
B. ป้อนค่าเวลาที่ต้องการรดน้ำ



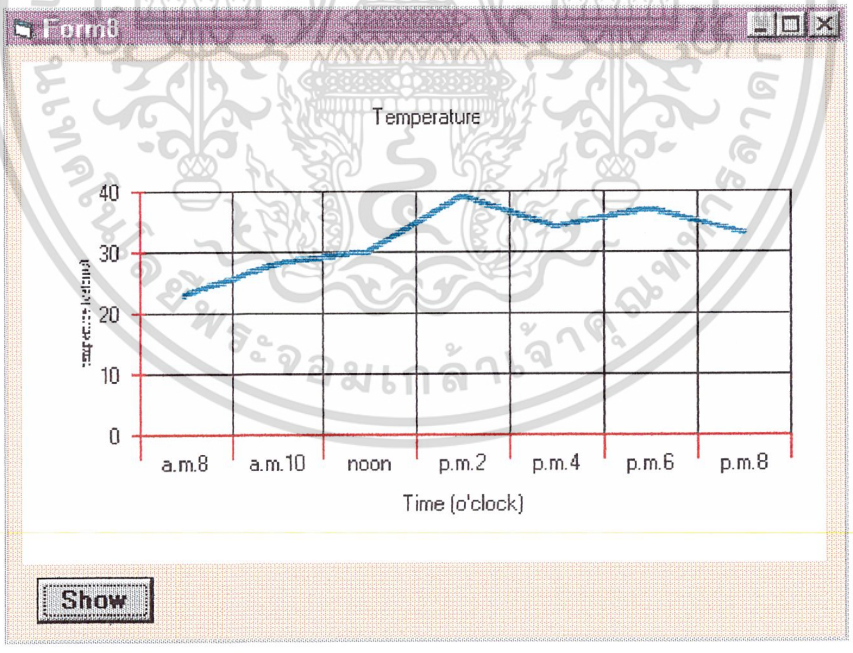
C. ส่งข้อมูลผ่าน Serial Port ไปให้ Microcontroller



D. แสดงผลข้อมูลของพืชที่ต้องการควบคุม



E. โหลดค่าอุณหภูมิจาก RAM



F. นำค่าอุณหภูมิที่โหลดมา Plot กราฟ

รูปที่ 4.4 แสดงส่วนติดต่อกับผู้ใช้เขียนด้วยโปรแกรมภาษา Basic

บทที่ 5

การทดลอง

จากการออกแบบวงจรตรวจวัดอุณหภูมิโดยใช้ Temperature sensor เบอร์ LM335 แล้วนำมาขยายสัญญาณด้วย Instrumentation Amplifier แล้วป้อนอินพุทให้ไมโครคอนโทรลเลอร์เพื่อควบคุมการเปิดปิดสวิตช์ Relay แล้วทำการทดลอง

5.1 <u>การทดลองคุณสมบัติการเป็นเชิงเส้นของ LM335</u>

1. ทำการปรับอุณหภูมิให้กับ LM335 ช่วง 20–50 องศาเซลเซียสเพิ่มขึ้นที่ละ 2 องศาเซลเซียสโดยใช้เทอร์โมมิเตอร์ปรอทเป็นตัววัด

2. วัดแรงดันเอาท์พุทจาก Instrumentation Amplifier

3. บันทึกค่าแรงดันเอาท์พุทลงในตารางแล้วเปรียบเทียบกับค่าอุณหภูมิที่วัดได้

<u>ผลการทดลอง</u>

อุณหภูมิ (องศาเซลเซียส)	แรงดันเอาท์พุท (โวลต์)
20	2
22	2.18
24	2.42
26	2.6
28	2.77
30	3.05
32	3.2
34	3.4
36	3.59
38	3.77
40	3.95
42	4.21
44	4.42
46	4.59
48	4.85
50	4.97

5.2 การทดลองการปิดเปิดสวิตช์ Relay

1. เขียนโปรแกรมให้เปิดสวิตช์ Relay ที่ 31 องศาเซลเซียส แล้วโหลดโปรแกรมลงในไมโครคอนโทรลเลอร์

2. ทำการปรับอุณหภูมิให้กับ LM335 ช่วง 20–50 องศาเซลเซียส เพิ่มขึ้นที่ละ 2 องศาเซลเซียสโดยใช้เทอร์โมมิเตอร์ปรอทเป็นตัววัด

3. สังเกตการเปิดปิดของสวิตช์ Relay แล้วบันทึกผล

ผลการทดลอง

อุณหภูมิ (องศาเซลเซียส)	สวิตช์ Relay
20	ปิด
22	ปิด
24	ปิด
26	ปิด
28	ปิด
30	ปิด
32	เปิด
34	เปิด
36	เปิด
38	เปิด
40	เปิด
42	เปิด
44	เปิด
46	เปิด
48	เปิด
50	เปิด

5.3 การทดลองวัดอุณหภูมิของบรรยากาศ และภายในโรงเพาะชำ

ทำการทดลองวัดค่าอุณหภูมิภายในเรือนเพาะชำขณะไม่มีการควบคุมอุณหภูมิ เทียบกับอุณหภูมิในบรรยากาศ ณ วันที่ 12 มีนาคม 2545 ตั้งแต่เวลา 12.30 น. ถึง 18.00 น.

ผลการทดลอง

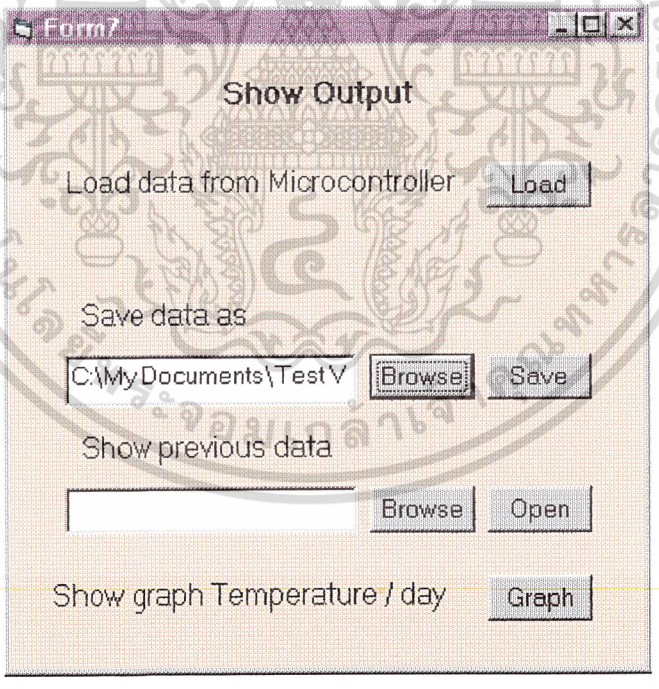
เวลา	อุณหภูมิในบรรยากาศ	อุณหภูมิในเรือนเพาะชำ
12.30	36	32
13.00	36	31.5
13.30	35.5	31.5
14.30	35	30.5
15.00	35	30
15.30	33.5	29.5
16.00	33.5	29
16.30	33	28.5
17.00	32.5	28
18.00	31	27

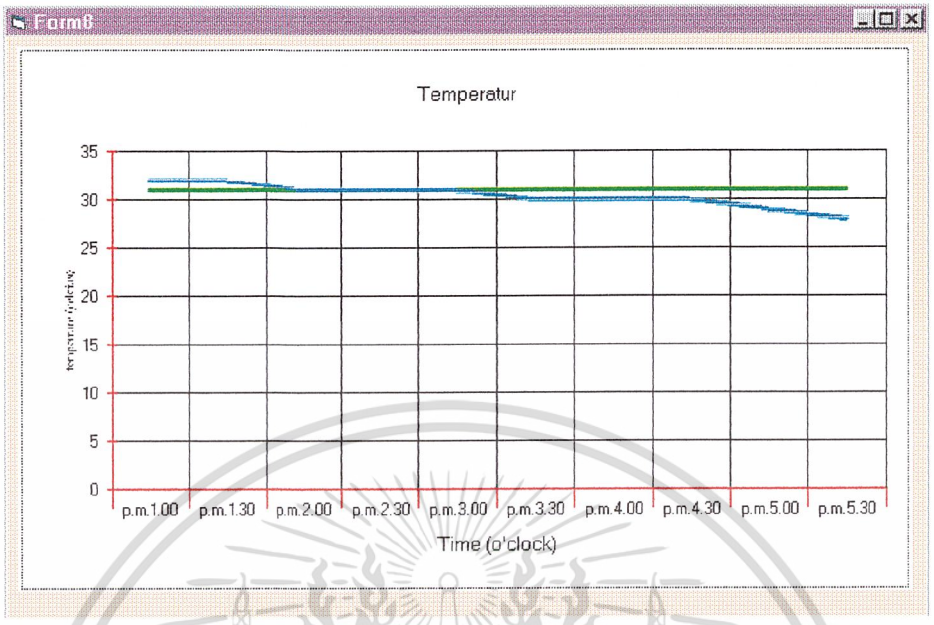
5.4 ทดลองทำการควบคุมอุณหภูมิภายในเรือนเพาะชำให้คงที่

ทำการทดลองวัดอุณหภูมิในเรือนเพาะชำขณะควบคุมให้ที่ ที่ 30 องศาเซลเซียส เทียบกับอุณหภูมิในบรรยากาศ ณ วันที่ 13 มีนาคม 2545 โดยเปิดเครื่องไว้ตั้งแ ต่เวลา 13.00 น. ถึง 17.30 น. แล้วทำการโหลดค่าที่เก็บในไมโครคอนโทรลเลอร์ โดยใช้ Interface ภาษา Visual basic 6 แล้วเก็บข้อมูลเป็น Textfile

ผลการทดลอง

เวลา	อุณหภูมิในเรือนเพาะชำ	อุณหภูมิที่ต้องการควบคุม
13.00	32	31
13.30	32	31
14.00	31	31
14.30	31	31
15.00	31	31
15.30	30	31
16.00	30	31
16.30	30	31
17.00	29	31
17.30	28	31



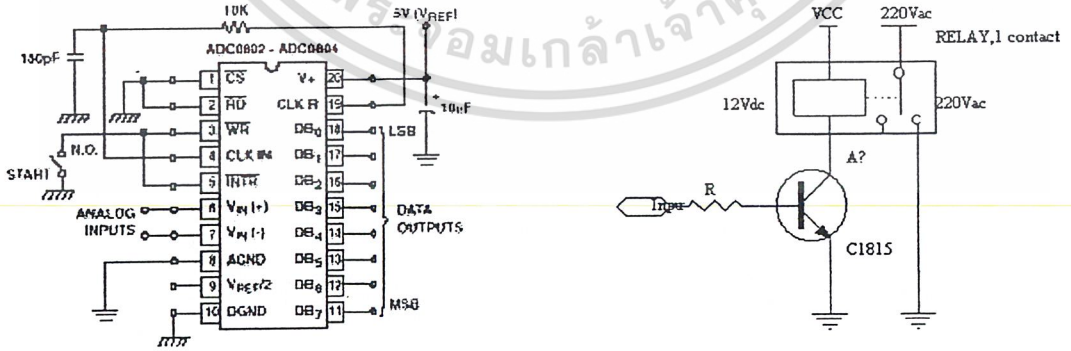


บทที่ 6
สรุปผลและวิจารณ์

6.1 ปัญหาที่พบและแนวทางแก้ไข

6.1.1 การออกแบบวงจรมีความล่าช้า เนื่องจากต้องพบกับปัญหาหลาย ๆ อย่างในการออกแบบ ได้แก่เรื่องอุปกรณ์ ต้องเลือกคุณสมบัติของอุปกรณ์ที่จะนำมาใช้งานได้อย่างมีประสิทธิภาพและมีราคาที่เหมาะสมกับงบประมาณ อุปกรณ์บางอย่างเป็นอุปกรณ์ที่มีประสิทธิภาพสูงแต่มีราคาที่สูงเกินไป ทำให้ไม่สามารถเลือกมาใช้ร่วมกับโครงงานชิ้นนี้ได้ หรืออุปกรณ์บางชนิดมีราคาถูกแต่มีประสิทธิภาพที่เพียงพอจะนำมาใช้งานภายในโครงงานได้

6.1.2 การออกแบบวงจร Digital to Analog Conversion และต้องการให้การทำงานของวงจรเป็นแบบ Free running ทำให้ต้องมีการต่อขา INT เชื่อมกับ ขา WR และต้องนำทั้ง 2 ขานี้ไปเชื่อมกับ Switch ซึ่งเดิมทางผู้จัดทำโครงงานต้องการสร้าง Switch จาก pulse ของไมโครคอนโทรลเลอร์ แต่เมื่อลองทดสอบการทำงานของ วงจร Digital to Analog Conversion แล้วปรากฏว่าไม่สามารถเริ่มต้นการ Conversion ได้ ทำให้ต้องมีการเปลี่ยนแปลงการออกแบบวงจร Switch ใหม่อีกครั้ง โดยทำให้เป็นการจ่าย Pulse ให้ทรานซิสเตอร์ ที่ทีหน้าที่ควบคุม Relay ซึ่งทำหน้าที่เปรียบเสมือนเป็น Switch แทน



รูปที่ 6.1 วงจร ACD และวงจรสวิตช์ Relay

6.1.3 เดิมเป้าหมายของการทำงานในเทอมที่ 1 คือต้องมีเรือนเพาะชำที่สามารถควบคุมอุณหภูมิได้ แต่เนื่องจากเหตุผล หลายๆ ประการเช่น ความไม่พร้อมของอุปกรณ์ อย่างปั๊มน้ำที่

อาจารย์จัดหามาให้เกิดความขัดข้องต้องส่งไปซ่อม หรือบริเวณที่ต้องการติดตั้งโรงเพาะชำบริเวณดาดฟ้าตึก B กำลังทำการปรับปรุงสถานที่อยู่ทำให้การจัดทำเรือนเพาะชำไม่มีความคืบหน้าเท่าที่ควร จนทำให้ไม่สามารถทำสำเร็จได้ในเทอมที่ 1

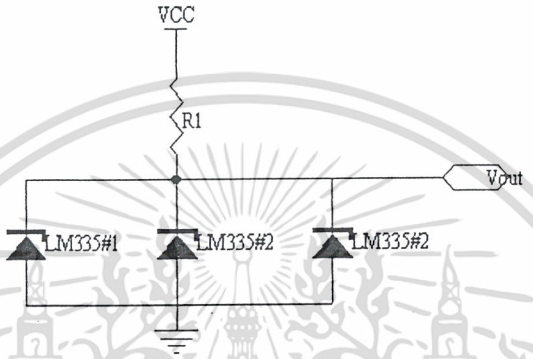
6.1.4 การออกแบบวงจรบางส่วนเกิดความผิดพลาด ส่งผลให้ลายปรินท์ผิดพลาดไปด้วย ทำให้ต้องทำการแก้ไข

6.1.5 ปัญหาที่เกิดขึ้นเนื่องจาก การออกแบบโครงสร้างของเรือนเพาะชำ เนื่องจากโครงงานชิ้นนี้เป็นโครงงานที่สร้างขึ้นเพื่อควบคุมระบบสมมุติที่เราทำการออกแบบขึ้น ให้มีสภาพแวดล้อมทางด้านอุณหภูมิแตกต่างกับสภาพแวดล้อมตามธรรมชาติ ดังนั้นปัญหาต่างๆ จึงเกิดขึ้นค่อนข้องมาก เช่นปัญหาเรื่องการควบคุมระบบให้มีความเสถียร เราอาจควบคมระบบให้มีอุณหภูมิต่างจากอุณหภูมิภายนอกได้ถึง 7 องศาในวันหนึ่ง แต่อีกวันหนึ่งเราสามารถควบคุมอุณหภูมิให้มีความแตกต่างจากสภาวะภายนอกเพียง 3 องศาเป็นต้น ทำให้เราไม่สามารถทราบประสิทธิภาพของระบบที่ถูกต้องแน่นอน นอกจากนี้ยังมีปัญหาเรื่องโครงสร้างของเรือนเพาะชำที่มีขนาดไม่เหมาะสม เช่นมีขนาดสั้นเกินไปเมื่อละอองน้ำของสปริงเกิลฉีดออกมา เมื่อโครงสร้างของเรือนเพาะชาค่อนข้างสั้นทำให้ละอองน้ำที่ฉีดออกมาถูกดูดออกไปภายนอกอย่างรวดเร็ว โดยที่ละอองน้ำยังไม่ทันจับกับความร้อนที่เกิดขึ้นภายในเรือนเพาะชำ โดยถ้าเรือนเพาะชำมีความยาวมากขึ้นความสามารถในการลดอุณหภูมิของเราก็อาจมีปริมาณมากขึ้น ทั้งนี้การแก้ไขอาจทำได้โดยการทดลองการทำงานของระบบจำนวนหลายๆ ครั้งเพื่อหาค่าเฉลี่ยของประสิทธิภาพที่แท้จริงของระบบ และจากผลการทดลองหลายๆ ครั้ง เราสามารถทราบข้อเสียของเรือนเพาะชำ และเราต้องทำการปรับปรุงโครงสร้างของเรื่อนเพาะชำเพื่อประสิทธิภาพที่ดีขึ้น

6.1.6 ปัญหาที่เกิดขึ้นเนื่องจากสปริงเกิล สปริงเกิลหมอกที่นำมาควบคุมอุณหภูมิ พ่นน้ำออกมาไม่ค่อยเป็นหมอกมากนัก ทำให้ขณะที่เราทำการลดอุณหภูมิ ภายในเรือนเพาะชำจึงค่อนข้างเปียก ทั้งที่เดิมประสิทธิภาพของสปริงเกิลแบบหมอก สามารถพ่นละอองนั้นในรูปที่เป็นหมอกมากกว่านี้ ทั้งนี้อาจเป็นเพราะว่า เราพ่นสปริงเกิลหลายจุดทำให้แรงอัดของปั๊มน้ำลดลงทำให้น้ำกลายเป็นหมอกน้อยลงด้วย เราอาจแก้ไขโดยการเปลี่ยนปั๊มน้ำให้เป็นปั๊มน้ำที่มีแรงอัดสูงขึ้น หรือเปลี่ยนไปใช้สปริงเกิลแบบควันที่ใช้ไฟฟ้าช่วยเปลี่ยนน้ำให้กลายเป็นควัน แต่การปรับปรุงทั้งสองอย่างต้องใช้งบประมาณสูงมาก และโครงงานนี้เป็นเพียงแบบจำลองการควบคุม ทำให้ผู้จัดทำโครงงานไม่ได้ทำการปรับปรุงแก้ไขในเรื่องนี้ โดยถ้ามีการนำโครงงานนี้ไปใช้จริง คงต้องมีการปรับปรุงแก้ไขต่อไป

6.1.7 ปัญหาที่เกิดขึ้นเนื่องจาก การคิดตั้ง Sensor อุณหภูมิ การที่วงจรควบคุมใช้ เซนเซอร์อุณหภูมิเพียงตัวเดียวทำให้เกิดปัญหาของตำแหน่งการติดตั้งเซนเซอร์อุณหภูมิให้มีประสิทธิภาพ

สูงสุดที่จะสามารถบอกค่าเฉลี่ยของเรือนเพาะชำได้ ผู้จัดทำโครงงานจึงตัดสินใจเลือกติดบริเวณกลางเรือนเพาะชำซึ่งความผิดอาจเกิดขึ้นได้ ในกรณีที่อุณหภูมิภายในเรือนเพาะชำมีความไม่สม่ำเสมอเกิดขึ้น และจากการค้นคว้าเพิ่มเติม เราอาจแก้ไขปัญหาเรื่องนี้โดยการเพิ่มจำนวนเซนเซอร์แล้วต่อขนานกันดังรูป



รูปที่ 6.2 การต่อขนานกันของเซนเซอร์

โดยอุณหภูมิที่ เซนเซอร์อุณหภูมิส่งออกมา คืออุณหภูมิต่ำที่สุดที่ เซนเซอร์ทั้ง 3 ตัววัดได้ ดังนั้นถ้าเราเพิ่มจำนวนเซนเซอร์แล้วนำไปติดที่จุดต่างๆ ของโครงสร้างเรือนเพาะชำ แล้วนำมาต่อขนานกันเราก็จะสามารถควบคุมอุณหภูมิในจุดต่างๆ ภายในเรือนเพาะชำได้

6.1.8 เดิมวงจรในเทอมที่ 1 ได้มีการนำวงจร Current to Voltage Converter และ Voltage to Current Converter มาเกี่ยวข้องเพื่อทำการส่ง Voltage ระยะไกล แต่เมื่อลองทำการศึกษาเรื่อง Serial Port แล้วปรากฏว่า Serial Port นั้นสามารถส่งข้อมูลได้อย่างมีประสิทธิภาพถึง 20 เมตร และนอกจากนี้ยังเกิดปัญหาในเรื่องการจ่ายไฟเลี้ยงให้วงจร Voltage to Current Converter ที่อยู่ติดกับเซนเซอร์เพราะเราต้องจ่ายไฟเลี้ยงในระยะไกลตามไปด้วยจึงเป็นการยุ่งยากและการสิ้นเปลืองสาย จึงตัดวงจรส่วนนี้ทิ้งแล้วเพิ่มความยาวของสายพอร์ตอนุรมแทน

6.1.9 การแก้ไขปัญหาเรื่องความชื้น การลดอุณหภูมิด้วยการใช้พัดลมและสปริงเกิล นั้นปัญหาที่เราไม่สามารถทำการควบคุมได้คือความชื้น เมื่อเราต้องการลดอุณหภูมิเราต้องทำการเปิดพัดลมและปั๊มน้ำเพื่อฉีดสปริงเกิล ดังนั้นเมื่อเราต้องการลดอุณหภูมิ ความชื้นจึงเพิ่มขึ้นตลอดเวลา อาจสามารถลดได้นิดหน่อยโดยการเปิดพัดลมค้างไว้เพื่อไล่ความชื้นออกไป แต่ก็เป็นเพียงปริมาณเล็กน้อยเท่านั้น ถ้าเราต้องการควบคุมความชื้นด้วยเราต้องเปลี่ยนวิธีการลดอุณหภูมิเช่นใช้ Air Conditioner แทนการใช้พัดลมและปั๊มน้ำ

ส่วนวงจรวัดความชื้นนั้นเนื่องจากติดปัญหาทางด้านงบประมาณ (เซนเซอร์ความชื้นมีราคาสูงมาก) และปัญหาทางด้านระยะเวลาทำให้ผู้จัดทำโครงงาน ไม่สามารถออกแบบให้เสร็จลุล่วงไปได้ทั้งที่มีการค้นคว้าข้อมูลไว้แล้ว

เซนเซอร์ความชื้นตัวนี้อยู่ในรูป Capacitor โดยค่า C จะเปลี่ยนแปลงตามความชื้นของบรรยากาศ เรานำไปแทนค่า C ในวงจรสร้างสัญญาณนาฬิกา ของ IC 555 โดยเอาท์พุทของ IC 555 จะเป็นสัญญาณนาฬิกาที่ขึ้นกับความชื้น เราสามารถใช้ Timer ในไมโครคอนโทรลเลอร์เป็นตัวนับความถี่สัญญาณนาฬิกาที่ IC 555 สร้างขึ้นได้ โดยมี look up table เป็นของมูลของความชื้นที่ความถี่นั้น ๆ

6.2 <u>สรุปและวิจารณ์</u>

จากโครงงานนี้เรามีเรือนเพาะชำอิเล็กทรอนิกส์ที่สามารถควบคุมอุณหภูมิได้และมีระบบให้น้ำอัตโนมัติโดยเราสามารถควบคุมผ่านเครื่องคอมพิวเตอร์ โดยมีการแสดงสถานะของเรือนเพาะชำ รวมทั้งสามารถป้อนอินพุทผ่านเครื่องคอมพิวเตอร์ได้ เช่นอุณหภูมิที่เราต้องการควบคุม ปริมาณการรดน้ำเป็นเวลา จำนวนครั้งของการรดน้ำ รวมไปถึงเวลาที่เราต้องการรดน้ำในแต่ละวัน และเราสามารถเก็บข้อมูลเหล่านี้ในรูปไฟล์ และสามารถพล็อตกราฟ อุณหภูมิที่เปลี่ยนไปในแต่ละวันได้

เมื่อเราป้อนอินพุตผ่านเครื่องคอมพิวเตอร์ โปรแกรม Visual Basic ก็จะส่งข้อมูลในรูปรหัส แอสกี ผ่านพอร์ตอนุกรม ไปยัง MCS-51 และ MCS-51 ก็จะแปลงข้อมูลเหล่านี้เป็น BCD Code หรือเลขฐาน 16 ตามแต่เราต้องการนำมันไปใช้งาน ในการเป็นค่าอ้างอิงที่เราจะนำมันไปใช้ในการควบคุมต่อไป

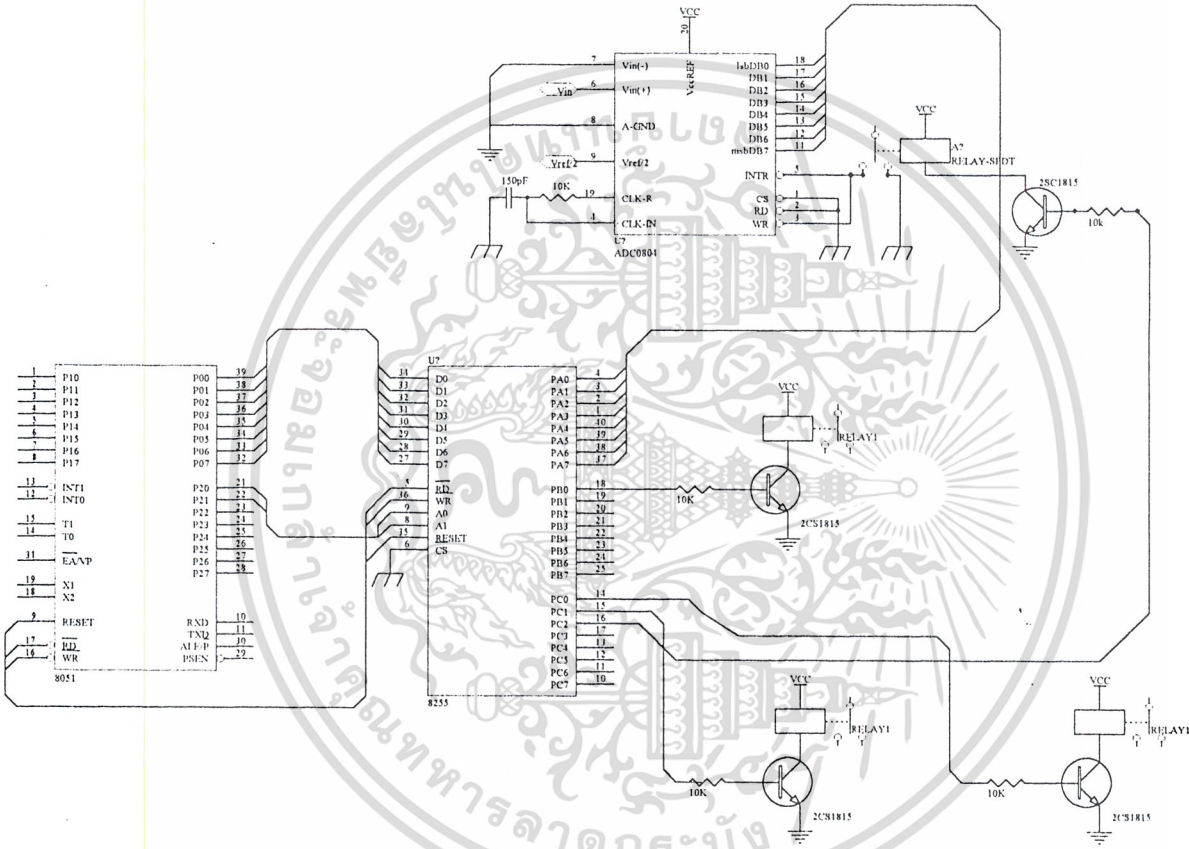
และเมื่ออุณหภูมิที่ Sensor อุณหภูมิตรวจจับอุณหภูมิได้ ค่ามากกว่าอุณหภูมิอ้างอิงที่เราต้องการทำการควบคุม ไมโครคอนโทรลเลอร์จะสั่งให้ Relay ที่มีหน้าที่ควบคุม พัดลมและปั๊มน้ำปิดวงจรเพื่อทำให้พัดลมและปั๊มน้ำสามารถทำงานได้ เพื่อทำการลดอุณหภูมิ

เช่นเดียวกับการรดน้ำ ไมโครคอนโทรลเลอร์จะทำการเปรียบเทียบเวลาที่เราต้องการรดน้ำกับเวลาจริง เมื่อถึงเวลาที่เราต้องการรดน้ำ ไมโครคอนโทรลเลอร์ก็จะสั่ง Relay ให้ สปริงเกิลทำงาน เพื่อรดน้ำ โดยระยะเวลาที่ Relay ทำงานก็ขึ้นอยู่กับอินพุทปริมาณน้ำที่เราต้องการรด

ปัญหาที่เกิดขึ้นขณะจัดทำโครงงาน ก็พบว่าปัญหาส่วนใหญ่ก็สามารถทำการปรับปรุงแก้ไขได้แต่ความยากง่ายในการไขปัญหาก็แตกต่างกันไป ดังที่เรากล่าวมาแล้วข้างต้น บางเรื่องเราก็สามารถปรับปรุงแก้ไขได้แต่ในบางเรื่องก็ไม่สามารถแก้ไขได้ เนื่องจากติดปัญหาในหลายๆ เรื่อง เช่นเรื่องงบประมาณ หรือระยะเวลา แต่ทั้งนี้ผู้จัดทำโครงงานชิ้นนี้ยังได้ความรู้พื้นฐานทาง

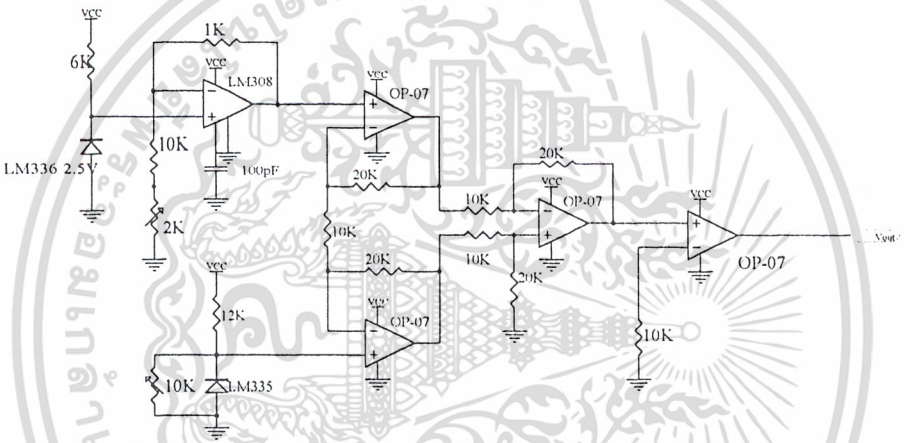
อิเล็กทรอนิกส์จากโครงงานชิ้นนี้หลายประการ เพราะเป็นโครงงานที่กว้างและครอบคลุม เช่น ความรู้ในเรื่อง ไมโครคอนโทรเลอร์, วงจร Power Electronics, Programming เพราะฉะนั้นโครงงานชิ้นนี้จึงเป็นโครงงานที่มีประโยชน์ และน่าสนใจและควรมีการปรับปรุงแก้ไขต่อไป เพื่อให้โครงงานชิ้นนี้มีประสิทธิภาพมากขึ้นและเหมาะสมในการนำไปใช้ประโยชน์ได้จริงต่อไป รวมทั้งผู้จัดทำยังได้ความรู้ทางอิเล็กทรอนิกส์ที่ครอบคลุมอีกด้วย

ภาคผนวก



รูป ข. วงจรส่วนควบคุมฮาร์ดแวร์

Title		
Size: B	Number	Revision
Date:	11-Mar-2002	Sheet of
File:	C:\PEWSCIF-I\CONTROL.SCH	Drawn By:



รูป ก. วงจรตรวจวัดอุณหภูมิ

Title			
Size: B	Number		Revision
Date:	11-Mar-2002	Sheet of	
File:	C:\PEWS\CIF4\TEMP-AM.SCH	Drawn By:	

```
1           PORTA    EQU     0000h
            PORTB    EQU     0100h
            PORTC    EQU     0200h
            P_CON    EQU     0300h
5           CODE     EQU     0030H
            HI_DAY   EQU     0031H
            LO_DAY   EQU     0032H
        HI_MONTH     EQU     0033H
        LO_MONTH     EQU     0034H
10      HI_YEAR      EQU     0035H
        LO_YEAR      EQU     0036H
            HI_HOUR  EQU     0037H
            LO_HOUR  EQU     0038H
            HI_MIN   EQU     0039H
15          LO_MIN   EQU     003AH
            HI_SEC   EQU     003BH
            LO_SEC   EQU     003CH
            HI_TEMP  EQU     003DH
            LO_TEMP  EQU     003EH
20          NUM      EQU     003FH
            QUAN     EQU     0040H
        HI_REFHOUR1  EQU     0041H
        LO_REFHOUR1  EQU     0042H
        HI_REFMIN1   EQU     0043H
25      LO_REFMIN1   EQU     0044H
        HI_REFHOUR2  EQU     0045H
        LO_REFHOUR2  EQU     0046H
        HI_REFMIN2   EQU     0047H
        LO_REFMIN2   EQU     0048H
30      HI_REFHOUR3  EQU     0049H
        LO_REFHOUR3  EQU     004AH
        HI_REFMIN3   EQU     004BH
        LO_REFMIN3   EQU     004CH
        BCD_HOUR     EQU     004DH
35      BCD_MIN      EQU     004EH
        BCD_SEC      EQU     004FH
        BCD_DAY      EQU     0050H
        BCD_MONTH    EQU     0051H
        BCD_YEAR     EQU     0052H
40      HEX_TEMP     EQU     0053H
        HEX_NUM      EQU     0054H
        HEX_QUAN          EQU     0055H
        BCD_HI_RFHOUR1    EQU     0056H
        BCD_LO_RFHOUR1    EQU     0057H
45      BCD_HI_RFHOUR2    EQU     0058H
        BCD_LO_RFHOUR2    EQU     0059H
        BCD_HI_RFHOUR3    EQU     005AH
        BCD_LO_RFHOUR3    EQU     005BH
        BCD_HI_RFMIN1     EQU     005CH
50      BCD_LO_RFMIN1     EQU     005DH
        BCD_HI_RFMIN2     EQU     005EH
        BCD_LO_RFMIN2     EQU     005FH
        BCD_HI_RFMIN3     EQU     0060H
        BCD_LO_RFMIN3     EQU     0061H
55      DATE              EQU     0062H
        MONTH        EQU     0063H
        YEAR         EQU     0064H
        HOURS        EQU     0065H
        SECONDS      EQU     0066H
60      MINUTES      EQU     0067H
```

```
DAY             EQU     0068H
CODE1           EQU     0069H
SCL             EQU     P2.2
SDA             EQU     P2.3
FLAG            EQU     002FH
I2C_ACK         BIT     FLAG.0
CHECK_STATE     BIT     FLAG.1
RTC_ID          EQU     11010000B   ;RTC SLAVE ADDRESS
I2C_ADDR        EQU     006BH
I2C_DATA        EQU     006CH
CONTROL         EQU     006DH
SET_TEMP        EQU     006EH
REAL_TEMP       EQU     006FH
ODD_EVEN        EQU     0070H
TIME_ON         EQU     0071H

                ORG     0000H
                SJMP    BEGIN

                ORG     0023H
                PUSH        PSW
                JB      RI,RECEIVE
                CLR     TI
                SJMP        FIN_INTR
RECEIVE:        MOV     A,SBUF
                MOV     CODE1,A
                MOV     R0,#CODE1
                CJNE    @R0,#4CH,FIN_INTR           ;'L' HMS
                LCALL   LOAD_DATA
FIN_INTR:       POP     PSW
                RETI

BEGIN:              acall       delay_10ms
                    acall       delay_10ms
                    acall       delay_10ms
                    mov     dptr,#p_con
                    mov     a,#82h
                    movx        @dptr,a

                MOV A,#0FFH
                MOV DPTR,#PORTC
                MOVX    @DPTR,A

                ACALL   DELAY_1S

                MOV A,#00H
                MOVX    @DPTR,A


SERIAL:             MOV     SCON,#01010010B     ;SELECT MODE1
                    MOV     A,PCON
                    CLR     ACC.7
                    MOV     PCON,A
                    MOV     TMOD,#20H           ;SELECT TIMER1
                    MOV     TH1,#-3             ;SELECT BUAD RATE
                    SETB    TR1                 ;SET MODE WORK

CH_READY:       JNB     RI,CH_READY
                CLR     RI                      ;CHECK INPUT
```

```
                    MOV     A,SBUF
                    MOV     R0,#CODE
                    MOV     @R0,A

125     SEND_DMY:   CJNE    @R0,#41H,SEND_HMS   ;'A' HMS
                    CALL    SEND_DMY1
                    SJMP    CH_READY
        SEND_HMS:   CJNE    @R0,#42H,SEND_TEMP  ;'B' DMY
                    CALL    SEND_HMS1
130                 SJMP    CH_READY
        SEND_TEMP:  CJNE    @R0,#43H,SEND_NUM   ;'C' QUAN
                    CALL    SEND_TEMP1
                    SJMP    CH_READY
        SEND_NUM:   CJNE    @R0,#44H,SEND_QUAN  ;'D' TEMP
135                 CALL    SEND_NUM1
                    SJMP    CH_READY
        SEND_QUAN:  CJNE    @R0,#45H,SEND_RHM   ;'E' NUM
                    CALL    SEND_QUAN1
                    SJMP    CH_READY
140     SEND_RHM:   CJNE    @R0,#46H,CH_READY   ;'F' REFHM
                    CALL    SEND_RHM1
                    LJMP    ADJ_HOUR

        SEND_DMY1:  JNB     TI,SEND_DMY1
145                 CLR     TI
                    MOV     A,#61H              ;'a'
                    MOV     SBUF,A
        REC_DAY1:   JNB     RI,REC_DAY1         ;RECIEVE REAL DAY
                    CLR     RI
150                 MOV     A,SBUF
                    MOV     R0,#HI_DAY
                    MOV     @R0,A
        REC_DAY2:   JNB     RI,REC_DAY2
                    CLR     RI
155                 MOV     A,SBUF
                    MOV     R0,#LO_DAY
                    MOV     @R0,A
        REC_MONTH1: JNB     RI,REC_MONTH1       ;RECIEVE REAL MONT
        H
                    CLR     RI
160                 MOV     A,SBUF
                    MOV     R0,#HI_MONTH
                    MOV     @R0,A
        REC_MONTH2: JNB     RI,REC_MONTH2       ;RECIEVE REAL MONT
        H
                    CLR     RI
165                 MOV     A,SBUF
                    MOV     R0,#LO_MONTH
                    MOV     @R0,A
        REC_YEAR1:  JNB     RI,REC_YEAR1        ;RECIEVE REAL YEAR
                    CLR     RI
170                 MOV     A,SBUF
                    MOV     R0,#HI_YEAR
                    MOV     @R0,A
        REC_YEAR2:  JNB     RI,REC_YEAR2
                    CLR     RI
175                 MOV     A,SBUF
                    MOV     R0,#LO_YEAR
                    MOV     @R0,A
                    RET
```

```
SEND_HMS1:  JNB     TI,SEND_HMS1
            CLR     TI
            MOV     A,#62H              ;'b'
            MOV     SBUF,A
REC_HOUR1:  JNB     RI,REC_HOUR1        ;RECIEVE REAL HOUR
            CLR     RI
            MOV     A,SBUF
            MOV     R0,#HI_HOUR
            MOV     @R0,A
REC_HOUR2:  JNB     RI,REC_HOUR2
            CLR     RI
            MOV     A,SBUF
            MOV     R0,#LO_HOUR
            MOV     @R0,A
REC_MIN1:   JNB     RI,REC_MIN1         ;RECIEVE REAL MIN
            CLR     RI
            MOV     A,SBUF
            MOV     R0,#HI_MIN
            MOV     @R0,A
REC_MIN2:   JNB     RI,REC_MIN2
            CLR     RI
            MOV     A,SBUF
            MOV     R0,#LO_MIN
            MOV     @R0,A
REC_SEC1:   JNB     RI,REC_SEC1         ;RECIEVE REAL SEC
            CLR     RI
            MOV     A,SBUF
            MOV     R0,#HI_SEC
            MOV     @R0,A
REC_SEC2:   JNB     RI,REC_SEC2
            CLR     RI
            MOV     A,SBUF
            MOV     R0,#LO_SEC
            MOV     @R0,A
            RET

SEND_TEMP1: JNB     TI,SEND_TEMP1
            CLR     TI
            MOV     A,#63H              ;'c'
            MOV     SBUF,A
REC_TEMP1:  JNB     RI,REC_TEMP1        ;RECIEVE REF TEMP
            CLR     RI
            MOV     A,SBUF
            MOV     R0,#HI_TEMP
            MOV     @R0,A
REC_TEMP2:  JNB     RI,REC_TEMP2
            CLR     RI
            MOV     A,SBUF
            MOV     R0,#LO_TEMP
            MOV     @R0,A
            RET

SEND_NUM1:  JNB     TI,SEND_NUM1
            CLR     TI
            MOV     A,#64H              ;'d'
            MOV     SBUF,A
REC_NUM:    JNB     RI,REC_NUM          ;RECEIVE NUM
            CLR     RI
            MOV     A,SBUF
```

```
                MOV     R0,#NUM
240             MOV     @R0,A
                RET

        SEND_QUAN1: JNB     TI,SEND_QUAN1
                CLR     TI
245             MOV     A,#65H              ;'e'
                MOV     SBUF,A
        REC_QUAN1:  JNB     RI,REC_QUAN1        ;RECIEVE REAL SEC
                CLR     RI
                MOV     A,SBUF
250             MOV     R0,#QUAN
                MOV     @R0,A
                RET

        SEND_RHM1:  JNB     TI,SEND_RHM1
255             CLR     TI
                MOV     A,#66H              ;'f'
                MOV     SBUF,A
        REC_REFH11: JNB     RI,REC_REFH11       ;RECIEVE REF HOUR1
                CLR     RI
260             MOV     A,SBUF
                MOV     R0,#HI_REFHOUR1
                MOV     @R0,A
        REC_REFH12: JNB     RI,REC_REFH12
                CLR     RI
265             MOV     A,SBUF
                MOV     R0,#LO_REFHOUR1
                MOV     @R0,A
        REC_REFM11: JNB     RI,REC_REFM11       ;RECIEVE REF MIN1
                CLR     RI
270             MOV     A,SBUF
                MOV     R0,#HI_REFMIN1
                MOV     @R0,A
        REC_REFM12: JNB     RI,REC_REFM12
                CLR     RI
275             MOV     A,SBUF
                MOV     R0,#LO_REFMIN1
                MOV     @R0,A

        CH_RHM1:    MOV     R0,#NUM
280             CJNE    @R0,#31H,REC_REFH21
                SJMP    FIN_RHM

        REC_REFH21: JNB     RI,REC_REFH21       ;RECIEVE REF HOUR2
                CLR     RI
285             MOV     A,SBUF
                MOV     R0,#HI_REFHOUR2
                MOV     @R0,A
        REC_REFH22: JNB     RI,REC_REFH22
                CLR     RI
290             MOV     A,SBUF
                MOV     R0,#LO_REFHOUR2
                MOV     @R0,A
        REC_REFM21: JNB     RI,REC_REFM21       ;RECIEVE REF MIN2
                CLR     RI
295             MOV     A,SBUF
                MOV     R0,#HI_REFMIN2
                MOV     @R0,A
        REC_REFM22: JNB     RI,REC_REFM22
```

Program control hardware

```
                CLR     RI
300             MOV     A,SBUF
                MOV     R0,#LO_REFMIN2
                MOV     @R0,A

    CH_RHM2:    MOV     R0,#NUM
305             CJNE    @R0,#32H,REC_REFH31
                SJMP    FIN_RHM

    REC_REFH31: JNB     RI,REC_REFH31       ;RECIEVE REF HOUR3
                CLR     RI
310             MOV     A,SBUF
                MOV     R0,#HI_REFHOUR3
                MOV     @R0,A
    REC_REFH32: JNB     RI,REC_REFH32
                CLR     RI
315             MOV     A,SBUF
                MOV     R0,#LO_REFHOUR3
                MOV     @R0,A
    REC_REFM31: JNB     RI,REC_REFM31       ;RECIEVE REF MIN3
                CLR     RI
320             MOV     A,SBUF
                MOV     R0,#HI_REFMIN3
                MOV     @R0,A
    REC_REFM32: JNB     RI,REC_REFM32
                CLR     RI
325             MOV     A,SBUF
                MOV     R0,#LO_REFMIN3
                MOV     @R0,A
    FIN_RHM:    RET

330 ;================ADJUST=============================;

    ADJ_HOUR:   MOV     A,HI_HOUR
                SUBB    A,#30H
                SWAP    A
335             MOV     BCD_HOUR,A
                MOV     A,LO_HOUR
                SUBB    A,#30H
                MOV     R1,#BCD_HOUR
                ORL     A,@R1
340             MOV     BCD_HOUR,A

    ADJ_MIN:    MOV     A,HI_MIN
                SUBB    A,#30H
                SWAP    A
345             MOV     BCD_MIN,A
                MOV     A,LO_MIN
                SUBB    A,#30H
                MOV     R1,#BCD_MIN
                ORL     A,@R1
350             MOV     BCD_MIN,A


    ADJ_SEC:    MOV     A,HI_SEC
                SUBB    A,#30H
355             SWAP    A
                MOV     BCD_SEC,A
                MOV     A,LO_SEC
                SUBB    A,#30H
```

```
                MOV     R1,#BCD_SEC
360             ORL     A,@R1
                MOV     BCD_SEC,A

        ADJ_DAY:MOV     A,HI_DAY
                SUBB    A,#30H
365             SWAP    A
                MOV     BCD_DAY,A
                MOV     A,LO_DAY
                SUBB    A,#30H
                MOV     R1,#BCD_DAY
370             ORL     A,@R1
                MOV     BCD_DAY,A

        ADJ_MONTH:MOV   A,HI_MONTH
                SUBB    A,#30H
375             SWAP    A
                MOV     BCD_MONTH,A
                MOV     A,LO_MONTH
                SUBB    A,#30H
                MOV     R1,#BCD_MONTH
380             ORL     A,@R1
                MOV     BCD_MONTH,A

        ADJ_YEAR:MOV    A,HI_YEAR
                SUBB    A,#30H
385             SWAP    A
                MOV     BCD_YEAR,A
                MOV     A,LO_YEAR
                SUBB    A,#30H
                MOV     R1,#BCD_YEAR
390             ORL     A,@R1
                MOV     BCD_YEAR,A

        ADJ_TEMP:MOV    A,HI_TEMP
                SUBB    A,#30H
395             MOV     B,#10
                MUL     AB
                MOV     HEX_TEMP,A
                MOV     A,LO_TEMP
                SUBB    A,#30H
400             MOV     R0,#HEX_TEMP
                ADD     A,@R0
                MOV     HEX_TEMP,A

        ADJ_NUM:MOV     A,NUM
405                 SUBB    A,#30H
                MOV     HEX_NUM,A

        ADJ_QUAN:MOV    A,QUAN                  ;ADJUST QUAN
                SUBB    A,#30H
410             MOV     HEX_QUAN,A

        BCD_RFTIME:     MOV     A,HI_REFHOUR1   ;ADJUST REFTIME
                SUBB        A,#30H
                MOV     BCD_HI_RFHOUR1,A
415
                MOV     A,LO_REFHOUR1
                SUBB    A,#30H
                MOV     BCD_LO_RFHOUR1,A
```

```
420                 MOV     A,HI_REFMIN1
                    SUBB    A,#30H
                    MOV     BCD_HI_RFMIN1,A

                    MOV     A,LO_REFMIN1
425                 SUBB    A,#30H
                    MOV     BCD_LO_RFMIN1,A

                    MOV     R0,#HEX_NUM
                    CJNE    @R0,#01H,RFTIME1
430                 SJMP    TO_RTC

        RFTIME1:    MOV     A,HI_REFHOUR2
                    SUBB    A,#30H
                    MOV     BCD_HI_RFHOUR2,A
435
                    MOV     A,LO_REFHOUR2
                    SUBB    A,#30H
                    MOV     BCD_LO_RFHOUR2,A
                    MOV     A,HI_REFMIN2
440                 SUBB    A,#30H
                    MOV     BCD_HI_RFMIN2,A
                    MOV     A,LO_REFMIN2
                    SUBB    A,#30H
                    MOV     BCD_LO_RFMIN2,A
445
                    MOV     R0,#HEX_NUM
                    CJNE    @R0,#02H,RFTIME2
                    SJMP    TO_RTC

450     RFTIME2:    MOV     A,HI_REFHOUR3
                    SUBB    A,#30H
                    MOV     BCD_HI_RFHOUR3,A

                    MOV     A,LO_REFHOUR3
455                 SUBB    A,#30H
                    MOV     BCD_LO_RFHOUR3,A

                    MOV     A,HI_REFMIN3
                    SUBB    A,#30H
460                 MOV     BCD_HI_RFMIN3,A

                    MOV     A,LO_REFMIN3
                    SUBB    A,#30H
                    MOV     BCD_LO_RFMIN3,A
465
        TO_RTC:         CLR CHECK_STATE


470                 MOV     SECONDS,BCD_SEC         ;SECOND
                            MOV     MINUTES,BCD_MIN         ;MINUT
        E
                            MOV     HOURS,BCD_HOUR          ;HOUR
                            MOV     DATE,BCD_DAY            ;DATE
                            MOV     MONTH,BCD_MONTH         ;MONTH
475                         MOV     YEAR,BCD_YEAR           ;YEAR
                    ACALL   RTC_WR
```

```
        ;=================================================;

480     MAIN:       SETB    EA
                    SETB    ES              ;ENABLE INTERRUPT SERI
        AL

        ;=================================================;

485     START_CON:  MOV     DPTR,#PORTB
                    MOVX    A,@DPTR

                    MOV     B,#100
                    MUL     AB
490                 MOV     A,B
                    ANL     A,#00000001B
                    JNZ     O_E
                    MOV     ODD_EVEN,#0
                    JMP     RESULT
495     O_E:        MOV     ODD_EVEN,#1

        RESULT:     MOV     A,B
                    MOV     B,#2
                    DIV     AB
500                 MOV     REAL_TEMP,A
                    MOV     R0,#REAL_TEMP
                    MOV     A,@R0
                    MOV     R0,#HEX_TEMP
                    CLR     C
505                 SUBB    A,@R0
                    JC      OFFSWITCH

        ONSWITCH:   MOV     DPTR,#PORTA
                    MOV     A,#0FFH
510                 MOVX    @DPTR,A
                    LJMP    TIME

        OFFSWITCH:  MOV     DPTR,#PORTA
                    MOV     A,#00H
515                 MOVX    @DPTR,A

                    MOV     R0,#REAL_TEMP
                    MOV     A,@R0
                    MOV     R0,#SET_TEMP
520

        ;=================TIME============================;

        TIME:       ACALL   RTC_RD
525
                    JNB     CHECK_STATE,LOAD_TIME
                    LJMP    TURN_OFF

        LOAD_TIME:  NOP
530
        FOR_TURN_OFF:MOV    R1,#HEX_NUM
                    CJNE    @R1,#0,NEXT_TIME1
                    LJMP    FINISH

535     NEXT_TIME1: MOV     A,HOURS
```

```
                ANL     A,#00110000B
                SWAP    A
                CJNE    A,BCD_HI_RFHOUR1,SKIP1

                MOV     A,HOURS
                ANL     A,#00001111B
                CJNE    A,BCD_LO_RFHOUR1,SKIP1

                MOV     A,MINUTES
                ANL     A,#01110000B
                SWAP    A
                CJNE    A,BCD_HI_RFMIN1,SKIP1

                MOV     A,MINUTES
                ANL     A,#00001111B
                CJNE    A,BCD_LO_RFMIN1,SKIP1

                MOV     A,SECONDS
                ANL     A,#01111111B
                CJNE    A,#0,SKIP1

                SETB    CHECK_STATE
                MOV     A,#0FFH
                MOV     DPTR,#PORTC
                MOVX    @DPTR,A
                JMP     FINISH

        SKIP1:  MOV     R1,#HEX_NUM
                CJNE    @R1,#1,NEXT_TIME2
                LJMP    FINISH

        NEXT_TIME2: MOV     A,HOURS
                ANL     A,#00110000B
                SWAP    A
                MOV     A,BCD_HI_RFHOUR2,SKIP2

                MOV     A,HOURS
                ANL     A,#00001111B
                CJNE    A,BCD_LO_RFHOUR2,SKIP2

                MOV     A,MINUTES
                ANL     A,#01110000B
                SWAP    A
                MOV     A,BCD_HI_RFMIN2,SKIP2

                MOV     A,MINUTES
                ANL     A,#00001111B
                MOV     A,BCD_LO_RFMIN2,SKIP2

                MOV     A,SECONDS
                ANL     A,#01111111B
                CJNE    A,#0,SKIP2

                SETB    CHECK_STATE
                MOV     DPTR,#PORTC
                MOV     A,#0FFH
                MOVX    @DPTR,A
                JMP     FINISH
```

```
        SKIP2:      MOV     R1,#HEX_NUM
                    CJNE    @R1,#2,NEXT_TIME3
                    LJMP    FINISH

600     NEXT_TIME3: MOV     A,HOURS
                    ANL     A,#00110000B
                    SWAP    A
                    CJNE    A,BCD_HI_RFHOUR3,FINISH

605                 MOV     A,HOURS
                    ANL     A,#00001111B
                    CJNE    A,BCD_LO_RFHOUR3,FINISH

                    MOV     A,MINUTES
610                 ANL     A,#01110000B
                    SWAP    A
                    CJNE    A,BCD_HI_RFMIN3,FINISH

                    MOV     A,MINUTES
615                 ANL     A,#00001111B
                    CJNE    A,BCD_LO_RFMIN3,FINISH

                    MOV     A,SECONDS
                    ANL     A,#01111111B
620                 CJNE    A,#0,FINISH

                    SETB    CHECK_STATE
                    MOV     DPTR,#PORTC
                    MOV     A,#0FFH
625                 MOVX    @DPTR,A

        FINISH:     LJMP    START_CON

630     TURN_OFF:   MOV     R0,#HEX_QUAN
                    CJNE    @R0,#1,FOURTY
                    MOV     TIME_ON,#2
                    JMP     COMPARE

635     FOURTY:     CJNE    @R0,#2,FIFTY
                    MOV     TIME_ON,#4
                    JMP     COMPARE

        FIFTY:      MOV     TIME_ON,#5
640
        COMPARE:    MOV     A,SECONDS
                    ANL     A,#01110000B
                    SWAP    A
                    CJNE    A,TIME_ON,GOINGBACK
645                 MOV     DPTR,#PORTC
                    MOV     A,#00H
                    MOVX    @DPTR,A
                    CLR     CHECK_STATE

650     GOINGBACK:  LJMP    START_CON

        ;=================================================

        RTC_WR:     MOV     I2C_ADDR,#RTC_ID
655                 LCALL   I2C_SLAVE
```

```
                MOV     I2C_DATA,#000H
                LCALL   I2C_DATA_WR

660             MOV     I2C_DATA,SECONDS
                LCALL   I2C_DATA_WR

                MOV     I2C_DATA,MINUTES
                LCALL   I2C_DATA_WR
665
                MOV     I2C_DATA,HOURS
                LCALL   I2C_DATA_WR

                MOV     I2C_DATA,DAY
670             LCALL   I2C_DATA_WR

                MOV     I2C_DATA,DATE
                LCALL   I2C_DATA_WR

675             MOV     I2C_DATA,MONTH
                LCALL   I2C_DATA_WR

                MOV     I2C_DATA,YEAR
                LCALL   I2C_DATA_WR
680
                MOV     I2C_DATA,CONTROL
                LCALL   I2C_DATA_WR

                LCALL   I2C_STOP
685             RET

        RTC_RD: MOV     I2C_ADDR,#RTC_ID

                LCALL   I2C_SLAVE
690
                MOV     I2C_DATA,#000H
                LCALL   I2C_DATA_WR

                LCALL   RTC_RD1
695             MOV     SECONDS,I2C_DATA
                LCALL   I2C_NACK_BIT

                LCALL   RTC_RD1
                MOV     MINUTES,I2C_DATA
700             LCALL   I2C_NACK_BIT

                LCALL   RTC_RD1
                MOV     HOURS,I2C_DATA
                LCALL   I2C_NACK_BIT
705
                LCALL   RTC_RD1
                MOV     DAY,I2C_DATA
                LCALL   I2C_NACK_BIT

710             LCALL   RTC_RD1
                MOV     DATE,I2C_DATA
                LCALL   I2C_NACK_BIT

                LCALL   RTC_RD1
715             MOV     MONTH,I2C_DATA
```

```
                    LCALL   I2C_NACK_BIT

                    LCALL   RTC_RD1
                    MOV     YEAR,I2C_DATA
720                 LCALL   I2C_NACK_BIT

                    LCALL   RTC_RD1
                    MOV     CONTROL,I2C_DATA
                    LCALL   I2C_NACK_BIT
725
                    LCALL   I2C_STOP
                    RET

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

        I2C_SLAVE:  PUSH    ACC
735                 SETB    I2C_ACK
                    MOV     A,I2C_ADDR
                    ACALL   I2C_START

                    MOV     R5,#008
740     I2C_SLAVE_1:RLC     A
                    MOV     SDA,C
                    ACALL   I2C_CLK
                    DJNZ    R5,I2C_SLAVE_1

745                 SETB    SDA
                    ACALL   I2C_DELAY
                    SETB    SCL
                    ACALL   I2C_DELAY
                    JB      SDA,I2C_SLAVE_2
750                 CLR     I2C_ACK
        I2C_SLAVE_2:CLR     SCL
                    POP ACC
                    RET

755     I2C_DATA_WR:PUSH    ACC
                    SETB    I2C_ACK
                    MOV     A,I2C_DATA
                    MOV     R5,#008
        I2C_DATA_WR_1:RLC   A
760                 MOV     SDA,C

                    ACALL   I2C_CLK
                    DJNZ    R5,I2C_DATA_WR_1
                    SETB    SDA
765                 ACALL   I2C_DELAY
                    SETB    SCL
                    ACALL   I2C_DELAY
                    JB      SDA,I2C_DATA_WR_2
                    CLR     I2C_ACK
770     I2C_DATA_WR_2:CLR   SCL
                    POP     ACC
                    RET

        I2C_DATA_RD:PUSH    ACC
775                 CLR     A
```

```
                MOV     R5,#008H
I2C_DATA_RD_1:  ACALL   I2C_DELAY
                SETB    SCL
                ACALL   I2C_DELAY
                MOV     C,SDA
                RLC     A
                CLR     SCL
                DJNZ    R5,I2C_DATA_RD_1

                MOV     I2C_DATA,A

                POP     ACC
                RET

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

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

I2C_CLK:        ACALL   I2C_DELAY
                SETB    SCL
                ACALL   I2C_DELAY
                CLR     SCL
                RET

I2C_NACK_BIT:   SETB    SDA
                ACALL   I2C_DELAY
                ACALL   I2C_CLK
                RET

I2C_DELAY:      MOV     R6,#00CH
I2C_DELAY_1:    NOP
                NOP
                DJNZ    R6,I2C_DELAY_1
                RET
;=================================================;

LOAD_DATA:      MOV     DPTR,#0000H     ;LOAD DATA
                MOV     R5,#07H
SE_DATA1_1:     JNB     TI,SE_DATA1_1
                CLR     TI
                MOVX    A,@DPTR
                ACALL   HEX_ASC_1
                MOV     SBUF,A
SE_DATA1_2:     JNB     TI,SE_DATA1_2
                CLR     TI
                MOVX    A,@DPTR
                ACALL   HEX_ASC_2
                MOV     SBUF,A
                INC     DPTR
```

```
                DJNZ    R5,SE_DATA1_1
                RET

        HEX_ASC_1:  MOV     B,#10
840             DIV     AB
                ADD     A,#30H
                RET

        HEX_ASC_2:  MOV     R7,A                ;REAL VALUE
845             MOV     B,#10
                DIV     AB
                MOV     B,#10
                MUL     AB
                MOV     R6,A                ;DEC*10
850             MOV     A,R7
                SUBB    A,R6
                ADD     A,#30H
                RET

855     delay:      mov r6,#02h
        wait2:      mov r7,#0ffh
        wait1:      djnz    r7,wait1
                    djnz    r6,wait2
                    ret
860
        delay_10ms:     mov r7,#010
        delay_10ms_1:   mov r6,#0e6h
        delay_10ms_2:   nop
                        nop
865                     djnz    r6,delay_10ms_2
                        djnz    r7,delay_10ms_1
                        ret

        delay_100ms:    mov r7,#100
870     delay_100ms_1:  mov r6,#0e6h
        delay_100ms_2:  nop
                        nop
                        djnz    r6,delay_100ms_2
                        djnz    r7,delay_100ms_1
875                     ret

        delay_1s:       mov r5,#100
        delay_1s_1:     acall   delay_10ms
                        djnz    r5,delay_1s_1
880                     ret
                        end


885
```

Program ส่วนติดต่อผู้ใช้งานเขียนด้วยภาษาเบสิก

```
Form1
Option Explicit
Public ChooseTime As Integer

Private Sub Next_Click()
If Option4.Value = True Then
ChooseTime = 1
ElseIf Option5.Value = True Then
ChooseTime = 2
ElseIf Option6.Value = True Then
ChooseTime = 3
End If
Form2.Show
End Sub

Private Sub Timer1_Timer()
ShowDate.Caption = Date
ShowTime.Caption = Time
End Sub

Form2
Option Explicit

Private Sub Command1_Click()
Form4.Show
End Sub

Private Sub Form_Load()
Hour1.Text = Format(Hour1.Text, "hh:mm")
Hour2.Text = Format(Hour2.Text, "hh:mm")
Hour3.Text = Format(Hour3.Text, "hh:mm")
If Form1.Option4.Value = True Then
```

```
Label2.Caption = "Feed water  1  time / day"
Label3.Caption = "1st time at"
ElseIf Form1.Option5.Value = True Then
Label2.Caption = "Feed water  2  time / day"
Label3.Caption = "1st time at"
Label4.Caption = "2nd time at"
ElseIf Form1.Option6.Value = True Then
Label2.Caption = "Feed water  3  time / day"
Label3.Caption = "1st time at"
Label4.Caption = "2nd time at"
Label5.Caption = "3rd time at"
End If
End Sub

Form 3
Option Explicit

Private Sub Form_Load()
name4.Caption = Form1.NameOfPlant.Text
TempInput3.Caption = Form1.TempInput.Text
If Form1.Option4.Value = True Then
frist4.Caption = "1st time at   " & Form2.Hour1.Text
TotleTime1.Caption = "1 time / day"
ElseIf Form1.Option5.Value = True Then
frist4.Caption = "1st time at   " & Form2.Hour1.Text
frist5.Caption = "2nd time at   " & Form2.Hour2.Text
TotleTime1.Caption = "2 time / day"
ElseIf Form1.Option6.Value = True Then
frist4.Caption = "1st time at   " & Form2.Hour1.Text
frist5.Caption = "2nd time at   " & Form2.Hour2.Text
frist6.Caption = "3rd time at   " & Form2.Hour3.Text
TotleTime1.Caption = "3 time / day"
End If
```

```
If Form1.Option1.Value = True Then
ShowQuanOfWater.Caption = " 300 cc."
ElseIf Form1.Option2.Value = True Then
ShowQuanOfWater.Caption = "  cc."
ElseIf Form1.Option3.Value = True Then
ShowQuanOfWater.Caption = "  cc."
End If
End Sub

Private Sub Timer1_Timer()
Dim TimeNow As String
ShowTime2.Caption = Format(Time, "hh:mm:ss")
ShowDate2.Caption = Now
TimeNow = ShowTime2.Caption
If Form1.Option4.Value = True Then
   If ShowTime2.Caption > Form2.Hour1.Text Then
      Label7.Caption = "Done"
   Else: Label7.Caption = "Not yet"
   End If
ElseIf Form1.Option5.Value = True Then
   If ShowTime2.Caption > Form2.Hour1.Text Then
      Label7.Caption = "Done"
   Else: Label7.Caption = "Not yet"
   End If
   If ShowTime2.Caption > Form2.Hour2.Text Then
      Label8.Caption = "Done"
   Else: Label8.Caption = "Not yet"
   End If
ElseIf Form1.Option6.Value = True Then
   If ShowTime2.Caption > Form2.Hour1.Text Then
      Label7.Caption = "Done"
   Else: Label7.Caption = "Not yet"
   End If
   If ShowTime2.Caption > Form2.Hour2.Text Then
```

```
        Label8.Caption = "Done"
    Else: Label8.Caption = "Not yet"
    End If
    If ShowTime2.Caption > Form2.Hour3.Text Then
        Label9.Caption = "Done"
    Else: Label9.Caption = "Not yet"
    End If
End If
End Sub

Private Sub Command1_Click()
Form7.Show
End Sub

Form 4
Option Explicit

Private ConfirmToSend As Variant
Private MyhouR As String
Private MyMin As String
Private Mysec As String
Private Myhour1 As String
Private Mymin1 As String
Private MySec1 As String
Private Mydate As String
Private Myyear As String
Private MyDate1 As String
Private Mymonth As String
Private MyMonth1 As String
Public refHour1 As String
Public refMin1 As String
Public refHour2 As String
Public refMin2 As String
Public refHour3 As String
```

```
Public refMin3 As String

Private Sub Form_Load()
MSComm1.Settings = "9600,n,8,1"
MSComm1.InputLen = 0
MSComm1.CommPort = 1
MSComm1.InputMode = comInputModeText
MSComm1.RThreshold = 1
MSComm1.SThreshold = 1
If Not MSComm1.PortOpen Then
MSComm1.PortOpen = True
MsgBox ("Port just Open")
Else: MsgBox ("Port Already Open")
End If
End Sub

Private Sub MSComm1_OnComm()
If MSComm1.CommEvent = 2 And MSComm1.InBufferCount = 1 Then
   ConfirmToSend = MSComm1.Input
   Label15.Caption = "ConfirmToSend is  " & ConfirmToSend
   Call sub_pro
End If
End Sub

Private Sub sub_pro()
If ConfirmToSend = "a" Then
   Call send_DMY
ElseIf ConfirmToSend = "b" Then
   Call send_HMS
ElseIf ConfirmToSend = "c" Then
   Call send_RefTemp
ElseIf ConfirmToSend = "d" Then
   Call send_NumOfFeed
ElseIf ConfirmToSend = "e" Then
```

```
    Call send_QuanOfWater
ElseIf ConfirmToSend = "f" Then
    Call send_refHM
End If
End Sub

Private Sub SendDate_Click()
Dim Myday As String
Myday = Form1.ShowDate.Caption
MyDate1 = Day(Myday)
If MyDate1 = Hex(MyDate1) Then
    Mydate = "0" & MyDate1
    Else: Mydate = MyDate1
End If
MyMonth1 = Month(Myday)
If MyMonth1 = Hex(MyMonth1) Then
    Mymonth = "0" & MyMonth1
    Else: Mymonth = MyMonth1
End If
Myyear = "02"
MSComm1.Output = "A"
Call MSComm1_OnComm
End Sub
Sub send_DMY()
Dim Day As String
Dim Month As String
Dim Year As String
Day = Mydate
Month = Mymonth
Year = Myyear
MSComm1.Output = Day
MSComm1.Output = Month
MSComm1.Output = Year
Label2.Caption = "Send Date  " & Mydate & ":" & Mymonth & ":" & Myyear
```

```
End Sub

Private Sub SendTime_Click()
Dim Mytime As String
Mytime = Form1.ShowTime.Caption
Myhour1 = Hour(Mytime)
If Myhour1 = Hex(Myhour1) Then
   MyhouR = "0" & Myhour1
   Else: MyhouR = Myhour1
   End If
Mymin1 = Minute(Mytime)
If Mymin1 = Hex(Mymin1) Then
   MyMin = "0" & Mymin1
   Else: MyMin = Mymin1
   End If
MySec1 = Second(Mytime)
   If MySec1 = Hex(MySec1) Then
   Mysec = "0" & MySec1
   Else: Mysec = MySec1
   End If
MSComm1.Output = "B"
Call MSComm1_OnComm
End Sub

Sub send_HMS()
Dim Hour As String
Hour = MyhouR & MyMin & Mysec
MSComm1.Output = Hour
Label4.Caption = "Send Time  " & MyhouR & ":" & MyMin & ":" & Mysec
End Sub

Private Sub SendTemp_Click()
MSComm1.Output = "C"
Call MSComm1_OnComm
```

```
End Sub

Sub send_RefTemp()
Dim RefTemp As String
   RefTemp = Form1.TempInput.Text
   MSComm1.Output = RefTemp
   Label6.Caption = "Send Reftemp = " & RefTemp
End Sub

Private Sub SendNum_Click()
MSComm1.Output = "D"
Call MSComm1_OnComm
End Sub

Sub send_NumOfFeed()
If Form1.Option4.Value = True Then
   MSComm1.Output = "1"
   Label8.Caption = "Send num = 1"
ElseIf Form1.Option5.Value = True Then
   MSComm1.Output = "2"
   Label8.Caption = "Send num = 2"
ElseIf Form1.Option6.Value = True Then
   MSComm1.Output = "3"
   Label8.Caption = "Send num = 3"
   End If
End Sub

Private Sub SendQuan_Click()
MSComm1.Output = "E"
Call MSComm1_OnComm
End Sub

Sub send_QuanOfWater()
If Form1.Option1.Value = True Then
```

```
    MSComm1.Output = "1"
    Label10.Caption = "Send QuanOfWater = 1"
  ElseIf Form1.Option2.Value = True Then
    MSComm1.Output = "2"
    Label10.Caption = "Send QuanOfWater = 2"
  ElseIf Form1.Option3.Value = True Then
    MSComm1.Output = "3"
    Label10.Caption = "Send QuanOfWater = 3"
  End If
  End Sub

  Private Sub SendRefTime_Click()
  Dim Refhour11 As String, Refhour21 As String, Refhour31 As String
  Dim RefMin11 As String, RefMin21 As String, RefMin31 As String
  If Form1.ChooseTime = 1 Then
  Refhour11 = Hour(Form2.Hour1.Text)
  RefMin11 = Minute(Form2.Hour1.Text)
    If Refhour11 = Hex(Refhour11) Then
    refHour1 = "0" & Refhour11
    Else: refHour1 = Refhour11
    End If
    If RefMin11 = Hex(RefMin11) Then
    refMin1 = "0" & RefMin11
    Else: refMin1 = RefMin11
    End If
  ElseIf Form1.ChooseTime = 2 Then
  Refhour11 = Hour(Form2.Hour1.Text)
  RefMin11 = Minute(Form2.Hour1.Text)
  Refhour21 = Hour(Form2.Hour2.Text)
  RefMin21 = Minute(Form2.Hour2.Text)
    If Refhour11 = Hex(Refhour11) Then
    refHour1 = "0" & Refhour11
    Else: refHour1 = Refhour11
    End If
```

```
    If RefMin11 = Hex(RefMin11) Then
    refMin1 = "0" & RefMin11
    Else: refMin1 = RefMin11
    End If
    If Refhour21 = Hex(Refhour21) Then
    refHour2 = "0" & Refhour21
    Else: refHour2 = Refhour21
    End If
    If RefMin21 = Hex(RefMin21) Then
    refMin2 = "0" & RefMin21
    Else: refMin2 = RefMin21
    End If
Else
Refhour11 = Hour(Form2.Hour1.Text)
RefMin11 = Minute(Form2.Hour1.Text)
Refhour21 = Hour(Form2.Hour2.Text)
RefMin21 = Minute(Form2.Hour2.Text)
Refhour31 = Hour(Form2.Hour3.Text)
RefMin31 = Minute(Form2.Hour3.Text)
    If Refhour11 = Hex(Refhour11) Then
    refHour1 = "0" & Refhour11
    Else: refHour1 = Refhour11
    End If
    If RefMin11 = Hex(RefMin11) Then
    refMin1 = "0" & RefMin11
    Else: refMin1 = RefMin11
    End If
    If Refhour21 = Hex(Refhour21) Then
    refHour2 = "0" & Refhour21
    Else: refHour2 = Refhour21
    End If
    If RefMin21 = Hex(RefMin21) Then
    refMin2 = "0" & RefMin21
    Else: refMin2 = RefMin21
```

```
        End If
        If Refhour31 = Hex(Refhour31) Then
        refHour3 = "0" & Refhour31
        Else: refHour3 = Refhour31
        End If
        If RefMin31 = Hex(RefMin31) Then
        refMin3 = "0" & RefMin31
        Else: refMin3 = RefMin31
        End If
    End If
    MSComm1.Output = "F"
    Call MSComm1_OnComm
    End Sub
    Sub send_refHM()
    Dim RHM1 As String
    Dim RHM2 As String
    Dim RHM3 As String
    RHM1 = refHour1 & refMin1
    RHM2 = refHour2 & refMin2
    RHM3 = refHour3 & refMin3
    If Form1.ChooseTime = 1 Then
        MSComm1.Output = RHM1
        Label12.Caption = "Send Reftemp1 = " & refHour1 & ":" & refMin1
        Label16.Caption = "Already send all data"
    ElseIf Form1.ChooseTime = 2 Then
        MSComm1.Output = RHM1
        MSComm1.Output = RHM2
        Label12.Caption = "Send Reftemp1 = " & refHour1 & ":" & refMin1
        Label13.Caption = "Send Reftemp2 = " & refHour2 & ":" & refMin2
        Label16.Caption = "Already send all data"
    Else
        MSComm1.Output = RHM1
        MSComm1.Output = RHM2
        MSComm1.Output = RHM3
```

```
    Label12.Caption = "Send Reftemp1 = " & refHour1 & ":" & refMin1
    Label13.Caption = "Send Reftemp2 = " & refHour2 & ":" & refMin2
    Label14.Caption = "Send Reftemp3 = " & refHour3 & ":" & refMin3
    Label16.Caption = "Already send all data"
End If
End Sub

Private Sub Status_Click()
Form3.Show
End Sub
```

Form 5

```
Option Explicit

Private Sub Command1_Click()
Form1.Show
End Sub
```

Form 7

```
Option Explicit
Private DataPerDay As Variant
Private Hour(1 To 12) As String
Private Min(1 To 12) As String
Private tempy(1 To 12) As Integer
Private daty(1 To 12) As Integer
Private plot(1 To 12) As Integer
Private Day As String, Month As String, Year As String
Private FileName As String

Private Sub Load_Click()
Form4.MSComm1.Output = "L"
'Label5.Caption = "Send " & Hex(Asc("L"))
If Form4.MSComm1.CommEvent = 2 And Form4.MSComm1.InBufferCount = 12 Then
DataPerDay = Form4.MSComm1.Input
```

```
Label4.Caption = DataPerDay
End If
End Sub

Private Sub Browse_Click()
CommonDialog1.Filter = "All files (*.*)|*.*|Text files" & "(*.txt)|*.txt"
CommonDialog1.ShowSave
NameTxt.Text = CommonDialog1.FileName
End Sub

Private Sub save_Click()
Dim iNum As Integer
Dim FileName As String
Dim i As Integer
Day = Mid(DataPerDay, 1, 2)
Month = Mid(DataPerDay, 3, 2)
Year = Mid(DataPerDay, 5, 2)
Hour(1) = Mid(DataPerDay, 7, 2)
Min(1) = Mid(DataPerDay, 9, 2)
tempy(1) = Mid(DataPerDay, 11, 2)
Hour(2) = Mid(DataPerDay, 13, 2)
Min(2) = Mid(DataPerDay, 15, 2)
tempy(2) = Mid(DataPerDay, 17, 2)
Hour(3) = Mid(DataPerDay, 19, 2)
Min(3) = Mid(DataPerDay, 21, 2)
tempy(3) = Mid(DataPerDay, 23, 2)
Hour(4) = Mid(DataPerDay, 25, 2)
Min(4) = Mid(DataPerDay, 27, 2)
tempy(4) = Mid(DataPerDay, 29, 2)
Hour(5) = Mid(DataPerDay, 31, 2)
Min(5) = Mid(DataPerDay, 33, 2)
tempy(5) = Mid(DataPerDay, 35, 2)
Hour(6) = Mid(DataPerDay, 37, 2)
Min(6) = Mid(DataPerDay, 39, 2)
```

```
tempy(6) = Mid(DataPerDay, 41, 2)
Hour(7) = Mid(DataPerDay, 43, 2)
Min(7) = Mid(DataPerDay, 45, 2)
tempy(7) = Mid(DataPerDay, 47, 2)
Hour(8) = Mid(DataPerDay, 49, 2)
Min(8) = Mid(DataPerDay, 51, 2)
tempy(8) = Mid(DataPerDay, 53, 2)
Hour(9) = Mid(DataPerDay, 55, 2)
Min(9) = Mid(DataPerDay, 57, 2)
tempy(9) = Mid(DataPerDay, 59, 2)
Hour(10) = Mid(DataPerDay, 61, 2)
Min(10) = Mid(DataPerDay, 63, 2)
tempy(10) = Mid(DataPerDay, 65, 2)
Hour(11) = Mid(DataPerDay, 67, 2)
Min(11) = Mid(DataPerDay, 69, 2)
tempy(11) = Mid(DataPerDay, 71, 2)
Hour(12) = Mid(DataPerDay, 73, 2)
Min(12) = Mid(DataPerDay, 75, 2)
tempy(12) = Mid(DataPerDay, 77, 2)
For i = 1 To 12
plot(i) = tempy(i)
Next i
Form8.Label1.Caption = "Temperature  " & Day & "/" & Month & "/" & Year
iNum = FreeFile
FileName = NameTxt.Text
Open FileName For Append As #iNum
Print #iNum, DataPerDay
Print #iNum, "Temperature at  " & Day & "/" & Month & "/" & Year
Print #iNum, Hour(1) & ":" & Min(1) & "  " & tempy(1) & " Celcius"
Print #iNum, Hour(2) & ":" & Min(2) & "  " & tempy(2) & " Celcius"
Print #iNum, Hour(3) & ":" & Min(3) & "  " & tempy(3) & " Celcius"
Print #iNum, Hour(4) & ":" & Min(4) & "  " & tempy(4) & " Celcius"
Print #iNum, Hour(5) & ":" & Min(5) & "  " & tempy(5) & " Celcius"
Print #iNum, Hour(6) & ":" & Min(6) & "  " & tempy(6) & " Celcius"
```

```
Print #iNum, Hour(7) & ":" & Min(7) & "  " & tempy(7) & " Celcius"
Print #iNum, Hour(8) & ":" & Min(8) & "  " & tempy(8) & " Celcius"
Print #iNum, Hour(9) & ":" & Min(9) & "  " & tempy(9) & " Celcius"
Print #iNum, Hour(10) & ":" & Min(10) & "  " & tempy(10) & " Celcius"
Print #iNum, Hour(11) & ":" & Min(11) & "  " & tempy(11) & " Celcius"
Print #iNum, Hour(12) & ":" & Min(12) & "  " & tempy(12) & " Celcius"
Close #iNum
End Sub

Private Sub Browse2_Click()
CommonDialog1.Filter = "All files (*.*)|*.*|Text files" & "(*.txt)|*.txt"
CommonDialog1.ShowOpen
OpenTxt.Text = CommonDialog1.FileName
End Sub

Private Sub Open_Click()
Dim iNum As Integer
Dim FileName As String
Dim DaTum As String
Dim i As Integer
iNum = FreeFile
FileName = OpenTxt.Text
Open FileName For Input As #iNum
Line Input #iNum, DaTum
Close #iNum
daty(1) = Mid(DaTum, 11, 2)
daty(2) = Mid(DaTum, 17, 2)
daty(3) = Mid(DaTum, 23, 2)
daty(4) = Mid(DaTum, 29, 2)
daty(5) = Mid(DaTum, 35, 2)
daty(6) = Mid(DaTum, 41, 2)
daty(7) = Mid(DaTum, 47, 2)
daty(8) = Mid(DaTum, 53, 2)
daty(9) = Mid(DaTum, 59, 2)
```

```
daty(10) = Mid(DaTum, 65, 2)
For i = 1 To 10
plot(i) = daty(i)
Next i
Form8.Label1.Caption = "Temperature along " & Mid(DaTum, 25, 2) & "/" & Mid(DaTum, 27, 2) & "/" &
Mid(DaTum, 29, 2)
End Sub

Private Sub Graph_Click()
Dim i As Integer
Dim arrValues(1 To 10, 1 To 3)
  arrValues(1, 1) = "p.m.1.00"
  arrValues(2, 1) = "p.m.1.30"
  arrValues(3, 1) = "p.m.2.00"
  arrValues(4, 1) = "p.m.2.30"
  arrValues(5, 1) = "p.m.3.00"
  arrValues(6, 1) = "p.m.3.30"
  arrValues(7, 1) = "p.m.4.00"
  arrValues(8, 1) = "p.m.4.30"
  arrValues(9, 1) = "p.m.5.00"
  arrValues(10, 1) = "p.m.5.30"

For i = 1 To 10
  arrValues(i, 2) = plot(i)
Next i

For i = 1 To 10
  arrValues(i, 3) = Form1.TempInput.Text
Next i

Form8.MSChart1.ChartData = arrValues
Form8.Show
End Sub
```

DALLAS SEMICONDUCTOR

DS1307
64 X 8 Serial Real Time Clock

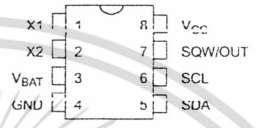
FEATURES

- Real time clock counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap year compensation valid up to 2100
- 56 byte nonvolatile RAM for data storage
- 2–wire serial interface
- Programmable squarewave output signal
- Automatic power fail detect and switch circuitry
- Consumes less than 500 nA in battery backup mode at 25°C
- Optional industrial temperature range –40°C to +85°C (IND)
- Available in 8–pin DIP or SOIC

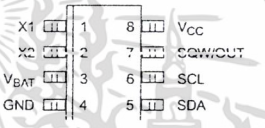
ORDERING INFORMATION

DS1307	Serial Timekeeping Chip; 8–pin DIP
DS1307Z	Serial Timekeeping Chip; 8–pin SOIC (150 mil)
DS1307N	8–pin DIP (IND)
DS1307ZN	8–pin SOIC (IND)

PIN ASSIGNMENT



DS1307 8–PIN DIP (300 MIL)



DS1307Z 8–PIN SOIC (150 MIL)

PIN DESCRIPTION

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

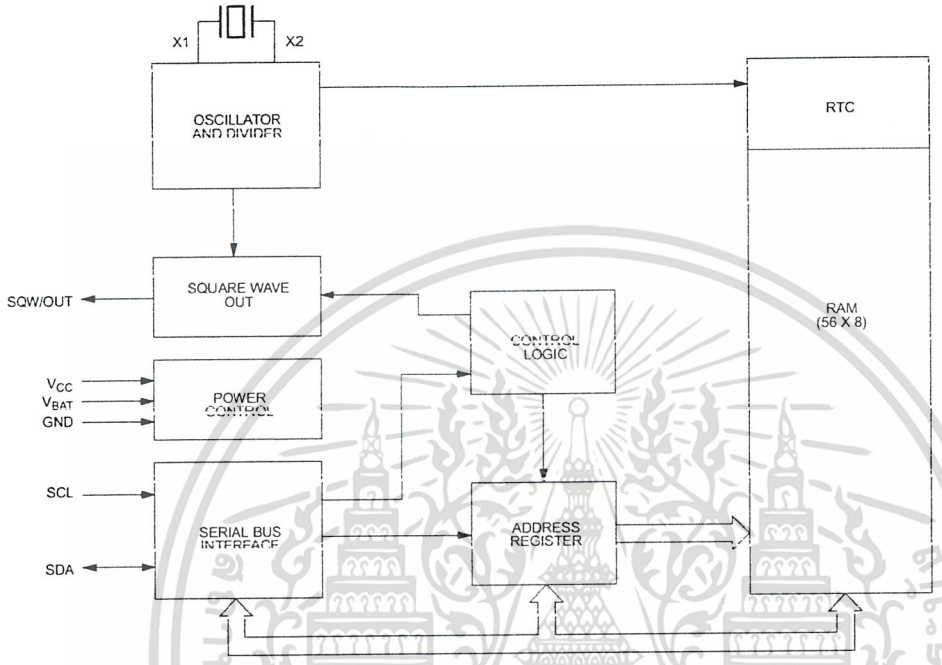
DESCRIPTION

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

OPERATION

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

DS1307 BLOCK DIAGRAM Figure 1



SIGNAL DESCRIPTIONS

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

V_{BAT} – Battery input for any standard 3 volt lithium cell or other energy source. Battery voltage must be held between 2.5 and 3.5 volts for proper operation. The nominal write protect trip point voltage at which access to the real time clock and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A Lithium battery with 35 mAh or greater will back up the DS1307 for more than 10 years in the absence of power.

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

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

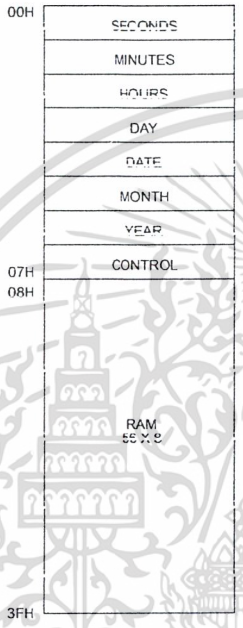
SQW/OUT (Square Wave/ Output Driver) – When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square wave frequencies (1 Hz, 4 KHz, 8 KHz, 32 KHz). The SQW/OUT pin is open drain which requires an external pull–up resistor.

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

RTC AND RAM ADDRESS MAP

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

DS1307 ADDRESS MAP Figure 2



CLOCK AND CALENDAR

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

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

DS1307 TIMEKEEPER REGISTERS Figure 3

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

CONTROL REGISTER

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

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

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

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

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

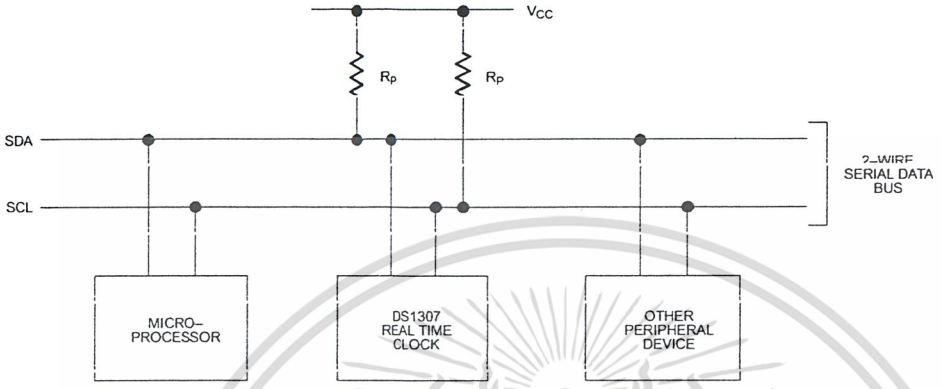
SQUAREWAVE OUTPUT FREQUENCY Table 1

RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1 Hz
0	1	4 KHz
1	0	8 KHz
1	1	32 KHz

2–WIRE SERIAL DATA BUS

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

TYPICAL 2–WIRE BUS CONFIGURATION Figure 4



The following bus protocol has been defined (see Figure 5).

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

Accordingly, the following bus conditions have been defined:

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

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

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

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

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between the START and the STOP conditions is not limited, and is determined by the master device. The information is transferred byte–wise and each receiver acknowledges with a ninth bit.

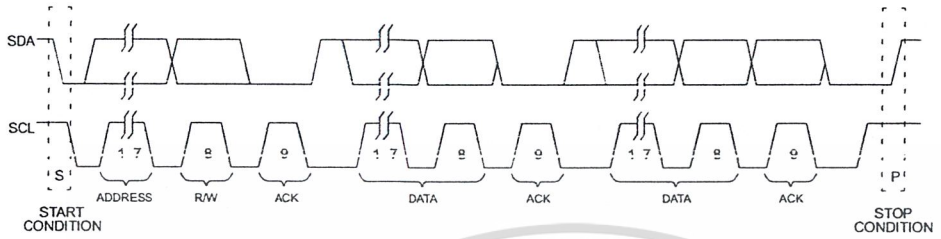
Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

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

DATA TRANSFER

Figures 5, 6, and 7 detail how data transfer is accomplished on the 2–wire bus. Depending on the state of the R/$\overline{W}$ bit in the transmission protocols as shown in Figures 6 and 7, two types of data transfer are possible:

DATA TRANSFER ON 2–WIRE SERIAL BUS Figure 5



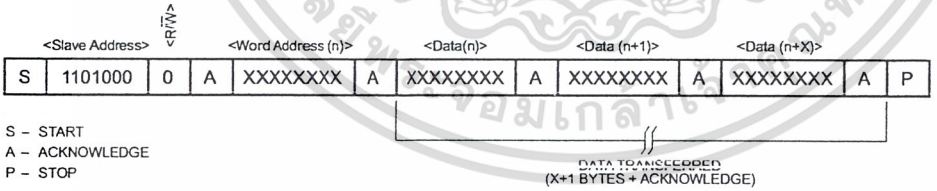
1. Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transfered with the most significant bit (MSB) first.
2. Data transfer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

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

The DS1307 may operate in the following two modes:

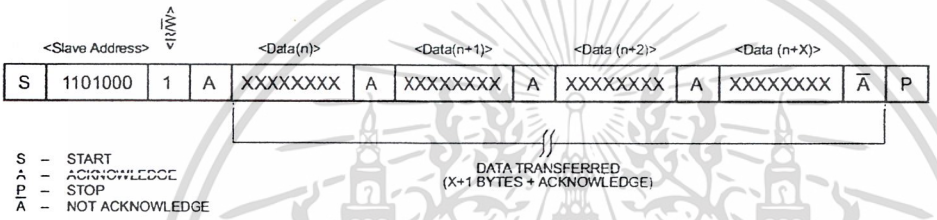
1. Slave receiver mode (DS1307 write mode): Serial data and clock are received through SDA and SCL. After each byte is received an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after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

DATA WRITE – SLAVE RECEIVER MODE Figure 6



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

DATA READ – SLAVE TRANSMITTER MODE Figure 7



ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	–0.5V to +7.0V
Operating Temperature	0°C to 70°C
Storage Temperature	–55°C to +125°C
Soldering Temperature	260°C for 10 seconds

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

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

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

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{CC}	4.5	5.0	5.5	V	1
Logic 1	V_{IH}	2.2		V_{CC}+0.3	V	1
Logic 0	V_{IL}	–0.3		+0.8	V	1
V_{BAT} Battery Voltage	V_{BAT}	2.5		3.5	V	1

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

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

AC ELECTRICAL CHARACTERISTICS

(0°C to 70°C; V_{CC}=4.5V to 5.5V)

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

NOTES:

1. All voltages are referenced to ground.
2. Logic zero voltages are specified at a sink current of 5 mA at V_{CC}=4.5V, V_{OL}=GND for capacitive loads.
3. I_{CCS} specified with V_{CC}=5.0V and SDA, SCL=5.0V.
4. V_{CC}=0V, V_{BAT}=3V.
5. After this period, the first clock pulse is generated.
6. A device must internally provide a hold time of at least 300 ns for the SDA signal (referred to the V_{IHMIN} of the SCL signal) in order to bridge the undefined region of the falling edge of SCL.
7. The maximum $t_{HD:DAT}$ has only to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.
8. C_B – total capacitance of one bus line in pF.
9. I_{CCA} – SCL clocking at max frequency = 100 KHz.
10. SCL only.
11. SDA and SQW/OUT

DALLAS SEMICONDUCTOR

DS275
Line-Powered RS-232 Transceiver Chip

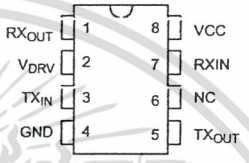
FEATURES

- Low–power serial transmitter/receiver for battery-backed systems
- Transmitter steals power from receive signal line to save power
- Ultra–low static current, even when connected to RS 232 E port
- Variable transmitter level from +5 to +12 volts
- Compatible with RS–232–E signals
- Available in 8–pin, 150–mil wide SOIC package (DS275S) and 14–pin TSSOP package
- Low–power CMOS

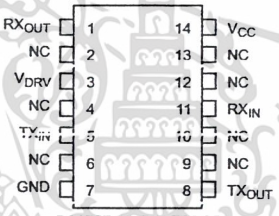
ORDERING INFORMATION

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

PIN ASSIGNMENT



DS275 8–PIN DIP (300 MIL)
DS275S 8–PIN SOIC (150 MIL)



DS275E 14–PIN TSSOP

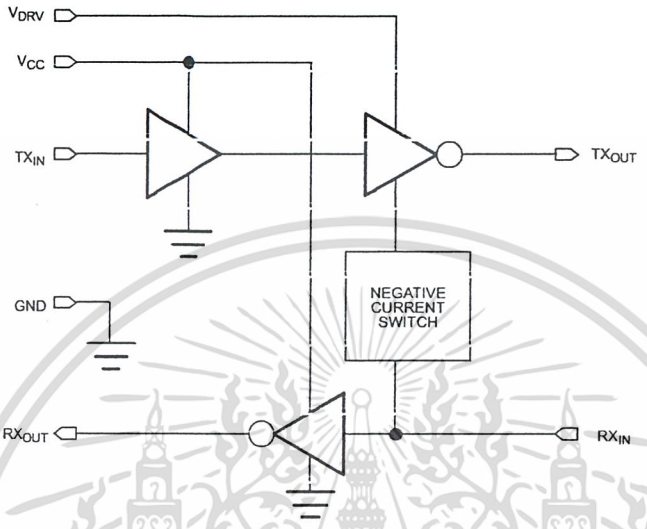
PIN DESCRIPTION

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

DESCRIPTION

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

DS275 BLOCK DIAGRAM Figure 1



OPERATION

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

RECEIVER SECTION

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

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

TRANSMITTER SECTION

TX_{IN} is the CMOS/TTL–compatible input for digital data from the user system. A logic 1 at TX_{IN} produces a mark (negative data signal) at TX_{OUT} while a logic 0 produces a space (positive data signal). As mentioned earlier, the transmitter section employs a unique driver design that uses the RX_{IN} line for swinging to negative levels. The RX_{IN} line must be in a marking or idle state to take advantage of this design; if RX_{IN} is in a spacing state, TX_{OUT} will only swing to ground. When TX_{OUT} needs to transition to a positive level, it uses the V_{DRV} power pin

for this level. V_{DRV} can be a voltage supply between 5 to 12 volts, and in many situations it can be tied directly to the +5 volt V_{CC} supply. *It is important to note that V_{DRV} must be greater than or equal to V_{CC} at all times.*

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

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

RS–232 COMPATIBILITY

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

APPLICATIONS INFORMATION

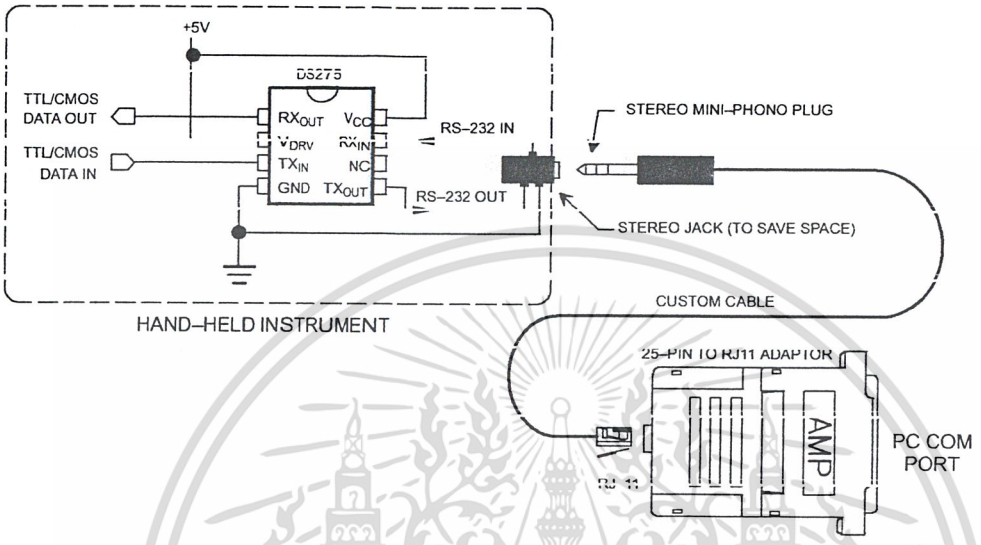
The DS275 is designed as a low–cost, RS–232–E interface expressly tailored for the unique requirements of battery–operated handheld products. As shown in the electrical specifications, the DS275 draws exceptionally low operating and static current. During normal operation when data from the handheld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

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

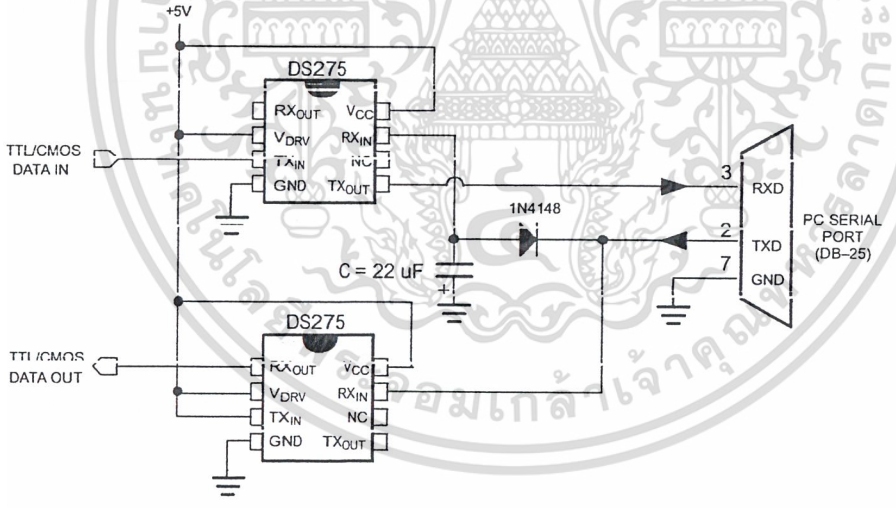
FULL–DUPLEX OPERATION

The DS275 is intended primarily for half–duplex operation; that is, RX_{IN} should remain idle in the marking state when transmitting data out TX_{OUT} and visa versa. However, the part can be operated full–duplex with most RS–232–E serial ports since signals swinging between 0 and +5V will usually be correctly interpreted by an RS–232–E receiver device. The 5–volt swing occurs when TX_{OUT} attempts to swing negative while RX_{IN} is at a positive voltage, which turns on an internal weak pull–down to ground for the TX_{OUT} driver's negative reference. So, transmit mark signals at TX_{OUT} may have voltage jumps from some negative value (corresponding to RX_{IN} marking) to approximately ground. One possible problem that may occur in this case is if the receiver at the other end requires a negative voltage for recognizing a mark. In this situation, the full– duplex circuit shown in Figure 3 can be used as an alternative. The 22 µF capacitor forms a negative–charge reservoir; consequently, when the TXD line is spacing (positive), TX_{OUT} still has a negative source available for a time period determined by the capacitor and the load resistance at the other end (3–7KΩ). This circuit was tested from 150–19,200 bps with error–free operation using a SN75154 Quad Line Receiver as the receiver for the TX_{OUT} signal. Note that the SN75154 can have a marking input threshold below ground; hence there is the need for TX_{OUT} to swing both positive and negative in full–duplex operation with this device.

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



FULL–DUPLEX CIRCUIT USING NEGATIVE–CHARGE STORAGE Figure 3



NOTE:

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

ABSOLUTE MAXIMUM RATINGS*

V_{CC}	–0.3 to +7.0 volts
V_{DRV}	–0.3 to +13.0 volts
RX_{IN}	±15 volts
TX_{IN}	–0.3 to V_{CC} + 0.3 volts
TX_{OUT}	±15 volts
RX_{OUT}	–0.3 to V_{CC} + 0.3 volts
Storage Temperature	–55°C to +125°C
Operating Temperature	0°C to 70°C

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

RECOMMENDED DC OPERATING CONDITIONS

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

DC ELECTRICAL CHARACTERISTICS

(0°C to 70°C; $V_{CC} = V_{DRV} = 5V \pm 10\%$)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
TX_{OUT} Level High	V_{OTXH}	3.5	4.0	5.0	V	6
TX_{OUT} Level Low	V_{OTXL}	–8.5	–9.0		V	7
TX_{OUT} Short Circuit Current	I_{SC}		+20	+85	mA	
TX_{OUT} Output Slew Rate	i_{SR}			30	V/µs	
Propagation Delay	t_{PD}		5		µs	8
RX_{IN} Input Threshold Low	V_{TL}	0.8	1.2	1.6	V	
RX_{IN} Input Threshold High	V_{TH}	1.6	2.0	2.4	V	
RX_{IN} Threshold Hysteresis	V_{HYS}	0.5	0.8		V	9
RX_{OUT} Output Current @ 2.4 V	I_{OH}	–1.0			mA	
RX_{OUT} Output Current @ 0.4 V	I_{OL}			3.2	mA	

NOTES:

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