**ภาควิชาครุศาสตร์วิศวกรรม**
**คณะครุศาสตร์อุตสาหกรรม**
**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**
**ใบรับรองปริญญานิพนธ์**

**ชื่อหัวข้อ** ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ
Energy Consumption Record and Control System for Split Type Air Condition

**ชื่อนักศึกษา**
1. นายนพดล โปธิ รหัสประจำตัว 48035498
2. นายปรีดา พรมบุตร รหัสประจำตัว 48035504
3. นายวิทยายุทธ ผะดาวงศ์ รหัสประจำตัว 48035544

**หลักสูตร** ครุศาสตร์อุตสาหกรรมบัณฑิต
**สาขาวิชา** วิศวกรรมอิเล็กทรอนิกส์
**อาจารย์ที่ปรึกษา** อ.โกศล ดราชู
**อาจารย์ที่ปรึกษาร่วม** อ.อมรชัย ชัยชนะ

| คณะกรรมการสอบปริญญานิพนธ์ | ลายมือชื่อ |
|---|---|
| 1. ผศ.สุชิน อาจหาญ | |
| 2. อ.โกศล ดราชู | |
| 3. อ.สุขสันต์ พาณิชพาพิบูล | |
| 4. อ.ประเสริฐ เคนพันค้อ | |
| 5. | |

**วัน/เดือน/ปีที่สอบ** วันพุธที่ 9 เดือนพฤษภาคม พ.ศ. 2550 เวลา 12.00 น.
**สถานที่สอบ** ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(รศ.สุรสิทธิ์ ราดรี)
หัวหน้าภาควิชาครุศาสตร์วิศวกรรม
วันที่ 30 เดือน พ.ค. พ.ศ. 50

<BT492432>
ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ

# ปริญญานิพนธ์

## ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ

## ENERGY CONSUMPTION RECORD AND CONTROL SYSTEM FOR SPLIT TYPE AIR CONDITION

นพดล โปธิ
ปรีดา พรมบุตร
วิทยายุทธ ผะดาวงค์



ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

# ปริญญานิพนธ์

**เรื่อง ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ**

**Energy Consumption Record and Control System for Split Type Air Condition**

**วัตถุประสงค์**

1. เพื่อศึกษาหลักการระบบบันทึกและการควบคุมการใช้พลังงานของระบบปรับอากาศ
2. เพื่อพัฒนาระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ
3. เพื่อสร้างระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ
4. เพื่อทดลองระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ
5. เพื่อนำระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศไปใช้งานจริง

**ประโยชน์ที่คาดว่าจะได้รับ**

1. ได้รับความรู้เกี่ยวกับหลักการของระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ
2. ได้แบบระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ
3. ได้เครื่องต้นแบบระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ
4. ได้ผลการทดลองการทำงานระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศรุ่นที่ 3
5. ได้ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ รุ่นที่ 3 มาใช้งาน

| | | |
|---|---|---|
| **ชื่อหัวข้อ** | ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ | |
| **นักศึกษา** | นายนพดล | โปธิ |
| | นายปรีดา | พรมบุตร |
| | นายวิทยายุทธ | ผะดาวงค์ |
| **อาจารย์ที่ปรึกษา** | อาจารย์โกศล | ตราชู |
| **อาจารย์ที่ปรึกษาร่วม** | อาจารย์อมรชัย | ชัยชนะ |
| **หลักสูตร** | ครุศาสตร์อุตสาหกรรมบัณฑิต | |
| **สาขาวิชา** | วิศวกรรมอิเล็กทรอนิกส์ | |
| **ปีการศึกษา** | 2549 | |


**บทคัดย่อ**

ปริญญานิพนธ์ฉบับนี้นำเสนอการพัฒนา และการสร้าง ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ โดยระบบนี้ประกอบด้วย ชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ ชุดแปลงสัญญาณ RS232/RS485 และภาคแสดงผลบนเครื่องคอมพิวเตอร์ ซึ่งสามารถแสดงผลเป็นตัวเลข และค่าอุณหภูมิที่เวลาจริงและที่บันทึกไว้ ระบบนี้สามารถช่วยให้การใช้พลังงานของระบบปรับอากาศถูกควบคุมให้อยู่ในปริมาณที่เหมาะสม โดยไม่ส่งผลกระทบต่อพฤติกรรมการใช้งานตามปกติ

| | |
|---|---|
| **Thesis Title** | Energy Consumption Record and Control System for Split Type Air Condition |
| **Students** | Mr.Noppadon Poti |
| | Mr.Preeda Promboot |
| | Mr.Vittayayoot Phadawong |
| **Advisor** | Mr.Koson Trachu |
| **Co-Advisor** | Mr.Amonchai Chaichana |
| **Education Level** | Bachelor of Science in Industrial Education |
| **Program in** | Electronics Engineering |
| **Academic Year** | 2006 |



**ABSTRACT**

This thesis presented a develop and development of Energy Consumption Record and Control System for Split Type Air condition. The system consisted of the measure, the control air condition, the RS232/RS485 converter and the display on computer. Which can show the number, graph and temperature at real time This system control energy of split type air condition with quantity properly and don't have effect with action.

# กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ลุล่วงไปได้ด้วยดี เนื่องมาจากความร่วมมือของสมาชิกภายในกลุ่มทุกคน ขอขอบคุณอาจารย์โกศล ตราชู อาจารย์อมรชัย ชัยชนะ คณาจารย์ รุ่นพี่ทุกท่านและเจ้าหน้าที่ภาควิชาครุศาสตร์วิศวกรรมทุกท่านที่ให้ความอนุเคราะห์ให้คำแนะนำ แนวความคิด ความรู้ต่างๆ แนวทางการแก้ปัญหาในการจัดทำปริญญานิพนธ์ ขอขอบคุณห้องสมุดคณะครุศาสตร์อุตสาหกรรม สำนักหอสมุดกลาง และเจ้าหน้าที่ห้องปริญญานิพนธ์คณะครุศาสตร์อุตสาหกรรม ที่ช่วยอำนวยความสะดวกและเอื้อเฟื้อสถานที่ในการค้นคว้าข้อมูล สุดท้ายที่สำคัญควรแก่การระลึกถึงอย่างยิ่ง บิดา มารดา และผู้มีพระคุณที่เป็นผู้ให้การสนับสนุนด้านการศึกษาและเป็นผู้ให้กำลังใจเมื่อยามรู้สึกท้อแท้ ให้กลับรู้สึกดีขึ้นอีกครั้ง ตั้งแต่อดีตจนถึงปัจจุบัน

# สารบัญ

# สารบัญ (ต่อ)

# สารบัญ (ต่อ)

# สารบัญตาราง

# สารบัญรูป

# สารบัญรูป (ต่อ)

# สารบัญรูป (ต่อ)

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาและความสำคัญ

ปัจจุบันระดับการใช้พลังงานไฟฟ้าภายในประเทศได้เพิ่มปริมาณสูงขึ้น โดยเฉพาะในอาคารเรียนทั่วไปจากการสังเกตการณ์ใช้พลังงานไฟฟ้าภายในห้องเรียนเพราะปกติพฤติกรรมของผู้ใช้จะเปิดใช้งานเครื่องปรับอากาศพร้อมกันหลายเครื่อง ในเวลาเรียนชั่วโมงแรก ซึ่งจะส่งผลให้ระดับของพลังงานไฟฟ้าที่ใช้สูงกว่าระดับปกติที่การไฟฟ้ากำหนดไว้ จึงทำให้ต้องเสียค่าใช้จ่ายเพิ่มขึ้นจากระดับปกติ และการปรับตั้งค่าอุณหภูมิของเครื่องปรับอากาศต่ำกว่า 25 องศาเซลเซียส รวมทั้งการที่ผู้ใช้เปิดเครื่องปรับอากาศทำงานผิดปกติ ส่งผลให้เกิดการสิ้นเปลืองพลังงานไฟฟ้า และระบบบันทึกและการควบคุมการใช้พลังงานของระบบปรับอากาศ รุ่นที่ 1 ยังไม่มีเสถียรภาพโดยที่ยังไม่สามารถควบคุมด้วยการสื่อสารผ่าน RS 485 ได้

จากปัญหาดังกล่าว จึงเสนอระบบบันทึกและการควบคุมการใช้พลังงานของระบบปรับอากาศ ที่มีความเสถียรภาพทางด้านวงจรและสามารถควบคุมด้วยการสื่อสารผ่าน RS 485 ได้ สามารถตรวจวัดอุณหภูมิ กระแส และแรงดัน ในห้องที่ติดตั้งระบบไว้ โดยไม่ต้องตรวจวัดที่เครื่องวัดอุณหภูมิโดยตรง เป็นการประหยัดเวลา มีความสะดวกในการทำงาน การเก็บข้อมูลก็สามารถทำได้ง่ายและสามารถดูค่าอุณหภูมิย้อนหลังได้

## 1.2 ขีดความสามารถของโครงงาน

โครงงานนี้มีขีดความสามารถดังนี้

1. สามารถส่งข้อมูลผ่านระบบสื่อสารแบบอนุกรม RS 485 ได้
2. สามารถปิดคอมเพรสเซอร์เมื่อตรวจสอบพบว่าไม่มีประสิทธิภาพในการทำงาน

## 1.3 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้แบ่งออกเป็นบทต่างๆ เพื่อสะดวกต่อการศึกษาและการทำความเข้าใจ โดยในแต่ละบทจะประกอบด้วยเนื้อหาดังต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปริญญานิพนธ์ ขีดความสามารถของโครงงานและเนื้อหาในบทต่างๆ โดยสังเขป

บทที่ 2 ประกอบด้วยทฤษฎีต่างๆ เกี่ยวกับ เครื่องปรับอากาศแบบแยกส่วนโวลท์มิเตอร์กระแสสลับ แอมมิเตอร์กระแสสลับแบบแคลมพ์ออน วงจรเปลี่ยนแรงดันไฟฟ้าเป็นกระแสไฟฟ้า อุปกรณ์ที่ใช้ในการตรวจวัดอุณหภูมิ การรับส่งข้อมูลแบบอนุกรม การอินเทอร์รัพท์ และการเขียนโปรแกรมด้วยภาษาซี

บทที่ 3 กล่าวถึงเนื้อหาที่เกี่ยวกับแผนผังการทำงานของโครงงาน ผังวงจรต่างๆ ที่ใช้ในโครงสร้างตลอดจนการออกแบบและการสร้างส่วนประกอบต่างๆ เช่นวงจร โวลท์มิเตอร์ แอมป์มิเตอร์ วงจรตรวจวัดอุณหภูมิ วงจรควบคุมการเปิด-ปิดระบบปรับอากาศแบบแยกส่วน และการโปรแกรมควบคุมและประมวลค่าแรงดัน กระแส และอุณหภูมิของระบบปรับอากาศแบบแยกส่วน พร้อมทั้งการทำงานของส่วนประกอบต่างๆ โดยละเอียด

บทที่ 4 ประกอบด้วยการทดลองและผลการทดลองต่างๆ เช่น การทดลองการแสดงผลค่าแรงดัน กระแส และอุณหภูมิ การทดลองควบคุมการทำงานของระบบปรับอากาศแบบแยกส่วน

บทที่ 5 เป็นการสรุปผลการจัดทำโครงงาน ปัญหาที่เกิดขึ้นและแนวทางในการแก้ไขรวมทั้งแนวทางในการพัฒนา

ภาคผนวก ก แสดงภาพ.เครื่องต้นแบบ การติดตั้ง การเชื่อมต่อกับอุปกรณ์อื่นๆ ขณะใช้งานจริง

ภาคผนวก ข ประกอบด้วยผังรายละเอียดวงจรและวงจรพิมพ์

ภาคผนวก ค แสดงรายการอุปกรณ์ที่ใช้งานในแต่ละวงจร

ภาคผนวก ง แสดงรายละเอียดและคุณสมบัติของอุปกรณ์ที่ใช้ในโครงงาน

ภาคผนวก จ แสดงแผนผังการทำงาน

ภาคผนวก ฉ รหัสต้นฉบับของโปรแกรมทั้งหมดที่สร้างขึ้นเพื่อประกอบการทำงานของโครงงาน

ภาคผนวก ช เป็นคู่มือการใช้งานโครงงานระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ

# บทที่ 2
# ทฤษฎีและหลักการ

## 2.1 กล่าวนำ

ในบทนี้จะกล่าวถึงทฤษฎีพื้นฐานต่างๆ ที่นำมาใช้ประกอบในการจัดทำระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศแบบแยกส่วน ซึ่งเนื้อหาต่างๆ จะได้กล่าวถึงต่อไป

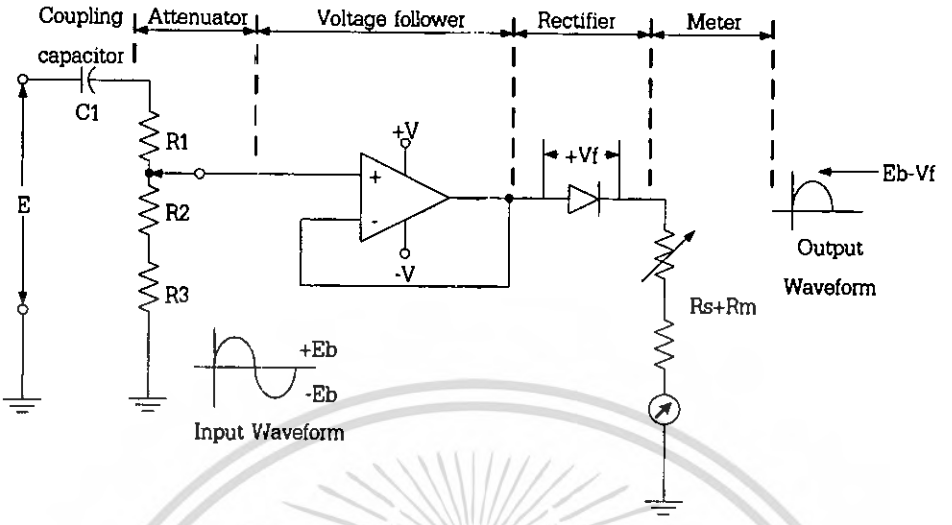
## 2.2 เครื่องปรับอากาศแบบแยกส่วน

เครื่องปรับอากาศแบบแยกส่วนเป็นแบบที่นิยมใช้กันมากตามบ้านพักอาศัยและสำนักงานในปัจจุบัน เพราะเสียงเงียบกว่าและการติดตั้งสะดวกกว่าเนื่องจากไม่ต้องรื้อหน้าต่างออกเช่นเดียวกับแบบติดหน้าต่าง เพียงแต่เจาะผนังเป็นรูสำหรับร้อยท่อชักชั่น ท่อลิควิด และสายไฟ เพียงเล็กน้อยเท่านั้น เครื่องปรับอากาศแบบแยกส่วนนี้จะแบ่งระบบวงจรน้ำยาเครื่องออกเป็น 2 ส่วนคือ ชุดคอยล์น้ำเย็นหรืออีวาพอเรเตอร์และชุดคอนเดนซิ่งยูนิต

## 2.3 เครื่องมือวัด

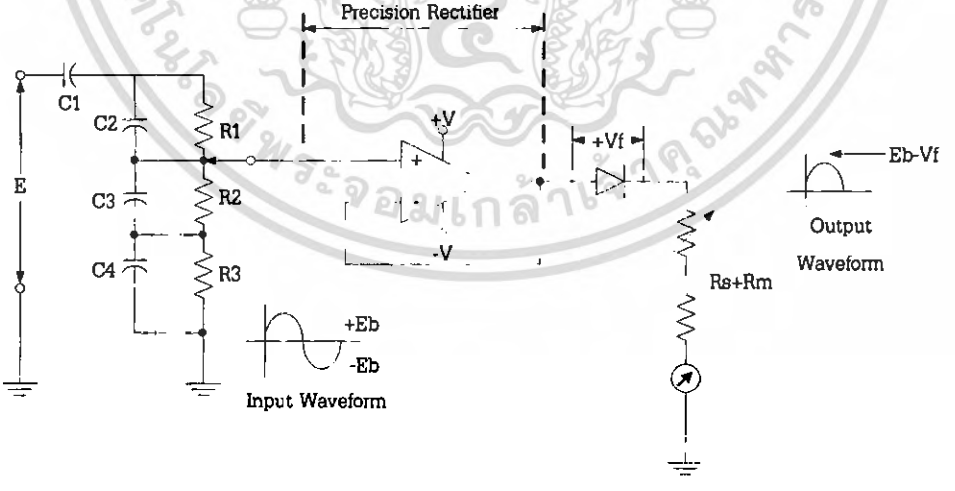
### 2.3.1 โวลต์มิเตอร์กระแสสลับอิเล็กทรอนิกส์

จากดีซีอิเล็กทรอนิกส์โวลต์มิเตอร์แบบวงจรขยายตามแรงดันไฟฟ้าจะทำให้เป็นวงจรโวลต์มิเตอร์กระแสสลับด้วยการต่อไดโอดอนุกรมกับขดลวดหมุนดังรูปที่ 2.7 จะเป็นโวลต์มิเตอร์กระแสสลับแบบเรียงกระแสไฟฟ้าครึ่งคลื่น เมื่อป้อนแรงดันไฟฟ้ากระแสสลับ แรงดัน E จะผ่าน $C_1$ ที่มีหน้าที่กัน (Block) ไม่ให้แรงดันไฟฟ้ากระแสตรงผ่านเข้ามา แรงดันไฟฟ้ากระแสสลับ แรงดัน E จะถูกลดทอนลงเป็นแรงดันไฟฟ้าอินพุต $E_B$ ป้อนให้ออปแอมป์ซึ่งเป็นแรงดันไฟฟ้าเอาต์พุตของออปแอมป์จะมีค่าเท่ากับแรงดันไฟฟ้าอินพุต $E_B$ การจัดวงจรเช่นนี้จะมีปัญหาค่าผิดพลาดจากแรงดันไฟฟ้า $V_F$ ของไดโอดเพราะว่าแรงดันไฟฟ้า $V_O$ ที่ตกคร่อมมิเตอร์จะมีค่าเท่ากับ $E_B - V_F = E_B - 0.7$ V ถ้า $E_B < 0.7$ V จะไม่สามารถวัดแรงดันค่าต่ำๆ ได้เลย



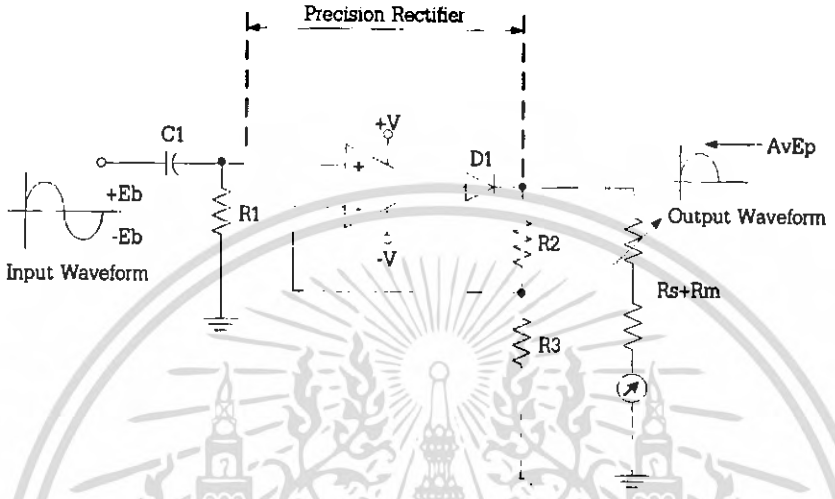

**รูปที่ 2.1** โวลต์มิเตอร์กระแสสลับแบบออปแอมป์เป็นวงจรขยายตามแรงดันไฟฟ้า

การป้องกันไม่ให้เกิดค่าผิดพลาดจากแรงดันไฟฟ้า Vf แก้ไขด้วยการย้ายจุดต่อสายสัญญาณการป้อนกลับจากเอาต์พุตของออปแอมป์เปลี่ยนมาต่อที่ขาแคโธดของไดโอดดังรูปที่ 2.8 จะได้แรงดันไฟฟ้า $V_o = Eb$ ตกคร่อมที่มิเตอร์จะมีความถูกต้องจากการวัดจึงเรียกว่าวงจรเรียงกระแสไฟฟ้าที่มีความเที่ยงตรง (Precision Rectifier) สังเกตว่ามี C2,C3,C4 ต่อคร่อมตัวต้านทานของวงจรลดทอนเพื่อชดเชยให้สัญญาณอินพุตไม่ให้ผิดเพี้ยนเรียกว่าคาปาซิเตอร์ชดเชย (Compensation Capacitor)



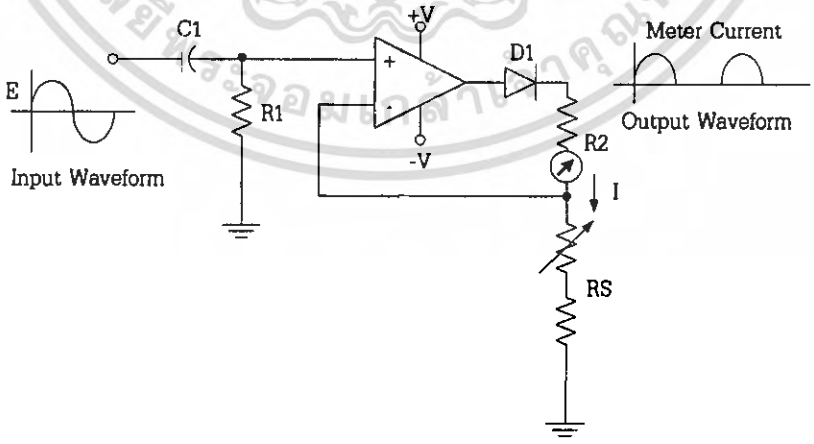

**รูปที่ 2.2** โวลต์มิเตอร์กระแสสลับแบบเรียงกระแสไฟฟ้าครึ่งคลื่นที่มีความเที่ยงตรง

วงจรรูปที่ 2.3 เป็นโวลต์มิเตอร์กระแสสลับที่มีอัตราขยายแรงดันไฟฟ้า $A_V = \frac{R2}{R3} + 1$ แรงดันไฟฟ้าอินพุต $E_P$ ป้อนให้วงจรได้ $V_O = A_V E_P$ ตกคร่อมมิเตอร์



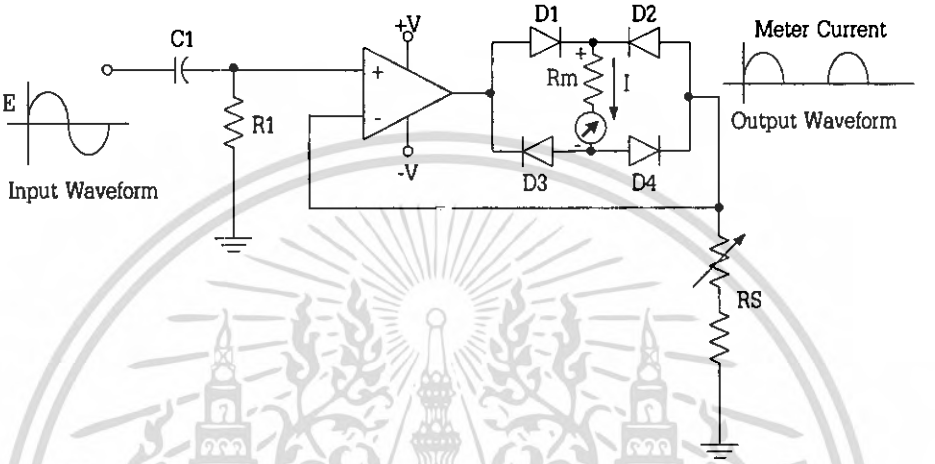

**รูปที่ 2.3** โวลต์มิเตอร์กระแสสลับแบบเรียงกระแสไฟฟ้าครึ่งคลื่นที่มีความเที่ยงตรง

วงจรรูปที่ 2.10 เป็นวงจรโวลต์มิเตอร์กระแสสลับที่เปลี่ยนแรงดันไฟฟ้าเป็นกระแสไฟฟ้าต่อขดลวดหมุนแทน $R_2$ เมื่อแรงดันไฟฟ้าอินพุตครึ่งไซเกิลบวก $E_P$ ถูกป้อนเข้ามาจะทำให้ไดโอดได้รับแรงดันไฟฟ้าไบอัสตรงจึงมีกระแสไฟฟ้า $I_P$ ไหลผ่าน $R_3$ จะได้ $I_P = \frac{E_P}{R_3}$ และกระแสไฟฟ้าเฉลี่ย $I_{av} = 0.318 I_P$



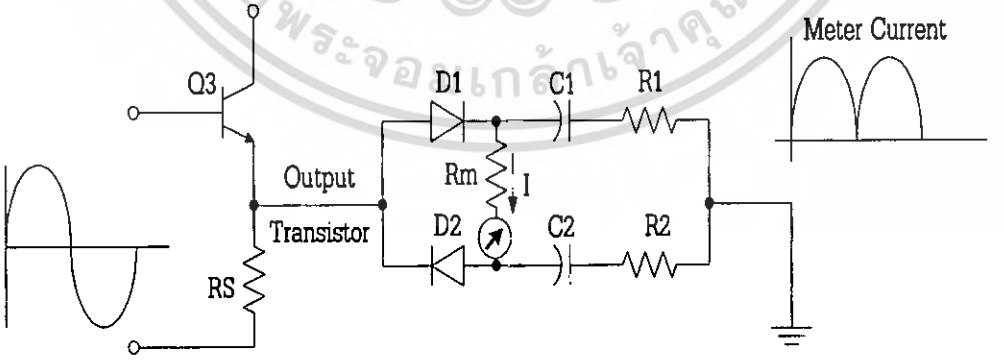

**รูปที่ 2.4** โวลต์มิเตอร์กระแสสลับแบบเปลี่ยนแรงดันไฟฟ้าเป็นกระแสไฟฟ้า

วงจรรูปที่ 2.5 เป็นโวลต์มิเตอร์กระแสสลับเรียงกระแสไฟฟ้าเต็มคลื่นแบบบริดจ์ โดยที่ไดโอด $D_1$, $D_4$ จะได้รับแรงดันไฟฟ้าไบอัสตรงขณะที่แรงดันไฟฟ้าอินพุตครึ่งไซเกิลบวกถูกป้อนเข้ามาและ $D_2$, $D_3$ จะได้รับแรงดันไฟฟ้าไบอัสตรงขณะที่แรงดันไฟฟ้าอินพุตครึ่งไซเกิลลบถูกป้อนเข้ามาจะได้กระแสไฟฟ้าเฉลี่ย $I_{av} = 0.636I_P$




**รูปที่ 2.5** โวลต์มิเตอร์กระแสสลับเรียงกระแสไฟฟ้าเต็มคลื่นแบบบริดจ์

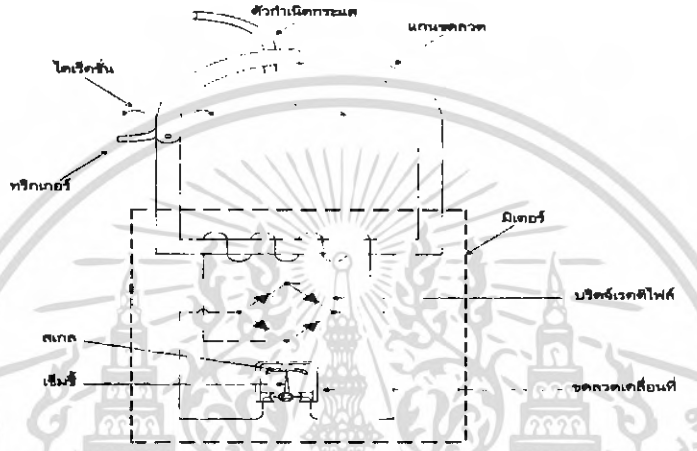
ส่วนรูปที่ 2.12 โวลต์มิเตอร์กระแสสลับที่เรียงกระแสไฟฟ้าเต็มคลื่นแบบครึ่งบริดจ์โดยจะใช้ไดโอด $D_1$, $D_2$ เพียง 2 ตัวและมีทรานซิสเตอร์ $Q_3$ จ่ายกระแสไฟฟ้าให้ $R_s$ ได้แรงดันไฟฟ้าป้อนให้วงจรเรียงกระแสไฟฟ้าแบบครึ่งคลื่น นอกจากนี้ $C_1$, $C_2$ มีไว้เพื่อกันกระแสไฟฟ้ากระแสตรงจากมิเตอร์และให้กระแสไฟฟ้ากระแสสลับผ่านไปได้




**รูปที่ 2.6** โวลต์มิเตอร์กระแสสลับเรียงกระแสไฟฟ้าเต็มคลื่นแบบครึ่งบริดจ์

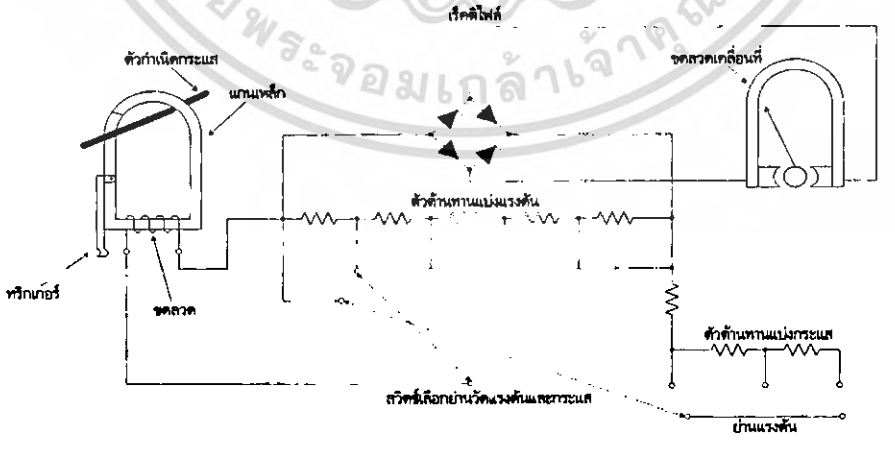
### 2.3.2 แอมมิเตอร์กระแสสลับแบบแคลมพ์ออน

แอมมิเตอร์กระแสสลับแบบแคลมพ์ออนดังรูปที่ 2.7 จะทำงานเหมือนกันกับหม้อแปลงกระแสไฟฟ้า โดยมีขดลวดปฐมภูมินำไปคล้องเข้ากับสายไฟฟ้า ทำให้เกิดการเหนี่ยวนำได้ กระแสไฟฟ้าไหลในขดลวดทุติยภูมิ โดยกระแสไฟฟ้านี้จะเป็นสัดส่วนกับกระแสไฟฟ้าที่ขดลวดปฐมภูมิ และนำไปป้อนให้วงจรเรียงกระแสเพื่อเปลี่ยนไฟฟ้ากระแสสลับเป็นไฟฟ้ากระแสตรงป้อนให้ขดลวดเคลื่อนที่




**รูปที่ 2.7** โครงสร้างของแอมมิเตอร์กระแสสลับแบบแคลมพ์ออนมิเตอร์

โดยปกติแล้วการวัดกระแสไฟฟ้าจะต้องต่อแอมมิเตอร์อนุกรมกับโหลด ถ้าวัดค่ากระแสไฟฟ้าสูงๆ แล้วจะมีอันตรายและไม่สะดวกในการปฏิบัติ จึงใช้แคลมพ์ออนมิเตอร์วัดกระแสไฟฟ้าแทน แคลมพ์ออนมิเตอร์จะมี 3 ส่วนคือโอห์มมิเตอร์ แอมมิเตอร์และโวลต์มิเตอร์ จะมีสวิตช์เปลี่ยนย่านวัดเหมือนทั่วไป ดังรูปที่ 2.7 และ 2.8




**รูปที่ 2.8** วงจรของแคลมพ์ออนโวลต์ – แอมมิเตอร์

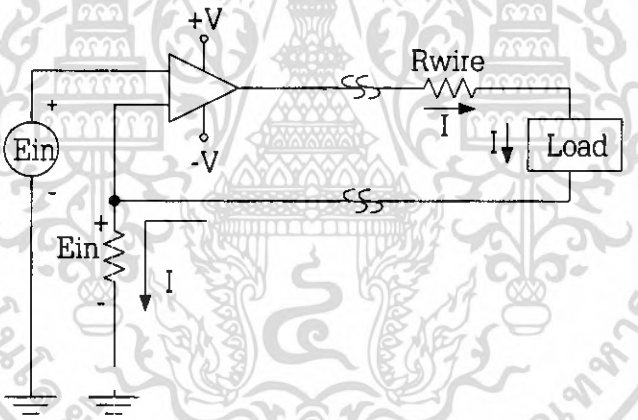
### 2.3.3 วงจรเปลี่ยนแรงดันไฟฟ้าเป็นกระแสไฟฟ้า

การส่งสัญญาณแรงดันไฟฟ้ามีปัญหาเรื่องอุณหภูมิ จุดต่อไม่ดี (Poor Connection) ความต้านทานสาย (Line Resistance) ที่เกิดระหว่างเอาต์พุตของวงจรกับโหลด การสูญเสียภายในสายเพียง 2-3 mV นั้น จะทำให้ค่าความผิดพลาดจากการวัดสัญญาณ ดังนั้นจึงเปลี่ยนให้ส่งเป็นสัญญาณกระแสไฟฟ้าแทนเพราะไม่มีปัญหาการสูญเสียในสายและจุดต่อ สัญญาณรบกวนภายในสายนำสัญญาณลดน้อยลง ทำให้มีประสิทธิภาพของการทำงานดีขึ้น

#### 2.3.3.1 วงจรเปลี่ยนแรงดันไฟฟ้าเป็นกระแสไฟฟ้าแบบโหลดต่อกราวด์

วงจรเปลี่ยนแรงดันไฟฟ้าเป็นกระแสไฟฟ้าแบบโหลดต่อกราวด์ การใช้งานขึ้นอยู่กับการนำค่าที่วัดนั้นมาแสดงผลหรือการควบคุมด้วยวงจรอิเล็กทรอนิกส์ที่ต้องการ โดยจะมีหลักการทำงานของวงจรดังนี้

วงจรขยายความแตกต่างจะใช้ในวงจรแบบโหลดต่อกราวด์เพื่อจ่ายกระแสไฟฟ้าให้โหลดดังรูปที่ 2.9 ตัวต้านทาน $R_1, R_2, R_3$ และ $R_4$ จะมีค่าเท่ากัน จะทำให้อัตราขยายเท่ากับ 1 ตัวต้านทาน $R_s$ ต่อระหว่างเอาต์พุตกับโหลด ส่วน $R_4$ จะไม่ต่อลงกราวด์แต่จะต่อที่ด้านบนของโหลด




**รูปที่ 2.9** วงจรเปลี่ยนแรงดันไฟฟ้าเป็นกระแสไฟฟ้าแบบโหลดต่อกราวด์

**สมการการออกแบบวงจร**

หา $V_{out}$

$$V_{out} = V_L + e_2 - e_1 \tag{2.1}$$

หา $V_{RS}$

แรงดันไฟฟ้าเอาต์พุตของวงจรจะตกคร่อม $R_S$

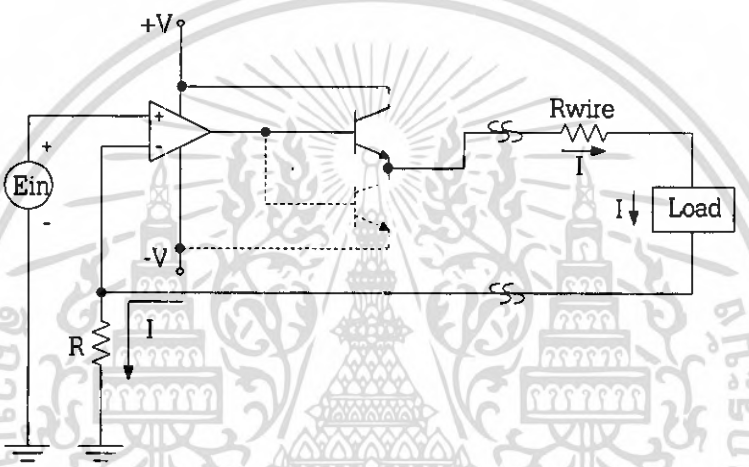
$$V_{RS} = V_{out} - V_L = (V_L + e_2 - e_1) - V_L$$

$$V_{RS} = e_2 - e_1 \tag{2.2}$$

หากระแสไฟฟ้าไหลผ่าน $R_s$

$$I_{RS} = \frac{V_{RS}}{R_S}$$

$$I_L = I_{RS} = \frac{e_2 - e_1}{R_S} \tag{2.3}$$




**รูปที่ 2.10** วงจรเปลี่ยนแรงดันไฟฟ้าเป็นกระแสไฟฟ้าแบบโหลดต่อกราวด์

จากรูปวงจรที่ 2.10 เอาต์พุตของออปแอมป์สามารถเพิ่มกระแสไฟฟ้าโดยการเพิ่มทรานซิสเตอร์ 1 หรือ 2 ตัวระหว่างเอาต์พุตของออปแอมป์กับช่วงต่อสัญญาณป้อนกลับแบบลบ วงจรขยายความแตกต่างต่อแรงดันไฟฟ้าอินพุต $e_{in}(e_{in} = e_2)$ ที่ขานอนอินเวอร์ติ้งและต่อซีโรหรือออฟเซต (Zero หรือ Offset) ที่ขาอินเวอร์ติ้ง มี $R_s$ ไว้ปรับสแปน ในทางปฏิบัติแล้ววงจรขยายความแตกต่างนี้จะนำวงจรขยายการวัด (Instrument Amplifier) มาทำงานแทน

### 2.3.4 อุปกรณ์ที่ใช้ในการตรวจวัดอุณหภูมิ

เครื่องมือที่ใช้วัดอุณหภูมิเรียกว่าเทอร์โมมิเตอร์ มีสารหลายชนิดที่สามารถนำมาประดิษฐ์เป็นเทอร์โมมิเตอร์ได้ เช่น ปรอท แก๊ส อากาศ น้ำ แอลกอฮอล์ เป็นต้น จากตารางที่ 2.1 จะเห็นว่าเทอร์โมมิเตอร์ที่ทำด้วยปรอท แก๊สไฮโดรเจน และอากาศนั้นต่างก็แบ่งระยะระหว่างจุดเยือกแข็งกับจุดเดือดของน้ำออกเป็น 100 ช่องเหมือนกัน แต่ถ้านำไปวัดที่ระดับอุณหภูมิต่างๆ จะอ่านแตกต่างกันเล็กน้อย ทั้งนี้ขึ้นอยู่กับการขยายตัวซึ่งเป็นคุณสมบัติเฉพาะของแต่ละสาร แต่โดยทั่วๆ ไปพบว่าการขยายตัวหรือหดตัวของแก๊สทุกๆ

องศาที่เปลี่ยนไปนั้น ค่อนข้างคงที่มากกว่าของเหลวและของแข็ง ดังนั้นสามารถใช้เทอร์โมมิเตอร์แก๊สเป็นหลักในการเทียบมาตราส่วนอุณหภูมิ ตารางที่ 2.1 เป็นการเปรียบเทียบเทอร์โมมิเตอร์แก๊สกับเทอร์มิเตอร์ปรอท โดยให้เทอร์โมมิเตอร์ไฮโดรเจนมีคุณสมบัติใกล้เคียงกับแก๊สอุดมคติ

**ตารางที่ 2.1** เปรียบเทียบคุณสมบัติเทอร์โมมิเตอร์แก๊สกับเทอร์โมมิเตอร์ปรอท

| เทอร์โมมิเตอร์ไฮโดรเจน (ปริมาตรคงที่) | เทอร์โมมิเตอร์อากาศ (ปริมาตรคงที่) | เทอร์โมมิเตอร์ปรอท |
|---|---|---|
| 0 | 0 | 0 |
| 20 | 20.008 | 20.091 |
| 40 | 40.001 | 40.111 |
| 60 | 59.990 | 60.086 |
| 80 | 79.987 | 80.041 |
| 100 | 100 | 100 |

โดยทั่วๆ ไปนิยมใช้ปรอทเป็นส่วนประกอบของเทอร์โมมิเตอร์เนื่องจากปรอทมีการขยายตัวและหดตัวค่อนข้างคงที่ทุกๆ องศาที่เปลี่ยนไป การขยายตัวและหดตัวน้อยมากเมื่อเทียบกับแก๊ส มีช่วงของการเป็นของเหลวกว้าง (ปรอทแข็งตัวที่ -38.87 °C เดือดที่ 356.9 °C) เทอร์โมมิเตอร์ทำด้วยปรอทจึงกะทัดรัด ไม่ต้องใช้หลอดแก้วยาว เทอร์โมมิเตอร์ที่นิยมใช้กันโดยทั่วไป คือ แบบเซลเซียสและฟาเรนไฮต์

จากหลักการในการตรวจวัดอุณหภูมิในหลายๆ วิธีดังกล่าว การตรวจวัดอุณหภูมิโดยอาศัยการเปลี่ยนแปลงคุณสมบัติทางไฟฟ้าถูกนิยมนำมาใช้กันมากที่สุด เพราะสัญญาณที่ได้จากอุปกรณ์เหล่านี้สามารถนำไปต่อรวมกับวงจรไฟฟ้าหรืออิเล็กทรอนิกส์ เพื่อการแสดงผลในเชิงตัวเลขหรือควบคุมระบบกระบวนการที่ต้องการ ดังนั้นในที่นี้จะขอกล่าวเพียงกลุ่มของอุปกรณ์ที่อาศัยหลักการเปลี่ยนแปลงคุณสมบัติทางไฟฟ้าเป็นสำคัญ

**2.3.4.1 อาร์ทีดี**

เป็นคำย่อมาจาก Resistance Temperature Detector หรือความต้านทานที่ใช้ในการตรวจวัดอุณหภูมิ โดยอาศัยหลักการเปลี่ยนแปลงค่าความต้านทางไฟฟ้าโดยความต้านทานจะเป็นสัดส่วนโดยตรงกับอุณหภูมิ

อาร์ทีดี ทำด้วยโลหะที่มีความยาวค่าหนึ่ง ซึ่งจะทำให้เกิดค่าความต้านทานที่ต้องการที่อุณหภูมิ 0 °C ลวดโลหะดังกล่าวนี้พันอยู่รอบแกนที่เป็นฉนวนไฟฟ้าและมีคุณสมบัติทนต่อความร้อน แกนที่ใช้ส่วนมากจะทำมาจากสารประเภทเซรามิก การพันขดลวดจะนิยมทำกันในขณะที่ขดลวดร้อนจนถึงอ่อนตัว หลังจากนั้น

จะต้องผ่านกระบวนการอบความร้อนคลายความเครียดที่อยู่ในขดลวดด้วยอุณหภูมิอย่างน้อย 500 °C เป็นเวลานานถึง 24 ชั่วโมง โดยทั่วไปจะถูกบรรจุอยู่ในฝักโลหะ ฉนวนที่ใช้จะเป็นแมกนีเซียมออกไซด์หรืออลูมิเนียมออกไซด์

อาร์ทีดีโดยทั่วไปแล้วทำมาจากโลหะที่มีความต้านทานต่ำ เช่น พลาตินัม ทองแดง นิเกิล สำหรับทังสเตนก็จะใช้เป็นอุปกรณ์ตรวจจับอุณหภูมิที่ต้องการย่านวัดสูงๆ แต่เนื่องจากมีความเปราะและแตกหักง่าย จึงไม่ค่อยนิยมนำมาใช้งาน ส่วนโลหะที่นิยมใช้ทำอาร์ทีดีและให้ผลตอบสนองที่เป็นเส้นตรง คือ พลาตินัม

**ตารางที่ 2.2** คุณสมบัติของอาร์ทีดีทำมาจากโลหะประเภทต่างๆ

| ชนิดของโลหะ | กระเปราะที่บรรจุ (Case) | ย่านอุณหภูมิ (องศา) | ความต้านทาน (Ω) | ค่าสัมประสิทธิ์โดยประมาณ ($\alpha$)Ω/Ω/°C |
|---|---|---|---|---|
| พลาสติก | แก้ว | -190 ถึง 540 | 25 ที่ 0 °C | 0.0039 |
| ห้องทดลอง | | | | |
| พลาตินัม | สแตนเลส | -200 ถึง 125 | 25 ที่ 0 °C | 0.0039 |
| (อุตสาหกรรม) | เหล็ก | -18 ถึง 540 | 25 ที่ 0 °C | 0.0039 |
| พลาตินัม | เซรามิก | -50 ถึง 600 | 1000 ที่ 0 °C | 0.0039 |
| (แบบแผ่นฟิล์ม) | อะลูมินาและแก้ว | -272 ถึง 200 | 26 ที่ 0 °C | 0.0037 |
| โรเดียม | ทองเหลือง | -75 ถึง 120 | 10 ที่ 0 °C | 0.0038 |
| เหล็ก | ทองเหลือง | 0 ถึง 120 | 10 ที่ 0 °C | 0.0038 |
| ทองแดง | | | | |

จำเป็นอย่างยิ่งที่ต้องพิจารณาสายที่ต่อระหว่างอาร์ทีดีกับวงจรบริดจ์มากเป็นพิเศษ หากสายที่ใช้ต่อมีความยาวมาก จะส่งผลต่ออุณหภูมิและค่าความต้านทานที่เกิดขึ้นกับวงจร ทำให้ผลของการวัดเกิดความผิดพลาด จึงจำเป็นต้องมีการแก้ไขปัญหาที่เกิดขึ้นเพื่อชดเชยค่าที่ผิดพลาด ซึ่งสามารถทำได้หลายวิธีด้วยกัน การต่อวงจรอาร์ทีดีแบบ 3 สาย เป็นแบบมาตรฐานที่นิยมใช้กันมากที่สุด ทั้งนี้สายตัวนำทั้ง 3 ต้องมีความยาวและขนาดเท่ากัน รวมทั้งอยู่ในสภาพแวดล้อมที่มีอุณหภูมิเดียวกันตลอด

**2.3.4.2 เทอร์มิสเตอร์**

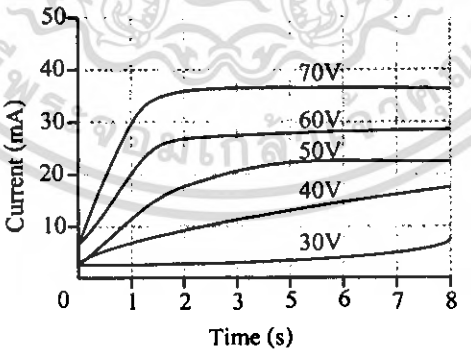
เป็นอุปกรณ์ตรวจวัดอุณหภูมิ ที่อาศัยการเปลี่ยนแปลงค่าความต้านทานเช่นเดียวกับอาร์ทีดี แต่เทอร์มิสเตอร์ทำจากคาร์บอนและสารกึ่งตัวนำ เช่น ออกไซด์ของโลหะ นิเกิล โคบอลต์ เหล็ก ทองแดง เยอมาเนียม

แมกนีเซียม และไทเตเนียม ซึ่งส่วนใหญ่จะนิยมใช้ออกไซด์ของแมกกานิสกับทองแดงและออกไซด์ของนิเกิลกับทองแดง สัมประสิทธิ์การเปลี่ยนแปลงค่าความต้านทานของเทอร์มิสเตอร์จะมีค่าสูง ข้อสำคัญที่ทำให้ต่างจากอาร์ทีดี คือ การเปลี่ยนแปลงค่าความต้านทานจะกลับกันกับอาร์ทีดี กล่าวง่ายๆ คือ ค่าความต้านทานจะลดลงเมื่ออุณหภูมิสูงขึ้น ซึ่งทั้งนี้เนื่องจากคุณสมบัติของสารกึ่งตัวนำ

เมื่อนำเทอร์มิสเตอร์มาใช้งานร่วมกับวงจรไฟฟ้า แรงดันตกคร่อมจะเพิ่มขึ้นตามกระแส แต่เมื่อถึงที่จุดหนึ่ง แรงดันไฟฟ้าตกคร่อมจะลดลงในขณะที่กระแสเพิ่มขึ้นเรื่อยๆ เนื่องจากคุณสมบัติของเทอร์มิสเตอร์มีค่าความต้านทานเป็นลบ ถ้าแรงดันไฟฟ้าที่ป้อนให้กับเทอร์มิสเตอร์มีค่าน้อย กระแสจะมีค่าน้อยตามไปด้วย ดังนั้นความร้อนที่เกิดขึ้นจากกระแสนี้ก็ยังไม่เพียงพอที่จะทำให้อุณหภูมิของเทอร์มิเตอร์มีค่าสูงตามอุณหภูมิของสภาพแวดล้อมในขณะนั้นได้ ภายใต้สภาวะเช่นนี้ กระแสจะเป็นสัดส่วนกับแรงดันไฟฟ้าตามกฎของโอห์ม

เมื่อแรงดันไฟฟ้าที่ป้อนมีค่าสูงขึ้น ทำให้กระแสสูงตามไปด้วย จะเกิดความร้อนทำให้อุณหภูมิเทอร์มิสเตอร์สูงเกินกว่าอุณหภูมิของสภาพแวดล้อมในขณะนั้นและค่าความต้านทานของเทอร์มิสเตอร์จะลดลง ซึ่งผลอันนี้จะทำให้กระแสไหลมากขึ้นและค่าความต้านทานยังลดลงไปเรื่อยๆ ค่ากระแสนี้จะค่อยๆ เพิ่มขึ้นเรื่อยๆ จนกระทั่งค่าสูญเสียของเทอร์มิสเตอร์ที่มีค่าเท่ากับพลังงานหรือกำลังที่ป้อนให้ภายใต้สิ่งแวดล้อมอุณหภูมิคงที่ ค่าความต้านทานของเทอร์มิสเตอร์จะเป็นฟังก์ชั่นของพลังงานที่สิ้นเปลืองภายในตัวของมันเอง จะต้องมีพลังงานที่เพียงพอที่จะทำให้อุณหภูมิของมันสูงกว่าอุณหภูมิของสภาพแวดล้อม

ในรูปที่ 2.11 แสดงให้เห็นถึงคุณลักษณะของกระแสกับเวลาของเทอร์มิสเตอร์ กระแสจะมีค่าสูงสุดเมื่อเวลาผ่านไปเพียงเล็กน้อย และขึ้นอยู่กับขนาดของแรงดันไฟฟ้าที่จ่ายให้ด้วย เมื่อเกิดความร้อนในตัวเตอร์มิสเตอร์ จะใช้เวลาชั่วขณะหนึ่งที่จะทำให้อุณหภูมิสูงขึ้นและกระแสไฟฟ้าจะมีค่าคงที่ ซึ่งในช่วงเวลาดังกล่าวนี้ถือได้ว่า เป็นเวลาที่ใช้ในการตอบสนองนั้นเอง ช่วงเวลานี้จะแปรผันไปตามแรงดันไฟฟ้าที่จ่ายให้




**รูปที่ 2.11** คุณลักษณะของกระแสกับเวลาของเทอร์มิสเตอร์

เนื่องจากสัมประสิทธิ์การเปลี่ยนแปลงของเทอร์มิสเตอร์มีค่าสูง การใช้เทอร์มิสเตอร์ต่อร่วมกับวงจรบริดจ์เหมือนอาร์ทีดีนั้น ทำให้สามารถอ่านค่าได้ละเอียดมากได้ในช่วงอุณหภูมิแคบๆ โดยสังเกตการณ์เปลี่ยนแปลงอุณหภูมิแม้เพียง 0.005 °C เท่านั้น ดังนั้นวงจรการวัดโดยทั่วไปจึงหันมาใช้วงจรแบ่งแรงดันและสายที่ต่อจากเทอร์มิสเตอร์มายังวงจรก็ไม่มีส่วนสร้างความผิดพลาดเหมือนอาร์ทีดี

**2.3.4.3 ไอซี DS1820**

ไอซี DS1820 เป็นไอซีระบบการสื่อสารข้อมูลอนุกรมแบบหนึ่งสายซึ่งถือได้ว่าเป็นระบบที่มีความชาญฉลาดและใช้จำนวนสายสัญญาณเพียง 1 เส้นเท่านั้น โดยไม่ต้องมีสายสัญญาณนาฬิกามาควบคุมจังหวะการถ่ายทอดข้อมูลเหมือนกับระบบสื่อสารข้อมูลอนุกรมในแบบอื่นสายข้อมูลจะทำหน้าที่เสมือนเป็นสายสัญญาณนาฬิกาในตัว ส่วนค่าของข้อมูลจะพิจารณาจากลักษณะของรูปสัญญาณที่ปรากฏบนสายสัญญาณในแต่ละช่องของเวลาซึ่งเรียกว่า ไทม์สล็อต (Time Slot) โดยคาบเวลาต่ำสุดและสูงสุดของสถานะต่างๆ ในการสื่อสารข้อมูลในแต่ละไทม์สล็อตมีการกำหนดขอบเขตไว้อย่างชัดเจนในแต่ละไทม์สล็อตนั้น รูปแบบการถ่ายทอดข้อมูลจะเป็นอะซิงโครนัสในระดับบิต ไม่มีการสื่อสารข้อมูลระหว่างไอซีแผงวงจรเดี่ยว

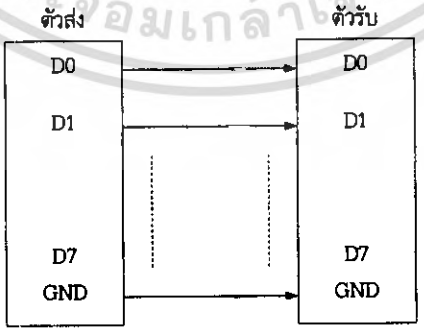
## 2.4 การรับส่งข้อมูลแบบอนุกรม

### 2.4.1 พื้นฐานการรับส่งข่อมูล

การรับส่งข้อมูลในระบบคอมพิวเตอร์ หมายถึง การรับส่งข้อมูลเป็นจำนวนไบต์ๆ ให้กับอุปกรณ์ที่เกี่ยวข้องกับคอมพิวเตอร์ ซึ่งแบ่งประเภทของการรับส่งข้อมูลได้ 2 แบบ

1. การรับส่งข้อมูลแบบขนาน (Parallel)
2. การรับส่งข้อมูลแบบอนุกรม (Serial)

การรับส่งข้อมูลแบบขนานเป็นการรับส่งข้อมูล จำนวน 1 ไบต์ ออกไปทางพอร์ต ในเวลาเดียวกันในระบบคอมพิวเตอร์ 1 ไบต์จะมีจำนวน 8 บิต คือ DO – D7 ถ้ามีการส่งข้อมูลแบบขนานจะใช้สายสัญญาณอย่างน้อย 9 เส้น คือ สาย Data 8 เส้น และสายกราวด์ 1 เส้นดังรูปที่ 2.12

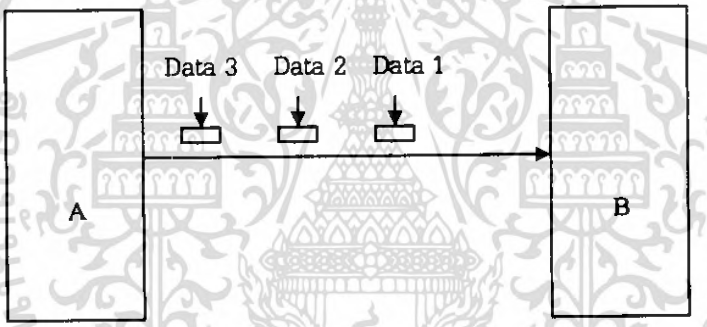



**รูปที่ 2.12** การรับส่งข้อมูลแบบขนาน

การรับส่งข้อมูลแบบอนุกรม คือ การรับส่งข้อมูลทีละบิต จนครบ 1 ไบต์ ถ้าต้องการส่งข้อมูล 1 ไบต์ คือ D0 – D7 อาจจะส่งบิต D0 ออกไปก่อนแล้วตามด้วย D1 ไปเรื่อยๆ จนถึง D7 การส่งข้อมูลทั้งสองแบบมีข้อดีข้อเสียแตกต่างกันคือ การส่งข้อมูลแบบขนานสามารถส่งข้อมูลได้เร็ว คือส่งเป็นระยะไกลๆ จะสิ้นเปลืองสายสัญญาณ ถ้าเป็นการส่งแบบอนุกรม เมื่อต้องการส่งข้อมูลเป็นระยะไกลๆ จะช่วยประหยัดสายสัญญาณเนื่องจากจะใช้สายอย่างน้อยเพียง 2 เส้น คือ สายสัญญาณกับสายกราวด์ แต่การรับส่งข้อมูลจะใช้เวลานานเนื่องจากเป็นการส่งทีละบิต ในบทนี้จะกล่าวถึงพื้นฐานการรับส่งข้อมูลแบบอนุกรม โดยจะเน้นที่ตัว MSC-51 เป็นสำคัญ

**2.4.1.1 การรับส่งข้อมูลแบบซิงโครนัส (Synchronous Input / Output)**

การรับส่งข้อมูลแบบนี้ไม่ว่าจะเป็นการส่งแบบอนุกรมหรือขนาน ข้อมูลแต่ละไบต์ในการรับส่งจะมีช่วงห่างกันแน่นอน เช่น การส่งข้อมูลจาก A ไป B ดังรูปที่ 2.19 Data 1 จะห่างจาก Data 2 เป็นเวลาและ Data 3 จะห่างจาก Data 2 เป็นเวลา t เช่นกัน ระบบนี้เหมาะสมกับงานที่ไม่มีความยุ่งยากมาก




**รูปที่ 2.13** การส่งข้อมูลแบบซิงโครนัส

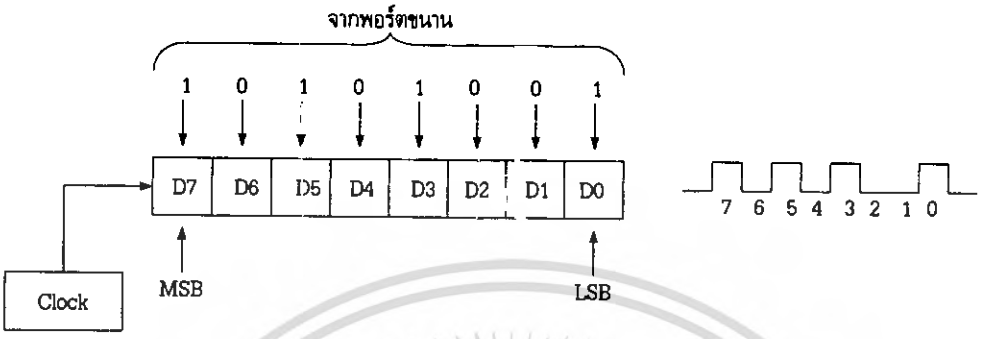
**2.4.1.2 การรับส่งข้อมูลแบบอะซิงโครนัส**

การรับส่งข้อมูลแบบนี้ ข้อมูลในการรับส่งจะไม่มีเวลาที่แน่นอน ซึ่งจะขึ้นอยู่กับความพร้อมของผู้ส่งและผู้รับ โดยจะมีสายสัญญาณตรวจสอบความพร้อมของระบบทั้งสองว่าพร้อมที่จะติดต่อกันหรือไม่ โดยสัญญาณที่เพิ่มขึ้นมาจากระบบแบบซิงโครนัส เรียกว่า สายสเตตัส (Status Line)

**2.4.2 รูปแบบของการรับส่งข้อมูลแบบอนุกรม**

เมื่อไมโครคอมพิวเตอร์ต้องการจะรับส่งข้อมูลแบบอนุกรม ตัวไมโครคอมพิวเตอร์จะส่งข้อมูลออกไปทางพอร์ทซึ่งเป็นพอร์ทแบบขนานก่อน จากนั้นจะมีอุปกรณ์มาต่อที่พอร์ท เพื่อแปลงข้อมูลแบบขนานให้เป็นแบบอนุกรมอีกทีหนึ่ง (Parallel – to – Serial Conversion) ตัวแปลงข้อมูลนี้อาจจะพิจารณาง่ายๆ ว่าเป็นชิพรีจิสเตอร์ ดังรูปที่ 2.14 เมื่อข้อมูลที่จะส่งอยู่ในชิพรีจิสเตอร์ แล้วตัวสัญญาณนาฬิกาจะเป็นตัวกระตุ้น

ให้ข้อมูลบิตต่ำออกไปในเวลาแรก จากนั้นจะส่งบิตต่อไปตามออกมา จากรูปที่ 2.14 จะเป็นการส่งข้อมูล A9H ออกไป




**รูปที่ 2.14** การส่งข้อมูลแบบอนุกรม

สำหรับตัวรับข้อมูลแบบอนุกรมนั้นเมื่อตัวรับข้อมูลทำงานจะเป็นการรับเข้ามาใน Shift Register แล้วส่งข้อมูลให้ไมโครคอมพิวเตอร์แบบขนาน (Serial to Parallel) ระบบคอมพิวเตอร์ในปัจจุบันจะมีตัวแปลง Parallel to Serial และ Serial to Parallel อยู่ในชิพไอซี เรียกว่า Universal Asynchronous Receiver Transmitter (UART) การส่งข้อมูลแบบอนุกรมต้องมีการเพิ่มเติมข้อมูลบางอย่างเข้าไปเพื่อให้การรับส่งข้อมูลสามารถทำงานได้ถูกต้องโดยมีการเติมค่าบิตต่างๆ ลงไปตามรูปที่ 2.20


**รูปที่ 2.15** บิตต่างๆ ของข้อมูลที่ส่งแบบอนุกรม

การส่งข้อมูลแบบ 8 บิต จะต้องส่งบิตแรกออกไปก่อน เรียกว่า บิตต้น (Start Bit) ถ้ามีการส่งข้อมูลหลายๆ ไบต์ออกมา บิตจะเป็นตัวบอกว่ามีข้อมูลใหม่มา โดยทั่วไปบิตเริ่มต้นมักมีระดับลอจิกเป็น "0" ถัดไปจะเป็นข้อมูลบิต D0 ถึง D7 จากนั้นจะตามด้วยบิตตรวจสอบความถูกต้อง (Parity Bit) ถ้าข้อมูล 8 บิตที่ส่งออกมา จำนวนของบิตที่มีค่าเป็น "1" เป็นจำนวนคู่ บิตนี้จะมีค่าเป็น 0 แต่ถ้าจำนวนของบิตที่มีค่าเป็น "1" เป็นจำนวนคี่ บิตนี้จะค่าเป็น 1 จากนั้นข้อมูลที่ส่งออกไปจะตามด้วยบิตสิ้นสุด (Stop Bit) เพื่อเป็นการบอกว่าข้อมูลที่ส่งมา 8 บิตนั้นหมดแล้ว ตัวบิตสิ้นสุดอาจมีจำนวนมากกว่า 1 บิตก็ได้ เช่น 1 5บิต, 2 บิต

บิตตรวจสอบความถูกต้องหรือบิตพาริตี้จะมีสองลักษณะคือพาริตี้คู่ (Even Parity) และพาริตี้คี่ (Odd Parity) ซึ่งสามารถเลือกได้ ถ้าหากระบุเป็นพาริตี้คู่หมายความว่าข้อมูลที่ส่งไปหรือไบต์นั้นมีจำนวนลอจิก "1" รวมกับบิตพาริตี้เป็นจำนวนคู่บิต ถ้าหากระบุเป็นพาริตี้คู่หมายความว่าข้อมูลที่ส่งไปหรือไบต์นั้นมี

จำนวนลอจิก "1" ของไบต์ข้อมูลที่ส่งไปรวมกับค่าบิตพาริตี้เป็นจำนวนคี่บิต ตัวอย่างเช่นถ้าเราส่งไบต์ข้อมูลที่มีค่าเป็น B2H หรือ 10110010B ออกไป ถ้าระบบระบุว่าเป็นพาริตี้เคี่ยวแล้วค่าของบิตพาริตี้จะต้องเป็นลอจิก "1" ซึ่งในไบต์ข้อมูลจะมีลอจิก "1" จำนวน 4 บิต และบิตพาริตี้เป็น "1" ดังนั้นมีลอจิก "1" ทั้งหมด 5 บิต ซึ่งเป็นคี่บิตแต่ถ้าระบบระบุว่าเป็นระบบพาริตี้คู่แล้วค่าพาริตี้จะต้องเป็นลอจิก "0" เมื่อทางภาครับได้รับข้อมูลเข้ามาก็ตรวจสอบว่าข้อมูลทั้งหมดมีลอจิก "1" เป็นคู่หรือคี่บิต ตรงกับการทำงานของระบบซึ่งออกแบบไว้หรือไม่ ซึ่งจะเป็นการตรวจสอบความถูกต้องได้ระดับหนึ่ง

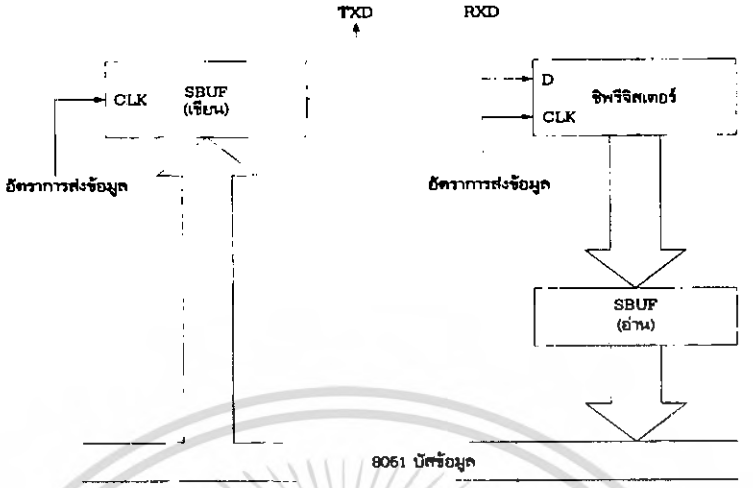
การสื่อสารข้อมูลแบบอนุกรม การกำหนดอัตราเร็วในการรับส่งข้อมูลจะบอกเป็นบิตต่อวินาที (Bit Per Second: BPS) ที่เรียกว่าอัตรา Baud หรือ Baud Rate โดยค่ามาตรฐานที่ใช้กันมีหลายค่าได้แก่ 110, 150, 300, 600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที

ใน MCS-51 นั้นจะมีพอร์ทสำหรับรับส่งข้อมูลแบบอนุกรมอยู่ภายในชิพ โดยไม่ต้องต่ออุปกรณ์ภายนอกเพิ่ม และเป็นพอร์ทสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ (Full-Duplex) คือสามารถรับส่งข้อมูลได้สองทิศทางในเวลาเดียว การควบคุมการใช้งานพอร์ทอนุกรมของ MCS-51 นี้สามารถโปรแกรมให้ทำงานรูปแบบต่างๆ ได้ 4 ประเภทหรือ 4 โหมดการทำงาน โดยโปรแกรมผ่านทางรีจิสเตอร์ SCON โดยโหมด 0 จะเป็นการสื่อสารแบบซิงโครนัสหรือแบบเข้าจังหวะเวลา ส่วนโหมด 1, 2 และ 3 เป็นการสื่อสารแบบอะซิงโครนัส

### 2.4.3 การรับส่งข้อมูลแบบอนุกรมใน MCS-51

พอร์ทอนุกรมของ MCS-51 จะใช้ขา TXD และ RXD ในการรับส่งข้อมูล โดยขาทั้งสองจะอยู่ในพอร์ท 3 คือ P3.1 หรือขา 11 TXD และ P3.0 หรือขา 10 เป็น RXD พอร์ทอนุกรมของ MCS-51 สามารถทำงานแบบ Full Duplex ได้ คือสามารถส่งและรับข้อมูลในเวลาเดียวกันได้ โดยในการรับและส่งข้อมูลจะมีบัฟเฟอร์สำหรับเก็บข้อมูลให้ใช้

รีจิสเตอร์ที่สำคัญในการรับส่งข้อมูล คือ SBUF และ SCON ซึ่งเป็นรีจิสเตอร์ที่อยู่ใน Special Function Registers (SFR) โดยรีจิสเตอร์ Serial Port Buffer (SBUF) จะอยู่ในตำแหน่ง 99H ถ้าเขียนข้อมูลไปที่ตำแหน่งนี้จะเป็นการส่งข้อมูลออกทางพอร์ทอนุกรม และถ้าอ่านข้อมูลจากตำแหน่งนี้จะเป็นการรับข้อมูลจากพอร์ทอนุกรม โดยใน SBUF จะประกอบด้วยบัฟเฟอร์ 2 ตัว สำหรับส่งและรับข้อมูล ดังรูปที่ 2.16




**รูปที่ 2.16** การรับส่งข้อมูลระหว่างรีจิสเตอร์ภายใน

สำหรับ Serial Port Register (SCON) ซึ่งอยู่ในตำแหน่ง 98H จะเป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับได้รีจิสเตอร์นี้จะทำหน้าที่ควบคุมและบอกสถานะต่างๆ ของการรับส่งข้อมูลแบบอนุกรม

สำหรับความเร็วของการส่งข้อมูล (Baud Rate) สามารถหาได้จากการหารสัญญาณนาฬิกาที่ใช้กับ MCS-51

### 2.4.4 รีจิสเตอร์ควบคุมพอร์ตอนุกรม

MCS-51 มีโหมดการทำงานของพอร์ตอนุกรมหลายโหมด ซึ่งสามารถโปรแกรมการทำงานได้โดยเขียนข้อมูลไปยังรีจิสเตอร์ SCON ความหมายของแต่ละบิตแสดงดังตารางที่ 2.3 และ 2.4

**ตารางที่ 2.3** บิตต่างๆ ของรีจิสเตอร์ SCON

| บิต | ชื่อ | ตำแหน่ง | ความหมาย |
|---|---|---|---|
| SCON.7 | SM0 | 9FH | บิตเลือกโหมดการทำงานบิต 0 |
| SCON.6 | SM1 | 9EH | บิตเลือกโหมดการทำงานบิต 1 |
| SCON.5 | SM2 | 9DH | บิตเลือกโหมดการทำงานบิต 2 |
| SCON.4 | REN | 9CH | บิตแฟลกกำหนดยอมให้มีการรับข้อมูล |
| SCON.3 | TB8 | 9BH | ค่าของบิต 9 สำหรับการส่งข้อมูลในโหมด 2 และ 3 สามารถ Set และ Clear ได้โดยโปรแกรม |
| SCON.2 | RB8 | 9AH | ค่าของบิต 9 เมื่อรับข้อมูลเข้ามา |

**ตารางที่ 2.3 (ต่อ)** บิตต่างๆ ของรีจิสเตอร์ SCON

| บิต | ชื่อ | ตำแหน่ง | ความหมาย |
|---|---|---|---|
| SCON.1 | TI | 99H | บิตแฟลกแสดงการอินเทอร์รัพท์ภายหลังการส่งข้อมูลออกโดยจะ Set เมื่อส่งข้อมูลออกไปหมดแล้วและสามารถ Clear ได้ด้วยโปรแกรม |
| SCON.0 | RI | 98H | แฟลกแสดงการอินเทอร์รัพท์ภายหลังรับข้อมูลเข้ามาสามารถ Clear ได้ด้วยโปรแกรม |

**ตารางที่ 2.4** โหมดต่างๆ ของการรับส่งแบบอนุกรม

| SM0 | SM1 | MODE | ความหมาย | Baud Rate |
|---|---|---|---|---|
| 0 | 0 | 0 | Shift Register | เปลี่ยนไม่ได้ ( Oscillator Frequency ÷12 ) |
| 0 | 1 | 1 | 8 - bit UART | สามารถเปลี่ยนแปลงได้โดยการกำหนดจาก Timer |
| 1 | 0 | 2 | 9 - bit UART | เปลี่ยนไม่ได้ (Oscillator Frequency ÷32 หรือ ÷64 ) |
| 1 | 1 | 3 | 9 - bit UART | สามารถเปลี่ยนแปลงได้โดยการกำหนดจาก Timer |

ก่อนที่จะพอร์ทอนุกรมจะต้องโปรแกรมให้กับ SCON เสียก่อนเพื่อกำหนดโหมดการทำงานและลักษณะต่างๆ เช่น

```
MOV   SCON, #01010010B
```

เป็นการกำหนดให้พอร์ทอนุกรมทำงานในโหมด 1 และอีนาเบิลให้มีการรับข้อมูลพร้อมกับการกำหนดให้ TI เป็น 1

ในการส่งข้อมูลทุกโหมดสามารถทำได้โดยเขียนข้อมูลไปยัง SBUF เมื่อข้อมูลถูกส่งไปแล้วบิต TI จะถูกเซตเป็น "1" ในการส่งข้อมูล จะต้องคอยตรวจสอบบิต TI เพราะว่าถ้า TI ยังไม่เป็น "1" แสดงว่าข้อมูลยังส่งไปไม่หมด ถ้าหากมีการเขียนข้อมูลไบต์ต่อไปยัง SBUF จะทำให้เกิดข้อผิดพลาดขึ้น สำหรับในการรับส่งข้อมูลบิต REN จะต้องเซตให้เป็น "1" ยกเว้นโหมด 0 เพื่ออนุญาตให้รับข้อมูลได้ เมื่อข้อมูลรับเข้ามาเรียบร้อยแล้ว บิต RI จะถูกเซตเป็น "1"

### 2.4.5 โหมดการรับ-ส่งข้อมูล

การรับส่งข้อมูลกับไมโครคอนโทรลเลอร์ MCS-51 นั้น ภายในชิพ MCS-51 จะมี UART อยู่ในตัว ซึ่งเป็นข้อดีของไมโครคอนโทรลเลอร์ ถ้าเป็นไมโครโปรเซสเซอร์ เช่น เบอร์ Z-80 ถ้าต้องการรับส่งทางพอร์ตอนุกรม โดยโหมดการรับส่งข้อมูลทางพอร์ตอนุกรมจะประกอบด้วย 4 โหมดการทำงานดังนี้

1) 8 - Bit Shift Register (Mode 0)

2) 8 - Bit UART with Variable Baud Rate (Mode 1)

3) 9 - Bit UART with Fixed Baud Rate (Mode 2)

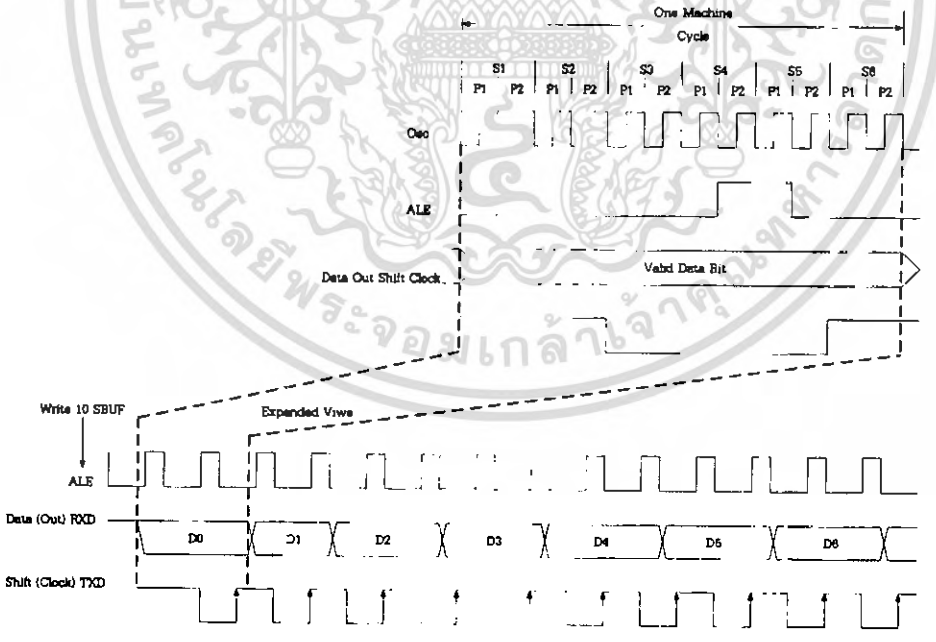
4) 9 -Bit UART with Variable Baud Rate (Mode 3)

ใน MCS-51 การสื่อสารทางพอร์ทอนุกรมจะมีอยู่ 4 ประเภท หรือ 4 โหมด ซึ่งกำหนดได้ที่บิต SM0 และ SM1 ใน SCON โดยจะมี 3 โหมด เป็นการสื่อสารแบบทางเดียวในลักษณะของข้อมูลที่ส่งจะมีบิตเริ่มต้น (Start Bit) และบิตจบ (Stop Bit) คล้ายกับการสื่อสารแบบ RS-232 ในระบบคอมพิวเตอร์ อีกโหมดหนึ่งจะเป็นการใช้พอร์ทอนุกรมในลักษณะชิพรีจิสเตอร์

**2.4.5.1 8-Bit Shift Register (Mode 0)**

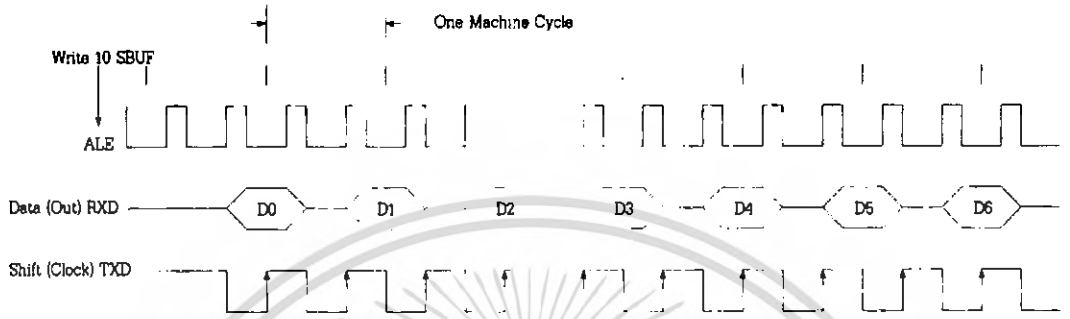
การทำงานในโหมดนี้จะใช้ขา RXD ในการรับส่งข้อมูลโดยต่อกับชิพรีจิสเตอร์ภายนอก ส่วนขา TXD จะเป็นสัญญาณนาฬิกาทางด้านเอาต์พุต เพื่อกระตุ้นรีจิสเตอร์ภายนอกให้เลื่อนบิตถ้ามีการส่งข้อมูลหรือรับข้อมูล 8 บิต จะเริ่มที่บิตต่ำสุดก่อน โดยมีค่าอัตรา Baud เท่ากับ 1/2 ของความถี่ที่ใช้บนชิพ

ในการส่งข้อมูลจะทำโดยเขียนข้อมูลไปที่รีจิสเตอร์ SBUF ข้อมูลจะถูกส่งออกมาทางขา RXD (P3.0) โดยจะสอดคล้องกับสัญญาณที่ออกมาทางขา TXD ซึ่งสัญญาณทางขา TXD จะถูกส่งออกมาทุกๆ รูปคลื่น โดยจะเป็นลอจิก "0" ใน S3P1 และจะกลับเป็นลอจิก "1" ใน S6P1 ซึ่งแสดงได้ดังรูปที่ 2.17



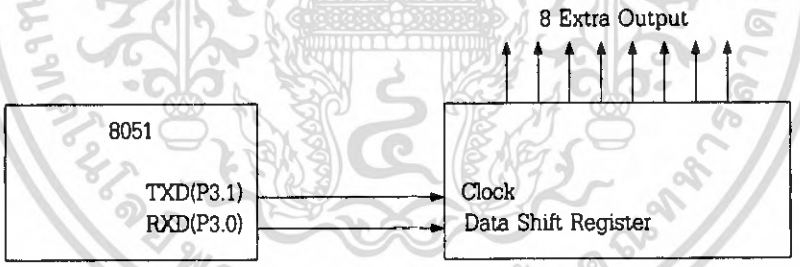

**รูปที่ 2.17** แผนผังเวลาการส่งข้อมูล

สำหรับการรับข้อมูลจะรับได้เมื่อเซตขา Receiver Enable Bit (REN) เป็น "1" และเคลียร์ขา Receiver Interrupt Bit (RI) เป็น "0" ข้อมูลจะเข้าสู่ MCS-51 เมื่อสัญญาณนาฬิกาถูกส่งออกไปทาง TXD ที่ขอบขาขึ้นของสัญญาณนาฬิกาบิตต่ำจะถูกส่งเข้ามาก่อนดังรูปที่ 2.17




**รูปที่ 2.18** แผนผังเวลาการรับข้อมูล

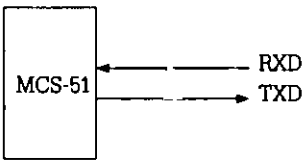
ในการประยุกต์ใช้งานโหมดนี้จะต้องมีไอซีชิพรีจิสเตอร์มาต่อภายนอก เช่น ถ้าหากต้องการส่งข้อมูลออกมาทางพอร์ตอนุกรม อาจต่อวงจรได้ดังรูปที่ 2.18 โดยใช้ไอซี Serial-to Parallel Shift Register โดยข้อมูลส่งออกมาทาง RXD และใช้ TXD เป็น Clock




**รูปที่ 2.19** การรับส่งข้อมูลออกโดยใช้ชิพรีจิสเตอร์ช่วย

**2.4.5.2 8-Bit UART with Variable Baud Rate (Mode 1)**

ในโหมดนี้จะเป็นการรับส่งข้อมูลแบบ 10 บิตซึ่งประกอบด้วยบิตเริ่มต้น (เป็น "0") ข้อมูล 8 บิตและบิตจบ (เป็น "1") นอกจากนี้ยังสามารถกำหนดค่าอัตรา Baud ได้โดยค่าอัตรา Baud นี้จะแปรตามตัวจับเวลาที่ 1 ในโหมดนี้ จะส่งข้อมูลออกทาง TXD และรับข้อมูลเข้าทาง RXD ถ้าเป็นการรับข้อมูลเข้าตัว Stop Bit จะเข้ามายังบิต RB8 ใน SCON




**รูปที่ 2.20** การรับส่งข้อมูลในโหมด 1

ค่าอัตรา Baud ที่ใช้ในการรับส่งข้อมูลจะกำหนดโดย Timer 1 หลังจากโปรแกรมไปใน Timer 1 แล้วสามารถเลือกค่าอัตรา Baud ได้อีกสองค่าคือคือ ค่าจาก Timer 1 Overflow หาร 32 กับค่าจาก Timer 1 Overflow หาร 16

การรับสร้างข้อมูลทำได้โดยการเขียนข้อมูล 8 บิตไปที่ SBUF โดยที่ 9 (Stop Bit) ให้เขียนลงใน TB8 ใน SCON จากนั้นข้อมูลจะถูกส่งออกมาทางขา TXD โดยส่งบิตเริ่มต้นออกมาก่อน ตามด้วยข้อมูล 8 บิตและจบด้วยบิตสิ้นสุดเมื่อข้อมูลถูกส่งออกไปหมดแล้วบิต Interrupt Flag (TI) จะเป็น "1" ดังนั้นในการเขียนข้อมูลใหม่ลงไปจะต้องตรวจสอบบิตนี้

ในการรับข้อมูล จะเริ่มเมื่อมีการเปลี่ยนแปลงลอจิกจาก 1 เป็น 0 ทางขา RXD หมายความว่าเริ่มรับบิตเริ่มต้น จากนั้นข้อมูลอีก 8 บิตจะถูกเก็บลงใน SBUF และบิตสิ้นสุดจะถูกเก็บในบิต RB8 ของรีจิสเตอร์ SCON เมื่อข้อมูลเข้ามาครบแล้ว บิต Interrupt Flag (RI) จะถูกเซตดังนั้นในการอ่านข้อมูลจะอ่านได้เมื่อบิต RI ถูกเซตแล้ว เมื่ออ่านข้อมูลไปแล้วจะต้องเคลียร์บิตนี้

**2.4.5.3 9-Bit UART with Fixed Baud Rate (Mode 2)**

การทำงานในโหมดนี้ไม่สามารถกำหนดค่าอัตรา Baud ได้ซึ่งค่าอัตรา Baud จะมีสองค่าคือ 1/64 และ 1/32 ของสัญญาณนาฬิกาบนชิพ การรับส่งข้อมูลจะเป็นชุดข้อมูล 9 บิต บิตเริ่มต้น บิตหยุด รวมเป็น 11 บิต โดยข้อมูล 9 บิตจะเป็นจำนวนข้อมูล 8 บิตและบิตที่โปรแกรมอีก 1 บิต โดยบิตนี้จะเป็นบิตที่ 9 ซึ่งจะเป็นพาริตี้บิตในการส่งข้อมูลจะต้องเขียนไปที่ บิต TB8 ในรีจิสเตอร์ SCON สำหรับการรับส่งข้อมูลบิตที่ 9 จะถูกเก็บในบิต RB8

**2.4.5.4 9-Bit UART with Variable Baud Rate (Mode 3)**

การทำงานในโหมดนี้คล้ายกับโหมด 2 แต่สามารถกำหนดค่าอัตรา Baud ได้โดยการโปรแกรมไปที่ Timer 1 หลังจากโปรแกรมแล้วยังสามารถเลือกได้อีก 2 ค่าคือ ความถี่ค่าเกินของ Timer 1 หารด้วย 16 และหารด้วย 32

### 2.4.6 การกำหนดค่าเริ่มต้นให้รีจิสเตอร์ในการรับส่งข้อมูล

การรับข้อมูล ถ้าจะให้ MCS-51 รับข้อมูลทางพอร์ทอนุกรมจะต้องโปรแกรมไปที่บิต Receiver Enable (REN) ในรีจิสเตอร์ SCON ให้เป็นลอจิก "1" ซึ่งอาจทำได้สองวิธีดังนี้

REN = 1

เป็นการเซตบิต REN ให้เป็น "1" หรืออาจทำโดยใช้คำสั่ง

SCON = xxx1xxxxB

ซึ่งเป็นการย้ายข้อมูลที่ทำให้บิต REN เป็น 1 สำหรับค่า x หมายความว่าเป็นอะไรก็ได้ขึ้นกับการใช้งานในโหมดต่างๆ

**ข้อมูลแบบ 9 บิต** ในการรับส่งข้อมูลที่มีบิตข้อมูลแบบ 9 บิต ได้แก่ การใช้งานในโหมด 2 และโหมด 3 การส่งข้อมูลบิตที่ 9 จะถูกเขียนในบิต TB8 โดยการเขียนโปรแกรม สำหรับการรับข้อมูลเข้ามาถึงบิตที่ 9 จะถูกเขียนลงในบิต RB8

**การเพิ่มบิต Parity** การส่งข้อมูลแบบ 9 บิต สามารถใช้บิตที่ 9 เป็นบิตพาริตี้ได้ ซึ่งบิตพาริตี้ จะอยู่ใน Program Status Word (PSW) โดยจะถูกเซตหรือเคลียร์ทุกๆ แมชชีนไซเคิลที่เกี่ยวข้องกับ Accumulator เช่น ถ้าจะส่งข้อมูลแบบ 8 บิต ตามด้วยบิตพาริตี้คู่เป็นบิตที่ 9 สามารถเขียนโปรแกรมได้ดังนี้

```
MOV   C , P        ; อ่านค่าบิต P มาเก็บใน C
MOV   TB8 , C      ; นำค่าบิตพาริตี้เขียนลงใน TB8
MOV   SBUF , A     ; ส่งข้อมูลไปทางพอร์ทอนุกรม
```

ถ้าเป็นแบบ Odd Parity ให้แก้ไขข้อมูลที่อ่านได้จากบิตพาริตี้เสียก่อนที่จะส่งออกไป ซึ่งเขียนโปรแกรมดังนี้

```
MOV   C , P        ; อ่านค่าบิตพาริตี้มาเก็บใน C
CPL   C            ; กลับค่าให้เป็น Odd Parity
MOV   TB8 , C      ; เขียนค่าลงใน TB8
MOV   SBUF , A     ; ส่งข้อมูลออกทางพอร์ทอนุกรม
```

การส่งข้อมูลแบบมีพาริตี้บิตด้วยไม่ใช่จะส่งได้แบบ 9 บิต หรือโหมด 2 และ 3 เท่านั้น ในโหมด 1 ซึ่งส่งข้อมูลแบบ 8 บิตก็สามารถทำได้ อย่างเช่นการส่งรหัส ASCII จะใช้บิตข้อมูล 7 บิต สำหรับบิตที่เหลืออีกหนึ่งบิตจะเป็นบิตพาริตี้รวมเป็น 8 บิต ซึ่งสามารถเขียนโปรแกรมได้ดังนี้

```
CLR   ACC.7        ; เคลียร์ค่าบิต 7 เพื่อใช้เป็นพาริตี้บิต
MOV   C , P        ; นำบิตพาริตี้มาเก็บใน C
MOV   ACC.7 , C    ; เขียนค่าบิตพาริตี้ลงในรีจิสเตอร์ A
```

```
MOV   SBUF , A        ; ส่งข้อมูลออกทางพอร์ทอนุกรม
```

แฟลกอินเตอร์รัพต์ เมื่อมีการรับส่งข้อมูลเสร็จสิ้นจะมีผลต่อแฟลกอินเตอร์รัปต์ (RI และ TI) ใน รีจิสเตอร์ SCON ซึ่งบิตเหล่านี้จะถูกเซตโดยอุปกรณ์แต่ต้องเคลียร์ด้วยโปรแกรม

บิต RI ถ้าถูกเซตหมายความว่าบัฟเฟอร์ที่ใช้รับข้อมูลเต็มให้อ่านข้อมูลไปได้แล้ว และบิตนี้สามารถใช้ อินเตอร์รัปต์ MCS-51 ได้ แต่ถ้าเขียนโปรแกรมจะใช้วิธีตรวจเช็คบิตนี้ ถ้าเป็น "1" จะสามารถอ่านข้อมูลมา เก็บในรีจิสเตอร์ A ได้ แต่ก่อนอ่านจะต้องเคลียร์ RI ก่อนเพื่อจะได้รับข้อมูลถัดไปได้ ซึ่งเขียนโปรแกรมได้ ดังนี้

```
WAIT : JNB   NI , WAIT     ; ถ้าบิตนี้ไม่เป็น "1" จะทำงานอยู่ที่เดิม
       CLR   RI            ; เคลียร์ RI
       MOV   A , SBUF      ; อ่านค่ามาเก็บใน A
```

บิต TI เมื่อส่งข้อมูลออกไปแล้วบิตนี้จะถูกเซต เป็นการบอกว่าบัฟเฟอร์ส่งข้อมูลว่างแล้วให้ส่งข้อมูล ใหม่เข้าไปได้ ซึ่งสามารถใช้บิตนี้อินเตอร์รัปต์ MCS – 51 ได้เช่นกัน แต่ถ้าเขียนโปรแกรมจะเขียนได้ดังนี้

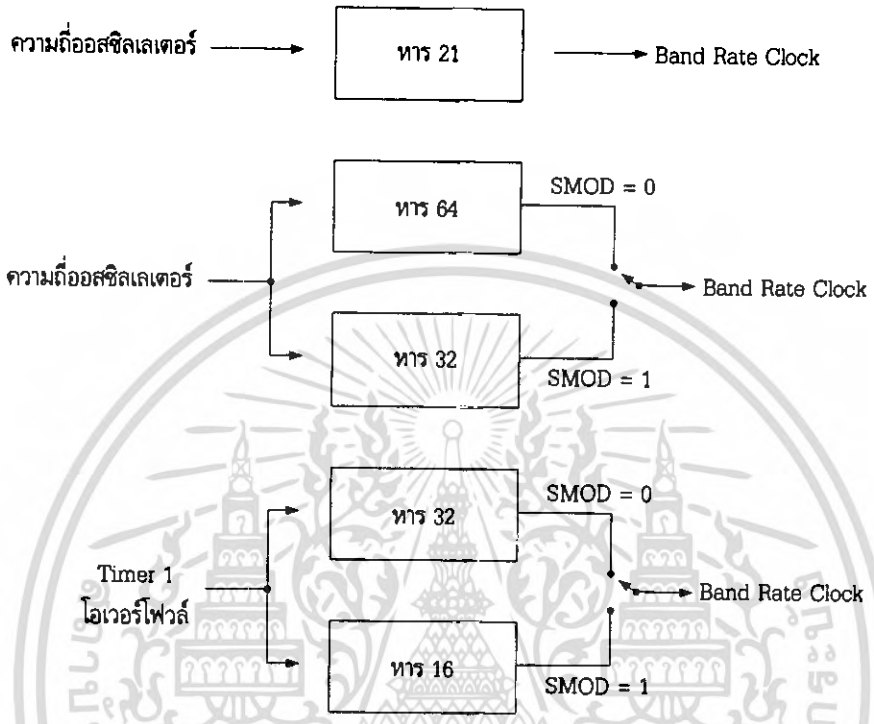
```
WAIT : JNB   RI , WAIT     ; ตรวจบิต TI ว่าเป็น "1" หรือยัง
       CLR   RI            ; เคลียร์ TI
       MOV   SBUF , A      ; เขียนข้อมูลลงไป
```

### 2.4.7 อัตราการส่งข้อมูลของพอร์ตอนุกรม

การรับส่งข้อมูลในโหมดต่างๆ พบว่าในโหมด 0 โหมด 2 ไม่สามารถกำหนดอัตรา Baud เองได้ โดย ในโหมด 0 ค่าอัตรา Baud จะมีค่าเท่ากับความถี่ของตัวกำเนิดความถี่หารด้วย 12 ในโหมด 1 จะมีสองค่าคือ ความถี่ของตัวกำเนิดความถี่หารด้วย 32 และหารด้วย 64 สองค่านี้เรียกว่า SMOD 0 และ SMOD 1 ซึ่ง สามารถกำหนดได้ในริจีสเตอร์ PCON บิตที่ 7 ในริจีสเตอร์ PCON นี้ ไม่สามารถเข้าถึงข้อมูลระดับบิตได้ การเขียนข้อมูลลงไปทีละบิตจะต้องใช้วิธีที่เรียกว่า "Read – Modify – Write" คือ อ่านขึ้นมาแก้ไขแล้วเขียน ลงไปใหม่ ตัวอย่างเช่น

```
MOV   A , PCON        ; อ่านค่าจาก PCON มาเก็บในรีจิสเตอร์ A
SETB  ACC.7           ; เซตบิต 7 (SMOD)
MOV   PCON , A        ; เขียนค่าลงไปใหม่ใน PCON
```

สำหรับโหมด 1 และโหมด 3 สามารถกำหนดค่าอัตรา Baud ได้โดยการโปรแกรมใน Timer1 ในการโปรแกรมแต่ละครั้งจะมี SMOD สองค่าเช่นกัน ค่าอัตรา Baud ของโหมดต่างๆ แสดงได้ดังรูปที่ 2.27




**รูปที่ 2.21** การกำหนดอัตราในโหมดต่างๆ

**การใช้ Timer 1 กำหนด Baud Clock**

การกำหนดค่าลงใน Timer 1 ทำได้โดยการโปรแกรมไปที่ TMOD ให้ทำงานแบบ 8 bit Auto Reload Mode (โหมด 2) โดยเขียนค่าเขียนที่ TH 1 ซึ่งโปรแกรมไปรีจิสเตอร์ TMOD ได้ดังนี้

```
MOV   TMOD   #0010xxxxB
```

ค่า x หมายความว่าเป็นอะไรก็ได้เพราะบิตเหล่านี้ใช้ใน Timer 0

ถ้าต้องการอัตรา Baud ต่ำๆ สามารถใช้ 16 Bit Mode ได้ โดยโปรแกรมเป็น TMOD = 0001xxxxB ค่าอัตรา Baud ที่ส่งออกจะมีค่าเท่ากับ ความถี่ของ Timer 1 เกิดค่าเกินหารด้วย 32 (หรือหารด้วย 16 ถ้าเป็น SMOD = 1)

การคำนวณหาค่าอัตรา Baud ที่กำหนดด้วย Timer 1 สามารถทำได้โดยใช้สมการต่อไปนี้

$$\text{Baud Rate} = \frac{2^{SMOD}}{32} \times \frac{\text{ความถี่ออสซิลเลเตอร์}}{12 \times [256 - TH1]} \qquad (2.4)$$

โดยที่ SMOD เป็นค่าของบิตภายในรีจิสเตอร์ PCON ซึ่งอาจมีค่าเป็น 0 หรือ 1

TH เป็นค่าภายในรีจิสเตอร์ TH1 ซึ่งใช้สำหรับย้อนกลับไปเรียก ค่าของการนับเวลา

รูปแบบทั่วไปของการหาค่าอัตรา Baud ในโหมด 1 และ 3 สามารถหาได้ดังนี้

$$\text{Baud Rate} = \frac{\text{Timer 1 Overflow Rate}}{32} \qquad (2.5)$$

ถ้าต้องการค่าอัตรา Baud เท่ากับ 1,200 สามารถคำนวณค่าความถี่ค่าเกินของ Timer 1 ได้ดังนี้

$$1200 = \frac{\text{Timer 1 Overflow Rate}}{32} \qquad (2.6)$$

จะได้ Timer 1 Overflow Rate เท่ากับ 38 kHz

ถ้าระบบ MCS - 51 ใช้ความถี่สัญญาณนาฬิกาจากแร่กำเนิดความถี่ เท่ากับ 12 MHz ตัว Timer 1 จะได้รับ Clock เท่ากับ 1 MHz หรือ 1,000 kHz ถ้าเราต้องการ Timer 1 ค่าเกินเท่ากับ 38.4 kHz ดังนั้น ค่าอัตราค่าเกินมีค่าเท่ากับ 1,000/38.4 = 26.04 Clock โดยค่าเกินจะเกิดเมื่อเกิดการเปลี่ยนจาก FFH เป็น 00H ดังนั้นจะต้องให้ Timer 1 นับไป 26 Count ดังนั้นค่าที่จะให้รีจิสเตอร์ TH1 มีค่าเท่ากับ -26 ซึ่งใช้ค่าที่ย้อนกลับไปดังนั้นเขียนคำสั่งได้ดังนี้

```
MOV   TH1 , #-26
```

ตัวโปรแกรมแอสแซมเบลอร์ทั่วไปจะแปลงค่า -26 เป็น 0E6H เอง จากที่ผ่านมาจะเห็นว่าความถี่อัตรา Baud จะมีความสัมพันธ์กับค่าสัญญาณนาฬิกาที่ใช้ จากแร่กำเนิดความถี่ในตารางที่ 3 จะเป็นค่าที่ต้องกำหนดใน Timer 1 เมื่อต้องการค่าอัตรา Baud ต่างๆ จะเห็นว่าถ้าใช้ความถี่ 18,432 MHz ค่า อัตรา Baud

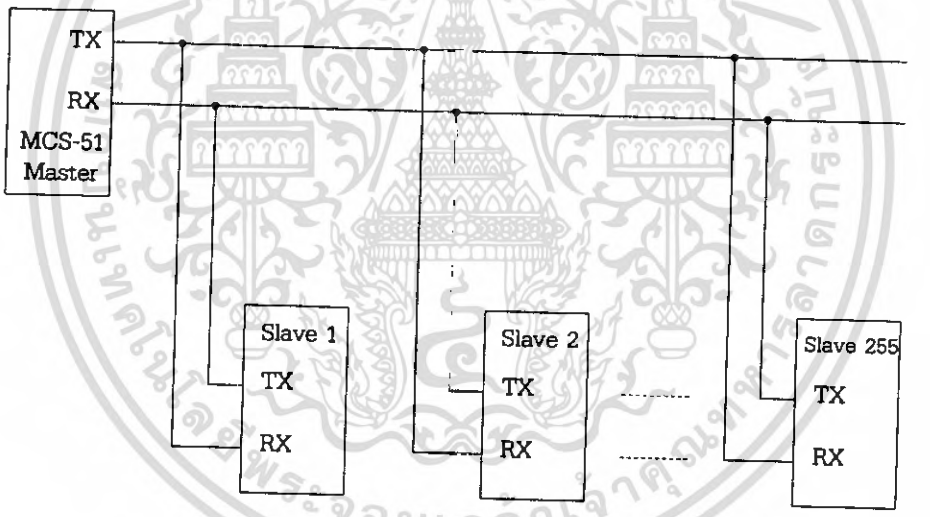
### 2.4.8 การสื่อสารข้อมูลระหว่าง MCS-51 หลายตัว

การใช้งานพอร์ตอนุกรมในโหมด 2 และโหมด 3 ของระบบ MCS - 51 สามารถนำมาใช้กับการสื่อสารของ MCS-51 หลายๆ ตัวได้ ที่เรียกกันว่า การสื่อสาร แบบระบบมัลติโปรเซสเซอร์ (Multiprocessor

System) โดยในระบบจะมี MCS-51 ตัวหนึ่งเป็นตัวควบคุมการทำงานหลักเรียกว่ามาสเตอร์ (Master) ส่วนตัวอื่นๆ ที่ต่ออยู่ในระบบจะเรียกว่าสลาฟ (Slave) โดยตัวสลาฟแต่ละตัวจะถูกกำหนดค่าแอดเดรสประจำตัวของมันถ้าข้อมูลที่ส่งมีค่าแอดเดรสตรงกับสลาฟตัวใดก็จะทำให้สลาฟตัวนั้นสามารถรับข้อมูลได้

เมื่อเริ่มต้นการทำงาน MCS-51 ทุกตัวจะถูกโปรแกรมให้พอร์ทอนุกรมทำงานในโหมดเดียวกัน และ MCS-51 ตัวที่เป็นสลาฟนั้นบิต SM2 จะต้องถูกเซตเป็น "1" ซึ่งจะทำให้สลาฟแต่ละตัวไม่มีการรับส่งข้อมูลซึ่งกันและกัน เมื่อตัวมาสเตอร์ต้องการส่งข้อมูลให้สลาฟจะทำการส่งค่าแอดเดรสของตัวที่ต้องการติดต่อออกมาก่อน ซึ่งจะมีทั้งหมด 9 บิต โดยค่าบิต TB8 จะเป็น "1" เรียกค่าข้อมูล 9 บิตนี้ว่าบัสตำแหน่ง ส่วนตัวสลาฟทุกตัวพร้อมที่จะรับข้อมูลเข้ามาได้เนื่องจากบิต SM2 เป็น "1" เมื่อรับข้อมูลเข้ามาแล้วค่าในบิต RB8 จะมีค่าเป็น "1" จากนั้นตัวสลาฟจะอ่านค่าข้อมูลใน SBUF มาเปรียบเทียบว่าตรงกับแอดเดรสของตัวมันหรือไม่ ถ้าตัวใดตรงกันก็จะทำให้ SM2 เป็น "0" จากนั้นจะเป็นการรับส่งข้อมูลในโหมดปกติ

จะเห็นว่าการกำหนดแอดเดรสของ MCS-51 แต่ละตัวมีขนาด 1 ไบต์ ดังนั้นสามารถต่อตัวสลาฟได้ทั้งหมด 255 ตัว ดังรูปที่ 2.21



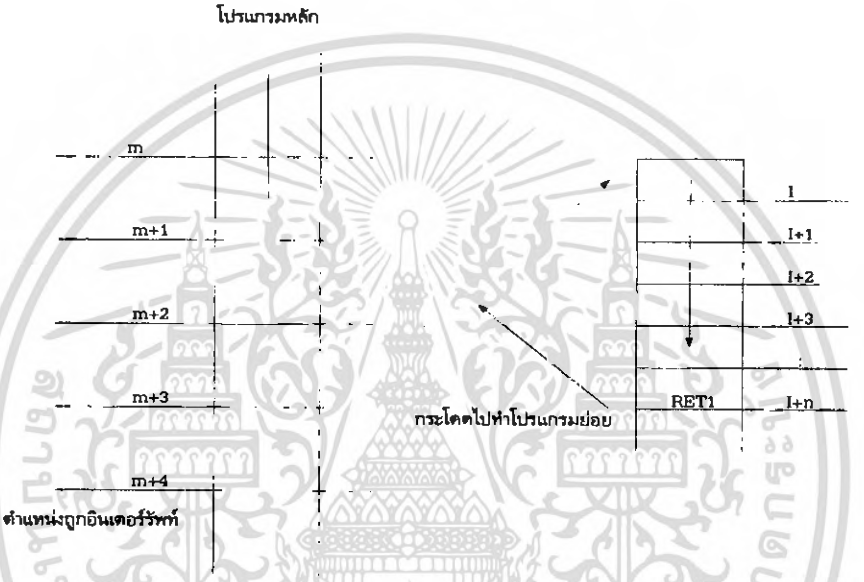

**รูปที่ 2.22** การต่อ MCS – 51 แบบ Multiprocessor

## 2.5 การอินเตอร์รัปต์

### 2.5.1 กระบวนการอินเตอร์รัปต์

ถ้าหากคอมพิวเตอร์กำลังทำงานโปรแกรมหลักอยู่ เมื่อมีการอินเตอร์รัปต์เข้ามาคอมพิวเตอร์จะหยุดโปรแกรมหลัก ไปทำงานโปรแกรมตอบสนองการอินเตอร์รัปต์ (Interrupt Service Routine) เมื่อทำโปรแกรมตอบสนองอินเตอร์รัปต์เสร็จ คอมพิวเตอร์จะกลับมาทำโปรแกรมเดิม

ถ้า CPU กำลังทำงานโปรแกรมหลักอยู่ เช่นกำลังทำคำสั่งในตำแหน่งของหน่วยความจำที่ m , m+1 m+2 ไปเรื่อยๆโดย PC จะชี้ที่ตำแหน่งที่จะอ่านคำสั่งถัดมา เมื่อโปรแกรมทำงานถึงตำแหน่งที่ m+3 แล้วเกิดการอินเตอร์รัปต์ขึ้น(ขณะนั้น PC อยู่ที่ m+4)โปรแกรมจะต้องทำงานโปรแกรมตอบสนองการอินเตอร์รัปต์โดยย้าย PC ไปที่ตำแหน่งที่เก็บโปรแกรมตอบสนองการอินเตอร์รัปต์ จากนั้นจะเก็บค่า PC เดิมในหน่วยความจำสแตค เมื่อคอมพิวเตอร์ทำงานโปรแกรมตอบสนองการอินเตอร์รัปต์เสร็จสิ้นลง จะคืนค่าในสแตค (m+4) ให้กับ PC เพื่อให้ PC ทำโปรแกรมหลักต่อไป



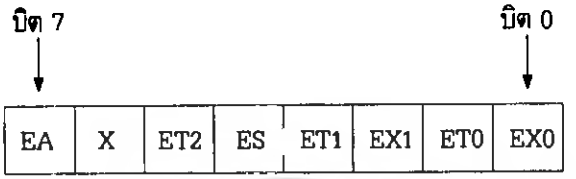

**รูปที่ 2.23** ขั้นตอนการทำงานของโปรแกรมเมื่อถูกอินเตอร์รัปต์

### 2.5.2 สัญญาณอินเตอร์รัปต์

แหล่งกำเนิดสัญญาณอินเตอร์รัปต์ที่ใช้กับ MCS-51 มีสองชนิดคือ อินเตอร์อินเตอร์รัปต์ภายในและภายนอก โดยอินเตอร์รัปต์ภายในจะเกิดขึ้นภายในตัว MCS-51 เอง ได้แก่สัญญาณจากไทเมอร์แฟลก 0 (TF0) ไทเมอร์แฟลก 1 (TF1) และพอร์ทอนุกรม สำหรับอินเตอร์รัปต์ภายนอกเกิดจากสัญญาณที่กระตุ้นเข้ามาทาง INT0 และ INT1 เมื่อมีสัญญาณอินเตอร์รัปต์จากแหล่งต่างๆ เข้ามาจะสามารถโปรแกรมได้ว่าจะให้ MCS-51 ยอมให้มีการอินเตอร์รัปต์ได้หรือไม่ โดยการโปรแกรมไปที่ รีจิสเตอร์ IE (Interrupt Enable) และถ้ามีสัญญาณอินเตอร์รัพท์มาจากแหล่งต่างๆ หลายแหล่งพร้อมกันก็จะสามารถจัดลำดับได้ว่า จะให้อินเตอร์รัปต์ใดเกิดก่อน โดยการโปรแกรมไปที่ อินเตอร์รัปต์ไพโอริตี้ IP (Interrupt Priority) รีจิสเตอร์ทั้งสองตัวมีรายละเอียดดังต่อไปนี้

**2.5.2.1 Interrupt Enables**

เป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ ใช้สำหรับกำหนดค่าว่าถ้าเกิดการอินเตอร์รัปต์จากแหล่งต่างๆ จะทำการอินเตอร์รัปต์เหล่านั้นหรือไม่ โดยรายละเอียดของบิตต่างๆ มีดังตารางที่ 2.5




**ตารางที่ 2.5** บิตต่างๆ ของรีจิสเตอร์ IE

| บิต | ชื่อบิต | ตำแหน่งบิต | รายละเอียด |
|---|---|---|---|
| IE.7 | EA | AFH | ถ้าเซตยอมให้มีการอินเตอร์รัปต์ |
| IE.6 | - | AEH | สงวนไว้ใช้งานในอนาคต |
| IE.5 | ET2 | ADH | Enable อินเตอร์รัปต์จาก Timer 2 (ใช้กับ 8052) |
| IE.4 | ES | ACH | Enable อินเตอร์รัปต์จากพอร์ทอนุกรม |
| IE.3 | ET1 | ABH | Enable อินเตอร์รัปต์จาก Timer 1 |
| IE.2 | EX1 | AAH | Enable อินเตอร์รัปต์จาก INT1 |
| IE.1 | ET0 | A9H | Enable อินเตอร์รัปต์จาก Timer 0 |
| IE.0 | EX0 | A8H | Enable อินเตอร์รัปต์จาก INT0 |

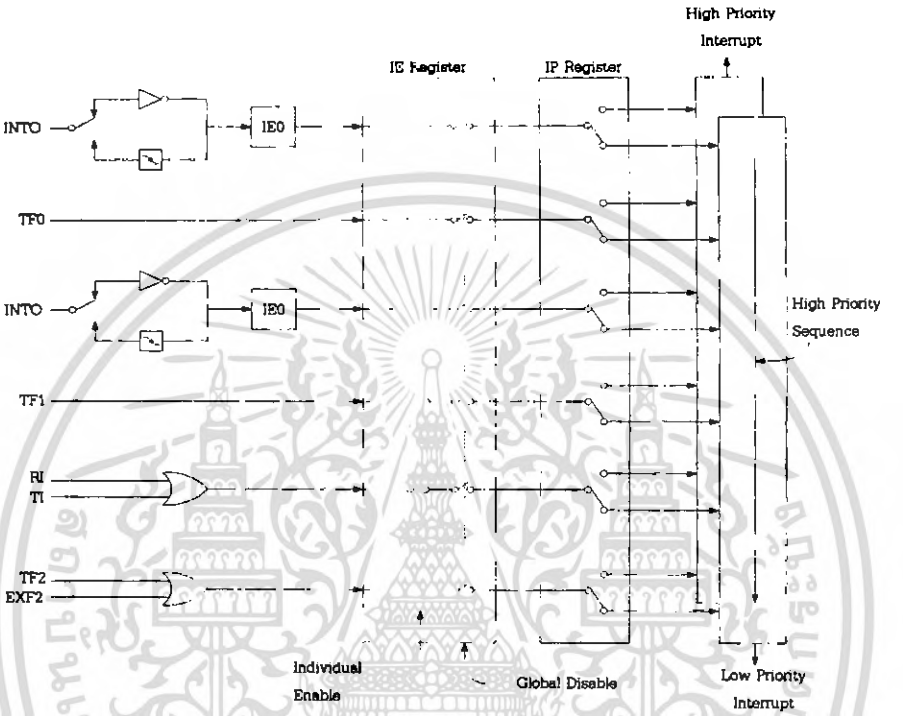
**2.5.2.2 Interrupt Priority**

เป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ ใช้ในการจัดลำดับความสำคัญของการอินเตอร์รัปต์ ซึ่งสามารถจัดได้สองลำดับ ถ้าเป็น "1" หมายความว่ามีความสูงสุดเป็น "0" หมายความว่ามีลำดับความสำคัญต่ำสุด ความหมายของบิตต่างๆ แสดงได้ดังตารางที่ 2.5 ถ้าหากกำหนดให้มีความสำคัญเป็น "1" เหมือนกันหมด MCS-51 จะจัดลำดับความสำคัญใหม่ดังนี้

**ตารางที่ 2.6** การจัดลำดับความสำคัญของการอินเตอร์รัปต์

| ลำดับ | อินเตอร์รัปต์ |
|---|---|
| 1 (สูงสุด) | IE0 |

| | |
|---|---|
| 2 | TFO |
| 3 | IE1 |
| 4 | TF1 |
| 5 (ต่ำสุด) | Serial Port |




**รูปที่ 2.24** รีจิสเตอร์ต่างๆ ที่เกี่ยวข้องกับการอินเตอร์รัปต์

**ตารางที่ 2.7** บิตและหน้าที่ต่างๆ ของรีจิสเตอร์ IP

| บิต | ชื่อบิต | ตำแหน่งบิต | รายละเอียด |
|---|---|---|---|
| IP.7 | - | - | ไม่ใช้งาน |
| IP.6 | - | - | ไม่ใช้งาน |
| IP.5 | PT2 | 0BDH | ใช้กับ Timer 2 (8052) |
| IP.4 | PS | 0BCH | ใช้กับพอร์ทอนุกรม |
| IP.3 | PT1 | 0BBH | ใช้กับ Timer 1 |
| IP.2 | PX1 | 0BAH | ใช้กับอินเตอร์รัปตจาก INT1 |
| IP.1 | PT0 | 0B9H | ใช้กับ Timer 2 |

| | | | |
|---|---|---|---|
| IP.0 | PX0 | 0B8H | ใช้กับอินเตอร์รัปตจาก INT0 |

จากตารางที่ 2.7 แสดงการอินเตอร์รัปต์จากแหล่งต่างๆ ที่มีผลกับ MCS-51 ถ้าเป็นเบอร์ 8051, 8031 จะถูกอินเตอร์รัปต์ได้ 5 แหล่ง ถ้าเป็นเบอร์ 8052 , 8032 จะถูกอินเตอร์รัปต์ได้ 6 แหล่ง โดยเพิ่มอินเตอร์รัปต์จาก Timer 2 ในรูปที่ 2.24 จะแสดงให้เห็นว่า ถ้า MCS-51 จะถูกอินเตอร์รัปต์ได้จะต้องเซตค่า Global Enable ในรีจิสเตอร์ IE นอกจากนี้ยังกำหนดได้ว่าจะให้อินเตอร์รัปต์ใดเกิดขึ้นได้ โดยการเซตค่า Interrupt Enable ของอินเตอร์รัปต์จากแหล่งต่างๆ ในรีจิสเตอร์ IE จากรูปยังแสดงให้เห็นอีกว่าเมื่อมีการอินเตอร์รัปต์เข้าจะมีผลต่อแฟลกใด เช่นถ้า INT0 เป็น "1" บิต IE0 จะเป็น "1" หมายความว่าถูกอินเตอร์รัปต์ โดยแฟลกต่างๆ ที่มีผลจากการถูกอินเตอร์รัปต์แสดงได้ดังตารางที่ 2.8

**ตารางที่ 2.8** แฟลกที่จะทำงานเมื่อถูกอินเตอร์รัปต์

| อินเตอร์รัปต์ | แฟลก | ประกอบอยู่ในรีจิสเตอร์ |
|---|---|---|
| External 0 | IE0 | TCON.1 |
| External 1 | IE1 | TCON.3 |
| Timer 1 | TF1 | TCON.7 |
| Timer 0 | TF0 | TCON.5 |
| Serial port | T1 | SCON.1 |
| Serial port | RI | SCON.0 |
| Timer 1 | TF2 | T2CON.7 (8052) |
| Timer 2 | EXF2 | T2CON.6 (8052) |

จากตารางจะเห็นว่า ถ้ามีการอินเตอร์รัปต์จากภายนอกเข้า ตัวที่จะอินเตอร์รัปต์ MCS-51 คือ บิตแฟลก IE0 ซึ่งอยู่ในรีจิสเตอร์ TCON ถ้ามีการสื่อสารแบบอนุกรม เมื่อข้อมูลถูกส่งไปหมดแล้วจะอินเตอร์รัปต์ MCS-51 ทางบิตแฟลก TI ถ้ารับข้อมูลหมดแล้วจะอินเตอร์รัปต์ MCS-51 ทางบิตแฟลก RI ซึ่งอยู่ในรีจิสเตอร์ SCON และถ้าใช้ Timer 0 ในการนับเมื่อเกิดค่าเกินสามารถอินเตอร์รัพท์ MCS-51 ได้ทางบิต TF0

### 2.5.3 การทำงานของระบบหลังถูกอินเตอร์รัปต์

เมื่อ MCS-51 ถูกอินเตอร์รัปต์ จะต้องกระโดดไปทำโปรแกรมตอบสนองการอินเตอร์รัปต์โดยตำแหน่งที่กระโดดไปเรียกว่า อินเตอร์รัปต์เวกเตอร์ (Interrupt Vectors) เมื่อทำการโปรแกรมตอบสนองการ

อินเตอร์รัปต์เรียบร้อยแล้ว MCS-51 จะกระโดดมาทำงานยังตำแหน่งเดิมโดยก่อนที่จะกระโดดไปทำโปรแกรมตอบสนองการอินเตอร์รัปต์จะต้องเก็บค่าตำแหน่งเดิมไว้ โดยเก็บค่า PC ลงหน่วยความจำสแตคซึ่งอยู่ที่หน่วยความจำที่ถูกชี้โดยรีจิสเตอร์ SP เมื่อทำโปรแกรมตอบสนองการอินเตอร์รัปต์เสร็จแล้วจะคืนค่าในหน่วยความจำสแตคให้ PC ตามเดิม ค่าอินเตอร์รัปต์เวคเตอร์ของ MCS-51 แสดงได้ดังตารางที่ 2.9

**ตารางที่ 2.9** อินเตอร์รัปต์เวกเตอร์ของMCS-51

| อินเตอร์รัปต์ | อินเตอร์รัปต์เวกเตอร์ |
|---|---|
| System Reset | 0000H |
| External 0 | 0003H |
| Timer 0 | 000BH |
| External 1 | 0013H |
| Timer 1 | 001BH |
| Serial Port | 0023H |
| Timer 2 | 002BH |

จากตารางจะเห็นว่าถ้าระบบถูกอินเตอร์รัปต์จากภายนอกทาง INT0 ตัว MCS-51 จะกระโดดไปทำงานที่ตำแหน่ง 0003H ถ้าระบบถูกอินเตอร์รัปต์จาก Timer 0 จะกระโดดไปทำงานตำแหน่ง 000BH

### 2.5.4 การออกแบบโปรแกรมอินเตอร์รัปต์

การเขียนโปรแกรมหลัก (Main Program) จะต้องกำหนดค่าให้ MCS-51 ถูกอินเตอร์รัปต์ด้วยอะไร และจะให้ MCS-51 ถูกอินเตอร์รัปต์ได้หรือไม่ โดยการโปรแกรมค่าต่างๆ ใน IE รีจิสเตอร์ ถ้ามีการอินเตอร์รัปต์จากสองแหล่งขึ้นไปควรมีการจัดลำดับความสำคัญในรีจิสเตอร์ IP ดังนั้นในโปแกรมหลักจะต้องมีการโปรแกรมดังนี้

1) โปรแกรมค่าในรีจิสเตอร์ IE

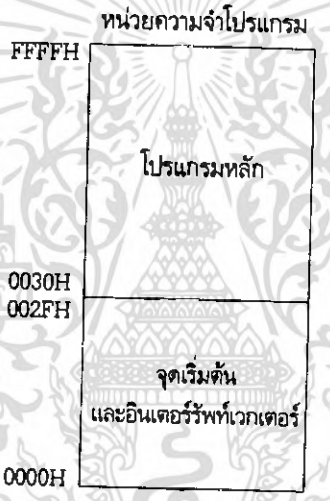
2) โปรแกรมค่าในรีจิสเตอร์ IP

สำหรับโปรแกรมการตอบสนองการอินเตอร์รัปต์ถือว่าเป็นโปรแกรมย่อยโปรแกรมหนึ่ง แต่จะต้องจบโปรแกรมย่อยด้วยคำ RETI (Return from Interrupt)

จากตารางอินเตอร์รัปต์เวคเตอร์ จะเห็นว่าถ้ากดรีเซตหรือให้ระบบเริ่มทำงาน โปรแกรมจะเริ่มทำงานที่ตำแหน่ง 0000H และตำแหน่งที่เก็บโปรแกรมหลักมีโอกาสที่จะทับกับหน่วยความจำโปรแกรมที่เก็บค่าอินเตอร์รัปต์เวกเตอร์ที่ตำแหน่ง 0003H ถ้าโปรแกรมยาวมากอาจจะไปทับตำแหน่ง 000BH ได้ซึ่งเป็น

ตำแหน่งของอินเตอร์รัปต์เวกเตอร์ของ Timer 0 ดังนั้นในการเขียนโปรแกรมหลัก ภายใน 3 ตำแหน่งแรกคือ 0000H,0001H,0002H จะต้องกระโดดไปที่อื่นก่อนเพื่อให้ข้ามอินเตอร์รัปต์เวกเตอร์ไปซึ่งอาจเขียนโปรแกรมได้ดังนี้

```
        ORG   0000H       ; เริ่มต้นโปรแกรม
        LJMP  MAIN        ; กระโดดไปโปรแกรมหลัก
        .....             ; เพื่อหนีอินเตอร์รัปต์เวกเตอร์
        ORG   0030H       ; ตำแหน่งเริ่มต้นของโปรแกรม
MAIN:   .....             ; เริ่มต้นโปรแกรมหลัก
        .....
```




**รูปที่ 2.25** การจัดตำแหน่งโปรแกรมในหน่วยความจำ

จากตัวอย่างโปรแกรมจะเห็นว่า เมื่อเริ่มต้นโปรแกรมหรือระบบถูกรีเซต ระบบจะทำงานตำแหน่งแรกคือคำสั่งกระโดดไปโปรแกรมหลัก ซึ่งอยู่ต่อจากโปรแกรมตอบสนอง โดยการอินเตอร์รัปต์ที่อยู่ที่ตำแหน่ง 0030H

**1. โปรแกรมการตอบการอินเตอร์รัปต์แบบสั้น**

จากตารางอินเตอร์รัปต์เวกเตอร์ จะเห็นว่าที่เก็บโปรแกรมอินเตอร์รัปต์แต่ละแห่งจะห่างกัน 8 ไบต์ ดังนั้นถ้ามีการอินเตอร์รัปต์จากแหล่งต่าง ๆ หลายแหล่งและโปรแกรมตอบสนองการอินเตอร์รัปต์บางโปรแกรมมีขนาดยาวเกิน 8 ไบต์ จะทำให้โปรแกรมจะไปทับกับตำแหน่งของโปรแกรมตอบสนองการอินเตอร์

รัปต์ของอินเตอร์รัปต์ถัดไป แต่ถ้าโปรแกรมการตอบสนองการอินเตอร์รัปต์ไม่ยาวมากเกินไปสามารถเขียนไปในตำแหน่งนั้นได้เลยดังโปรแกรมต่อไปนี้

```
ORG    0000H
LJMP   MAIN          ; กระโดดไปโปรแกรมหลัก
ORG    000BH         ; ตำแหน่งเริ่มต้นของอินเตอร์รัปต์ Timer 0
TOISR  :.....
RETI                 ; กลับโปรแกรมหลัก
MAIN   :.....        ; โปรแกรมหลัก
```

จากตัวอย่างโปรแกรมจะใช้อินเตอร์รัปต์จาก Timer 0 เมื่อระบบเริ่มทำงานจะทำตำแหน่ง 0000H โดยกระโดดไปโปรแกรมหลักซึ่งอยู่ที่ตำแหน่งต่อจากโปรแกรมตอบสนองการอินเตอร์รัปต์เมื่อมีการอินเตอร์รัปต์เวกเตอร์ Timer 0 ระบบจะทำโปรแกรมตำแหน่งที่ 00BH ซึ่งเป็นอินเตอร์รัปต์เวกเตอร์ของ Timer 0 โดยโปรแกรมตอบสนองการอินเตอร์รัปต์จะจบด้วยคำสั่ง RETI เพื่อกลับสู่โปรแกรมหลักต่อไป

**2. โปรแกรมตอบสนองการอินเตอร์รัปต์ขนาดใหญ่**

ในกรณีที่มีการอินเตอร์รัปต์จากหลายแหล่ง และโปรแกรมตอบสนองการอินเตอร์รัปต์แต่ละโปรแกรมยาวเกิน 8 ไบต์ จะไม่สามารถเขียนโปรแกรมตอบสนองการอินเตอร์รัปต์เวกเตอร์ได้ ซึ่งจะแก้ปัญหานี้ได้โดยกำหนดให้ตำแหน่งของอินเตอร์รัปต์เวกเตอร์ให้ทำโปรแกรมกระโดด โดยกระโดดไปที่ตำแหน่งเก็บโปรแกรมตอบสนองการอินเตอร์รัปต์เขียนไว้ที่ตำแหน่งอื่นดังตัวอย่างต่อไปนี้

```
ORG    0000H         ; เริ่มโปรแกรมของระบบ
LJMP   MAIN          ; กระโดดไปโปรแกรมหลัก
ORG    000BH         ; ตำแหน่งของอินเตอร์รัปต์ Timer 0
LJMP   LED1          ; กระโดดไปโปรแกรมตอบสนองการอินเตอร์รัปต์ชื่อ
                       LED1
ORG    0030H         ; ตำแหน่งหลังอินเตอร์รัปต์เวกเตอร์
MAIN:  .....         ; โปรแกรมหลัก
.....
LED:   .....         ; โปรแกรมการตอบสนองการอินเตอร์รัปต์ Timer 1
.....
RETI                 ; กลับสู่โปรแกรมหลัก
```

จากโปรแกรมจะเห็นว่า เมื่อระบบทำงานจะต้องทำที่ตำแหน่ง 0000H โดยกระโดดไปทำโปรแกรมหลักที่ตำแหน่งต่อจาก 0030H เพราะตำแหน่งดังกล่าวข้ามอินเตอร์รัปต์เวกเตอร์จากตำแหน่งต่างๆ ไปแล้วเมื่อมีการอินเตอร์รัปต์จาก Timer 0 โปรแกรมจะต้องทำงานที่ตำแหน่ง 000BH แต่โปรแกรมตอบสนองการอินเตอร์รัปต์ยาวมากที่ตำแหน่ง 000BH จึงทำให้โปรแกรมกระโดดไปที่โปรแกรมตอบสนองการอินเตอร์รัปต์ชื่อ LED1 ซึ่งอยู่ท้ายโปรแกรม เมื่อจบโปรแกรมจะจบด้วยคำสั่ง RETI เพื่อกลับไปโปรแกรมหลักต่อไป

## 2.6 การเขียนโปรแกรมด้วยภาษาซี

### 2.6.1 โปรแกรม C51

โปรแกรม C51 เป็นซีคอมไพเลอร์ที่ออกแบบมาสำหรับการเขียนภาษาซี กับไมโครคอนโทรลเลอร์เบอร์ MCS-51 โดยทั่วไปนักพัฒนาไมโครคอนโทรลเลอร์ที่เขียนด้วยภาษาแอสเซมบลีจะเขียนโปรแกรมและแปลให้เป็นออปเจกต์ไฟล์ (นามสกุล .OBJ) หรือเฮ็กไฟล์ (นามสกุล .HEX) จากนั้นจะนำโปรแกรมลงบนชิพไมโครคอนโทรลเลอร์ แต่การเขียนโปรแกรมด้วยภาษาซีจะมีขั้นตอนต่างๆ มากขึ้นดังนี้

1. เขียนชุดคำสั่งบนโปรแกรม Editor ให้มีนามสกุลเป็น .C
2. กำหนดพรีโพรเซสเซอร์ (Preprocessor) สำหรับกำหนดการคอมไพล์โปรแกรม ซึ่งอาจกำหนดไว้ใน Source Code หรือกำหนดตอนคอมไพล์ก็ได้
3. คอมไพล์โปแกรม ในที่นี้จะใช้โปรแกรม C51 ในการคอมไพล์ และจะได้ไฟล์ออปเจกต์ ที่มีนามสกุลเป็น .OBJ ออกมา
4. เชื่อมต่อชุดคำสั่งเสริม (Link) โดยใช้โปรแกรม L51 ซึ่งจะทำการรวมชุดคำสั่งต่างๆ ที่อ้างอิงถึงกัน และโปรแกรมจะสร้างออปเจกต์ที่สมบูรณ์ออกมา
5. แปลงเป็นไฟล์ Hexadecimal File ด้วยโปรแกรม OHS51 ซึ่งจะได้ไฟล์นามสกุล HEX ออกมาถ้าหากใช้คอมไพเลอร์ตัวอื่นๆ ขั้นตอนเหล่านี้อาจแตกต่างกันไป

### 2.6.2 คำสั่งพื้นฐานและการประกาศตัวแปร

```
# include<stdio.h>
main( )
{
        Printf ( " Hello Program C \ n" );
}
```

เมื่อรันโปรแกรมคอมพิวเตอร์จะแสดงว่า Hello Program C ออกทางจอภาพ จากโปรแกรมจะพบว่าบรรทัดแรกจะเป็นการเรียกฟังก์ชันที่ใช้ติดต่อกับไอโอมาตรฐานออกมา จากนั้นก็เรียกใช้ printf ที่เก็บอยู่ใน Stdio.h นั้น สำหรับการเขียนโปรแกรมกับไมโครคอนโทรลเลอร์การทำงานต่างๆ จะเป็นการกระทำกับพอร์ต รีจิสเตอร์ และหน่วยความจำต่างๆ ดังนั้นผลลัพธ์จากการทำงานของโปรแกรมจะเกี่ยวข้องกับการย้ายข้อมูลระหว่างรีจิสเตอร์ และหน่วยความจำต่างๆเป็นส่วนใหญ่

ถ้าหากต้องการเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์ส่งค่าตั้งแต่ 0 ถึง 255 ออกไปทางพอร์ต P1 สามารถเขียนโปรแกรมด้วยภาษาแอสเซมบลีได้ดังนี้

```
        ORG   0000H
        MOV   R1  #00D
LOOP :  MOV   P1 , R1
        INC   R1
        CJNE  R1 #255D , LOOP
```

แต่ถ้าหากเขียนโปรแกรมเป็นภาษาซีจะได้เป็น

```
#include < reg 52.h >
Main ( )
{
    Unsigned char i;
    For ( I = 0;I < = 255; i++)
    P1 = i
}
```

ในบรรทัดแรกจะคล้ายกับบรรทัดแรกของการเขียนโปรแกรมบนเครื่องคอมพิวเตอร์ PC โดยจะบอกให้คอมไพเลอร์รู้จักตัวแปรและรีจิสเตอร์ต่างๆ ของ MCS-51 โดยในไฟล์ reg52.h จะบรรจุชื่อจีจิสเตอร์และหน่วยความจำต่างๆ ของ 8052 เอาไว้

บรรทัดที่สองจะเป็นชื่อฟังก์ชันการทำงานหลัก (main) โดยคำสั่งการทำงานต่างๆ จะอยู่ในเครื่องหมายปีกกาเปิดและปีกกาปิด ต่อมาจะเป็นการประกาศตัวแปร i ให้เป็นตัวแปรแบบ char ไม่คิดเครื่องหมายซึ่งจะเก็บข้อมูลได้ตั้งแต่ 0 ถึง 255 บรรทัดต่อไปจะเป็นประโยคคำสั่งวนลูปซึ่งจะทำให้โปรแกรมทำงานต่อเนื่องโดยนำค่า 0 ถึง 255 ส่งออกไปทางพอร์ต P1 ครั้งละค่า

การเขียนโปรแกรมด้วยภาษาซีนี้สามารถเขียนไดเร็กทีฟควบคุม (Control Directive) ได้ซึ่งจะเป็นการบอกว่าจะให้ทำการใดๆ ขณะคอมไพล์โปรแกรม โดยอาจเขียนตอนคอมไพล์ หรือเขียนภายในโปรแกรมก็ได้ โดยการกำหนดในส่วนของ Preprocess ซึ่งจะใช้คำว่า #pragma นำหน้าส่วนของไดเร็กทีฟควบคุมนั้นมีอยู่หลายคำสั่งสำหรับรายละเอียดต่างๆ จะกล่าวในภายหลัง สำหรับคำสั่งที่ใช้กันมากที่สุดคือคำสั่ง code ซึ่งจะบอกว่าให้สร้างภาษาแอสเซมบลีขณะที่ทำการแปลโปรแกรมด้วย

### 2.6.3 โครงสร้างของภาษาซี

ด้วยภาษซีเป็นภาษาที่สามารถเขียนโปรแกรมเป็นแบบโครงสร้างได้ โดยโปรแกรมจะแบ่งการทำงานต่างๆ ออกเป็นกลุ่มๆ หรือฟังก์ชั่น โดยฟังก์ชั่นเหล่านั้นสามารถเรียกขึ้นมาใช้ใหม่ได้ ในการเขียนโปรแกรมจะต้องระบุไว้ว่าในโปรแกรมนั้นมีฟังก์ชันใดให้ใช้บ้าง แต่ทุกโปรแกรมจะต้องมีฟังก์ชันหลักที่มีชื่อว่า main ( ) เสมอ

#### 2.6.3.1 ตัวแปรและค่าคงที่

การใช้งานตัวแปรและค่าคงที่ต่างๆ จะต้องมีการประกาศชื่อตัวแปรขึ้นมาเสียก่อนเมื่อมีการคอมไพล์โปรแกรมตัวคอมไพล์เลอร์จะเตรียมพื้นที่ในหน่วยความจำแรมเอาไว้สำหรับเก็บตัวแปรและค่าคงที่เหล่านั้นในการประกาศตัวแปรสามารถทำได้ดังนี้

ประเภทของข้อมูล ชื่อตัวแปร [,.....] ;

การประกาศตัวแปรจะต้องเริ่มด้วยชื่อประเภทของข้อมูล และตามด้วยชื่อตัวแปรโดยจะประกาศครั้งละกี่ตัวก็ได้ ส่วนชื่อประเภทของข้อมูล และตามด้วยชื่อตัวแปรโดยจะประกาศครั้งละกี่ตัวก็ได้ ส่วนชื่อของตัวแปรนั้นจะซ้ำกันไม่ได้และต้องไม่ซ้ำกับชื่อของคำสงวน (Keywords) ของคอมไพเลอร์ตัวนั้นๆ สำหรับประเภทของข้อมูลในการประกาศตัวแปรเป็นดังนี้

**ตารางที่ 2.10** การประกาศตัวแปรของภาษาซี

| ประเภทของข้อมูล | ขนาด (บิต) | ค่าที่เก็บได้ |
|---|---|---|
| bit | 1 | 0 ถึง 1 |
| char | 8 | 0 ถึง 255 |
| Unsigned char | 8 | 0 ถึง 255 |
| int | 16 | -32768 ถึง 32767 |
| Unsigned int | 16 | 0 ถึง 65535 |
| long | 32 | -2147483648 ถึง +2147483647 |
| Unsigned long | 32 | 0 ถึง 4294967295 |
| float | 32 | -1.17549e -38 ถึง e +3.402823 +38 |

สำหรับคำสงวนเป็นคำที่คอมไพเลอร์รู้จักและถูกใช้งานเฉพาะ ไม่สามารถนำมาตั้งชื่อตัวแปรและฟังก์ชันได้ คำสงวนของโปรแกรม C51 เป็นดังต่อไปนี้

| | | |
|---|---|---|
| _at_ | idata | sfr |
| alien | interrupt | sfr18 |
| bdata | large | small |
| bit | pdata | _task_ |
| code | _priority_ | using |
| compact | reentrant | xdata |
| data | sbit | |

### 2.6.4 ตัวดำเนินการในภาษาซี

ตัวดำเนินการจะเป็นตัวที่ใช้กระทำกับตัวแปร ค่าคงที่ต่างๆ ให้รวมเป็นค่าเดียวกันโดยอาจกระทำทางคณิตศาสตร์หรือกระทำทางลอจิกก็ได้ ในการเขียนโปรแกรมด้วยภาษาซีนั้น ตัวดำเนินการจะแบ่งออกเป็นสองกลุ่มใหญ่ๆ คือตัวดำเนินการกระทำกับตัวถูกกระทำตัวเดียว (Single Operand Operators หรือ unary operator) และตัวดำเนินการ ที่กระทำกับตัวถูกกระทำสองตัว (Two Operands Operators หรือ binary operator)

**1. ตัวดำเนินการกระทำกับตัวถูกกระทำตัวเดียว**

ตัวดำเนินการประเภทนี้จะกระทำกับตัวถูกกระทำเพียงตัวเดียว ประกอบด้วยตัวดำเนินการต่างๆ ดังนี้

| | |
|---|---|
| - | ลบ ( Negate ) |
| ~ | กลับค่าลอจิกของบิตข้อมูล (Bit Wise Complement) |
| ! | กลับค่าทางลอจิก (Logic Complement) |
| ++ | เพิ่มค่าขึ้นหนึ่งค่า (Increment) |
| -- | ลดค่าลงหนึ่งค่า (Decrement) |
| * | ตัวดำเนินการทางพอยน์เตอร์ |
| & | ตำแหน่งหน่วยความจำของตัวแปร |

**2. ตัวดำเนินการที่กระทำกับตัวถูกกระทำสองตัว**

ตัวดำเนินการประเภทนี้จะกระทำกับตัวถูกกระทำสองตัว ถ้าหากมีตัวถูกกระทำหลายๆ ตัวสามารถนำมาเขียนรวมกันเป็นประโยคได้ ซึ่งประกอบไปด้วยตัวดำเนินการที่ใช้กำหนดค่า ตัวดำเนินการทดสอบค่าซึ่งจะให้ผลลัพธ์เป็นค่าทางลอจิก (จริง, เท็จ) ตัวดำเนินการทางคณิตศาสตร์ และตัวดำเนินการทางลอจิก ดังต่อไปนี้

| | |
|---|---|
| = | กำหนดค่าในประโยค (Assignment) |
| + | บวก |
| - | ลบ |
| * | คูณ |
| / | หาร |
| % | เศษที่เหลือจากการหาร (Modulo) |
| && | การแอนด์ (Logical AND) |
| ll | การออร์ (Logical OR) |
| & | การแอนด์แบบบิตต่อบิต (Bit Wise AND) |
| ! | การออร์แบบบิตต่อบิต (Bit Wise OR) |
| ^ | การเอ็กคลูซีฟออร์แบบบิตต่อบิต (Bit Wise Exclusive OR) |
| << | เลื่อนบิตไปทางซ้าย |
| >> | เลื่อนบิตไปทางขวา |
| = = | ทดสอบว่าเท่ากันหรือไม่ |
| != | ทดสอบว่าไม่เท่ากันหรือไม่ |
| > | ทดสอบว่ามากกว่าหรือไม่ |
| < | ทดสอบว่าน้อยกว่าหรือไม่ |
| >= | ทดสอบว่ามากกว่าหรือเท่ากับหรือไม่ |
| <= | ทดสอบว่าน้อยกว่าหรือเท่ากับหรือไม่ |

นอกจากนี้ตัวดำเนินการบางประเภทสามารถนำมารวมกันเป็น Compound Operators ได้ ตัวอย่างเช่น ถ้ามีประโยค y = y* 2; อาจเขียนตัวดำเนินการรวมเป็น y*= 2 ; รูปแบบของตัวดำเนินการที่รวมกันได้เป็นดังนี้

| | |
|---|---|
| + = | บวกและให้เท่ากับบวกด้วย |
| - = | ลบและให้เท่ากับลบด้วย |
| * = | หารและให้เท่ากับหารด้วย |
| %= | หารแบบบวกและให้เท่ากับ |

& = ทำการแอนด์และให้เท่ากับ

l= ทำการออร์และให้เท่ากับ

^= ทำการเอ็กคลูซีฟออร์และให้เท่ากับ

<<= เลื่อนบิตไปทางซ้ายและไม่เท่ากับ

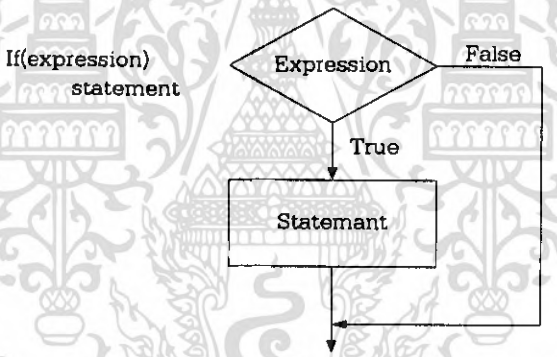
>>= เลื่อนบิตทางขวาและไม่เท่ากับ

### 2.6.5 ประโยคควบคุมในภาษาซี

การทำงานของโปรแกรมนั้นจะทำคำสั่งแต่ละคำสั่งเรียงลำดับกันไป และเราสามารถให้โปรแกรมตัดสินใจในการเลือกได้ หรือให้ทำงานใดงานหนึ่งซ้ำๆ ตามเงื่อนไขที่กำหนดได้โดยใช้คำสั่งควบคุมในภาษาซี นั้นจะมีประโยคคำสั่งควบคุมที่ใช้ในการเลือกทำและทำงานซ้ำๆ ดังนี้

**1. ประโยค IF / ELSE**

ประโยคคำสั่งนี้จะใช้ควบคุมทิศทางการทำงานของโปรแกรมโดยจะถูกแปลออกมาเป็นคำสั่งในภาษาแอสเซมบลีดังนี้ jz , jnz , jb , jnb , jc และ jnc โดยมีรูปแบบประโยคคำสั่งดังนี้

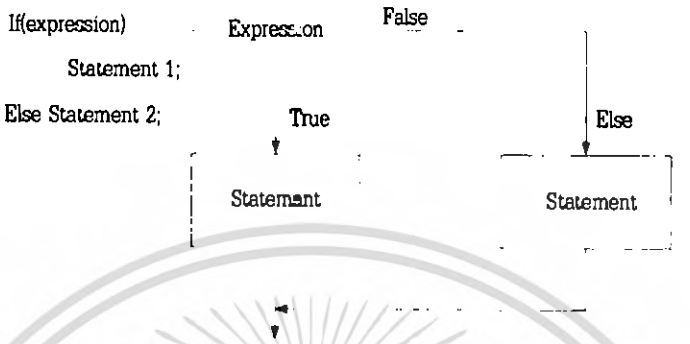



**รูปที่2.26** ประโยคทดสอบการทำงานของสเตตเมนต์แบบทางเลือกเดียว

ประโยคนี้ใช้ในการทดสอบว่าจะทำสเตตเมนต์ที่ตามมาหรือไม่ ถ้าค่าใน Expression เป็นจริงหรือมีค่าไม่เป็นศูนย์จะทำสเตตเมนต์ที่ตามมา ถ้าเป็นเท็จหรือมีค่าเป็นศูนย์จะไม่ทำ และสเตตเมนต์ที่จะทำงานนั้นอาจเป็นประโยคคำสั่งประโยคเดียว หรือเป็นสเตตเมนต์ซ้อนก็ได้ (ต้องมีปีกกาคลุม) ตัวอย่างเช่น ถ้าค่าของพอร์ต P1 มีค่าไม่เป็น 0 ให้ตัวแปร C เป็นศูนย์ จะเขียนได้ดังนี้

```
If (P1 != 0)
C = 0
```

ถ้าหากเป็นการทำงานเลือกทำแบบมีสองทางเลือก และต้องการทำงานเพียงอย่างใดอย่างหนึ่งจะใช้ประโยค if – else ซึ่งมีรูปแบบดังนี้




**รูปที่ 2.27** ประโยคทดสอบการทำงานของสเตตเมนต์แบบ 2 ทางเลือก

**2. ประโยค Switch**

การเลือกทำที่มีทางเลือกหลายๆ ทางเลือกนั้นเราสามารถนำประโยค if – else มาซ้อนกันได้ แต่จะทำให้มองดูเข้าใจได้ยาก ในภาษาซีมีประโยค switch ที่ใช้ในการเลือกทำอย่างใดอย่างหนึ่งจากหลายๆ ทางเลือกโดยมีรูปแบบของประโยคดังนี้

```
switch(k)
{
        Case 1:     statement 1:
                    break;
        case 2:     statement 2:
                    break;
        case 3:     statement 3
                    break;
              :           :
              :           :
        default:    statement n;
}
```

การทำงานของโปรแกรมจะนำค่าในตัวแปร k ที่อยู่ในวงเล็บหลัง switch มาเปรียบเทียบว่าเท่ากับค่าคงที่หลังคำสั่ง case ตัวใด และจะทำงานสเตตเมนต์ที่ตามหลัง case นั้น และจะออกนอกปีกกาของ switch เมื่อพบคำสั่ง break โดยสเตตเมนต์ที่ทำงานนั้นจะเป็นสเตตเมนต์ซ้อนก็ได้ แต่ถ้าค่าในตัวแปร k ไม่เท่ากับค่าคงที่ค่าใดเลยโปรแกรมจะทำสเตตเมนต์ที่ตามหลัง default

### 2.6.6 การทำซ้ำ

คำสั่งให้โปรแกรมทำงานซ้ำถือว่าเป็นประโยคคำสั่งควบคุมอย่างหนึ่ง การทำซ้ำหรือที่เรียกว่าการทำลูปนั้นจะมีประโยคคำสั่งอยู่สามประเภทคือ for, while และ do-while ซึ่งแต่ละแบบจะต่างกันตรงเงื่อนไขของการทำซ้ำ

#### 2.6.6.1 ประโยค for

ประโยคคำสั่งนี้จะใช้กรณีที่มีจำนวนรอบของการทำซ้ำที่แน่นอน โดยมีรูปแบบดังนี้

```
For (initialization ;condition ; increment)
        Statement ;
```

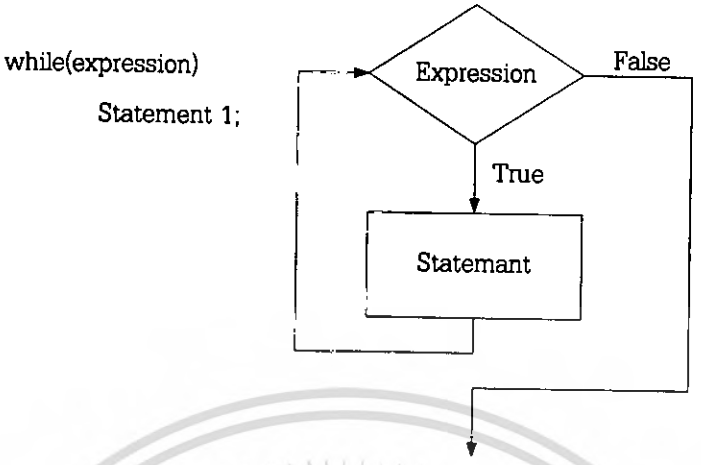
โดยที่ initialization เป็นค่ากำหนดเริ่มต้นให้กับตัวแปรของการทำลูป condition เป็นเงื่อนไขที่ใช้ทดลองการทำซ้ำครั้งต่อไป ซึ่งจะเป็นการกระทำลอจิก increment เป็นการเพิ่มค่าให้ตัวแปร ในการทำซ้ำแต่ละครั้ง สำหรับ statement จะเป็นสเตตเมนต์ของคำสั่งที่จะทำซ้ำ ซึ่งอาจเป็น สเตตเมนต์รวมก็ได้ ตัวอย่างเช่น

```
Unsigned char x;
For (x = 0; x < = 255; x++)
        P1 = x;
```

การทำงานตามชุดคำสั่งข้างบนจะเป็นการส่งค่า 0 ถึง 255 ออกมาทางพอร์ต P1 โดยเริ่มต้นจะประกาศตัวแปร x สำหรับนับการทำซ้ำ และให้ x เท่ากับ 0 ต่อมาส่ง x ออกทาง P1 และตรวจสอบว่า x น้อยกว่าหรือเท่ากับ 255 จริงหรือไม่ ถ้าจริงให้เพิ่มค่า x ขึ้นอีกหนึ่ง (x++) และทำซ้ำเรื่อยๆ

#### 2.6.6.2 ประโยค while

การทำซ้ำแบบนี้จะตรวจสอบเงื่อนไขก่อนการทำซ้ำ ถ้าเงื่อนไขก่อนการทำซ้ำ ถ้าเงื่อนไขเป็นจริงจะทำสเตตเมนต์ที่กำหนด และทดสอบเงื่อนไขใหม่ ถ้าเงื่อนไขเป็นเท็จจะออกจากการทำซ้ำทันที โดยมีรูปแบบดังนี้




**รูปที่ 2.28** ประโยคตรวจสอบเงื่อนไขก่อนการทำซ้ำ

ในส่วนของ Expression นั้นสามารถตรวจสอบค่าคงที่ได้ด้วย ถ้าค่าคงที่ได้ด้วย ถ้าค่าไม่เท่ากับศูนย์จะทำซ้ำ ถ้าค่าเท่ากับศูนย์จะไม่ทำซ้ำ ตัวอย่างเช่น ถ้าหากต้องการให้ค่าลอจิกทางพอร์ต P1 ของ MCS-51 ทุกบิตมีค่าลอจิกกลับไปมาจะเขียนคำสั่งได้ดังนี้

```
While (1)
{
    P1 = 0x55;
    P1 = 0x0AA
}
```

**2.6.6.3 ประโยค do-while**

การทำซ้ำประเภทนี้จะตรวจสอบเงื่อนไขภายหลังการทำสเตตเมนต์แต่ละครั้ง ถ้าหากเงื่อนไขเป็นเท็จจะออกจากการทำซ้ำทันทีโดยมีรูปแบบดังนี้

```
Void delay (unsigned int x)
{
    Unsigned char j;
    While (x--)

    {
```

```
            For (j = 0; j < 125; j ++);
        }
    }
```

จากชุดคำสั่งด้านบนจะสร้างฟังก์ชันชื่อ delay ขึ้นมาและมีการประกาศตัวแปร j สำหรับใช้ภายในฟังก์ชันและจะผ่านค่าเข้าไปในฟังก์ชันผ่านทางตัวแปร x ต่อมาเมื่อพบกับประโยค while จะลดค่า x ลงหนึ่งค่า ถ้าหากค่า x ยังไม่เป็นศูนย์จะทำสเตตเมนต์ที่ตามมาโดยวนลูป for จำนวน 125 ครั้ง และกลับไปลดค่า x ในประโยค while ใหม่จนกว่าค่า x จะเป็นศูนย์

**2.6.7 อาร์เรย์ พอยน์เตอร์ และสตรักเจอร์**

ในการเขียนโปรแกรมด้วยภาษาซีถ้าหากต้องการใช้งานตัวแปรหลายตัวเราสามารถประกาศชื่อตัวแปรออกมาหลายตัวได้ เช่น x1, x2, x3, ...,xn แต่ถ้าหากตัวแปรทุกตัวใช้เก็บข้อมูลประเภทเดียวกันเราสามารถประกาศเป็นตัวแปรแบบอาร์เรย์ (Array) ได้ การประกาศตัวแปรแบบอาร์เรย์สามารถทำได้ดังรูปแบบต่อไปนี้

ประเภทของข้อมูล ชื่ออาร์เรย์ [ขนาดอาร์เรย์] ;

ถ้าเป็นอาร์เรย์แบบสองมิติสามารถประกาศได้ดังนี้

ประเภทของข้อมูล ชื่ออาร์เรย์ [ขนาดอาร์เรย์] [ขนาดอาร์เรย์] ;

ตัวอย่างเช่น char x[8];

เป็นการประกาศตัวแปรชื่อ x จำนวน 8 เซล แต่ละเซลจะเก็บข้อมูลประเภทตัวอักขระ การอ้างถึงอาร์เรย์ x แต่ละเซลจะใช้อินเด็กซ์เป็นตัวอ้าง เช่น x[0] เป็นการอ้างถึงเซลแรก นอกจากนี้การประกาศตัวแปรอาร์เรย์สามารถทำกำหนดค่าข้อมูลเข้าไปในตัวแปรอาร์เรย์เลยได้ ตัวอย่างเช่น

```
unsigned        char ab[ ] = {0xa , 0x9 , 0x5 , 0x6};
```

จะเห็นว่าในการประกาศตัวแปรอาร์เรย์ ab จะไม่ระบุขนาดของอาร์เรย์ ระบบจะจองหน่วยความจำเท่ากับค่าที่กำหนด การประกาศแบบนี้เซลแรก ab[0] จะเก็บค่า 0A ฐานสิบหกขนาดหนึ่งไบต์ได้

ในระบบไมโครคอนโทรลเลอร์ เมื่อมีการประกาศตัวแปรอาร์เรย์ ระบบจะจองหน่วยความจำสำหรับเก็บตัวแปรอาร์เรย์นั้น ตัวแปรประเภทนี้สามารถนำไปประยุกต์ใช้งานในการเขียนโปรแกรมแบบเปิดตารางได้

ในการอ้างถึงข้อมูลแต่ละเซลในอาร์เรย์สามารถใช้ตัวแปรพอยน์เตอร์ชี้ไปที่ตำแหน่งของอาร์เรย์ได้โดยตรง โดยพอยน์เตอร์จะเป็นตัวแปรที่ใช้เก็บแอดเดรสหรือตำแหน่งหน่วยความจำ ตัวดำเนินการที่ใช้กับพอยน์เตอร์คือ & และ * ตัวอย่างเช่น ถ้ามีการประกาศตัวแปรเป็น

```
Char j;
```

จะเป็นการประกาศหน่วยความจำชื่อ j สำหรับเก็บตัวอักขระ เราสามารถอ้างแอดเดรสของตัวแปร j ได้ดังนี้

```
*j;
```

ถ้าหากมีการประกาศตัวแปรพอยน์เตอร์สำหรับเก็บแอดเดรสของตัวแปร j จะทำได้ดังนี้

```
char   *dptr;
dptr = &j;
```

เป็นการประกาศตัวแปรพอยน์เตอร์ชื่อ dptr และให้ชี้ไปที่แอดเดรสของตัวแปร j ถ้าหากมีการประกาศตัวแปรเป็น

```
int   ax[20];
```

หมายความว่าอาร์เรย์ชื่อ ax จะมีทั้งหมด 20 เซล แต่ละเซลจะเก็บเลขจำนวนเต็ม การอ้างถึงข้อมูลแต่ละเซลจะเขียนเป็น ax[i] โดยที่ I เป็นค่าอินเด็กซ์ และถ้าประกาศตัวแปร ip ให้เป็นตัวแปรพอยน์เตอร์ชี้ไปที่อาร์เรย์ของเลขจำนวนเต็มสามารถทำได้ดังต่อไปนี้

```
int   *ip
ip  =  &ax[0];
```

จะทำให้ตัวแปร ip ชี้ไปที่ตำแหน่งของเซล ax[0] ถ้าหากต้องการนำข้อมูลที่อยู่ในตัวแปร ax[0] มาใส่ในตัวแปร x สามารถทำได้ดังนี้

```
X = *ip;
```

และถ้า ip ชี้ไปที่ ax[0] ถ้าหากมีการอ้างเป็น ip + 1 จะเป็นการชี้ไปที่ ax[1] ดังนั้นถ้าหากมีการอ้างเป็น ip + i จะเป็นการอ้างไปที่อาร์เรย์ ax[i] ได้

ในระบบไมโครคอนโทรลเลอร์บางครั้งจะต้องมีการออกแบบหน่วยความจำ ROM และ RAM ต่ออยู่ภายนอกตัวไมโครคอนโทรลเลอร์ และอุปกรณ์ต่างๆ ที่ต่ออยู่กับไมโครคอนโทรลเลอร์จะมีตำแหน่งหรือแอดเดรสที่แน่นอน สามารถใช้พอยน์เตอร์ชี้ไปที่ตำแหน่งความจำภายนอกได้ ตัวอย่างเช่น

```
char  *abs_ptr  =  0x8000;
```

เป็นการประกาศตัวแปรพอยน์เตอร์ชื่อ abs_ptr ให้ชี้ไปที่ตำแหน่ง 8000H

จากที่ผ่านมาจะพบว่าตัวแปรประเภทอาร์เรย์นั้นอาจมองว่าเป็นกลุ่มของข้อมูลได้ โดยข้อมูลในกลุ่มนั้นจะเป็นข้อมูลประเภทเดียวกัน ถ้าหากต้องการประกาศตัวแปรเป็นกลุ่มของข้อมูลที่ในกลุ่มนั้นเป็นชนิดต่างกัน จะต้องประกาศตัวแปร เป็นแบบโครงสร้าง หรือสตักเจอร์ (Structure) ซึ่งมีรูปแบบดังนี้

```
struct  {
    ประเภทของข้อมูล ชื่อตัวแปร;
    .....
    }ชื่อโครงสร้าง;
```

โดยกลุ่มของข้อมูลที่ประกาศขึ้นนั้นจะอยู่ในเครื่องหมายปีกกา และเราวสามารถอ้างไปที่ข้อมูลตัวใดๆ ได้ โดยใช้เครื่องหมาย (.) ตัวอย่างเช่น

```
struct  {
            unsigned long s;
            unsigned int t;
            unsigned char done;
        }state;
```

จะเป็นการประกาศตัวแปรโครงสร้างชื่อ state ซึ่งจะใช้หน่วยความจำทั้งหมด 7 ไบต์ (long 4 + int 2 + char 1) ถ้าหากต้องการใส่ข้อมูลค่า 321 ให้กับตัวแปร t ในโครงสร้างจะทำได้ดังนี้

```
state.t = 321;
```

ถ้าหากมีการประกาศตัวแปรดังต่อไปนี้จะทำให้ผลลัพธ์มีค่าเหมือนกับตัวแปรแบบโครงสร้างในตัวอย่างที่ผ่านมา

```
#define uchar unsigned char
#define uint unsigned  ←   ประกาศตัวแปรใหม่
Struct stateform{
        Unsigned long s;
        Uint t;  ←   เรียกใช้ในโครงสร้าง
        Uchar done;
        };
```

จะเห็นว่าจะใช้ #define ประกาศตัวแปรประเภทของข้อมูลใหม่ขึ้นมา และถูกเรียกใช้ในตัวแปรโครงสร้าง ถ้าหากต้องการให้ตัวแปรโครงสร้างนี้มีชื่อว่า state จะทำได้ดังนี้

```
Struct stateform      state;
       ↑                ↑
ข้อมูลแบบโครงสร้าง    ชื่อตัวแปร
```

ถ้าหากต้องการประกาศตัวแปรแบบโครงสร้างหลายตัว สามารถทำได้ในรูปของอาร์เรย์ของโครงสร้าง (Array of Structures) ดังตัวอย่างต่อไปนี้

```
#define uchar unsigned char
#define uint unsigned
Struct stateform {
        Unsigned long s;
        Uint done;
      };
Struct stateform state[20]
```

จะเป็นการประกาศตัวแปรอาร์เรย์ชื่อ state ที่มีทั้งหมด 20 เซล แต่ละเซลจะเป็นตัวแปรแบบโครงสร้างขนาด 7 ไบต์ ทำให้ตัวแปรนี้ใช้หน่วยความจำทั้งหมด 140 ไบต์ (20×7)

### 2.6.8 การเขียนโปรแกรมจัดการหน่วยความจำ

การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ MCS-51 จะใช้การจัดหน่วยความจำแบบ Harvard โดยแยกหน่วยความจำออกเป็นส่วนของหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล สำหรับหน่วยความจำข้อมูลนั้นยังถูกแบ่งออกเป็นหน่วยความจำภายในตัว MCS-51 เอง และหน่วยความจำภายนอก ในการขยายพอร์ตเพิ่มเพื่อใช้กับตัว MCS-51 ตำแหน่งของพอร์ตจะเป็นตำแหน่งของหน่วยความจำภายนอกนี้ด้วย

สำหรับหน่วยความจำภายในของ MCS-51 ยังถูกแบ่งออกเป็น หน่วยความจำ 128 ไบต์แรก (ตำแหน่ง 00-7FH) ซึ่งสามารถเข้าถึงข้อมูลได้แบบ Direct และ Indirect ส่วนตำแหน่ง 80H ถึง FFH จะเป็นรีจิสเตอร์ฟังก์ชั่นพิเศษ หรือ SFR (Special Function Register) ซึ่งจะใช้การเข้าถึงข้อมูลแบบ Direct รีจิสเตอร์ฟังก์ชันพิเศษ ซึ่งจะใช้การเข้าถึงข้อมูลแบบ Direct

ในการใช้ภาษาซีกับ MCS-51 จะต้องมีการจัดวางหน่วยความจำให้ตัวแปรต่างๆ ที่ถูกประกาศขึ้นนั้นสามารถอยู่ในหน่วยความจำที่เหมาะสมได้ โดยการจัดหน่วยความจำของตัวคอมไพเลอร์มีรูปแบบดังนี้

TINY เป็นการจัดหน่วยความจำให้ตัวคอมไพเลอร์มองว่าไม่มีหน่วยความจำแรมภายนอก ซึ่งจะใช้กับ MCS - 51 แบบที่เป็นชิงเกิลชิป เช่น เบอร์ 89C8051

SMALL หน่วยความจำแบบนี้คอมไพเลอร์จะมองว่าแรมและรอมอยู่ซ้อนกันภายในพื้นที่ 64 กิโลไบต์ ตัวแปรต่างๆ ที่ถูกประกาศใช้งาน หรือประกาศเป็นแบบโลเคิลจะถูกกำหนดอยู่ในหน่วยความจำข้อมูลภายใน (Internal Data Memory) ขนาดของหน่วยความจำสแตกจะถูกกำหนดตามที่ใช้งานจริง รูปแบบหน่วยความจำโหมดนี้จะเป็นค่าที่โปรแกรมคอมไพเลอร์ได้กำหนดไว้แล้ว

COMPACT หน่วยความจำแบบนี้จะมีลักษณะเดียวกับหน่วยความจำแบบ SMALL แต่ตัวแปรที่ถูกประกาศใช้งาน หรือประกาศแบบโลเคิลจะถูกกำหนดอยู่ในหน่วยความจำภายนอก ทำให้ตัวแปรสามารถมีได้มากกว่าแบบ SMALL แต่การทำงานของโปรแกรมที่ต้องประมวลผลกับตัวแปรจะทำงานได้ช้ากว่าแบบ SMALL เพราะแบบ SMALL ตัวแปรจะอยู่ในแรมภายในชิป

LARGE หน่วยความจำแบบนี้จะมีขนาดใหญ่ ตัวแปรทั้งหมดจะถูกกำหนดอยู่ในหน่วยความความจำข้อมูลภายนอก

การเขียนโปรแกรมด้วยภาษาซีให้กับไมโครคอนโทรลเลอร์ด้วย C51นี้สามารถประกาศตัวแปรต่างๆ แบบเจาะจงได้ โดยใช้แมโครที่ถูกกำหนดไว้ในไฟล์ absacc.h ดังต่อไปนี้

CODE แทนพื้นที่หน่วยความจำโปรแกรมภายนอก

| | |
|---|---|
| DATA | แทนพื้นที่หน่วยความจำข้อมูลภายใน 128 ไบต์แรก (ตำแหน่ง 00H ถึง 7FH) |
| IDATA | แทนเนื้อหาที่หน่วยความจำข้อมูลภายใน 256 ไบต์ |
| BDATA | แทนตำแหน่งของหน่วยความจำระดับบิต 128 บิต (20H ถึง 2FH)-ของหน่วยความจำภายใน |
| XDATA | แทนตำแหน่งหน่วยความจำภายนอก |
| PDATA | แทนตำแหน่งหน่วยความจำภายนอก 256 ไบต์แรก |

ในการกำหนดชื่อตัวแปรต่างๆ เพื่อแทนตำแหน่งหน่วยความจำนั้นมีรูปแบบดังนี้

ชนิดของตัวแปร ชื่อแมโคร ตัวแปร

ตัวอย่างเช่น

```
Char CODE text [ ] = "COMPUTER"
Unsigned char XDATA AB[100]
Unsigned int IDATA x, y, z
```

การกำหนดบรรทัดแรกจะทำให้คำว่า COMPUTER ถูกเก็บอยู่ในหน่วยความจำโปรแกรมภายนอก บรรทัดที่สองจะทำให้อาร์เรย์ AB ถูกประกาศเอาไว้ในหน่วยความจำข้อมูลภายนอกบรรทัดที่สามจะทำให้ตัวแปร x, y และ z ถูกประกาศเอาไว้ในหน่วยความจำข้อมูลภายใน

ในการเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์ติดต่อกับพอร์ตจะต้องทำการโอนย้ายข้อมูลกับพอร์ตตำแหน่งแอดเดรสนั้นๆ ซึ่งการอ้างแอดเดรสของพอร์ตสามารถใช้แมโครได้เช่นกัน ทำให้สามารถแทนพอร์ตด้วยตัวแปรต่างๆ ได้โดยแมโครที่ใช้จะเก็บอยู่ในไฟล์ absacc.h โดยมีชื่อดังต่อไปนี้

| | |
|---|---|
| CBYTE , CWORD | แทนไบต์หรือเวิร์ดของหน่วยความจำโปรแกรม |
| DBYTE , DWORD | แทนไบต์หรือเวิร์ดของหน่วยความจำข้อมูลภายใน |
| PBYTE , PWORD | แทนไบต์หรือเวิร์ดของหน่วยความจำภายนอก 256 ตำแหน่งแรก |
| XBYTE , XWORD | แทนไบต์หรือเวิร์ดของหน่วยความจำภายนอกทั้งหมด |

ในการกำหนดตัวแปรให้แทนตำแหน่งต่างๆ จะต้องใช้ร่วมกับไดเร็กทีฟ #define ดังตัวอย่างต่อไปนี้

```
#define port8 XBYTE [0x8000
```

จะเป็นการแทนตัวแปร port8 ด้วยหน่วยความจำภายนอกตำแหน่ง 8000H ถ้าหากต้องการส่งค่า FF ไปยังหน่วยความจำตำแหน่ง 8000H สามารถทำได้ดังนี้

```
port8 = 0x0FF;
```

ถ้าหากต้องการอ่านข้อมูลจากตำแหน่ง 8000H มาเก็บในตัวแปร x สามารถทำได้ดังนี้

```
Char x;
x = port8;
```

การเข้าถึงข้อมูลระดับบิตและรีจิสเตอร์พิเศษ ในโปรแกรม C51 ถ้ามีการประกาศตัวแปรและกำหนดให้เป็นหน่วยความจำแบบ BDATA จะเป็นการใช้งานในแอดเดรสหน่วยความจำที่สามารถเข้าถึงข้อมูลในระดับบิตได้ นอกจากนี้สามารถใช้คำสั่ง C51 กำหนดตัวแปรแบบบิตขึ้นมาได้อีกด้วย ตัวอย่างเช่น

```
Int BDATA ibase;        /* Bit - addressable int */
Char BDATA bary[4];     /* Bit - addressable array */
Sbit mybit0 = ibase^0;  /* Bit 0 of ibase */
Sbit mybit15 = ibase^15;        /* Bit 15 of ibase */
Sbit Ary07 = bary[0]^7  /* Bit 7 of bary[0] */
```

บรรทัดแรกจะเป็นการประกาศตัวแปรชื่อ ibase สำหรับเก็บจำนวนเต็มลงในหน่วยความจำพื้นที่ที่สามารถเข้าถึงข้อมูลในระดับบิตได้ บรรทัดต่อมาประกาศตัวแปรอาร์เรย์จำนวน 4 ไบต์ลงในหน่วยความจำระดับบิต ในบรรทัดที่สามจะใช้คำสั่ง sbit กำหนดตัวแปรขึ้นมาใหม่ แทนบิตต่ำสุด โดยใช้สัญลักษณ์ carat ('^') ซึ่งจำทำให้ตัวแปรชื่อ mybit0 แทนบิตที่ 0 ของหน่วยความจำชื่อ ibase บรรทัดที่สี่จะเป็นตัวแปรชื่อ mybit15 แทนบิตสูงสุดของตัวแปร ibase และบรรทัดสุดท้ายจะให้ตัวแปรชื่อ Ary07 แทนบิตที่ 7 ของ bary[0] สำหรับขนาดของบิตของตัวแปร จะขึ้นกับประเภทของข้อมูลที่กำหนด เช่น ถ้าเป็นข้อมูลแบบ char และ unsigned char ข้อมูลบิตจะอยู่ในช่วง 0 ถึง 7 บิต ถ้าเป็นแบบ int, unsigned int, short และ unsigned short จะอยู่ในช่วง 0 ถึง 15 บิต

ถ้าต้องการกำหนดตัวแปรชื่อ OUT แทนบิตสูงสุดของพอร์ต P1 ของ MCS-51 สามารถทำได้ดังนี้

```
Sbit OUT = P1^7;
```

พื้นที่หน่วยความจำภายในของ MCS-51 ส่วนหนึ่งจะเป็นพื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษ (SFR) เช่น รีจิสเตอร์ควบคุมไทเมอร์ รีจิสเตอร์ตัวนับ พอร์ตอนุกรม พอร์ตอินพุตเอาต์พุตต่างๆ ซึ่งอยู่ในหน่วยความจำภายในตั้งแต่ตำแหน่ง 0x80 ถึง 0xFF ถ้าหากต้องการกำหนดชื่อใหม่แทนสัญลักษณ์เหล่านั้น สามารถใช้คำสั่ง sfr ในการประกาศได้ โดยใช้เครื่องหมายเท่ากับ (=) ระบุค่าแอดเดรสให้กับตัวแปรนั้น เช่น

```
Sfr P0 = 0x80;          /* Port_0 address 80H */
Sfr P1 = 0x90;          /* Port_1 address 90H */
Sfr P2 = 0xA0;          /* Port_2 address A0H */
Sfr P3 = 0xB0;          /* Port_3 address B0H */
```

แต่ถ้าหากต้องการกำหนดชื่อแทนรีจิสเตอร์ฟังก์ชันพิเศษที่มีขนาด 16 บิต จะต้องใช้คำสั่ง sfr 16 และระบุแอดเดรสไบต์ต่ำให้กับตัวแปรนั้น ตัวอย่างเช่น

```
sfr16   T2 = 0xcc       /* Timer 2 : T2L 0CCH , T2H 0CDH */
```

เป็นการให้ T2 แทนไทเมอร์ 2 ซึ่งรีจิสเตอร์นี้จะมีขนาด 2 ไบต์ ไบต์ต่ำหรือ T2L จะอยู่ในแอดเดรส 0CCH และไบต์สูง T2H จะอยู่ในแอดเดรส 0CDH

สำหรับในการเขียนโปรแกรมด้วย C51 นั้นจะเห็นว่ารีจิสเตอร์บางตัว ตำแหน่งบิตบางตำแหน่งเราสามารถอ้างชื่อขึ้นมาได้เลย เนื่องจากในไฟล์ res51.h ที่ได้ include ขึ้นมาเป็นไฟล์ที่กำหนดรีจิสเตอร์ต่างเอาไว้แล้ว เช่น P0, P1, P2 และ P3 สำหรับตำแหน่งบิตของรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ เช่น TF1, TR1, TF0, TR0, IE1, IT!, IE0 และ IT0 ซึ่งเป็นบิตที่อยู่ในรีจิสเตอร์ TCON ก็กำหนดไว้แล้วเช่นกัน โดยในไฟล์ reg51.h ส่วนหนึ่งได้กำหนดไว้ดังนี้

```
/* BYTE register */
Sfr P0   = 0x80;
Sfr P1   = 0x90;
Sfr P2   = 0xA0;
Sfr P3   = 0xB0;
```

# บทที่ 3
# การออกแบบ การสร้าง และการทำงาน

## 3.1 กล่าวนำ

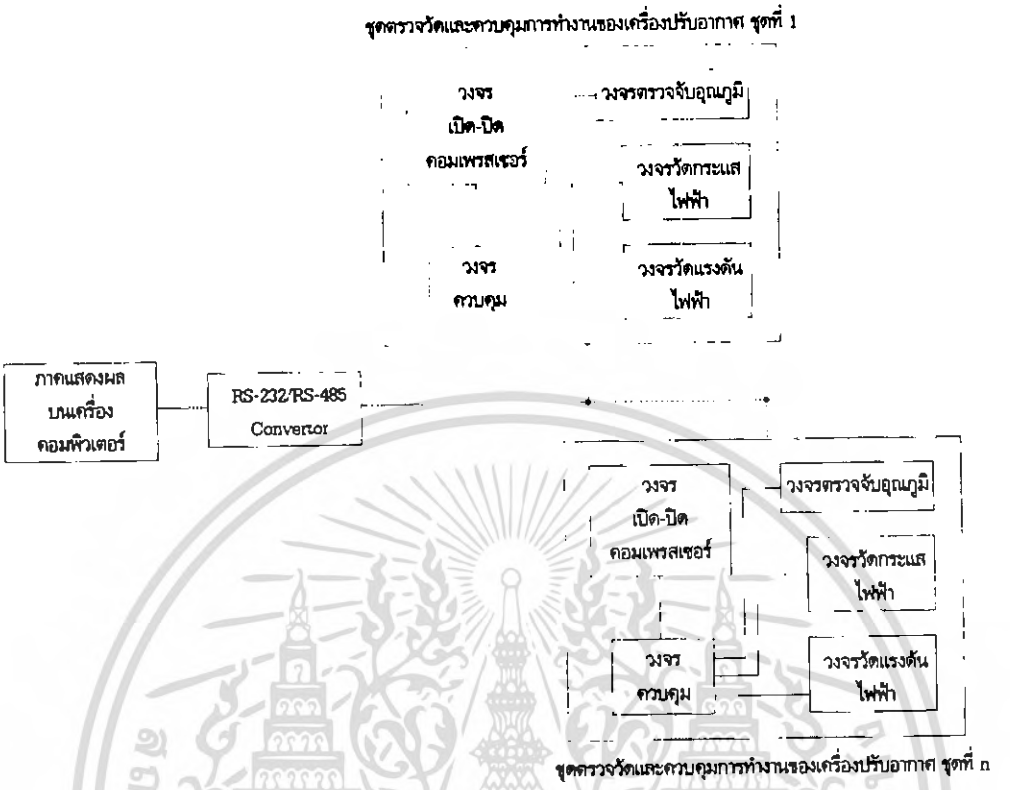
การออกแบบและการสร้างระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศประกอบไปด้วย 3 ส่วนคือ ชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ ชุดแปลงสัญญาณ RS232/RS485 และภาคแสดงผลบนเครื่องคอมพิวเตอร์ วงจรของชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศประกอบด้วย วงจรควบคุม วงจรตรวจจับอุณหภูมิ วงจรวัดแรงดันไฟฟ้า วงจรวัดกระแสไฟฟ้า วงจรเปิด-ปิดคอมเพรสเซอร์ และวงจรแหล่งจ่ายไฟ ส่วนของภาคแสดงผลบนเครื่องคอมพิวเตอร์ใช้โปรแกรม Visual Basic ในการแสดงผล

## 3.2 การออกแบบวงจร

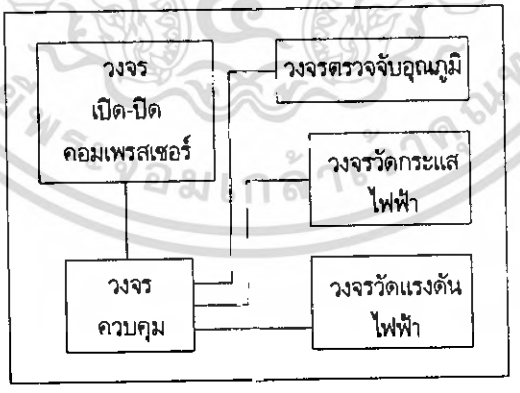
วิธีการออกแบบวงจรที่ใช้งานในระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศออกแบบโดยการกำหนดขั้นตอนการทำงานในแผนผังการทำงาน ดังรูปที่ 3.1

การออกแบบวงจรแบ่งออกเป็นส่วนใหญ่ๆ ได้ดังรูปที่ 3.1 (ก) ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศนี้ใช้มาตรฐานการเชื่อมต่อแบบอนุกรม RS-485เชื่อมต่อกับชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศและทำการแปลงสัญญาณมาเป็น RS-232 เพื่อติดต่อสื่อสารกับข้อมูลกับคอมพิวเตอร์และแสดงผลบนเครื่องคอมพิวเตอร์

ในรูปที่ 3.1 (ข) เป็นชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ ภายในชุดนี้จะประกอบด้วยวงจรต่างๆ ดังนี้ วงจรควบคุม วงจรเปิด-ปิดคอมเพรสเซอร์ วงจรตรวจจับอุณหภูมิ วงจรวัดแรงดันไฟฟ้า และวงจรวัดกระแสไฟฟ้า โดยมีวงจรนี้จะทำหน้าที่ตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ แล้วนำค่าที่ได้นั้นส่งไปยังชุดแปลงสัญญาณRS-485 เป็น RS-232 เพื่อทำการติดต่อและแสดงผลบนหน้าจอคอมพิวเตอร์ต่อไป




ก แผนผังการทำงานของระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ



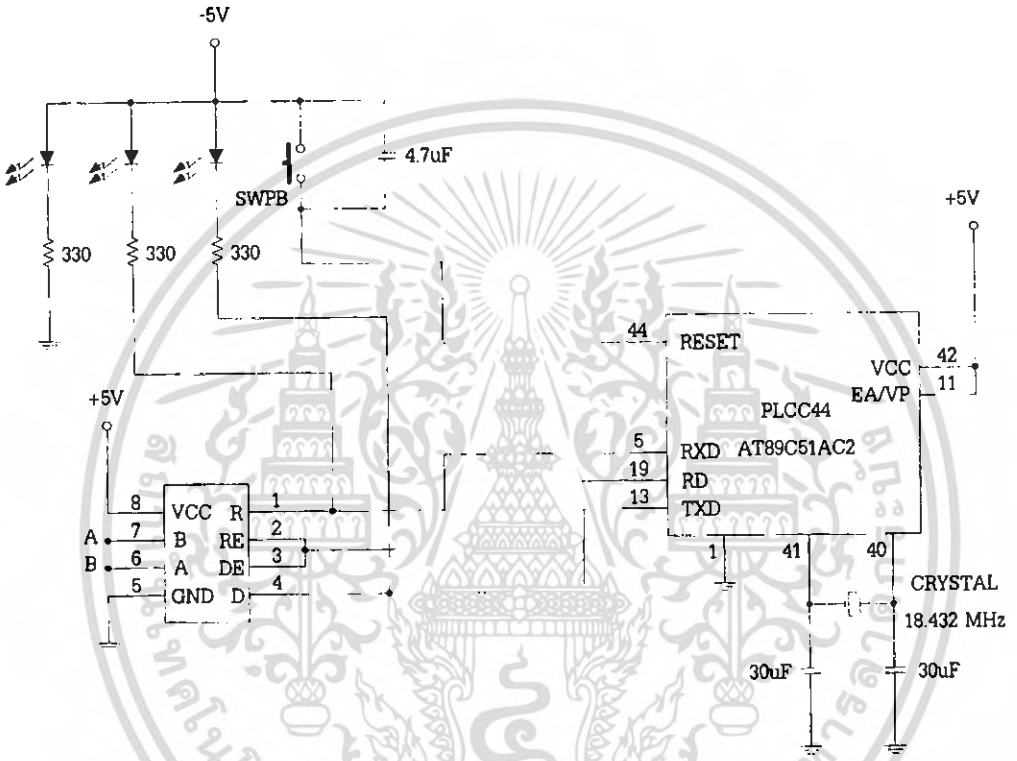

ข ชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ

**รูปที่ 3.1** โครงสร้างชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ

## 3.3 ชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ

### 3.3.1 วงจรควบคุม

วงจรควบคุมใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 เป็นตัวควบคุมหลักในการประมวลผลค่าข้อมูล โดยมีวงจรรวม (IC) ตรวจวัดอุณหภูมิเบอร์ DS1820 ทำการตรวจวัดอุณหภูมิ แล้วส่งค่าที่ตรวจวัดได้มายังไมโครคอนโทรลเลอร์เพื่อทำการประมวลผล

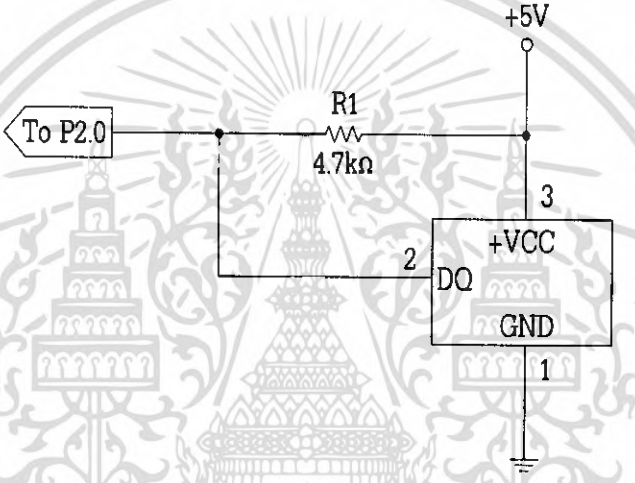



**รูปที่ 3.2** วงจรควบคุม

การทำงานของวงจร เมื่อ DS1820 ส่งข้อมูลมายังขาที่ 21 ของไมโครคอนโทรลเลอร์ ไมโครคอนโทรลเลอร์ก็จะทำการประมวลผล และส่งข้อมูลที่ประมวลผลได้มาแสดงผลที่หน้าจอคอมพิวเตอร์ ในขณะเดียวกันก็ทำหน้าที่รับส่งข้อมูลซึ่งเป็นค่าข้อมูลต่างๆ ส่งผ่านพอร์ทอนุกรม RS-485 โดยไอซีที่ใช้ในการรับส่งผ่านพอร์ทอนุกรม RS-485 คือไอซีเบอร์ SN75176 เชื่อมต่อกับขา Tx, Rx, และRD ของไมโครคอนโทรลเลอร์ ข้อมูลที่ส่งมาจะถูกแปลงเพื่อส่งผ่านมาตรฐานอนุกรมแบบ RS-232 ด้วยวงจรแปลง RS-232/RS-485

### 3.3.2 วงจรตรวจจับอุณหภูมิ

วงจรตรวจจับอุณหภูมิ DS1820 ทำหน้าที่เป็นอุปกรณ์ตรวจวัดอุณหภูมิ การติดต่อระหว่างวงจรตรวจวัดอุณหภูมิกับวงจรประมวลผลค่าอุณหภูมิ และรับ-ส่งข้อมูลผ่าน RS-485 การเชื่อมต่อวงจรรวม DS1820 กับไมโครคอนโทรลเลอร์เป็นการเชื่อมต่อข้อมูลแบบสายสัญญาณเพียงเส้นเดียววงจรรวมหมายเลข DS1820 มีขาต่อใช้งานเพียง 3 ขา คือ ขา DQ ซึ่งเป็นขาที่เชื่อมต่อเพื่อส่งข้อมูลในระบบบัส ขาไฟเลี้ยงและขากราวด์ เนื่องจากสายสัญญาณของระบบบัสนี้ต้องทำการกำหนดสภาวะปกติไว้ที่ลอจิกสูง ดังนั้นในการเชื่อมต่อใช้งานจึงต้องทำการต่อตัวความต้านทานค่า 4.7 กิโลโอห์ม พูลอัพกับไฟเลี้ยงในวงจรด้วย รูปวงจรตรวจจับอุณหภูมิ แสดงดังรูปที่ 3.3

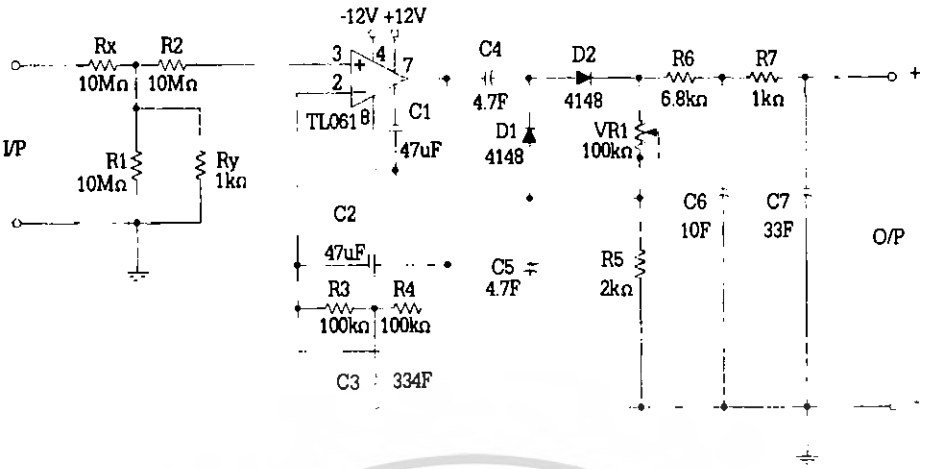



**รูปที่ 3.3** วงจรตรวจจับอุณหภูมิ

### 3.3.3 วงจรวัดแรงดันไฟฟ้า

วงจรวัดแรงดันทำหน้าที่วัดแรงดันไฟฟ้าจากเครื่องปรับอากาศ เพื่อที่จะรับค่าที่ได้นั้นส่งต่อไปยังสายนำสัญญาณเพื่อที่จะแสดงผลออกหน้าจอต่อไป

จากรูปที่ 3.4 วงจรวัดแรงดันไฟฟ้ามีลักษณะการทำงานของวงจร คือ สัญญาณจะถูกป้อนให้กลับวงจรแบ่งแรงดันไฟฟ้า แล้วผ่าน R2 ไปยังขา 3 ของไอซีเบอร์ TL061 โดยมี R1 ทำหน้าที่เป็นอินพุทอิมพีแดนซ์ของวงจร ไอซีเบอร์ TL061 จะทำหน้าที่ขยายสัญญาณอินพุทแล้วป้อนให้กับวงจรเร็กติไฟเออร์แบบโวลท์เตจดับเบิ้ล ได้เป็นแรงดันไฟฟ้ากระแสตรงผ่าน R7 ไปเข้าวงจรต่อไป เอาท์พุทที่ได้จะไม่เกิน 2 โวลต์ โดยจะถูกควบคุมจากการปรับค่าของ VR1 ให้เหมาะสม

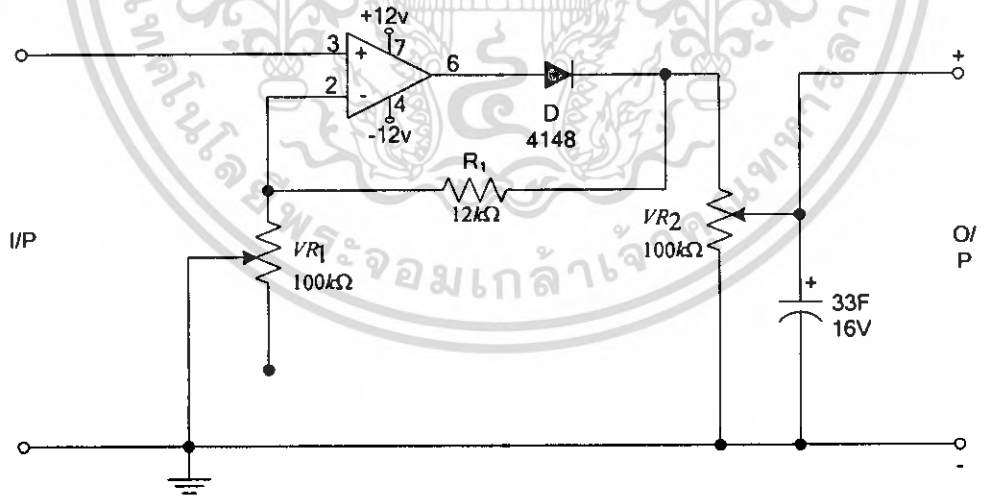



**รูปที่ 3.4** วงจรวัดแรงดันไฟฟ้า

### 3.3.4 วงจรวัดกระแสไฟฟ้า

วงจรวัดกระแสไฟฟ้าทำหน้าที่วัดกระแสไฟฟ้าจากเครื่องปรับอากาศ เพื่อที่จะรับค่าที่ได้นั้นส่งต่อไปยังสายนำสัญญาณเพื่อที่จะแสดงผลออกหน้าจอต่อไป

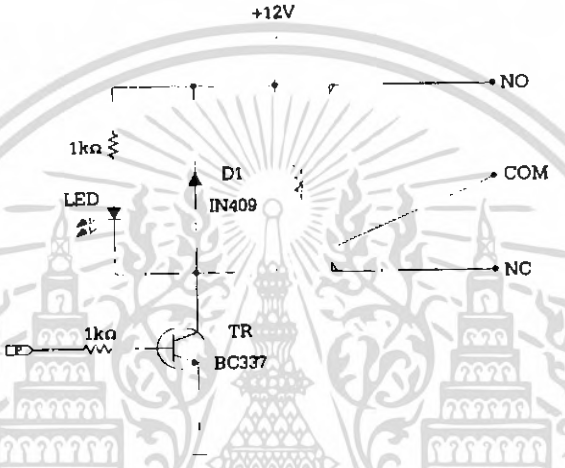
ในรูปที่ 3.5 วงจรวัดกระแสไฟฟ้ามีลักษณะการทำงานของวงจร คือ สัญญาณอินพุตที่ได้จาก CT (Current Transformer) เข้าที่ขา 3 ของไอซีออปแอมป์เบอร์ OP07 จะทำหน้าที่ขยายสัญญาณอินพุทแล้วป้อนให้กับวงจรเร็กติไฟเออร์แบบโวลท์เตจซิงเกิ้ล ให้เป็นแรงดันไฟฟ้ากระแสตรง โดยมี C ทำหน้าที่เป็นตัวฟิลเตอร์ และ VR2 ทำหน้าที่ปรับแรงดันไฟฟ้ากระแสตรงในด้านเอาต์พุตไม่ให้เกิน 3 โวลต์




**รูปที่ 3.5** วงจรวัดกระแสไฟฟ้า

### 3.3.5 วงจรเปิด-ปิดคอมเพรสเซอร์

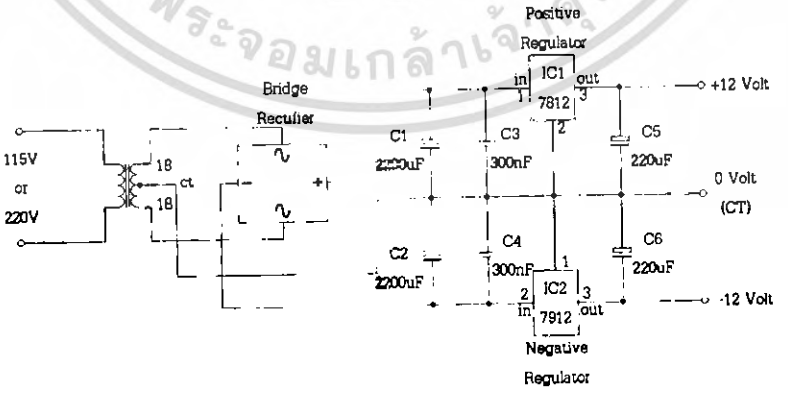
วงจรเปิด-ปิดคอมเพรสเซอร์จะมีทรานซิสเตอร์เบอร์ BC 337 ทำหน้าที่เป็นตัวสวิตช์เปิด-ปิด โดยจะมีสัญญาณจากพอร์ทของ MCS-51 ไบอัสให้กับขา B ของทรานซิสเตอร์ทำให้กระแสไหลผ่านตัวทรานซิสเตอร์ โดยเมื่อจ่ายแรงดันชั้วบวกให้กับรีเลย์ที่ขา 1 เมื่อ MCS-51 สั่งให้รีเลย์ทำงาน ทรานซิสเตอร์จะถูกวงจรไบอัสให้กระแสสามารถไหลผ่านขา E ลงกราวด์ได้ และสามารถทำให้รีเลย์ทำงานได้ โดยแสดงรูปวงจรเปิด-ปิดคอมเพรสเซอร์ไว้ในรูปที่ 3.6




**รูปที่ 3.6** วงจรเปิด-ปิดคอมเพรสเซอร์

### 3.3.6 วงจรแหล่งจ่ายไฟ

วงจรแหล่งจ่ายไฟทำหน้าที่จ่ายแรงดันไฟฟ้าให้กับชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ

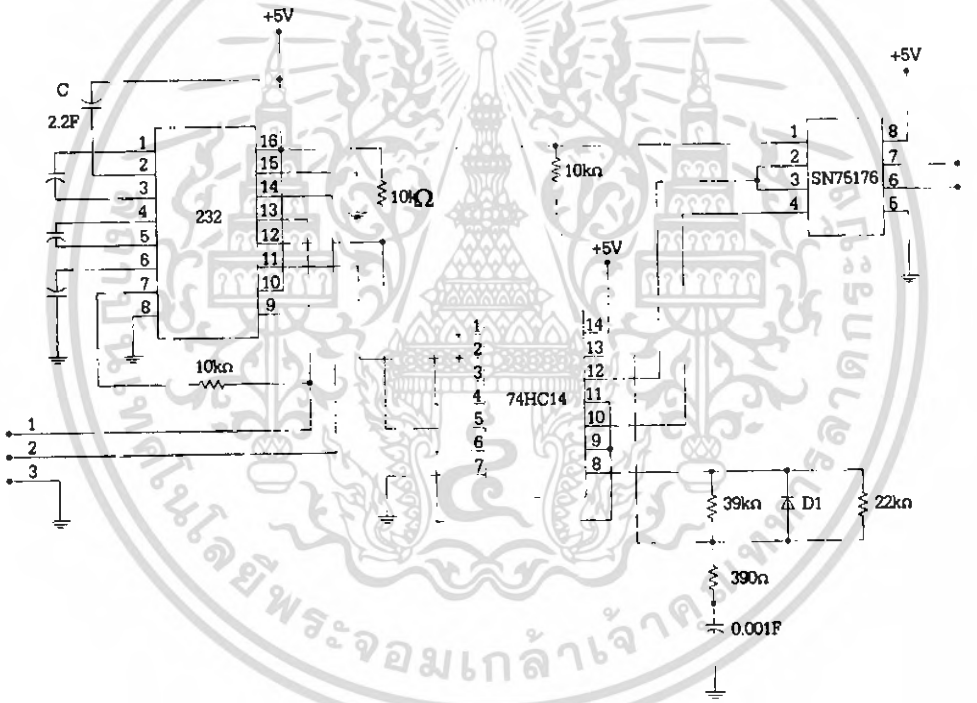



**รูปที่ 3.7** วงจรแหล่งจ่ายไฟ

จากรูปที่ 3.7 วงจรแหล่งจ่ายไฟมีลักษณะการทำงานของวงจร คือ ทำหน้าที่แปลงสัญญาณไฟฟ้ากระแสสลับเป็นแรงดันไฟฟ้ากระแสตรง โดยมีวงจรไดโอดบริดจ์เป็นวงจรเรียงกระแส แล้วมีตัวเก็บประจุค่า 2200 ไมโครฟารัดเป็นตัวฟิลเตอร์แล้วใช้ไอซีเบอร์ 7812 เป็นตัวพิกัดแรงดันขนาด 12โวลต์ ไอซีเบอร์ 7805 เป็นตัวพิกัดแรงดัน 5 โวลต์ ไอซีเบอร์ 7912 เป็นตัวพิกัดแรงดันขนาด -12 โวลต์ และไอซีเบอร์ 7905 เป็นตัวพิกัดขนาด -5 โวลต์

## 3.4 ชุดแปลงสัญญาณ RS232/RS485

ไอซีเบอร์ 232 ทำหน้าที่ในการส่งผ่านสัญญาณจากพอร์ทอนุกรมของคอมพิวเตอร์ ซึ่งอยู่ในรูปของการสื่อสารแบบมาตรฐานของ RS 232 และไอซีเบอร์ SN 75176 ทำหน้าที่ในการส่งผ่านสัญญาณในรูปของการสื่อสารแบบมาตรฐานของ RS 485 โดยมีไอซีเบอร์ 74HC14 ทำหน้าที่เป็นตัวควบคุมการแปลงสัญญาณของ RS 232 กับ RS 485 เพื่อให้สามารถทำการติดต่อสื่อสารกันได้




**รูปที่ 3.8** วงจรแปลงสัญญาณ RS232/RS485

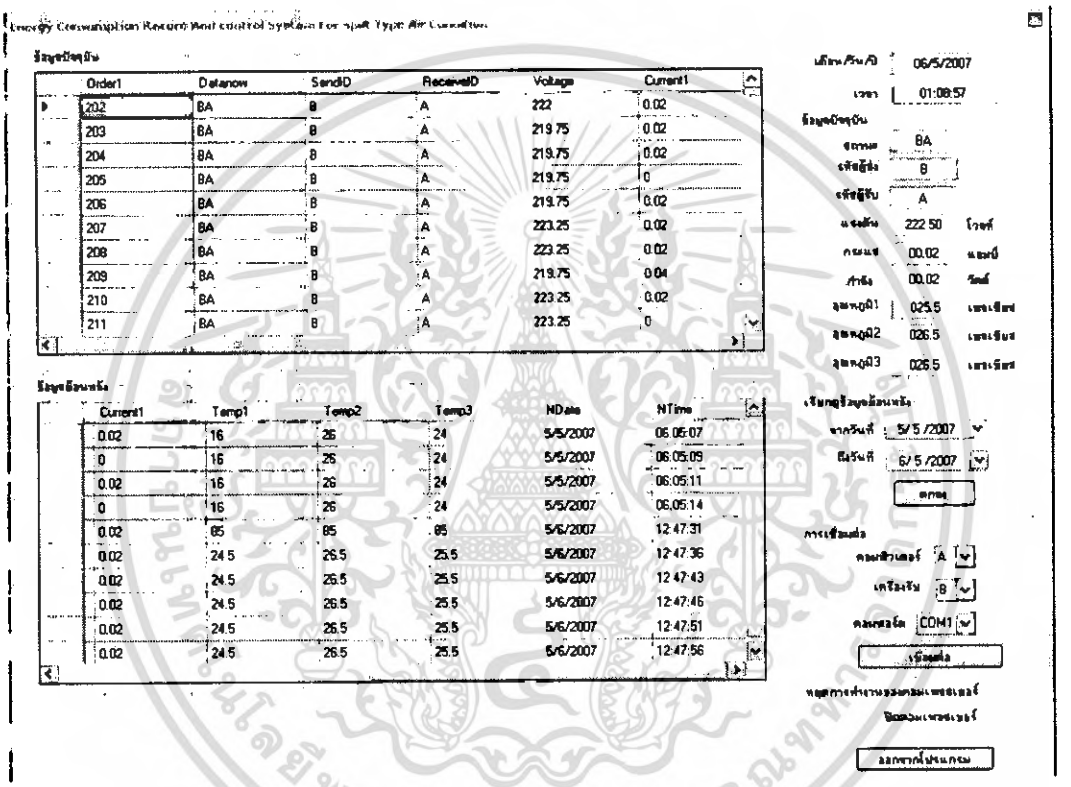
## 3.5 ภาคแสดงผลบนเครื่องคอมพิวเตอร์

ในการสร้างรูปแบบหน้าจอแสดงผลได้เลือกใช้โปรแกรม Visual Basic ในการสร้าง การออกแบบหน้าจอแสดงผลที่เครื่องคอมพิวเตอร์ของระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศที่สร้างขึ้นจะเริ่มต้นโดยการออกแบบหน้าจอหลักของโปรแกรม ที่หน้าจอหลักของโปรแกรมจะมีเมนูคำสั่งที่จะ

เลือกใช้คำสั่งย่อยต่างๆ รูปแบบของหน้าต่างของคำสั่งย่อยนั้นการการออกแบบก็จะแตกต่างกันตามลักษณะการใช้งานของโปรแกรมย่อยนั้นๆ รายละเอียดของการออกแบบของทุกหน้าต่างจะกล่าวถึงดังต่อไปนี้

### 3.5.1 หน้าจอแสดงข้อมูลการใช้พลังงาน

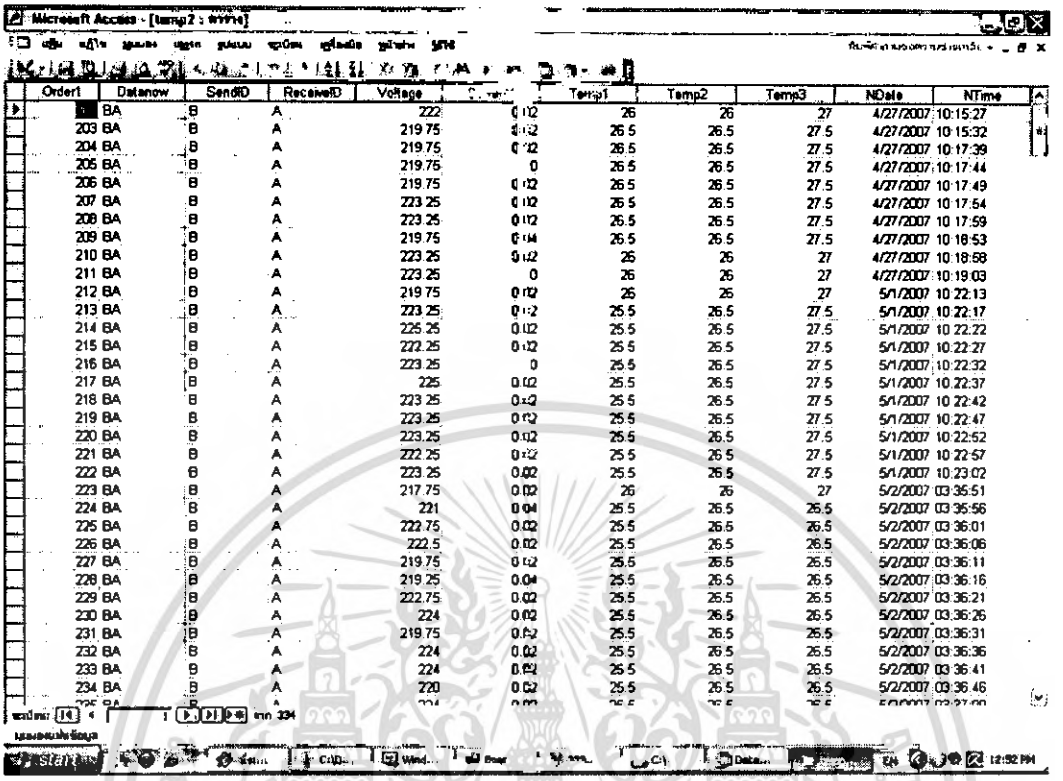
หน้าจอแสดงข้อมูลการใช้พลังงานจะทำหน้าที่ในการแสดงผลค่าของแรงดันไฟฟ้า กระแสไฟฟ้า และค่าอุณหภูมิ ซึ่งในการใช้งานสามารถแสดงค่าต่างๆ เหล่านี้เป็นตัวเลขและกราฟได้ โดยสามารถเรียกดูข้อมูลที่เป็นปัจจุบันและย้อนหลังได้อีกด้วย หน้าจอแสดงผลข้อมูลการใช้พลังงานแสดงดังรูปที่ 3.9




**รูปที่ 3.9** หน้าจอแสดงผลข้อมูลการใช้พลังงาน

### 3.5.2 หน้าจอแสดงข้อมูลทั้งหมด

ในการทำงานของโปรแกรมต่างๆ ข้อมูลจะถูกเก็บไว้ในฐานข้อมูล และสามารถที่จะเรียกดูข้อมูลต่างๆ เหล่านี้ได้จากหน้าจอแสดงผลโดยจะแสดงในรูปที่ 3.10



| Order1 | Datanow | SendID | ReceiveID | Voltage | Temp1 | Temp2 | Temp3 | NDate | NTime |
|---|---|---|---|---|---|---|---|---|---|
| 1 | BA | B | A | 222 | 26 | 26 | 27 | 4/27/2007 | 10:15:27 |
| 203 | BA | B | A | 219.75 | 26.5 | 26.5 | 27.5 | 4/27/2007 | 10:15:32 |
| 204 | BA | B | A | 219.75 | 26.5 | 26.5 | 27.5 | 4/27/2007 | 10:17:39 |
| 205 | BA | B | A | 219.75 | 26.5 | 26.5 | 27.5 | 4/27/2007 | 10:17:44 |
| 206 | BA | B | A | 219.75 | 26.5 | 26.5 | 27.5 | 4/27/2007 | 10:17:49 |
| 207 | BA | B | A | 223.25 | 26.5 | 26.5 | 27.5 | 4/27/2007 | 10:17:54 |
| 208 | BA | B | A | 223.25 | 26.5 | 26.5 | 27.5 | 4/27/2007 | 10:17:59 |
| 209 | BA | B | A | 219.75 | 26.5 | 26.5 | 27.5 | 4/27/2007 | 10:18:53 |
| 210 | BA | B | A | 223.25 | 26 | 26 | 27 | 4/27/2007 | 10:18:58 |
| 211 | BA | B | A | 223.25 | 26 | 26 | 27 | 4/27/2007 | 10:19:03 |
| 212 | BA | B | A | 219.75 | 26 | 26 | 27 | 5/1/2007 | 10:22:13 |
| 213 | BA | B | A | 223.25 | 25.5 | 26.5 | 27.5 | 5/1/2007 | 10:22:17 |
| 214 | BA | B | A | 225.25 | 25.5 | 26.5 | 27.5 | 5/1/2007 | 10:22:22 |
| 215 | BA | B | A | 222.25 | 25.5 | 26.5 | 27.5 | 5/1/2007 | 10:22:27 |
| 216 | BA | B | A | 223.25 | 25.5 | 26.5 | 27.5 | 5/1/2007 | 10:22:32 |
| 217 | BA | B | A | 225 | 25.5 | 26.5 | 27.5 | 5/1/2007 | 10:22:37 |
| 218 | BA | B | A | 223.25 | 25.5 | 26.5 | 27.5 | 5/1/2007 | 10:22:42 |
| 219 | BA | B | A | 223.25 | 25.5 | 26.5 | 27.5 | 5/1/2007 | 10:22:47 |
| 220 | BA | B | A | 223.25 | 25.5 | 26.5 | 27.5 | 5/1/2007 | 10:22:52 |
| 221 | BA | B | A | 222.25 | 25.5 | 26.5 | 27.5 | 5/1/2007 | 10:22:57 |
| 222 | BA | B | A | 223.25 | 25.5 | 26.5 | 27.5 | 5/1/2007 | 10:23:02 |
| 223 | BA | B | A | 217.75 | 26 | 26 | 27 | 5/2/2007 | 03:35:51 |
| 224 | BA | B | A | 221 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:35:56 |
| 225 | BA | B | A | 222.75 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:36:01 |
| 226 | BA | B | A | 222.5 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:36:06 |
| 227 | BA | B | A | 219.75 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:36:11 |
| 228 | BA | B | A | 219.25 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:36:16 |
| 229 | BA | B | A | 222.75 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:36:21 |
| 230 | BA | B | A | 224 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:36:26 |
| 231 | BA | B | A | 219.75 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:36:31 |
| 232 | BA | B | A | 224 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:36:36 |
| 233 | BA | B | A | 224 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:36:41 |
| 234 | BA | B | A | 220 | 25.5 | 26.5 | 26.5 | 5/2/2007 | 03:36:46 |

**รูปที่ 3.10** หน้าจอแสดงข้อมูลทั้งหมด

ค่าที่เก็บไว้ในฐานข้อมูลจะประกอบด้วย ผู้ส่ง ผู้รับ ค่าแรงดันไฟฟ้า ค่ากระแสไฟฟ้า ค่าอุณหภูมิจุดที่ 1 ค่าอุณหภูมิจุดที่ 2 และค่าอุณหภูมิจุดที่ 3 โดยสามารถที่จะเรียกดูข้อมูลที่เป็นปัจจุบันและย้อนหลังได้

# บทที่ 4

# การทดลองและผลการทดลอง

## 4.1 กล่าวนำ

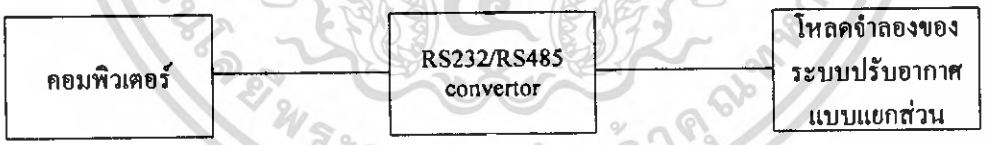
ระบบบันทึกและควบคุมการทำงานของระบบปรับอากาศที่สร้างขึ้นประกอบด้วย 3 ส่วน คือ ชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ ชุดRS232/RS485 Converter และภาคแสดงผลบนเครื่องคอมพิวเตอร์ โดยชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศจะประกอบด้วย วงจรควบคุม วงจรตรวจจับอุณหภูมิ วงจรวัดแรงดันไฟฟ้า วงจรวัดกระแสไฟฟ้า วงจรเปิด-ปิดคอมเพรสเซอร์ และวงจรแหล่งจ่ายไฟ โดยมีชุด RS232/RS485 Converter เป็นตัวเชื่อมต่อ ส่วนของภาคแสดงผลบนเครื่องคอมพิวเตอร์ใช้โปรแกรม Visual Basic ในการสร้าง การทดลองของระบบนี้เพื่อหาความสามารถของระบบว่าเป็นไปตามขีดความสามารถหรือไม่ ระบบมีประสิทธิภาพเพียงใด มีส่วนการทำงานใดบกพร่องบ้าง เพื่อที่จะได้แก้ไขและปรับปรุงให้ดีขึ้นต่อไป ซึ่งการทดลองและผลการทดลองเป็นดังต่อไปนี้

## 4.2 การทดลองการแสดงผลเมื่อเชื่อมต่อกับโหลดจำลองของระบบปรับอากาศ

### 4.2.1 การทดลอง

ลำดับขั้นการทดลอง

1. เชื่อมต่อระบบดังรูปที่ 4.1




**รูปที่ 4.1** การเชื่อมต่อกับโหลดจำลองของระบบปรับอากาศ

2. เปิดสวิตช์การทำงานของโหลดจำลองของระบบปรับอากาศ

3. เข้าสู่โปรแกรมของระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศโดยจะแสดงผลบนหน้าจอคอมพิวเตอร์

4. ทำการวัดค่ากำลังไฟฟ้าที่โหลดจำลองของระบบปรับอากาศ

### 4.2.2 ผลการทดลอง

ตารางที่ 4.1 ผลการทดลองการเชื่อมต่อกับโหลดทดลองจำลองของระบบปรับอากาศ

| จำนวนหลอดไฟ | ค่ากำลังไฟฟ้าที่รับได้ (วัตต์) | ผลการทดลอง (แอมแปร์) |
|---|---|---|
| 1 | 100 | 0.46 |
| 2 | 200 | 0.91 |
| 3 | 300 | 1.36 |
| 4 | 400 | 1.91 |
| 5 | 500 | 2.25 |
| 6 | 600 | 2.72 |
| 7 | 700 | 3.15 |
| 8 | 800 | 3.61 |
| 9 | 900 | 4.15 |
| 10 | 1000 | 4.52 |
| 11 | 1100 | 4.95 |
| 12 | 1200 | 5.41 |
| 13 | 1300 | 5.86 |
| 14 | 1400 | 6.31 |
| 15 | 1500 | 6.75 |
| 16 | 1600 | 7.21 |
| 17 | 1700 | 7.65 |
| 18 | 1800 | 8.11 |
| 19 | 1900 | 8.56 |
| 20 | 2000 | 9.12 |

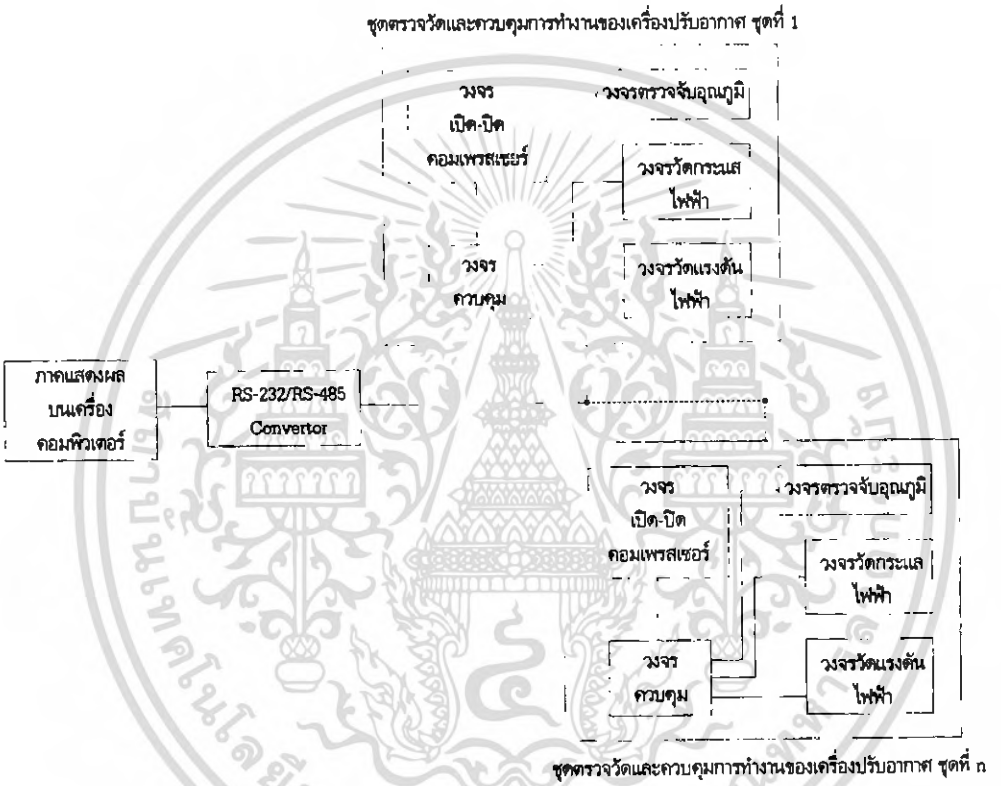
จากผลการทดลองการเชื่อมต่อกับโหลดจำลองของระบบปรับอากาศค่าที่ได้จากผลการทดลองนั้นเป็นไปตามจำนวนโหลดที่ใช้จริง เช่นหลอดไฟหนึ่งหลอดก็จะมีค่ากำลังไฟฟ้าเท่ากับ 100 วัตต์ เป็นต้น ซึ่งสามารถทำให้ทราบถึงการรับส่งข้อมูลได้

## 4.3 การทดลองการแสดงผลเมื่อเชื่อมต่อชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ

### 4.3.1 การทดลอง

ลำดับขั้นการทดลอง

1. เชื่อมต่อระบบดังรูปที่ 4.2




**รูปที่ 4.2** การเชื่อมต่อระบบการทดลอง

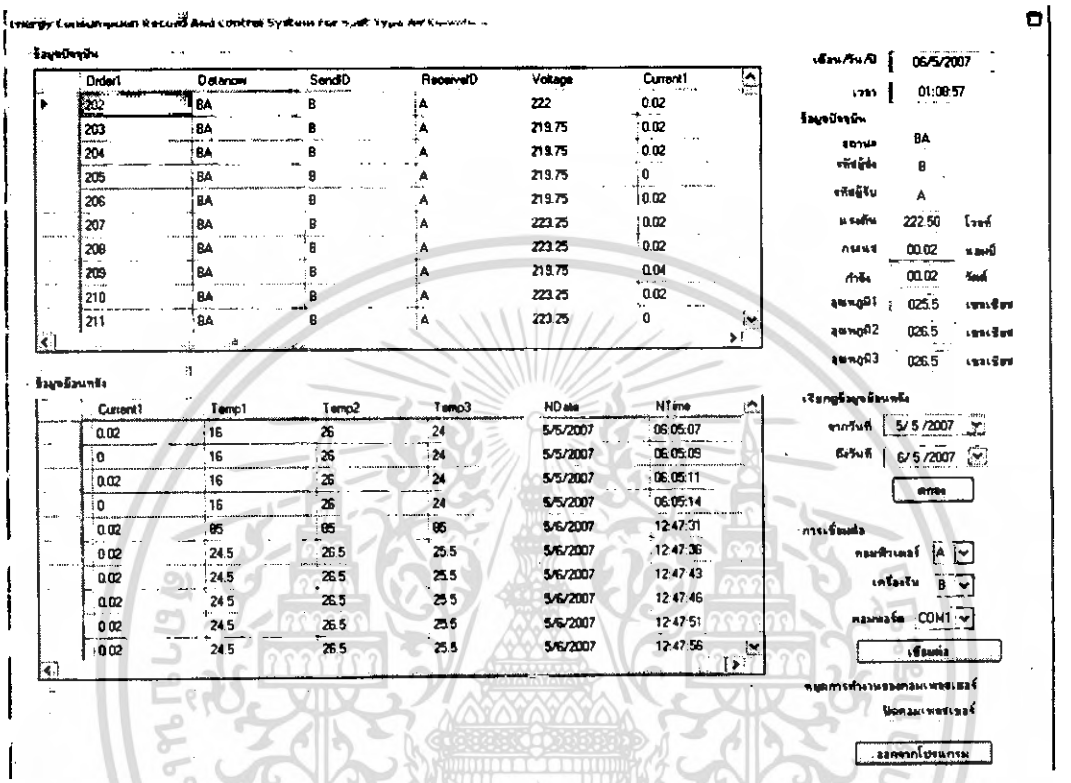
2. เปิดสวิตช์การทำงานของชุดชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ

3. เข้าสู่โปรแกรมของระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ โดยจะแสดงผลบนหน้าจอคอมพิวเตอร์

4. ทำการวัดค่าต่างๆ

### 4.3.2 ผลการทดลอง

ผลการทดลองแสดงได้ดังรูปที่ 4.3




**รูปที่ 4.3** ผลการทดลองการแสดงผลเมื่อเชื่อมต่อชุดตรวจวัดและควบคุมการทำงานเครื่องปรับอากาศ

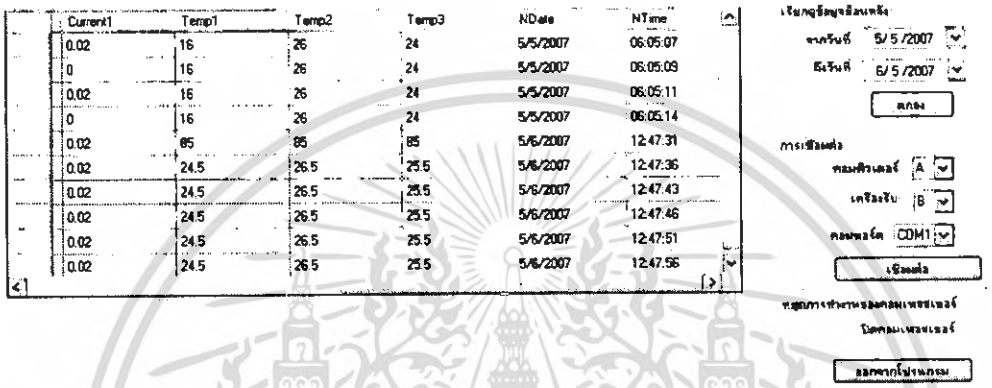
จากผลการทดลอง ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศนี้ สามารถที่จะแสดงค่าที่รับได้เป็นตัวเลข โดยสามารถเรียกดูค่าต่างๆ เหล่านี้ย้อนหลังได้อีกด้วย

## 4.4 การทดลองการเรียกดูข้อมูลย้อนหลัง

### 4.4.1 การทดลอง

ทำการทดลองโดยการวัดค่าต่างๆ ทิ้งไว้ แล้วสังเกตการเปลี่ยนแปลงแล้วทำการดูข้อมูลย้อนหลังได้จากฐานข้อมูล

ผลการทดลองแสดงดังรูปที่ 4.4




**รูปที่ 4.4** ผลการทดลองการเรียกดูข้อมูลย้อนหลัง

# บทที่ 5
# บทสรุป

## 5.1 บทสรุป

ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศประกอบไปด้วย 3 ส่วนคือ ชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ ชุดแปลงสัญญาณ RS232/RS485 และภาคแสดงผลบนเครื่องคอมพิวเตอร์ วงจรของชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศประกอบด้วย วงจรควบคุม วงจรตรวจจับอุณหภูมิ วงจรวัดแรงดันไฟฟ้า วงจรวัดกระแสไฟฟ้า วงจรเปิด-ปิดคอมเพรสเซอร์ และวงจรแหล่งจ่ายไฟ โดยระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศนี้สามารถตรวจวัดค่าของกระแสไฟฟ้า แรงดันไฟฟ้า และอุณหภูมินำไปแสดงผลบนหน้าจอคอมพิวเตอร์ โดยที่หน้าจอคอมพิวเตอร์นั้นจะแสดงค่าข้อมูลต่างๆ ในลักษณะที่เป็นตัวเลข โดยค่าของข้อมูลเหล่านี้จะมีการเปลี่ยนแปลงอยู่ตลอดเวลา และสามารถที่จะเรียกดูผลของข้อมูลย้อนหลังได้ ตามช่วงเวลา/วัน/สัปดาห์/เดือน/ปี ที่ต้องการ โดยระบบนี้สามารถส่งข้อมูลผ่านระบบสื่อสารแบบอนุกรม RS 485 ได้อีกด้วย

ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศนี้เหมาะสำหรับผู้ที่สนใจในการประหยัดพลังงานไฟฟ้า เพื่อที่จะเป็นส่วนช่วยในการลดค่าใช้จ่ายของค่าไฟฟ้า และเสริมสร้างลักษณะนิสัยที่ดีในการใช้พลังงานอีกด้วย

## 5.2 ปัญหาและแนวทางแก้ไข

จากการดำเนินการสร้างและทดสอบโครงงานพบว่ามีปัญหาเกิดขึ้นในการทำงานหลายประการ ซึ่งสรุปได้ดังนี้

1. **ปัญหา** วงจรวัดแรงดันเกิดการระเบิดบ่อยครั้งเนื่องจากแรงดันที่เข้ามามากเกินไป
   **แนวทางแก้ไข** เพิ่มหม้อแปลงไฟฟ้าให้กับวงจรเพื่อที่จะเปลี่ยนแปลงแรงดันให้ลดน้อยลง
2. **ปัญหา** โปรแกรม Visual Basic.Net ที่ใช้ส่วนใหญ่จะมีเนื้อหาในหนังสือไม่เพียงพอในการเขียนโปรแกรม
   **แนวทางแก้ไข** ทำการค้นคว้าข้อมูลจากอินเตอร์เน็ตและปรึกษากับผู้มีความรู้แล้วนำข้อมูลที่ได้มาใช้ในการเขียนโปรแกรมต่อไป
3. **ปัญหา** การส่งสัญญาณในบางครั้ง ข้อมูลไม่แสดงที่บนหน้าจอเครื่องคอมพิวเตอร์
   **แนวทางแก้ไข** เปลี่ยนสายนำสัญญาณของพอร์ท RS 232 ใหม่ เนื่องจากสายตรงหัวต่อหลุด

## 5.3 แนวทางการพัฒนา

1. สามารถนำชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศไปใช้กับห้องอื่นๆ ได้
2. เปลี่ยนจากการส่งสัญญาณจากคู่สายโทรศัพท์เป็นการส่งในระบบ Local Area Network (LAN)
3. ระบบที่สร้างขึ้นนี้ใช้กับระบบปรับอากาศได้เพียงอย่างเดียว ควรพัฒนาให้มีการใช้งานในด้านอื่นๆ ได้ด้วย

# บรรณานุกรม

จิตปกรณ์ เตโชพันธ์ และคณะ."ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศแบบแยกส่วน" ปริญญานิพนธ์ ครุศาสตร์อุตสาหกรรมบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม คณะครุศาสตร์อุตสาหกรรม, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2547

ธีรวัฒน์ ประกอบผล. **การประยุกต์ใช้งานไมโครคอนโทรลเลอร์.** พิมพ์ครั้งที่7. กรุงเทพฯ : สำนักพิมพ์ ส.ส.ท. 2546.

ธีรวัฒน์ ประกอบผล. **การพัฒนาไมโครคอนโทรลเลอร์ด้วยภาษาซี.** พิมพ์ครั้งที่ 2. กรุงเทพฯ : สำนักพิมพ์ ส.ส.ท. 2545.

ศุภชัย สมพานิช. **Database Programming กับ VB.NET.**นนทบุรี : สำนักพิมพ์อินโฟเพรส. 2545.

ศักรินทร์ โสนันทะ. **เครื่องมือวัดและการวัดทางไฟฟ้า.** กรุงเทพฯ : ซีเอ็ดยูเคชั่น. 2545.

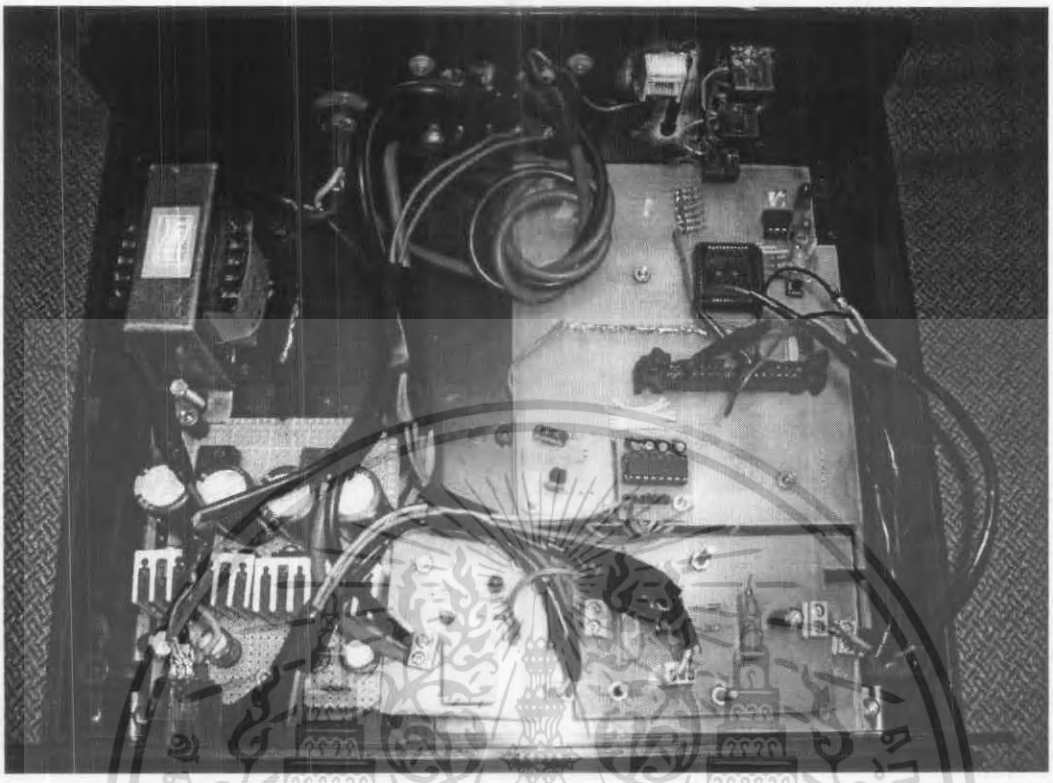
สมศักดิ์ สุโมตยกุล. **หลักการทำงานและเทคนิคการตรวจซ่อมเครื่องทำความเย็นและเครื่องปรับอากาศ.** กรุงเทพฯ : ซีเอ็ดยูเคชั่น. 2545.

สุรสิทธิ์ คิวประสพศักดิ์ และนันทนี แขวงโสภา. **อินไซท์ Visual Basic NET.** กรุงเทพฯ : สำนักพิมพ์โปรวิชั่น. 2546

# ภาคผนวก ก
# เครื่องต้นแบบ

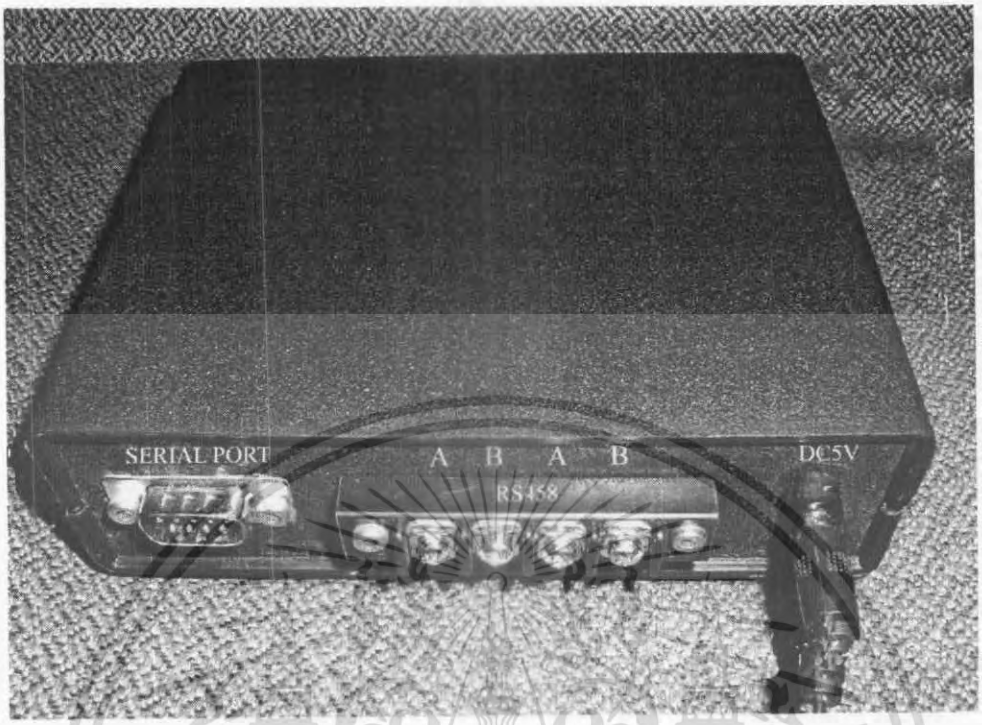

**รูปที่ ก.1** ภาพด้านหน้าและด้านบนของชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ


**รูปที่ ก.2** ภาพด้านหลังของชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ



**รูปที่ ก.3** ภาพภายในของชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ


**รูปที่ ก.4** ภาพด้านหน้าและด้านบนชุด RS232/RS485 Converter

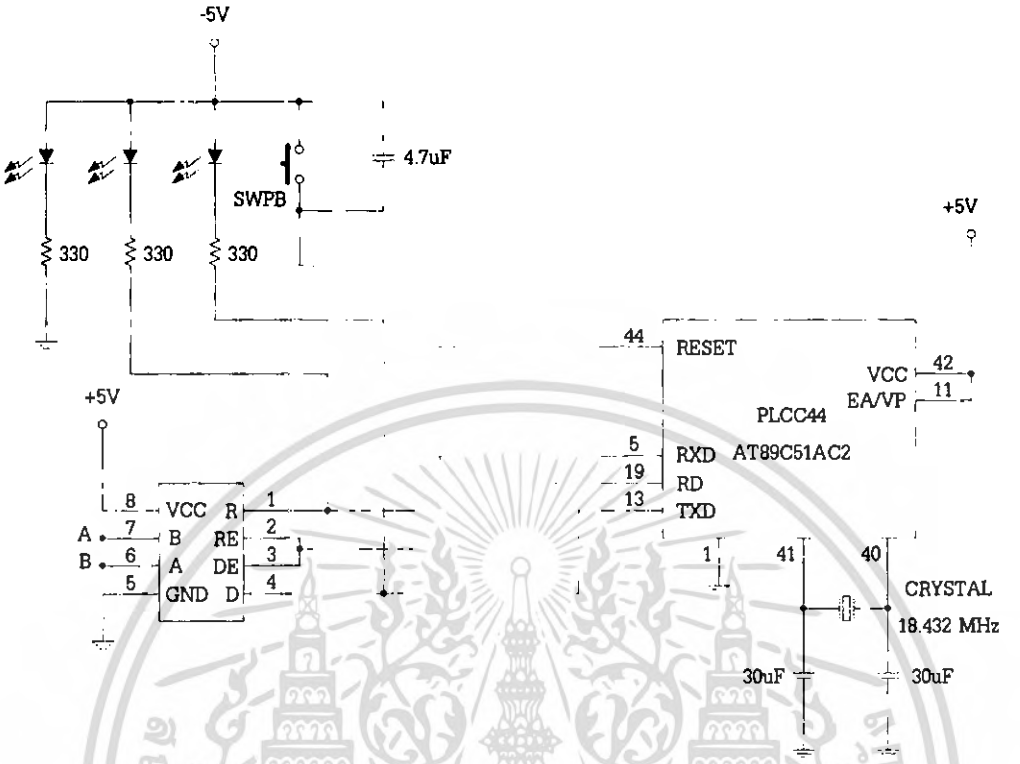



**รูปที่ ก.5** ภาพด้านหลังของชุด RS232/RS485 Converter

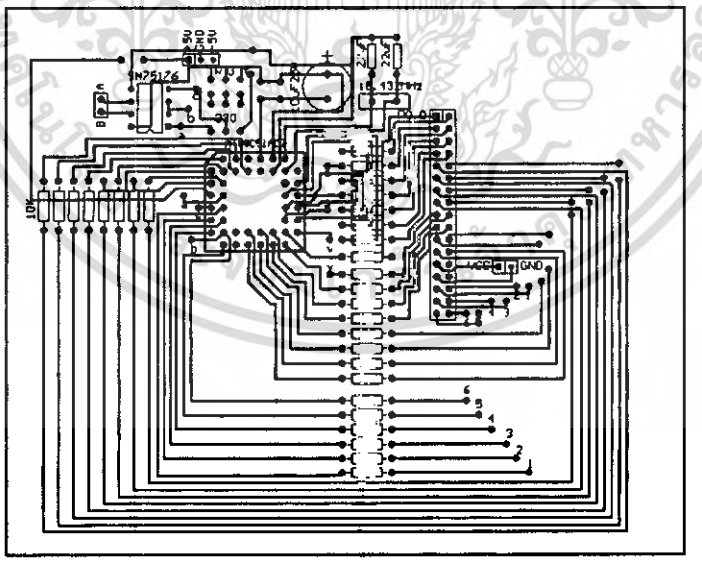

**รูปที่ ก.6** ภาพภายในของชุด RS232/RS485 Converter

# ภาคผนวก ข

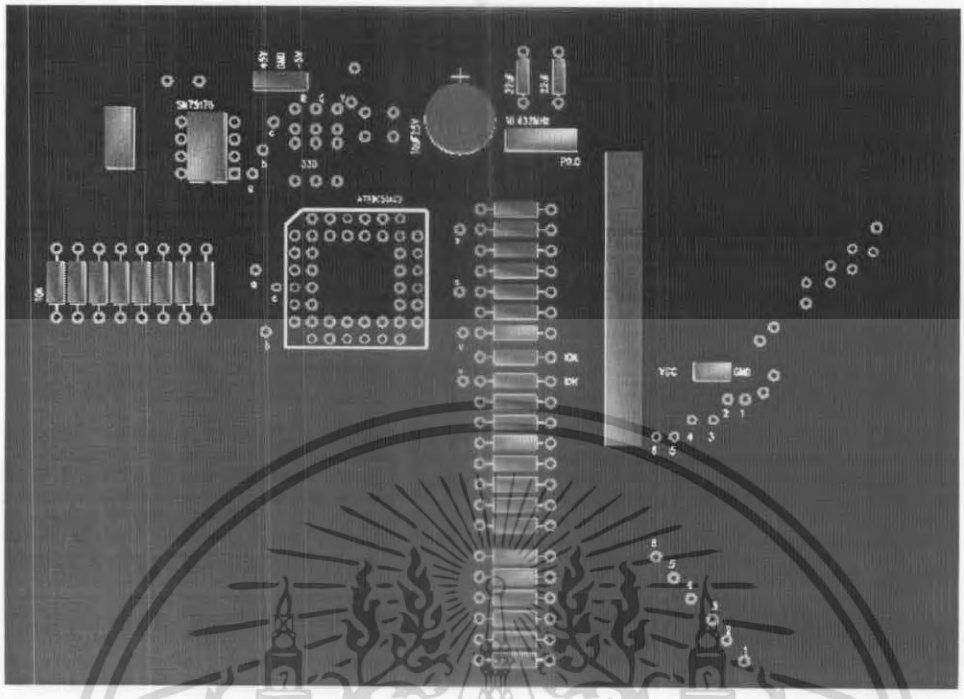
# วงจรและแผ่นวงจรพิมพ์




**รูปที่ ข.1** วงจรควบคุม

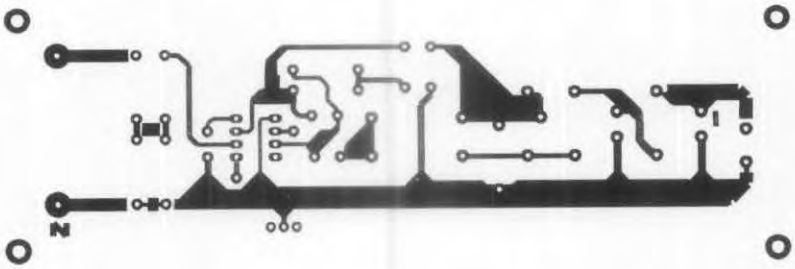


**รูปที่ ข.2** แผ่นวงจรพิมพ์วงจรควบคุม

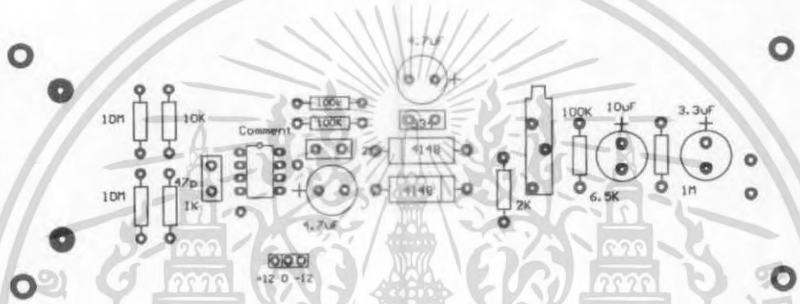


**รูปที่ ข.3** ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจรควบคุม

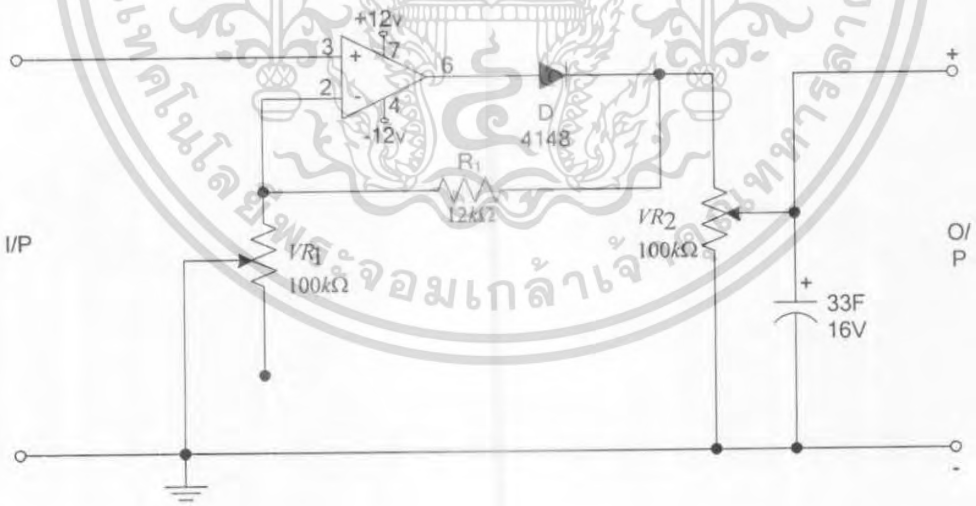

**รูปที่ ข.4** วงจรวัดแรงดันไฟฟ้า



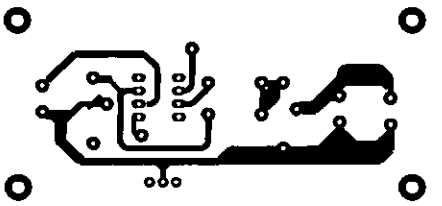
**รูปที่ ข.5** แผ่นวงจรพิมพ์วงจรวัดแรงดันไฟฟ้า



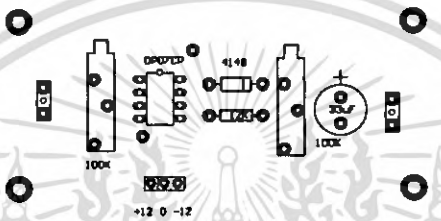

**รูปที่ ข.6** ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจรวัดแรงดันไฟฟ้า



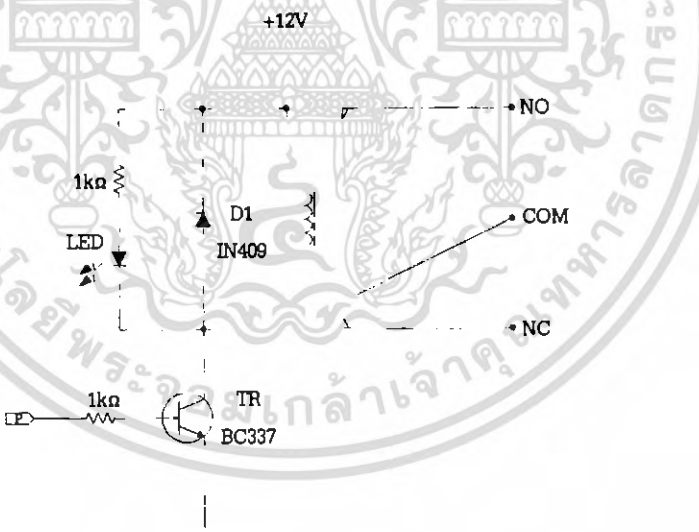

**รูปที่ ข.7** วงจรวัดกระแสไฟฟ้า



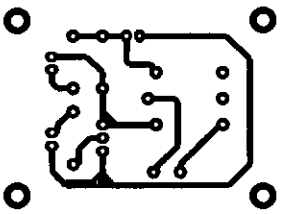
**รูปที่ ข.8** แผ่นวงจรพิมพ์วงจรวัดกระแสไฟฟ้า



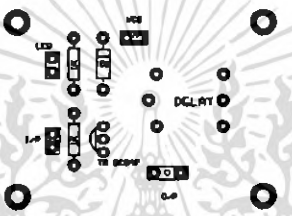

**รูปที่ ข.9** ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจรวัดกระแสไฟฟ้า



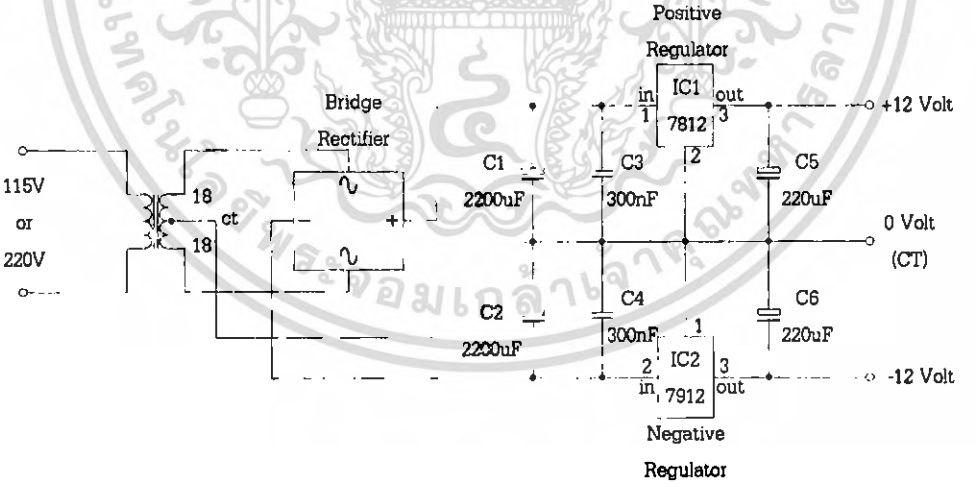

**รูปที่ ข.10** วงจรเปิด-ปิดคอมเพรสเซอร์



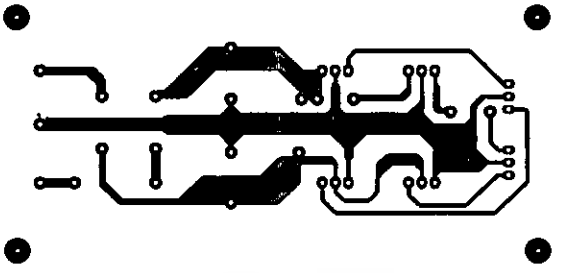
**รูปที่ ข.11** แผ่นวงจรพิมพ์วงจรเปิด-ปิดคอมเพรสเซอร์



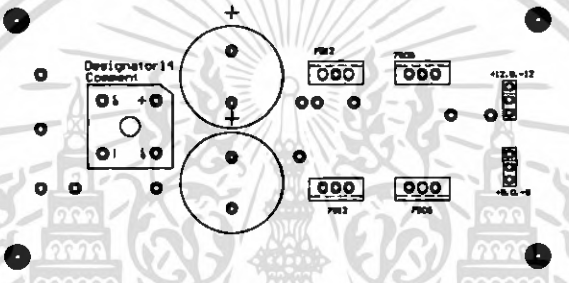
**รูปที่ ข.12** ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจรเปิด-ปิดคอมเพรสเซอร์



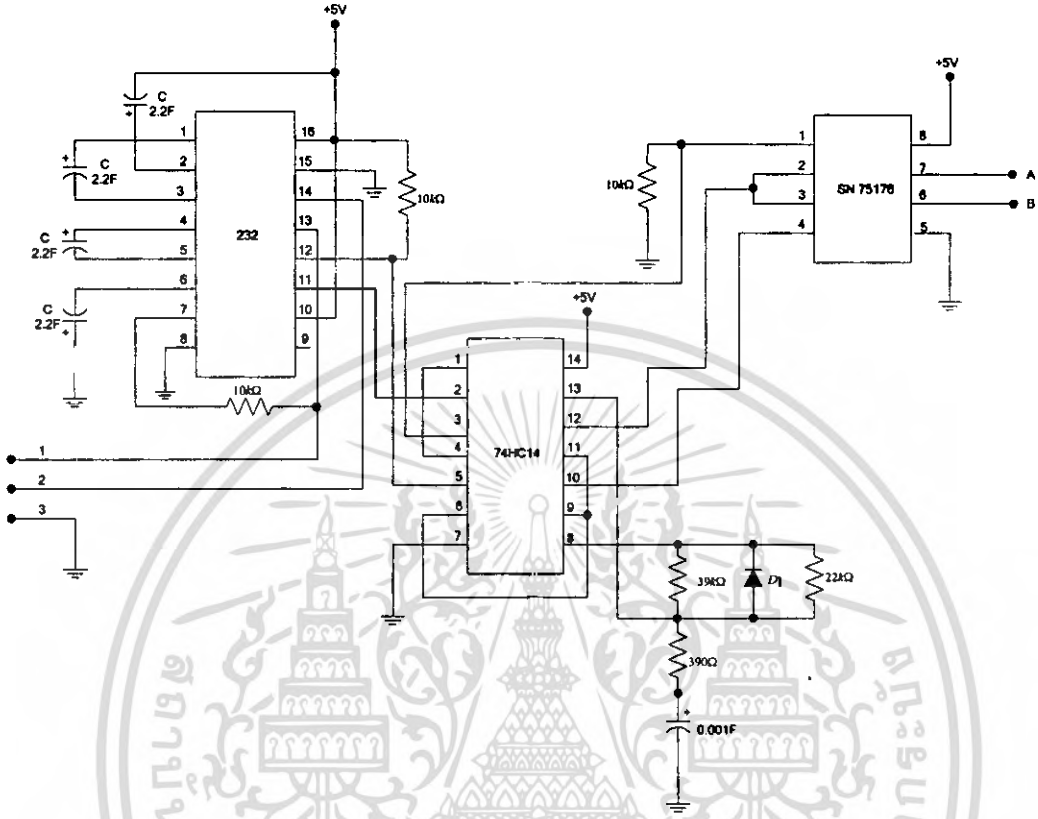

**รูปที่ ข.13** วงจรแหล่งจ่ายไฟ



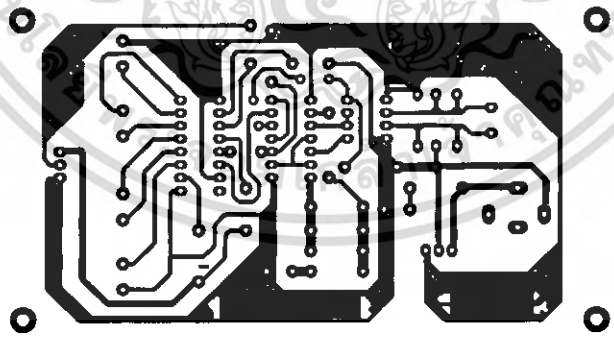
**รูปที่ ข.14** แผ่นวงจรพิมพ์วงจรแหล่งจ่ายไฟ



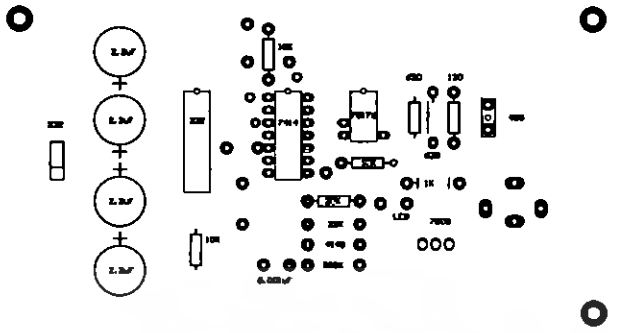
**รูปที่ ข.15** ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจรแหล่งจ่ายไฟ




**รูปที่ ข.16** วงจรแปลงสัญญาณ RS232/RS485



**รูปที่ ข.17** แผ่นวงจรพิมพ์วงจรแปลงสัญญาณ RS232/RS485



**รูปที่ ข.18** ตำแหน่งการวางอุปกรณ์แผ่นวงจรพิมพ์วงจรแปลงสัญญาณ RS232/RS485

# ภาคผนวก ค

# รายการอุปกรณ์

**ตารางที่ ค.1** รายการของอุปกรณ์ของวงจรควบคุม

| ชื่ออุปกรณ์ | รายละเอียด | จำนวน |
|---|---|---|
| **วงจรรวม** | | |
| IC1 | SN75176 | 1 ตัว |
| IC2 | AT 89C51AC2 | 1 ตัว |
| IC3 | MAX 232 | 1 ตัว |
| **อุปกรณ์สารกึ่งตัวนำ** | | |
| LED | สีแดง สีเหลือง สีเขียว | 3 ตัว |
| **ตัวเก็บประจุ** | | |
| C1 | 4.7 μF | 1 ตัว |
| C2,C3 | 30 pF | 2 ตัว |
| **ตัวความต้านทาน** | | |
| R1,R2,R3 | 330 Ω | 3 ตัว |
| **อุปกรณ์อื่นๆ** | | |
| SW | กดติดปล่อยดับ | 1 ตัว |
| XTAL | 18.432 MHz | 1 ตัว |

**ตารางที่ ค.2** รายการของอุปกรณ์ของวงจรวัดแรงดันไฟฟ้า

| ชื่ออุปกรณ์ | รายละเอียด | จำนวน |
|---|---|---|
| **วงจรรวม** | | |
| IC1 | TL061 | 1 ตัว |
| **อุปกรณ์สารกึ่งตัวนำ** | | |
| D1-D2 | 4148 | 2 ตัว |
| **ตัวเก็บประจุ** | | |
| C1 | 47 pF | 1 ตัว |
| C2 | 22 pF | 1 ตัว |
| C3 | 334 F | 1 ตัว |
| C4,C5 | 4.7 F | 2 ตัว |

**ตารางที่ ค.2 (ต่อ)** รายการของอุปกรณ์ของวงจรวัดแรงดันไฟฟ้า

| ชื่ออุปกรณ์ | รายละเอียด | จำนวน |
|---|---|---|
| **ตัวเก็บประจุ** | | |
| C6 | 10 F | 1 ตัว |
| C7 | 3. F | 1 ตัว |
| **ตัวความต้านทาน** | | |
| Rx,R1 | 10 MΩ | 2 ตัว |
| Ry | 1 kΩ | 1 ตัว |
| R2 | 10 kΩ | 1 ตัว |
| R3,R4 | 100 kΩ | 2 ตัว |
| R5 | 2 kΩ | 1 ตัว |
| R6 | 6.8 kΩ | 1 ตัว |
| R7 | 1 MΩ | 1 ตัว |
| VR1 | 100 kΩ | 1 ตัว |

**ตารางที่ ค.3** รายการของอุปกรณ์ของวงจรวัดกระแสไฟฟ้า

| ชื่ออุปกรณ์ | รายละเอียด | จำนวน |
|---|---|---|
| **วงจรรวม** | | |
| IC1 | OP07 | 1 ตัว |
| **อุปกรณ์สารกึ่งตัวนำ** | | |
| D1 | 4148 | 1ตัว |
| **ตัวเก็บประจุ** | | |
| C1 | 33 μF 16V | 1 ตัว |
| **ตัวความต้านทาน** | | |
| R1 | 12 kΩ | 1 ตัว |
| VR1,VR2 | 100 kΩ | 2 ตัว |

**ตารางที่ ค.4** รายการของอุปกรณ์ของวงจรเปิด-ปิดคอมเพรสเซอร์

| ชื่ออุปกรณ์ | รายละเอียด | จำนวน |
|---|---|---|
| **อุปกรณ์สารกึ่งตัวนำ** | | |
| D1 | IN409 | 1 ตัว |
| LED | สีแดง | 1 ตัว |
| **ตัวความต้านทาน** | | |
| R1,R2 | 1 kΩ | 2 ตัว |
| **อุปกรณ์อื่นๆ** | | |
| TR1 | BC337 | 1 ตัว |
| รีเลย์ | รีเลย์ | 1 ตัว |

**ตารางที่ ค.5** รายการของอุปกรณ์ของวงจรแหล่งจ่ายไฟ

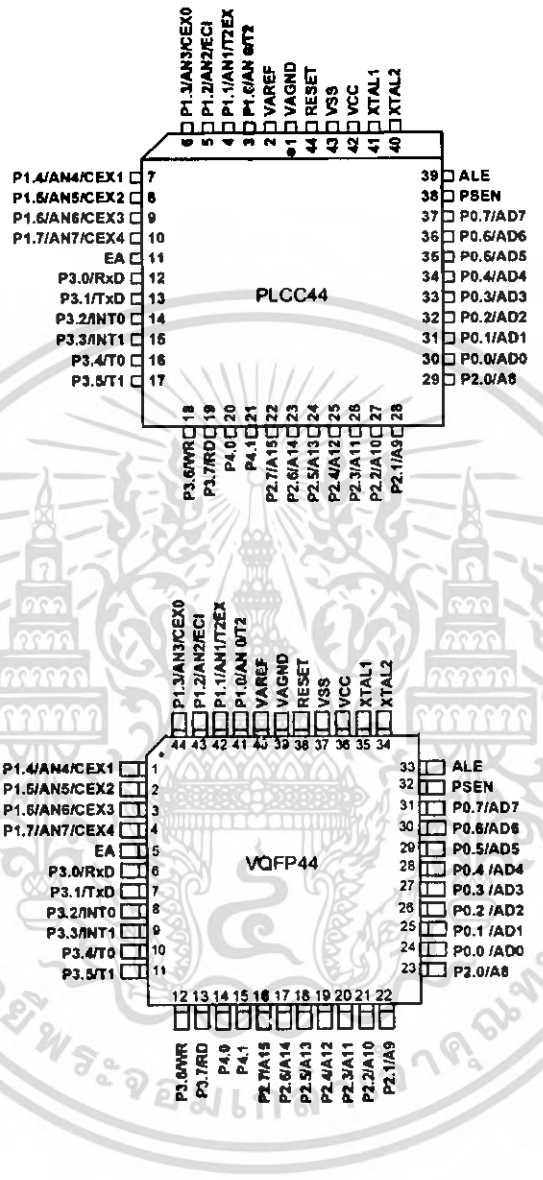
| ชื่ออุปกรณ์ | รายละเอียด | จำนวน |
|---|---|---|
| **วงจรรวม** | | |
| IC1 | 7812 | 1 ตัว |
| IC2 | 7805 | 1 ตัว |
| IC3 | 7912 | 1 ตัว |
| IC4 | 7905 | 1 ตัว |
| **อุปกรณ์สารกึ่งตัวนำ** | | |
| D1-D4 | IN4148 | 4 ตัว |
| **ตัวเก็บประจุ** | | |
| C1,C2 | 220 µF 50V | 2 ตัว |
| **อุปกรณ์อื่นๆ** | | |
| T1 | หม้อแปลง 220/12-0-12 | 1 ตัว |

**ตารางที่ ค.6** รายการของอุปกรณ์ของวงจรแปลงสัญญาณ RS232/RS485

| ชื่ออุปกรณ์ | รายละเอียด | จำนวน |
|---|---|---|
| **วงจรรวม** | | |
| IC1 | 232 | 1 ตัว |
| IC2 | SN75176 | 1 ตัว |
| IC3 | 74HC14 | 1 ตัว |
| **อุปกรณ์สารกึ่งตัวนำ** | | |
| D1 | 4148 | 1 ตัว |
| **ตัวเก็บประจุ** | | |
| C1-C4 | 2.2 μF | 4 ตัว |
| C5 | 0.001 μF | 1 ตัว |
| **ตัวความต้านทาน** | | |
| R1,R2,R3 | 10 kΩ | 3 ตัว |
| R4 | 39 kΩ | 1 ตัว |
| R5 | 390 Ω | 1 ตัว |
| R6 | 22 kΩ | 1 ตัว |

# ภาคผนวก ง
# รายละเอียดและคุณสมบัติของอุปกรณ์

**Pin Configuration**



| Pin Name | Type | Description |
|---|---|---|
| VSS | GND | Circuit ground |
| VCC | | Supply Voltage |
| VAREF | | Reference Voltage for ADC |
| VAGND | | Reference Ground for ADC |
| P0.0:7 | I/O | **Port 0:**<br>Is an 8-bit open drain bi-directional I/O port. Port 0 pins that have 1's written to them float, and in this state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pull-ups when emitting 1's.<br>Port 0 also outputs the code Bytes during program validation. External pull-ups are required during program verification. |
| P1.0:7 | I/O | **Port 1:**<br>Is an 8-bit bi-directional I/O port with internal pull-ups. Port 1 pins can be used for digital input/output or as analog inputs for the Analog Digital Converter (ADC). Port 1 pins that have 1's written to them are pulled high by the internal pull-up transistors and can be used as inputs in this state. As inputs, Port 1 pins that are being pulled low externally will be the source of current ($I_{IL}$, see section "Electrical Characteristic") because of the internal pull-ups. Port 1 pins are assigned to be used as analog inputs via the ADCCF register (in this case the internal pull-ups are disconnected).<br>As a secondary digital function, port 1 contains the Timer 2 external trigger and clock input; the PCA external clock input and the PCA module I/O.<br>P1.0/AN0/T2<br>Analog input channel 0,<br>External clock input for Timer/counter2.<br>P1.1/AN1/T2EX<br>Analog input channel 1,<br>Trigger input for Timer/counter2.<br>P1.2/AN2/ECI<br>Analog input channel 2,<br>PCA external clock input.<br>P1.3/AN3/CEX0<br>Analog input channel 3,<br>PCA module 0 Entry of input/PWM output.<br>P1.4/AN4/CEX1<br>Analog input channel 4,<br>PCA module 1 Entry of input/PWM output.<br>P1.5/AN5/CEX2<br>Analog input channel 5,<br>PCA module 2 Entry of input/PWM output.<br>P1.6/AN6/CEX3<br>Analog input channel 6,<br>PCA module 3 Entry of input/PWM output.<br>P1.7/AN7/CEX4<br>Analog input channel 7,<br>PCA module 4 Entry of input/PWM output.<br>Port 1 receives the low-order address byte during EPROM programming and program verification.<br>It can drive CMOS inputs without external pull-ups. |
| P2.0:7 | I/O | **Port 2:**<br>Is an 8-bit bi-directional I/O port with internal pull-ups. Port 2 pins that have 1's written to them are pulled high by the internal pull-ups and can be used as inputs in this state. As inputs, Port 2 pins that are being pulled low externally will be a source of current ($I_{IL}$, see section "Electrical Characteristic") because of the internal pull-ups. Port 2 emits the high-order address byte during accesses to the external Program Memory and during accesses to external Data Memory that uses 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1's. During accesses to external Data Memory that use 8 bit addresses (MOVX @RI), Port 2 transmits the contents of the P2 special function register.<br>It also receives high-order addresses and control signals during program validation.<br>It can drive CMOS inputs without external pull-ups. |

| Pin Name | Type | Description |
|---|---|---|
| P3.0:7 | I/O | **Port 3:**<br>Is an 8-bit bi-directional I/O port with internal pull-ups. Port 3 pins that have 1's written to them are pulled high by the internal pull-up transistors and can be used as inputs in this state. As inputs, Port 3 pins that are being pulled low externally will be a source of current ($I_{IL}$, see section "Electrical Characteristic") because of the internal pull-ups.<br>The output latch corresponding to a secondary function must be programmed to one for that function to operate (except for TxD and $\overline{WR}$). The secondary functions are assigned to the pins of port 3 as follows:<br><br>P3.0/RxD:<br>Receiver data input (asynchronous) or data input/output (synchronous) of the serial interface<br>P3.1/TxD:<br>Transmitter data output (asynchronous) or clock output (synchronous) of the serial interface<br>P3.2/$\overline{INT0}$:<br>External interrupt 0 input/timer 0 gate control input<br>P3.3/$\overline{INT1}$:<br>External interrupt 1 input/timer 1 gate control input<br>P3.4/T0:<br>Timer 0 counter input<br>P3.5/T1:<br>Timer 1 counter input<br>P3.6/$\overline{WR}$:<br>External Data Memory write strobe; latches the data byte from port 0 into the external data memory<br>P3.7/$\overline{RD}$:<br>External Data Memory read strobe; Enables the external data memory.<br>It can drive CMOS inputs without external pull-ups. |
| P4.0:1 | I/O | **Port 4:**<br>Is an 2-bit bi-directional I/O port with internal pull-ups. Port 4 pins that have 1's written to them are pulled high by the internal pull-ups and can be used as inputs in this state. As inputs, Port 4 pins that are being pulled low externally will be a source of current (IIL, on the datasheet) because of the internal pull-up transistor.<br>P4.0<br>P4.1:<br>It can drive CMOS inputs without external pull-ups. |
| RESET | I/O | **Reset:**<br>A high level on this pin during two machine cycles while the oscillator is running resets the device. An internal pull-down resistor to VSS permits power-on reset using only an external capacitor to VCC. |
| ALE | O | **ALE:**<br>An Address Latch Enable output for latching the low byte of the address during accesses to the external memory. The ALE is activated every 1/6 oscillator periods (1/3 in X2 mode) except during an external data memory access. When instructions are executed from an internal Flash ($\overline{EA}$ = 1), ALE generation can be disabled by the software. |
| PSEN | O | **$\overline{PSEN}$:**<br>The Program Store Enable output is a control signal that enables the external program memory of the bus during external fetch operations. It is activated twice each machine cycle during fetches from the external program memory. However, when executing from of the external program memory two activations of PSEN are skipped during each access to the external Data memory. The PSEN is not activated for internal fetches. |
| EA | I | **$\overline{EA}$:**<br>When External Access is held at the high level, instructions are fetched from the internal Flash when the program counter is less then 8000H. When held at the low level,AT89C51AC2 fetches all instructions from the external program memory. |
| XTAL1 | I | **XTAL1:**<br>Input of the inverting oscillator amplifier and input of the internal clock generator circuits.<br>To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected. To operate above a frequency of 16 MHz, a duty cycle of 50% should be maintained. |
| XTAL2 | O | **XTAL2:**<br>Output from the inverting oscillator amplifier. |

intersil™

# ICL7135

Data Sheet | December 2000 | File Number 3093.2

## $4^1/_2$ Digit, BCD Output, A/D Converter

The Intersil ICL7135 precision A/D converter, with its multiplexed BCD output and digit drivers, combines dual-slope conversion reliability with ±1 in 20,000 count accuracy and is ideally suited for the visual display DVM/DPM market. The 2.0000V full scale capability, auto-zero, and auto-polarity are combined with true ratiometric operation, almost ideal differential linearity and true differential input. All necessary active devices are contained on a single CMOS IC, with the exception of display drivers, reference, and a clock.

The ICL7135 brings together an unprecedented combination of high accuracy, versatility, and true economy. It features auto-zero to less than 10µV, zero drift of less than 1µV/°C, input bias current of 10pA (Max), and rollover error of less than one count. The versatility of multiplexed BCD outputs is increased by the addition of several pins which allow it to operate in more sophisticated systems. These include $\overline{\text{STROBE}}$, OVERRANGE, UNDERRANGE, RUN/$\overline{\text{HOLD}}$ and BUSY lines, making it possible to interface the circuit to a microprocessor or UART.

## Features

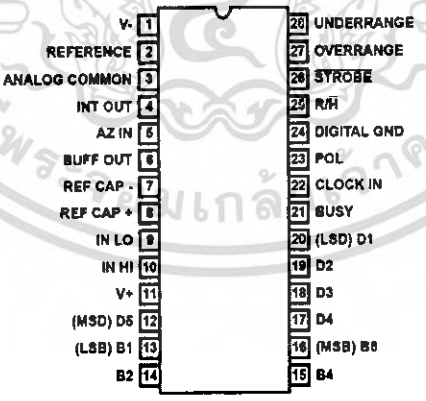
- Accuracy Guaranteed to ±1 Count Over Entire ±20000 Counts (2.0000V Full Scale)
- Guaranteed Zero Reading for 0V Input
- 1pA Typical Input Leakage Current
- True Differential Input
- True Polarity at Zero Count for Precise Null Detection
- Single Reference Voltage Required
- Overrange and Underrange Signals Available for Auto-Range Capability
- All Outputs TTL Compatible
- Blinking Outputs Gives Visual Indication of Overrange
- Six Auxiliary Inputs/Outputs are Available for Interfacing to UARTs, Microprocessors, or Other Circuitry
- Multiplexed BCD Outputs

## Ordering Information

| PART NUMBER | TEMP. RANGE (°C) | PACKAGE | PKG. NO. |
|---|---|---|---|
| ICL7135CPI | 0 to 70 | 28 Ld PDIP | E28.6 |

## Pinout

ICL7135
(PDIP)
TOP VIEW



## Absolute Maximum Ratings

Supply Voltage V+ ..... +6V
V- ..... -9V
Analog Input Voltage (Either Input) (Note 1) ..... V+ to V-
Reference Input Voltage (Either Input) ..... V+ to V-
Clock Input Voltage ..... GND to V+

## Operating Conditions

Temperature Range ..... 0°C to 70°C

## Thermal Information

Thermal Resistance (Typical, Note 2) ..... $\theta_{JA}$ (°C/W)
PDIP Package ..... 55
Maximum Junction Temperature ..... 150°C
Maximum Storage Temperature Range ..... -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) ..... 300°C

*CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.*

NOTES:

1. Input voltages may exceed the supply voltages provided the input current is limited to ±100µA.
2. $\theta_{JA}$ is measured with the component mounted on a low effective thermal conductivity test board in free air. See Tech Brief TB379 for details.

## Electrical Specifications

$V+ = +5V$, $V- = -5V$, $T_A = 25°C$, $f_{CLK}$ Set for 3 Readings/s, Unless Otherwise Specified

| PARAMETER | | TEST CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|---|
| **ANALOG** (Notes 3, 4) | | | | | | |
| Zero Input Reading | | $V_{IN} = 0V$, $V_{REF} = 1.000V$ | -00000 | +00000 | +00000 | Counts |
| Ratiometric Error (Note 4) | | $V_{IN} = V_{REF} = 1.000V$ | -3 | 0 | +3 | Counts |
| Linearity Over ± Full Scale (Error of Reading from Best Straight Line) | | $-2V \le V_{IN} \le +2V$ | - | 0.5 | 1 | LSB |
| Differential Linearity (Difference Between Worse Case Step of Adjacent Counts and Ideal Step) | | $-2V \le V_{IN} \le +2V$ | - | 0.01 | - | LSB |
| Rollover Error (Difference in Reading for Equal Positive and Negative Voltage Near Full Scale) | | $-V_{IN} = +V_{IN} = 2V$ | - | 0.5 | 1 | LSB |
| Noise (Peak-to-Peak Value Not Exceeded 95% of Time), $e_N$ | | $V_{IN} = 0V$, Full scale = 2.000V | - | 15 | - | µV |
| Input Leakage Current, $I_{ILK}$ | | $V_{IN} = 0V$ | - | 1 | 10 | pA |
| Zero Reading Drift (Note 7) | | $V_{IN} = 0V$, 0°C to 70°C | - | 0.5 | 2 | µV/°C |
| Scale Factor Temperature Coefficient, $T_C$ (Notes 5 and 7) | | $V_{IN} = +2V$, 0°C to 70°C Ext. Ref. 0ppm/°C | - | 2 | 5 | ppm/°C |
| **DIGITAL INPUTS** | | | | | | |
| Clock In, Run/Hold (See Figure 2) | $V_{INH}$ | | 2.8 | 2.2 | - | V |
| | $V_{INL}$ | | - | 1.6 | 0.8 | V |
| | $I_{INL}$ | $V_{IN} = 0V$ | - | 0.02 | 0.1 | mA |
| | $I_{INH}$ | $V_{IN} = +5V$ | - | 0.1 | 10 | µA |
| **DIGITAL OUTPUTS** | | | | | | |
| All Outputs, $V_{OL}$ | | $I_{OL} = 1.6mA$ | - | 0.25 | 0.40 | V |
| B1, B2, B4, B8, D1, D2, D3, D4, D5, $V_{OH}$ | | $I_{OH} = -1mA$ | 2.4 | 4.2 | - | V |
| BUSY, STROBE, OVERRANGE, UNDERRANGE, POLARITY, $V_{OH}$ | | $I_{OH} = -10µA$ | 4.9 | 4.99 | - | V |
| **SUPPLY** | | | | | | |
| +5V Supply Range, V+ | | | +4 | +5 | +6 | V |
| -5V Supply Range, V- | | | -3 | -5 | -8 | V |
| +5V Supply Current, I+ | | $f_C = 0$ | - | 1.1 | 3.0 | mA |
| -5V Supply Current, I- | | $f_C = 0$ | - | 0.8 | 3.0 | mA |
| Power Dissipation Capacitance, $C_{PD}$ | | vs Clock Frequency | - | 40 | - | pF |
| **CLOCK** | | | | | | |
| Clock Frequency (Note 6) | | | DC | 2000 | 1200 | kHz |

NOTES:

3. Tested in $4\frac{1}{2}$ digit (20,000 count) circuit shown in Figure 3. (Clock frequency 120kHz.)
4. Tested with a low dielectric absorption integrating capacitor, the 27kΩ INT OUT resistor shorted, and $R_{INT} = 0$. See Component Value Selection Discussion.
5. The temperature range can be extended to 70°C and beyond as long as the auto-zero and reference capacitors are increased to absorb the higher leakage of the ICL7135.
6. This specification relates to the clock frequency range over which the ICL7135 will correctly perform its various functions See "Max Clock Frequency" section for limitations on the clock frequency range in a system.
7. Parameter guaranteed by design or characterization. Not production tested.

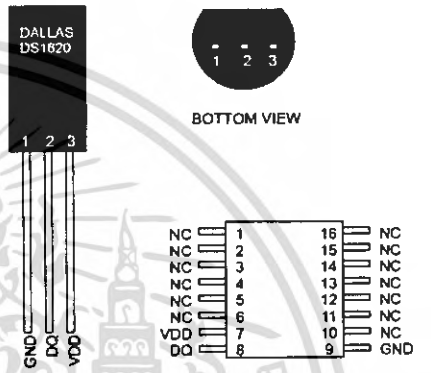
**DALLAS SEMICONDUCTOR**

# DS1820
## 1–Wire™ Digital Thermometer

### FEATURES

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

### PIN ASSIGNMENT




DS1820
PR35 PACKAGE
See Mech. Drawings Section

DS1820S
16–PIN SSOP
See Mech. Drawings Section

### PIN DESCRIPTION

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

### DESCRIPTION

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

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

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

## DETAILED PIN DESCRIPTION

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

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

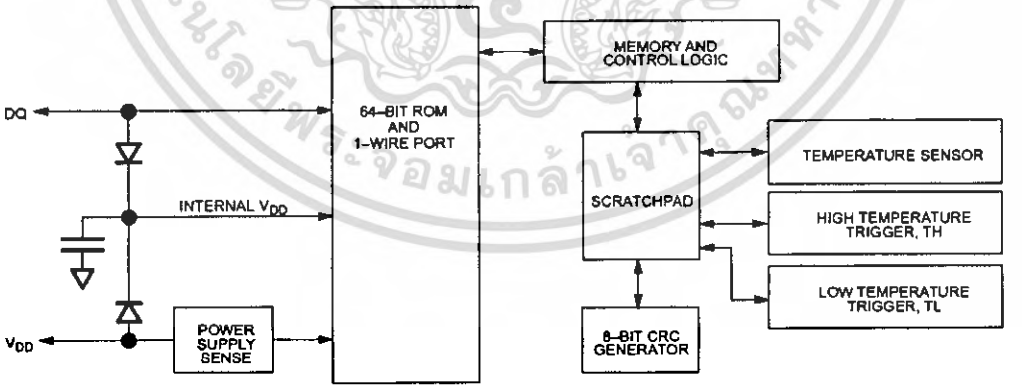
## OVERVIEW

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

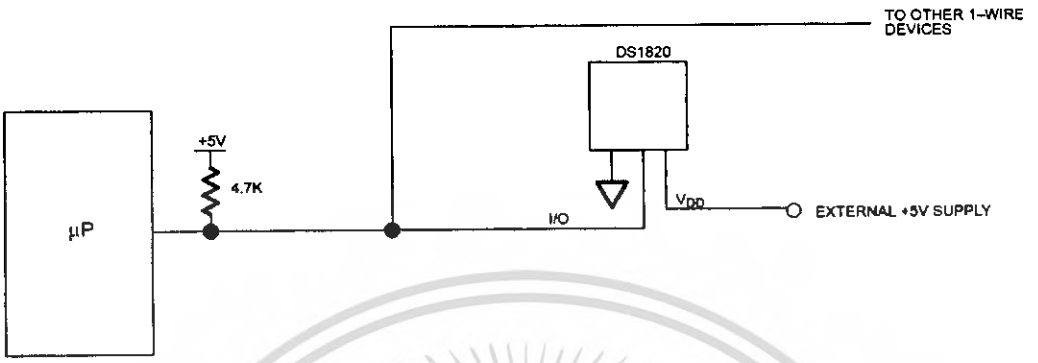
Communication to the DS1820 is via a 1–Wire port. With the 1–Wire port, the memory and control functions will not be available before the ROM function protocol has been established. The master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM,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

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

## DS1820 BLOCK DIAGRAM Figure 1



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




## OPERATION – MEASURING TEMPERATURE

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

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

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

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

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

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

| MSB | | | | | | | | LSB |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

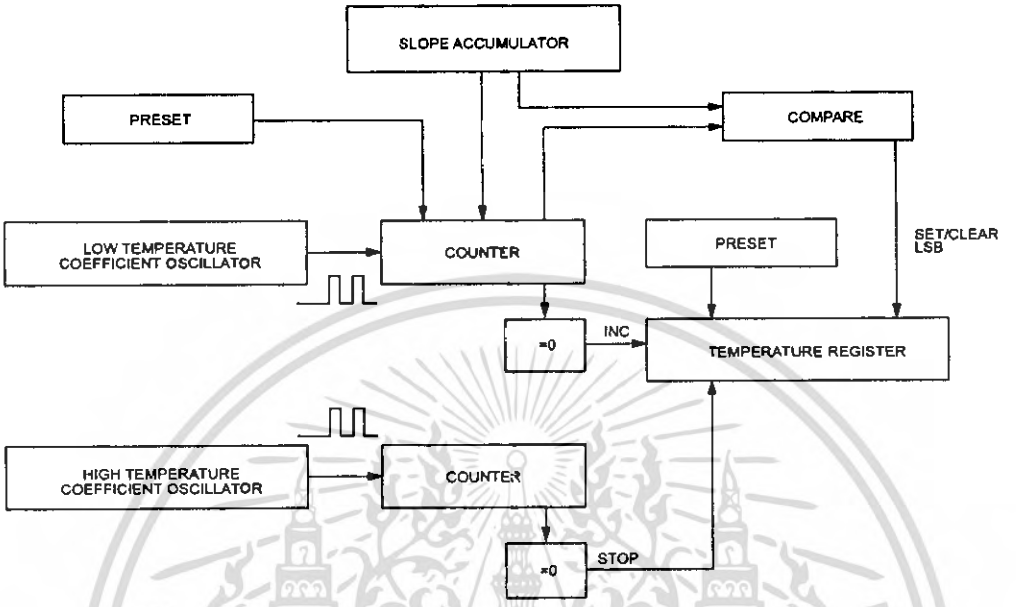
= –25°C

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

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

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

**TEMPERATURE MEASURING CIRCUITRY** Figure 4




**TEMPERATURE/DATA RELATIONSHIPS** Table 1

| TEMPERATURE | DIGITAL OUTPUT (Binary) | DIGITAL OUTPUT (Hex) |
|---|---|---|
| +125°C | 00000000 11111010 | 00FA |
| +25°C | 00000000 00110010 | 0032h |
| +1/2°C | 00000000 00000001 | 0001h |
| +0°C | 00000000 00000000 | 0000h |
| –1/2°C | 11111111 11111111 | FFFFh |
| –25°C | 11111111 11001110 | FFCEh |
| –55°C | 11111111 10010010 | FF92h |

## OPERATION – ALARM SIGNALING

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

# AppKit:
## Using the SN75176 in an RS485 Network

This AppKit shows how to create a bi-directional RS485 network using the Texas Instruments SN75176 Differential Bus Transceiver chip and a Parallax BASIC Stamp single-board computer.

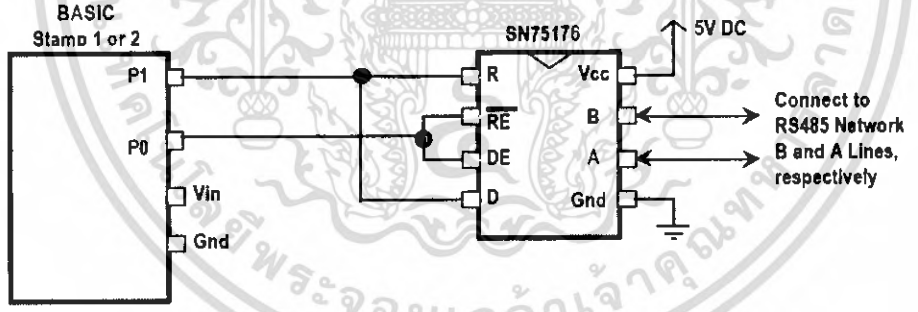
### Description

RS485 is commonly used to provide strong serial signals able to withstand long cable distances (up to 4000 feet) at high baud rates in potentially noisy electrical environments. Two wires carrying an RS485 signal (the A and B lines) provide a signal base from which many devices can communicate. Twisted pair wire is recommended for long distances, but for short distances we recommend using 24 guage wire. Up to 32 devices can be connected to an RS485 data line with the SN75176 and communicate using a data protocol. This is referred to as an RS485 network, or an RS485 drop network.

The heart of this communication is the Texas Instruments SN75176 Differential Bus Transceiver chip. This chip converts RS485 signals to RS232 TTL-level signals allowing devices that traditionally communicate over standard RS232 serial connections to communicate over a single two-wire RS485 network.

### Hardware Interface

The SN75176 chip has only 8 pins. Here is how it is connected to the BASIC Stamp:




The SN75176 requires 30 mA, therefore we recommend driving only one of these chips from the Stamp's Vdd pin provided no other components require current. Setting RE (Not Receiver Enable) to Low, the R (Receiver) pin is enabled, allowing the Stamp to receive any data coming over the A and B RS485 network lines. Setting DE (Driver Enable) to High allows the Stamp to transmit data over the RS485 network. Vcc and Vdd are electronic terms with the same meaning.

Receiving and transmitting serial data using the BASIC Stamp works like this:

To Transmit data:

1) Set P0 to High.
2) Use SEROUT on P1 to transmit the data.

To Receive Data:

1) Set P0 to Low.
2) Use SERIN to receive serial data on P1

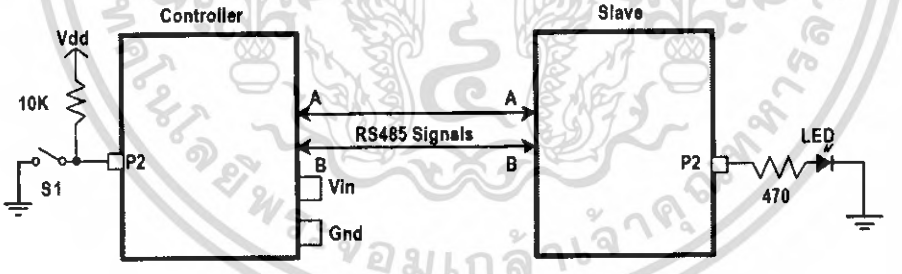
### Setting Up the Network

An RS485 network operates with a controller/slave communication protocol. One controller commands everything, and one or more slaves respond to the commands. This example uses one BASIC Stamp as the controller, and one BASIC Stamp as the slave. The circuit shown below can easily be expanded to have up to 32 BASIC Stamps communicating on one RS485 network by connecting additional slaves to the network A and B lines.

Construct the network by following these steps (see figure below):

1) Build two identical circuits as shown below. Designate one to be the controller and the other to be the slave.
2) On the controller connect the supplied switch between the BASIC Stamp's P2 pin and ground, and the 10K resistor between P2 and Vcc as a pull-up resistor. This makes the line +5 V unless a button is pushed in which case it goes to 0 V.
3) On the slave connect the supplied LED and resistor in series between the BASIC Stamp's P2 pin and ground.
4) Program the controller Stamp with the supplied program "control.bas" if you are using a Stamp 1, or "control.bs2" of you are using a Stamp 2.
5) Program the slave BASIC Stamp with the supplied program "slave.bas" if you are using a Stamp 1, or "slave.bs2" if you are using a Stamp 2.
6) Flick the switch on the Controller. The Slave LED will turn on or off depending on the controller's switch position.

See the program listings for details of the software communication process.



Order this document by LM311/D

MOTOROLA

# LM311
# LM211

## Highly Flexible Voltage Comparators

The ability to operate from a single power supply of 5.0 V to 30 V or ±15 V split supplies, as commonly used with operational amplifiers, makes the LM211/LM311 a truly versatile comparator. Moreover, the inputs of the device can be isolated from system ground while the output can drive loads referenced either to ground, the $V_{CC}$ or the $V_{EE}$ supply. This flexibility makes it possible to drive DTL, RTL, TTL, or MOS logic. The output can also switch voltages to 50 V at currents to 50 mA. Thus the LM211/LM311 can be used to drive relays, lamps or solenoids.

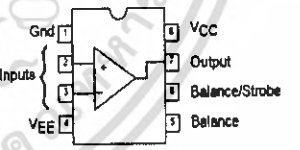
**HIGH PERFORMANCE VOLTAGE COMPARATORS**

SEMICONDUCTOR TECHNICAL DATA

N SUFFIX
PLASTIC PACKAGE
CASE 626

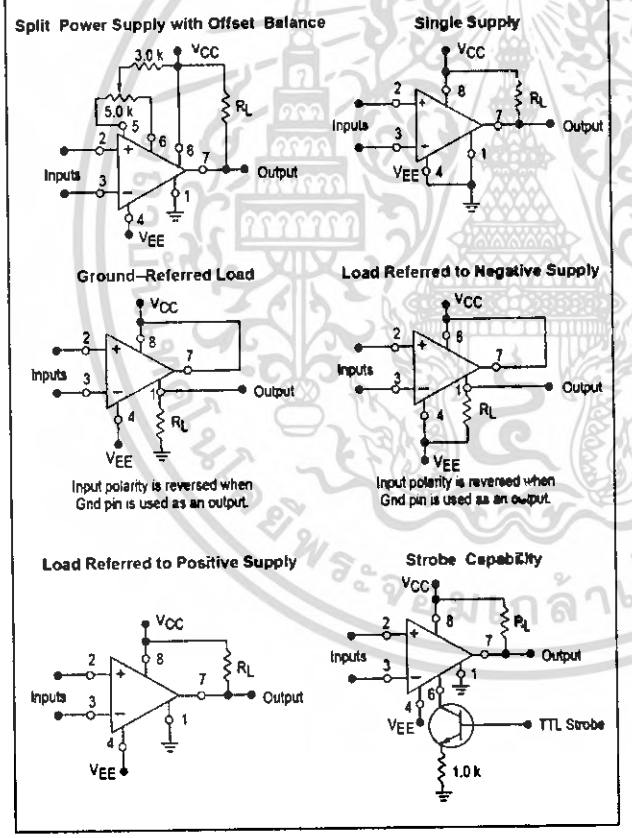
D SUFFIX
PLASTIC PACKAGE
CASE 751
(SO–8)

**PIN CONNECTIONS**




(Top View)

**Typical Comparator Design Configurations**




**ORDERING INFORMATION**

| Device | Operating Temperature Range | Package |
|---|---|---|
| LM211D | $T_A$ = 25° to +85°C | SO–8 |
| LM311D<br>LM311N | $T_A$ = 0° to +70°C | SO–8<br>Plastic DIP |



Rev 5

## LM311 LM211

**MAXIMUM RATINGS** ($T_A$ = +25°C, unless otherwise noted.)

| Rating | Symbol | LM211 | LM311 | Unit |
|---|---|---|---|---|
| Total Supply Voltage | $V_{CC}$ + $\|V_{EE}\|$ | 36 | 36 | Vdc |
| Output to Negative Supply Voltage | $V_O$ –$V_{EE}$ | 50 | 40 | Vdc |
| Ground to Negative Supply Voltage | $V_{EE}$ | 30 | 30 | Vdc |
| Input Differential Voltage | $V_{ID}$ | ±30 | ±30 | Vdc |
| Input Voltage (Note 2) | $V_{in}$ | ±15 | ±15 | Vdc |
| Voltage at Strobe Pin | – | $V_{CC}$ to $V_{CC}$–5 | $V_{CC}$ to $V_{CC}$–5 | Vdc |
| Power Dissipation and Thermal Characteristics<br>Plastic DIP<br>Derate Above $T_A$ = +25°C | <br>$P_D$<br>$1/\theta_{JA}$ | <br>625<br>5.0 | | <br>mW<br>mW/°C |
| Operating Ambient Temperature Range | $T_A$ | –25 to +85 | 0 to +70 | °C |
| Operating Junction Temperature | $T_{J(max)}$ | +150 | +150 | °C |
| Storage Temperature Range | $T_{stg}$ | –65 to +150 | –65 to +150 | °C |

**ELECTRICAL CHARACTERISTICS** ($V_{CC}$ = +15 V, $V_{EE}$ = –15 V, $T_A$ = 25°C, unless otherwise noted [Note 1].)

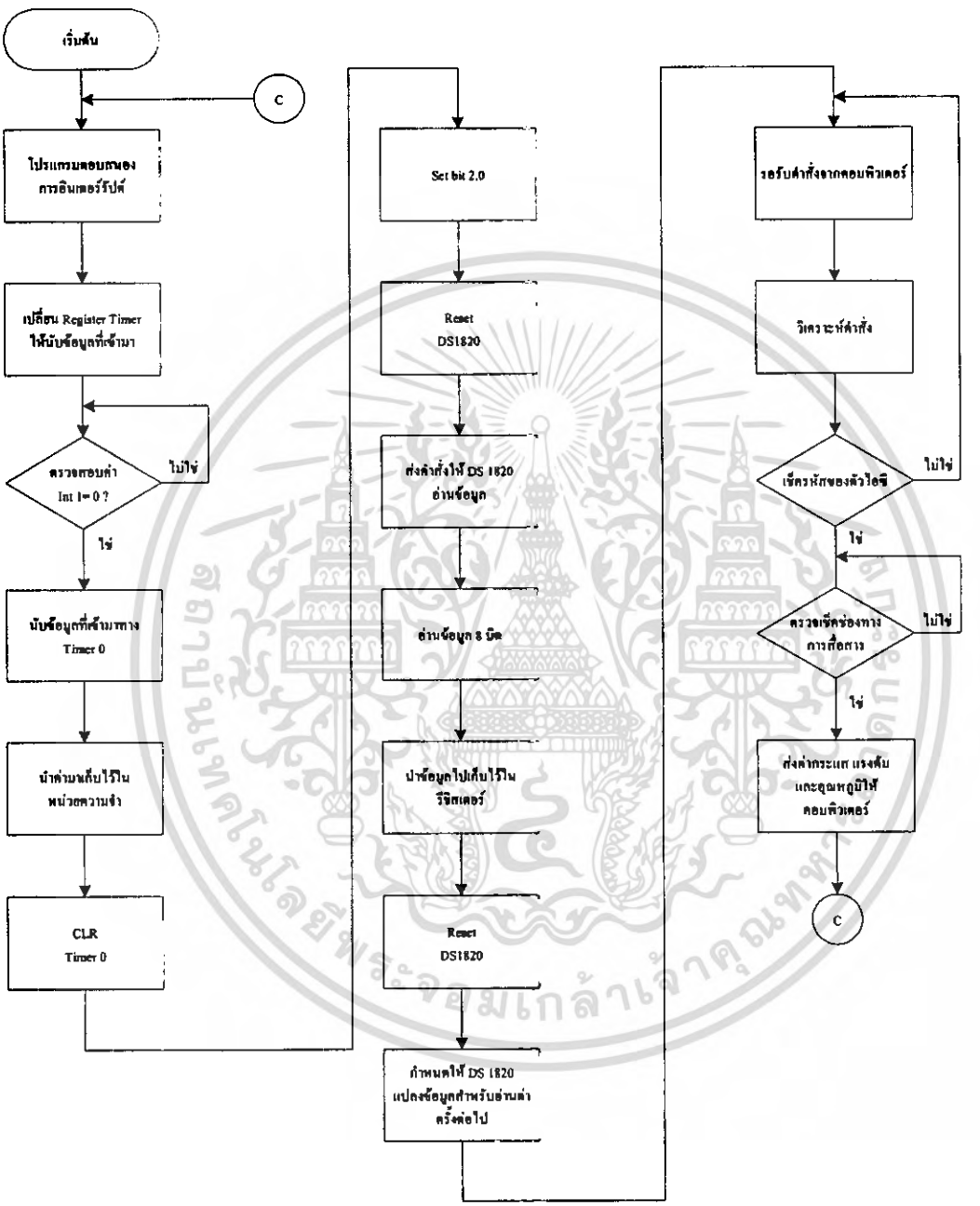
| Characteristic | Symbol | LM211 Min | LM211 Typ | LM211 Max | LM311 Min | LM311 Typ | LM311 Max | Unit |
|---|---|---|---|---|---|---|---|---|
| Input Offset Voltage (Note 3)<br>$R_S \le$ 50 kΩ, $T_A$ = +25°C<br>$R_S \le$ 50 kΩ, $T_{low} \le T_A \le T_{high}$* | $V_{IO}$ | <br>–<br>– | <br>0.7<br>– | <br>3.0<br>4.0 | <br>–<br>– | <br>2.0<br>– | <br>7.5<br>10 | mV |
| Input Offset Current (Note 3) $T_A$ = +25°C<br>$T_{low} \le T_A \le T_{high}$* | $I_{IO}$ | –<br>– | 1.7<br>– | 10<br>20 | –<br>– | 1.7<br>– | 50<br>70 | nA |
| Input Bias Current $T_A$ = +25°C<br>$T_{low} \le T_A \le T_{high}$* | $I_{IB}$ | –<br>– | 45<br>– | 100<br>150 | –<br>– | 45<br>– | 250<br>300 | nA |
| Voltage Gain | $A_V$ | 40 | 200 | – | 40 | 200 | – | V/mV |
| Response Time (Note 4) | | – | 200 | – | – | 200 | – | ns |
| Saturation Voltage<br>$V_{ID} \le$ –5.0 mV, $I_O$ = 50 mA, $T_A$ = 25°C<br>$V_{ID} \le$ –10 mV, $I_O$ = 50 mA, $T_A$ = 25°C<br>$V_{CC} \ge$ 4.5 V, $V_{EE}$ = 0, $T_{low} \le T_A \le T_{high}$*<br>$V_{ID} \le$ 6.0 mV, $I_{sink} \le$ 8.0 mA<br>$V_{ID} \le$ 10 mV, $I_{sink} \le$ 8.0 mA | $V_{OL}$ | <br>–<br>–<br><br>–<br>– | <br>0.75<br>–<br><br>0.23<br>– | <br>1.5<br>–<br><br>0.4<br>– | <br>–<br>–<br><br>–<br>– | <br>–<br>0.75<br><br>–<br>0.23 | <br>–<br>1.5<br><br>–<br>0.4 | V |
| Strobe "On" Current (Note 5) | $I_S$ | – | 3.0 | – | – | 3.0 | – | mA |
| Output Leakage Current<br>$V_{ID} \ge$ 5.0 mV, $V_O$ = 35 V, $T_A$ = 25°C, $I_{strobe}$ = 3.0 mA<br>$V_{ID} \ge$ 10 mV, $V_O$ = 35 V, $T_A$ = 25°C, $I_{strobe}$ = 3.0 mA<br>$V_{ID} \ge$ 5.0 mV, $V_O$ = 35 V, $T_{low} \le T_A \le T_{high}$* | | <br>–<br>–<br>– | <br>0.2<br>–<br>0.1 | <br>10<br>–<br>0.5 | <br>–<br>–<br>– | <br>–<br>0.2<br>– | <br>–<br>50<br>– | <br>nA<br>nA<br>µA |
| Input Voltage Range ($T_{low} \le T_A \le T_{high}$*) | $V_{ICR}$ | –14.5 | –14.7 to 13.8 | +13.0 | –14.5 | –14.7 to 13.8 | +13.0 | V |
| Positive Supply Current | $I_{CC}$ | – | +2.4 | +6.0 | – | +2.4 | +7.5 | mA |
| Negative Supply Current | $I_{EE}$ | – | –1.3 | –5.0 | – | –1.3 | –5.0 | mA |

* $T_{low}$ = –25°C for LM211, = 0°C for LM311; $T_{high}$ = +85°C for LM211, = +70°C for LM311

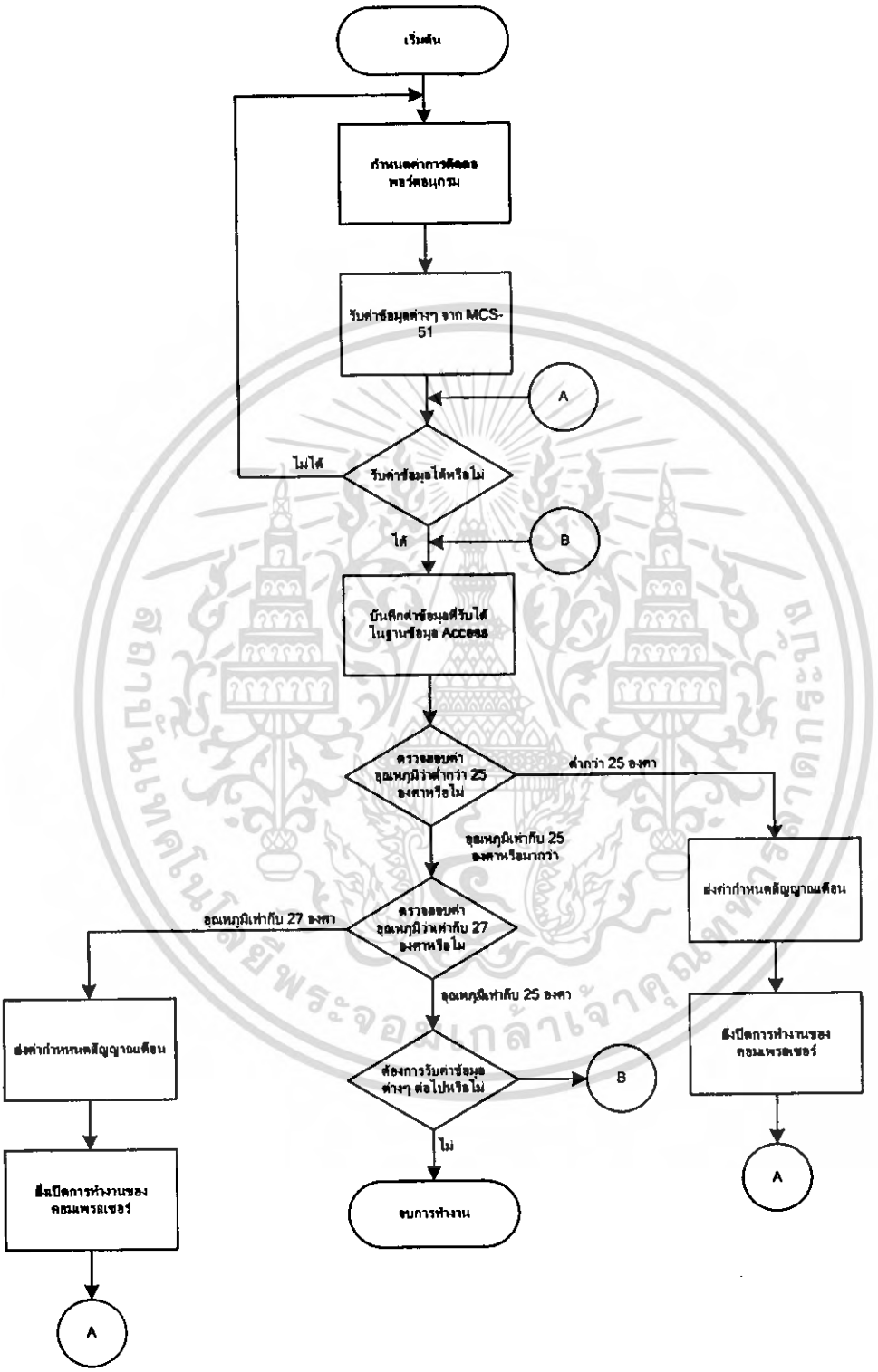
**NOTES:** 1. Offset voltage, offset current and bias current specifications apply for a supply voltage range from a single 5.0 V supply up to ±15 V supplies.
2. This rating applies for ±15 V supplies. The positive input voltage limit is 30 V above the negative supply. The negative input voltage limit is equal to the negative supply voltage or 30 V below the positive supply, whichever is less.
3. The offset voltages and offset currents given are the maximum values required to drive the output within a volt of either supply with a 1.0 mA load. Thus, these parameters define an error band and take into account the "worst case" effects of voltage gain and input impedance.
4. The response time specified is for a 100 mV input step with 5.0 mV overdrive.
5. Do not short the strobe pin to ground; it should be current driven at 3.0 mA to 5.0 mA.

# ภาคผนวก จ

# ผังงาน




**รูปที่ จ.1** ผังงานของโปรแกรมการทำงานของชุดตรวจวัดค่าต่างๆ และการแสดงบนเครื่องคอมพิวเตอร์



**รูปที่ จ.2** ผังงานของโปรแกรมหลักการรับส่งค่าที่เครื่องคอมพิวเตอร์

# ภาคผนวก ฉ
# รหัสต้นฉบับของโปรแกรม

# โปรแกรมควบคุมการทำงานของเครื่อง

```
#include <t89c51ac2.h>
#include<stdio.h>
                    // Convert Crystal 11.0592 ->> 18.432 MHz    = N Multiple 1.63 ,or eahch while = 5us.
sbit TMDAT = P0^0;
sbit TMDAT1 = P0^1;
sbit TMDAT2 = P0^2;
sbit LED_0 = P0^3;

unsigned int value_converted=0x0000;/* converted value */
bit end_of_convertion=0;        /* software flag */
void serialport_initialize (void)  // Open Serial Port to Connect.
{
        TMOD = 0x20; // timer 1 mode 2 auto reload
        SCON = 0x52; // 8-bit UART mode
        TH1= 0xfb; // 9600 8n1
        TR1 = 1; // run timer1
}
void ADC_INITIALIZE(void)
{
        ADCF = 0x03;
        /* init prescaler for adc clock */
        /* Fadc = Fperiph/(2*(32-PRS)), PRS -> ADCLK[4:0] */
        ADCLK = 0x06;           /* Fosc = 16 MHz, Fadsc = 153.8khz */
        ADCON = 0x20;           /* Enable the ADC */
        EA = 1;                 /* enable interrupts */
        EADC = 1;               /* enable ADC interrupt */
}

unsigned char getByte()
{
        while(RI==0);
        RI = 0;
        return SBUF;
}
void readByte(unsigned char *tmp)
{
        if(RI) *tmp = getByte();
}
void sendByte(unsigned char tmp)
{
        while (TI == 0);
        SBUF = tmp;
        TI = 0;
}
```

```
void sendString(unsigned char *tmp)
{
        unsigned char j;
        j = 0;
        while (*(tmp+j)!= 0)
        {
                sendByte(*(tmp+j));
                j++;
        }
}
void putHEX(unsigned char ACCU)
{
   If (ACCU > 9) sendByte(ACCU+55); // convert to ASCII for A-F
    else sendByte(ACCU+48); // and for ASCII 0-9
}
void putHEXX(unsigned char i)
{
        unsigned char a;
                a = i & 0xf0; a >>= 4; putHEX(a);
                putHEX(i & 0x0f);
}
void put_3digits_DEC(unsigned char buf)//send buf as DEC to serial      //**********************************
{
        unsigned char DEC[3];
        DEC[0] = buf/100;  DEC[1] = (buf%100)/10;  DEC[2] = buf%10;
        putHEX(DEC[0]);  putHEX(DEC[1]);  putHEX(DEC[2]);
}
void put_4digits_DEC(unsigned int x)//send buf as DEC to serial  //**********************************
{
        unsigned char DEC[4];
        DEC[0] = (x %10000)/1000;  DEC[1] = (x %1000)/100;
        DEC[2] = (x % 100)/10; DEC [3] = x % 10;
        putHEX(DEC[0]);  putHEX(DEC[1]);  putHEX(DEC[2]);  putHEX(DEC[3]);
}
 /*
 * FUNCTION_PURPOSE:this function setup Adc with channel 0 and 1, and start
 * 10bits convertion.
 * FUNCTION_INPUTS:void
 * FUNCTION_OUTPUTS:void
 */
unsigned int Read_cahnnel(unsigned char CH)
{
        ADCON &= ~0x07;          /* Clear the channel field ADCON[2:0] */
  //ADCON |= 0x06;          /* Select channel 6 */
  ADCON |= CH;                                          // get seting channel to read (0 - 7)
  ADCON &= ~0x40;          /* standard mode */
  ADCON |= 0x08;          /* Start conversion */
```

```
  while(!end_of_convertion);     /* wait end of convertion */
  end_of_convertion=0;           /* clear software flag */
  //return ((step * value_converted));// return ADC Volt_In to Function
  return (value_converted);                    // return ADC Read from INTERRUPT to Function
  }
  ////////////////////////////----------------------////////////////////////////
void dmsec (unsigned int count)
 {                    /* mSec Delay 11.0592 Mhz */
//unsigned int i;                                        /* Keil v5.2 */
unsigned char i;
while (count)
   {
        //i = 326; while (i>0) i--;  // 200 ************************
                                    i = 163; while (i>0) i--;  // 200 ************************
        count--;
   }
 }
void tmreset (unsigned char chipno)
 {                                  /* Reset TX */
        unsigned int i; //**********************
        //unsigned char i;
        switch (chipno)
                {
                case 0:
                TMDAT = 0;                                  /* mov p2.0,0 */
                break;
                case 1:
                TMDAT1 = 0;                                 /* mov p2.1,0 */
                break;
                case 2:
                TMDAT2 = 0;                                 /* mov p2.2,0 */
                break;
                }
        i = 167;                // 103
        //i = 200;              // 103

        while (i>0)                         /* 103 loop = 900 uS */
        i--;                                /* Approx 900 uS */
        switch (chipno)
                {
                case 0:
                TMDAT = 1;                  /* mov p2.0,0 */
                break;
                case 1:
                TMDAT1 = 1;                 /* mov p2.1,0 */
                break;
                case 2:
```

```
                    TMDAT2 = 1;                    /* mov p2.2,0 */
                    break;
            }
            i = 7; // 4
            while (i>0)   /* delay */
            i--,
 }
void tmpre (unsigned char chipno)
 {                  /* Wait for Presence RX */
unsigned int i; //*****************************
//unsigned char i;
switch(chipno)
{
case 0:
                    while (TMDAT) ;                /* loop until tmdat = 0 */
                    while (~TMDAT);                /* loop until tmdat = 1 */
                    break;
case 1:
                    while (TMDAT1) ;       /* loop until tmdat1 = 0 */
                    while (~TMDAT1);       /* loop until tmdat1 = 1 */
                    break;

case 2:
                    while (TMDAT2) ;       /* loop until tmdat2 = 0 */
                    while (~TMDAT2);       /* loop until tmdat2 = 1 */
                    break;
            }
i = 7; //4
                    while (i>0)
            i--;
 }
bit tmrbit (unsigned char chipno)
 {                  /* read one bit */
unsigned int i;//*****************************
//unsigned char i;
bit dat;
switch (chipno)
            {
                    case 0:
                            TMDAT = 0; i++;          i++; // 1
                            TMDAT = 1; i++; i++; i++; i++; // 2
                            dat = TMDAT;
                            break;
                    case 1:
                            TMDAT1 = 0; i++; i++;
                            TMDAT1 = 1; i++; i++; i++; i++;
                            dat = TMDAT1;
```

```
                                    break;
                            case 2:
                                    TMDAT2 = 0; i++; i++;
                                    TMDAT2 = 1; i++; i++; i++; i++
                                    dat = TMDAT2;
                                    break;
                }
                            i = 13;         // 8
while (i>0) i--;        /* Approx 65 uS */
                            return (dat);

 }
unsigned char tmrbyte (unsigned char chipno)
 {                          /* read one byte */
unsigned char i,j,dat;
                dat = 0;
                for (i=1;i<=8;i++)
   {
                j = tmrbit (chipno);
                dat = (j << 7) | (dat >> 1);
   }
                return (dat); // test with get some value for dat before and + tmrbit after, then show it,former bit must not change.
 }
void tmwbyte (unsigned char chipno, unsigned char dat)
 {                          /* write one byte */
                            unsigned int i;//**********************************
                            //unsigned char i;
                unsigned char j;
                bit testb;
                for (j=1;j<=8;j++)
   {
                testb = dat & 0x01;
                dat = dat >> 1;
                if (testb)
     {
switch(chipno) {
                case 0:
        TMDAT = 0;              /* Write 1 */
        i++; i++; i++; i++; // 2                  /* Approx 4 uS */
        TMDAT = 1;
        i = 13;                     // 8
                                    while (i>0) i--;        /* Approx 65 uS */
                                    break;
                case 1:
        TMDAT1 = 0;             /* Write 1 */
        i++; i++; i++; i++;                                                 /* Approx 4 uS */
        TMDAT1 = 1;
```

```
        i = 13;
                                while (i>0) i--;          /* Approx 65 uS */
                                break;
            case 2:
        TMDAT2 = 0;             /* Write 1 */
        i++; i++; i++; i++;                                   /* Approx 4 uS */
        TMDAT2 = 1;
        i = 13;
                                while (i>0) i--;          /* Approx 65 uS */
                                break;
            }
        }
else
        {
            switch(chipno) {
            case 0:
        TMDAT = 0;              /* Write 0 */
        i = 13;                         //8
                                while (i>0) i--;          /* Approx 65 uS */
        TMDAT = 1;
        i++; i++; i++; i++;   // 2      /* Approx 4 uS */
                                break;
            case 1:
        TMDAT1 = 0;             /* Write 0 */
        i = 13;
                                while (i>0) i--;          /* Approx 65 uS */
        TMDAT1 = 1;
        i++; i++; i++; i++;                               /* Approx 4 uS */
                                break;
            case 2:
        TMDAT2 = 0;             /* Write 0 */
        i = 13;
                                while (i>0) i--;          /* Approx 65 uS */
        TMDAT2 = 1;
        i++; i++; i++; i++;                               /* Approx 4 uS */
                                break;
                                        }
      }
    }
  }

void tmstart (unsigned char chipno)
 {                      /* ds1820 start convert */
                        tmreset (chipno);
            tmpre (chipno);
            dmsec (1); // 1 may like former
            tmwbyte (chipno, 0xcc);                       /* skip rom */
```

```
            tmwbyte (chipno,0x44);                          /* convert */
 }
unsigned char tmrtemp (unsigned char chipno)  //Char
 {
                                    /* read temp */
unsigned char a,b;
            tmreset (chipno);
            tmpre (chipno);
            dmsec (1);  // 1
            tmwbyte (chipno, 0xcc);  /* skip rom */
            tmwbyte (chipno, 0xbe);  /* convert  */
            a = tmrbyte(chipno);    /* LSB */ //byte 1
            b = tmrbyte(chipno);    /* MSB */ //byte 2 //
                        //if(b == 0xff) return((a<<1)|1);  // Chaek Negative value
                        //else return ((a&0xfffe)<<1);          // Positive Value
                        //*tem = (a>>1) + 1;
                        //c = ((c|a)<< 1)|1;   // add 1 for check negative in last bit.(bit 0)
                        //*tem = c;
                        return(a);
 }
unsigned char DS1820_READ(unsigned char No)//char
{
            //unsigned char Data;
            unsigned char Data;
            tmreset(No); // TMER
            tmpre(No);
            tmstart(No);
            Data = tmrtemp(No);// bit 0 = Net/Pos Sign, bit1 = 0.5 Sign, bit 2-8 = Temp Data. (Total 9 bits)
            // Cal Point and Negative
            return (Data);
            //return ((Deta>>1)+1);
}
Command()
{
            unsigned char a,rx = 0;
            readByte(&rx);
            switch(rx)
            {
            case 13:  // Title
                        rx = 0;
                        sendString("\r\n PRESS CHANELL TO READ DATA (1-7)");
                        break;
            case '1':     //ADC1       Volt
                        rx = 0;
                        //////use a for keep reading end Multiple//
                        sendString(" \r\n Voltage = ");
                        //put_4digits_DEC(Read_cahnnel(0)* 0.2933);//can read 0 - 7 CH  =  8 CH(s)
```

```
                printf("%f",(Read_cahnnel(0)* 0.2933));// max = 300 V
                sendString(" Volt(s)");
                break;
        case '2':   // ADC2        Current
                rx = 0;
                sendString("\r\n Current = ");
                //put_4digits_DEC(Read_cahnnel(1)*0.004883);//can read 0 - 7 CH  =  8 CH(s)
                printf("%f",(Read_cahnnel(1)*0.019552));// max = 20 A
                sendString(" Ampare(s)");
                break;
        case '3':   // Temp1
                rx = 0;
                sendString("\r\n ROOM1's TEMPERATURE = ");
                a = DS1820_READ(0);
                if(a&0x01) {
                put_3digits_DEC(DS1820_READ(0)>>1);
                sendString(".5");
                                }
                else {put_3digits_DEC(DS1820_READ(0)>>1);  // send mani Value
                sendString(".0");}
                 // read and send temperature from chip DS1820 number 0 (0-2)
                sendString(" DEGREES(CELCIUS)");
                        break;
case '4'://Temp2
                rx = 0;
                sendString("\r\n ROOM2's TEMPERATURE = ");
                //put_3digits_DEC(DS1820_READ(1));// read and send temperature from chip DS1820 number 0 (0-2
                a = DS1820_READ(1);
                if(a&0x01) {
                put_3digits_DEC(DS1820_READ(1)>>1);
                sendString(".5");
                }
                else {put_3digits_DEC(DS1820_READ(1)>>1); // send mani Value
                sendString(".0");}
                sendString(" DEGREES(CELCIUS)");
                        break;
case '5'://Temp3
                rx = 0;
                sendString("\r\n ROOM3's TEMPERATURE = ");
                a = DS1820_READ(2);
                if(a&0x01) {
                put_3digits_DEC(DS1820_READ(2)>>1);
                sendString(".5");
                                }
                else {put_3digits_DEC(DS1820_READ(2)>>1); // send mani Value
                sendString(".0");}
                put_3digits_DEC(DS1820_READ(2));  //**read and send temperature from chip DS1820 number 0 (0-2)
```

```
                    sendString(" DEGREES(CELCIUS)");
                        break;
            case '6'://P0.3(relay)
                    rx = 0;
                    sendString("\r\n LED P0.3 = 0x08");
                    LED_0 = 0x08;
                    break;
            case '7'://P0.3(relay)
                    rx = 0;
                    sendString("\r\n LED P0.3 = 0x00");
                    LED_0 = 0x00;
                    break;
            }
            }
void main (void)
{
serialport_initialize();
ADC_INITIALIZE();
putHEXX(0x00);//

while(1)
            {
            Command();
            //getByte(&rx);
            /*
            if(rx == 13)
            {
            sendString("\r\n");
            put_4digits_DEC(Read_cahnnel(0)*0.2933); // Max = 300 V   300/1023 : Each step of 10 bit ADC.
            put_3digits_DEC(Read_cahnnel(1)*0.004888); // Max = 5A     : 5/1023 :  Each step of 10 bit ADC.
            put_3digits_DEC(DS1820_READ(0));
            put_3digits_DEC(DS1820_READ(1));
            put_3digits_DEC(DS1820_READ(2));
            }
            */
            //rx = 0;
            }
}

/*
 * FUNCTION_PURPOSE:Adc interrupt, save ADDH and ADDL into an unsigned int
 * FUNCTION_INPUTS:void
 * FUNCTION_OUTPUTS:void
 */
void il_Adc(void) interrupt 8
{
ADCON &= ~0x10;             /* Clear the End of conversion flag */
```

```
value_converted = ADDH<<2;        /* save 8 msb bits */
value_converted |= (ADDL & 0x03);  /* save 2 lsb bits */
end_of_convertion=1;              /* set flag */
}
```

**โปรแกรมการแสดงผลบนจอเครื่องคอมพิวเตอร์**

```
Namespace WinApp4
    Public Class Form1
        Inherits System.Windows.Forms.Form

        Dim Conn As OleDbConnection = New OleDbConnection
        Dim olecmd As OleDbCommand
        Dim portsetting As Form3 = New Form3
        Dim commPortIndex As Integer
        Dim commSpeedIndex As Integer
        Dim commModeIndex As Integer
        Dim commFlowIndex As Integer
        Dim runmode As Integer
        Dim string1, string2 As String
        Dim ds As DataSet = New DataSet
        Dim dataAdap As OleDbDataAdapter

#Region " Windows Form Designer generated code "

        Public Sub New()
            MyBase.New()

            'This call is required by the Windows Form Designer.
            InitializeComponent()
            runmode = 0
            'Add any initialization after the InitializeComponent() call

        End Sub

        'Form overrides dispose to clean up the component list.
        Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
            If disposing Then
                If Not (components Is Nothing) Then
                    components.Dispose()
                End If
            End If
            MyBase.Dispose(disposing)
        End Sub

        'Required by the Windows Form Designer
        Private components As System.ComponentModel.IContainer
```

```
'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
Friend WithEvents DataGrid1 As System.Windows.Forms.DataGrid
Friend WithEvents Button1 As System.Windows.Forms.Button
Friend WithEvents AddValue As System.Windows.Forms.Button
Friend WithEvents SerialConnection1 As Sax.Communications.SerialConnection
Friend WithEvents Button3 As System.Windows.Forms.Button
Friend WithEvents Label7 As System.Windows.Forms.Label
Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
Friend WithEvents Label8 As System.Windows.Forms.Label
Friend WithEvents Label9 As System.Windows.Forms.Label
Friend WithEvents Label10 As System.Windows.Forms.Label
Friend WithEvents Label11 As System.Windows.Forms.Label
Friend WithEvents Label12 As System.Windows.Forms.Label
Friend WithEvents Label13 As System.Windows.Forms.Label
Friend WithEvents Label14 As System.Windows.Forms.Label
Friend WithEvents Label15 As System.Windows.Forms.Label
Friend WithEvents Label16 As System.Windows.Forms.Label
Friend WithEvents Label17 As System.Windows.Forms.Label
Friend WithEvents Label18 As System.Windows.Forms.Label
Friend WithEvents Label19 As System.Windows.Forms.Label
Friend WithEvents Label1 As System.Windows.Forms.Label
Friend WithEvents Label2 As System.Windows.Forms.Label
Friend WithEvents Label3 As System.Windows.Forms.Label
Friend WithEvents Label4 As System.Windows.Forms.Label
Friend WithEvents Label5 As System.Windows.Forms.Label
Friend WithEvents rxbox As System.Windows.Forms.TextBox
Friend WithEvents txbox As System.Windows.Forms.TextBox
Friend WithEvents voltbox As System.Windows.Forms.TextBox
Friend WithEvents ampbox1 As System.Windows.Forms.TextBox
Friend WithEvents ampbox2 As System.Windows.Forms.TextBox
Friend WithEvents tempbox1 As System.Windows.Forms.TextBox
Friend WithEvents tempbox2 As System.Windows.Forms.TextBox
Friend WithEvents OleDbDataAdapter1 As System.Data.OleDb.OleDbDataAdapter
Friend WithEvents OleDbSelectCommand1 As System.Data.OleDb.OleDbCommand
Friend WithEvents OleDbInsertCommand1 As System.Data.OleDb.OleDbCommand
Friend WithEvents OleDbUpdateCommand1 As System.Data.OleDb.OleDbCommand
Friend WithEvents OleDbDeleteCommand1 As System.Data.OleDb.OleDbCommand
Friend WithEvents OleDbConnection1 As System.Data.OleDb.OleDbConnection
Friend WithEvents DataSet61 As WindowsApplication4.DataSet6
Friend WithEvents DataSet11 As WindowsApplication4.DataSet1
Friend WithEvents OleDbCommand1 As System.Data.OleDb.OleDbCommand
Friend WithEvents SerialConnection2 As Sax.Communications.SerialConnection
Friend WithEvents voltchart As AxOWC10.AxChartSpace
Friend WithEvents ampchart2 As AxOWC10.AxChartSpace
```

```
Friend WithEvents tempchart1 As AxOWC10.AxChartSpace
Friend WithEvents tempchart2 As AxOWC10.AxChartSpace
Friend WithEvents ampchart1 As AxOWC10.AxChartSpace
<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.components = New System.ComponentModel.Container
    Dim resources As System.Resources.ResourceManager = New System.Resources.ResourceManager(GetType(Form1))
    Me.DataGrid1 = New System.Windows.Forms.DataGrid
    Me.DataSet61 = New WindowsApplication4.DataSet6
    Me.Button1 = New System.Windows.Forms.Button
    Me.AddValue = New System.Windows.Forms.Button
    Me.Button3 = New System.Windows.Forms.Button
    Me.Label7 = New System.Windows.Forms.Label
    Me.TextBox1 = New System.Windows.Forms.TextBox
    Me.rxbox = New System.Windows.Forms.TextBox
    Me.txbox = New System.Windows.Forms.TextBox
    Me.voltbox = New System.Windows.Forms.TextBox
    Me.ampbox1 = New System.Windows.Forms.TextBox
    Me.ampbox2 = New System.Windows.Forms.TextBox
    Me.tempbox1 = New System.Windows.Forms.TextBox
    Me.tempbox2 = New System.Windows.Forms.TextBox
    Me.Label8 = New System.Windows.Forms.Label
    Me.Label9 = New System.Windows.Forms.Label
    Me.Label10 = New System.Windows.Forms.Label
    Me.Label11 = New System.Windows.Forms.Label
    Me.Label12 = New System.Windows.Forms.Label
    Me.Label13 = New System.Windows.Forms.Label
    Me.Label14 = New System.Windows.Forms.Label
    Me.Label15 = New System.Windows.Forms.Label
    Me.Label16 = New System.Windows.Forms.Label
    Me.Label17 = New System.Windows.Forms.Label
    Me.Label18 = New System.Windows.Forms.Label
    Me.Label19 = New System.Windows.Forms.Label
    Me.Label1 = New System.Windows.Forms.Label
    Me.Label2 = New System.Windows.Forms.Label
    Me.Label3 = New System.Windows.Forms.Label
    Me.Label4 = New System.Windows.Forms.Label
    Me.Label5 = New System.Windows.Forms.Label
    Me.OleDbDataAdapter1 = New System.Data.OleDb.OleDbDataAdapter
    Me.OleDbDeleteCommand1 = New System.Data.OleDb.OleDbCommand
    Me.OleDbConnection1 = New System.Data.OleDb.OleDbConnection
    Me.OleDbInsertCommand1 = New System.Data.OleDb.OleDbCommand
    Me.OleDbSelectCommand1 = New System.Data.OleDb.OleDbCommand
    Me.OleDbUpdateCommand1 = New System.Data.OleDb.OleDbCommand
    Me.DataSet11 = New WindowsApplication4.DataSet1
    Me.OleDbCommand1 = New System.Data.OleDb.OleDbCommand
    Me.SerialConnection2 = New Sax.Communications.SerialConnection(Me.components)
    Me.voltchart = New AxOWC10.AxChartSpace
```

```
        Me.ampchart2 = New AxOWC10.AxChartSpace
        Me.tempchart1 = New AxOWC10.AxChartSpace
        Me.tempchart2 = New AxOWC10.AxChartSpace
        Me.ampchart1 = New AxOWC10.AxChartSpace
        CType(Me.DataGrid1, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.DataSet61, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.DataSet11, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.voltchart, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.ampchart2, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.tempchart1, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.tempchart2, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.ampchart1, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
        '
        'DataGrid1
        '
        Me.DataGrid1.DataMember = ""
        Me.DataGrid1.DataSource = Me.DataSet61.Table1
        Me.DataGrid1.Font = New System.Drawing.Font("Angsana New", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.DataGrid1.HeaderForeColor = System.Drawing.SystemColors.ControlText
        Me.DataGrid1.Location = New System.Drawing.Point(368, 16)
        Me.DataGrid1.Name = "DataGrid1"
        Me.DataGrid1.Size = New System.Drawing.Size(640, 296)
        Me.DataGrid1.TabIndex = 0
        '
        'DataSet61
        '
        Me.DataSet61.DataSetName = "DataSet6"
        Me.DataSet61.Locale = New System.Globalization.CultureInfo("en-US")
        '
        'Button1
        '
        Me.Button1.BackColor = System.Drawing.SystemColors.InactiveCaption
        Me.Button1.FlatStyle = System.Windows.Forms.FlatStyle.Flat
        Me.Button1.Font = New System.Drawing.Font("EucrosiaUPC", 11.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Button1.Location = New System.Drawing.Point(840, 672)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New System.Drawing.Size(104, 32)
        Me.Button1.TabIndex = 1
        Me.Button1.Text = "จบการทำงาน"
        '
        'AddValue
        '
        Me.AddValue.BackColor = System.Drawing.SystemColors.InactiveCaptionText
        Me.AddValue.FlatStyle = System.Windows.Forms.FlatStyle.Flat
```

```
        Me.AddValue.Font = New System.Drawing.Font("EucrosiaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.AddValue.Location = New System.Drawing.Point(544, 600)
        Me.AddValue.Name = "AddValue"
        Me.AddValue.TabIndex = 6
        Me.AddValue.Text = "Add"
        '
        'Button3
        '
        Me.Button3.Font = New System.Drawing.Font("Angsana New", 15.75!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Button3.Location = New System.Drawing.Point(840, 624)
        Me.Button3.Name = "Button3"
        Me.Button3.Size = New System.Drawing.Size(104, 32)
        Me.Button3.TabIndex = 25
        Me.Button3.Text = "Connection"
        '
        'Label7
        '
        Me.Label7.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label7.Location = New System.Drawing.Point(400, 352)
        Me.Label7.Name = "Label7"
        Me.Label7.Size = New System.Drawing.Size(96, 24)
        Me.Label7.TabIndex = 26
        Me.Label7.Text = "ข้อมูลน้ำจุบัน"
        '
        'TextBox1
        '
        Me.TextBox1.Location = New System.Drawing.Point(512, 350)
        Me.TextBox1.Name = "TextBox1"
        Me.TextBox1.Size = New System.Drawing.Size(248, 29)
        Me.TextBox1.TabIndex = 27
        Me.TextBox1.Text = ""
        '
        'rxbox
        '
        Me.rxbox.Location = New System.Drawing.Point(512, 384)
        Me.rxbox.Name = "rxbox"
        Me.rxbox.Size = New System.Drawing.Size(160, 29)
        Me.rxbox.TabIndex = 28
        Me.rxbox.Text = ""
        '
        'txbox
        '
        Me.txbox.Location = New System.Drawing.Point(512, 424)
        Me.txbox.Name = "txbox"
```

```
Me.txbox.Size = New System.Drawing.Size(160, 29)
Me.txbox.TabIndex = 29
Me.txbox.Text = ""
'
'voltbox
'
Me.voltbox.Location = New System.Drawing.Point(512, 464)
Me.voltbox.Name = "voltbox"
Me.voltbox.Size = New System.Drawing.Size(160, 29)
Me.voltbox.TabIndex = 30
Me.voltbox.Text = ""
'
'ampbox1
'
Me.ampbox1.Location = New System.Drawing.Point(512, 504)
Me.ampbox1.Name = "ampbox1"
Me.ampbox1.Size = New System.Drawing.Size(160, 29)
Me.ampbox1.TabIndex = 31
Me.ampbox1.Text = ""
'
'ampbox2
'
Me.ampbox2.Location = New System.Drawing.Point(512, 544)
Me.ampbox2.Name = "ampbox2"
Me.ampbox2.Size = New System.Drawing.Size(160, 29)
Me.ampbox2.TabIndex = 32
Me.ampbox2.Text = ""
'
'tempbox1
'
Me.tempbox1.Location = New System.Drawing.Point(512, 584)
Me.tempbox1.Name = "tempbox1"
Me.tempbox1.Size = New System.Drawing.Size(160, 29)
Me.tempbox1.TabIndex = 33
Me.tempbox1.Text = ""
'
'tempbox2
'
Me.tempbox2.Location = New System.Drawing.Point(512, 624)
Me.tempbox2.Name = "tempbox2"
Me.tempbox2.Size = New System.Drawing.Size(160, 29)
Me.tempbox2.TabIndex = 34
Me.tempbox2.Text = ""
'
'Label8
'
```

```
        Me.Label8.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label8.Location = New System.Drawing.Point(400, 386)
        Me.Label8.Name = "Label8"
        Me.Label8.Size = New System.Drawing.Size(96, 24)
        Me.Label8.TabIndex = 35
        Me.Label8.Text = "หมายเลขผู้ส่ง"
        '
        'Label9
        '
        Me.Label9.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label9.Location = New System.Drawing.Point(400, 426)
        Me.Label9.Name = "Label9"
        Me.Label9.Size = New System.Drawing.Size(96, 24)
        Me.Label9.TabIndex = 36
        Me.Label9.Text = "หมายเลขผู้รับ"
        '
        'Label10
        '
        Me.Label10.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label10.Location = New System.Drawing.Point(400, 466)
        Me.Label10.Name = "Label10"
        Me.Label10.Size = New System.Drawing.Size(96, 24)
        Me.Lebel10.TabIndex = 37
        Me.Label10.Text = "แรงดัน1"
        '
        'Label11
        '
        Me.Label11.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label11.Location = New System.Drawing.Point(400, 506)
        Me.Label11.Name = "Label11"
        Me.Label11.Size = New System.Drawing.Size(96, 24)
        Me.Label11.TabIndex = 38
        Me.Label11.Text = "กระแส1"
        '
        'Label12
        '
        Me.Label12.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label12.Location = New System.Drawing.Point(400, 546)
        Me.Label12.Name = "Label12"
        Me.Label12.Size = New System.Drawing.Size(96, 24)
        Me.Label12.TabIndex = 39
        Me.Label12.Text = "กระแส2"
```

```
        '
        'Label13
        '
        Me.Label13.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label13.Location = New System.Drawing.Point(400, 586)
        Me.Label13.Name = "Label13"
        Me.Label13.Size = New System.Drawing.Size(96, 24)
        Me.Label13.TabIndex = 40
        Me.Label13.Text = "คุณหนูมิ1"
        '
        'Label14
        '
        Me.Label14.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label14.Location = New System.Drawing.Point(400, 626)
        Me.Label14.Name = "Label14"
        Me.Label14.Size = New System.Drawing.Size(96, 24)
        Me.Label14.TabIndex = 41
        Me.Label14.Text = "คุณหนูมิ2"
        '
        'Label15
        '
        Me.Label15.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label15.Location = New System.Drawing.Point(696, 466)
        Me.Label15.Name = "Label15"
        Me.Label15.Size = New System.Drawing.Size(96, 24)
        Me.Label15.TabIndex = 42
        Me.Label15.Text = "โจทย์"
        '
        'Label16
        '
        Me.Label16.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label16.Location = New System.Drawing.Point(696, 506)
        Me.Label16.Name = "Label16"
        Me.Label16.Size = New System.Drawing.Size(96, 24)
        Me.Label16.TabIndex = 43
        Me.Label16.Text = "แชมป์"
        '
        'Label17
        '
        Me.Label17.Font = New System.Drawing.Font("AngsanaUPC", 14.25! System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label17.Location = New System.Drawing.Point(696, 546)
        Me.Label17.Name = "Label17"
```

```
        Me.Label17.Size = New System.Drawing.Size(96, 24)
        Me.Label17.TabIndex = 44
        Me.Label17.Text = "แอมป์"
        '
        'Label18
        '
        Me.Label18.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label18.Location = New System.Drawing.Point(696, 584)
        Me.Label18.Name = "Label18"
        Me.Label18.Size = New System.Drawing.Size(96, 24)
        Me.Label18.TabIndex = 45
        Me.Label18.Text = "องศาเซลเซียส"
        '
        'Label19
        '
        Me.Label19.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label19.Location = New System.Drawing.Point(696, 624)
        Me.Label19.Name = "Label19"
        Me.Label19.Size = New System.Drawing.Size(96, 24)
        Me.Label19.TabIndex = 46
        Me.Label19.Text = "องศาเซลเซียส"
        '
        'Label1
        '
        Me.Label1.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label1.Location = New System.Drawing.Point(32, 64)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(72, 24)
        Me.Label1.TabIndex = 48
        Me.Label1.Text = "แรงดัน"
        '
        'Label2
        '
        Me.Label2.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label2.Location = New System.Drawing.Point(32, 192)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(72, 24)
        Me.Label2.TabIndex = 50
        Me.Label2.Text = "กระแส1"
        '
        'Label3
        '
```

```
        Me.Label3.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label3.Location = New System.Drawing.Point(24, 312)
        Me.Label3.Name = "Label3"
        Me.Label3.Size = New System.Drawing.Size(72, 24)
        Me.Label3.TabIndex = 52
        Me.Label3.Text = "กระแส2"
        '
        'Label4
        '
        Me.Label4.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label4.Location = New System.Drawing.Point(24, 424)
        Me.Label4.Name = "Label4"
        Me.Label4.Size = New System.Drawing.Size(72, 24)
        Me.Label4.TabIndex = 54
        Me.Label4.Text = "อุณหภูมิ1"
        '
        'Label5
        '
        Me.Label5.Font = New System.Drawing.Font("AngsanaUPC", 14.25!, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Label5.Location = New System.Drawing.Point(24, 552)
        Me.Label5.Name = "Label5"
        Me.Label5.Size = New System.Drawing.Size(72, 24)
        Me.Label5.TabIndex = 56
        Me.Label5.Text = "อุณหภูมิ2"
        '
        'OleDbDataAdapter1
        '
        Me.OleDbDataAdapter1.DeleteCommand = Me.OleDbDeleteCommand1
        Me.OleDbDataAdapter1.InsertCommand = Me.OleDbInsertCommand1
        Me.OleDbDataAdapter1.SelectCommand = Me.OleDbSelectCommand1
        Me.OleDbDataAdapter1.TableMappings.AddRange(New System.Data.Common.DataTableMapping() {New
System.Data.Common.DataTableMapping("Table", "Table1", New System.Data.Common.DataColumnMapping() {New
System.Data.Common.DataColumnMapping("Amp1", "Amp1"), New System.Data.Common.DataColumnMapping("Amp2", "Amp2"), New
System.Data.Common.DataColumnMapping("Index", "Index"), New System.Data.Common.DataColumnMapping("rx", "rx"), New
System.Data.Common.DataColumnMapping("Temp1", "Temp1"), New System.Data.Common.DataColumnMapping("Temp2", "Temp2"), New
System.Data.Common.DataColumnMapping("tx", "tx"), New System.Data.Common.DataColumnMapping("Volt", "Volt")})})
        Me.OleDbDataAdapter1.UpdateCommand = Me.OleDbUpdateCommand1
        '
        'OleDbDeleteCommand1
        '
        Me.OleDbDeleteCommand1.CommandText = "DELETE FROM Table1 WHERE ([Index] = ?) AND (Amp1 = ? OR ? IS NULL AND Amp1
IS NUL" & _
        "L) AND (Amp2 = ? OR ? IS NULL AND Amp2 IS NULL) AND (Temp1 = ? OR ? IS NULL AND " & _
        "Temp1 IS NULL) AND (Temp2 = ? OR ? IS NULL AND Temp2 IS NULL) AND (Volt = ? OR ?" & _
```

```
        " IS NULL AND Volt IS NULL) AND (rx = ? OR ? IS NULL AND rx IS NULL) AND (tx = ? " & _
        "OR ? IS NULL AND tx IS NULL)"
        Me.OleDbDeleteCommand1.Connection = Me.OleDbConnection1
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Index",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Index",
System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Amp1",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Amp1",
System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Amp11",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Amp1", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Amp2",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Amp2",
System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Amp21",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Amp2",
System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Temp1",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Temp1",
System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Temp11",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Temp1",
System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Temp2",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Temp2",
System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Temp21",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Temp2",
System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Volt",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Volt",
System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Volt1",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Volt",
System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_rx", System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "rx", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_rx1",
System.Data.OleDb.OleDbType.VarWChar, 50, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "rx", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbDeleteCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_tx", System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "tx", System.Data.DataRowVersion.Original, Nothing))
```

```
        Me.OleDbDeleteCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_tx1",
System.Data.OleDb.OleDbType.VarWChar, 50, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "tx", System.Data.DataRowVersion.Original, Nothing))
        '
        'OleDbConnection1
        '
        Me.OleDbConnection1.ConnectionString = "Jet OLEDB:Global Partial Bulk
Ops=2;Jet OLEDB:Registry Path=;Jet OLEDB:Database L" & _
        "ocking Mode=1;Data Source=""D:\project\project48-04-02.mdb"";Jet OLEDB:Engine Type" & _
        "=5;Provider=""Microsoft.Jet.OLEDB.4.0"";Jet OLEDB:System database=;Jet OLEDB:SFP=F" & _
        "alse;persist security info=False;Extended Properties=;Mode=Share Deny None;Jet O" & _
        "LEDB:Encrypt Database=False;Jet OLEDB:Create System Database=False;Jet OLEDB:Don" & _
        "'t Copy Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;U" & _
        "ser ID=Admin;Jet OLEDB:Global Bulk Transactions=1"
        '
        'OleDbInsertCommand1
        '
        Me.OleDbInsertCommand1.CommandText = "INSERT INTO Table1(Amp1, Amp2,
rx, Temp1  Temp2, tx, Volt) VALUES (?, ?, ?, ?, ?," & _
        " ?, ?)"
        Me.OleDbInsertCommand1.Connection = Me.OleDbConnection1
        Me.OleDbInsertCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Amp1", System.Data.OleDb.OleDbType.Integer, 0, "Amp1"))
        Me.OleDbInsertCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Amp2",
System.Data.OleDb.OleDbType.Integer, 0, "Amp2"))
        Me.OleDbInsertCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("rx", System.Data.OleDb.OleDbType.VarWChar, 50, "rx"))
        Me.OleDbInsertCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Temp1", System.Data.OleDb.OleDbType.Integer, 0, "Temp1"))
        Me.OleDbInsertCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Temp2", System.Data.OleDb.OleDbType.Integer, 0, "Temp2"))
        Me.OleDbInsertCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("tx", System.Data.OleDb.OleDbType.VarWChar, 50, "tx"))
        Me.OleDbInsertCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Volt", System.Data.OleDb.OleDbType.Integer, 0, "Volt"))
        '
        'OleDbSelectCommand1
        '
        Me.OleDbSelectCommand1.CommandText = "SELECT Amp1, Amp2, [Index], rx, Temp1, Temp2, tx, Volt FROM Table1"
        Me.OleDbSelectCommand1.Connection = Me.OleDbConnection1
        '
        'OleDbUpdateCommand1
        '
        Me.OleDbUpdateCommand1.CommandText = "UPDATE Table1 SET Amp1 = ?,
Amp2 = ?, rx = ?, Temp1 = ?, Temp2 = ?, tx = ?, Volt " & _
        "= ? WHERE ([Index] = ?) AND (Amp1 = ? OR ? IS NULL AND Amp1 IS NULL) AND (Amp2 =" & _
        " ? OR ? IS NULL AND Amp2 IS NULL) AND (Temp1 = ? OR ? IS NULL AND Temp1 IS NULL)" & _
```

```
        " AND (Temp2 = ? OR ? IS NULL AND Temp2 IS NULL) AND (volt =? OR ? IS NULL AND V" & _
        "olt IS NULL) AND (rx =? OR ? IS NULL AND rx IS NULL) AND (tx =? OR ? IS NULL A" & _
        "ND tx IS NULL)"
        Me.OleDbUpdateCommand1.Connection = Me.OleDbConnection1
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Amp1", System.Data.OleDb.OleDbType.Integer, 0, "Amp1"))
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Amp2", System.Data.OleDb.OleDbType.Integer, 0, "Amp2"))
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("rx", System.Data.OleDb.OleDbType.VarWChar, 50, "rx"))
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Temp1", System.Data.OleDb.OleDbType.Integer, 0, "Temp1"))
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Temp2", System.Data.OleDb.OleDbType.Integer, 0, "Temp2"))
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("tx", System.Data.OleDb.OleDbType.VarWChar, 50, "tx"))
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Volt", System.Data.OleDb.OleDbType.Integer, 0, "Volt"))
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_Index", System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Index", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Amp1",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "Amp1", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Amp11",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "Amp1", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Amp2",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "Amp2", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Amp21",
System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Amp2", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Temp1",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "Temp1", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Temp11",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "Temp1", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Temp2",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "Temp2", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_Temp21",
System.Data.OleDb.OleDbType.Integer, 0, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "Temp2", System.Data.DataRowVersion.Original, Nothing))
```

```
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_Volt", System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Volt", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_Volt1", System.Data.OleDb.OleDbType.Integer, 0,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "Volt", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_rx", System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "rx", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_rx1",
System.Data.OleDb.OleDbType.VarWChar, 50, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "rx", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New
System.Data.OleDb.OleDbParameter("Original_tx", System.Data.OleDb.OleDbType.VarWChar, 50,
System.Data.ParameterDirection.Input, False, CType(0, Byte), CType(0, Byte), "tx", System.Data.DataRowVersion.Original, Nothing))
        Me.OleDbUpdateCommand1.Parameters.Add(New System.Data.OleDb.OleDbParameter("Original_tx1",
System.Data.OleDb.OleDbType.VarWChar, 50, System.Data.ParameterDirection.Input, False,
CType(0, Byte), CType(0, Byte), "tx", System.Data.DataRowVersion.Original, Nothing))
        '
        'DataSet11
        '
        Me.DataSet11.DataSetName = "DataSet1"
        Me.DataSet11.Locale = New System.Globalization.CultureInfo("en-US")
        '
        'OleDbCommand1
        '
        Me.OleDbCommand1.CommandText = "SELECT Amp1, Amp2, [Index], rx, Temp1, Temp2, tx, Volt FROM Table1"
        Me.OleDbCommand1.Connection = Me.OleDbConnection1
        '
        'SerialConnection2
        '
        Me.SerialConnection2.Break = False
        Me.SerialConnection2.Encoding = CType(resources.GetObject("SerialConnection2.Encoding"), System.Text.Encoding)
        Me.SerialConnection2.Options = New Sax.Communications.SerialOptions("COM1", 115200,
Sax.Communications.Parity.None, 8, Sax.Communications.CommStopBits.One, False, False, True, False, True, True)
        Me.SerialConnection2.TransferPath = Nothing
        Me.SerialConnection2.TransferProtocol = Sax.Communications.TransferProtocol.XmodemCrc
        '
        'voltchart
        '
        Me.voltchart.DataSource = Nothing
        Me.voltchart.Enabled = True
        Me.voltchart.Location = New System.Drawing.Point(128, 24)
        Me.voltchart.Name = "voltchart"
        Me.voltchart.OcxState = CType(resources.GetObject("voltchart.OcxState"), System.Windows.Forms.AxHost.State)
        Me.voltchart.Size = New System.Drawing.Size(208, 104)
        Me.voltchart.TabIndex = 57
```

```
'
'ampchart2
'
Me.ampchart2.DataSource = Nothing
Me.ampchart2.Enabled = True
Me.ampchart2.Location = New System.Drawing.Point(128, 280)
Me.ampchart2.Name = "ampchart2"
Me.ampchart2.OcxState = CType(resources.GetObject("ampchart2.OcxState"), System.Windows.Forms.AxHost.State)
Me.ampchart2.Size = New System.Drawing.Size(208, 104)
Me.ampchart2.TabIndex = 59
'
'tempchart1
'
Me.tempchart1.DataSource = Nothing
Me.tempchart1.Enabled = True
Me.tempchart1.Location = New System.Drawing.Point(128, 408)
Me.tempchart1.Name = "tempchart1"
Me.tempchart1.OcxState = CType(resources.GetObject("tempchart1.OcxState"), System.Windows.Forms.AxHost.State)
Me.tempchart1.Size = New System.Drawing.Size(208, 104)
Me.tempchart1.TabIndex = 60
'
'tempchart2
'
Me.tempchart2.DataSource = Nothing
Me.tempchart2.Enabled = True
Me.tempchart2.Location = New System.Drawing.Point(128, 536)
Me.tempchart2.Name = "tempchart2"
Me.tempchart2.OcxState = CType(resources.GetObject("tempchart2.OcxState"), System.Windows.Forms.AxHost.State)
Me.tempchart2.Size = New System.Drawing.Size(208, 104)
Me.tempchart2.TabIndex = 61
'
'ampchart1
'
Me.ampchart1.DataSource = Nothing
Me.ampchart1.Enabled = True
Me.ampchart1.Location = New System.Drawing.Point(128, 152)
Me.ampchart1.Name = "ampchart1"
Me.ampchart1.OcxState = CType(resources.GetObject("ampchart1.OcxState"), System.Windows.Forms.AxHost.State)
Me.ampchart1.Size = New System.Drawing.Size(208, 104)
Me.ampchart1.TabIndex = 62
'
'Form1
'
Me.AutoScaleBaseSize = New System.Drawing.Size(9, 22)
Me.BackColor = System.Drawing.SystemColors.InactiveCaptionText
Me.ClientSize = New System.Drawing.Size(1028, 746)
Me.Controls.Add(Me.ampchart1)
```

```
        Me.Controls.Add(Me.tempchart2)
        Me.Controls.Add(Me.tempchart1)
        Me.Controls.Add(Me.ampchart2)
        Me.Controls.Add(Me.voltchart)
        Me.Controls.Add(Me.Label5)
        Me.Controls.Add(Me.Label4)
        Me.Controls.Add(Me.Label3)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.Label19)
        Me.Controls.Add(Me.Label18)
        Me.Controls.Add(Me.Label17)
        Me.Controls.Add(Me.Label16)
        Me.Controls.Add(Me.Label15)
        Me.Controls.Add(Me.Label14)
        Me.Controls.Add(Me.Label13)
        Me.Controls.Add(Me.Label12)
        Me.Controls.Add(Me.Label11)
        Me.Controls.Add(Me.Label10)
        Me.Controls.Add(Me.Label9)
        Me.Controls.Add(Me.Label8)
        Me.Controls.Add(Me.tempbox2)
        Me.Controls.Add(Me.tempbox1)
        Me.Controls.Add(Me.ampbox2)
        Me.Controls.Add(Me.ampbox1)
        Me.Controls.Add(Me.voltbox)
        Me.Controls.Add(Me.txbox)
        Me.Controls.Add(Me.rxbox)
        Me.Controls.Add(Me.TextBox1)
        Me.Controls.Add(Me.Label7)
        Me.Controls.Add(Me.Button3)
        Me.Controls.Add(Me.AddValue)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.DataGrid1)
        Me.Font = New System.Drawing.Font("Microsoft Sans Serif", 14.25!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(222, Byte))
        Me.Name = "Form1"
        Me.Text = "ข้อมูลการใช้พลังงาน"
        CType(Me.DataGrid1, System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.DataSet61, System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.DataSet11, System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.voltchart, System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.ampchart2, System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.tempchart1, System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.tempchart2, System.ComponentModel.ISupportInitialize).EndInit()
        CType(Me.ampchart1, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)
```

```
    End Sub

#End Region




โปรแกรมควบคุมหน้าจอหลัก

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        OleDbDataAdapter1.Fill(DataSet61, "table1")

End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Me.Close()
End Sub

Private Sub AddValue_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles AddValue.Click

        Dim sqlAdd As String = ""       'ตัวแปรเก็บข้อมูลชุดคำสั่ง

        If (rxbox.Text = "") Or (txbox.Text = "") Or (voltbox.Text = "") Or (ampbox1.Text =
"") Or (ampbox2.Text = "") Or (tempbox1.Text = "") Or (tempbox2.Text = "") Then
            MessageBox.Show("กรุณาป้อนข้อมูลให้ครบ", "ป้อนข้อมูลไม่ครบ", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End If
        Try
            sqlAdd = "Insert into table1(rx,tx,Volt,Amp1,Amp2,Temp1,Temp2)"
            sqlAdd &= " values('" & rxbox.Text & "' , '" & txbox.Text & "' , '" &
voltbox.Text & "' , '" & ampbox1.Text & "' , '" & ampbox2.Text & "' , '" & tempbox1.Text & "' , '" & tempbox2.Text & "')"

            With Conn
                If State = ConnectionState.Open Then Close()
                .ConnectionString = Me.OleDbConnection1.ConnectionString
                Open()
            End With

            Olecmd = New OleDbCommand
            With olecmd
                .CommandType = CommandType.Text
                .CommandText = sqlAdd
                .Connection = Conn
                .ExecuteNonQuery()
```

```
        End With

    Catch errortoAdd As Exception
        MessageBox.Show("cannot add data , data entry error.")

    Finally
        Conn.Close()
    End Try


    Dim strConn As String = "Provider=Microsoft.JET.OLEDB.4.0;data source=D:\\project\Serial.mdb"

    If (rxbox.Text = "") Or (txbox.Text = "") Or (voltbox.Text = "") Or (ampbox1.Text =
"") Or (ampbox2.Text = "") Or (tempbox1.Text = "") Or (tempbox2.Text = "") Then
    End If

    sqlAdd = "INSERT INTO Serial1(rx,tx,Volt,Amp1,Amp2,Temp1,Temp2)"
    sqlAdd &= " values('" & rxbox.Text & "' , '" & txbox.Text& "' , '" & voltbox.Text
& "', '" & ampbox1.Text & "' , '" & ampbox2.Text & "' , '" & tempbox1.Text & "' , '" & tempbox2.Text & "')"


    Try
        With Conn
            If State = ConnectionState.Open Then Close()
            .ConnectionString = strConn
            Open()
        End With

        Olecmd = New OleDbCommand
        With olecmd
            CommandType = CommandType.Text
            .CommandText = sqlAdd
            .Connection = Conn
            .ExecuteNonQuery()
        End With

    Catch errortoAdd As Exception
        MessageBox.Show("cannot add data , data entry error.")

    Finally
        Conn.Close()
    End Try

    dataAdap = New OleDbDataAdapter(sqlAdd, Conn)
    dataAdap.Fill(DataSet11, "Serial1")'ใส่ข้อมูลใน dataset ชื่อว่า Serial
```

```
        OleDbDataAdapter1.Fill(DataSet61, "table1")
        voltchart.CtlRefresh()
        ampchart1.CtlRefresh()
        ampchart2.CtlRefresh()
        tempchart1.CtlRefresh()
        tempchart2.CtlRefresh()
End Sub

Private Sub AddValue_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        voltchart.Refresh()
        ampchart1.Refresh()
        ampchart2.Refresh()
        tempchart1.Refresh()
        tempchart2.Refresh()

        string1 = TextBox1.Text

        rxbox.Text = string1.Substring(3, 1) ' rx
        txbox.Text = string1.Substring(2, 1) ' tx
        voltbox.Text = string1.Substring(5, 3) 'volt
        ampbox1.Text = string1.Substring(8, 2) 'amp1
        ampbox2.Text = string1.Substring(10, 2) 'amp2
        tempbox1.Text = string1.Substring(12, 2) 'temp1
        tempbox2.Text = string1.Substring(14, 2) 'temp2
End Sub

Private Sub loadfrom3_road(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
        portsetting.Show()
End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
        portsetting.ShowDialog()
        Dim commIndex As Integer
        commPortIndex = portsetting.Comboport.SelectedIndex
        commSpeedIndex = portsetting.Combospeed.SelectedIndex
        commModeIndex = portsetting.Combomode.SelectedIndex
        commFlowIndex = portsetting.Combocontrol.SelectedIndex()
        If SerialConnection2.IsOpen Then
          SerialConnection2.Close()
        Else
          Try
            Dim theParity As Parity = Parity.Even
            If commModeIndex = 0 Then theParity = Parity.None

            Dim theDataBits As Integer = 7
            If commModeIndex = 0 Then theDataBits = 8
```

```
            SerialConnection2.Options = New SerialOptions(portsetting.Comboport.Text, _
Integer.Parse(portsetting.Combospeed.Text), _
theParity, _
theDataBits, _
CommStopBits.One, _
portsetting.Combocontrol.SelectedIndex = 3, _
portsetting.Combocontrol.SelectedIndex = 2, _
portsetting.Combocontrol.SelectedIndex = 1, _
False, _
True, _
True)
            SerialConnection2.Open()

        Catch exception As Exception
            MessageBox.Show(Me, String.Format("Device Chat is unable to open the
port:" + ControlChars.Cr + ControlChars.Lf + "{0}", exception.Message), "Device Chat", MessageBoxButtons.OK, MessageBoxIcon.Stop)

        End Try
    End If
End Sub

Private Sub serialConnection2_DataAvailable(ByVal sender As Object, ByVal e As EventArgs) Handles SerialConnection2.DataAvailable
    ' Time to pause after reading in order to allow input buffer to fill up more
    ' Dim mseconds As Integer = Convert.ToInt32(DelayTime.Text)

    'If mseconds > 0 Then
    'Thread.Sleep(mseconds)
    'End If
    Dim nBytes As Integer = SerialConnection2.Available

    ' Notification line saying we have new data
    'listConversation.BeginUpdate()
    'listConversation.Items.Add(String.Format("{0} bytes of new data has arrived on {1}", _
    '        nBytes, CType(sender, SerialConnection).Options.PortName))
    'listConversation.SelectedIndex = listConversation.Items.Count - 1
    'listConversation.EndUpdate()

    ' Now read the det and report it
    If nBytes > 0 Then
        Dim buffer(nBytes - 1) As Byte
        SerialConnection2.Read(buffer, 0, buffer.Length)

        Dim tmpstr, tmpstr1 As String
        Tmpstr = Utilities.CEncode(buffer)
        'If (tmpstr = "r") Then
        'tmpstr = ""
        'Else
```

```
            tmpstr1 = tmpstr
            'End If
            'TextBox2.Text = tmpstr1
            TextBox1.Text.Insert(0, tmpstr1)

            If TextBox1.Text.Length = 0 Then
               Return ' Nothing to do
            End If
            Try
               Dim ab As Byte() = Utilities.CDecode(TextBox1.Text)
               'Say(System.Environment.UserName, TextBox1.Text)
               SerialConnection2.Write(ab, 0, ab.Length)
            Catch
               MessageBox.Show(Me, String.Format("Error Cannot send"))
            End Try
            TextBox1.Text = ""
            TextBox1.Focus()
            '  Say(serialConnection.Options.PortName, Utilities.CEncode(buffer))
         End If
End Sub 'serialConnection_DataAvailable

   End Class
End Namespace
```

# ภาคผนวก ช
# คู่มือการใช้งาน

# คู่มือการใช้งาน
# ระบบบันทึกและควบคุมการใช้พลังงานของ
# ระบบปรับอากาศ


ภาควิชาครุศาสตร์วิศวกรรม
คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

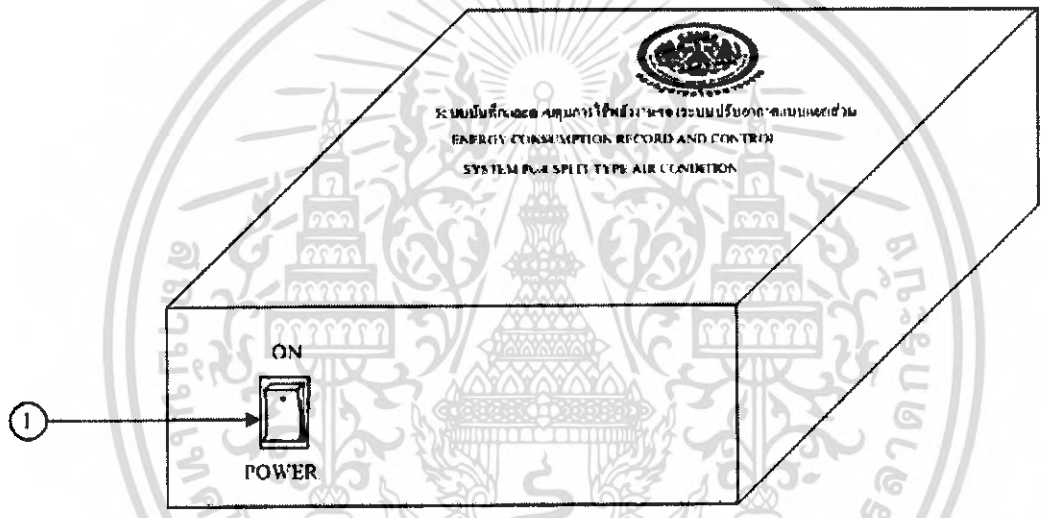
## 1. คำแนะนำเบื้องต้น

ก่อนที่จะลงมือใช้งานระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ ควรทำการศึกษาคู่มือการใช้งานให้เข้าใจ เพื่อที่จะใช้งานได้อย่างถูกต้อง และป้องกันการเสียหายที่จะเกิดขึ้นกับระบบบันทึกและควบคุมการใช้งานของระบบปรับอากาศ

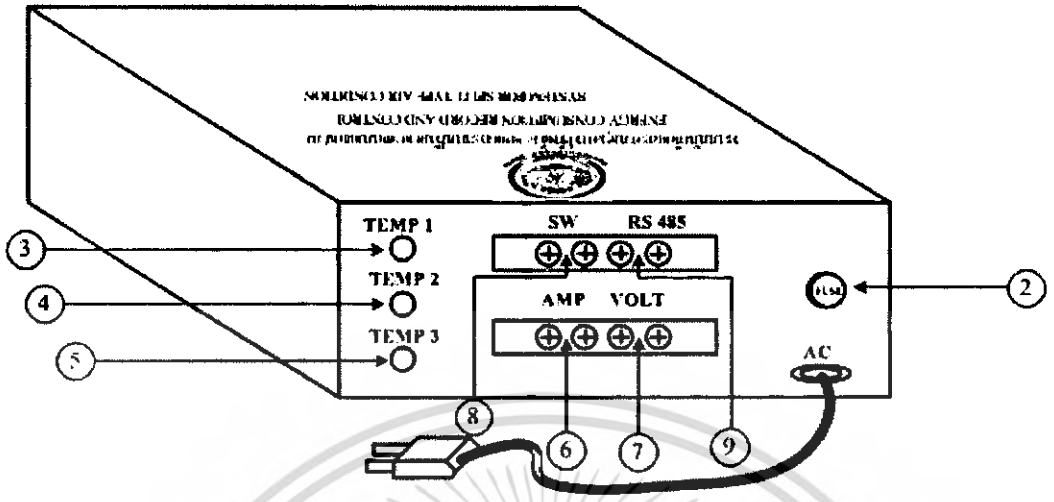
## 2. ส่วนประกอบและปุ่มควบคุม

ระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศประกอบด้วย 3 ส่วน ดังนี้

**2.1 ส่วนของชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ**




รูปที่ ช.1 ส่วนประกอบและปุ่มควบคุมด้านหน้าของชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ (ด้านล่างและด้านหน้า)

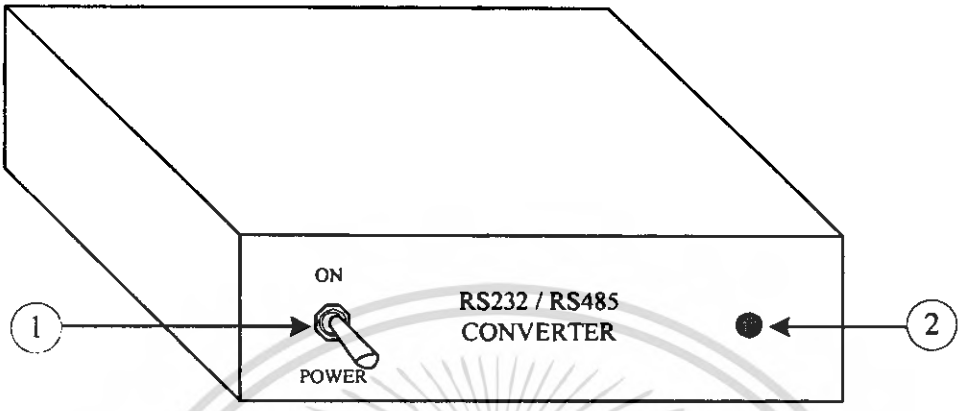



รูปที่ ช.2 ส่วนประกอบและปุ่มควบคุมด้านหลังของชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศ (ด้านหน้าและด้านบน)

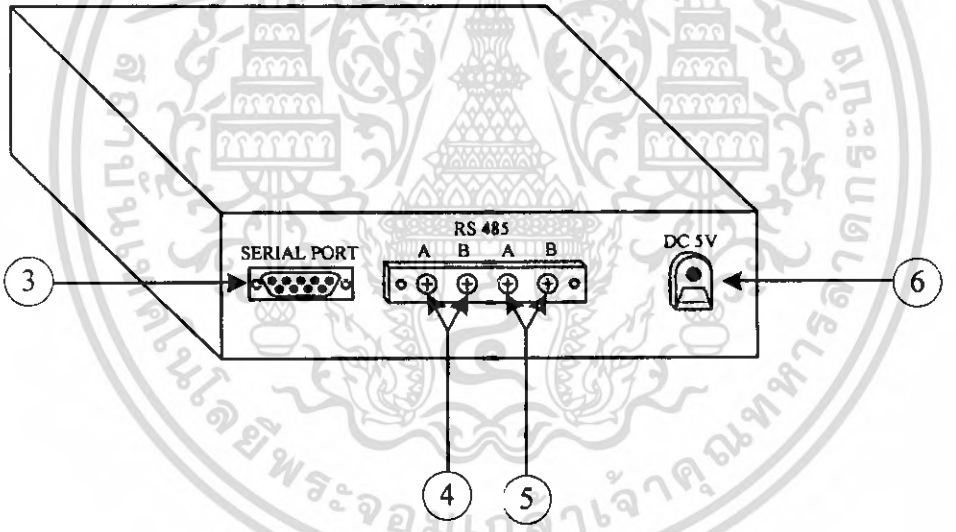
จากรูปที่ ช.1 และ ช.2 มีรายละเอียดต่างๆ ดังนี้

1. สวิตช์เปิด-ปิดเครื่อง
2. ฟิวส์
3. ตัวตรวจจับอุณหภูมิจุดที่ 1
4. ตัวตรวจจับอุณหภูมิจุดที่ 2
5. ตัวตรวจจับอุณหภูมิจุดที่ 3
6. กระแสไฟฟ้า
7. แรงดันไฟฟ้า
8. สวิตช์
9. RS 485

**2.2 ส่วนของชุดแปลงสัญญาณ RS232/RS485**




**รูปที่ ช.3** ส่วนประกอบและปุ่มควบคุมด้านหน้าชุดแปลงสัญญาณ RS232/RS485 (ด้านล่างและด้านหน้า)




**รูปที่ ช.4** ส่วนประกอบและปุ่มควบคุมด้านหลังส่วนของชุดแปลงสัญญาณ RS232/RS485 (ด้านบนและด้านหน้า)

จากรูปที่ ช.3 และ ช.4 มีรายละเอียดต่างๆ ดังนี้

(1) สวิตช์เปิด-ปิดเครื่อง

(2) หลอดแอลอีดีแสดงสถานะการทำงาน

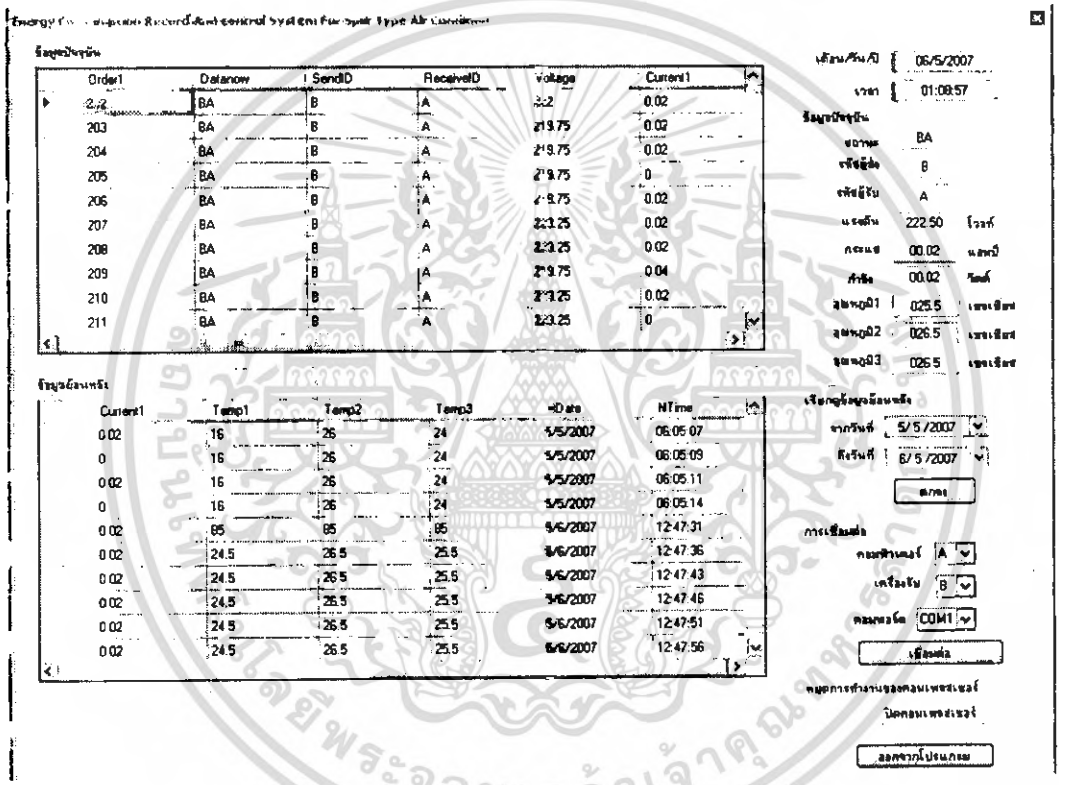
③ จุดต่อสายไปยังเครื่องคอมพิวเตอร์

④ จุดต่อสาย

⑤ จุดต่อสาย

⑥ แหล่งจ่ายไฟฟ้ากระแสตรง 5 โวลต์

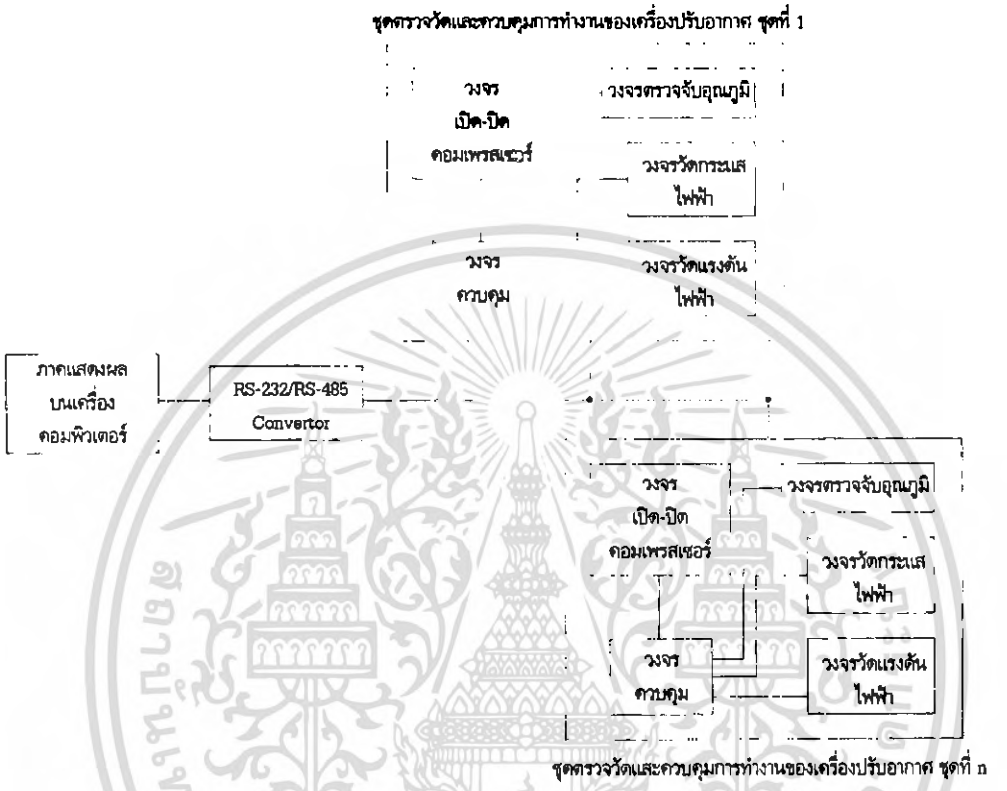
### 2.3 ส่วนแสดงผลบนเครื่องคอมพิวเตอร์




**รูปที่ ข.5** ส่วนประกอบของส่วนแสดงผลบนเครื่องคอมพิวเตอร์

## 3 การติดตั้งและการใช้งาน

3.1 เชื่อมต่อระบบดังรูปที่ จ.6




**รูปที่ ช.6** การเชื่อมต่อระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ

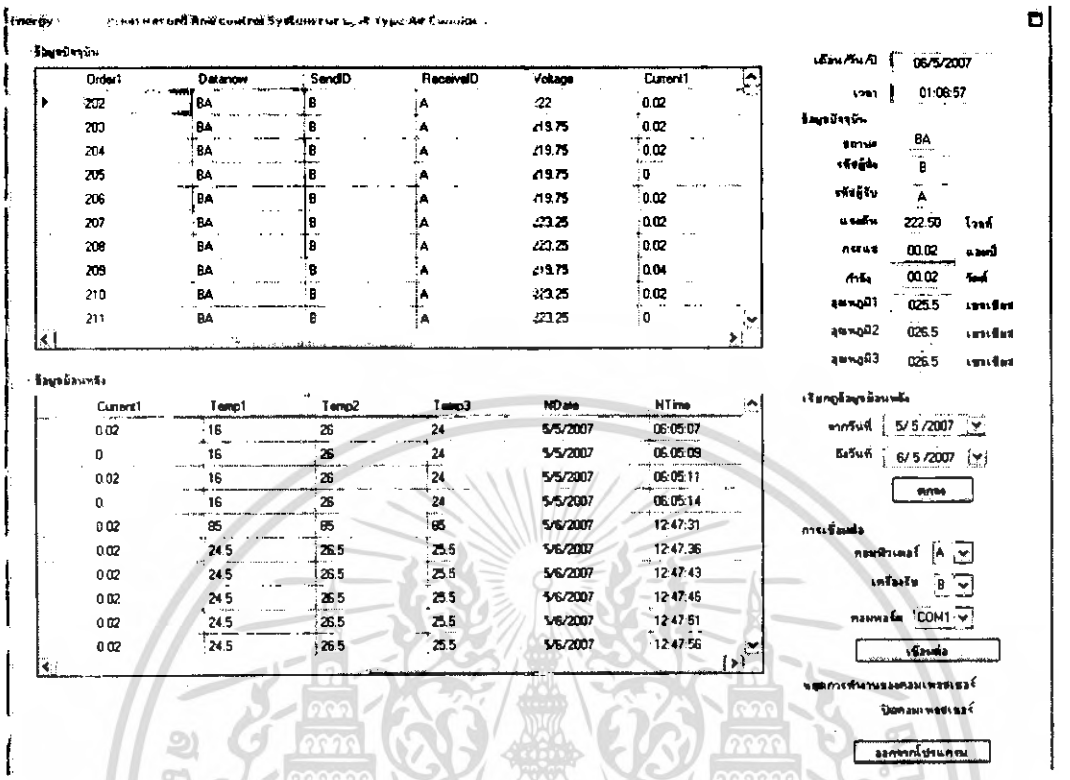
3.2 เชื่อมต่อกับส่วนควบคุมของระบบปรับอากาศ

3.3 ติดตั้งจุดวัดอุณหภูมิภายในห้อง

3.4 ทำการเปิดสวิตช์ของชุดตรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศและชุด RS232/RS485 Converter

3.5 เปิดโปรแกรมของระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ

3.6 ค่าที่ตรวจวัดได้จะแสดงออกมาทั้งตัวเลขและกราฟ




**รูปที่ ช.7** ค่าต่างๆ ที่วัดได้แสดงบนเครื่องคอมพิวเตอร์

## 4.การแก้ปัญหาเบื้องต้น

เมื่อท่านประสบปัญหาในการใช้งานระบบบันทึกและควบคุมการใช้พลังงานของระบบปรับอากาศ สามารถตรวจสอบแนวทางแก้ไขปัญหาเบื้องต้นได้จากตารางต่อไปนี้

| อาการ | สาเหตุและ/หรือวิธีแก้ไข |
|---|---|
| ค่าต่างๆ ไม่แสดงที่หน้าจอคอมพิวเตอร์ | ตรวจดูจุดเชื่อมต่อต่างๆ , ตรวจดูการตั้งค่าพอร์ตของคอมพิวเตอร์ว่าถูกต้องหรือไม่ |
| ไม่มีการส่งสัญญาณจากชุดRS232/RS485 Converter | ตรวจเช็คไฟเลี้ยง 5 โวลต์ว่ามีการจ่ายไฟหรือไม่ |

## 5. การดูแลรักษาและข้อควรระวัง

### 5.1 การดูแลรักษา

1. ควรปิดสวิตช์และดึงปลั๊กออกทุกครั้งเมื่อไม่ใช้งาน
2. ตรวจเช็คและทำความสะอาดชุดรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศและชุด RS232/RS485 Converter ให้อยู่ในสภาพพร้อมใช้งานอยู่เสมอ

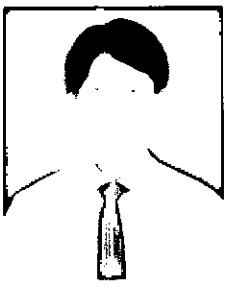
### 5.2 ข้อควรระวัง

ชุดรวจวัดและควบคุมการทำงานของเครื่องปรับอากาศและชุด RS232/RS485 Converter ก่อนที่จะถอดปลั๊กออกควรปิดสวิตช์ก่อนทุกครั้ง

## 6. ข้อมูลจำเพาะ

| คุณสมบัติ | รายละเอียด |
|---|---|
| ไอซีตรวจวัดอุณหภูมิ | เบอร์ DS 1820 |
| การแสดงผลบนเครื่องคอมพิวเตอร์ | ตัวเลขและกราฟ |
| แหล่งจ่ายพลังงาน | ไฟฟ้ากระแสสลับ 220 โวลต์ ความถี่ 50 เฮิรตซ์ |

# ประวัติผู้แต่ง

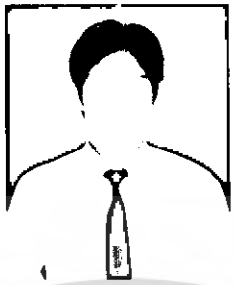


| | |
|---|---|
| **ชื่อ-สกุล** | นายนพดล โปธิ |
| **วัน เดือน ปีเกิด** | 23 เมษายน พ.ศ. 2528 |
| **ภูมิลำเนา** | 26/2 หมู่.10 ตำบลอินทขิล อำเภอแม่แตง |
| | จังหวัดเชียงใหม่ 50150 |
| **ประวัติการศึกษา** | |
| ประถมศึกษา | โรงเรียนวัดหนองออน |
| มัธยมศึกษาตอนต้น | โรงเรียนแม่แตง |
| ประกาศนียบัตรวิชาชีพ | วิทยาลัยเทคนิคเชียงใหม่ |
| ประกาศนียบัตรวิชาชีพชั้นสูง | วิทยาลัยเทคนิคเชียงใหม่ |
| **ปริญญาตรี** | สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์ |
| | ภาควิชาครุศาสตร์วิศวกรรม |
| | คณะครุศาสตร์อุตสาหกรรม สจล. |
| **คติพจน์** | ความพยายามอยู่ที่ไหนความสำเร็จอยู่ที่นั้น |

# ประวัติผู้แต่ง

| | |
|---|---|
| **ชื่อ-สกุล** | นายปรีดา พรมบุตร |
| **วัน เดือน ปีเกิด** | 24 กันยายน พ.ศ. 2527 |
| **ภูมิลำเนา** | 37 หมู่.7 ตำบลยุหว่า อำเภอสันป่าตอง จังหวัดเชียงใหม่ 50120 |
| **ประวัติการศึกษา** | |
| ประถมศึกษา | โรงเรียนวัดกู่คำ |
| มัธยมศึกษาตอนต้น | โรงเรียนสันป่าตองวิทยาคม |
| ประกาศนียบัตรวิชาชีพ | วิทยาลัยเทคนิคเชียงใหม่ |
| ประกาศนียบัตรวิชาชีพชั้นสูง | วิทยาลัยเทคนิคเชียงใหม่ |
| **ปริญญาตรี** | สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์<br>ภาควิชาครุศาสตร์วิศวกรรม<br>คณะครุศาสตร์อุตสาหกรรม สจล. |
| **คติพจน์** | ทำวันนี้ให้ดีที่สุด |

# ประวัติผู้แต่ง



**ชื่อ-สกุล** นายวิทยายุทธ ผะดาวงศ์

**วัน เดือน ปีเกิด** 16 พฤจิกายน พ.ศ. 2527

**ภูมิลำเนา** 119 หมู่ 3 ตำบลไร่สีสุก อำเภอเสนางคนิคม
จังหวัดอำนาจเจริญ 37290

**ประวัติการศึกษา**

ประถมศึกษา โรงเรียนชุมชนบ้านไร่สีสุก จังหวัดอำนาจเจริญ

มัธยมศึกษาตอนต้น โรงเรียนชุมชนบ้านไร่สีสุก จังหวัดอำนาจเจริญ

ประกาศนียบัตรวิชาชีพ วิทยาลัยเทคนิคอำนาจเจริญ จังหวัดอำนาจเจริญ

ประกาศนียบัตรวิชาชีพชั้นสูง วิทยาลัยเทคนิคอำนาจเจริญ จังหวัดอำนาจเจริญ

**ปริญญาตรี** สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
ภาควิชาครุศาสตร์วิศวกรรม
คณะครุศาสตร์อุตสาหกรรม สจล.

**คติพจน์** อดีตผ่านมาแล้ว อนาคตยังมาไม่ถึง อยู่กับปัจจุบันแล้วทำให้ดีที่สุด