สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การสร้างโปรแกรมควบคุมโมดูลมาตรฐาน สำหรับ MCS - 51

BUILDING STANDARD CONTROLLER MODULE PROGRAM FOR MCS-51

นายรุสดี แวดือรามัน
นายสุชาติ เพชรประสมกูล

เลขหมู่.....
เลขทะเบียน.....50435
วัน,เดือน,ปี.....13 พ.ค. 2547

.b.....
.i.....

ปริญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2545

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

BUILDING STANDARD CONTROLLER MODULE PROGRAM FOR MCS-51

RUSDEE WAEDUERAMUN

SUCHAT PETPRASOMGUL

A THESIS SUBMITTED IN PATIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2002

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์ การสร้างโปรแกรมควบคุมโมดูลมาตรฐาน สำหรับ MCS - 51

BUILDING STANDARD CONTROLLER MODULE PROGRAM FOR MCS-51

นักศึกษาผู้จัดทำ นายรุสดี แวดือรามัน รหัสประจำตัว 43015533

นายสุชาติ เพชรประสมกูล รหัสประจำตัว 43015541

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2545

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รศ.พิพัฒน์ เลาหสงคราม	

วัน/เดือน/ปี ที่สอบ วันที่ 25 เดือนมีนาคม พ.ศ. 2546

สถานที่สอบ ณ ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

ภาควิชารับรองแล้ว

(ผศ.ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปริญญานิพนธ์	การสร้างโปรแกรมควบคุมโมดูลมาตรฐาน สำหรับ MCS-51 BUILDING STANDARD CONTROLLER MODULE PROGRAM FOR MCS-51
นักศึกษาผู้จัดทำ	นายรุสดี แวดือรามัน นายสุชาติ เพชรประสมกูล
อาจารย์ที่ปรึกษา	รศ.พิพัฒน์ เลาหสงคราม
ปีการศึกษา	2545

บทคัดย่อ

ไมโครคอนโทรลเลอร์เป็นอุปกรณ์ที่นิยมใช้กันมากในงานควบคุมลักษณะต่างๆ และในการเขียนโปรแกรมควบคุมนั้นทุกส่วนของโปรแกรมมักจะรวมกันอยู่ใน 1 ไฟล์ 1 โปรแกรมซึ่งถ้าโปรแกรมมีขนาดเล็กก็จะไม่ยุ่งยากมากต่อการตรวจสอบแก้ไขและต่อการศึกษาของผู้อื่น แต่ถ้าโปรแกรมมีขนาดใหญ่ความยุ่งยากและปัญหาข้างต้นก็จะเกิดขึ้นทันที

โปรเจ็คนี้จึงได้มองการทำงานกับคอนโทรลเลอร์เป็นอีกแนวหนึ่ง เพื่อให้ง่ายต่อการศึกษาและตรวจสอบแก้ไขโปรแกรม โดยการสร้างโปรแกรมที่เป็นโมดูลซึ่งแต่ละโมดูลแยกอิสระจากกันและเมื่อต้องการเรียกใช้โมดูลใดก็จะต้องสร้างโปรแกรมเมนขึ้นมาเพื่อเรียกใช้โมดูลเหล่านั้น ซึ่งการทำเช่นนี้จะลดความสับสนในการแก้ไขและค้นหาโปรแกรม และโมดูลที่เขียนขึ้นมาทั้งหมดก็จะเป็นมาตรฐานตายตัวเมื่อนำไปใช้ก็ไม่ต้องแก้ไขใดๆในส่วนโปรแกรม แต่อาจจะต้องแก้ไขบ้างในส่วนของค่าคงที่ต่างๆ ที่เปลี่ยนไปในแต่ละงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Thesis Title Building Standard Controller Module Program For MCS-51

Authors Rusdee Waedueramun

Suchat Petprasomkul

Thesis Advisor Assoc.Prof.Phiphat Laohasongkram

Year 2002

ABSTRACT

Microcontroller(MCS-51) is a famous device in control work and when we work with microcontroller by built assembly language. It has problems if we built large program. The problems are difficult for checking and testing.

This thesis presents a new way for work with microcontroller. It easy to check and test. Every program is independent. The way is built each program to module. If we want to use module, we can make main module for select each standard module. In project each module are standard module, it don't edit source code but you must edit variable for match with your task.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี เพราะได้รับความเมตตาจาก รองศาสตราจารย์ พิพัฒน์ เลาหสงคราม ที่ได้ให้คำแนะนำแก่ผู้วิจัยตลอดมา อีกทั้งยังเอื้อเฟื้ออุปกรณ์เครื่องมือต่างๆ ในการทำปริญญานิพนธ์นี้ ผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณ อาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่าน ที่ให้คำแนะนำอันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

และที่ลืมเสียมิได้ได้ ขอกราบขอบพระคุณบิดาและมารดา อันเป็นที่รักยิ่ง และคุณลุงซึ่งเป็นที่เคารพรักเป็นอย่างสูง ที่เป็นกำลังใจและแรงบันดาลใจในการทำปริญญานิพนธ์ฉบับนี้

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ(ต่อ)

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ(ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมาและเหตุจูงใจของการวิจัย

เนื่องจากปัจจุบันได้มีการใช้อุปกรณ์ประเภท ไมโครคอนโทรลเลอร์กันอย่างแพร่หลาย ซึ่งจะใช้ตัวไมโครคอนโทรลเลอร์ในการควบคุมอุปกรณ์ที่จะนำมาต่อเชื่อมกัน หรือ Moduleที่จะนำมาเชื่อมต่อกันเพื่อจะให้สามารถติดต่อและควบคุม Moduleเหล่านี้ได้ ซึ่งหากเราต้องการที่จะเพิ่ม Module เหล่านี้เข้าไปให้กับตัวไมโครคอนโทรลเลอร์ ก็จำเป็นที่จะต้องเขียนโปรแกรมเพื่อควบคุม Module ให้กับตัวไมโครคอนโทรลเลอร์

1.2 วัตถุประสงค์

1.สามารถเขียนโปรแกรมในลักษณะ Linker/Linking Location ได้

2.สามารถเขียนโปรแกรมที่เป็นโมดูลมาตรฐานสำหรับไมโครคอนโทรลเลอร์ได้

1.3 ขอบเขตของปริญญานิพนธ์

-ขอบเขตทางด้าน Soft Ware

จะเขียนโปรแกรมของโมดูลมาตรฐานในลักษณะ Linker/Linking Location ซึ่งจะใช้ภาษาแอสแซมบลี้ในการเขียนโปรแกรม โดยหากต้องการเพิ่มโมดูลใดๆ ให้กับตัวของไมโครคอนโทรลเลอร์ เราก็สามารถนำโปรแกรมที่เป็นโมดูลมาตรฐานที่เก็บไว้ใน Library มา Add และทำการลิงค์ระหว่างโมดูลเหล่านั้น

-ขอบเขตทางด้าน Hard Ware

1.สร้างบอร์ดตัวมาสเตอร์ โดยใช้ไมโครคอนโทรลเลอร์เบอร์ AT89C52

2.สร้างบอร์ดตัวสเลฟ แสดงผลโดยใช้ไมโครคอนโทรลเลอร์เบอร์ AT89C52

3.สร้างบอร์ดตัวสเลฟโดยใช้ไอซีขยายพอร์ตอินพุตเอาท์พุทเบอร์ PCF8574A

4. สร้างบอร์ดตัวสเลฟโดยใช้ไอซี ADC/DCA เบอร์ PCF8591

5.สร้างบอร์ดแสดงผลแบบ 7-Segment แบบมัลติเพล็กซ์

6. สร้างบอร์ดตัวสเลฟโดยใช้ไอซีฐานเวลา DS1307

7. สร้างบอร์ดตัวสเลฟตรวจวัดอุณหภูมิ DS1820

8. สร้างบอร์ดแสดงผลแบบจอ LCD

9. สร้างบอร์ด Keypad (4x4)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.4 ขั้นตอนการศึกษาและดำเนินการ

1.ศึกษาทฤษฎีการเขียนโปรแกรมภาษาแอสเซมบลีในลักษณะ Linker/Linking Location

2.ศึกษาการทำงานของ MCS-51 และทดลองเขียนโปรแกรมภาษาแอสแซมบลี้ในลักษณะ Linker/Linking Location ลงใน MCS-51

3.ศึกษาระบบบัส I^2C

4..ศึกษาการทำงานของไอซีขยายพอร์ตบนระบบบัส I^2C เบอร์ PCF 8574A

5.ศึกษาการทำงานของไอซี ADC/DCAบนระบบบัส I^2C เบอร์ PCF 8591

6.ศึกษาการทำงานของไอซีตรวจวัดอุณหภูมิ DS1820

7.ศึกษาการทำงานของไอซีสร้างฐานเวลาจริง

8.ศึกษาการทำงานของโมดูลแสดงผลแบบผลึกเหลว

9.ศึกษาการทำงานของโมดูลแสดงผลแบบ LED 7-Segment Multiplex

10.เขียนโปรแกรมรับส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์ตัวมาสเตอร์และตัวสเลฟในลักษณะ Linker/Linking Location และทดลองรับส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์ในรูปแบบของบัส I^2C

11.เขียนโปรแกรมรับข้อมูลจากคีย์แพดไปแสดงผลยัง LED 7-Segment Multiplex และไปแสดงผลยัง LCD ที่ไมโครคอนโทรลเลอร์ที่เป็นมาสเตอร์ต่ออยู่

12.เขียนโปรแกรมรับส่งข้อมูลระหว่างอุปกรณ์มาสเตอร์กับไอซีขยายพอร์ตเบอร์ PCF8574A บนระบบบัส I^2C ไปแสดงผลยัง LCD

13.เขียนโปรแกรมรับส่งข้อมูลระหว่างอุปกรณ์มาสเตอร์กับไอซี ADC/DAC เบอร์ PCF8591 บนระบบบัส I^2C ไปแสดงผลยัง LCD

14.เขียนโปแกรมรับค่าวันเวลาจากไอซีสร้างฐานเวลา และไปแสดงผลยัง LCD ที่ตัวมาสเตอร์และแก้ไขวันเวลาจากคีย์แพดได้

15.เขียนโปรแกรมติดต่อระหว่างคอมพิวเตอร์กับไมโครคอนโทรลเลอร์โดยติดต่อกันทาง Serial Port โดยใช้โมดูลตรวจวัดอุณหภูมิ ตรวจวัดอุณหภูมิและนำค่าที่ได้ไปแสดงผลยังคอมพิวเตอร์และแสดงผลยัง LCD ที่ตัวสเลฟต่ออยู่ โดยสามารถสั่งงานให้ ON หรือ OFF Relay จากคอมพิวเตอร์เมื่ออุณหภูมิสูงขึ้นหรือต่ำลง (เป็นลักษณะ Manual) หรือ ให้ ON-OFF อัตโนมัติก็ได้

16.ทดลองรับส่งข้อมูลกันระหว่างอุปกรณ์มาสเตอร์กับอุปกรณ์สเลฟทั้งหมด

17.สรุปผลการทดลองและปัญหาการทำงานพร้อมทั้งข้อเสนอแนะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.5 ประโยชน์ที่คิดว่าจะได้รับ

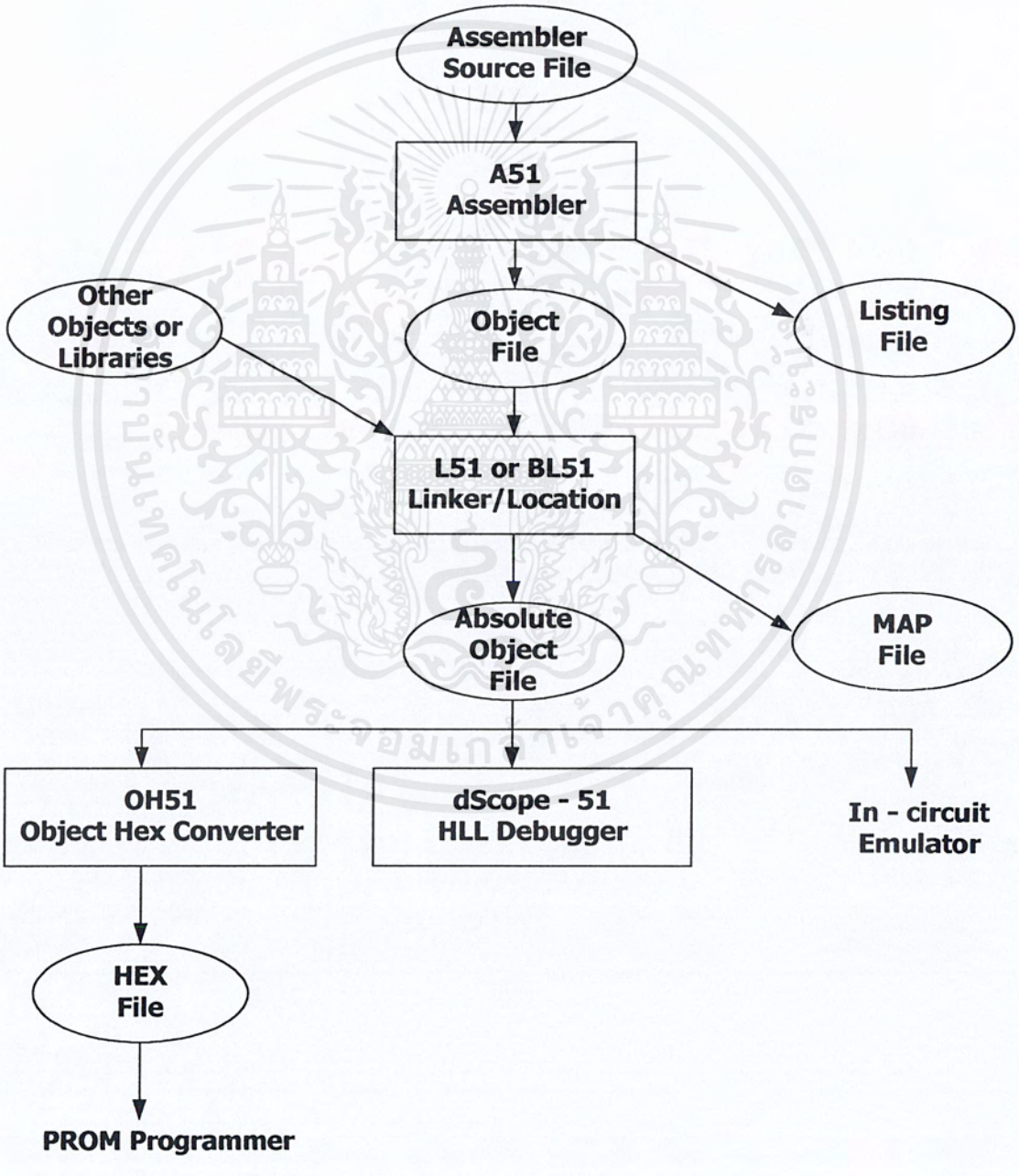
จะได้โปรแกรมที่เป็นโมดูลมาตรฐานสำหรับไมโครคอนโทรลเลอร์ (MCS-51) ซึ่งจะง่ายและสะดวกต่อการใช้งาน และเพื่อเป็นข้อมูลแก่ผู้ที่สนใจนำไปใช้งานและพัฒนาในการควบคุมกระบวนการ ซึ่งในปัจจุบันนี้มีการใช้งานไมโครคอนโทรลเลอร์กันอย่างแพร่หลาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ทฤษฎีหรือหลักการทำงาน

2.1 เทคนิคการเขียนโปรแกรมภาษาแอสแซมบลี้ด้วยวิธีการ Linker & Linking / Location

ขั้นตอนการสร้าง Link File



ภาพที่ 2.1 แสดงขั้นตอนการสร้าง Link File

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปสามารถอธิบายขั้นตอนการทำงานได้ดังนี้

1.ทำการสร้าง Source File

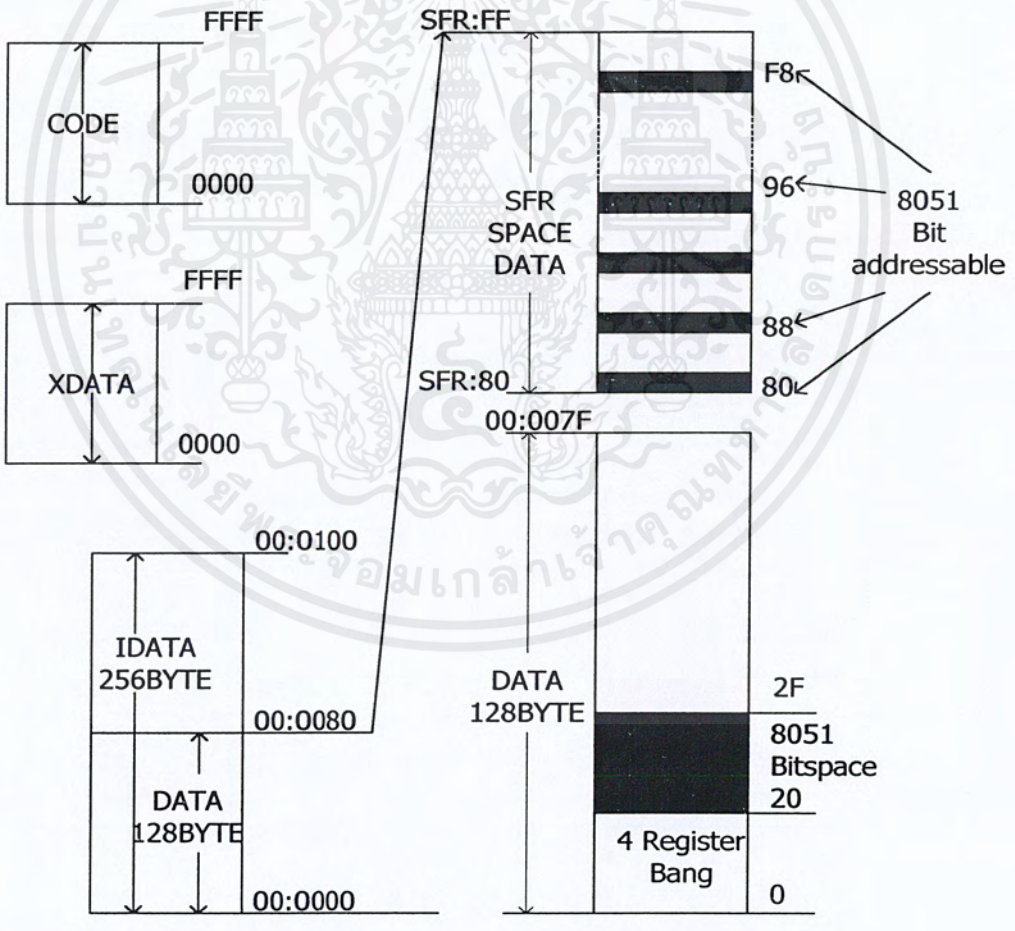
2.ใช้ A51 Assembler ซึ่งจะได้ List File และ Object File ภายหลังจากการ Assembler แล้ว

3.ใช้ L51หรือ BL51 ทำการลิงค์ Object File หรือ Libraries File เข้าด้วยกัน ก็จะได้ Absolute Object File และ MAP File

4.ทำการ Debug โปรแกรมด้วยโดยการใช้ข้อมูลจาก Absolute Object File ด้วย dScope 51

5.หากต้องการนำ Absolute Object File ลงใน Hard Ware ที่ใช้งานจริง ต้องแปลงเป็น HEX File ก่อน แล้วจึงเขียนลง PROM หรือ EPROM

2.1.1 การแบ่งพื้นที่หน่วยความจำในไมโครคอนโทรลเลอร์เมื่อใช้เทคนิคการเขียนแบบ Linker&Linking /Location



ภาพที่ 2.2 แสดงการแบ่งพื้นที่หน่วยความจำในไมโครคอนโทรลเลอร์ เมื่อใช้เทคนิคการเขียนแบบ Linker&Linking /Location

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแบ่งพื้นที่หน่วยความจำในไมโครคอนโทรลเลอร์

DATA	00:0000 - 00:007F	เป็นหน่วยความจำข้อมูลภายใน
Bit	00:0020 - 00:002F	เป็นหน่วยความจำข้อมูลภายใน ที่สามารถเข้าถึงข้อมูลในระดับบิตได้
XDATA	01:0000 - 01:FFFF	เป็นหน่วยความจำข้อมูลภายนอก
CODE	FF:0000 - FF:FFFF	เป็นหน่วยความจำโปรแกรมภายนอก

2.1.2 Writing Assembly Program (การเขียนโปรแกรมภาษาแอสแซมบลี้)

ในส่วนของการเขียนภาษาแอสแซมบลี้บนโปรแกรมไมโครวิชั่นนี้จะมีลักษณะที่แตกต่างจากการเขียนภาษาแอสแซมบลี้โดยทั่วไปเล็กน้อย ซึ่งจะมีลักษณะวิธีการเขียนและส่วนประกอบต่างๆ ดังนี้

2.1.2.1 Assembly Statements

Program Source File ของแอสแซมบลี้จะประกอบไปด้วย Assembler Control,Assembler directive หรือ 8051 Assembly Language instruction(mnumenic) ดังตัวอย่าง

```
$TITLE(Demo Program #1)
        ORG   0000H
        JMP   $
        END
```

จากตัวอย่างประกอบไปด้วย 4 Statement

$TITLE คือ Assembler control , *ORG* และ *END* คือส่วนของ Assembler Directive

JMP คือส่วนของ mnumenic

2.1.2.2 Comments(คำอธิบาย)

เราอาจจะใส่คำอธิบาย (Comment) ในตัวโปรแกรมเพื่อขยายในส่วนนั้นซึ่งการจะใส่คำอธิบายนั้นจะต้องใส่เครื่องหมาย(;)นำหน้าเสมอเพื่อไม่ให้มีส่วน(Cancel)ในการทำงานของตัวโปรแกรม ดังเช่นตัวอย่าง

```
;This is a comment
        NOP   ;This is also a comment
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.2.3 Symbols(สัญลักษณ์)

เราสามารถใช้สัญลักษณ์ในการกำหนด Value,Text Block,Address,Register หรือตัวเลขและเครื่องหมาย

-Symbol Names จะประกอบไปด้วย 31 ตัวดังต่อไปนี้

A – Z , a – z , 0 – 9 , _ , and ?

ส่วนสัญลักษณ์ที่เป็นชื่อสามารถนำหน้าด้วยตัวอักษร ยกเว้น ตัวเลข 0 – 9 และสัญลักษณ์นี้สามารถกำหนดได้หลายวิธี ซึ่งเราสามารถใช้สัญลักษณ์กำหนดค่าจำนวนเต็มให้กับตัวแปร โดยการใช้คำสั่ง EQU หรือ SET ดังตัวอย่าง

ตัวอย่าง

```
NUMBER_FIVEEQU   5
TRUE_FLAG        SET    1
FLASH_FLAG       SET    0
```

และสามารถกำหนดสัญลักษณ์ให้เป็นคำอธิบายก็ได้

ตัวอย่าง

```
LABEL1:          DJNZ   R0,LABEL1
```

หรือสามารถกำหนดสัญลักษณ์ให้อ้างถึงที่ตั้งของหน่วยความจำ

ตัวอย่าง

```
SERIAL_BUFFER    DATA   99H
```

2.1.2.4 Label

Label คือชนิดของสัญลักษณ์ที่เรากำหนดซึ่ง LABELนั้นจะต้องกำหนดสถานที่ตั้ง และเมื่อกำหนดแล้ว LABEL ก็จะแสดงตำแหน่งออกมา ซึ่งหลักของการใช้สัญลักษณ์ที่เป็นชื่อจะถูกใช้เป็น LABEL ทั้งหมด

เมื่อมีการกำหนด LABEL จะต้องให้ข้อความแรกมี COLON (:)ตามหลัง และจะต้องมีการ TAB หรือ SPACES เพื่อเป็นการกำหนดขอบเขตของแต่ละส่วนดังเช่นตัวอย่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่าง

```
LABEL1:    DS    2
LABEL2:                              ;Label by itself
NUMBER:    DB    27, 33, 'STRING', 0    ;Label at a message
COPY:      MOV   R6, #12H               ;Label in a program
```

2.1.2.5 Operands

Operands คือ ข้อความหรือส่วนที่ขยาย กับรายละเอียดในส่วนที่เป็นคำสั่ง และคำสั่งในการ Assembler นั้นต้องการ Operand ที่เป็นค่าคงที่หรือสัญลักษณ์ ดังตัวอย่าง

ตัวอย่าง

```
VVV        EQU   3
           DS    10H
```

Operand ที่ต้องการนั้น จะขึ้นอยู่กับชนิดของชุดคำสั่ง ซึ่งในตารางจะเป็นชนิดของ Operand

ตารางที่ 2.1 แสดงชนิดของ Operand

Operand	**Description**
Immediate Data	สัญลักษณ์หรือค่าคงที่ ใช้เป็นค่าตัวเลข
Direct Bit Addresses	สัญลักษณ์หรือค่าคงที่ ใช้อ้างถึง Bit Address
Program Addresses	สัญลักษณ์หรือค่าคงที่ ใช้อ้างถึง Code Address
Direct Address Data	สัญลักษณ์หรือค่าคงที่ ใช้อ้างถึง Data Address
Indirect Address	ใช้อ้างถึงที่ตำแหน่งของ Memory
Special Assembler Symbol	ชื่อ Register

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.3 Special Assembler Symbols

ใน 8051 นั้นจะมี Register ดังนี้

ตารางที่ 2.2 แสดงชนิดของ Register

Register	Description
A	รีจีสเตอร์นี้จะมีขนาด 8 บิต ซึ่งในคำสั่งช่วยจำจะอ้างอิงถึงรีจีสเตอร์ชนิดนี้ ซึ่งคำสั่งที่จะอ่านหรือเก็บข้อมูลกับหน่วยความจำภายนอกจะต้องกระทำผ่านรีจีสเตอร์นี้เท่านั้น
DPTR	รีจีสเตอร์ DPTR จะมีขนาด 16 บิต ซึ่งจะใช้ชี้ตำแหน่ง Address ในหน่วยความจำ XDATA หรือ CODE
PC	Program counter มีขนาด 16 บิต ซึ่งมันจะใส่ค่า Address ของชุดคำสั่งที่ Run ไปแล้ว
C	Carry Flag จะแสดงค่าสถานะของการเกิดการทดบิต
AB	Register A และ B นั้นจะใช้ในชุดคำสั่ง MUL และ DIV
R0 – R7	จะเป็น Bank Register ที่มีขนาด 8 บิต
AR0 – AR7	จะใช้แสดงค่าตำแหน่งข้อมูลของ แบงค์รีจีสเตอร์ R0 – R7 ซึ่งเราสามารถเลือกใช้แบงค์รีจีสเตอร์ใดก็ได้ โดยการใช้คำสั่ง *USING* ซึ่งจะอธิบายในส่วนต่อไป

2.1.3.1 Immediate Data

ในส่วนของ Operand Immediate Data จะเป็นการกำหนดค่าของข้อมูลลงไปในหน่วยความจำได้เลย ซึ่งในการใช้ชุดคำสั่งนั้นจะต้องมีเครื่องหมาย(#)นำหน้าส่วนที่เป็น Data เสมอ

<u>ตัวอย่าง</u>

```
MOV     A,#0E0H         ;Load 0E0H into the accumulator
MOV     DPTR,#8000H     ;Load 8000H into the data pointer
ANL     A,#128H                 ;AND Accumulator with 128H
XRL     R0,#0FFH        ;XOR RO with 0FFH
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.3.2 Indirect Address

เป็นการกำหนดข้อมูลลงในหน่วยความจำโดยทางอ้อม ซึ่งจะมีชนิดของ Memory ที่ใช้ในการอ้างอิงถึงคือ

2.1.3.3 IDATA

หากใช้ Memoryในส่วนนี้ จะต้องใช้ Register R0 หรือ R1 ซึ่ง Register เหล่านี้จะมีอยู่ภายใน DATA Memory ที่ตำแหน่ง 0H..07H โดยเราจะใช้คำสั่ง

ตัวอย่าง

```
; Buffer is a symbol with class IDATA or DATA
MOV     R0,#BUFFER      ;Load the address
MOV     A,@R0           ;The indirect access
```

2.1.3.4 XDATA

XDATA memory สามารถใช้โดยใช้คำสั่ง MOVX โดย Register DPTR หรือ Register R0,R1

ตัวอย่าง

```
;XBUFFER is a symbol with class XDATA
MOV     DPTR,#XBUFFER   ;Load Address
MOVX    @DPTR,A         ;access via DPTR
MOV     R1,#XBUFFER     ;load Address
MOVX    A,@R1           ;access via R0 or R1
```

2.1.3.5 CODE

CODE Memory สามารถใช้คำสั่ง MOVC ได้ โดย Register DPTR

ตัวอย่าง

```
;TABLE is a symbol of class CODE
MOV     DPTR,#TABLE     ;Load address   of table
MOV     A,#3            ;Load offset into table
MOVC    A,@A+DPTR       ;access via MOVC instruction
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.3.6 Direct Data Address

Direct Data Addres จะเป็นกำหนดค่าของข้อมูลลงในAddressที่อ้างถึงโดยสามารถใช้ SFR ของ 8051 ในการกระทำในส่วนของ Operand ได้

ตัวอย่าง

```
;Accesses to DATA Space
VALUE   DATA    20H
        MOV     50H,A
        MOV     R0,VALUE

;Accesses to EDATA space
EVAR    EDATA   1000H
        MOV     R5,EDATA    2000H
        MOV     EVAR,R4
```

2.1.3.7 Direct Bit Address

Direct Bit Address จะเป็นกำหนดค่าของข้อมูลลงใน Address ที่อ้างถึง โดยเป็นการกำหนดในลักษณะการกระทำแบบบิต สามารถใช้ SFR ของ 8051 ในการกระทำในส่วนของ Operand ได้ ซึ่งในการระบุถึงบิต ณ.ตำแหน่งของข้อมูลที่อยู่นั้นจะต้องใช้สัญลักษณ์ (.) และตามด้วยตำแหน่งของบิตที่ต้องการกระทำดังตัวอย่าง

```
;Access to BIT class
;Also variables with the class DATA BITADDRESSABLE can be accessed
        SETB    20H.6       ;set bit 6 in location 20H
        CLR     10          ;clear bit  2 in location 21H ,this is the bit
                            address 10
                            ;Accesses to EBIT space
                            ;also variables with the class DATA can be
                            accessed
        MOV     40H.5,C
        SETB    DPL.7       ;set bit 7 in
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.3.8 Program Addressing

Program Addressing คือ ตำแหน่งของ Operand ที่อยู่ในหน่วยความจำ CODE

2.1.4 Assembler Directive (คำสั่งในการ Assembler)

ในส่วนของ Assembler จะแยกประเภทของคำสั่งดังนี้

2.1.4.1 Segment Control

เป็นการกำหนดส่วนของหน่วยความจำที่ต้องการใช้

Generic Segments: *SEGMENT,RSEG*

ตัวอย่าง *MYPROG SEGMENT CODE*

จากตัวอย่างเป็นการกำหนด SEGMENT ชื่อ MYPROG ซึ่งจะอยู่ในส่วนของ Code Memory

ตัวอย่าง *RSEG MYPROG*

จากตัวอย่างคำสั่ง RSEG จะใช้เมื่อเราต้องการที่จะเขียนโปรแกรมหรือกำหนดค่าลงในส่วนของ SEGMENT MYPROG

Absolute Segment: *CSEG,DSEG,BSEG,ISEG,XSEG*

ตัวอย่าง

BSEG AT ADDRESS(FOR BIT SEGMENT)
CSEG AT ADDRESS(FOR CODE SEGMENT)
DSEG AT ADDRESS(FOR DATA SEGMENT)
ISEG AT ADDRESS(FOR IDATA SEGMENT)
XSEG AT ADDRESS(FOR XDATA SEGMENT)

จากตัวอย่างเป็นการกำหนด ADDRESS เริ่มต้นที่จะทำงานของ SEGMENT ต่างๆ

2.1.4.2 Memory initialization

DB เป็นการกำหนดข้อมูลลงไปในส่วนของ SEGMENT CODE ในขนาด 8 BIT

DW เป็นการกำหนดข้อมูลลงไปในส่วนของ SEGMENT CODE ในขนาด 16 BIT

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.4.3 Program Linkage

EXTRN เป็นการประกาศชื่อของ โปรแกรมย่อย ซึ่งอยู่ในโปรแกรมอื่นที่เราต้องการดึงมาใช้ในโปรแกรมหลักของเรา รูปแบบการใช้เช่น

ตัวอย่าง *EXTRN CODE (PUT_CRLF, PUTSTRING)*

PUBLIC เป็นการประกาศชื่อ โปรแกรมย่อย ที่อยู่ในโปรแกรมเพื่อให้โปรแกรมอื่นสามารถดึง Segment เหล่านี้ไปใช้งานได้ รูปแบบการใช้เช่น

ตัวอย่าง *PUBLIC PUT_CRLF, PUTSTRING*

NAME เป็นการกำหนดชื่อของ MODUL

ตัวอย่าง *NAME MODULNAME*

2.1.4.4 Adress control

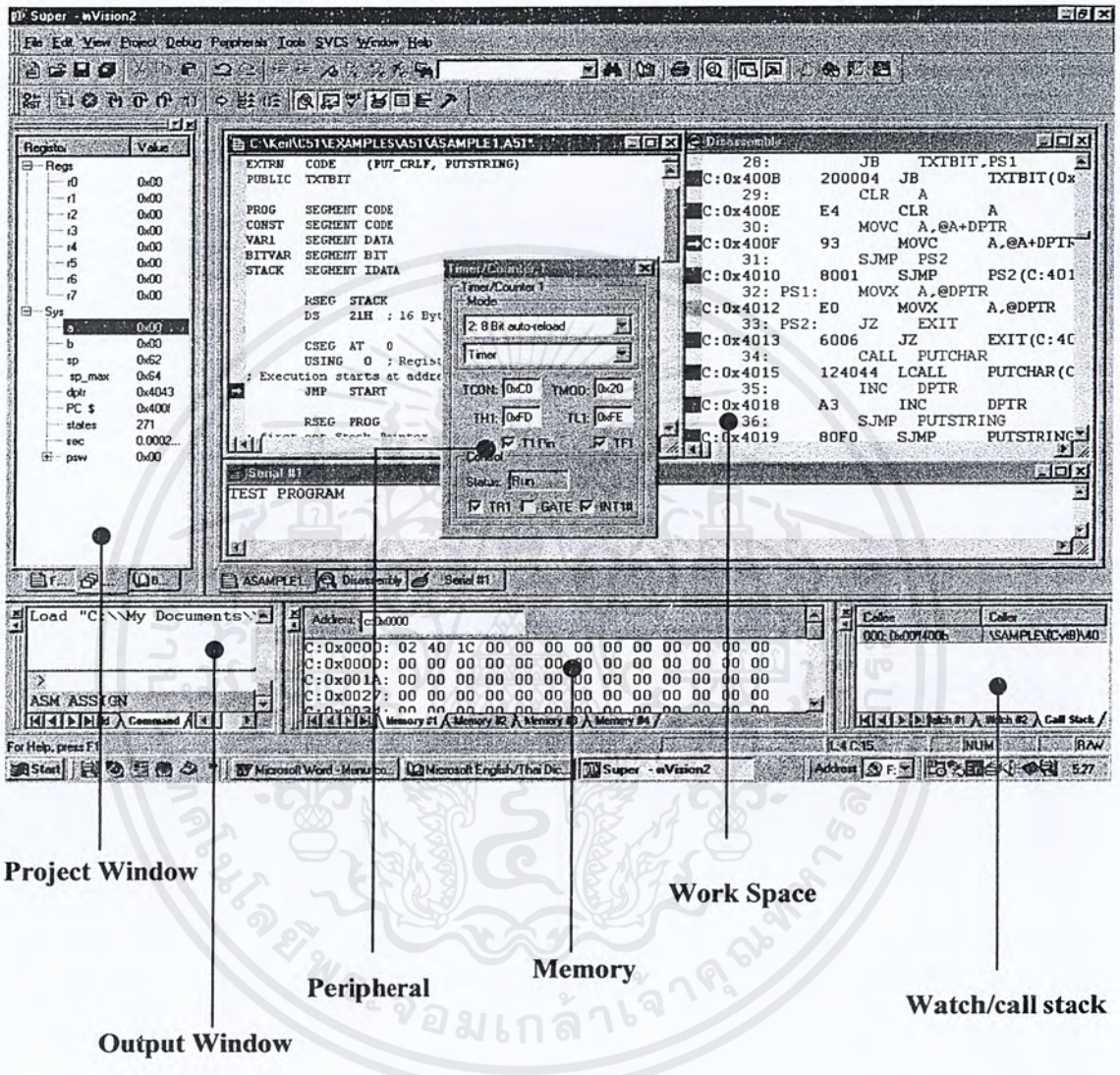
USING ใช้กำหนดตำแหน่งของ REGISTER BANK ที่จะใช้งาน

ตัวอย่าง *USING 0* (USES BANK 0)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 โปรแกรมไมโครวิชั่น

หน้าต่างการทำงานของไมโครวิชั่น

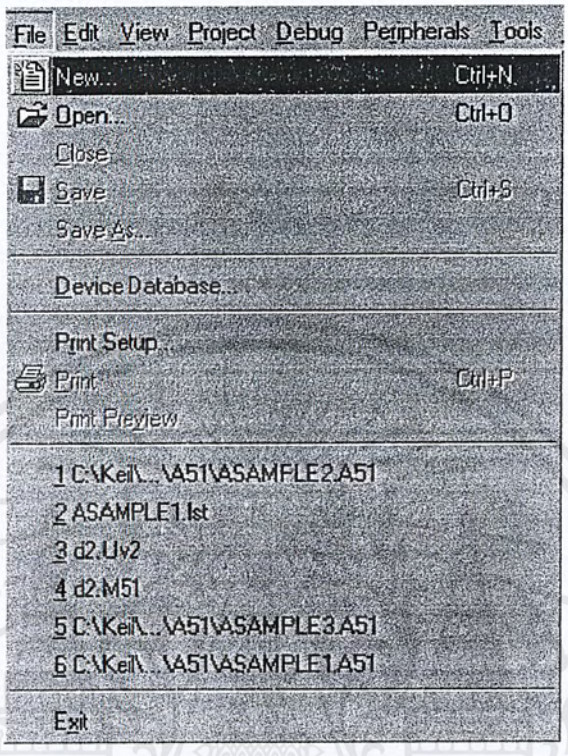


ภาพที่ 2.3 แสดงหน้าต่างการทำงานของโปรแกรมไมโครวิชั่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.1 การใช้คำสั่งในเมนู

เมนู File Menu และ คำสั่ง File

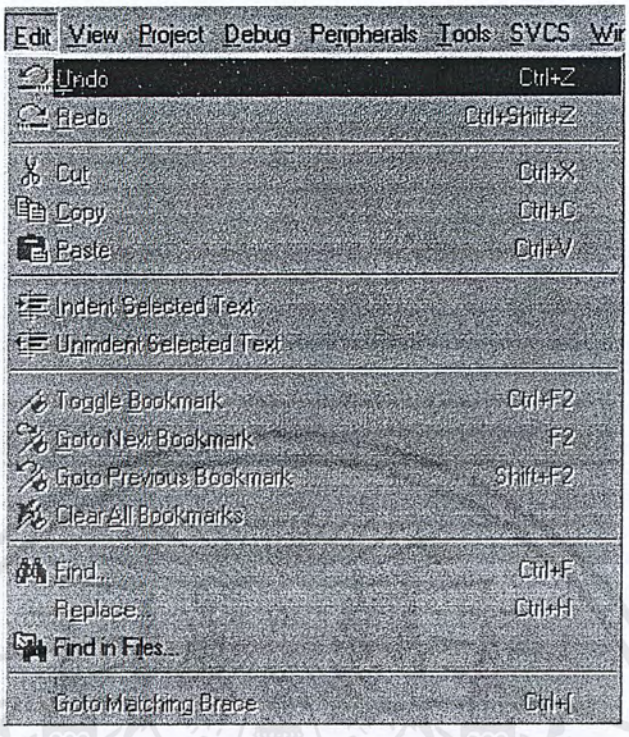


ภาพที่ 2.4 แสดงหน้าต่างเมนู File Menu และ คำสั่ง File

1.New	ต้องการสร้างไฟล์โปรแกรมใหม่
2.Open	เปิดไฟล์โปรแกรมจากภายนอก
3.Close	ปิดไฟล์โปรแกรม
4.Save	บันทึกไฟล์โปรแกรม
Save As	บันทึกไฟล์โปรแกรมที่ต้องการสร้างชื่อใหม่
Device Database..	ข้อมูลของ ไมโครคอนโทรลเลอร์
Print Setup	ตั้งค่าเครื่องพิมพ์
5.Print	พิมพ์
Print Preview	ภาพก่อนพิมพ์
1..9	เปิดไฟล์โปรแกรมที่ใช้เมื่อเร็วๆนี้
Exit	ออกจากโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมนูEdit และ คำสั่ง Editor



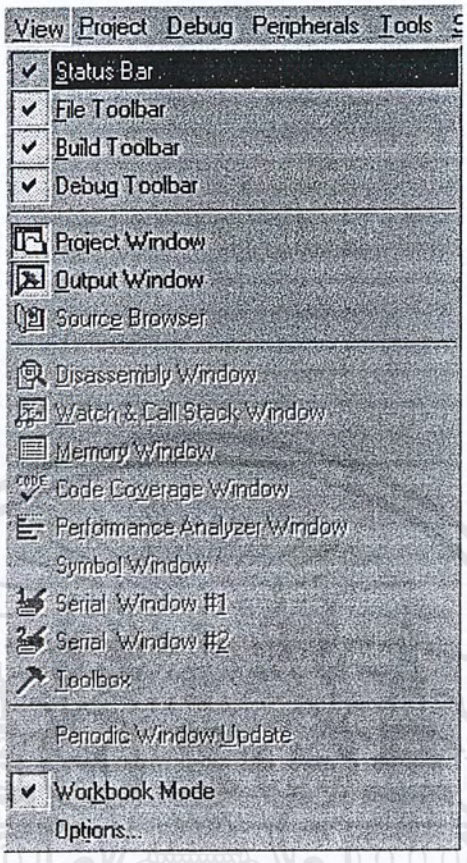
ภาพที่ 2.5 แสดงหน้าต่างเมนูEdit และ คำสั่ง Editor

1.Undo	เลิกทำครั้งสุดท้าย
2.Redo	ซ้ำจากการเลิกทำครั้งสุดท้าย
3.Cut	ตัดข้อความที่ได้เลือกไว้
4.Copy	คัดลอกข้อความที่ได้เลือกไว้
5.Past	วางข้อความที่ได้เลือกไว้
6.Ident Select Text	เลื่อนข้อความที่ได้เลือกไว้ ไปทางด้านขวาที่ละแท็ป
7. Unident Select Text	เลื่อนข้อความที่ได้เลือกไว้ ไปทางด้านซ้ายที่ละแท็ป
8.Toggle Bookmark	เครื่องหมายในบรรทัดที่ต้องการ
9.Goto Next Bookmark	ไปยังบรรทัดที่ทำเครื่องหมายไว้
10.Goto Previous Bookmark	กลับไปยังบรรทัดที่ทำเครื่องหมายไว้
11.Clear All Bookmark	ลบเครื่องหมายที่ทำไว้ทั้งหมด
12.Find	ค้นหาคำในไฟล์โปรแกรมนั้น
13.Replace	ค้นหาคำในไฟล์โปรแกรมนั้น และแทนที่คำที่ต้องการเปลี่ยน
14.Find And File……	ค้นหาชื่อไฟล์โปรแกรมและที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

เมนู View



ภาพที่ 2.6 แสดงหน้าต่างเมนูView

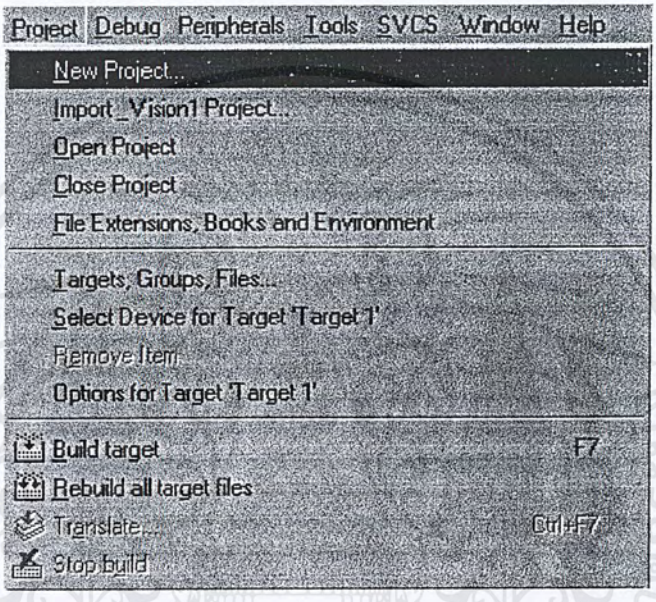
1.Status Bar	แสดงบาร์ แสดงสถานะ
2.File Toolbar	แสดงคำสั่งของชุดเมนู File
3.Build Toolbar	แสดงคำสั่ง Build Target (คือการสร้างไฟล์ลิงค์)
4.Debug Toolbar	แสดงคำสั่งของชุดเมนู Debug
5.Project Window	แสดงหน้าต่าง Project
6.Output Window	แสดงหน้าต่าง Output
7.Source Browser	แสดงหน้าต่าง Source Browser
8.Disassembly Window	แสดงหน้าต่าง Disassembly
9.Watch & Call Stack Window	แสดงหน้าต่าง Watch & Call Stack
10.Memory Window	แสดงหน้าต่าง Memory
11.Code Converge Window	แสดงหน้าต่าง Code Converge
12.Performance Analyzer Window	แสดงหน้าต่าง Performance Analyzer
13.Symbol Window	แสดงหน้าต่าง Symbol

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

50435

14.Serial Window #1	แสดงหน้าต่าง Serial #1
15.Serial Window #2	แสดงหน้าต่าง Serial #2
16.Toolbox	แสดงหน้าต่าง Toolbox
17.Workbook Mode	แสดงกรอบการทำงานกับ tab windows
18.Option	เปลี่ยนสี,ตัวอักษร,แป้นลัดและอื่นๆ

เมนู Project และ ชุดคำสั่ง Project



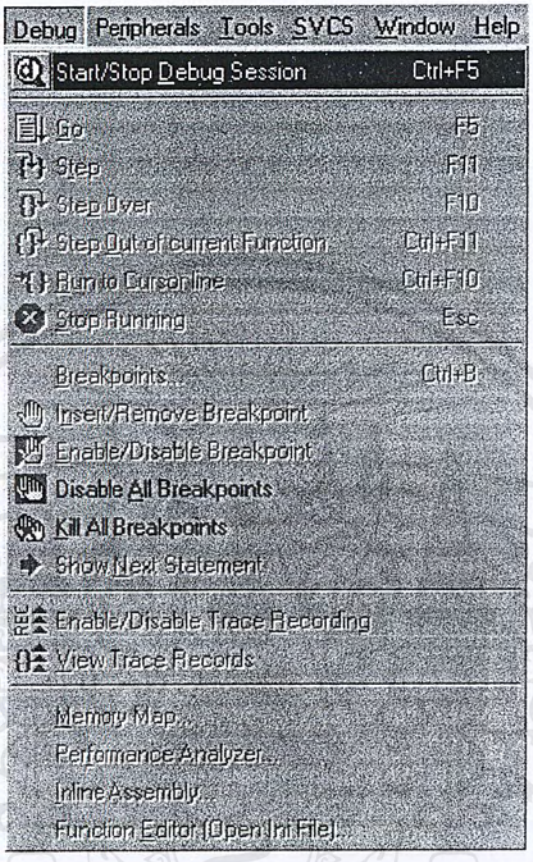
ภาพที่ 2.7 แสดงหน้าต่างเมนู Project และ ชุดคำสั่ง Project

1.New Project	สร้าง Project ใหม่
2.Import_Vision1 Project…	ปรับ File เวอร์ชัน1 เพื่อใช้เวอร์ชัน2
3.Open Project….	เปิด Project
4.Close Project….	ปิด Project
5.File Extensions,Books and Environment	เลือกนามสกุลของFileต่างๆที่ใช้สร้างProject
6.Target,Group,File….	สร้าง Target,Source Groupและ File ใหม่
7.Select Device For Target	เลือก CPU จากฐานข้อมูล
8.Remove Item	ลบกลุ่มหรือไฟล์โปรแกรมออกจาก Project
9.Option for Target	เปลี่ยนOption สำหรับtar,GroupหรือFile
10.Build Target	การ Link Source File ทั้ง Target
11.Rebuild All target File	การ Link Source File ทั้ง Target

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

13.Translate	การ compile Source File
14.Stop Build	หยุดการทำงานของ Build Target

เมนู Debug และ ชุดคำสั่ง Debug



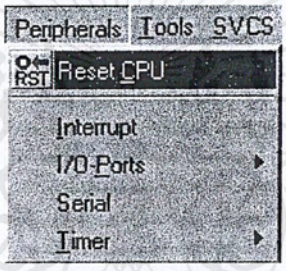
ภาพที่ 2.8 แสดงหน้าต่างเมนู Debug และ ชุดคำสั่ง Debug

1.Start/Stop Debugging	เริ่มหรือหยุด Debug โปรแกรม
2.Go	Run โปรแกรม
3.Step	Run โปรแกรมเป็น Step
4.Step Over	Run คำสั่งที่มีภายในตัวโปรแกรมเท่านั้น
5.Step Out of current Function	
6.Stop Running	หยุดการ Run Program
7.Breakpoints	แสดงตำแหน่งของจุด Breakpoint
8.Insert/Remove Breakpoint	กำหนดหรือยกเลิกตำแหน่งจุด Breakpoint
9.Enable/Disable Breakpoint	อนุญาติและไม่อนุญาติให้ใช้จุด Breakpoint
10.Disable All Breakpoints	ไม่อนุญาติทั้งหมด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

11.Kill All Breakpoints	ลบจุด Breakpoint ทั้งหมด
12.Show Next Statement	การ Run โดยไม่แสดง Code Program
13. Enable/Disable Trace Recording	บันทึกโปรแกรมในส่วนที่ Run ไปแล้ว
14.View Trace Records	แสดงส่วนที่บันทึกไว้
15.Memory Map…	แสดงการใช้งานของตำแหน่ง Memory
16.Performance Analyzer…	วิเคราะห์ประสิทธิภาพของโปรแกรม
17.Inline Assembly…	ไปยังบรรทัดที่กำหนด
18.Function Editor (Open Ini File)	แก้ไข File ในขณะที่กำลัง Debug

เมนู Peripherals

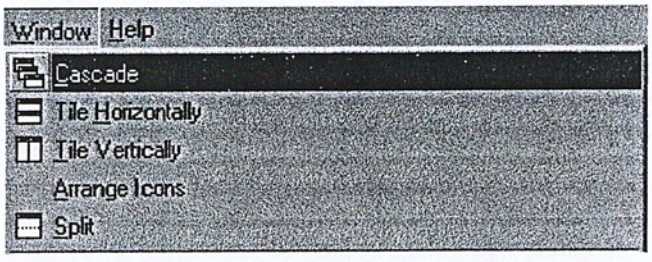


ภาพที่ 2.9 แสดงหน้าต่างเมนู Peripherals

1.Reset CPU	รีเซ็ท CPU.
2.Interrupt	แสดงอินเตอร์รับของ CPU เมื่อRun Program
3.I/O-Port	แสดงค่าของ Port ต่างๆ
4.Serial	แสดงข้อมูลเกี่ยวกับ Serial Port ทั้งหมด
5.Timer	แสดงค่าภายใน Timer ต่างๆของ CPU

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

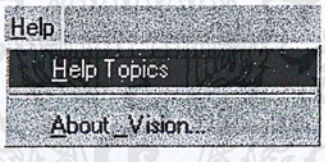
เมนู Window



ภาพที่ 2.10 แสดงหน้าต่างเมนู Window

1.Cascade	แสดง File แบบขั้นบันได
2. Tile Horizontally	แสดง File แบบแบ่งทางแนวนอน
3.Tile Vertically	แสดง File แบบแบ่งทางแนวตั้ง
Arrange Icons	จัดเรียง icon File ต่างๆ
4.Split	แบ่งการแสดง File เป็น 4 ส่วน

เมนู Help



ภาพที่ 2.11 แสดงหน้าต่างเมนู Help

1.Help Topics	หัวข้อที่ต้องการทราบ
2.About vision	รายละเอียดของ Software

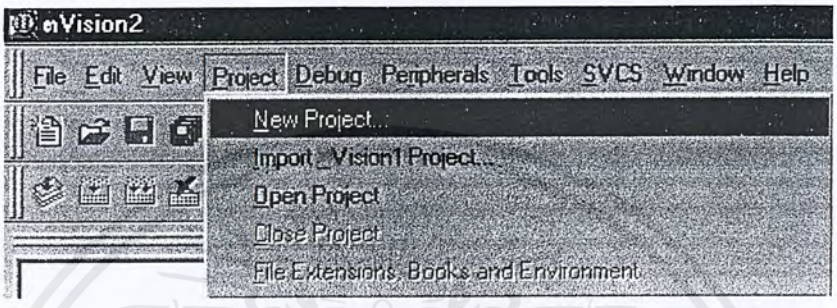
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.2 การสร้างโปรเจค

Create a Project

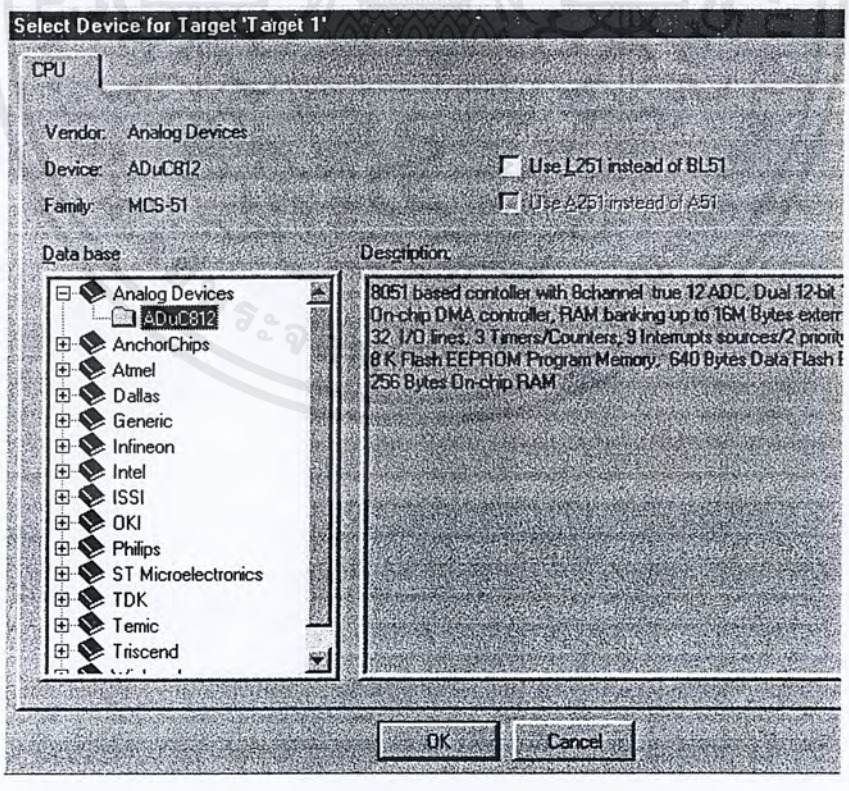
μ Vision2 ประกอบไปด้วยระบบปฏิบัติการที่ออกแบบมาอย่างเรียบง่ายโดยใช้ CPU ตระกูล MCS-51 ฉะนั้น จึงจำเป็นที่จะต้องปฏิบัติตามลำดับขั้นตอนการสร้าง Project ดังนี้

1. เปิดโปรแกรม μ Vision2 จากนั้นไปที่เมนู Project เพื่อเลือกที่จะสร้าง Project ใหม่ขึ้นมา โดยทำการเลือกที่ New Project แสดงดังภาพ



ภาพที่ 2.12 แสดงหน้าต่างเมนู Project

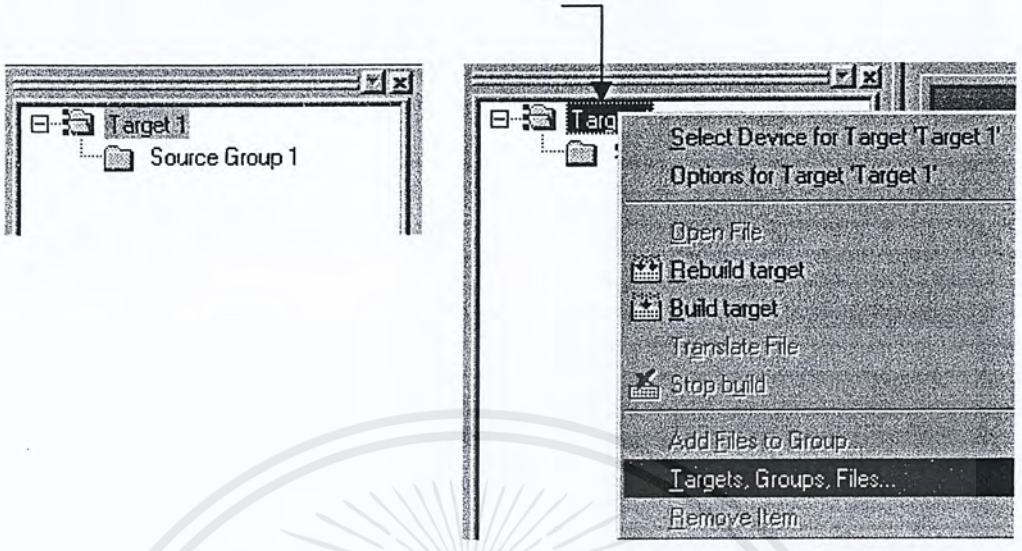
2. จากนั้นจะต้องเลือกชนิดของ CPUที่เหมาะสมกับ Project ที่ต้องการ



ภาพที่ 2.13 แสดงหน้าต่างการเลือกชนิดของ CPU

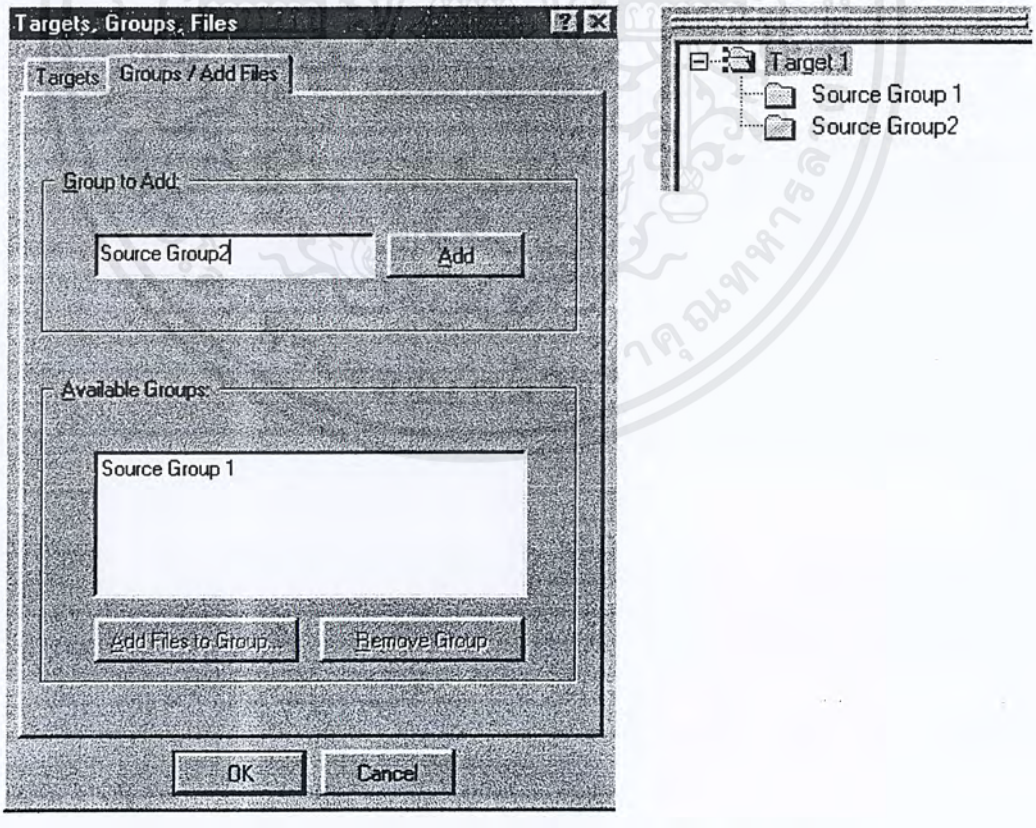
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.หลังจากเลือกชนิดของ CPU ที่ต้องการแล้วตัวโปรแกรมจะสร้าง Group File มาให้หรือจะสร้างขึ้นมาเพิ่มขึ้นอีก โดยคลิกเมาส์ด้านขวาที่ Target ดังภาพ



ภาพที่ 2.14 แสดงหน้าต่างการเลือก Group File

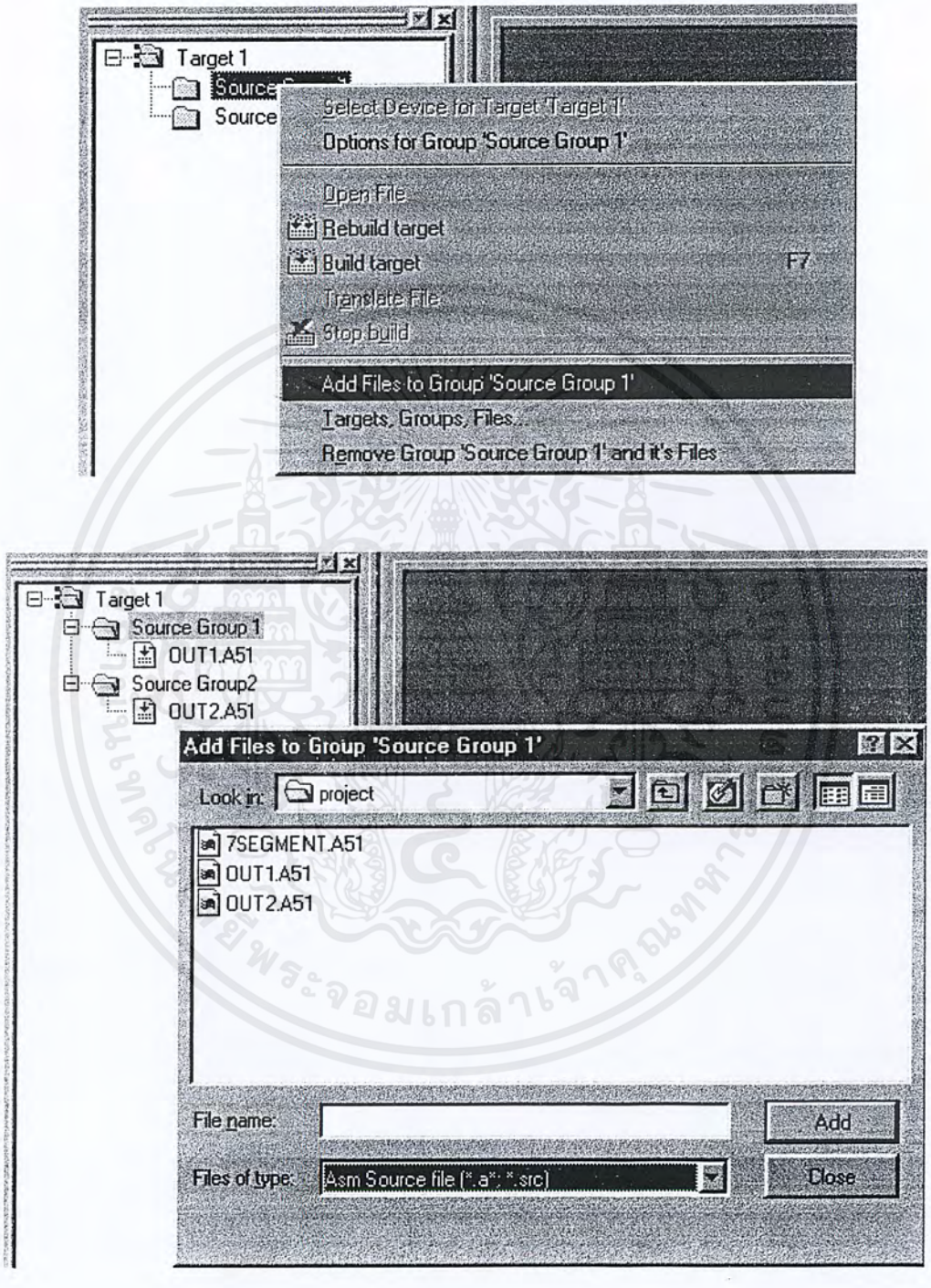
4.จากนั้นพิมพ์ ชื่อ Group File ที่ต้องการ และ Add ลงไป



ภาพที่ 2.15 แสดงหน้าต่างแสดงการเพิ่ม Group File

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.เมื่อได้ Source Group ที่ต้องการแล้ว จากนั้น Add File ที่ต้องการโดยการ คลิ๊กเมาส์ขวาที่ Source Group ที่ต้องการ

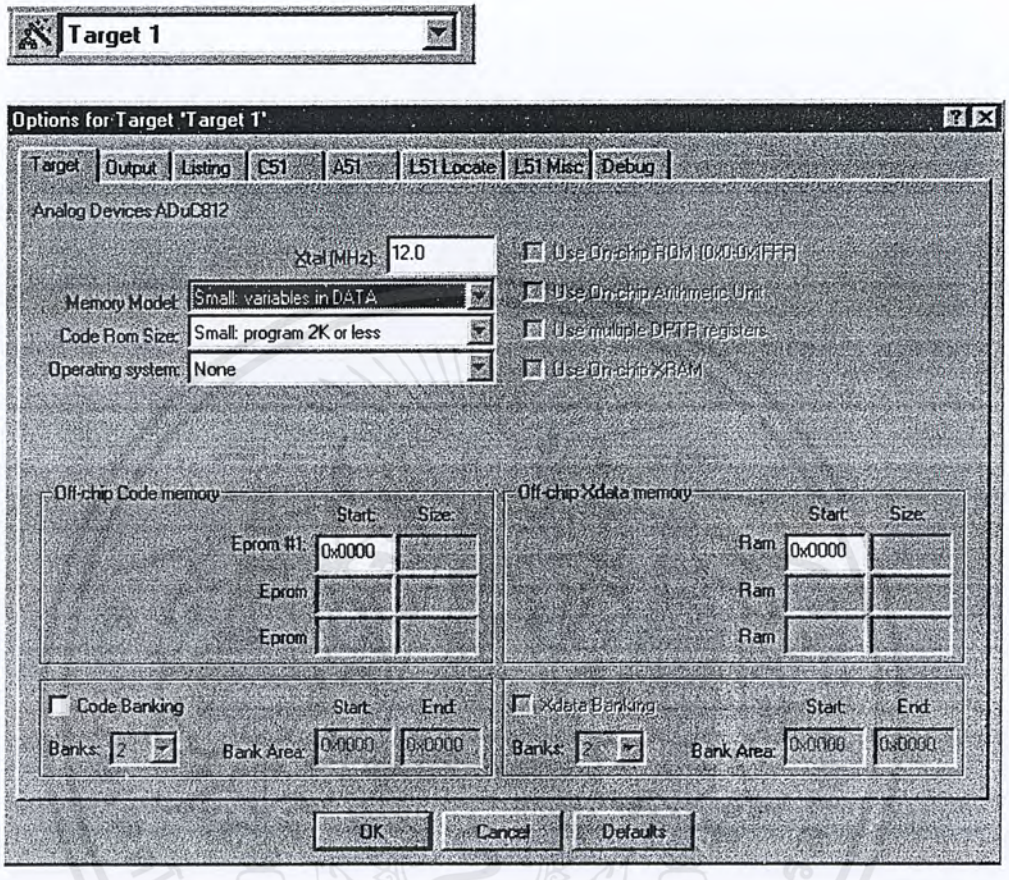


ภาพที่ 2.16 แสดงหน้าต่างการ Add File ใหม่ ลงใน Source Group

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. ก่อนที่จะทำการ Assembler โปรแกรมนั้นจำเป็นต้องมีการกำหนดค่าต่างๆ ของ Target นั้นว่าต้องมีคุณสมบัติใดบ้างซึ่งมีดังนี้

Dialog Target



ภาพที่ 2.17 แสดงหน้าต่างของ Dialog Target

Xtal	แสดงความถี่ของคริสตอล
Memory Model	จำลอง Memory ที่ใช้ในการ Compile โปรแกรม ซึ่งแบ่งเป็น 3 รูปแบบคือ

1.Small: variables in Data เมื่อโปรแกรมที่เขียนขึ้นต้องติดต่อกับหน่วยความภายในเป็นส่วนมาก ควรเลือกใช้ Memory Model แบบนี้

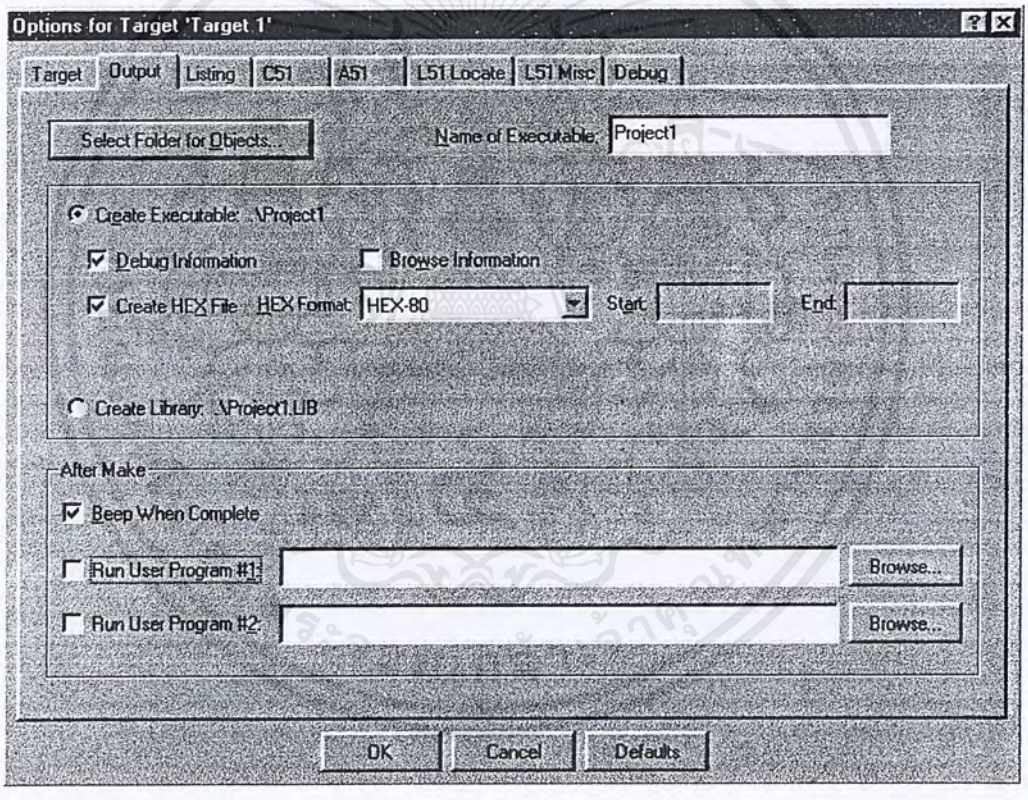
2.Compact: variables in Pdata โดย Model แบบนี้จะกำหนดให้ใช้หน่วยความจำภายนอกได้สูงสุด 256 bytes และสามารถเข้าถึงหน่วยความจำได้โดยใช้รีจิสเตอร์ R0 และ R1 อ้างในตำแหน่งไบต์ต่ำ และใช้ port2 อ้างตำแหน่งไบต์สูง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.Large: variables in Xdata โดย Model แบบนี้จะกำหนดให้ใช้หน่วยความจำภายนอกได้สูงสุด 64 Kbytes และสามารถเข้าถึงหน่วยความจำภายนอกได้โดยใช้รีจิสเตอร์ DPTR

Code Rom Size	กำหนดขนาดของหน่วยความจำโปรแกรม
Operating system	ระบบปฏิบัติการ RTX-Real Time ซึ่งมีประโยชน์เมื่อต้องทำงานหลายอย่างใน CPU เพียงตัวเดียว
Off-chip{Code,Xdata}memory	แสดงตำแหน่งเริ่มต้นและขนาดของ EPROM
Code,XdataBanking	แสดงการขยายหน่วยความจำ โดยต้องขึ้นอยู่กับตัว Hardware ด้วย
Code Banking	ใช้ในกรณีที่ Hardware ต้องการขยายหน่วยความจำโปรแกรมภายนอกเพิ่ม

Dialog Output

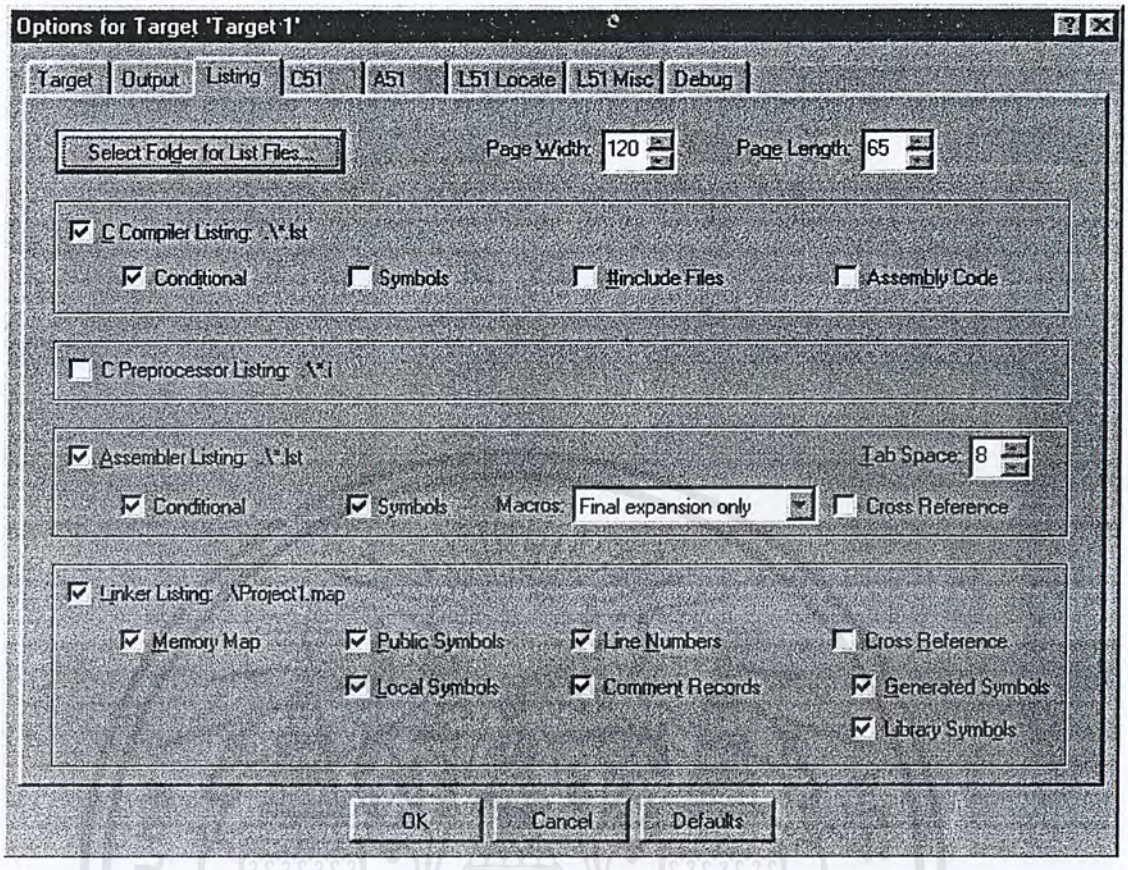


ภาพที่ 2.18 แสดงหน้าต่างของ Dialog Output

Select Folder for Object….	เลือก Folder เพื่อเก็บ Object File
Creat Executable	สร้าง File ที่ใช้ในการ Debug
Creat Library	สร้าง Library file
Run User Program #1,#2	ต้องการ Debug โปรแกรมด้วยโปรแกรมตัวอื่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Dialog Listing

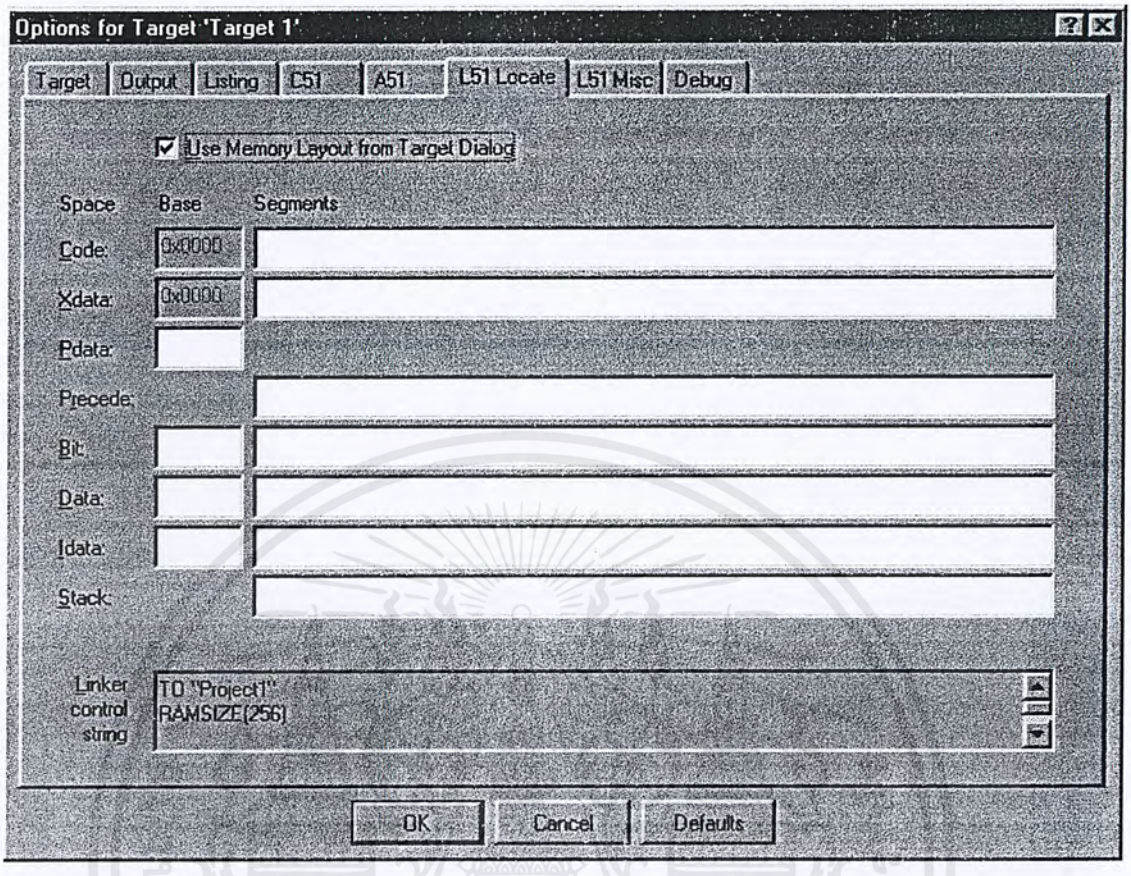


ภาพที่ 2.19 แสดงหน้าต่างของ Dialog Listing

Select Folder for List File...	กำหนดโฟลเดอร์ที่จะเก็บ List File
Page Wide,Length	กำหนดจำนวนบรรทัดและจำนวนตัวอักษร ในแต่ละบรรทัดของ List File
Assembler Listing	กำหนดให้แสดงตำแหน่งของSegment ต่างๆ ในแต่ละโมดูล
Linker Listing	กำหนดให้โปรแกรมสร้างไฟล์นามสกุล .M51 ซึ่งจะแสดงตำแหน่งของ Memory ต่างๆ ที่ได้กำหนดไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

L51 Locate



ภาพที่ 2.20 แสดงหน้าต่างการทำงานของ Dialog L51 Locate

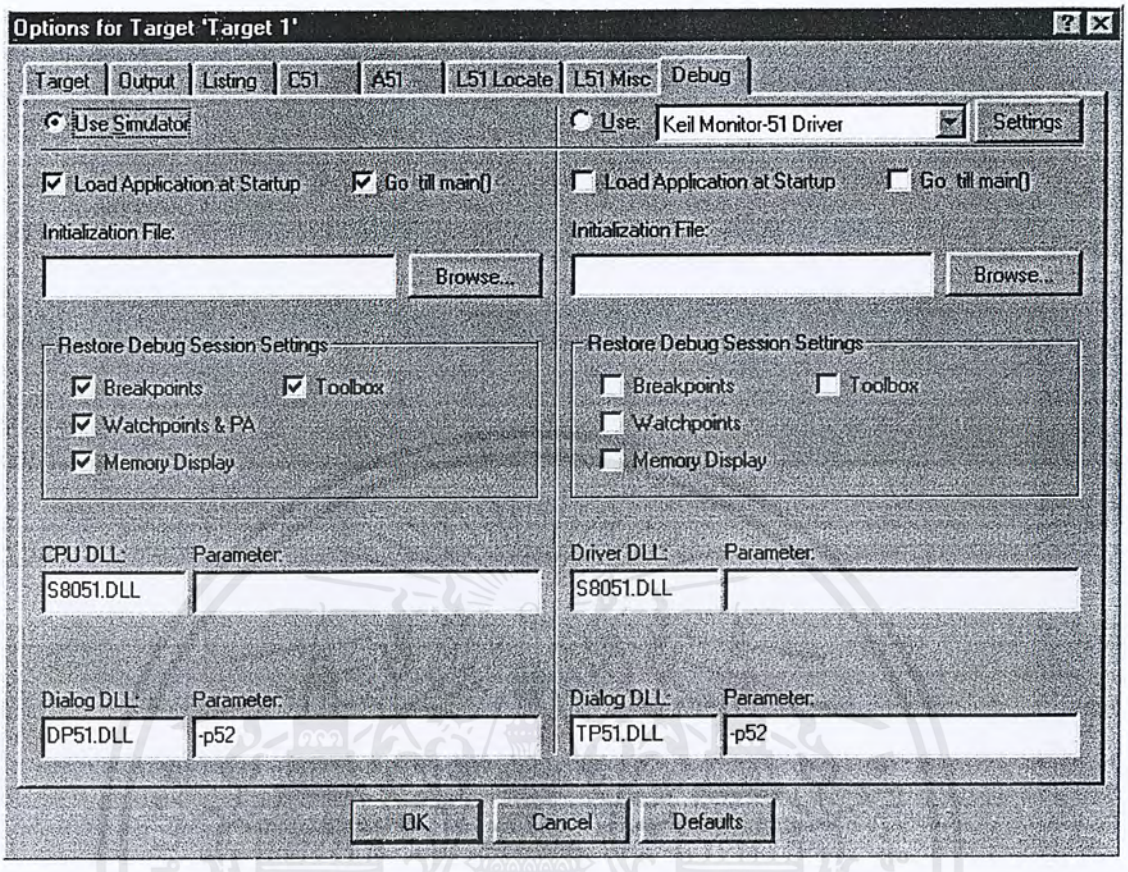
กำหนดตำแหน่งของ Segment ต่างๆ ในส่วนของ Memory ที่เลือกไว้ ตัวอย่างคำสั่งเช่น

จากคำสั่ง Segment Var1,Var2 ซึ่งเป็น Segment ในส่วนของ DATA Memory จะถูกวาง ณ ตำแหน่ง 30h และ 20h ตามลำดับ

ในส่วนของ Segment อื่นๆก็ใช้การสั่งในรูปแบบเดียวกัน ในกรณีที่เราไม่กำหนดค่าใดๆให้โปรแกรมจะจัดเรียงตำแหน่ง Segment ให้เองโดยจะเรียงส่วนที่มีขนาดเล็กก่อน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Dialog Debug



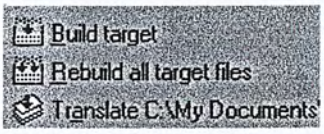
ภาพที่ 2.21 แสดงหน้าต่างการทำงานของ Dialog L51 Locate

ในส่วนของ Debug ในขณะนี้ ใช้ได้เฉพาะ ในส่วน Use Simulator เท่านั้น

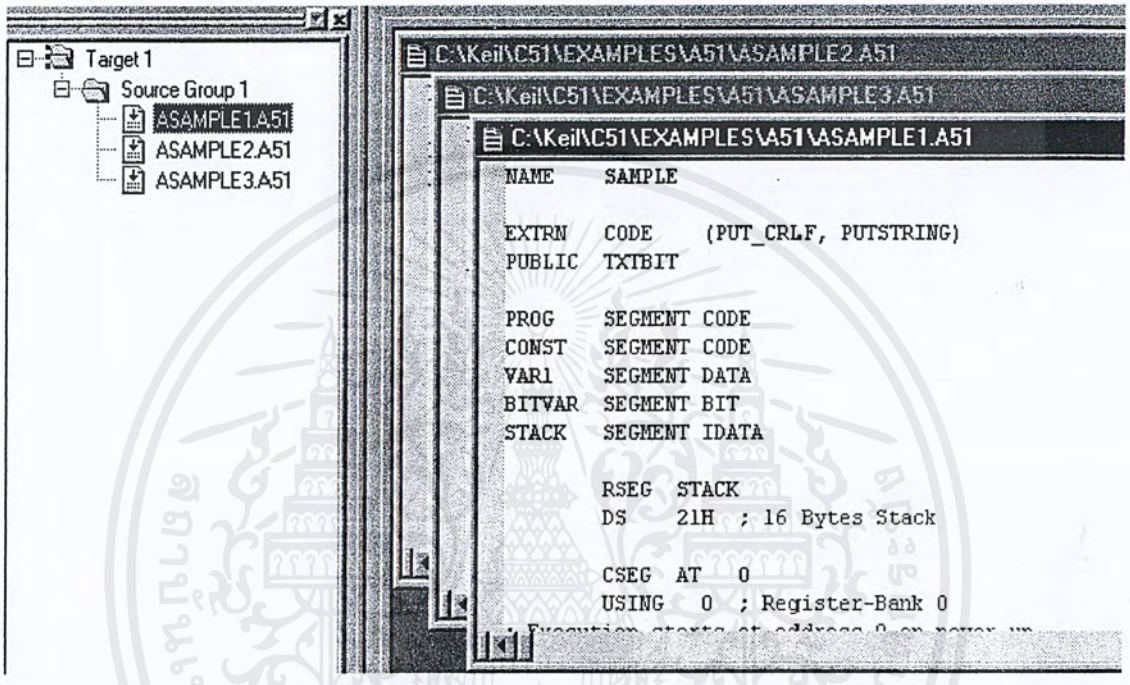
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.3 การทดสอบโปรแกรม

หลังจากที่ Add File เข้ามาที่ Source Group แล้ว และทำการ Build โปรแกรมแล้ว

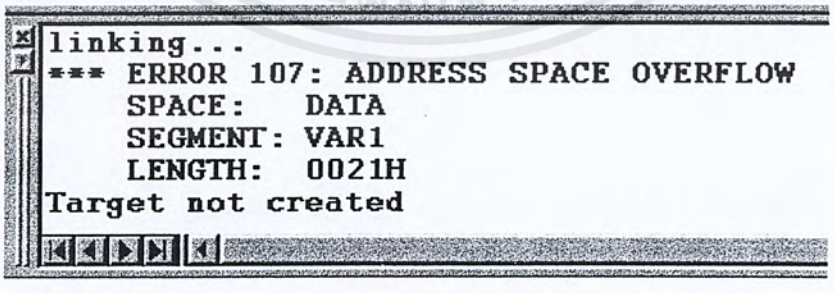


โปรแกรมก็จะสร้าง File ที่ใช้สำหรับการ Debug และแสดงรายละเอียด แต่ละ Segment ว่าอยู่ใน Memory ส่วนใดบ้าง ซึ่งได้แก่ File ที่มีนามสกุลดังนี้ เช่น .LST,.M51,.OBJ และ .HEX และหน้าต่างจะแสดงดังรูป



ภาพที่ 2.22 แสดงหน้าต่างการทดสอบโปรแกรม

ส่วนในกรณีที่โปรแกรมมี Error ก็จะแสดงรายละเอียดดังรูปข้างล่าง



ภาพที่ 2.23 แสดงหน้าต่างการทดสอบโปรแกรมในกรณีที่โปรแกรมมี Error

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อเราทดสอบโปรแกรมว่าไม่มี Error แล้วพร้อมที่จะ Debug ได้ โดยการใช้คำสั่ง Start/Stop Debug Session 

จากนั้นทำการ Run Program โดยที่เราสามารถที่จะ Run เป็น Step หรือ Run เป็นครั้งเดียวก็ได้ นอกจากนี้โปรแกรมยังแสดงหน้าต่างที่สำคัญซึ่งได้แก่

1.หน้าต่าง Project จะแสดงค่าของ Register ต่างๆ

2.หน้าต่าง Output จะแสดงตำแหน่งของ File ที่กำลัง Run อยู่ในขณะนั้น

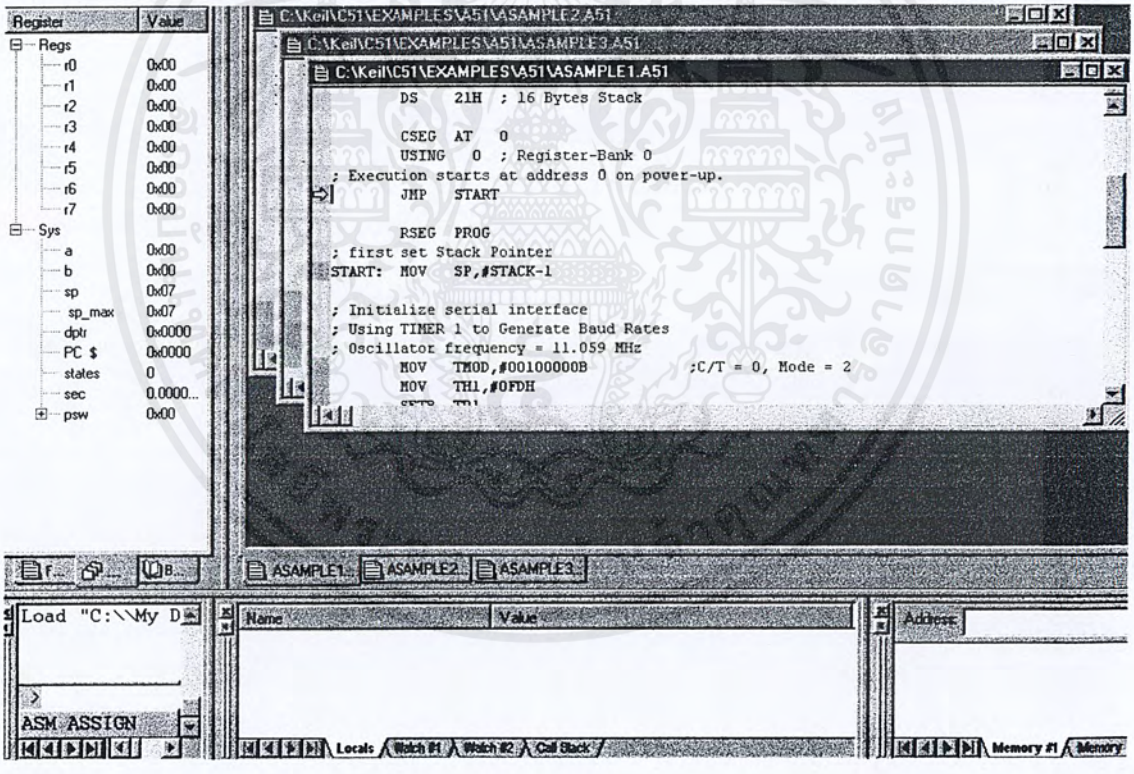
3.หน้าต่าง Memory จะแสดงค่าของ Address

4.หน้าต่าง Watch/call stack

5.Dialog Peripheral

6.Work space

ตัวอย่างรูปเมื่อเราเริ่ม Start Debug



ภาพที่ 2.24 แสดงหน้าต่างเมื่อเราเริ่ม Start Debug

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 โปรแกรมวิชวลเบสิก

เป็นที่ยอมรับในวงการกันดีว่าตั้งแต่วินโดวส์ 95 และวินโดว์ NT เป็นต้นมา วินโดวส์นั้นได้ช่วยให้คอมพิวเตอร์มีประสิทธิภาพในการทำงานสูงกว่าดอสดังนั้นในคอมพิวเตอร์ที่ใช้วินโดวส์ถ้าเขียนโปรแกรมสำหรับดอสเพื่อรันในดอสของวินโดวส์ก็เท่ากับไม่ได้ใช้วินโดวส์ที่มีอยู่เลย เพราะโปรแกรมจะลดความสามารถของวินโดวส์ให้เหลือเท่ากับคอมพิวเตอร์ที่ใช้ดอสเท่านั้น เพราะฉะนั้นสำหรับคอมพิวเตอร์ที่ใช้วินโดวส์ การเขียนโปรแกรมเพื่อรันในวินโดวส์จึงเป็นทางเดียวที่จะทำให้เกิดประโยชน์สูงสุด 2 ประการคือ

1. โปรแกรมสำหรับวินโดวส์จะนำความสามารถของวินโดวส์มาใช้ได้อย่างเต็มที่
2. การเขียนโปรแกรมสำหรับวินโดวส์ทำได้ง่ายกว่าการเขียนโปรแกรมสำหรับดอส

โปรแกรมหรือแอปพลิเคชั่นในวินโดวส์นั้นได้มีวิธีการเขียน 2 แบบ คือการเขียนโปรแกรมแบบวิชวลหรือวิชวลโปรแกรมมิ่ง (Visual Programming) และการเขียนโปรแกรมแบบนอนวิชวลหรือนอนวิชวลโปรแกรมมิ่ง (None Visual Programming) "วิชวล หมายถึง มองเห็นการเขียนโปรแกรม หมายความว่า การเขียนแบบนี้เราจะมองเห็นส่วนประกอบต่าง ๆ ของโปรแกรมนั้นทันที" ส่วนการเขียนโปรแกรมแบบนินวิชวลเป็นการเขียนโปรแกรมด้วยอักษรและเครื่องหมาย ซึ่งจะยังไม่เห็นส่วนประกอบของโปรแกรมในทันที แต่สามารถจะมองเห็นส่วนประกอบต่างๆ ของโปรแกรมได้ในภายหลังคือตอนรันโปรแกรม ดังนั้นการเขียนโปรแกรมแบบวิชวลจึงช่วยให้ผู้เขียนโปรแกรมสามารถเขียนได้สะดวกและรวดเร็วกว่า ในการทำงานด้านโปรแกรมมิ่งนั้นนับตั้งแต่ เริ่มต้นในการพัฒนาโปรแกรมต่าง ๆ นั้นจะมีโปรแกรมภาษาต่าง ๆ มากมายที่ใช้ในการพัฒนางานหรือสร้างเป็นโปรแกรมประยุกต์ขึ้นมาใช้งานสำหรับการทำงานในองค์กรหรือพัฒนาขึ้นมาใช้งานส่วนตัวแต่โดยส่วนใหญ่แล้ว โปรแกรมเหล่านี้จะเป็นโปรแกรมที่ต้องใช้ความจำเป็นเลิศเกือบทั้งสิ้น เนื่องจากโปรแกรมเหล่านั้นโดยส่วนใหญ่จะเป็นโปรแกรมประเภทการเขียนโดยการขี้คำสั่งเฉพาะต่าง ๆ ในการสร้างหรือควบคุมการทำงานของส่วนต่าง ๆ ที่ต้องการโดยเรียกการใช้งานโปรแกรมเหล่านี้ว่า "การโค๊ดดิ้ง" แต่สำหรับโปรแกรม Visual Basic 6.0 นั้นโปรแกรมพัฒนาที่การผสมผสานกันระหว่างการโค๊ดดิ้งและจับวาง (แกรกแอนด์ดรอป) นั่นก็คือในการทำงานนั้นสามารถที่จะกำหนดหรือสร้าง Object ต่าง ๆ โดยการใช้เครื่องมือต่าง ๆ ที่ตัวโปรแกรมมีมาให้โดย ไม่ต้องเขียนคำสั่งเพื่อสร้าง Object ต่างๆ เหล่านั้นขึ้นมาใช้งานและสามารถที่จะเขียนคำสั่งเพื่อใช้ในการสร้างเงื่อนไขพิเศษอื่น ๆ ที่ใช้ในการทำงานได้อีกด้วย

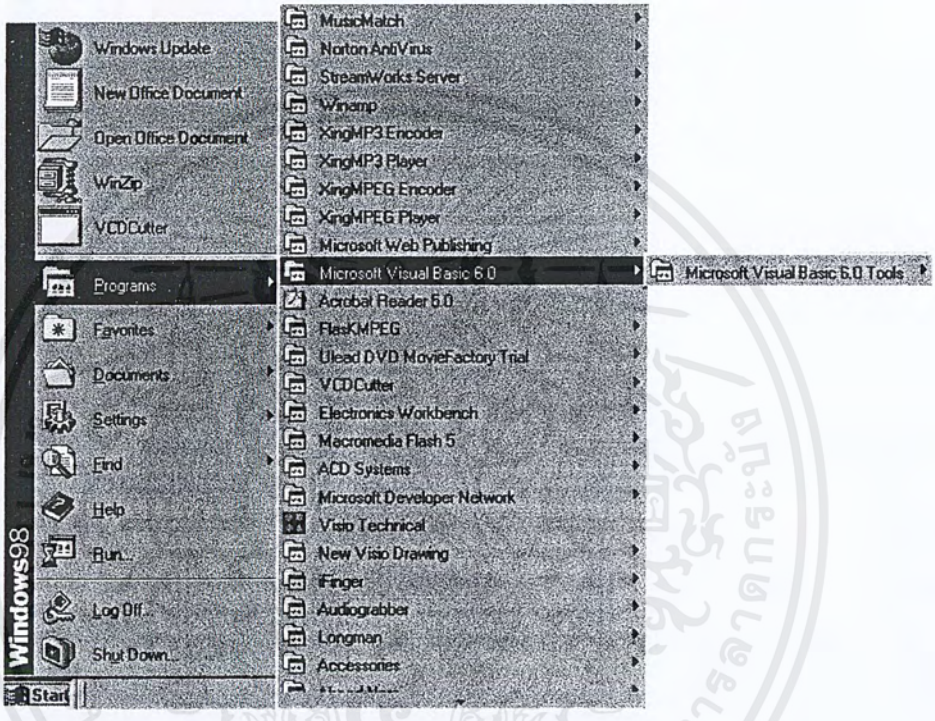
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเปิดโปรแกรมขึ้นมาใช้งาน

ในการเปิดโปรแกรมขึ้นมาใช้งานนั้นสามารถที่จะทำได้โดยวิธีการดังต่อไปนี้

- คลิกปุ่ม 

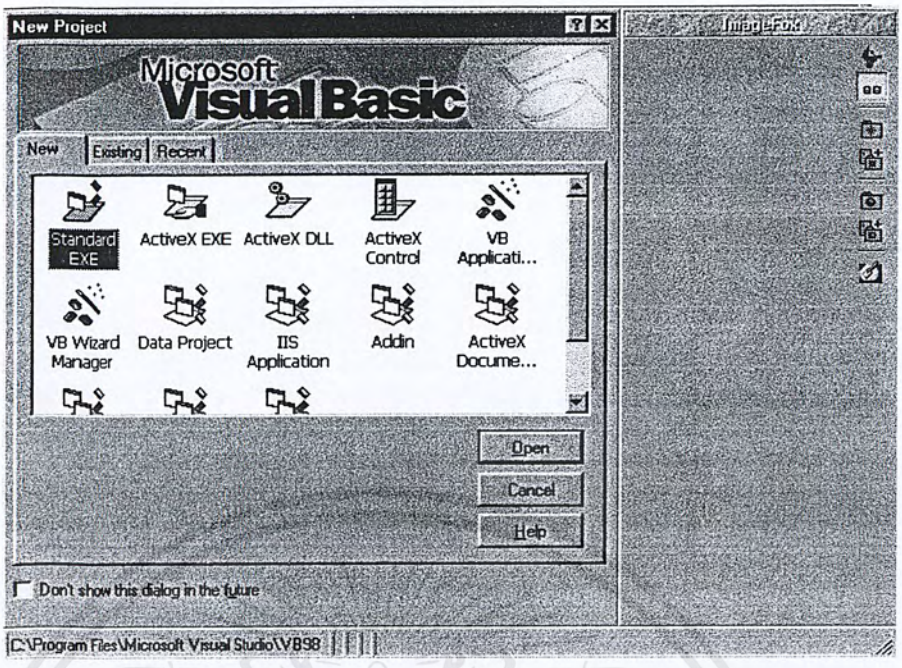
- เลือกเมนู Programs
- เลือกเมนูย่อย Microsoft Visual Studio 6.0
- คลิกไอคอน Microsoft Visual Basic 6.0



ภาพที่ 2.25 ขั้นตอนเปิดโปรแกรม Microsoft Visual Basic 6.0

เมื่อได้เปิดโปรแกรม Microsoft Visual Basic 6.0 ขึ้นมาแล้วนั้นจะปรากฏดังภาพที่ 2.26 และสำหรับส่วนประกอบในการทำงานต่าง ๆ ที่ใช้ในการทำงานนั้นจะได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.26 เมื่อเริ่มเปิดโปรแกรม Microsoft Visual Basic 6.0

จากรูปนั้นจะเห็นว่าจะปรากฏไดอะล็อกบล็อกซ์ New Project ขึ้นมาก่อนที่จะเข้าไปใช้งานโปรแกรมซึ่งในไดอาร์ี่ New Project นั้นจะประกอบด้วย 3 ส่วนด้วยกันคือ

New

ส่วนนี้เป็นส่วนที่ใช้ในการเลือกสร้างงานใหม่ขึ้นมา ซึ่งจะมีรูปแบบของการสร้างงานให้เลือกใช้หลากหลายรูปแบบด้วยกัน

Existing

ส่วนนี้จะเป็นส่วนที่ใช้ในการเปิดงานที่ได้สร้างขึ้นจากโปรแกรม Visual Basic ในเวอร์ชั่นต่าง ๆ ที่มีอยู่แล้วซึ่งเก็บไว้ในที่ต่าง ๆ ที่ต้องการขึ้นมาใช้งาน ได้ตามต้องการ

Recent

ในส่วนนี้จะเป็นการเลือกเปิดไฟล์ที่ได้เคยเปิดขึ้นมาใช้งานแล้วปิดลงไป ซึ่งในส่วนนี้จะปรากฏชื่อไฟล์ต่าง ๆ ที่ได้เคยเปิดขึ้นมาใช้งานแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.1 คุณสมบัติและข้อดีของ Visual Basic

ในVisual Basic มีคุณสมบัติที่ดีหลายประการกล่าวคือ ในการเขียนโปรแกรมแบบเดิมนั้น จะต้องมานั่งออกแบบหน้าจอระบุตำแหน่งการแสดงผลคิดหาขั้นตอนการทำงานและอื่น ๆ จากนั้น จึงทำการเขียนโปรแกรม แต่ใน Visual Basic จะใช้หลักการของภาพและการมาองเห็น โดยเริ่มจากออกแบบวินโดวส์ย่อยหรือที่ใน Visual Basic เรียกว่า ฟอร์ม ในฟอร์มจะประกอบด้วยสิ่งต่างๆ ที่จะทำงานด้วยหรือเรียกว่าเป็น Object เช่น ข้อความ, ช่องรับข้อความ, Scroll Bar,หรือปุ่ม(Button) เมื่อกำหนดสิ่งต่าง ๆ เหล่านี้คราบตามต้องการแล้วจึงระบุว่าองค์ประกอบแต่ละอย่างทำงานอย่างไร โดยเขียนโปรแกรมย่อยใส่เข้าไปกับ Object เหล่านี้ที่ต้องทำแบบนี้ ก็เพราะว่าการทำงานใน Windows เป็นแบบที่เรียกว่า Event-Driven คือขึ้นกับเหตุการณ์ (Even) การเขียนโปรแกรมแบบเดิมคือ สั่งงานตามลำดับยุ่งยากมาก

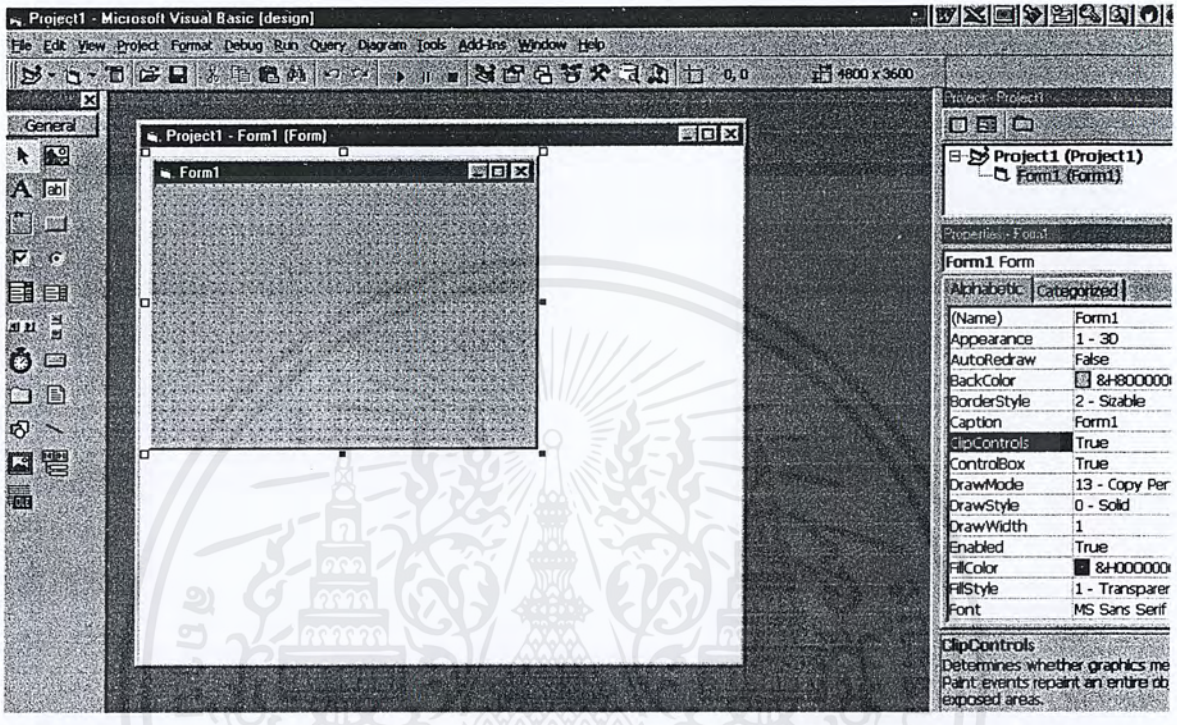
Visual Basicนี้ยังคงไว้ด้วยข้อดีต่าง ๆ ของ Microsoft Quick Basic และยังมีการเพิ่มเติมคุณสมบัติหลากหลายที่จะสนับสนุนให้ตัวมันเองเป็นโปรแกรมพื้นฐาน ซึ่งใช้สำหรับในการพัฒนาโปรแกรมบน Microsoft Windows อีกด้วย ตัวอย่างเช่น กราฟฟิกเอาต์พุทที่สามารถถูกส่งออกไปยังส่วนต่าง ๆ ของวินโดวส์หรือแม้กระทั่งส่งไปยังเครื่องพิมพ์ สามารถเลือกสีสำหรับงานกราฟฟิกได้มากกว่า 16 ล้านเฉดสี (โดยวินโดวส์จะจัดการแสดงผลกราฟฟิกนั้นตามที่ต้องการ หรือจะลดลงมาได้เท่าที่ฮาร์ดแวร์ของเครื่องนั้น ๆ จะสนับสนุนในการแสดงผลได้) โดยที่ไม่ต้องกังวลในส่วนนี้ว่าจะมีกระบวนการในการจัดการอย่างไรไม่ว่าในตอนนี้หรือต่อไปในอนาคต

ข้อดีเหนือ Quick Basic อีกอย่างหนึ่งก็คือ การจัดการตัวแปรใน Visual Basic มีกฎเกณฑ์ซึ่งง่ายในการเข้าใจและจดจำ เพราะว่าโปรแกรมตัวนี้ถูกพัฒนาให้ง่ายและมีประสิทธิภาพโดยที่โปรแกรมของ Visual Basic จะประกอบไปด้วยไฟล์ 2 แบบ คือ From และ Module ยกเว้นเมื่อ Declare globally ที่อื่น ตัวแปรและค่าคงที่ในโปรแกรมย่อยและฟังก์ชั่นนั้นจะเป็น Local สำหรับกระบวนการที่เกิดขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.2 ส่วนประกอบต่าง ๆ ที่ในการทำงาน

เมื่อทำการเปิดโปรแกรมขึ้นมาใช้งานแล้วนั้นจะปรากฏส่วนประกอบต่าง ๆ ของโปรแกรม ควรทราบก่อนที่จะเริ่มใช้งานโปรแกรมในการสร้างงานต่าง ๆ เพราะจะทำให้เลือกใช้เครื่องมือหรืออุปกรณ์ในส่วนต่าง ๆ ได้สะดวกและรวดเร็วมากยิ่งขึ้น



ภาพที่ 2.27 ส่วนประกอบต่าง ๆ ของโปรแกรม Visual Basic 6.0

Title Bar

จะเป็นส่วนที่แสดงชื่อของ Application ที่กำลังเปิดใช้งานอยู่ขณะนั้น

Minimize Button

เป็นปุ่มที่ใช้สำหรับคลิกเพื่อหดหรือซ่อนส่วนของ Window หรือหน้าต่างทำการไว้ โดยจะปรากฏหรือแสดงที่หน้าจอเฉพาะส่วนของ Title Bar ของหน้าต่างดังกล่าวไว้และหากต้องการแสดงหรือ Show หน้าต่างอีกครั้งให้ใช้เมาส์คลิกซ้ำที่ปุ่มเดิม ซึ่งปุ่มดังกล่าวจะอยู่ทางด้านขวาของ Title Bar ของหน้าต่างนั้น ๆ

Maximize Button or Restore Button

จะเป็นปุ่มที่ใช้สำหรับคลิกเพื่อขยายขนาดใหญ่ขึ้นจนเต็มหน้าจอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Close Button

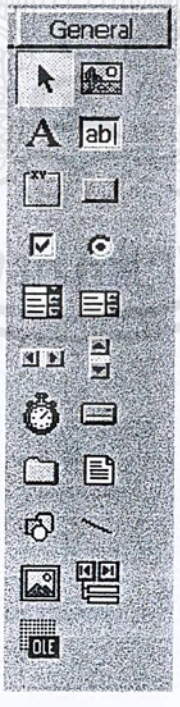
เป็นปุ่มสำหรับคลิกเพื่อปิดหรือจบการทำงานของหน้าต่างนั้น ๆ

Menu bar

เป็นแถบของเมนูหลักที่บรรจุคำสั่ง ย่อยหลายคำที่ใช้ในการทำงานไว้ สามารถเขียนใช้งานเมนูได้ โดยการใช้เมาส์คลิกที่ชื่อของเมนูที่ต้องการบน Menu Bar หรืออาจเรียกเมนูดังกล่าวได้โดยการกดปุ่มที่คีย์บอร์ดดังนี้กดปุ่ม Alt ค้างไว้พร้อมกับกดตัวอักษรที่ขีดเส้นใต้ของชื่อเมนูที่ต้องการเปิดดังตัวอย่างต่อไปนี้ หากต้องการใช้คำสั่งย่อยที่อยู่ในเมนู และที่ชื่อของเมนูบนจะมีเส้นขีดไว้ที่ใต้ตัวอักษร ดังนั้นในการเรียกใช้คำสั่งย่อยในเมนูดังกล่าวจำเป็นต้องเปิดเมนูก่อนโดยการกดปุ่มที่คีย์บอร์ดดังนี้ Alt+F

Tool Bar

เป็นส่วนที่ปรากฏอยู่ด้านใต้ของเมนูบาร์ ซึ่งเป็นส่วนที่รวบเอาไอคอนเล็ก ๆ มากมายเอาไว้ ซึ่งแต่ไอคอนเหล่านี้ก็เปรียบเสมือนคำสั่งหนึ่ง ๆ ของเมนู ดังนั้นไอคอนในส่วนนี้จึงถูกออกแบบเพื่อให้การเลือกใช้คำสั่งของเมนูรวดเร็วและมีลักษณะที่สื่อความหมายกับผู้ใช้มากขึ้น



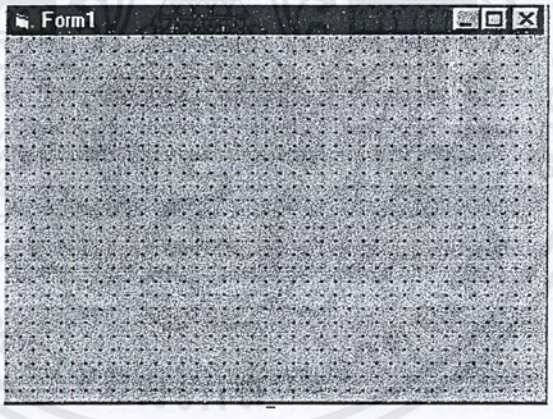
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Tool Box

เป็นหน้าต่างที่รวบรวมเอาเครื่องมือต่าง ๆ สำหรับใช้ในขณะออกแบบฟอร์ม ซึ่งการรวมเอาเครื่องมือตัวใดให้กับฟอร์มผู้ใช้สามารถทำได้โดยการคลิ๊กที่คอนโทรลนั้น เพื่อเลือกแล้วนำมาวางลงในฟอร์มโดยวิธีการจากแล้ววาง หรือโดยวิธีการดับเบิลคลิ๊กที่ไอคอนคอนโทรลในทูลบ๊อกซ์ก็ได้ ซึ่ง Visual Basicก็จะทำการวางคอนโทรลลงในฟอร์มปัจจุบันโดยอัตโนมัติ

Form

ในการแสดงหน้าต่างเพื่อที่จะสื่อการทำงานกับผู้ใช้ ทั้งนี้เนื่องจาก Visual Basic ได้รับการออกแบบให้โปรแกรมเมอร์สามารถนำมาใช้ออกแบบแอพพลิเคชั่นในลักษณะของการสื่อด้วยรูป ดังนั้นการสื่อการทำงานต่างๆ ระหว่างแอพพลิเคชั่นกับผู้ใช้จะต้องกระทำผ่านฟอร์ม โดยในโปรเจคหนึ่งๆ ซึ่งสามารถมีได้หลายฟอร์มและภายในฟอร์มต่างๆนั้น ก็จะถูกใช้ในการบรรจุคอนโทรลต่างๆ เช่น text box, picture box หรือ image เอาไว้ ดังนั้นฟอร์มจึงทำหน้าที่เป็นตัวบรรจุ (container) และฟอร์มก็ยังเป็นออบเจ็กต์ตัวหนึ่งของ Visual Basicที่อนุญาตให้ผู้ใช้แก้ไขคุณสมบัติได้ในขณะออกแบบจากหน้าต่างๆ ซึ่งคุณสมบัติและสามารถควบคุมพฤติกรรมต่างๆ ของฟอร์มได้โดยผ่านทางวิธีเช่นเดียวกับออบเจคอื่น ๆ

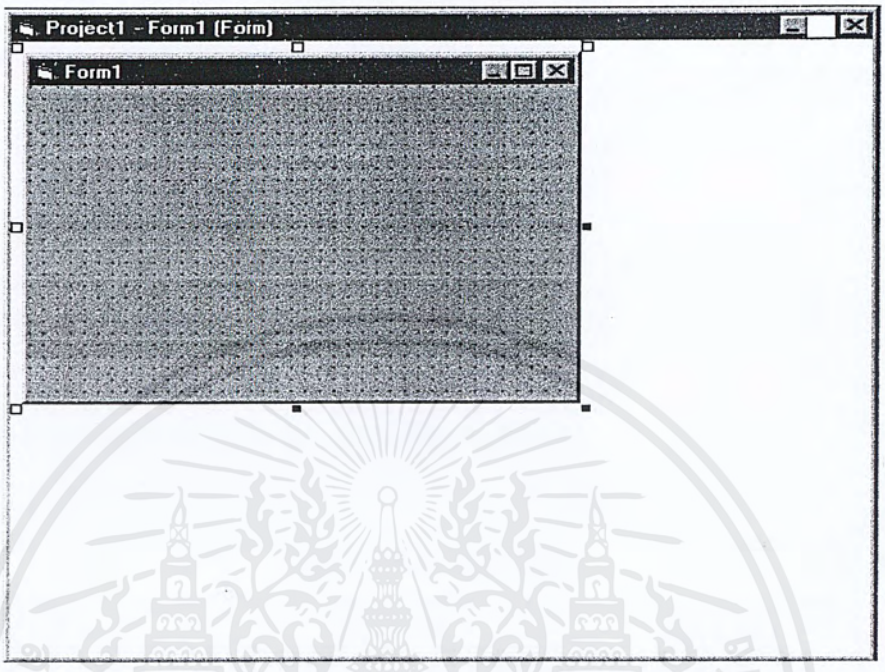


ภาพที่ 2.28 ส่วนของ Form

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Project Container Window

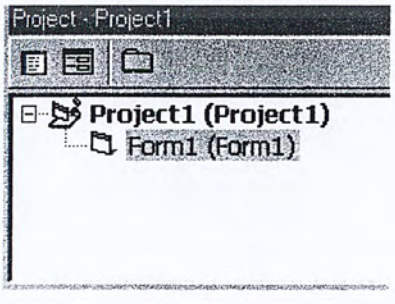
คือบริเวณทั้งหมดของฉากด้วยหลังที่เป็นสีขาว ของ From ซึ่งใช้สำหรับเป็นพื้นที่หรือหน้าต่างๆ ที่ใช้บรรจุ Form หรือ สร้าง Form



ภาพที่ 2.29 ส่วนประกอบต่าง ๆ ของProject Container Window

Project Explore Window

เป็นหน้าต่างที่รวบรวมรายชื่อของฟอร์ม โมดูลไฟล์ Custom Control (Class Module) หรือไฟล์ทรัพยากร (resource file) สำหรับการสร้างแอพพลิเคชันหนึ่ง ๆ ซึ่งการรวมเอาไฟล์เหล่านี้เข้ามาด้วยกันเพื่อสร้างแอพพลิเคชั่นภายใต้ Visual Basic/win เรียกว่า โปรเจค (project) หรือโครงการซึ่งโดยปกติ Visual Basic จะจัดเก็บไฟล์โปรเจคโดยจะมีนามสกุล .MAK โดยจะมีการจัดเก็บแยกออกจากไฟล์อื่น ๆ โดยเด็ดขาด

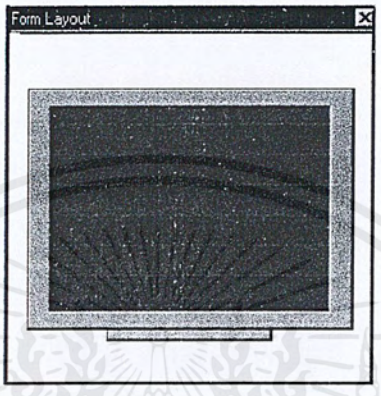


ภาพที่ 2.30 ส่วนประกอบต่าง ๆ ของ Project Explore Window

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Form Layout Windows

เป็นส่วนที่ใช้กำหนดตำแหน่งของ Form ที่ต้องการแสดงบนหน้าจอและบนหน้าต่างที่ทำงานนอกจากนี้ในการกำหนดตำแหน่งของ Form จะสามารถเลือกกำหนดได้จากการกำหนดค่าตัวแปรต่างๆ หรือคุณสมบัติต่างๆ ที่เกี่ยวกับการจัดวางตำแหน่งได้ในส่วนของ Properties ได้เช่นกัน



ภาพที่ 2.31 ส่วนประกอบต่าง ๆ ของ Form Layout Windows

Pop Up Menu or Shortcut Menu

เป็นส่วนของเมนูคำสั่งต่าง ๆ ที่จะปรากฏหรือแสดงเมื่อมีการกดปุ่มด้านขวาของเมาส์หรือที่เรียกว่า คลิกขวา

ภาพที่ 2.32 ส่วนประกอบต่าง ๆ ของ Pop Up Menu or Shortcut Menu

Tool Tips

เป็นส่วนที่ใช้บอกหรือแสดงชื่อของเครื่องมือหรือปุ่มต่าง ๆ ที่ตำแหน่งของ Mouse Pointer ชี้ อยู่ในขณะนั้น

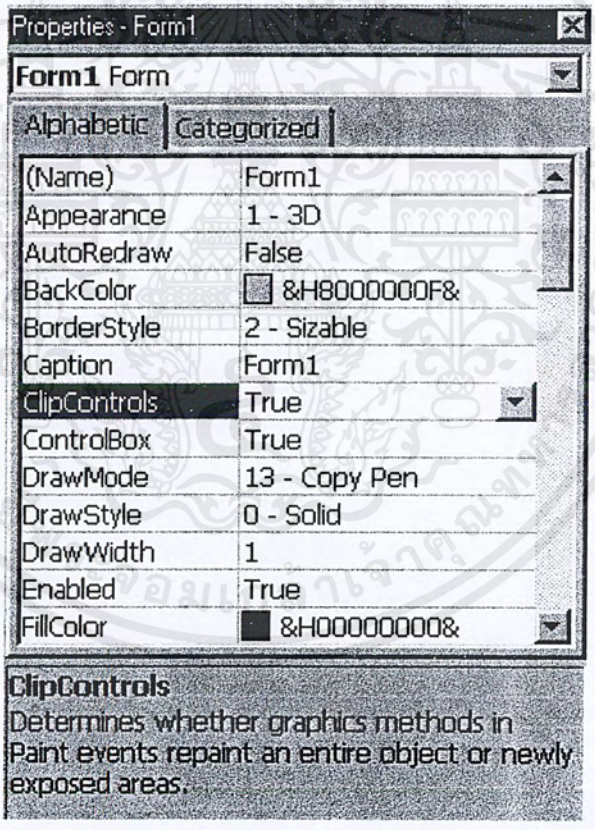
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Pointer

เป็นเครื่องมือที่ใช้ในการเลือกเมนูคำสั่งและเครื่องมือต่าง ๆ

Properties Windows

จะเป็นหน้าต่างที่รวบรวมคุณสมบัติทั้งหมดของฟอร์มหรือคอนโทรลเอาไว้ ซึ่งคุณสมบัติทั้งหมดที่ปรากฏในหน้าต่างนี้จะเป็นคุณสมบัติที่ผู้ใช้สามารถกำหนดค่าได้ในขณะออกแบบ เช่น Caption, Text, Left, Top หรือ Back Color เป็นต้น เมื่อผู้ใช้ทำการแก้ไขค่าคุณสมบัติต่างๆ ในหน้าต่างคุณสมบัตินี้ก็จะส่งผลต่อคอนโทรลตัวนั้นทันที ซึ่งบางคุณสมบัติสามารถแสดงให้เห็นการเปลี่ยนแปลงคุณสมบัติได้ทันที เช่น Left, Top หรือ Fore Color เป็นต้น ส่วนคุณสมบัติบางอย่าง จะแสดงผลให้เห็นก็ต่อเมื่อผู้อ่านมีการรับแอพพลิเคชั่นเท่านั้น เช่น Enabled หรือ Visible เป็นต้น



ภาพที่ 2.33 ส่วนประกอบต่าง ๆ ของ Properties Windows

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.3 คอนโทรลของVB/Win (custom control)

คอนโทรลเป็นเครื่องมืออย่างหนึ่งที่ช่วยให้การสร้างแอพพลิเคชั่นด้วย Visual Basic สามารถกระทำได้ง่ายและรวดเร็วยิ่งขึ้น ซึ่งในความเป็นจริงคอนโทรลก็คือไฟล์ไดนามิกลิงค์ไลบรารีทั่ว ๆ ไปของวินโดวส์โดยทั่วไปมักจะถูกเขียนด้วยภาษาซี หรือ Visual C++ เพียงแต่ไฟล์เหล่านี้จะมีนามสกุล .VBX หรือ .OCX ซึ่งเรียกว่า custom control เนื่องจากคอนโทรลเหล่านี้ได้รับการออกแบบเป็นพิเศษเพื่อให้สามารถใช้งานร่วมกับ Visual Basic ได้อย่างไม่มีปัญหาแล้ว ดังนั้นไฟล์เหล่านี้จะถูกย้ายไปจัดเก็บไว้ในไดเรคทอรีย่อย system ของวินโดวส์และไม่คอมแพตทิเบิลกับเวอร์ชันที่กำลังติดตั้งใหม่ ไฟล์ custom control ที่มีอยู่ในไดเรคทอรีย่อย system ของวินโดวส์นั้นก็จะถูกย้ายไปจัดเก็บไว้ในไดเรคทอรีของ Visual Basic เวอร์ชันนั้น ๆ แทนผู้ใช้สามารถโหลดไฟล์คอนโทรล มารวมกับโปรเจคได้ด้วยการใช้คำสั่ง Add File ในเมนู File แต่ถ้าหากโปรเจคไม่มีการใช้คอนโทรลใด ๆ ก็ให้ผู้ใช้ลบคอนโทรลตัวนั้น ๆ ออกจากโปรเจคซึ่งสามารถทำได้ด้วยการใช้คำสั่ง Remove Fileในเมนู File ทั้งนี้เพื่อจะให้การโหลดโปรเจคของ Visual Basicสามารถกระทำได้รวดเร็วยิ่งขึ้น เมื่อผู้ใช้นั้นได้มีการสร้างแอพพลิเคชั่นลงในไดเรคทอรีย่อย system ของวินโดสว์ของเครื่องผู้ใช้ด้วย (วิธีการสร้างแผ่นดิสก์สำหรับติดตั้งแอพพลิเคชั่น สามารถทำได้โดยใช้ Setup Wizard หรือชุดโค้ดที่มีในไดเรคทอรีย่อย System ในไดเรคทอรีของ Visual Basic)

คอนโทรลแต่ละตัวในความเป็นจริงก็คือโค้ดชุดหนึ่ง (Library) ที่สามารถทำงานได้เฉพาะด้านตามที่กำหนดและตัวคอนโทรลก็มีคุณสมบัติเฉพาะตัวที่ผู้อ่านสามารถแก้ไขได้นอกจากนี้เมื่อมีการกระทำใด ๆ กับคอนโทรล เช่น การคลิ๊ก การเปลี่ยนค่า เป็นต้น เหตุการณ์เกี่ยวข้องของคอนโทรลก็จะถูกเรียก (Event-Driven) ซึ่งผู้ใช้สามารถที่จะเขียนโค้ดเพื่อควบคุมการทำงานเมื่อเกิดเหตุการณ์ได้โดยการเขียนโค้ดลงในส่วนโพรซีเดอร์เหตุการณ์ (Event Procedure) ซึ่งจะมีรายละเอียดคร่าว ๆ ของคอนโทรลแต่ละตัวดังนี้

Pointer เป็นเคอร์เซอร์ ที่ใช้สำหรับเลือก เลื่อนตำแหน่ง หรือขยายขนาดของฟอร์ม หรือคอนโทรล ซึ่งคอนโทรลตัวนี้เป็นเครื่องมือของเอนไวรอนเมนท์ ดังนั้นผู้ใช้จึงไม่สามารถนำไปใช้ในแอพพลิเคชั่น

Picture box ตัวคอนโทรลนี้ใช้สำหรับแสดงบิตแม็พ ไอคอน ไฟล์ดาต้า วิธีการใช้ในการแสดงผลกราฟฟิกเช่น Pset หรือ Line เป็นต้น หรือแสดงข้อความใดก็ได้ นอกจากนี้สามารถใช้บรรจุคอนโทรลอื่น ๆ ได้อีกด้วย (container)

Labelใช้แสดงข้อความที่ไม่สามารถแก้ไขได้โดยผู้ใช้ ยกเว้นแก้ไขโดยการเขียนโค้ดเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้


Command button ปุ่มคำสั่งซึ่งผู้ใช้สามารถเลือกได้โดยการคลิกที่ปุ่ม


Frame Frame ใช้เป็นตัวบรรจุ (Container)คอนโทรลอื่น ๆนอกจากนี้ ยังใช้เป็นตัวรวมกลุ่ม (Group)คอนโทรลอื่น ๆ เช่น Option button เป็นต้น


Text box กรอบสี่เหลี่ยมสำหรับกรอกข้อความต่าง ๆ ซึ่งจะถูกใช้เป็นกล่องสำหรับตัวอักษรที่ถูกคีย์โดยผู้ใช้


Check box ปุ่มสำหรับเลือกสถานะถูกหรือผิดหรือสถานะใช่หรือไม่ใช่


Option button ปุ่มที่ถูกใช้รวมกันเป็นกลุ่มเพื่อให้ผู้ใช้ได้ใช้เป็นตัวเลือก (อย่างใดอย่างหนึ่ง)


Combo box คอนโทรลที่รวมเอาความสามารถของ text box และ list box เอาไว้ด้วยกัน


List box คอนโทรลที่ใช้ในการแสดงผลรายการของข้อความเพื่อให้ผู้ใช้สามารถเลือกได้


Horizontal scroll bar แถบเลื่อนในแนวนอนซึ่งสามารถกำหนดค่าในช่วงที่ต้องการเลือกได้


Vertical scroll bar แถบเลื่อนในแนวตั้งซึ่งผู้ใช้สามารถกำหนดค่าในช่วงที่ต้องการเลือกได้


Time นาฬิกาที่นี้ถูกใช้ในการสร้างเหตุการณ์โดยที่อ้างอิงกับช่วงเวลา ซึ่งสามารถเทียบเคียงได้กับประโยค In Timer ของ Quick BASIC

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.4 ภาษาโปรแกรมของ Visual Basic

2.3.4.1 ตัวแปรและค่าคงที่

ตัวแปรใน Visual Basic มีความยาวได้ถึง 40 ตัวอักษร สามารถตั้งชื่อตัวแปรโดยผสมจากตัวอักษร ตัวเลขและ underscore (_) แต่ตัวแรกต้องเป็นอักษรเท่านั้นและห้ามตั้งชื่อตรงกับคำที่สงวนไว้ (Reserved word) ซึ่งได้แก่ฟังชันก์และคำสั่งต่าง ๆ ตัวแปรแบ่งออกเป็น 7 ชนิด ดังตารางที่ 2.3

ตารางที่ 2.3 ตัวแปรและค่าคงที่ในโปรแกรม Visual Basic

ชนิด	คำอธิบาย	ตัวอักษรบอกชนิด	ขอบเขต
Integer	2- byte integer	%	-32,768 จนถึง 32,767
Long	4- byte integer	&	-2,147,678จนถึง 2,147,483,647
Single	4- byte floating-point Number	!	-3.402823E38จนถึง-1.401298E-45
Double	8- byte floating-point Number	#	-1.797693134862232222D308จนถึง –4.94065645841247D-324 (ค่าลบ) 4.94065645841247D-324 จนถึง 1.797693134862232222D308
Currency	8- byte Number with Fixed decimal point	@	-922337203668547705808จนถึง 92233720366854770 5808
String	String of characters	$	0 จนถึงประมาณ65,000 ตัวอักษร
Variant	Date/time}floating-Pint number, or string		Date valued : วันที่ 1 มากราคม ปี ค.ศ. 0000 ถึงวันที่ 31 ธันวาคม ปี ค.ศ. 9999

Integer

ไว้สำหรับเก็บค่าจำนวนเต็มในช่วง –32,768 ถึง 32,767 ตัวแปรชนิดนี้สามารถคำนวณได้เร็วมาก แต่มีข้อจำกัดด้านค่าที่เก็บได้ ซึ่งอยู่ในขอบเขตแคบกว่าแบบอื่น ตัวแปรชนิดนี้ระบุด้วย % เช่น A%

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Long

เป็นค่าจำนวนเต็มเช่นเดียวกับชนิด Integer แต่ขยายช่วงไปเป็น –2,147,483,648 ถึง 2,147,483,647 ใช้เนื้อที่ 4 ไบต์และใช้ตัวอักษร & เป็นตัวระบุชนิด เช่น A&

Single

ใช้เนื้อที่ในการเก็บ 4 ไบต์เก็บค่าตัวเลขที่เป็นทศนิยม ซึ่งไม่ต้องการความละเอียดมากนัก (-3.102823E38 ถึง –1.401298ฎ-45) ใช้ ! ในการระบุชนิด เช่น A!

Double

คล้ายกับชนิด Single แต่สามารถเก็บค่าได้มากกว่า และใช้เนื้อที่เก็บเป็น 2 เท่า ของ Single (8 ไบต์) อย่างไรก็ตามก็ใช้เวลาการคำนวณมากกว่าด้วย ระบุชนิดด้วย # เช่น A#

Currency

ใช้เก็บค่าทางการเงิน โดยเฉพาะเป็นแบบทศนิยมคงที่ 4 ตำแหน่ง ใช้เนื้อที่ 8 ไบต์ การระบุถึงชนิดใช้ตัวอักษร @ เช่น A@

String

ใช้เก็บข้อความหรือตัวอักษรซึ่งไม่สามารถนำไปคำนวณได้ ความยาวของข้อความคือ 65,5000 ตัวอักษร เนื้อที่ที่ใช้เก็บขึ้นอยู่กับขนาดของข้อความที่เก็บ ระบุชนิดด้วย $ เช่น a$

Variant

ตัวแปรชนิดนี้เพิ่มเริ่มมีใช้ใน Visual Basic 2.0 ถูกออกแบบมาใช้เก็บค่าต่าง ๆ กัน โดยสามารถเก็บค่าชนิดใดก็ได้ 6 ชนิดข้างต้น ไม่ว่าจะเป็นตัวเลข ตัวอักษร หรือวันที่ ข้อมูลชนิดนี้มีประโยชน์ในกรณีที่ต้องการประกาศตัวแปร ซึ่งอาจเก็บค่าได้หลายแบบ โดยที่ไม่สามารถระบุได้แน่นอนว่าเป็นแบบใด ปัญหาคือจะทำงานช้ากว่าตัวแปรแบบอื่น เพราะ Visual Basic ต้องมีการตรวจสอบชนิดของตัวแปรประเภทนี้ทุกครั้งก่อนจะใช้งาน

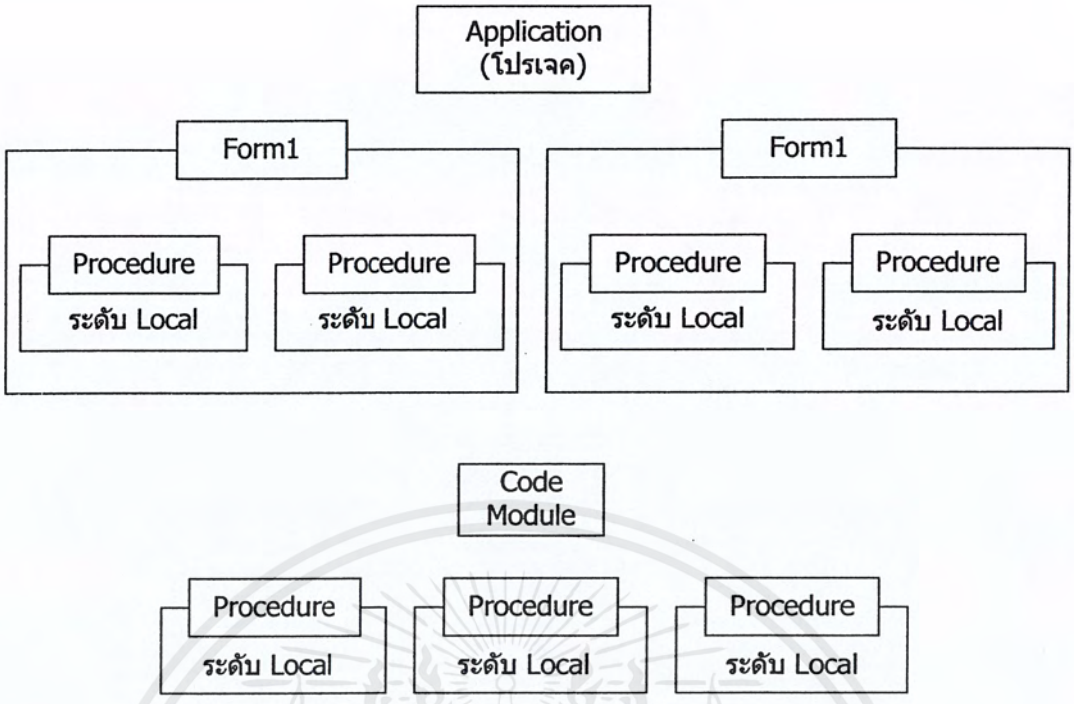
2.3.5 ขอบเขตของตัวแปร

ทั้งตัวแปรและค่าคงที่ที่ได้กล่าวถึงตอนต้น โดยปกติจะมีขอบเขตของการใช้งาน หรือใช้งานได้ในแต่ละส่วนของโปรแกรมเท่านั้น ซึ่งขอบเขตของตัวแปร (Scope) นี้แบ่งออกได้เป็น 3 ระดับ คือ

2.3.5.1 Local

ตัวแปรชนิดนี้จะเป็นที่รู้จักกันมาก ซึ่งใช้ได้ในระดับโพรซีเดอร์หรือฟังก์ชันที่ประกาศใช้ตัวแปรนั่นเอง ส่วนมากมักจะใช้กับการประกาศตัวแปรสำหรับการคำนวณหร้อจะใช้เก็บค่าชั่วคราว ตัวแปรในระดับ Local จะถูกยกเลิกไปเมื่อการทำงานของโพรซีเดอร์หรือฟังก์ชันนั้นสิ้นสุดลง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.34 แสดงขอบเขตของการใช้ตัวแปรและค่าคงที่ในระดับต่าง ๆ

2.3.5.2 Module

ตัวแปรในระดับโมดูลสามารถอ้างอิงถึง สามารถใช้ร่วมกันได้หลายฟังก์ชันหรือหลายโพรซีเดอร์ซึ่งอยู่ในโมดูลนั้น ไม่ว่าจะเป็นในโมดูลของโปรแกรมที่ประกอบด้วยรูทีนย่อย ๆ หลาย ๆ รูทีน หรือในโมดูลของฟอร์มซึ่งอยู่ในรูทีนสำหรับทำงานกับ Object ในฟอร์ม

2.3.6 Global

ในตัวแปรระดับ Global นี้สามารถจะอ้างอิงได้จากจุดใด ๆ ก็ตามใน Application ทุกรูทีนหรือทุกส่วนจะสามารถนำไปใช้ได้การประกาศใช้จะเหมือนกับระดับโมดูล คือประกาศค่าคงที่ในส่วน Declaration เพียงแต่ว่าจะต้องมีคำ "Global" นำหน้าชื่อตัวแปร สำหรับในการประกาศค่าคงที่ในระดับ Global ก็ใช้คำว่า "Global"นำหน้าเช่นกัน

```
Global A As Integer
Global Const BAT_Rate = 70/100
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.7 โอเปอร์เรเตอร์

โอเปอเรเตอร์ คือสิงที่จะก่อให้การกระทำเพื่อให้ได้ผลลัพธ์ออกมาพบโอเปอร์เรเตอร์ได้ในนิพจน์ทั่ว ๆ ไป ซึ่งแบ่งออกเป็นกลุ่มโอเปอร์เรเตอร์ทางคณิตศาสตร์ การเปรียบเทียบ และทางตรรกะ ดังตารางที่ 2.4 โดยสัญลักษณ์ในวงเล็บคือ โอเปอร์เรเตอร์ที่ใช้ในนิพจน์

ตารางที่ 2.4 แสดงกลุ่มของโอเปอร์เรเตอร์

ทางคณิตศาสตร์	การเปรียบเทียบ	ทางตรรกะ
ยกกำลัง ( )	เท่ากัน (=)	Not
ค่าลบ (-)	ไม่เท่ากัน (≠)	And
คูณ,หาร (*,/)	น้อยกว่า (<)	Or
หารจำนวนเต็ม (\)	มากกว่า (>)	Xor
Module (Mod)	น้อยกว่าหรือเท่ากับ (<=)	Eqv
บวก,ลบ (+,-)	มากกว่าหรือเท่ากับ(>=)	Imp
เชื่อมข้อความ (String)(& หรือ +)	เหมือน (Like)	Is

2.3.8 ประโยคคำสั่ง

เป็นประโยคที่ส่งให้เกิดการทำงานตามที่ผู้เขียนโปรแกรมต้องการ ประโยคคำสั่งหลัก ๆ ได้แก่ ประโยคสำหรับกำหนดค่า ประโยคหมายเหตุ ประโยคประกาศ และประโยคเพื่อควบคุมลำดับการทำงานของโปรแกรมนอกเหนือจากนี้จะเป็นประโยคเพื่อการทำงานอื่น ๆ เช่น ทำงานทางกราฟฟิก เป็นต้น

2.3.8.1 ประโยคสำหรับกำหนดค่า

ใช้สัญลักษณ์ " =" เพื่อเป็นการกำหนดค่าทางขวามือของเครื่องหมายให้กับตัวแปรหรือคุณสมบัติ (Property) ทางซ้ายมือ เช่น

```
A = 5
Text1.Text = "Hi!"
```

2.3.8.2 ประโยคหมายเหตุ

สำหรับการใส่หมายเหตุต่อคำอธิบายการทำงานของโปรแกรม ซึ่งการใส่หมายเหตุในโปรแกรมนี้ถือว่าเป็นเรื่องสำคัญเนื่องจากสิ่งนี้จะช่วยป้องกันการหลงลืมของคนเขียนโปรแกรมเองหรืออธิบายให้ผู้อ่านซึ่งต้องมาใส่การทำงานของโปรแกรมได้ง่ายขึ้น มิฉะนั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะพบว่าวันหนึ่งมีผู้ไม่เข้าใจโปรแกรมส่วนนี้ทำงานอย่างไรมีไว้เพื่ออะไร ซึ่งอาจเกิดกับตัวผู้เขียนโปรแกรมเองก็ได้ ประโยคหมายเหตุนี้อาจขึ้นต้นประโยคด้วยคำว่า "Rem" ที่ย่อมาจาก คำว่า "Remark" หรือใช้สัญลักษณ์

" " นำหน้าประโยคก็ได้ ดังตัวอย่าง

Rem This function is use to open database files

'This function is use to close database files

2.3.8.3 ประโยคประกาศ

ใช้ประกาศถึงตัวแปร ค่าคงที่ และฟังก์ชันหรือรูทีนย่อย สำหรับการประกาศตัวแปรและค่าคงที่ได้กล่าวไว้แล้วในข้างต้นว่าแบ่งออกเป็น Global, Module หรือ Local ส่วนการประกาศฟังก์ชันหรือรูทีนย่อยใช้คำสั่ง "declare"ซึ่งมีรูปแบบดังนี้

Declare Sub globalname Lib "libname" [Alias aliasname"]

[([[ByVal] variable [As type][,ByVal] variable [As type]]...])]

Declare Function globalname Lib "libname" [As type][,ByVal]

variable [As type]]...])] [As type]

2.3.9 ประโยคควบคุมลำดับการทำงาน

แบ่งออกเป็นหลายกลุ่มย่อยมีทั้งประโยคตัดสินใจ ประโยคการทำงานวนรอบ (loop) การกระโดดแบบมีเงื่อนไข รวมถึงคำสั่งจบการทำงานของโปรแกรมทั้งแบบชั่วคราวและถาวร ซึ่งจะขอกล่าวเรียงตามลำดับดังนี้

2.3.9.1 IF...Then...Else

เป็นคำสั่งตัดสินใจว่าจะทำงานหรือไม่ตามเงื่อนไขที่กำหนด สามารถกำหนดเพิ่มได้ว่าหากไม่เป็นไปตามเงื่อนไขให้ทำงานใด ซึ่งอยู่หลังคำว่า "Else" อีกทั้งยังกำหนดให้ตรวจสอบหลายเงื่อนไขได้ด้วย "Else"

```
If condition1 Then
        [Statementblock-1]
[Else If condition2 Then
        [Statementblock-2]]...
[Else
        [Statementblock-n]]...
End If
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3.9.2 Select Case

ในบางครั้งการเขียนประโยคตัดสินใจด้วย IF…Then…Else จะทำให้โปรแกรมเยิ่นเย้อ และอ่านยาก ประโยค Select Case จะทำหน้าที่คล้ายกับ IF…Then…Else แต่ให้ความหมายกระชับกว่าโดยใช้กับค่าต่าง ๆ ของนิพจน์เดียวเท่านั้น

```
Select Case testexpression
    [Case exp1
        [Statementblock-1]
    [Case exp2
        [Statementblock-2]]…
    [Case Else
        [Statementblock-n]
    End Select
```

2.3.9.3 For …Next

เพื่อกำหนดให้มีการทำงานซ้ำ ๆ กันเป็นจำนวนครั้งที่ต้องการ รูปแบบคำสั่งได้แก่

```
For counter = start to end [Step increment]
    [Statements]
Next [counter]
```

ทำการกำหนดค่าเริ่มต้น (Start) ให้กับ Counter เพื่อเป็นตัวนับให้มีการ ทำงานในส่วน Statements ไปเรื่อย ๆ จนกว่าค่า end โดยในการทำงานแต่ละรอบจะเพิ่มค่า Counter ขึ้นตามค่าใน Increment

2.3.9.4 Do…Loop

ใช้เมื่อต้องการให้ทำงานวนรอบแบบมีเงื่อนไข ประโยค Do…Loop นี้สามารถกำหนดให้ตรวจสอบเงื่อนไขก่อนหรือหลังการทำงานในแต่ละรอบก็ได้โดยใส่ค่า “While” หรือ “ Until ” ข้างหลัง Do หรือ Loop ตามลำดับการใช้ while จะหมายถึงให้ทำงานวนรอบเมื่อนิพจน์ที่ตามมาให้ค่าจริง (True) คำทำงานจนกว่านิพจน์จะกลายเป็นเท็จ (False) หรือพูดอีกนัยหนึ่งคือทำงานไปจนกว่านิพจน์เป็นจริงนั่นเอง

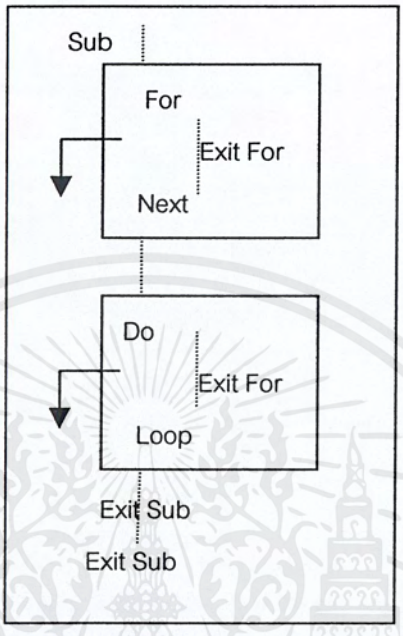
```
Do [{While/Unit} condition]
    [Statementblock]0
Loop
หรือ
Do
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

[Statementblock]

Loop [{While/Unit} condition]

2.3.9.5 หยุดการทำงานของโปรแกรม



ภาพที่ 2.35 แสดงการทำงานจากส่วนของโปรแกรมแบบต่าง ๆ ด้วยคำสั่ง Exit

ใน Visual Basic มีทั้งคำสั่งในการหยุดการทำงานแบบชั่วคราวและถาวรรวมทั้งจบการทำงานในวนรอบ (Loop) มีฟังก์ชันหรือโพรซีเดอร์ด้วย การหยุดทำงานชั่วคราวใช้คำสั่ง Stop ซึ่ง จะเป็นคำสั่งโดดไม่ต้องมีพารามิเตอร์ประกอบส่วนการหยุดโปรแกรมอย่างถาวรใช้คำสั่ง End ข้อแตกต่างของ 2 คำสั่งนี้คือ Stop จะไม่ยกเลิกตัวแปรและจะยังไม่ปิดไฟล์ที่เปิดค้างอยู่ให้ อีกทั้งยังสามารถสั่งให้ทำงานต่อไปภายหลังได้โดยเลือกหัวข้อ Continue จากเมนู Run หรือหัวข้อ Single Step ในเมนู Debug ส่วน END จะยกเลิกตัวแปรและปิดไฟล์ทั้งหมดที่เปิดค้างอยู่ในโปรแกรมนั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4 ไมโครคอนโทรลเลอร์ MCS–51

MCS–51 เป็นไมโครคอนโทรลเลอร์แบบชิปเดี่ยวที่มีข้อดีเมื่อเทียบกับไมโครโปรเซอร์ขนาด 8 บิตตระกูลอื่น ดังนี้

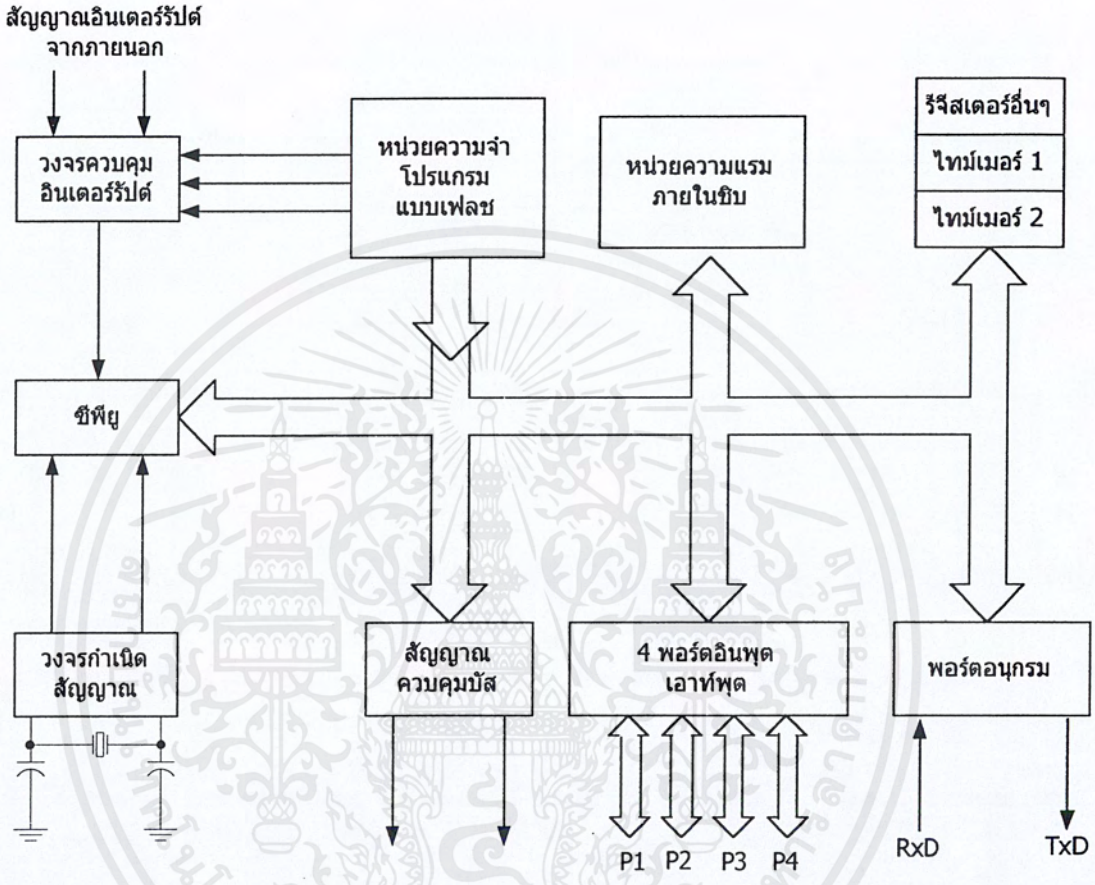
- มีหน่วยความจำสำหรับเก็บข้อมูลทั่วไป (RAM) บรรจุไว้ภายใน 128 – 256 ไบต์
- มีหน่วยความจำสำหรับเก็บโปรแกรมควบคุมการทำงานอยู่ภายในจำนวน 4 กิโลไบต์
- มีวงจรตั้งเวลาวงจรนับขนาด 16 บิตอยู่ 2 ตัวภายใน
- มีวงจรรับส่งข้อมูลอนุกรมได้ 2 ทิศทาง
- มีสัญญาณนาฬิกาภายในตัว
- มีพอร์ทที่สามารถรับหรือส่งข้อมูลได้ 2 ทิศทางจำนวน 4 พอร์ท พอร์ทละ 8 บิต

นอกจากนี้ MCS–51 ยังมีคุณสมบัติอย่างอื่น ๆ ที่น่าสนใจคือ

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8บิต
- ภายในมีหน่วยความจำโปรแกรมแบบเฟลชสามารถลบและเขียนได้พันครั้ง
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแรม ในบางเบอร์มีหน่วยความจำแบบอีอีพรอมเพิ่มเติม
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิบ
- มีวงจรสื่อสารอนุกรมแบบ SPI สำหรับในอนุกรม AT89Sxx
- ต้องการแหล่งจ่ายไฟ 5 โวลท์เพียงชุดเดียว
- มีหน่วยความจำสำหรับเก็บโปรแกรมควบคุมการทำงานอยู่ภายในชิบ
- สามารถใช้หน่วยความจำ สำหรับโปแกรม และข้อมูลที่อยู่ภายในชิบอย่างละ 64 กิโลไบต์
- มีคำสั่งคูณและหารเลขขนาด 8 บิต ในตัวเอง
- จัดลำดับความสำคัญของสัญญาณอินเตอร์รัปต์ได้ 2 ระดับ
- รับส่งข้อมูลแบบอนุกรมได้ในตัว โดยสามารถกำหนดอัตราเร็วในการรับส่งข้อมูลได้ตั้งแต่ 300 – 375 กิโลบิตต่อวินาที่
- สามารถประมวลผลแบบบูลีนเพื่อใช้งานควบคุมโดยเฉพาะ
- มีรีจีสเตอร์สำหรับใช้งานเป็นไทม์เมอร์หรือเคาน์เตอร์เพื่อนำจำนวนสัญญาณนาฬิกาภายในชิบหรือ นับการเปลี่ยนสถานะของสัญญาณภายนอกขนาด 16 บิต จำนวน 2 ตัว เพื่อใช้สำหรับนับจำนวนพัลส์หรือใช้วัดช่วงเวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- หน่วยความจำสำหรับเก็บข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลได้ทั้งระดับไบต์และระดับบิตเพื่อให้การออกแบบโปรแกรมและการควบคุมระบบงานทำได้ง่ายขึ้น

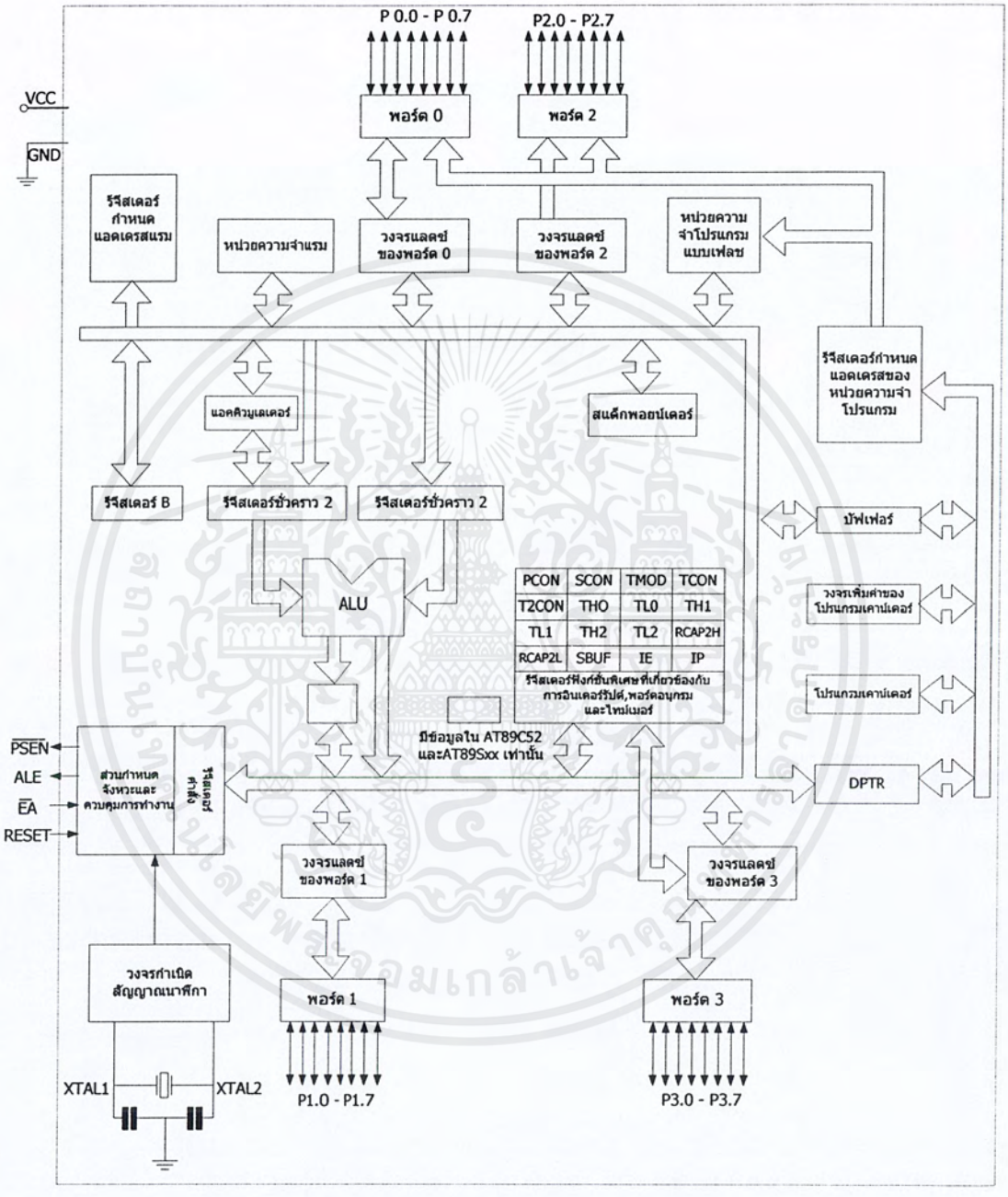


ภาพที่ 2.36 โครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.1 สถาปัตยกรรมของไมโครคอนโทรลเลอร์ 8051

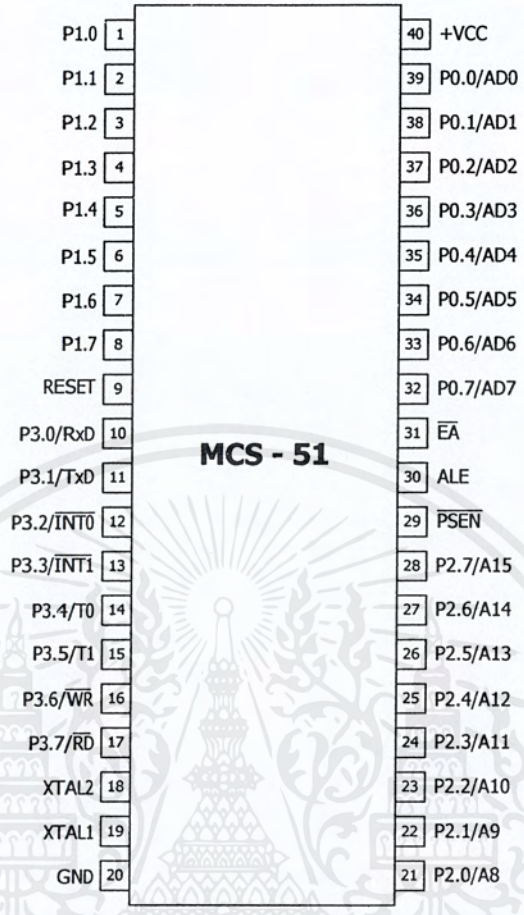
โครงสร้างภายในชิบไมโครคอนโทรลเลอร์ 8051 ชิปเดี่ยวแสดงดังภาพที่ 2.37 ซึ่งอธิบายถึงส่วนย่อยๆ ภายใน 8051



ภาพที่ 2.37 แสดงโครงสร้างภายในชิปไมโครคอนโทรลเลอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.2 ขาใช้งานต่างๆ ของไมโครคอนโทรลเลอร์ มีดังนี้



ภาพที่ 2.38 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์

- V_{CC} ขา 40 เป็นขาที่ต้องป้อนไฟเลี้ยง +5 โวลท์เข้าไปเพื่อให้วงจรสามารถทำงานได้

- V_{SS} ขา 20 เป็นขาที่ต้องต่อกราวด์ของแหล่งจ่ายไฟ

- RST ขา 9 ขารีเซ็ทนี้จะรีเซ็ทการทำงานของ 8051 ถ้าป้อนสัญญาณที่มีสภาวะลอจิก 1 ที่ขานี้จะเป็นการรีเซ็ทการทำงาน กลับไปเริ่มการทำงานจากคำสั่งที่อยู่ในหน่วยความจำตำแหน่ง 0000

- ALE ขา 30 ใช้เป็นขาส่งสัญญาณออกไปภายนอก เพื่อควบคุมการเลตช์ค่าตำแหน่งไบต์ต่ำจากพอร์ท 0 ในระหว่างการติดต่อหน่วยความจำสำหรับเก็บโปรแกรมหรือข้อมูลภายนอก

- PSEN ขา 29 ใช้ส่งสัญญาณเพื่ออ่านคำสั่งจากโปแกรม ที่เก็บไว้ในหน่วยความจำภายนอกชิฟ

- XTAL1 ขา 19 ใช้ต่อคริสตอลภายนอก โดยเป็นอินพุทเข้าสู่วงจรออสซิลเลเตอร์

- XTAL2 ขา 18 ใช้ต่อคริสตอลภายนอก โดยเป็นเอาท์พุทออกจากวงจรออสซิลเลเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- PORT 0 เป็นพอร์ทขนาน 8 บิต อยู่ที่ขา 39 ถึง 32 เริ่มจากบิต 0 ถึงบิต 7 พอร์ท 0 นี้ ใช้ได้ทั้งการรับส่งตำแหน่งและข้อมูลกับหน่วยความจำหรือใช้รับส่งข้อมูลก็ได้ นอกจากนี้ยังใช้งานได้หลายอย่างดังนี้

1.ใช้สำหรับส่งค่าตำแหน่งหน่วยความจำที่ต้องการติดต่อด้วย โดย 8 บิตล่างจะถูกส่งออกไปทางพอร์ท 0 และ 8 บิตบนถูกส่งออกทาง พอร์ท 2

2.ใช้สำหรับการส่งข้อมูลกับData Memoryหรือใช้รับข้อมูลจาก Program Memory

3.ใช้รับส่งข้อมูลออกทางพอร์ทโดยตรง

- PORT 1 เป็นพอร์ทขนานขนาด 8 บิต อยู่ที่ขา 1 ถึง 8 เริ่มที่บิต 0 ถึงบิต 7 ตามลำดับใช้หน้าที่เป็นตัวรับส่งข้อมูลเท่านั้น ไม่สามารถส่งตำแหน่งได้

- PORT 2 เป็นพอร์ทขนานขนาด 8 บิตอยู่ที่ขา 21 ถึง 28 เริ่มจาก บิต 0 ถึง บิต 7 ตามลำดับ ใช้งานเพียง 2 ลักษณะคือ

1.ใช้ส่งค่าตำแหน่งหน่วยความจำภายนอก ที่ต้องการติดต่อทำงานร่วมกันกับพอร์ท 0

2. ใช้เป็นพอร์ทรับส่งข้อมูลกับภายนอก

- PORT 3 เป็นพอร์ทขนานขนาด 8 บิตอยู่ที่ขา 10 ถึง 17 เริ่มที่บิต 0 ถึง 7 ตามลำดับ นอกจากจะใช้งานเหมือนพอร์ทอื่นๆ แล้วยังใช้งานอื่นโดยใช้คำสั่งควบคุมดังนี้

P3.0 (RxD)	เป็นขาที่รับข้อมูลแบบอนุกรม
P3.1 (TxD)	เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม
P3.2 (INTO)	ใช้รับสัญญาณขัดจังหวะจากภายนอก
P3.3 (INT1)	ใช้รับสัญญาณขัดจังหวะภายใน
P3.4 (T0)	ใช้เป็นขารับสัญญาณให้เคาน์เตอร์ของไทม์เมอร์ 0
P3.5 (T1)	ใช้เป็นขารับสัญญาณให้เคาน์เตอร์ของไทม์เมอร์ 1
P3.6 (WR)	ใช้เป็นขาสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอก
P3.7 (RD)	ใช้เป็นขาสัญญาณควบคุมการอ่านข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอก

พอร์ท 3 ของ MCS – 51 ถูกใช้เป็นพอร์ทอนุกรม จะใช้ขา TxD และ RxD ในการรับส่งข้อมูล โดยทั้งสองจะอยู่ในพอร์ท 3 คือ P3.1 หรือขา 11 เป็น TxD และ P3.0 หรือขา 10 เป็น RxD พอร์ทอนุกรม MCS–51 สามารถทำงานเป็น Full Duplex ได้ คือสามารถรับและส่งข้อมูลในเวลาเดียวกันได้ โดยมีการรับส่งข้อมูลแบบบัฟเฟอร์สำหรับเก็บข้อมูลให้ใช้

รีจีสเตอร์ที่สำคัญในการรับส่งข้อมูลคือ SBUF และ SCON ซึ่งเป็นรีจีสเตอร์ที่อยู่ใน Special Function Register โดยรีจีสเตอร์ Serial Port Buffer (SBUF) จะอยู่ในตำแหน่ง 99H

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้าเขียนข้อมูลลงในตำแหน่งนี้จะเป็นการส่งข้อมูลออกทางพอร์ทอนุกรม และถ้าอ่านข้อมูลจากตำแหน่งนี้จะเป็นการรับข้อมูลจากพอร์ตทอนุกรม โดยใน SBUF จะประกอบด้วย บัฟเฟอร์ 2 ตัว สำหรับส่งและรับข้อมูล

สำหรับ Serial Port Register (SCON) ซึ่งอยู่ที่ตำแหน่ง 98H จะเป็นรีจีสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ รีจีสเตอร์นี้จะทำหน้าที่ควบคุมและบอกสถานะต่างๆของการรับส่งข้อมูลแบบอนุกรม

สำหรับความเร็วของการส่งข้อมูล (Baud Rate) สามารถหาได้จากการหาสัญญาณนาฬิกาที่ใช้กับ MCS-51

ตารางที่ 2.5 Serial Port Control Register

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 สำหรับส่งข้อมูลในโหมด 2 และโหมด 3 สามารถเซตและเคลียร์ได้โดยซอฟแวร์
SCON.2	RB8	9AH	ค่าของบิต 9 เมื่อรับข้อมูลเข้ามา
SCON.1	TI	99H	บิตแฟลกแสดงการอินเตอร์รัปต์ภายหลังการส่งข้อมูลออกไป โดยจะเซตเมื่อส่งข้อมูลออกไปหมดแล้ว และสามารถเคลียได้โดยซอฟแวร์
SCON.0	RI	98H	แฟลกแสดงการอินเตอร์รัปต์ภายหลังการรับข้อมูลเข้ามาสามารถเคลียได้โดยซอฟแวร์

ตารางที่ 2.6 แสดงโหมดต่างๆ ของการรับส่งแบบอนุกรม

SM0	SM1	MODE	ความหมาย	Band Rate
0	0	0	Shift Register	เปลี่ยนแปลงไม่ได้ (Oscillator Frequency)
0	1	1	8-Bit UART	สามารถเปลี่ยนแปลงได้โดยกำหนดจากไทม์เมอร์
1	0	2	9-Bit UART	เปลี่ยนแปลงไม่ได้ (Oscillator Frequency/12 หรือ /64)
1	1	3	9-Bit UART	สามารถเปลี่ยนแปลงได้โดยกำหนดจากไทม์เมอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4.3 ไทม์เมอร์/เคาน์เตอร์

ไมโครคอนโทรลเลอร์ในตระกูล มีรีจีสเตอร์พิเศษที่สามารถเลือกใช้เป็นไทม์เมอร์หรือเคาน์เตอร์อย่างใดอย่างหนึ่ง รีจีสเตอร์ประเภทนี้มีอยู่ด้วยกัน 2 ตัวแต่ละตัวขนาด 16 บิต เรียกไทม์เมอร์ 0 และไทม์เมอร์ 1 ตามลำดับ

ไทม์เมอร์นั้นค่าในรีจีสเตอร์ที่ใช้เป็นไทม์เมอร์ที่ถูกเลือกใช้งานจะถูกเพิ่มเติมขึ้น ในทุกๆ ช่วงของทุกแมชชีนไซเคิลเคาน์เตอร์นั้น ค่าในรีจีสเตอร์ที่ใช้เป็นเคาน์เตอร์ที่ถูกเลือกใช้งานจะถูกเพิ่มค่าที่ละ 1 เมื่อมีการเปลี่ยนสถานะ

2.4.3.1 ไทม์เมอร์ 0 และไทม์เมอร์ 1

สามารถเลือกการทำงานให้เป็นไทม์เมอร์ หรือเคาน์เตอร์ก็ได้ โดยการกำหนดค่าบิตในรีจีสเตอร์ใช้งานโดยเฉพาะ โดยหากบิตนี้มีค่าเป็น 0 หมายถึงเลือกใช้งานเป็นไทม์เมอร์ ถ้าบิตนี้เป็น 1 หมายถึงเลือกใช้งานเป็นเคาน์เตอร์

นอกจากจะเลือกการทำงานของรีจีสเตอร์ให้เป็นไทม์เมอร์ หรือเคาน์เตอร์ได้แล้ว ในแต่ละการทำงานยังมีการทำงานย่อยอยู่อีก 4 แบบ ตามความเหมาะสมของการใช้งาน

โหมด 0 จะใช้รีจีสเตอร์ขนาด 8 บิตเป็นตัวนับโดยมีการเพิ่มค่าครั้งละ 1 ทุกครั้ง และ นับสัญญาณได้ครบ 32 ครั้ง โดยในโหมนี้รีจีสเตอร์ที่ใช้นับเพียง 13 บิต (8 บิตในรีจีสเตอร์ TLx รวมกับ 5 บิตใน THx)

โหมด 1 การทำงานเหมือนกับโหมด 0 เว้นแต่ค่าในรีจีสเตอร์ถูกใช้งานครบทั้ง 16 บิตนั่นเอง คือไทม์เมอร์หรือเคาน์เตอร์ในโหมดนี้มีขนาด 16 บิต

โหมด 2 ในโหมดนี้จะกำหนดรีจีสเตอร์ใช้งานในการนับเพียง 8 บิต (จากรีจีสเตอร์ TX) ที่มีการโหลดค่าด้วยค่าในรีจีสเตอร์ THx การใช้งานโหมดนี้มีไว้เพื่อใช้สร้างสัญญาณอินเตอร์รัปต์ที่มีคาบเวลาคงที่

โหมด 3 ในโหมดนี้ไทม์เมอร์ 1 จะไม่มีการนับแต่ไทม์เมอร์จะบังคับให้รีจีสเตอร์ TL0 ของไทม์เมอร์ 0 ถูกใช้เป็นไทม์เมอร์เพียงอย่างเดียว การทำงานโหมด 3 มีไว้เพื่อใช้งานที่ต้องการไทม์เมอร์หรือเคาน์เตอร์ขนาด 8 บิตเพิ่มขึ้น

ตารางที่ 2.7 แสดงค่าในรีจีสเตอร์ TMOD ค่าต่างๆสำหรับ Timer 0

โหมด	ฟังก์ชั่นไทม์เมอร์ 0	ควบคุมจากโปรแกรม	TMOD ควบคุมจากฮาร์ดแวร์ภายนอก
0	13 Bit Timer	00H	08H
1	16 Bit Timer	01H	09H
2	8 Bit Auto Reload	02H	0AH
3	Two 8 Bit Counter	03H	0BH

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่2.8 แสดงค่าในรีจีสเตอร์ TMOD ค่าต่างๆสำหรับ Counter 0

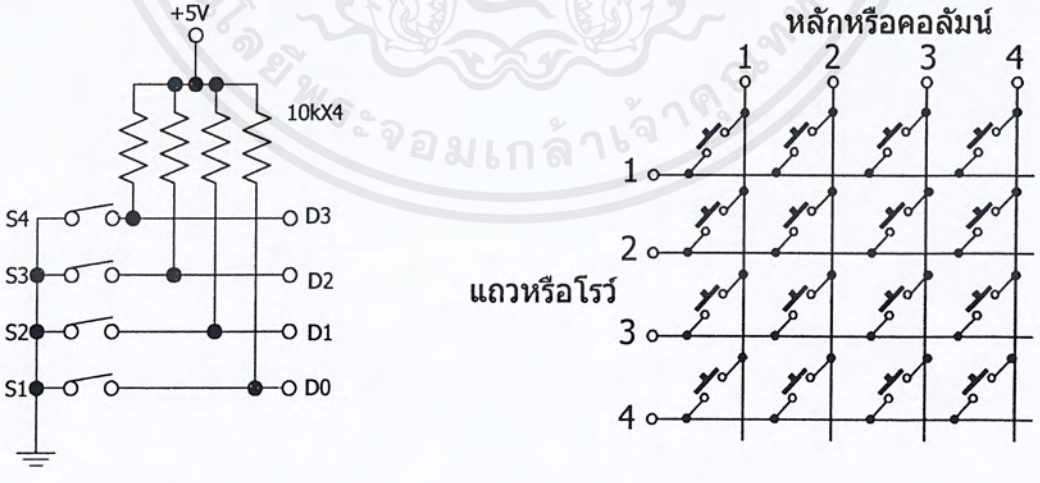
โหมด	ฟังก์ชั่นไทม์เมอร์ 0	ควบคุมจากโปรแกรม	TMOD ควบคุมจากฮาร์ดแวร์ภายนอก
0	13 Bit Timer	04H	0CH
1	16 Bit Timer	05H	0DH
2	8 Bit Auto Reload	06H	0EH
3	Two 8 Bit Counter	07H	0FH

ตารางที่ 2.9 แสดงค่าในรีจีสเตอร์ TMOD ค่าต่างๆสำหรับ Timer 1

โหมด	ฟังก์ชั่นไทม์เมอร์ 0	ควบคุมจากโปรแกรม	TMOD ควบคุมจากฮาร์ดแวร์ภายนอก
0	13 Bit Timer	00H	80H
1	16 Bit Timer	10H	90H
2	8 Bit Auto Reload	20H	A0H
3	Two 8 Bit Counter	30H	B0H

2.5 คีย์แพด

การอ่านค่าหรือการกด สวิทช์จะมีด้วยกัน 2 ลักษณะใหญ่ คือ ต่อเข้ากับไฟเลี้ยงหรือ กราวด์โดยตรง เมื่อสวิทช์ตัวใดต่อวงจรสามารถอ่านค่าได้โดยตรง ดังรูปที่ 2.39(ก) วงจรในลักษณะนี้จะไม่มีความซับซ้อน สามารถอ่านค่าได้ง่ายและรวดเร็ว แต่ข้อเสียก็คือ ถ้าหากสวิทช์มีจำนวนมากทำให้จำนวนของสายข้อมูลจะมีมากตามไปด้วยทำให้ระบบและวงจรมีขนาดใหญ่และสิ้นเปลือง



(ก) (ข)

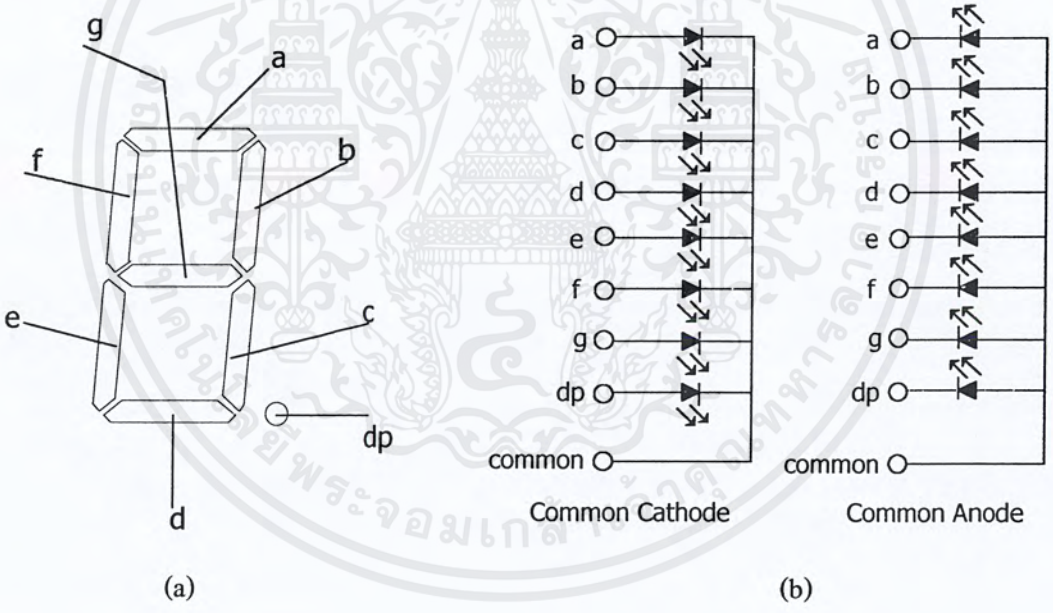
ภาพที่ 2.39 แสดงวงจรการต่อสวิตช์แบบต่อเข้าไฟเลี้ยงกับกราวด์และการต่อสวิตช์แบบเมตริกหรือคีย์แพด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในภาพที่ 2.39(ข) วงจรแผงสวิทช์อีกลักษณะหนึ่งคือ การต่อวงจรแบบเมตริกซ์ (matrix switch) จะถูกต่อกันทั้งในแนวตั้งและแนวนอน จะเรียกแนวตั้งว่า หลักหรือ คอลัมน์ (column) แนวนอนจะเรียกว่าแถวหรือ (row) ดังนั้นค่าของสวิทช์จะต้องประกอบด้วยตำแหน่งที่แน่นอน สวิทช์แบบนี้จะมีข้อดีคือสารมารถรองรับการเพิ่มของสวิทช์ได้อย่างสะดวก เพียงทำการแก้ไขซอฟต์แวร์เล็กน้อยเท่านั้น ทำให้เป็นที่นิยมใช้กันมากในระบบควบคุมแบบกึ่งอัตโนมัติและอัตโนมัติ ที่มีจำนวนสวิทช์มากกว่า 8 ตัว ในการใช้งานทั่วไปจะเรียกว่า คีย์แพด (key pad)

2.6 LED แบบ ตัวเลข 7 ส่วน (7-Segment)

LED ตัวเลข 7 ส่วนประกอบขึ้นจาก LED จำนวน 7 ตัวที่บรรจุอยู่ในตัวถังเดียวกันและได้รับการจัดเรียงเป็นรูปตัวเลข LED แต่ละตัวจะถูกเรียกว่า ส่วน หรือ เซกเมนต์ (Segment) แต่ละส่วนหรือเรียกว่า เซกเมนต์มีชื่อเรียกแตกต่างกันตามตำแหน่งที่ได้รับการจัดวางคือ a,b,c,d,e,f และ g ดังแสดงในภาพที่ 2.40 ส่วน dp เป็น LED อีก 1 ตัวที่บรรจุอยู่ใน LED ตัวเลข 7 ส่วนนี้ใช้เป็นตัวแสดงจุดทศนิยมในกรณีที่มีการแสดงผลในลักษณะเลขที่มีทศนิยม



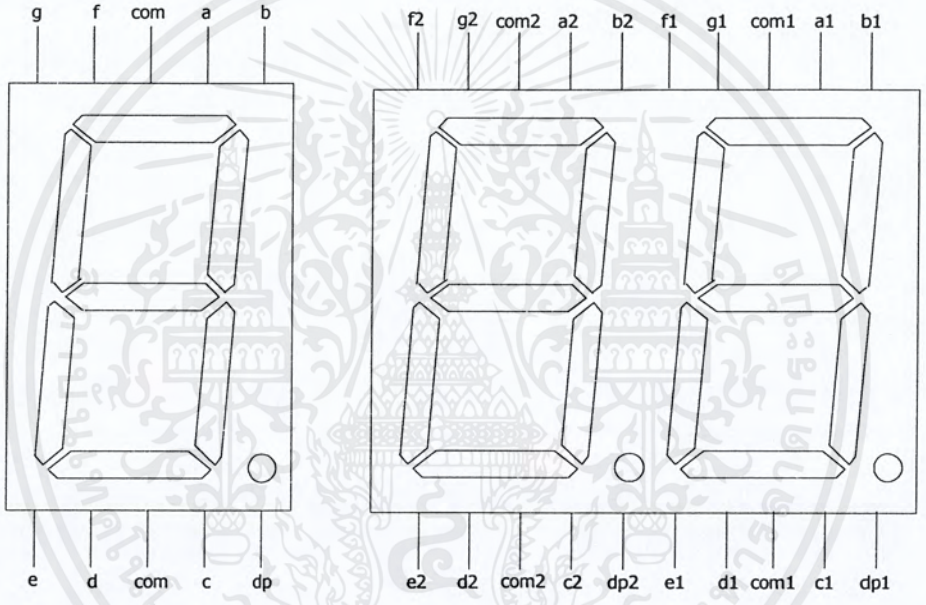
ภาพที่ 2.40 (a) รูปร่างและการกำหนดชื่อเซกเมนต์ต่างๆ ของ LED ตัวเลข 7 ส่วน
(b) วงจรภายในของ LED ตัวเลข 7 ส่วน ทั้งแบบแคโทดร่วมและแอโนดร่วม

LED ทุกตัวที่บรรจุอยู่ใน LED ตัวเลข 7 ส่วน นี้มีขาต่อร่วมกัน ซึ่งก็มีทั้งแบบต่อขาแคโทดร่วมกันเรียกว่าแบบแคโทดร่วม (Common cathode) และแบบต่อขาแอโนดร่วมกันเรียกว่าแบบแอโนดร่วม (common anode) การขับให้ LED ตัวเลข 7 ส่วนแบบแคโทดร่วมสว่างจะต้องจ่ายไฟลบเข้าที่ขาร่วม แล้วจ่ายไฟบวกเข้าที่ขาแอโนด ซึ่งก็คือขาของแต่ละเซกเมนต์นั่นเอง ดังแสดง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในภาพที่ 2.41 ในขณะที่ LED ตัวเลข 7 ส่วนแบบแอโนดร่วมจะต้องจ่ายไฟบวกเข้าที่ขาร่วม แล้วจ่ายไฟลบเข้าที่ขาแคโทด ซึ่งเป็นขาของแต่ละเซกเมนต์ ดังแสดงในภาพที่ 2.41(b)

LED ตัวเลข 7 ส่วนมีจำหน่ายทั้งแบบตัวเดี่ยว,ตัวคู่ และแบบที่มีมากกว่า 2 หลัก แต่ที่นิยมใช้งานและส่วนแบบตัวเดี่ยวมีขาต่อใช้งานและหาได้ง่ายมี 2 แบบคือ แบบตัวเดี่ยวและแบบตัวคู่ โดยมีการจัดขาดังในภาพที่ 2.41 จะเห็นได้ว่า LED ตัวเลข 7 ส่วนแบบตัวเดี่ยวมีขาต่อใช้งาน 10 ขา คือขา a, b, c, d, e, f, g, dp และขาร่วม (Common) ซึ่งมี 2 ขา ถ้าเป็น LED 7 ส่วนแบบตัวคู่จะมีขาต่อใช้งาน 20 ขา แบ่งขาเป็นขา a, b, c, d, e, f, g, dp และขาร่วม (Common) ซึ่งมี 2 ขา รวม 16 ขา และขาร่วมอีกหลักละ 2 ขา การต่อขาร่วมของแต่ละหลักทั้ง 2 ขานั้น สามารถต่อใช้งานเพียงขาเดียวได้ เนื่องจากโครงสร้างภายในของ LED ตัวเลข 7 ส่วน ขาร่วมนี้ต่อถึงกันอยู่แล้ว



ภาพที่ 2.41 แสดงการจัดขาของ LED ตัวเลข 7 ส่วนทั้งแบบตัวเดี่ยวและตัวคู่

2.6.1 การขับ LED ตัวเลข 7 ส่วนแบบหลักเดี่ยว

ไมโครคอนโทรลเลอร์ไม่ควรนำมาขับ LED 7 ส่วนโดยตรงเนื่องจากความสามารถในการจ่ายกระแสเอาท์พุตรวมไม่สูงมากนักจึงต้องอาศัยไอซีบัฟเฟอร์มาช่วยในการขับ LEDและที่เอาท์พุตของไอซีบัฟเฟอร์ที่ต่อกับ LED ตัวเลข 7 ส่วนต้องมีตัวต้านทานจำกัดกระแสให้แก่ LED และในทุกเซกเมนต์ การกำหนดให้ LED ตัวเลข 7 ส่วนแสดง ข้อมูลเป็นตัวเลขหรือสัญญลักษณ์ใดๆ ก็ตามต้องมีการกำหนดรูปแบบการแสดงผลของเซกเมนต์ต่างๆด้วยข้อมูลแต่ละบิตของไมโครคอนโทรลเลอร์ แล้วใช้วิธีการเปิดตารางหรือ Look up table

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6.2 การขับ LED ตัวเลข 7 ส่วนแบบมัลติเพล็กซ์

ในกรณีที่ต้องการให้ไมโครคอนโทรลเลอร์ MCS – 51 ขับตัวเลข 7 ส่วนมากกว่า 1 หลัก จะต้องใช้เทคนิคที่เรียกว่า การแสดงผลแบบมัลติเพล็กซ์ (Multiplex) อันเป็นวิธีการขับให้ LED ตัวเลข 7 ส่วนสว่างที่ละหลักด้วยอัตราเร็วที่ตาของมนุษย์ไม่สามารถตรวจจับได้ทัน จึงดูเหมือนว่า LED ตัวเลข 7 ส่วนทุกหลักติดสว่างในเวลาเดียวกัน

การแสดงผลแบบมัลติเพล็กซ์นี้มีประโยชน์หลายประการดังนี้

1.ช่วยลดพลังงานที่ไฟฟ้าที่ใช้ ทำให้ขนาดของแหล่งจ่ายไฟลดลง ขนาดของระบบโดยรวมเล็กลงด้วย

2.ช่วยให้ไมโครคอนโทรลเลอร์สามารถขับ LED ตัวเลข 7 ส่วนได้มากกว่า 1 หลักโดยที่ใช้จำนวนพอร์ตเพิ่มเติมเฉพาะการติดต่อกับขาคอมม่อนเท่านั้น

3.ลดจำนวนตัวต้านทานที่ใช้ในการจำกัดกระแสของ LED ตัวเลข 7 ส่วนในแต่ละเซกเมนต์ ยกตัวอย่าง LED ตัวเลข 7 ส่วน หนึ่งหลักต้องใช้ตัวต้านทานจำกัดกระแส 8 ตัวถ้าหากขับ LED ตัวเลข 7 ส่วน 4 หลักโดยตรง ต้องใช้ตัวต้านทานมากถึง 32 ตัว ในขณะหากใช้การแสดงผลแบบมัลติเพล็กซ์ยังคงใช้งานตัวต้านทานจำกัดกระแสให้ LED ในแต่ละเซกเมนต์เพียง 8 ตัว ไม่ว่าจะขับ LED ตัวเลข 7 ส่วนกี่หลักก็ตาม

การขับ LED ตัวเลข 7 ส่วนแบบมัลติเพล็กซ์จะทำการต่อขาแต่ละเซกเมนต์ร่วมกันคือ เซกเมนต์ a ของทุกหลักจะต่อถึงกันไล่เรียงไปจนถึงเซกเมนต์ g ในบางงานที่ต้องใช้จุ dp ก็ต้องต่อขาของจุด dp ร่วมกันด้วยการควบคุมให้ LED ตัวเลข 7 ส่วนหลักใดสว่าง ทำได้โดยการจ่ายไฟเข้าที่ขาร่วมของ LED ตัวเลข 7 ส่วนหลักนั้นๆ ยกตัวอย่างหาก LED ตัวเลข 7 ส่วนที่ใช้เป็นแบบแคโทดร่วม หากต้องการให้ LED ตัวเลข 7 ส่วนหลักที่ 3 ติด ก็ให้ต่อขาร่วมของหลักที่ 3 ลงกราวด์หรือจ่ายไฟลบ LED ตัวเลข 7 ส่วนหลักที่ 3 ก็จะติดตามข้อมูลที่ส่งเข้ามายังของแต่ละเซกเมนต์

2.7 โมดูลแสดงผลแบบผลึกเหลว(LCD)

ในโมดูล LCD จะมีส่วนประกอบหลักๆ อยู่ 3ส่วนคือ

1.ตัวแสดงผล (Display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็นโดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

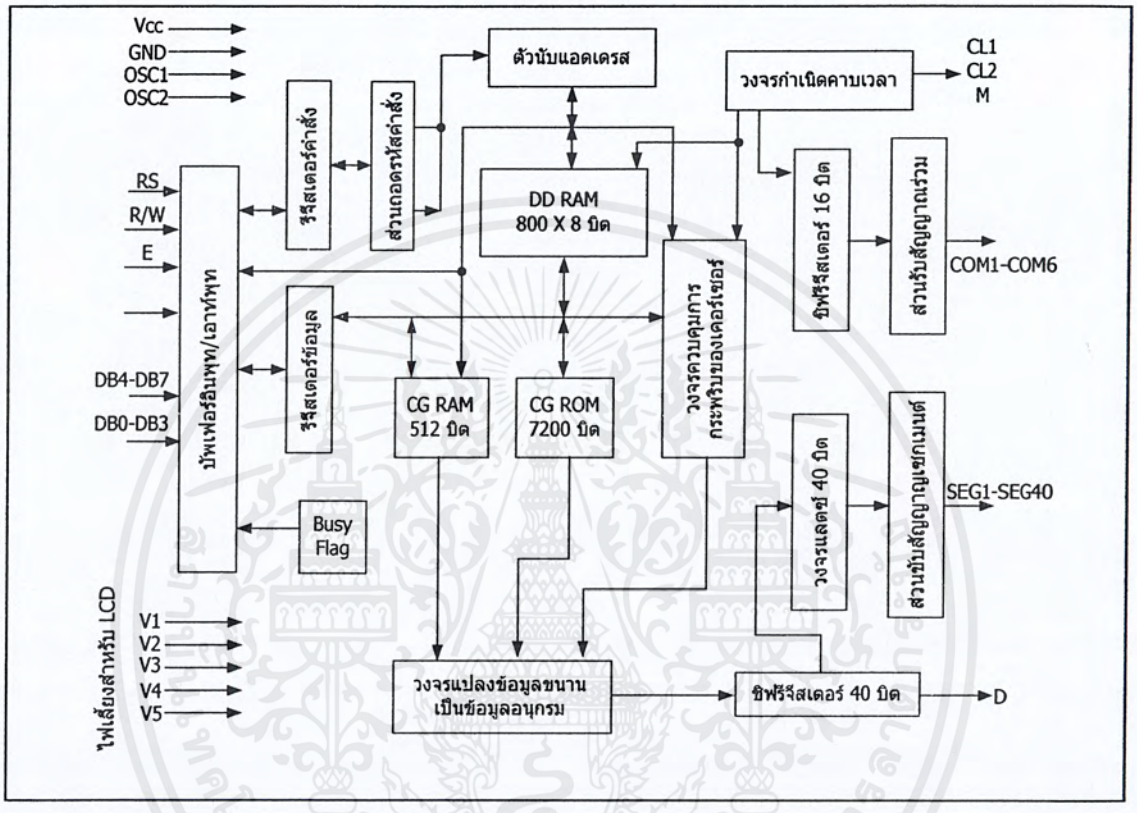
2.ตัวควบคุม (Controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCD เช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปที่ นิยมใช้คือ เบอร์ HD44780 และ เบอร์ HD61830 โดยเบอร์ HD44780 จะใช้ควบคุม LCD แบบอักขระ HD61830 ใช้ควบคุม LCD แบบกราฟิก

3.ตัวขับ (Driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.1 โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับ โครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อน ในที่นี้จะอ้างเกี่ยวกับโมดูล LCD แบบอักขระ เพราะสามารถเข้าใจได้ง่าย ดังภาพที่ 2.42 เป็นบล็อกไดอะแกรมภายในชิปควบคุม LCD เบอร์ HD44780 ซึ่งใช้ในโมดูล LCD แบบอักขระ



ภาพที่ 2.42 แสดงไดอะแกรมการทำงานของโมดูล LCD

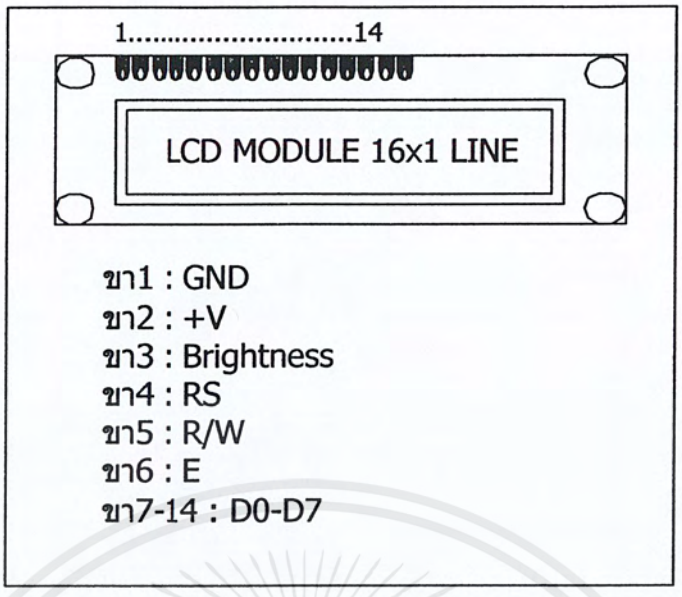
บัฟเฟอร์อินพุตเอาท์พุต เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจิสเตอร์คำสั่ง (Instruction Register :IR) เป็นรีจิสเตอร์ใช้รับคำสั่งจากอุปกรณ์ภายนอกเพื่อนำไปควบคุมการแสดงผล

รีจิสเตอร์ข้อมูล (Data Register :DR) เป็นรีจิสเตอร์ที่ใช้รับคำสั่งจากอุปกรณ์ภายนอกเพื่อนำไปควบคุมการแสดงผล

แรมเก็บข้อมูลการแสดงผล (Display Data RAM :DDRAM) เป็นหน่วยความจำแรมทำหน้าที่เก็บข้อมูลที่มาจากรีจิสเตอร์ DR ตัวควบคุมจะนำข้อมูลใน DDRAM นี้ไปเปิดตาราง (Look up lable)ของตัวอักษรที่เก็บไว้ในหน่วยความจำรอมและแรมเก็บตัวอักษรเพื่อนำไปแสดงผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.43 แสดงรูปร่างและการจัดขาโมดูล LCD แบบอักขระ

ตารางที่ 2.10 แสดงการทำงานของขา RS,R/W และ E ของโมดูล LCD แบบอักขระ

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

2.7.2 ขาใช้งานของโมดูล LCD ขนาด 16 ตัวอักษร 1 บรรทัด (LCD 16x1)

โมดูล LCD ขนาด 16x1 มีขาต่อใช้งานทั้งสิ้น 14 ขา ซึ่งแต่ละขามีการทำงานดังนี้

V_{SS} (ขา 1) : ต่อกราวด์

V_{DD} (ขา 2) : ต่อไฟเลี้ยง +5 โวลท์

Vo (ขา 3) : เป็นขารับอินพุตเพื่อปรับความเข้มของการแสดงผล

RS (ขา4) : เป็นขาอินพุทใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้น ว่าเป็นคำสั่งสำหรับรีจีสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจีสเตอร์ DR โดยถ้าขานี้เป็น “0” ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

R/W (ขา 5) : เป็นขาที่ใช้เลือกในการเขียนหรืออ่านข้อมูลกับโมดูล LCD ถ้าเป็น “0” เป็นการกำหนด ให้เขียนข้อมูล แต่ถ้าเป็น “1” จะเป็นการอ่านข้อมูล

E (ขา 6) : เป็นขาสำหรับรับสัญญาณพัลส์เอ็นเอเบิลโมดูล LCD ให้ทำงาน

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

2.7.3 คำสั่งการควบคุมโมดูล LCD

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

1.คำสั่งเคลียตัวแสดงผล (Clear Display)

มีข้อมูลเป็น 01H เป็นคำสั่งที่ใช้เขียนข้อมูลช่องว่าง หรือ Space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมคำสั่งเอ็กซีคิวต์คำสั่งนี้ จะทำการกำหนดแอดเดรส ของ DDRAM เป็น“0” เคอร์เซอร์จะกลับไปอยู่ที่ตำแหน่งซ้ายมือสุดของจอแสดงผลแล้วเซต I/D ให้เป็น “1”

2.คำสั่ง Return Home

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น “1” เป็นคำสั่งให้เคอร์เซอร์เคลื่อนที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอแสดงผลจะไม่เปลี่ยนแปลง นั่นคือ ข้อมูลคำสั่งของคำสั่งนี้จะเป็น 02H หรือ 03H ก็ได้

3.คำสั่งเลือกโหมดการป้อนข้อมูล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต6	บิต5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	0	1	I/D	S

บิต S เป็นบิตที่ใช้กำหนดลักษณะของการแสดงผล เมื่อมีการป้อนข้อมูล ถ้าหากบิต S เป็น “1” เมื่อเกิดข้อมูลใหม่บนหน้าจอแสดงผล ตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักษรข้อมูลเดิมจะถูกดันไปทางด้านซ้าย แต่ถ้าหากบิตนี้เป็น “0” เมื่อเกิดข้อมูลข้อมูลใหม่ตัวเคอร์เซอร์จะเลื่อนไปทางขวามือ

บิต I/D เป็นบิตที่ใช้กำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้วแอดเดรสของ DDRAM จะเพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น “1” แอดเดรสของ DDRAM จะเพิ่มขึ้น แต่ถ้าบิตนี้เป็น “0” แอดเดรสของบิตนี้จะลดลง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังนั้น ข้อมูลคำสั่งที่เกิดขึ้นสำหรับคำสั่งนี้ได้แก่ 04H – 07H (4 ข้อมูลคำสั่ง) และที่ใช้บ่อยคือ 06H หมายถึง กำหนดเมื่อเกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางขวามือ แอดเดรสของ DDRAM จะเพิ่มขึ้น

4. คำสั่งการควบคุมการแสดงผล

มีรายละเอียดของรูปแบบข้อมูลดังนี้

บิต 7	บิต6	บิต5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	1	D	C	B

บิต D ใช้ควบคุมการเปิดปิดจอแสดงผล ถ้าบิตนี้เป็น “1” จะเป็นการเปิดจอแสดงผล ถ้าเป็น “0” จะเป็นการปิดจอแสดงผล

บิต C ใช้ควบคุมการแสดงตัวเคอร์เซอร์บนจอแสดงผล ถ้าต้องการให้มีตัวเคอร์เซอร์แสดงผลบนจอแสดงผล ต้องกำหนดบิตนี้ให้เป็น “1” ถ้ากำหนดบิตนี้ให้เป็น “0” จะเป็นการปิดเคอร์เซอร์ หรือไม่แสดงเคอร์เซอร์

บิต B ใช้ควบคุมการกระพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น “1” เคอร์เซอร์จะกระพริบ

ดังนั้นจะมีข้อมูลคำสั่งได้ตั้งแต่ 08H-0FH (8 รูปแบบคำสั่ง) ที่ใช้บ่อยคือ 0CH เป็นการสั่งให้เปิดจอแสดงผล แต่ไม่แสดงเคอร์เซอร์ และ 0FH เป็นคำสั่งให้เปิดจอแสดงผล แสดงเคอร์เซอร์และสั่งให้เคอร์เซอร์กระพริบ

5.คำสั่งควบคุมการเลื่อนเคอร์เซอร์และข้อมูลตัวอักษร

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต6	บิต5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	1	S/C	R/L	*	*

การควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผล ขึ้นอยู่กับการกำหนดบิต S/C และ R/L ซึ่งสามารถสรุปได้ดังนี้

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.คำสั่งกำหนดฟังก์ชั่นการทำงาน

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

บิต 7	บิต6	บิต5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	1	DL	N	F	*	*

บิต DL ใช้กำหนดจำนวนบิตที่ใช้ติดต่อส่งผ่านข้อมูล ถ้าบิตนี้เป็น “0” จะเป็นการติดต่อแบบ 4 บิต แต่ถ้าเป็น “1” จะเป็น 8 บิต

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล ถ้าเป็น “0” จะแสดงผล 1 บรรทัด ถ้าเป็น“1”จะแสดง2บรรทัด และต้องการให้แสดงผลมากกว่า2บรรทัด ก็กำหนดบิต N นี้ให้เป็น“1” จุดที่น่าสังเกตคือ โมดูล LCD แบบ 16 ตัวอักษร 1 บรรทัด แม้จะมีผลการแสดงผลเพียง 1 บรรทัด แต่จะต้องกำหนด N ให้เป็น“1” เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่องคือ 00H และ 40H

บิต F ใช้เลือกความละเอียดของตัวอักษรให้การแสดงผล ถ้าบิตนี้เป็น “0” จะเป็นการแสดงผลแบบ 5x7 จุด และถ้าเป็น “1” จะแสดงผลเป็นแบบ 5x10 จุด

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

7.คำสั่งเลือกแอดเดรสของ CGRAM

เมื่อต้องการกำหนดแอดเดรสของ CGRAM ต้องกำหนดให้บิต 7 เป็น “0” บิต6 เป็น “1” ส่วน 6 บิตที่เหลือจะแทนด้วยค่าแอดเดรสของ CGRAM จะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ ก่อนที่จะอ่านข้อมูลหรือเขียนข้อมูลให้ CGRAM โดยแอดเดรสของ CGRAM อยู่ระหว่าง 00H - 3FH

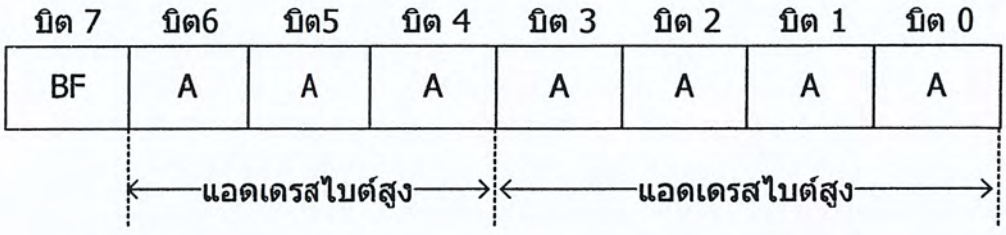
8.คำสั่งเลือกแอดเดรสของ DDRAM

ใช้ในการเลือกแอดเดรสของ DDRAM ก่อนที่จะทำการอ่านหรือเขียนข้อมูล โดยบิต 7 จะต้องเป็น “1” และข้อมูลอีก 7 บิตที่เหลือจะเป็นค่าแอดเดรสของ DDRAM จะอยู่ระหว่าง 8CH-0FFH ทั้งนี้จำนวนแอดเดรสยังขึ้นอยู่กับการกำหนดสถานะที่บิต N เป็น “0” แอดเดรสของ DDRAM จะอยู่ระหว่าง 80H-0CH และถ้าบิต N เป็น “1” แอดเดรสของ DDRAM จะมี 2 ช่วง คือ 8CH-87H และ 0C0H-0C7H

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

9.คำสั่งอ่านแฟลก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้



เป็นคำสั่งที่ใช้อ่านแฟลก BUSY (BF) โดยแฟลกนี้จะเป็นตัวบอกสถานะของตัวควบคุม LCD ว่าพร้อมจะรับข้อมูลหรือไม่ ถ้าหากบิต BF เป็น "0" แสดงว่าตัว LCD พร้อมรับข้อมูลหรือเขียนคำสั่ง ถ้าเป็น "1" แสดงว่า ขณะนี้ตัวควบคุม LC D ยังอยู่ในกระบวนการทำงานภายในหรือกำลังประมวลผลอยู่ ยังไม่พร้อมรับข้อมูล เมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา R/W เป็น "1" ด้วย แต่สัญญาณที่ RS ยังต้องเป็น "0" อยู่ เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง นอกจากนี้ยังใช้เป็น คำสั่งอ่านข้อมูลแอดเดรสของ CGRAM และ DDRAM ด้วย โดย บิต 0 ถึง บิต 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

2.7.4 การเขียนคำสั่งและข้อมูลให้แก่โมดูล LCD

ในการเขียนข้อมูลเพื่อควบคุมให้โมดูล LCD แสดงผลตามที่ผู้ใช้งานต้องการต้องส่งคำสั่ง (Instruction) แล้วกำหนดการทำงานให้แก่โมดูล LCD ก่อน จากนั้นจึงค่อยส่งข้อมูล (Data) ที่ต้องการแสดงผล เนื่องจากบัสข้อมูลของ โมดูล LCD มี 8 เส้นคือ D0-D7 และใช้เป็นทางผ่านของทั้งคำสั่งและข้อมูล ดังนั้นในการส่งคำสั่งและข้อมูลจึงต้องอาศัยการกำหนดสัญญาณลอจิกที่ขา RS ได้ลอจิก "0" หมายความว่า ข้อมูลที่ป้อนให้แก่โมดูล LCD ขณะนั้นเป็นคำสั่ง ในทางตรงข้ามหากขา RS ได้รับลอจิก "1" ข้อมูลที่ป้อนให้ขณะนั้นเป็นข้อมูลที่ใช้ในการแสดงผล

เมื่อต้องการเขียนข้อมูลใน CGRAM และ DDRAM เริ่มต้นต้องกำหนดแอดเดรสที่ต้องการอ่านหรือเขียนก่อนโดยใช้คำสั่งเลือกแอดเดรสจากนั้นกำหนดให้ขา RS เป็น "1" เพื่อแจ้งให้ตัวควบคุมภายในโมดูล LCD ทราบว่าข้อมูลที่ปรากฏต่อไปนี้เป็นข้อมูลปกติไม่ใช้คำสั่ง

ในกรณีที่ต้องการอ่านข้อมูลต้องกำหนดให้ขา R/W เป็น "1" ข้อมูลขนาด 8 บิต (หรือ4 บิต) ก็จะปรากฏบนบัสข้อมูล โดยข้อมูลที่อ่านออกมาจะได้เป็นข้อมูลจากแอดเดรสของ CGRAM หรือ DDRAM ตามที่ต้องการ

ในกรณีที่ต้องการเขียนข้อมูล เมื่อกำหนดแอดเดรสและป้อนลอจิก "1" ให้ขา RS แล้วต้องกำหนดให้ขา R/W เป็น "0" ข้อมูลที่อยู่บนบัสข้อมูลจะถูกเขียนลงในรีจีสเตอร์ DR จากนั้นจึงถ่ายทอดลงใน DDRAM ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.11 แสดงค่าในรีจีสเตอร์ TMOD ค่าต่างๆสำหรับ Counter 1

โหมด	ฟังก์ชั่นไทม์เมอร์ 0	ควบคุมจากโปรแกรม	TMOD ควบคุมจากฮาร์ดแวร์ภายนอก
0	13 Bit Timer	40H	C0H
1	16 Bit Timer	50H	D0H
2	8 Bit Auto Reload	60H	E0H
3	Two 8 Bit Counter	--	--

2.7.5 รีจีสเตอร์สำหรับใช้งานทั่วไปใน MCS-51

รีจีสเตอร์ A,B และรีจีสเตอร์ใช้งานทั่วไป R0-R7 ซึ่งอยู่ในหน่วยความจำสำหรับเก็บข้อมูลทั่วไปในชิปบริเวณ 128 ไบต์แรก รีจีสเตอร์ใช้งานทั่วไป R0-R7 ใน MCS-51 มีอยู่ด้วยกันทั้งหมด 4 กลุ่มกระทำโดยการรีเซตหรือเคลียร์บิต RS0,RS1 ในรีจีสเตอร์ใช้งานเฉพาะ PSW

รีจีสเตอร์ฟังชั่นพิเศษใน MCS-51

ใน MCS-51 รีจีสเตอร์จะใช้หน่วยความจำ RAM ภายในชิป โดยส่วนหนึ่งเป็นรีจีสเตอร์พิเศษ(Special Function Register: SFR) ซึ่งมีทั้งหมด 21 ตัว โดยรีจีสเตอร์พิเศษต่างๆ จะเริ่มที่หน่วยความจำตั้งแต่ 80H ถึง FFH ซึ่งมีทั้งหมด 128 ตำแหน่ง แต่จะเป็นรีจีสเตอร์ฟังก์ชั่นพิเศษเพียง 21 ตำแหน่ง แต่ถ้าเป็น 8032/8051 จะใช้เพียง 26 ตำแหน่งหรือมี SFR 26 ตัว

ตาราง 2.12 รีจีสเตอร์ใช้เฉพาะ PSW (Program Status Word) เข้าถึงข้อมูลในระดับบิต

บิต	ชื่อบิต	ตำแหน่ง	ความหมาย
PSW.7	CY	D7H	Cary Flag
PSW.6	AC	D6H	Auxiliary Flag
PSW.5	F0	D5H	Flag 0
PSW.4	RS1	D4H	บิตสำหรับเลือก รีจีสเตอร์ แบงค์1
PSW.3	RS0	D3H	บิตสำหรับเลือก รีจีสเตอร์ แบงค์0 00 = Bank 0 ;Address 00H – 07H 01 = Bank 1 ;Address 08H – 0FH 02 = Bank 2 ;Address 10H – 17H 03 = Bank 3 ;Address 18H – 1FH
PSW.2	OV	D2H	Overflow Flag
PSW.1	-	D1H	Reserved

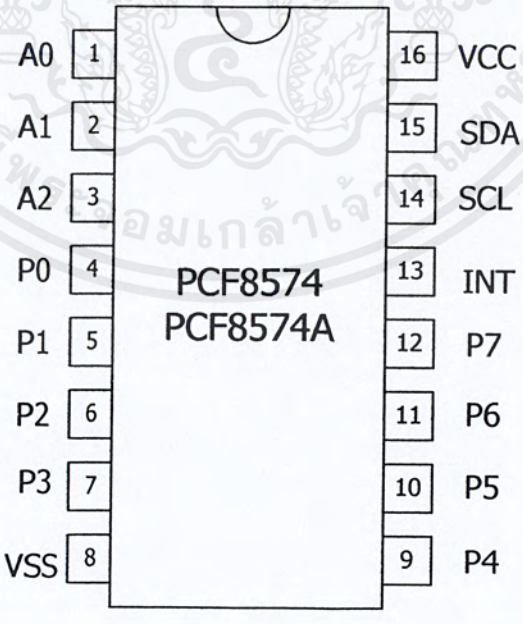
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8 การขยายจำนวนพอร์ตอินพุตเอาต์พุตด้วยไอซี PCF8574A

ข้อมูลเบื้องต้นของ PCF8574Aไอซี PCF8574A มีคุณสมบัติดังนี้

- ทำงานที่ระดับแรงดัน 2.5 V ถึง 6 V
- กินกระแสในสภาวะสแตนด์บายต่ำเพียง 10 μA
- ใช้การเชื่อมต่อแบบบัส I^2C
- มีเอาท์พุทอินเตอร์รัปต์แบบเดรนเปิด
- เอาท์พุทสามารถขับกระแสได้สูงโดยสามารถนำไปขับLEDได้โดยตรงและสามารถแลตค่าได้
- สามารถกำหนดตำแหน่งแอดเดรสของไอซีแต่ละตัวได้ทางฮาร์ดแวร์ ด้วยขา A0-A2 พ่วงกันได้ถึง 8 ตัว

การจัดขาของไอซี PCF8574A แสดงในรูปที่ 3.5 การทำงานของแต่ละขาแสดงในตารางที่ 2.13 ของพอร์ตทั้ง 8 ขาของ PCF8574A สามารถกำหนดให้เป็นอินพุตหรือเอาท์พุตได้โดยอิสระโดยไม่จำเป็นต้องใช้คำสั่งควบคุมเพื่อเลือกให้เป็นขาเอาท์พุตหรืออินพุต ลักษณะวงจรภายในของพอร์ตอินพุตเอาท์พุต ลักษณะภายในของพอร์ตอินพุตเอาท์พุตแสดงในภาพที่ 2.45 เมื่อจ่ายไฟให้กับ PCF8574A ครั้งแรก ขาพอร์ตทั้ง 8 ขา จะมีลอจิกเป็น “1” ซึ่งจะเป็นการจ่ายกระแสมาจากแหล่งจ่ายกระแสคงที่ภายในตัวไอซี ทำให้มีกระแสในขณะลอจิก “1” นี้เพียง 100 μA เท่านั้น ในกรณีที่ต้องการให้มีแหล่งจ่ายกระแสสูงๆ จำเป็นต้องต่อตัวต้านทานพูลอัปไว้ที่ขาพอร์ตเหล่านี้ด้วย



ภาพที่ 2.44 การจัดขาของไอซีขยายพอร์ตอินพุตเอาท์พุต PCF8574/PCF8574A

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อต้องการให้ขาพอร์ตเหล่านี้เป็นพอร์ตอินพุตจะต้องส่งสัญญาณให้ขาเหล่ามีมีลอจิก “1” เสียก่อน เมื่อขาอินพุตได้รับสัญญาณจากภายนอกป้อนเข้ามา ไอซี PCF8574A จะสร้างสัญญาณอินเตอร์รัปต์ (INT) ป้อนให้ไมโครคอนโทรลเลอร์หรือคอมพิวเตอร์รับรู้แทนการต้องคอยตรวจสอบขาอินพุตอยู่ตลอดเวลาสัญญาณอินเตอร์รัปต์นี้จะถูกรีเซตเมื่อมีการอ่านข้อมูลหรือมีการเปลี่ยนแปลงค่าของอินพุตไปสู่ค่าเดิม

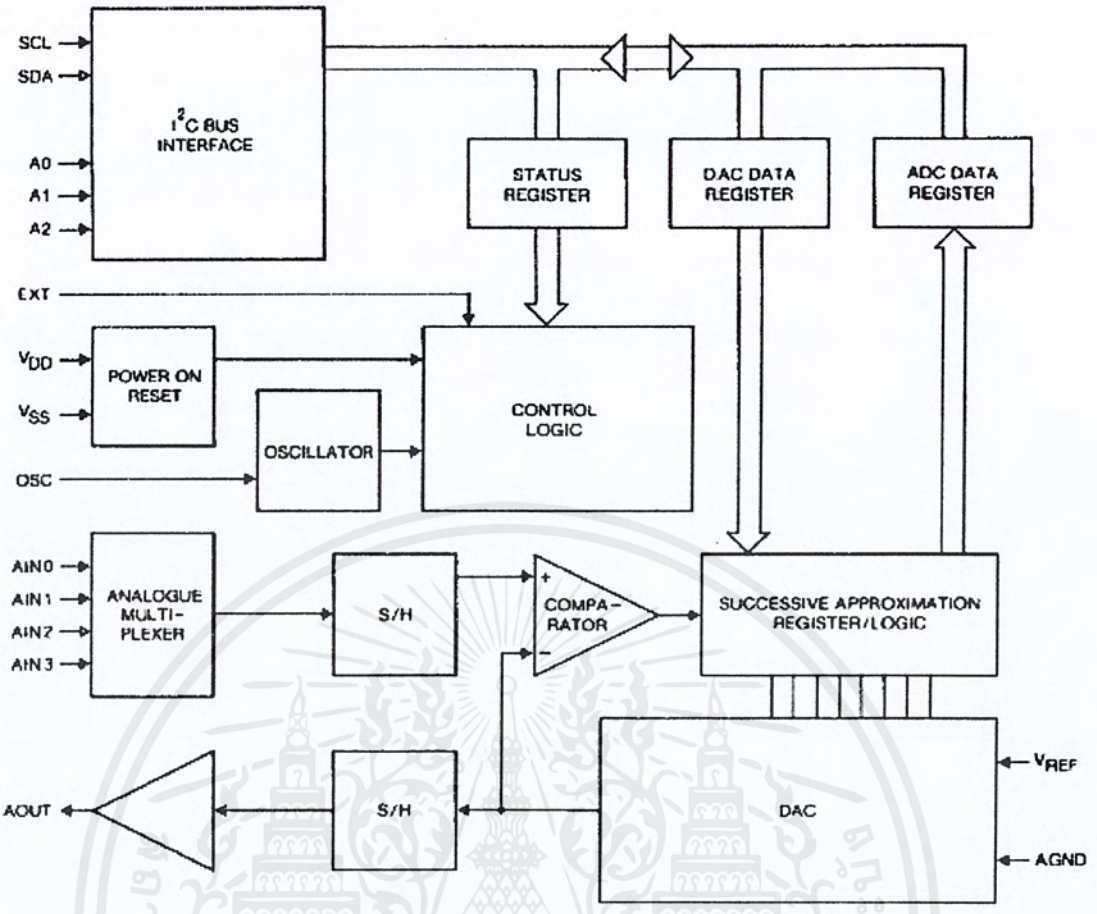
2.8.1 การเขียนโปแกรมควบคุม PCF8574A

เนื่องจาก PCF8574A มีการเชื่อมต่อแบบบัส I^2C ดังนั้นการติดต่อจึงสามารถใช้โปรแกรมย่อยการติดต่อกับอุปกรณ์สเลฟ, การสร้างสภาวะเริ่มต้น, สภาวะหยุด, การอ่านและการเขียนข้อมูล ส่วนที่ต้องเปลี่ยนแปลงสำหรับ PCF8574A คือข้อมูลกำหนดแอดเดรส โดยข้อมูลของ PCF8574A มีรูปแบบดังนี้

บิต 7	บิต6	บิต5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	1	1	A2	A1	A0	R/$\overline{W}$

บิต A0,A1,A2 ใช้ในการระบุ PCF8574A ที่ใช้บนบอร์ดในกรณีที่มีการต่อ PCDF8574 มากกว่า 1 ตัว โดยค่าของ A0-A2 จะมีความแตกต่างกันไปในแต่ละตัว สามารถกำหนดได้ทางฮาร์ดแวร์ โดยการต่อขา A0-A2 เข้ากับไฟเลี้ยง +5V เพื่อกำหนดลอจิกเป็น “1” หรือกราวด์ เพื่อกำหนดเป็นลอจิก “0” ดังนั้นค่าแอดเดรสของ PCF8574A 1ตัว จึงต้องกำหนดสถานะที่กำหนดทางฮาร์ดแวร์ที่ขา A0-A2 มารวมด้วยจึงสมบูรณ์ ส่วนบิต R/W ใช้กำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับไอซี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.45 วงจรภายในขาของพอร์ตของไอซี PCF8574A

ตารางที่ 2.13 รายละเอียดหน้าที่การทำงานของแต่ละขาของไอซี PCF8574A/8574

ชื่อ	ตำแหน่งขา	หน้าที่
A0	1	อินพุตแอดเดรสตัวที่ 1
A1	2	อินพุตแอดเดรสตัวที่ 2
A2	3	อินพุตแอดเดรสตัวที่ 3
P0	4	พอร์ตอินพุตเอาท์พุต 2 ทิศทางบิต 0
P1	5	พอร์ตอินพุตเอาท์พุต 2 ทิศทางบิต 1
P2	6	พอร์ตอินพุตเอาท์พุต 2 ทิศทางบิต 2
P3	7	พอร์ตอินพุตเอาท์พุต 2 ทิศทางบิต 3
VSS	8	กราวด์
P4	9	พอร์ตอินพุตเอาท์พุต 2 ทิศทางบิต 4
P5	10	พอร์ตอินพุตเอาท์พุต 2 ทิศทางบิต 5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 2.13(ต่อ) รายละเอียดหน้าที่การทำงานของแต่ละขาของไอซี PCF8574A/8574

ชื่อ	ตำแหน่งขา	หน้าที่
P6	11	พอร์ตอินพุตเอาท์พุต 2 ทิศทางบิต 6
P7	12	พอร์ตอินพุตเอาท์พุต 2 ทิศทางบิต 7
INT	13	ขาเอาท์พุตอินเตอร์รัปต์(ทำงานที่ลอจิก 0)
SCL	14	ขาสัญญาณนาฬิกาสำหรับ I^2C บัส
SDA	15	ขาข้อมูลสำหรับ I^2C บัส
VDD	16	ไฟเลี้ยง

2.9 การใช้งานไอซี ADC/DAC เบอร์ PCF8591

2.9.1 การแปลงสัญญาณอะนาล็อกดิจิตอลแบบซักเซสซีฟแอปพร็อกซิเมชั่น

การแปลงสัญญาณอะนาล็อกเป็นดิจิตอล (ADC) ที่ได้รับความนิยมสูงและมีประสิทธิภาพดี คือการแปลงแบบแอปพร็อกซิเมชั่น ถ้าจะแปลเป็นไทยอาจจะเรียกกระบวน ADC "เป็นการแปลงแบบประมาณค่าใกล้เคียง"ไดอะแกรมการทำงานแสดงในรูปที่ 3.7 ส่วนสำคัญหลักคือ วงจรเปรียบเทียบแรงดัน, วงจรแปลงสัญญาณดิจิตอลเป็นอะนาล็อกหรือDAC, สัญญาณนาฬิกาและส่วนควบคุมโลจิก

วงจร ADC แบบซักเซสซีฟแอปพร็อกซิเมชั่นนี้ จะใช้รีจีสเตอร์เลขฐานสองหรือไบนารีรีจีสเตอร์ในการส่งข้อมูลดิจิตอลของวงจร DAC ภายใน แต่ละบิตของรีจีสเตอร์จะเซตและรีเซตโดยการควบคุมจากวงจรควบคุมโดยอธิบายการทำงาน ดังนี้

กำหนดให้แรงดันอะนาล็อกอินพุท (Vin) มีค่า 13.5 V

1.ส่งสัญญาณเริ่มต้นการทำงาน (Start Converter) มายังซักเซสซีฟแอปพร็อกซิเมชั่นรีจีสเตอร์

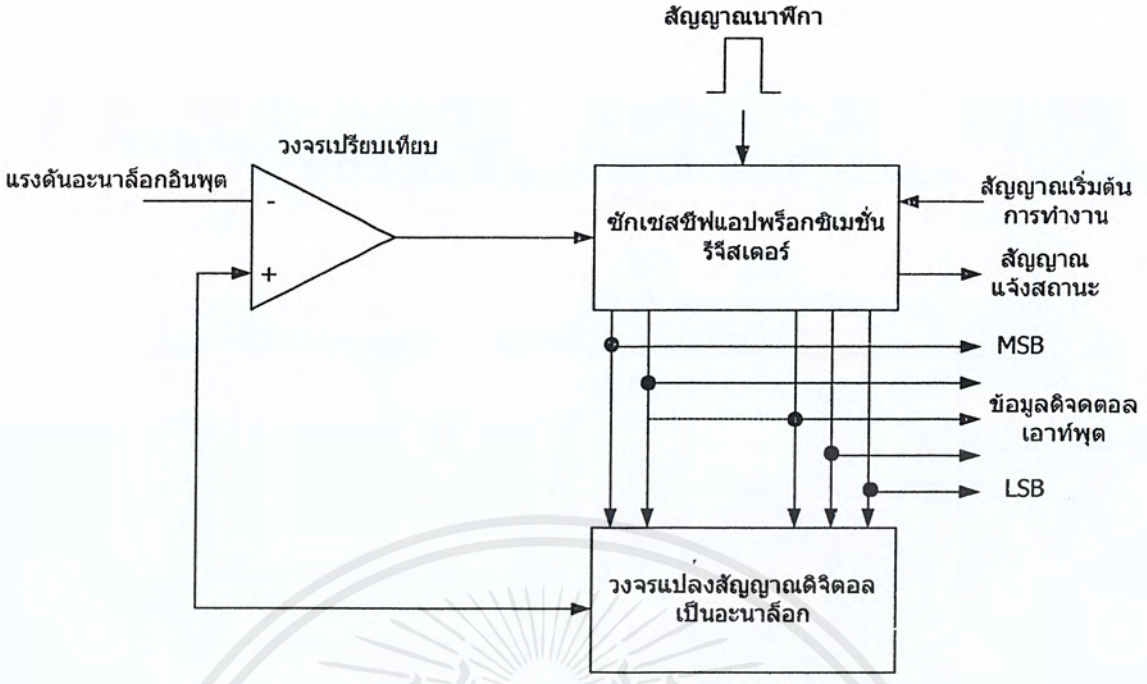
2.ขณะนี้สถานะของรีจีสเตอร์ไม่ว่าง (Busy) สัญญาณนาฬิกาลูกแรกถูกส่งเข้ามาเพื่อกำหนดให้ค่าของรีจีสเตอร์เท่ากับ 0000

3.เอาต์พุตของ DAC จะเป็น 0V ส่งไปในวงจรเปรียบเทียบ เพื่อเปรียบเทียบแรงดันกับ Vin ในขณะนี้จะได้เอาท์พุตเท่ากับ –5V กำหนดเป็นลอจิก "0"

4.เมื่อสัญญาณนาฬิกาลูกต่อไปเข้ามา จะทำการเซตบิต MSB ของรีจีสเตอร์เป็น "1"

5.ในกรณีนี้ ADC ขนาด 4 บิต ดังนั้นการที่บิต MSB เซต จะทำให้วงจร DAC แปลงค่าเป็นแรงดัน 8 V นำไปเปรียบเทียบที่วงจรเปรียบเทียบแรงดัน แต่ก็ยังน้อยกว่า Vin ดังนั้นเอาท์พุตของวงจรเปรียบเทียบยังคงเป็น "0" ทำให้รีจีสเตอร์ยังคงค่าบิต MSB ให้เป็น "1" ต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.46 ไดอะแกรมการทำงานของวงจร ADC แบบซักเซสซีฟแอปพร็อกซิเมชั่น

6.ต่อมาบิต B2 (ถัดจากบิต MSB 1บิต เนื่องจากมี 4 บิต กำหนดบิต MSB=B3) จะเซต ซึ่งจะมีค่าเท่ากับ 4V นำไปรวมกับค่าของบิต MSB ที่มีอยู่ 8V เช่น 12V นำไปเปรียบเทียบกับ Vin ก็ยังน้อยกว่า รีจีสเตอร์จึงยังคงค่า B2 ไว้ที่ "1" เช่นกัน

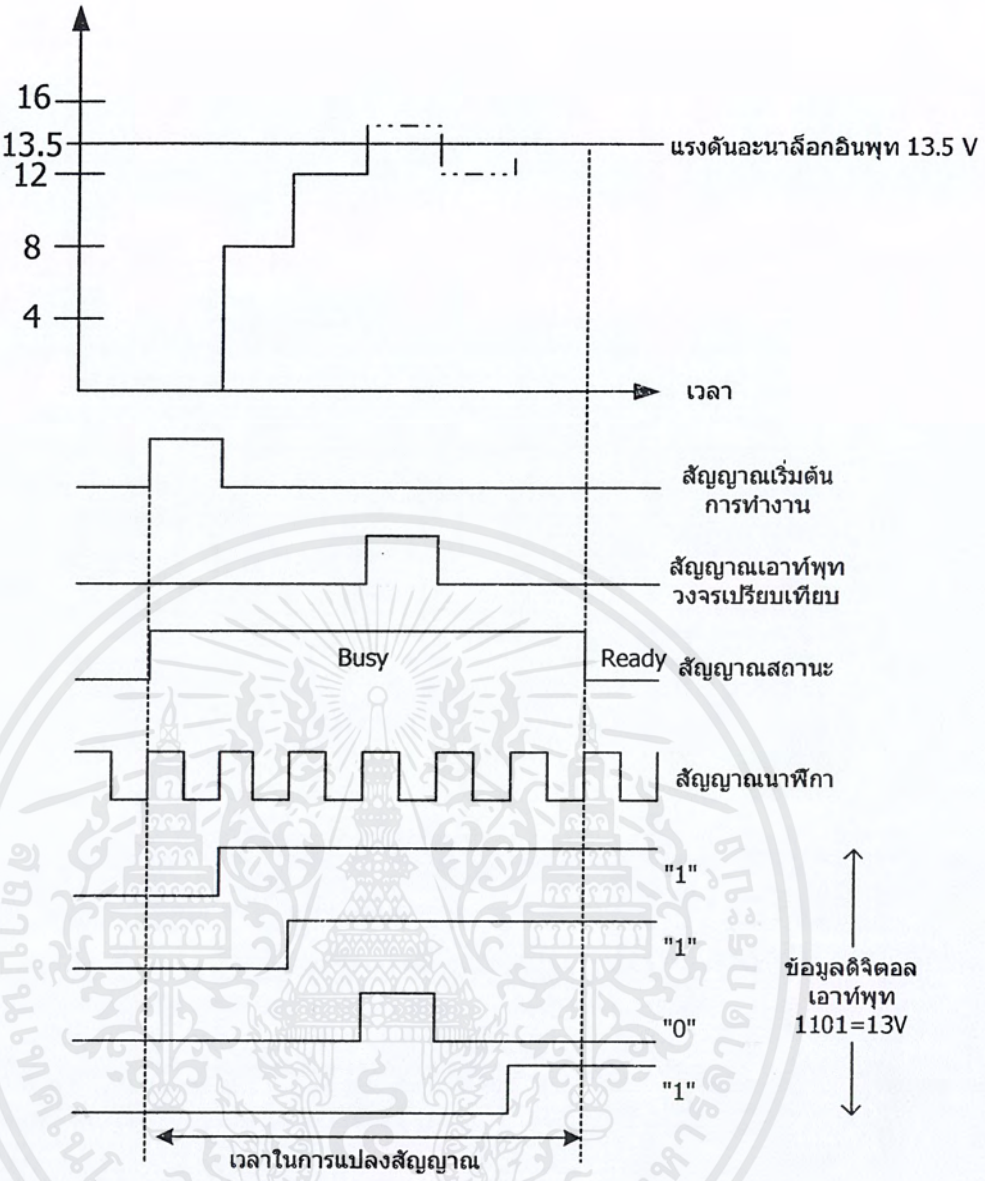
7.ต่อมาบิต B1 จะเซตให้แรงดันเอาท์พุตมา DAC กลายเป็น 8+4+12 = 14 V ซึ่งมากกว่า Vin ทำให้วงจเปรียบเทียบเกิดการเปลี่ยนสถานะเป็น "1" ซึ่งส่งสัญญาณมาควบคุมให้ B1 กลายเป็น "0"

8.เมื่อบิต LSB ถูกเซตจะมีค่าแรงดัน 1V เข้ามารวมกับค่าของ B3,B2 และ B1 เป็น 8+4+0+1 = 13 V นำไปเปรียบเทียบกับ Vin ปรากฏว่าน้อยกว่า Vin ทำให้ที่บิต B0 หรือ LSB มีค่าเป็น "1"

9.ขณะนี้ทุกบิตในรีจีสเตอร์ถูกนำมาแปลงค่าเรียบร้อยแล้ว ทำให้สถานะของรีจีสเตอร์กลับมาเป็น "พร้อมทำงาน (Ready)"

10.ข้อมูลดิจิตอลที่ได้จากการ ADC แบบนี้ จะมีค่า 1101_2 หรือ 13V ซึ่งใกล้เคียงกับ Vin 13.5 V มากที่สุด ถ้าหากรีจีสเตอร์มีจำนวนบิตมากกว่านี้ ความละเอียดของข้อมูลที่แปลงได้จะมีความใกล้เคียงมากขึ้น ช่วงเวลาของการแปลงสัญญาณจะเริ่มต้นตั้งแต่สัญญาณนาฬิกาลูกแรกถูกส่งเข้าไปเตรียมระบบ ไปจนถึงเมื่อสภาวะของ รีจีสเตอร์กลับมาเป็น "พร้อมทำงาน" อีกครั้งหนึ่ง ซึ่งจะต้องใช้จำนวนสัญญาณนาฬิกาเท่ากับ n+1 พัลส์ โดย n เท่ากับจำนวนบิตของรีจีสเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.47 ไดอะแกรมเวลาแสดงการทำงานของวงจร ADC แบบซักเซสซีฟแอปพร็อกซิเมชั่น

2.9.2 ความเที่ยงตรงของวงจร ADC

เป็นการเปรียบเทียบแรงดันอะนาล็อกของวงจร ADC กับแรงดันที่ควรจะเกิดขึ้นจริง ยกตัวอย่างที่ข้อมูลดิจิตอลสูงสุดของวงจร ขนาด 8 บิตเมื่อเทียบเป็นแรงดันอะนาล็อกควรมีค่าเท่ากับ 5.0000 V แต่จากการคำนวณในตัวอย่างก่อนหน้านี้ได้ค่าแรงดัน 4.9804 V นั่นคือเกิดความผิดพลาดไป 0.0195 V แต่การบอกค่าความเที่ยงตรงของวงจร ADC มักจะระบุเป็นจำนวนที่เทียบกับ VLSB ดังนั้นในวงจร ADC ขนาด 8 บิต ที่ยกตัวอย่างนี้จะมีค่าความเที่ยงตรง (หรือบางทีเรียกเป็นค่าความผิดพลาด) เป็น ±1/2LSB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.9.3 ค่าเวลาในการแปลงสัญญาณ

เป็นค่าของเวลาที่วงจรADC แบบวงจรนับแรมป์และแบบซักเซสซีฟแอปพร็อกซิเมชั่น ใช้ในการแปลงสัญญาณอะนาล็อกเป็นดิจิตอลจนเสร็จสิ้นพารามิเตอร์ตัวนี้มักจะปรากฏอยู่ในคุณสมบัติของไอซีทำงานเป็นวงจร ADC เมื่อไอซีแปลงสัญญาณเสร็จสิ้นลง จะส่งสัญญาณที่เรียกว่า EOC (End Of Conversion) ออกมา

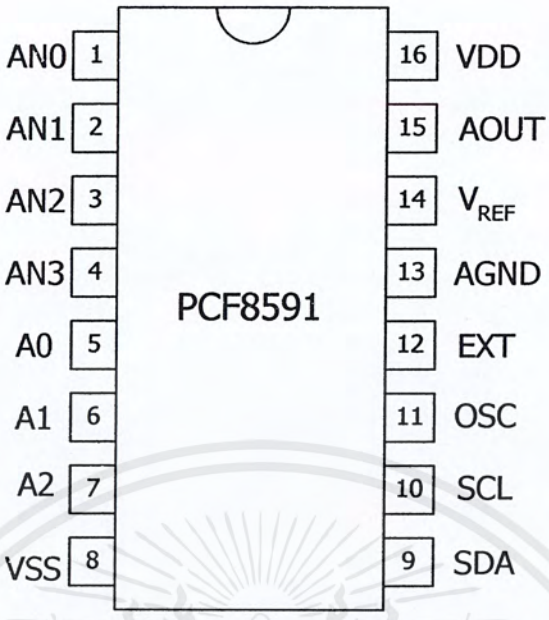
ค่าเวลาในการแปลงสัญญาณของวงจร ADC จะขึ้นอยู่กับจำนวนบิตของวงจร, ค่าความถี่ของสัญญาณนาฬิกาที่ใช้ในการแปลงสัญญาณและขนาดของสัญญาณอะนาล็อกอินพุต

2.9.4 ข้อมูลเบื้องต้นของ PCF8591

เนื่องจากไอซีเบอร์นี้จะมีวงจร ADC แบบซักเซสซีฟแอปพร็อกซิเมชั่นขนาด 8 บิตสูงถึง 4 ช่อง ทั้งยังมีวงจร DAC อีก 1 ช่องด้วยระบบการเชื่อมต่อเป็นแบบบัส I2C ทำให้ใช้สายสัญญาณเพียง 2 เส้น ทั้งยังสามารถเชื่อมต่อกันสูงสุดได้ถึง 8 ตัว ทำให้ได้วงจร ADC รวมสูงสุดถึง 32 ช่อง และวงจร DAC รวม 4 ช่อง สามารถนำไปประยุกต์ใช้งานได้อย่างกว้างขวาง มีรายละเอียดคุณสมบัติดังนี้

- ทำงานโดยใช้แหล่งจ่ายไฟชุดเดียว
- ทำงานที่แรงดัน 2.5 ถึง 6 V
- กินกระแสขณะสแตนบายต่ำ
- ติดต่อกับไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ผ่านระบบบัส I^2C
- เลือกตำแหน่งแอดเดรสทางฮาร์ดแวร์จากขา A0,A1,A2 ทำให้สามารถต่อพ่วงกันสูงสุดได้ถึง 8 ตัว
- อัตราการส่งข้อมูล (Sampling) ขึ้นอยู่กับความเร็วของสัญญาณนาฬิกาบนบัส I^2C
- วงจรแปลงสัญญาณอะนาล็อกเป็นดิจิตอล (ADC) สามารถรับสัญญาณอะนาล็อกได้ 4 ช่อง ทั้งยังเลือกได้ว่าทำงานแบบแยกช่องหรือทำงานแบบวงจร ดิฟเฟอร์เรนเชียล
- การอ่านค่าสามารถกำหนดให้เลื่อนช่องอินพุตโดยอัตโนมัติได้
- สัญญาณอะนาล็อกมีระดับแรงดันตั้งแต่ Vss ถึง VDD
- วงจรแปลงสัญญาณอะนาล็อกเป็นดิจิตอลเป็นแบบซักเซสซีฟแอปพร็อกซิเมชั่น ขนาด 8 บิต
- มีวงจรแปลงสัญญาณดิจิตอลเป็นอะนาล็อกขนาด 8 บิต 1 ช่อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



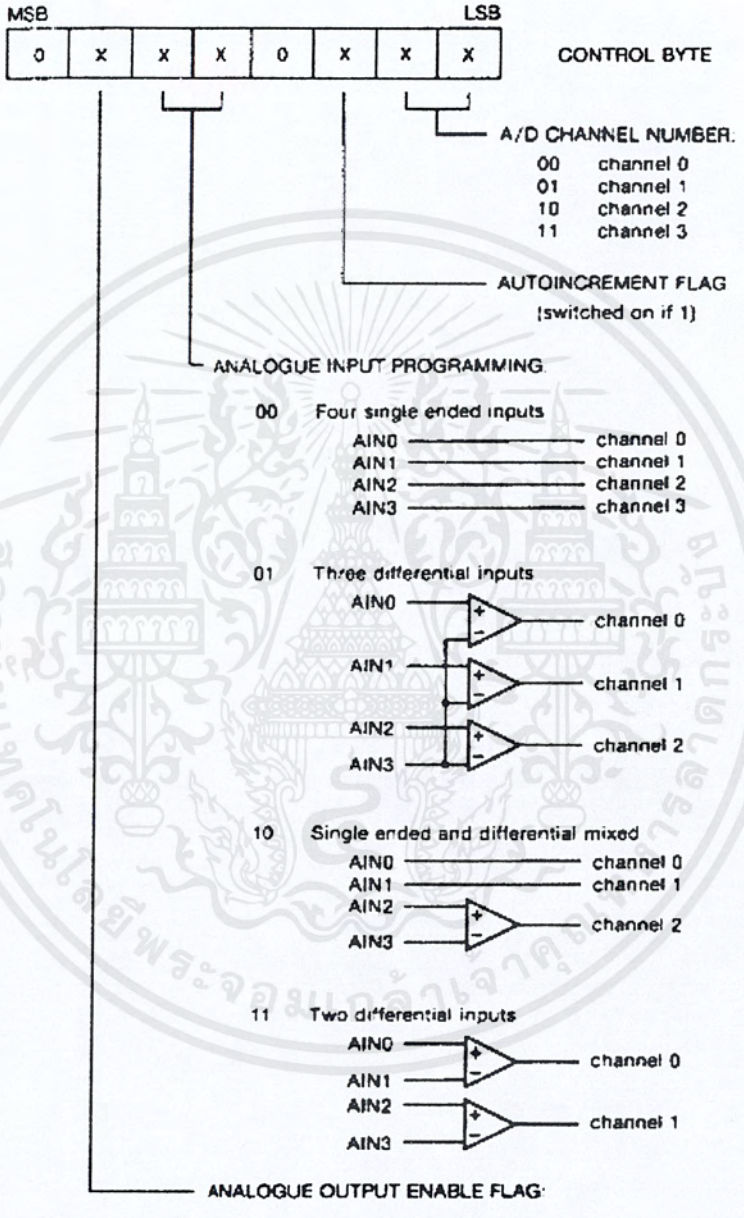
ภาพที่ 2.48 การจัดขาของไอซี ADC/DAC ขนาด 8 บิตผ่านบัส I^2C เบอร์ PCF8591

PCF8591 สามารถทำหน้าที่เป็นไอซีแปลงสัญญาณอะนาล็อกเป็นดิจิตอลขนาด 8 บิต 4 ช่อง และทำหน้าที่เป็นไอซีแปลงสัญญาณดิจิตอลเป็นอะนาล็อกได้ในคราวเดียวกัน ด้วยการควบคุมผ่านระบบบัส I^2C ทำให้สามารถต่อพ่วงไอซี PCF8591 ได้สูงสุดถึง 8 ตัว รองรับการอ่านค่าสัญญาณอะนาล็อกอินพุตได้สูงสุดถึง 32 ช่อง และสามารถส่งสัญญาณอะนาล็อกเอาต์พุตสูงสุดได้ถึง 8 ช่อง ด้วยการกำหนดแอดเดรสจากขา A0,A1 และ A2 การจัดขาของ PCF8591 แสดงในภาพที่ 2.48 ส่วนรายละเอียดตำแหน่งขาต่าง ๆ มีดังนี้

- ขา AN0-AN3 (ขา 1-4) เป็นขาอินพุตสำหรับป้อนสัญญาณอะนาล็อกที่ต้องการแปลงค่า
- ขา A0-A2 (ขา 5-7) เป็นขาสำหรับกำหนดข้อมูลแอดเดรสทางฮาร์ดแวร์ปกติจะทำการต่อลงกราวน์ แต่ถ้ามีการใช้งาน PCF8591 มากกว่า 1 ตัว ต้องกำหนดการต่อขา A0-A2 ของ PCF8591 ให้ไม่ตรงกัน จึงทำให้สามารถใช้งานได้สูงสุดถึง 8 ตัว
- ขา Vss (ขา 8) เป็นขาต่อกราวน์
- ขา SDA,SCL (ขา 9 และ 10) เป็นขาเชื่อมต่อระบบบัส I^2C
- ขา OSC (ขา 11) เป็นขาสำหรับการต่อสัญญาณนาฬิกาภายนอก เมื่อขา EXT ต่อกับไฟ+5V และจะทำงานเป็นขาเอาท์พุตสัญญาณนาฬิกาถ้าขา EXT ต่อลงกราวด์
- ขา EXT (ขา 12) เป็นขาสำหรับเลือกแหล่งกำเหนิดสัญญาณนาฬิกา ถ้าต่อไฟ +5V จะเป็นการเลือกใช้สัญญาณนาฬิกาจากภายนอก โดยต่อสัญญาณนาฬิกาเข้าที่ขา OSC ถ้าต่อขานี้ลงกราวด์ จะเป็นการเลือกสัญญาณนาฬิกาจากภายใน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ขา AGND (ขา 13) เป็นขากราวด์ของแรงดันอ้างอิง ปกติต่อวงกราวด์
- ขา V_{REF} (ขา 14) เป็นขาสำหรับป้อนแรงดัน ปกติต่อไฟเลี้ยง +5V
- ขา AOUT (ขา 15) เป็นขาเอาท์พุตของวงแปลงสัญญาณดิจิตอลเป็นอะนาล็อก
- ขา V_{DD} (ขา 16) เป็นขาต่อไฟเลี้ยง จ่ายได้ตั้งแต่ +2 ถึง +6 V ปกติใช้+5V



ภาพที่ 2.49 รายละเอียดข้อมูลควบคุมที่เขียนลงในรีจีสเตอร์ควบคุมภายในไอซี PDF8591

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.9.5 รายละเอียดฟังก์ชั่นต่าง ๆ ของ PCF8591

ตำแหน่งแอดเดรส ในระบบบัส I^2Cการติดต่ออุปกรณ์แต่ละตัวต้องระบุแอดเดรสของอุปกรณ์เห่านั้นอย่างชัดเจน ถ้าเป็นการอ้างถึง 7 บิตข้อมูลกำหนดแอดเดรส 4 บิตบนจะเป็นค่าแอดเดรสเฉพาะอุปกรณ์ตัวนั้นๆ ที่กำหนดมาจากผู้ผลิต ผู้ใช้งานไม่สามารถเปลี่ยนแปลงได้ สำหรับไอซี PCF8591 จะมีค่าเท่ากับ 1001 (ฐานสอง) ข้อมูล 3 บิตถัดมาจะเป็นค่าของแอดเดรสที่ผู้ใช้งานสามารถกำหนดได้ทางฮาร์ดแวร์เพื่อเลือกไอซีPCF8591ที่ต้องการติดต่อด้วยในกรณีที่การต่อใช้งาน PCF8591 มากกว่า 1 ตัว ส่วนบิต LSB ใช้ในการกำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับไอซีตัวนั้น ๆ

ข้อมูลควบคุม

หลังจากส่งข้อมูลกำหนดแอดเดรสให้แก่ PCF8591 แล้ว ต้องส่งข้อมูลควบคุมตามไปด้วยเพื่อกำหนดคุณสมบัติของวงจรแปลงสัญญาณอะนาล็อกเป็นดิจิตอลและวงจรแปลงสัญญาณดิจิตอลเป็นอะนาล็อกภายใน PCF8591 โดยมีรายละเอียดข้อมูลในแต่ละบิตดังในภาพที่ 2.49

บิต 6 ของข้อมูลควบคุมใช้สำหรับการเอ็นเอเบิลขาอะนาล็อกเอาท์พุต เมื่อต้องกำหนดให้ขานี้เป็น “1”

บิต 4 และบิต 5 ของข้อมูลควบคุมใช้สำหรับการกำหนดรูปแบบของสัญญาณอะนาล็อกอินพุตที่ป้อนให้แก่ PCF8591

บิต 2 ใช้สำหรับเลือกรูปแบบการอ่านข้อมูลจากขาอินพุตอะนาล็อกว่าจะเป็นการอ่านจากอินพุตเดียวหรืออ่านแบบเรียงลำดับทุกอินพุท ถ้าต้องการเลือกให้อ่านแบบเรียงลำดับต้องกำหนดให้บิตนี้เป็น “1”

บิต 0 และบิต 1 ใช้สำหรับกำหนดช่องของอินพุทอะนาล็อกที่ต้องการอ่าน ถ้าต้องการกำหนดให้บิต 2 เป็น “1” หลังจากอ่านค่าของ “0” และบิต “1” แล้วในการอ่านอ่านครั้งต่อไปจะเป็นการอ่านค่าอินพุทจากช่องที่ 1 ข้อมูลควบคุมทั้งหมดจะถูกเก็บไว้ในรีจีสเตอร์ควบคุมภายในรีจีสเตอร์ภายใน PCF8591 เมื่อจ่ายไฟให้แก่ PCF8591 ครั้งแรกบิตต่างๆ ของข้อมูลภายในรีจีสเตอร์ควบคุมจะเป็น “0”

ออสซิลเลเตอร์

วงจรออสซิลเลเตอร์ภายใน PCF8591 จะสร้างสัญญาณนาฬิกาสำหรับการแปลงสัญญาณอะนาล็อกเป็นดิจิตอล เมื่อต้องการใช้วงออสซิลเลเตอร์ภายใน ขา EXT ต้องต่อลงกราวด์ ถ้าต้องการใช้ออสซิลเตอร์จากภายนอก ขา EXT ต้องต่อกับไฟบวก และป้อนสัญญาณนาฬิกาเข้าที่ขา OSC ของ PCF8591 โดยความถี่ของสัญญาณนาฬิกาสูงสุดที่ป้อนให้กับออสซิลเลเตอร์เท่ากับ 1.25 MHz

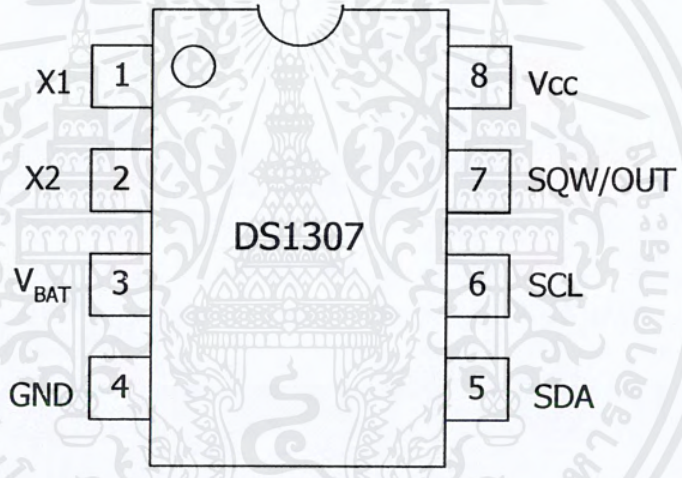
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.10 การใช้งานไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก (RTC)

2.10.1 คุณสมบัติทางเทคนิคของ DS1307 ไอซีสร้างฐานเวลา

- เป็นไอซีรีลไทม์คล็อก ให้ข้อมูลตั้งแต่วินาทีจนถึงปี รวมถึงการปรับวันในปีอธิกสุรด้วย สามารถให้ข้อมูลเวลาได้อย่างเที่ยงตรงถึงปีคริสตศักราช 2100
- มีหน่วยความจำนอนโวลาไทล์แรม 56 ไบต์อยู่ภายในสามารถใช้เก็บข้อมูลทั่วไปได้ใช้การเชื่อมต่อแบบระบบบัส I^2C
- มีวงจรตรวจจับไฟเลี้ยงต่ำหรือหายไปอย่างอัตโนมัติ และสามารถรักษาข้อมูลเวลาไว้ได้แม้ไม่มีไฟเลี้ยงไอซี

2.10.2 รายละเอียดขาต่อใช้งานของ DS1307



ภาพที่ 2.50 การจัดขาของไอซี DS1307 ไอซีสร้างฐานเวลาจริง

หน้าที่การทำงานของแต่ละขา

Vcc,GND (ขา 8,4) ต่อกับไฟเลี้ยง +5V

V_{BAT} (ขา3) ใช้ต่อกับแบตเตอรี่ 3V เพื่อรักษาการทำงานของวงจรสร้างฐานเวลาของ DS1307 ให้คงอยู่ต่อไป แม้ว่าไม่มีไฟเลี้ยงจ่ายให้แก่ DS1307 ชนิดแบตเตอรี่ที่ เหมาะสมคือแบตเตอรี่แบบลิเที่ยม ซึ่งมีความจุ 40mAhr หรือมากกว่า จะสามารถรักษาข้อมูลได้นานถึง 10 ปี ที่อุณหภูมิ 25 องศาเซลเซียส

SDA,SCL (ขา5 และ 6)เป็นขาสำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์บนระบบบัส I^2C

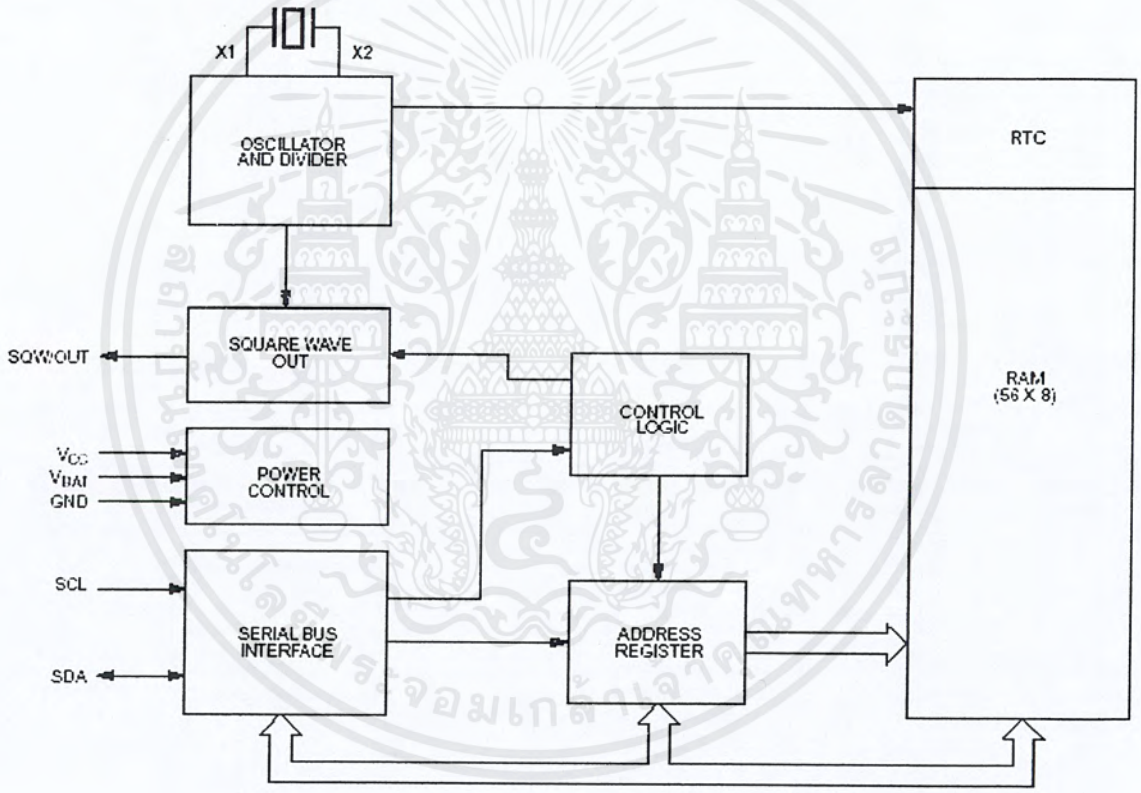
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$\overline{\text{SQW OUT}}$ (ขา 7) ที่ขานี้จะมีสัญญาณพัลส์ออกมา โดยสามารถเลือกความถี่ได้ 1 KHz, 4.096 KHz, 8.192 KHz และ 32 KHz ในการต่อใช้งานต้องต่อตัวต้านทาน 1k พูลอัปไว้ที่ขานี้ด้วย

X1,X2 (ขา1 และ ขา2) ใช้ต่อกับความถี่คริสตอลความถี่มาตรฐาน 32.768 KHz เพื่อใช้เป็นฐานเวลาในการสร้างค่าเวลาจริงในการใช้งานต้องต่อคริสตอลเข้ากับขาทั้งสองนี้และที่แต่ละขาต้องต่อตัวเก็บประจุค่าต่ำๆ ประมาณ 15pF คร่อมกราวด์ด้วย

2.10.3 การทำงานของ DS1307

ไอซี DS1307 จัดการเชื่อมต่อในระบบบัส I^2C โดยจะทำงานเป็นอุปกรณ์สเลฟเสมอ ดังนั้นการติดต่อเพื่อใช้งานจึงต้องกำหนดรูปแบบตามที่กำหนดไว้ในการติดต่อแบบบัสI^2C ในภาพที่ 2.51 แสดงส่วนประกอบหลักที่สำคัญและไดอะแกรมการทำงานของ DS1307



ภาพที่ 2.51 โครงสร้างภายในของไอซีรีลไทม์คล็อกเบอร์ DS1307

วงจรออสซิลเลเตอร์ถือเป็นหัวใจหลักของไอซี เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลเวลาจริง ในขณะที่ DS1307 ทำงานที่ขา SQW/OUT จะมีสัญญาณพัลส์ส่งออกมาตลอดเวลาในกรณีที่มีการเอ็นเอเบิลวงจรกำเนิดสัญญาณพัลส์ที่รีจีสเตอร์ควบคุม ค่าความถี่ของสัญญาณสามารถเลือกได้ 4 ค่า คือ 1 KHz, 4.096 KHz, 8.192 KHz และ 32 KHz พร้อมกันนั้นจะมีการเก็บค่าของเวลาไว้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในหน่วยความจำนอนโวลาไทล์แรม ซึ่งมีขนาดรวม 64 ไบต์ แต่จัดสรรให้ใช้เก็บข้อมูลเวลา 8 ไบต์ และเป็นหน่วยความจำสำหรับเก็บข้อมูลทั่วไปสำหรับผู้ใช้งานอีก 56 ไบต์

วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสภาวะของไฟเลี้ยงไอซี หากไฟเลี้ยงต่ำกว่า 1.25 x V_{BAT} ก็จะควบคุมให้ DS1307 หยุดการทำงาน รีเซตค่าตัวนับแอดเดรสภายใน ทำให้ไม่สามารถติดต่อกับ DS1307 ได้ ดังนั้นในการใช้งาน DS1307 ต้องระมัดระวังอย่าให้ไฟเลี้ยงตกต่ำกว่า 1.25 x V_{BAT} หรือประมาณ 3.75 โวลท์ ในกรณีที่ใช้ V_{BAT} เท่ากับ 3โวลท์ ถ้าหากไฟเลี้ยงมีค่าต่ำกว่า V_{BAT} ไอซี DS1307 จะเข้าสู่โหมดสำรองกระแสต่ำทันที จะไม่มีการส่งสัญญาณพัลส์ออกมาที่ขา SQW/OUT แต่วงจรสร้างฐานเวลายังคงทำงานเพื่อให้ค่าของเวลาเดินไปอย่างไม่ผิดพลาด เมื่อมีไฟเลี้ยงปรากฏขึ้นอีกครั้ง DS1307 ก็สามารถให้ค่าของเวลาที่เป็นจริงแก่ผู้ใช้งานต่อไป

วงจรสื่อสารอนุกรมภายใน DS1307 ได้รับการกำหนดให้ทำงานตามรูปแบบของบัส I^2C เป็นช่องทางการสื่อสารระหว่างDS1307กับอุปกรณ์มาสเตอร์ผู้ใช้งานสามารถเข้าถึงหน่วยความจำที่ใช้เก็บค่าเวลาและหน่วยความจำใช้งานโดยการเขียนข้อมูลตามรูปแบบที่กำหนดในระบบบัส I^2C

2.10.4 การจัดสรรหน่วยความจำใน DS1307

ในภาพที่ 2.52 (ก) แสดงการจัดสรรพื้นที่ของหน่วยความจำภายใน DS1307 พื้นที่ 7 ไบต์แรกตั้งแต่แอดเดรส 00H-06H เป็นพื้นที่ของรีจีสเตอร์ค่าเวลาใช้ในเก็บข้อมูลเกี่ยวกับเวลาไบต์ต่อมาที่แอดเดรส 07H เป็นพื้นที่ของรีจีสเตอร์ของรีจีสเตอร์ควบคุมการทำงานของการควบคุมการทำงานของ DS1307 ในภาพที่ 2.52 (ข) แสดงรายละเอียดของรีจีสเตอร์ค่าเวลาและรีจีสเตอร์ค่าเวลาและรีจีสเตอร์ค่าเวลาและรีจีสเตอร์ควบคุมของ DS1307

ด้วยการจัดสรรพื้นที่แบบนี้ ทำให้ผู้ใช้งานสามารถเรียกข้อมูลเวลาออกมาได้ตามที่ต้องการโดยไม่จำเป็นต้องอ่านค่าทั้งหมดก็ได้ ซึ่งค่าของเวลาทั้งหมดจะอยู่ในรูปของเลขฐานสิบ สำหรับการแสดงผลเวลาในรูปของชั่วโมงสามารถเลือกได้ว่าต้องการแบบ 12 หรือ 24 ชั่วโมง โดยที่บิต 6 ของแอดเดรส 02H และเมื่อเลือกแบบ 12 ชั่วโมง ที่บิต 5 ในแอดเดรสเดียวกันจะใช้ในการแสดงค่า AM/PM โดยถ้าบิตนี้เป็น “1” หมายถึง ค่าชั่วโมงในขณะนี้เป็นช่วงเวลาหลังเที่ยงคืน ในกรณีที่เป็นแบบ 24 ชั่วโมง บิตนี้จะใช้ในการแสดงค่า 2 ของหลักสิบในหน่วยชั่วโมง

2.10.5 รีจีสเตอร์ควบคุม

มีแอดเดรสอยู่ที่ 07H มีรายละเอียดของแต่ละบิตดังนี้

OUT (Output control) : ใช้ในการควบคุมระดับลอจิกที่ขา $\overline{\text{SQW OUT}}$ ในกรณีที่ดิสเอเบิลการกำเนิดสัญญาณสี่เหลี่ยม โดยถ้าบิตนี้เป็น “1” ที่ขา $\overline{\text{SQW OUT}}$ ก็จะเป็น “1” ถ้าบิตนี้เป็น “0” ที่ขา $\overline{\text{SQW OUT}}$ ก็จะเป็น “0”

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

00H	วินาที
	นาที
	ชั่วโมง
	วัน
	วันที่
	เดือน
	ปี
07H	รีจีสเตอร์ควบคุม
08H	แรม 56 ไบต์
3FH	

(ก)

บิต7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต1	บิต0	ค่าของข้อมูล
CH	ข้อมูลวินาที(หลักสิบ)			ข้อมูลวินาที(หลักหน่วย)				00-59
x	ข้อมูลนาที(หลักสิบ)			ข้อมูลนาที(หลักหน่วย)				00-59
x	12 ชั่วโมง / 24 ชั่วโมง	ชั่วโมง(หลักสิบ) / AM/PM	ข้อมูลชั่วโมง (หลักสิบ)	ข้อมูลชั่วโมง (หลักหน่วย)				01-12 / 00-23
x	x	x	x	x	ข้อมูลวันในสัปดาห์			1-7
x	x			ข้อมูลวันที่ (หลักหน่วย)				01-28/29 01-30 01-31
x	x	x	ข้อมูลเดือน (หลักสิบ)	ข้อมูลเดือน (หลักหน่วย)				01-12
ข้อมูลปี (หลักสิบ)				ข้อมูลปี (หลักหน่วย)				00-99
OUT	x	x	SQWE	x	x	RS1	RS1	

(ข)

ภาพที่ 2.52 (ก) การจัดสรรหน่วยความจำแรมภายใน DS1307
(ข) รายละเอียดของรีจีสเตอร์เก็บค่าเวลาและรีจีสเตอร์ควบคุมของ DS1307

SQWE (Square Wave Enable) :ใช้ในการเอ็นเอเบิลวงจรกำเนิดสัญญาณพัลส์ที่ขา SQW OUT ถ้าต้องการให้มีสัญญาณพัลส์ออกให้กำหนดบิตนี้เป็น "1"

RS1,RS0 (Rate Select) : ใช้ในการเลือกสัญญาณพัลส์ที่ออกจากขา SQW OUT ซึ่งมีรายละเอียดต่อไปนี้

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

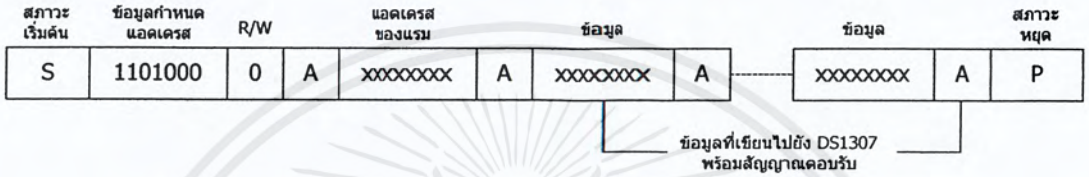
2.10.6 โหมดการทำงานของ DS1307

มีด้วยกัน 2โหมดคือ โหมดเขียนข้อมูลและโหมดการอ่านข้อมูล ในการใช้งาน DS1307 ตามปกติจะใช้งานเฉพาะโหมดอ่านข้อมูลเท่านั้นเนื่องจากไมโครคอนโทรลเลอร์จะติดต่อกับ DS1307 เพื่อการอ่านข้อมูลของเวลาไปใช้งาน โหมดการเขียนข้อมูลจะถูกใช้งานก็ต่อเมื่อต้องการตั้งค่าเวลาใหม่และต้องการเขียนข้อมูลลงในหน่วยความจำทั่วไป อย่างไรก็ตามเมื่อเริ่มติดต่อกับ DS1307 จำเป็นอย่างยิ่งที่จะต้องเข้าสู่โหมดการเขียนข้อมูลก่อนเพื่อกำหนดแอดเดรสที่ต้องการอ่านข้อมูล จากนั้นจึงเปลี่ยนโหมดการทำงานมาเป็นโหมดการอ่านข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.10.7 โหมดการเขียนข้อมูล

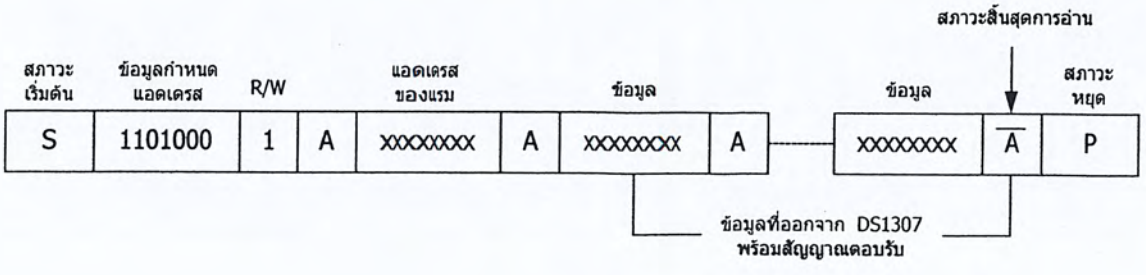
มีรูปแบบดังภาพที่ 2.53 เริ่มต้นเมื่อไมโครคอนโทรลเลอร์ทำการกำหนดสภาวะเริ่มต้น (Start :S) จากนั้นส่งข้อมูลกำหนดแอดเดรส 1101000 ตามด้วยข้อมูลเลือกการเขียน นั่นคือค่า 0 จากนั้นจะรอคอยการตรวจรับจาก DS1307 ขั้นตอนต่อมาคือ ส่งข้อมูลเพื่อเลือกแอดเดรสที่ต้องการเขียน จากนั้นรอคอยการตอบรับจาก DS1307 เมื่อมีการตอบรับมาเรียบร้อย ก็เริ่มทยอยเขียนข้อมูลส่งไปทีละแอดเดรส หลังจากเขียนข้อมูลในแต่ละแอดเดรส จะต้องหยุดรอการตอบรับจาก DS1307 ทุกครั้งจึงสามารถเขียนข้อมูลต่อไปได้ เมื่อเขียนเรียบร้อยแล้วให้ส่งสภาวะหยุด (Stop :P) เป็นอันสิ้นสุดกระบวนการเขียนข้อมูล



ภาพที่ 2.53 รูปแบบของข้อมูลสำหรับติดต่อกับ DS1307 ในโหมดการเขียนข้อมูล

2.10.8โหมดการอ่านข้อมูล

มีรูปแบบแสดงดังภาพที่ 2.54 เริ่มต้นการทำงานเหมือนกับโหมดการเขียนข้อมูล ไมโครคอนโทรลเลอร์กำหนดสภาวะเริ่มต้นแล้วส่งข้อมูลกำหนดแอดเดรสตามด้วยข้อมูลเลือกการอ่าน ซึ่งเท่ากับ 1 จากนั้นรอการตอบรับจาก DS1307 เมื่อตอบรับเรียบร้อย DS1307 จะทยอยส่งข้อมูลออกมาให้ไมโครคอนโทรลเลอร์ครั้งละ 1 แอดเดรส หรือ 1 ไบต์ โดยแอดเดรสที่เลือกอ่านข้อมูลจะต้องมีการกำหนดมาก่อนล่วงหน้าด้วยโหมดการเขียนข้อมูลวิธีการง่ายๆ คือ เข้าสู่โหมดการเขียนข้อมูลก่อน เมื่อถึงจังหวะที่ต้องเขียนข้อมูล ให้ทำการสร้างสภาวะเริ่มต้นและส่งข้อมูล แอดเดรสใหม่อีกครั้ง ตามด้วยเลือกโหมดการอ่านข้อมูล ข้อมูลที่ออกมาจาก DS1307 ก็จะเป็น ข้อมูลจาก แอดเดรสที่กำหนดไว้ก่อนหน้านี้



ภาพที่ 2.54 รูปแบบของข้อมูลสำหรับติดต่อกับ DS1307 ในโหมดการอ่านข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.11 การเชื่อมต่อกับไอซีตรวจจับอุณหภูมิ

2.11.1 ระบบการสื่อสารข้อมูลอนุกรมแบบหนึ่งสาย (1- Wire™ Serial Bus)

ระบบการสื่อสารข้อมูลแบบนี้จะใช้สายสัญญาณเพียง 1 เส้น โดยไม่ต้องมีสายสัญญาณนาฬิกามาควบคุมจังหวะการถ่ายทอดข้อมูลเหมือนกับระบบการสื่อสารข้อมูลแบบอื่นๆ เนื่องจากสายข้อมูลนั้นจะทำหน้าที่เสมือนหนึ่งเป็นสายสัญญาณนาฬิกาในตัวส่วนค่าของข้อมูลจะพิจารณาจากลักษณะของรูปสัญญาณที่ปรากฏบนสายในแต่ละช่องของของเวลาหหรือเรียกว่าไทม์สล็อต (Time – Slot) โดยคาบเวลาต่ำสุดและสูงสุดของสถานะต่างๆ ในการสื่อสารข้อมูลในแต่ละไทม์สล็อต มีการกำหนดขอบเขตไว้อย่างชัดเจนการถ่ายทอดข้อมูลจะเกิดในแต่ไทม์สล็อตนั้น รูปแบบการถ่ายทอดข้อมูล จะเป็นแบบอะซิงโครนัสในระดับบิตไม่มีการกำหนดความยาวของข้อมูลระดับไบต์ ระบบการสื่อสารแบบนี้เหมาะที่จะใช้ในการสื่อสารข้อมูลระหว่างไอซีบนแผงวงจรเดียวกัน หรือสร้างเป็นโครงข่ายสื่อสารแบบทวิสต์แพร์ก็ได้

2.11.2 คุณสมบัติทางเทคนิคของระบบบัสหนึ่งสาย

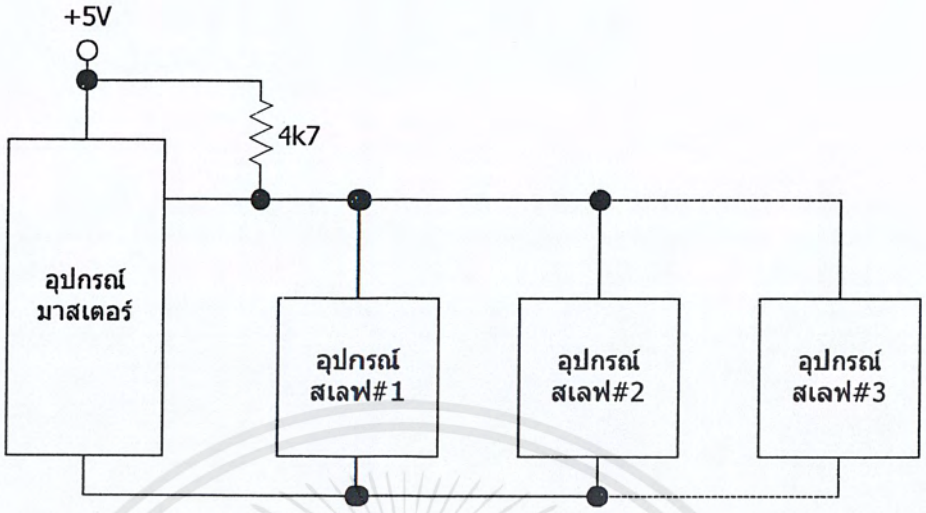
สาบสัญญาณแบบระบบบัสหนึ่งสายนี้จะเป็นสายสัญญาณแบบสองทิศทาง แต่ข้อมูลจะสามารถเดินทางได้ในทิศทางเดียวภายในช่วงเวลาหนึ่งๆนั่นคือ มีลักษณะคล้ายกับระบบสื่อสารข้อมูลแบบฮาล์ฟดูเพล็กซ์ (Half-Duplex) ตัวอย่างที่เห็นได้ชัดคือ การใช้งานวิทยุสื่อสารหรือวิทยุสมัครเล่น อุปกรณ์ระบบบัสต้องมีการระบุอย่างชัดเจนว่าตัวใดเป็นอุปกรณ์มาสเตอร์ ตัวใดเป็นอุปกรณ์ สเลฟ โดยส่วนใหญ่อุปกรณ์มาสเตอร์คือไมโครคอนโทรลเลอร์ ส่วนอุปกรณ์สเลฟได้แก่ ไอซีตรวจจับอุณหภูมิ,ไอซีหน่วยความจำแรม, เป็นต้น อุปกรณ์มาสเตอร์จะเป็นตัวจัดเตรียมความพร้อมของสายสัญญาณและควบคุมการถ่ายทอดข้อมูลบนสายสัญญาณนั้น ข้อมูลทั้งหมดไม่ว่าจะเป็นข้อมูลควบคุมหรือใช้งานจะถูกส่งลงบนสายสัญญาณที่มีอยู่เพียงเส้นเดียวนี้ทั้งหมด ในระหว่างการทำงานอุปกรณ์มาสเตอร์และสเลฟที่สามารถเป็นได้ทั้งตัวส่งและตัวรับ ขึ้นอยู่กับเงื่อนไขของการทำงานในขณะนั้น เช่น ถ้าหากมีการเขียนข้อมูลจากอุปกรณ์มาสเตอร์ไปยังอุปกรณ์สเลฟ ตัวส่งคืออุปกรณ์มาสเตอร์ ตัวรับคืออุปกรณ์สเลฟ ในทางตรงข้าม หากเป็นการอ่านจากข้อมูลสเลฟ ตัวส่งจะกลายเป็นอุปกรณ์สเลฟ และตัวรับคืออุปกรณ์มาสเตอร์ ในระบบบัส 1 สายต้องมีอุปกรณ์มาสเตอร์เพียงตัวเดียวเท่านั้น

2.11.3 คุณสมบัติไทม์สล๊อต

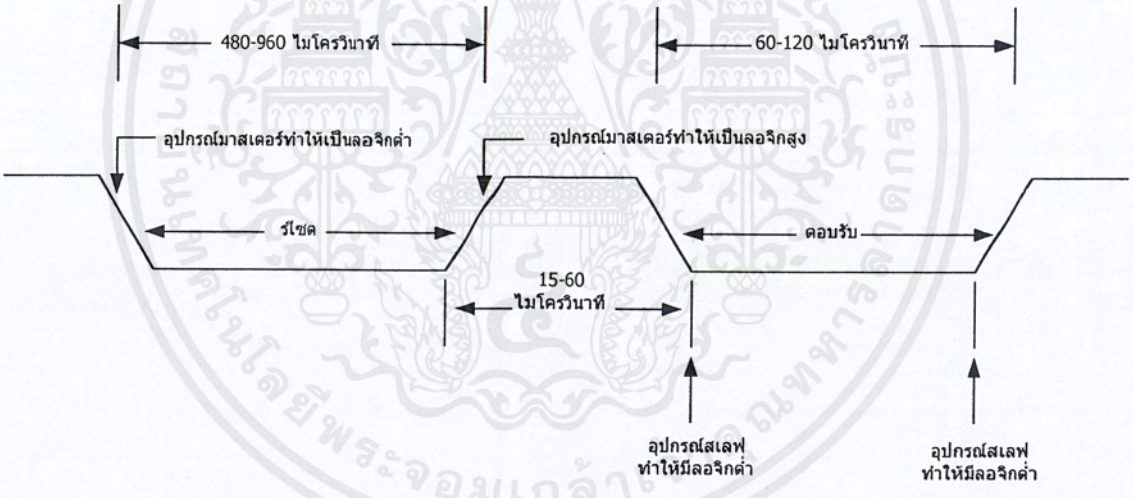
อุปกรณ์มาสเตอร์จะเป็นอุปกรณ์เพียงตัวเดียวบนระบบบัสหนึ่งสายนี้ที่สามารถทำการอินนิเชียลสายสัญญาณได้ โดยอุปกรณ์มาสเตอร์จะกำเนิดจุดเริ่มต้นของไทม์สล๊อตด้วยการทำให้สายสัญญาณเป็นลอจิกต่ำในช่วงเวลเหนึ่ง จากนั้นก็จะทำให้กลับมาเป็นลอจิกสูง ถ้าหากอุปกรณ์สเลฟต้องการส่งข้อมูลมายังอุปกรณ์มาสเตอร์ อุปกรณ์สเลฟจะเป็นตัวควบคุมสภาวะ ของสัญญาณ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต่อไป จนเสร็จสิ้นกระบวนการ แต่ถ้าหากอุปกรณ์มาสเตอร์ต้องการส่งข้อมูลก็สามารถดำเนินการต่อไปได้เลย



ภาพที่ 2.55 การเชื่อมต่อบนระบบบัสหนึ่งสาย



ภาพที่ 2.56 ไทม์สล๊อตการรีเซตและการตอบรับของอุปกรณ์บนระบบบัสหนึ่งสาย

ฟังก์ชั่นไทม์สล๊อตที่กำหนดโดยอุปกรณ์มาสเตอร์มีด้วยกัน 4 ฟังก์ชั่น คือ

- ไทม์สล๊อตของการรีเซต (Reset)
- การอ่านข้อมูล (Read Data)
- การเขียนข้อมูล “1” (Write One)
- การเขียนข้อมูล “0” (Write Zero)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไทม์สล็อตในการรีเซตใช้ในการเริ่มต้นติดต่อกับอุปกรณ์สเลฟในขณะทีไทม์สล็อตการอ่านจะสำหรับอ่านข้อมูลที่ส่งมาจากอุปกรณ์สเลฟ ส่วนไทม์สล็อตการเขียนข้อมูล “1” และ “0” ใช้สำหรับเขียนข้อมูลไปยังอุปกรณ์สเลฟผ่านสายสัญญาณของระบบ

ทางด้านอุปกรณ์สเลฟมีฟังก์ชั่นไทม์สล็อตอยู่ทั้งสิ้น 3 ฟังก์ชั่น คือ

- ไทม์สล็อตของการตอบสนอง (Presence)
- การเขียนข้อมูล “1” (Write One)
- การเขียนข้อมูล “0” (Write Zero)

ไทม์สล็อตของการตอบสนองใช้สำหรับตอบสนองการติดต่ออุปกรณ์มาสเตอร์ โดยการให้อุปกรณ์สเลฟตัวที่ถูกเลือกจากอุปกรณ์มาสเตอร์จะต้องส่งสัญญาณตอบสนองลงบนสายสัญญาณเพื่อแจ้งให้อุปกรณ์มาสเตอร์ทราบว่า และขณะนี้เองเราสามารถติดต่อกันได้แล้วส่วนไทม์สล็อตการเขียนข้อมูล “1” และ “0” ใช้สำหรับส่งข้อมูลไปยังอุปกรณ์มาสเตอร์ผ่านสายสัญญาณของระบบนี้ซึ่งจะสัมพันธ์กับไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์

การแยกแยะฟังก์ชั่นแต่ละไทม์สล็อตจะใช้ความยาวของคาบเวลาและลักษณะของรูปสัญญาณเป็นตัวกำหนด และทุกครั้งที่มีการเปลี่ยนแปลงฟังก์ชั่นต้องทำให้ฟังก์ชั่นอยู่ในสภาวะว่างเสมอ ซึ่งก็คือการทำให้สายสัญญาณเป็นลอจิกสูงอย่างน้อยเป็นเวลา 1 ไมโครวินาที

2.11.4ไทม์สล็อตการรีเซตและตอบสนอง

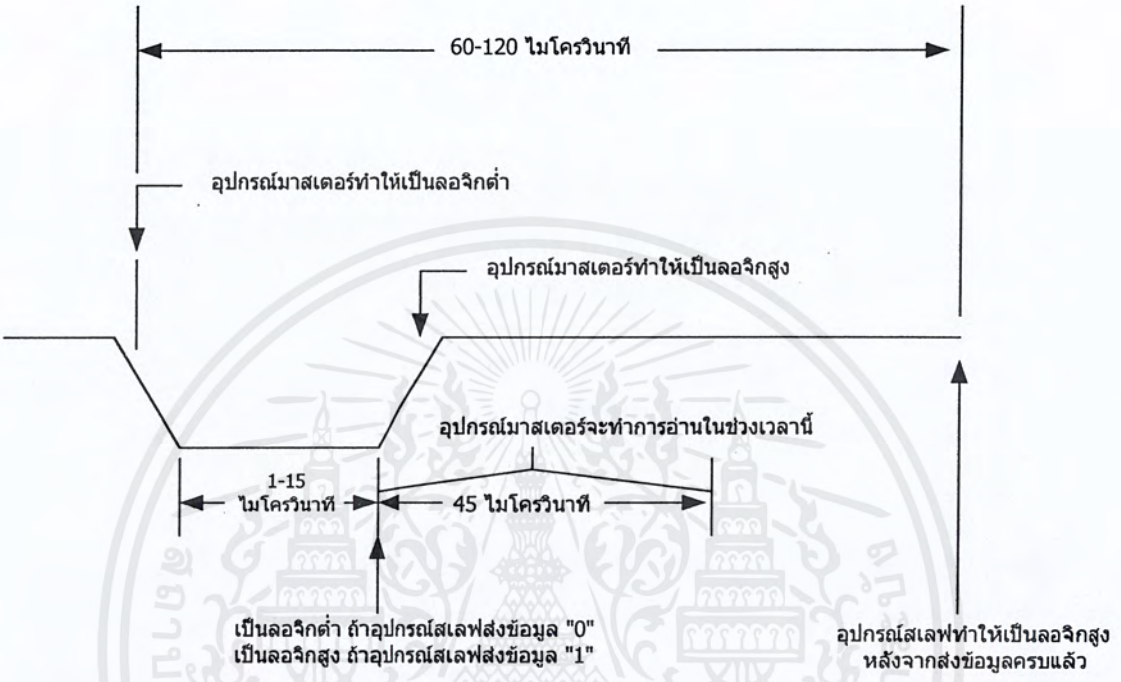
อุปกรณ์มาสเตอร์ทำให้เกิดการรีเซตบนสายสัญญาณเพื่อแจ้งแก่อุปกรณ์สเลฟ โดยการทำให้สายสัญญาณเป็นลอจิกต่ำต่อเนื่องอย่างน้อย 480 ไมโครวินาที และจะต้องทำให้สายสัญญาณกลับมาเป็นลอจิกสูงภายใน 480 ไมโครวินาทีหลังจากนั้น ถ้าหากอุปกรณ์สเลฟต่ออยู่บนอุปกรณ์สายสัญญาณจะต้องมีการตอบสนองสัญญาณรีเซตนั้นด้วยสัญญาณตอบสนอง (Present) โดยการทำให้สายสัญญาณเป็นลอจิกต่ำต่อเนื่องนานประมาณ 60-240 ไมโครวินาที หลังจากสัญญาณรีเซตปรากฏประมาณ15-60 ไมโครวินาทีในภาพที่ 2.56แสดงไทม์สล็อตของการรีเซตและการตอบสนอง

2.11.5ไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์และเขียนข้อมูลของอุปกรณ์สเลฟ

เมื่อต้องการอ่านข้อมูลจากอุปกรณ์สเลฟ อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 1-15 ไมโครวินาที จากนั้นต้องทำให้สถานะของสายกลับมาเป็นลอจิกสูง อุปกรณ์สเลฟจะส่งข้อมูลมาให้อุปรณ์มาสเตอร์โดยถ้าให้ข้อมูลเป็น “0” อุปกรณ์สเลฟจะทำให้สายสัญญาณเป็นลอจิกต่ำนานประมาณ 45 ไมโครวินาที แล้วทำให้สายสัญญาณกลับมาสู่สภาวะลอจิกสูงอีกครั้ง แต่ถ้าเป็นข้อมูล “1” อุปกรณ์สเลฟจะทำให้สายสัญญาณเป็นลอจิกสูงต่อเนื่องกันไปอีก 45 ไมโคร วินาที รวมเวลาทั้งหมดในไทม์สล็อตนี้ประมาณ 60-120 ไมโครวินาที นั่นคือในไทม์สล็อตจะต้องใช้เวลาไม่เกิน 120 ไมโครวินาที ในขณะที่อุปกรณ์มาสเตอร์จะใช้เวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการอ่านข้อมูลอยู่ระหว่าง 15 และ 60 ไมโครวินาทีหลังจากเริ่มต้นไทม์สล็อตนี้ ในภาพที่ 2.57 แสดงรูปสัญญาณของไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์ซึ่งจะมีลักษณะเหมือนกับการเขียนข้อมูลของอุปกรณ์สเลฟและไทม์สล็อตทั้งสองจะเกิดขึ้นในช่วงเวลาเดียวกันกล่าวคือ เมื่ออุปกรณ์มาสเตอร์อ่าน อุปกรณ์ สเลฟก็ต้องทำการเขียน

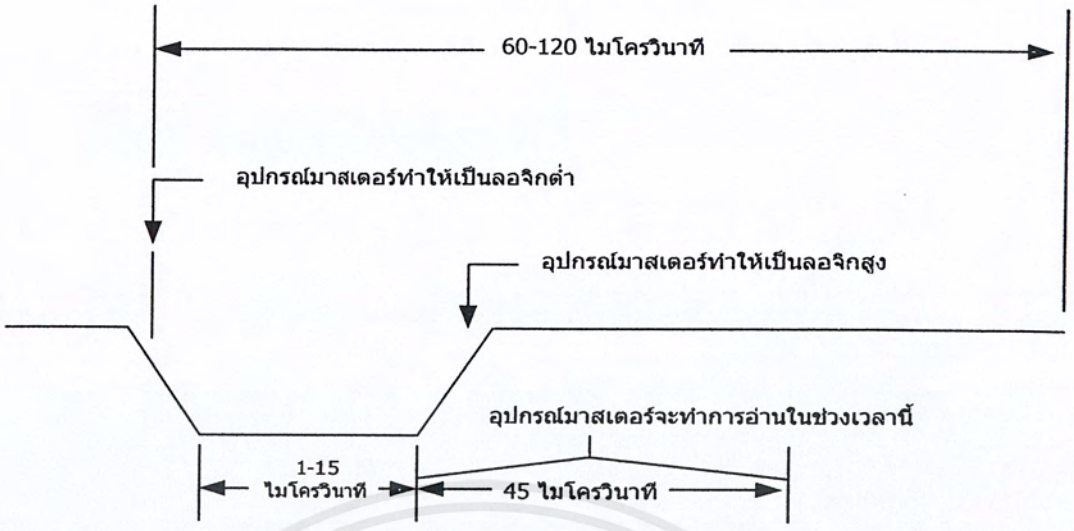


ภาพที่ 2.57 ไทม์สล็อตการอ่านข้อมูลของอุปกรณ์มาสเตอร์ซึ่งตรงกับไทม์สล็อตการเขียนข้อมูลของอุปกรณ์สเลฟ

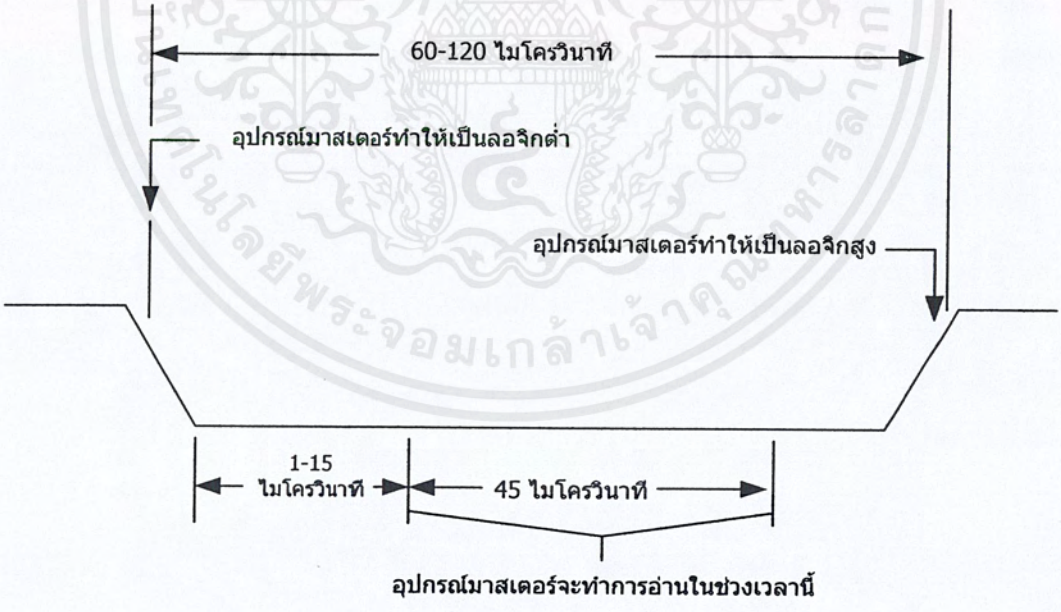
2.11.6 ไทม์สล็อตการเขียนข้อมูลของอุปกรณ์มาสเตอร์

เมื่ออุปกรณ์มาสเตอร์ต้องการเขียนข้อมูล อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำประมาณ 1-15 ไมโครวินาที จากนั้นต้องทำให้สถานะของสายกลับมาเป็นลอจิกสูง แล้วดำเนินการเขียนข้อมูลได้ในทันทีถ้าข้อมูลที่ต้องการเขียนเป็น “0” อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกต่ำ นานประมาณ 45 ไมโครวินาทีแล้วทำให้สายสัญญาณกลับมาสู่สภาวะลอจิกสูงอีกครั้ง แต่ถ้าต้องการเขียนข้อมูล “1” อุปกรณ์มาสเตอร์จะทำให้สายสัญญาณเป็นลอจิกสูงต่อเนื่องไปอีก 45 ไมโครวินาที รวมเวลาทั้งหมดในไทม์สล็อตนี้ประมาณ 60-120 ไมโครวินาที ในภาพที่ 2.58 แสดงรูปสัญญาณของไทม์สล็อตการเขียนข้อมูลของอุปกรณ์มาสเตอร์ซึ่งก็จะมีลักษณะเหมือนกับการอ่านข้อมูลของอุปกรณ์สเลฟ และไทม์สล็อตทั้งสองจะเกิดขึ้นในช่วงเวลาเดียวกัน กล่าวคืออุปกรณ์มาสเตอร์เขียน อุปกรณ์สเลฟก็ต้องทำการอ่านข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 2.58 ไทม์สล็อตการเขียนข้อมูล "1" ของอุปกรณ์มาสเตอร์ ซึ่งตรงกับไทม์สล็อตการอ่านข้อมูลของอุปกรณสเลฟ



ภาพที่ 2.59 ไทม์สล็อตการเขียนข้อมูล "0" ของอุปกรณ์มาสเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.11.7 รูปแบบของการสื่อสารข้อมูลแบบหนึ่งสาย

ในการติดต่อสื่อสารข้อมูลในระบบบัส 1 สายอุปกรณ์มาสเตอร์จะสามารถติดต่อกับส่วนที่เป็นอุปกรณ์สเลฟได้ครั้งละ 1 ตัวเท่านั้น ดังนั้นอุปกรณ์สเลฟแต่ละตัวต้องมีข้อมูลกำหนดแอดเดรสเฉพาะตัวโดยจะเก็บไว้ในหน่วยความจำรอมของอุปกรณ์สเลฟนั้นๆ โดยปกติอุปกรณ์สเลสบัส1สายนี้จะมีหน่วยความจำขนาด 64 บิตหรือ 8 ไบต์ สำหรับเก็บข้อมูลต่างๆที่สำคัญแต่ละตัว ซึ่งประกอบด้วย

1. รหัสของตระกูล จำนวน 8 บิต
2. เลขหมายประจำตัว (Serial Number) จำนวน 8 บิต
3. รหัสตรวจสอบความผิดพลาด (CRC:Cylical Reduncy Check) จำนวน 8 บิต

ผู้ใช้งานสามารถอ่านข้อมูลประจำตัวของอุปกรณ์สเลฟนั้นได้ด้วยการใช้คำสั่งอ่านหน่วยความจำรอม (Read ROM) ในกรณีที่บนสายสัญญาณมีอุปกรณ์สเลฟเพียงตัวเดียวไม่จำเป็นต้องอ้างแอดเดรสในการติดต่อ

รูปแบบการติดต่อบนระบบบัส 1 สายจะเริ่มต้นเมื่ออุปกรณ์มาสเตอร์ทำการรีเซตและกำหนดแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ถ้าหากมีอุปกรณ์สเลฟเพียงตัวเดียวสามารถข้ามขั้นตอนการติดต่อหน่วยความจำรอมในอุปกรณ์สเลฟได้ จะเรียกวิธีการดังกล่าว การไม่ติดต่อหน่วยความจำรอม หรือ สคริปรอม (Skip ROM) จากนั้นรอการตอบรับจากอุปกรณ์สเลฟ เมื่อการตอบรับสมบูรณ์ก็สามารถเริ่มขั้นตอนการอ่านหรือเขียนข้อมูลได้ต่อไป

2.11.8 ไอซีตรวจจับอุณหภูมิ DS1820

เป็นไอซีตรวจจับอุณหภูมิที่ใช้การติดต่อแบบระบบบัส 1สาย มีขาต่อใช้งานเพียง 3 ขา คือ DQ ซึ่งเป็นขาเชื่อมต่อกับระบบบัส,ขาต่อไฟเลี้ยงภายนอก และขากราวด์ ซึ่งมีโครงสร้างภายในดังภาพที่ 2.60

หัวใจสำคัญของ DS1820 อยู่ที่ตัวตรวจจับอุณหภูมิและหน่วยความจำความเร็วสูงเรียกว่า (Scratchpad) ซึ่งมีขนาด 9 ไบต์ มีการจัดสรรหน่วยความจำส่วนนี้แสดงในภาพที่ 2.61 โดยเมื่อวัดอุณหภูมิที่ได้ก็จะนำค่าที่วัดได้นี้มาเก็บไว้ในสแครตแพดที่ไบต์ 0 และไบต์ 1 ทั้งนี้เนื่องจากไอซี DS1820 สามารถให้ข้อมูลละเอียดถึง 16 บิต เมื่อนำมาแปลงข้อมูลเป็นเลขฐานสิบจึงสามารถแสดงความละเอียดของอุณหภูมิได้ถึง 0.5 องศาเซลเซียส และ 0.9 องศากาเรการ์อน โดยมีย่านวัดอุณหภูมิ –55 ถึง –125 องศาเซลเซียส หรือ –67 ถึง +257 องศาฟาเรนไฮต์ โดยค่าขององศาฟาเรนต์ไฮต์จะต้องใช้การแปลงหน่วยเข้ามาช่วยใช้เวลาในการแปลงค่าอุณหภูมิเป็นข้อมูลดิจิตอลประมาณ 200 มิลลิวินาที สามารถกำหนดขอบเขตของอุณหภูมิที่ทำการวัดได้ และให้แจ้งเตือนอุณหภูมิสูงขึ้นหรือต่ำลงถึงค่าที่กำหนด โดยอุณหภมิที่กำหนดนี้จะเก็บไว้ในสแครตซ์แพดในไบต์ที่ 2 และ3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

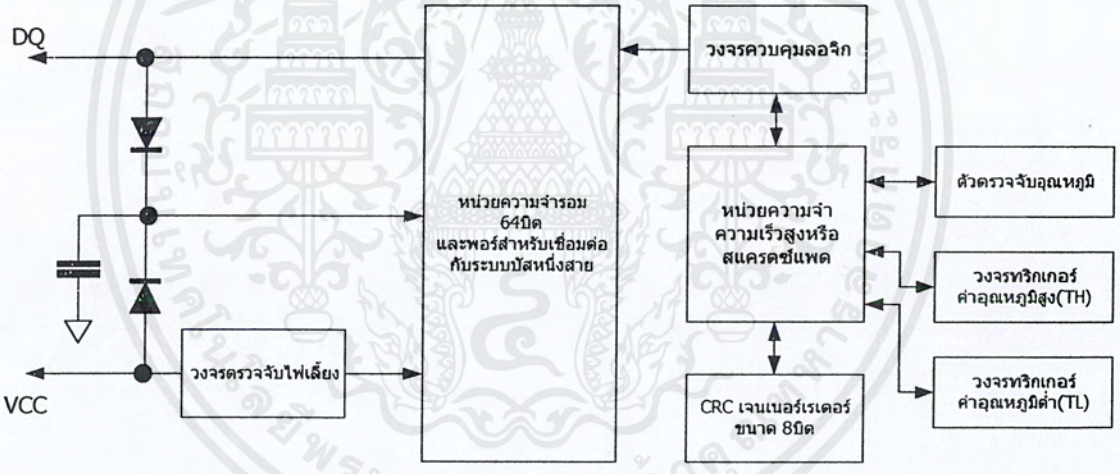
2.11.9 คำสั่งเพื่อควบคุมการทำงานของ DS1820

ในการติดต่อกับไอซี DS1820 จะต้องมีคำสั่งที่ต้องส่งให้แก่ DS1820 เพื่อกำหนดรูปแบบการทำงาน คำสั่งที่ใช้มากที่สุดมีด้วยกัน 3 คำสั่งดังนี้

1. คำสั่งไม่ติดต่อกับหน่วยความจำรอม หรือสคริปรอม (Scrip ROM) เนื่องจากในการใช้งาน DS1820 โดยปกติแล้วจะมี DS1820 อยู่บนสายเพียงตัวเดียว จึงไม่จำเป็นต้องใช้ข้อมูลกำหนดแอดเดรส ดังนั้นจึงไม่ต้องติดต่อกับหน่วยความจำรอมเพื่ออ่านข้อมูล ข้อมูลของคำสั่งสคริปรอมที่ต้องส่งให้ DS1820 คือ 0CCH

2. คำสั่งแปลงอุณหภูมิ (Convert T) มีค่าเท่ากับ 44H เมื่อส่งคำสั่งนี้ให้ DS1820 จะต้องทำการวนลูปรออย่างน้อง 200 มิลลิวินาที เพื่อให้ DS1820 ได้ใช้เวลานี้ในการแปลงค่าอุณหภูมิเป็นข้อมูลดิจิตอลมาเก็บไว้ในสแครตซ์แพด

3. คำสั่งอ่านข้อมูลจากสแครตซ์แพด (Read Scratchpad) มีค่าเท่ากับ 0BEH เมื่อส่งคำสั่งนี้ DS1820 จะทยอยส่งค่าอุณหภูมิออกมาทั้งหมด 9 ไบต์



ภาพที่ 2.60 โครงสร้างการทำงานภายในของไอซีตรวจจับอุณหภูมิ DS1820

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

	ไบต์
ข้อมูลอุณหภูมิไบต์ต่ำ (TL)	0
ข้อมูลอุณหภูมิไบต์สูง (TH)	1
ข้อมูลอุณเหภูมิค่าสูง	2
ข้อมูลอุณหภูมิค่าต่ำ	3
สำรองไว้	4
สำรองไว้	5
รีจีสเตอร์เก็บค่าการนับ	6
รีจีสเตอร์เก็บค่าการนับต่อ ^{0} C	7
CRC	8

ภาพที่ 2.61 การจัดสรรพื้นที่ของสแครตซ์แพด

2.12 ความรู้เบื้องต้นของระบบบัส I^2C

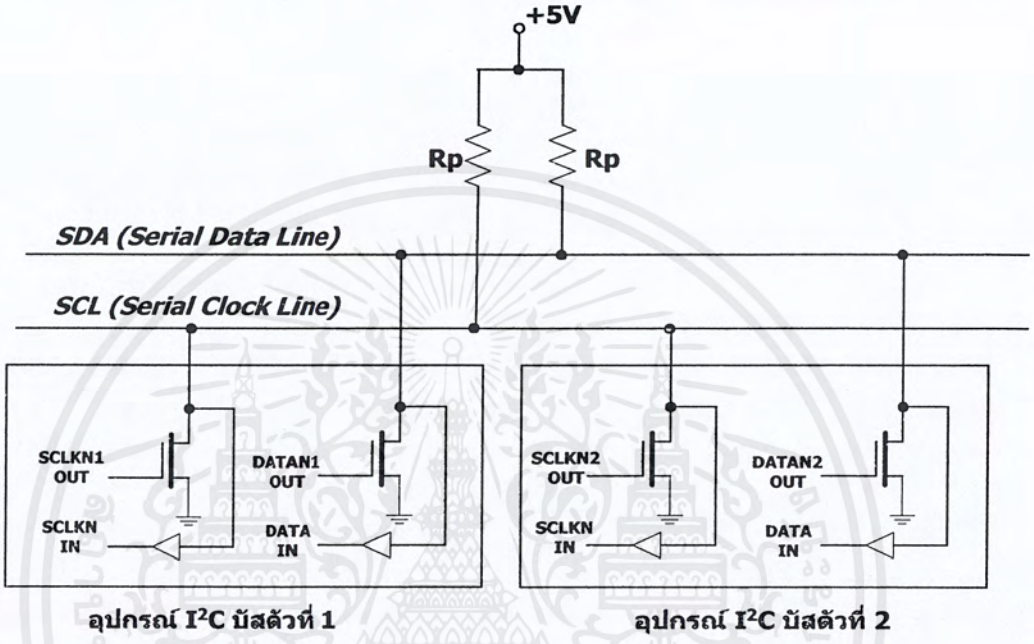
I^2C ย่อมาจาก Inter – IC Comunication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยระบบบัส I^2C ได้รับการพัฒนาขึ้นโดยฟิลิปส์(Philips) ด้วยจุดมุ่งหมายหลัก คือต้องการให้ไอซีหรือโมดูลสามารถติดต่อสั่งงานและควบคุมภายใต้สายสัญญาณเพียงสองเส้นคือ สายข้อมูลอีกเส้นหนึ่งคือสายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงานการต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานกัน ส่วนการกำหนดแอดเดรสสำหรับติดต่ออุปกรณ์แต่ละตัวจะใช้กำหนดสภาวะลอจิกที่ขาแอดเดรส

สายข้อมูลบนบัส I^2C มีชื่อเรียกอย่างเป็นทางการว่า สายข้อมูลอนุกรม หรือ SDA (Serial Data Line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรม หรือ SCL (Serial Clock Line)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.12.1 คุณสมบัติทั่วไปของบัส I^2C

สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (Bi-Direction Line) ต้องมีตัวต้านทานพูลอัปกับแรงดัน +5V ไว้ตลอดเวลา เพื่อให้สายสัญญาณมีสถานะลอจิกสูงในขณะที่ไม่มีการต่อใช้งาน ทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสองวงจรเอาท์พุทของอุปกรณ์ที่ต่ออยู่บนบัส I^2C ต้องมีลักษณะ OPEN-DRAIN หรือ OPEN COLLECTOR



ภาพที่ 2.62 แสดงโครงสร้างของวงจรที่ต่อเข้ากับบัส I^2C

อัตราการถ่ายเทข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาทีโดยจะอยู่ในโหมดปกติ (Standard Mode) และสูงถึง 400 กิโลบิตติวินาทีในโหมดความเร็วสูง (Fast Mode) อุปกรณ์ที่ต่อร่วมอยู่บนบัส I^2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL 1 ไม่เกิน 400 pF การเข้าถึงอุปกรณ์บนบัส I^2C ใช้ข้อมูลการเข้าถึง 2 ค่า คือ 7 บิต (7-Bit Addressing) และ 10 บิต (10 Bit Addressing)

2.12.2 หลักการของบัส

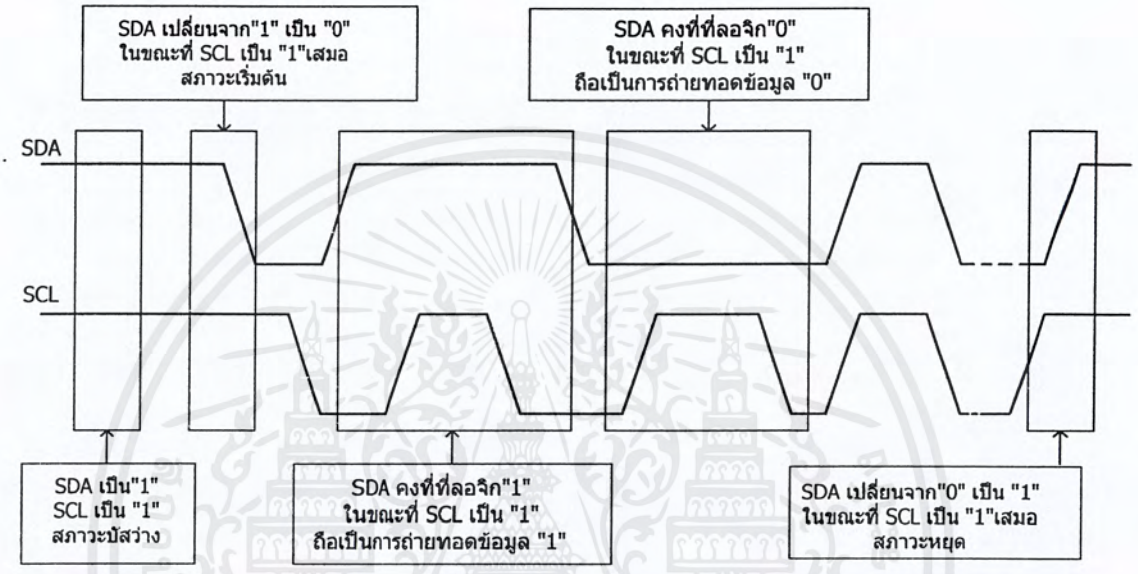
บัสประกอบด้วยสายสัญญาณ 2 เส้น คือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัส I^2C สามารถมีได้มากมาย ดังนั้นจึงต้องมีการติดต่อบนบัสหรือ เรียกว่า Protocal เพื่อให้ผู้ใช้งานทราบว่าขณะนี้อุปกรณ์ใดติดต่อกันอยู่ อุปกรณ์ใดเป็นตัวส่งหรือตัวรับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.12.2.1 ข้อกำหนดสำคัญของการติดต่อบนบัส คือ

1. การถ่ายทอดข้อมูลที่จะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

2.ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูงสายข้อมูลต้องรักษาข้อมูลไว้อย่าให้เกิดการเปลี่ยนแปลงเด็ดขาด มิฉะนั้นสัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน



ภาพที่ 2.63 ไดอะแกรมเวลาแสดงสถานะต่างๆ ในบัส I^2C

2.12.2.2 สภาวะที่เกิดขึ้นบนบัส

สภาวะที่เกิดขึ้นบนบัส มีด้วยกัน 5 สภาวะ ด้วยกัน คือ

1. บัสว่าง (Bus not busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

2. เริ่มต้นการถ่ายทอดข้อมูล (Start data tranfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่าสภาวะเริ่มต้น

3.หยุดการถ่ายทอดข้อมูล (Stop data tranfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้น นี้ว่า สภาวะหยุด

4.ข้อมูลดำรงอยู่บนบัส (Data valid) สภาวะนี้เกิดขึ้นหลังจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่า เป็น “0” หรือ “1” ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตาม ที่ต้องการให้มีการถ่ายเท

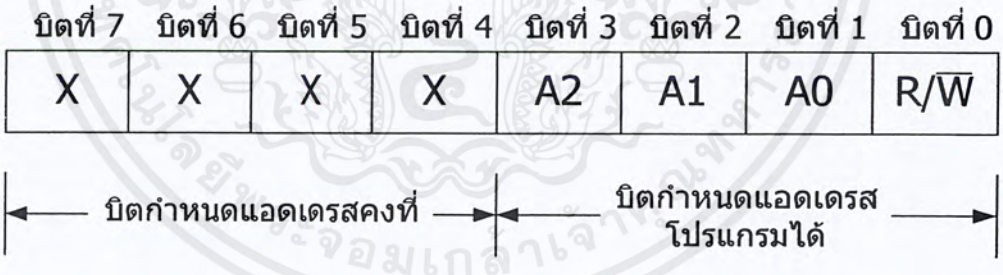
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลอย่างสมบูรณ์ สถานะที่ SDA ต้องคงที่ ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูงหาก เกิดการเปลี่ยนแปลงสถานะลอจิกที่สาย SCL มีลอจิกสูงอยู่นั้นอุปกรณ์มาสเตอร์ที่ทำการควบคุม การถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุด หรือ สภาวะเริ่มต้นก็ได้ ทำให้ข้อมูล การถ่ายทอดนั้นเกิดความผิดพลาดขึ้น

5.รับรู้ข้อมูล (Acknowledge) เกิดขึ้นหลังจากถ่ายทอดข้อมูล จากตัวส่งมายังตัวรับ เกิดขึ้น อย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิต เรียกว่า บิตรับรู้ (Acknowledge bit) มีสภาวะ เป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณ รับรู้พิเศษ ซึ่งสัมพันธ์กับสัญญาณนาฬิกา อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้น ก็จะกำเนิดบิตรับรู้ที่มีสถานะลอจิกต่ำ เพื่อตอบสนองให้ทราบว่ารับรู้ข้อมูลในแต่ละไบต์เรียบร้อย

2.12.3 การทำงานบนบัส I^2C

ก่อนที่จะเริ่มการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่างๆที่ต่ออยู่บนบัส ต้องมีการอ้างอิงถึง อุปกรณ์เสียก่อน โดยการอ้างถึงอุปกรณ์บนบัส I^2C นั้นจะมีการอ้างถึงแบบ 7 บิตหรือ 10 บิต ในกรณีที่มีอุปกรณ์ที่ต่ออยู่บนบัสไม่มาก ใช้การอ้างถึงแบบ 7 บิตก็พอ แต่ถ้ามีอุปกรณ์ต่ออยู่บนบัส มากกว่า 127 แอดเดรส จำเป็นต้องอ้างถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อย แล้วก็จะเริ่มถ่ายโอนข้อมูลกันต่อไปดังนั้นหัวใจสำคัญในอันดับแรกของการทำงานบนบัส I^2C คือ การอ้างถึงอุปกรณ์แต่ละตัว



ภาพที่ 2.64 รูปแบบของข้อมูลกำหนดแอดเดรสที่ใช้ในการอ้างถึงแบบ 7 บิต

2.12.4 การอ้างถึงแบบ 7 บิต (7-bit addressing)

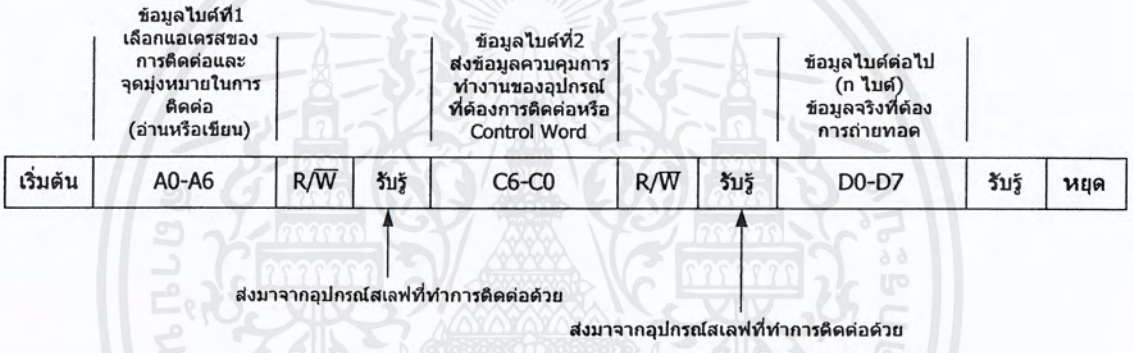
ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้น คือข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ ติดต่อโดยมีรูปแบบแสดงในภาพที่ 2.65 ใน 7 บิตบนรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรส ของอุปกรณ์สเลฟ ที่ต้องการติดต่อโดยแบ่งเป็นบิตกำหนดแอดเดรสคงที่ (Fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลง แก้ไข ได้ถัดมาอีก 3 บิตเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (Programable address bit) โดยผู้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I^2C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือการเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้นๆ หากบิต LSB เป็น "0" หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์ตัวนั้น ถ้าเป็น "1" จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ

ข้อมูลไบต์ต่อมาคือ ข้อมูลควบคุม (Control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลที่แตกต่างกันไป ยกตัวอย่าง ไอซีขยายพอร์ตทมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุต บิตใดเป็นบิตเอาท์พุต ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลไบต์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง (data) หลังจากได้มีการถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาทุกครั้งด้วย

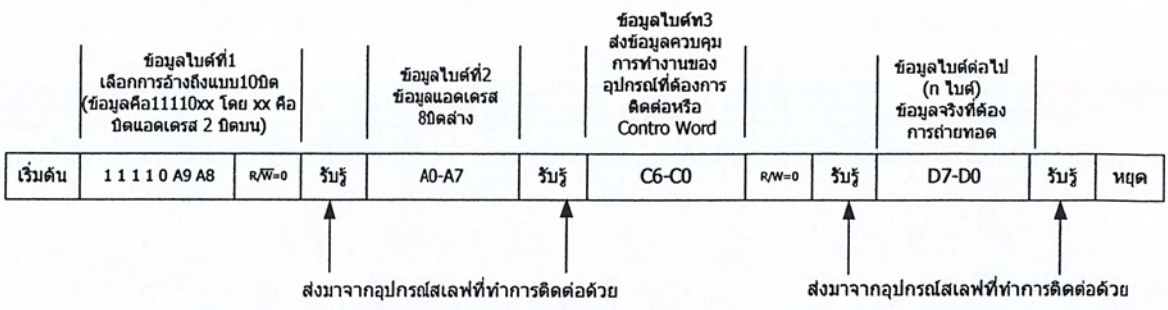


ภาพที่ 2.65 รูปแบบของข้อมูลอนุกรมมี่ใช้ติดต่อกับอุปกรณ์บัส I^2C เมื่อมีการอ้างถึง 7 บิต

2.12.5 การอ้างถึง แบบ 10 บิต

จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิตแอดเดรสของอุปกรณ์สเลฟตัวที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วยข้อมูลไบต์ต่อมาจึงเป็นข้อมูลควบคุมข้อมูลหลังจากนั้นจึงเป็นข้อมูลจริงที่ใช้ในการติดต่อเช่นเดียวกันกับการอ้างแบบ 7 บิต หลังการถ่ายทอดข้อมูลครบทุกไบต์ต้องมีสภาวะการ รับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

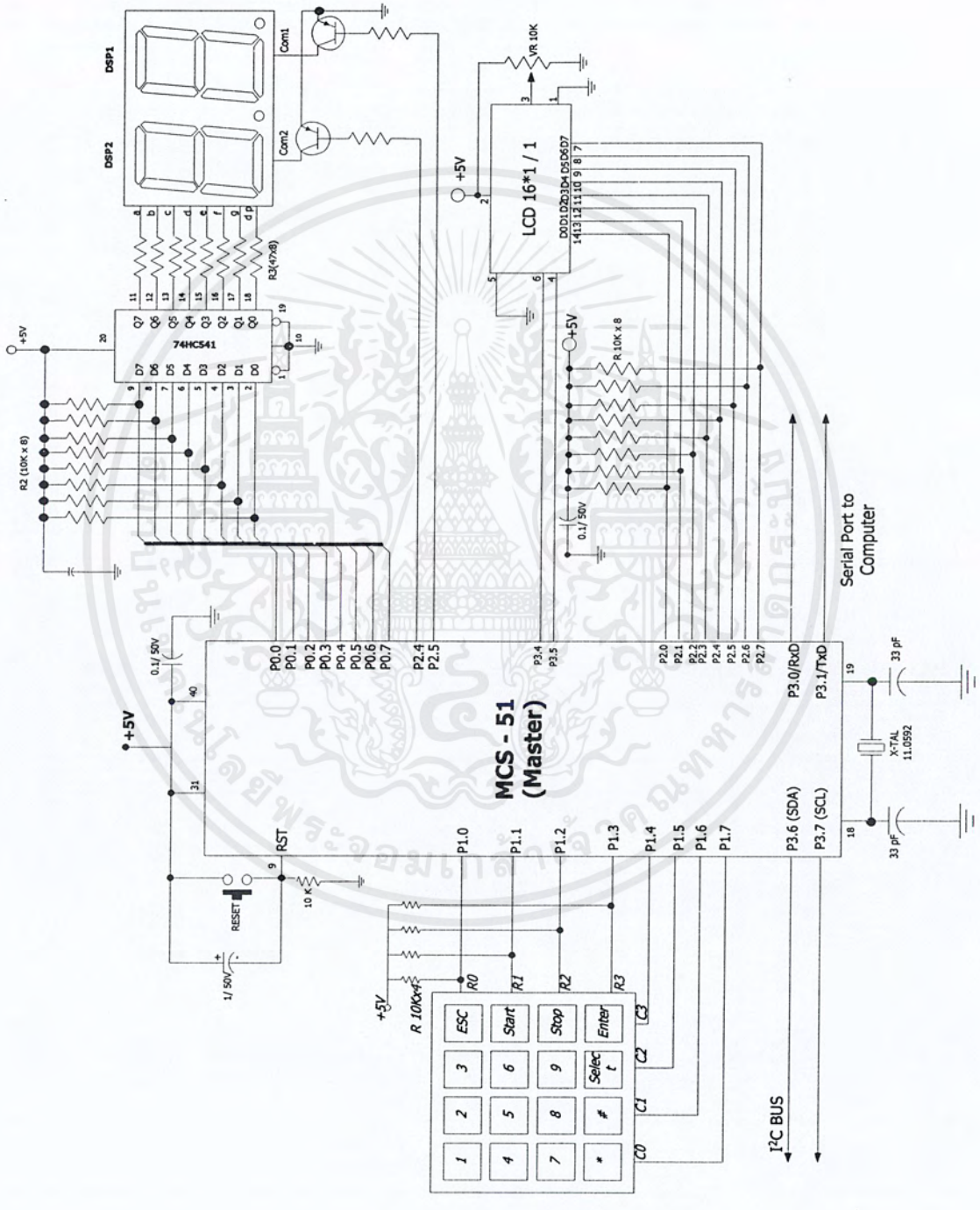


ภาพที่ 2.66 รูปแบบของข้อมูลอนุกรมมี่ใช้ติดต่อกับอุปกรณ์บัส I^2C เมื่อมีการอ้างถึง 10 บิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
การออกแบบวงจรและโปรแกรม

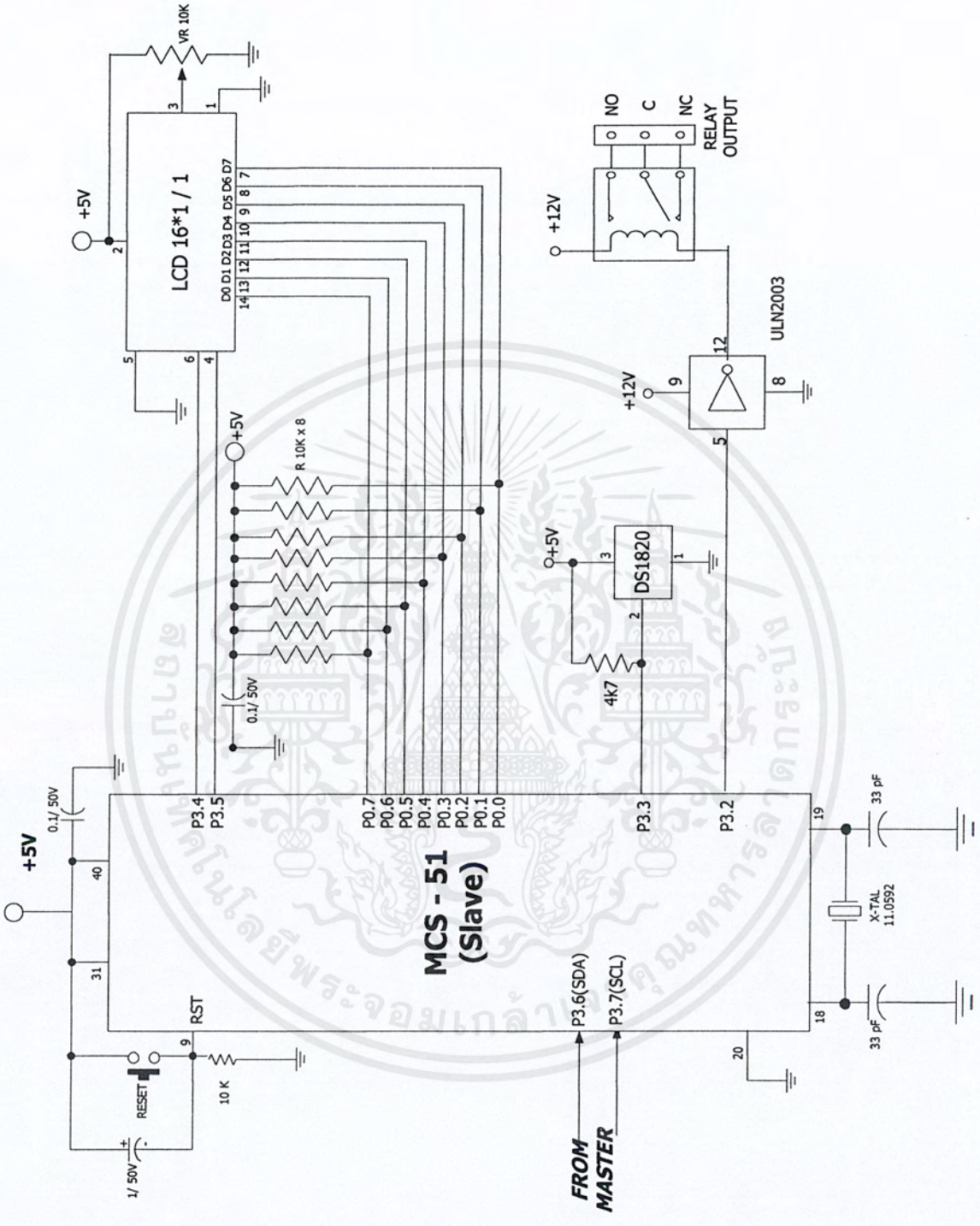
3.1 วงจรของไมโครคอนโทรลเลอร์ตัวมาสเตอร์ที่ต่อร่วมกับโมดูลต่างๆ



ภาพที่ 3.1 แสดงวงจรของไมโครคอนโทรลเลอร์ตัวมาสเตอร์ที่ต่อร่วมกับโมดูลต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

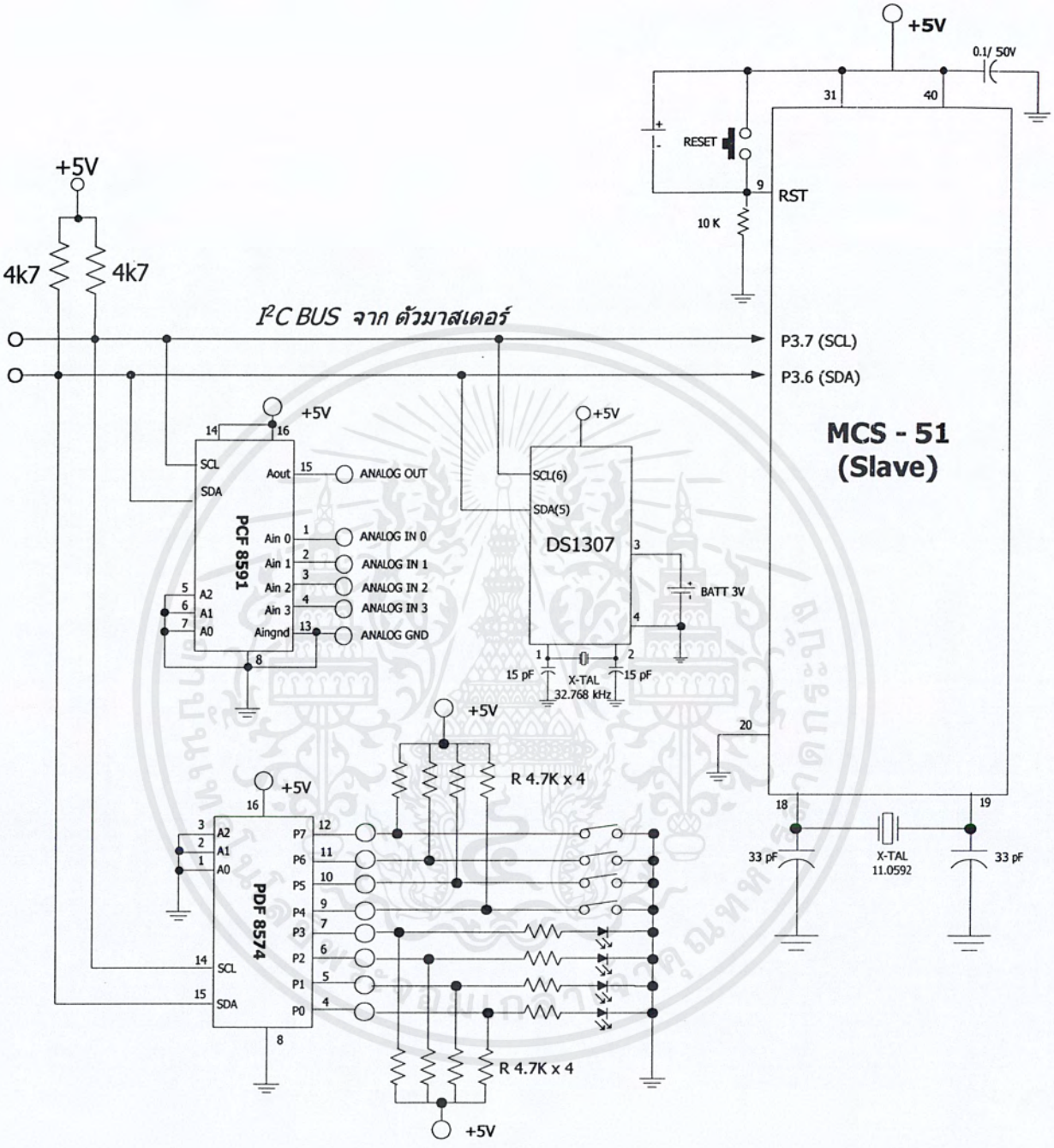
3.2 วงจรของไมโครคอนโทรลเลอร์ตัวสเลฟที่ต่อร่วมกับโมดูลต่างๆ



ภาพที่ 3.2 แสดงวงจรของไมโครคอนโทรลเลอร์ตัวสเลฟที่ต่อร่วมกับโมดูลต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

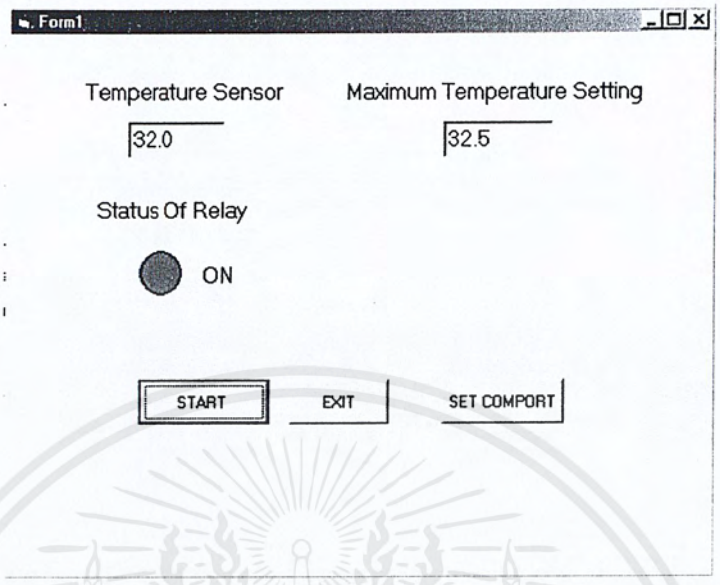
3.3 วงจรของตัวสเลฟที่ต่อร่วมกันบนบัส I^2C



ภาพที่ 3.3 แสดงวงจรของตัวสเลฟที่ต่อร่วมกันบนบัส I^2C

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4 ส่วนแสดงผลของตัวตรวจวัดอุณหภูมิโดยผ่านทาง Serial Port



ภาพที่ 3.4 แสดงหน้าต่างของการแสดงผลอุณหภูมิที่รับมาจากไมโครคอนโทรลเลอร์

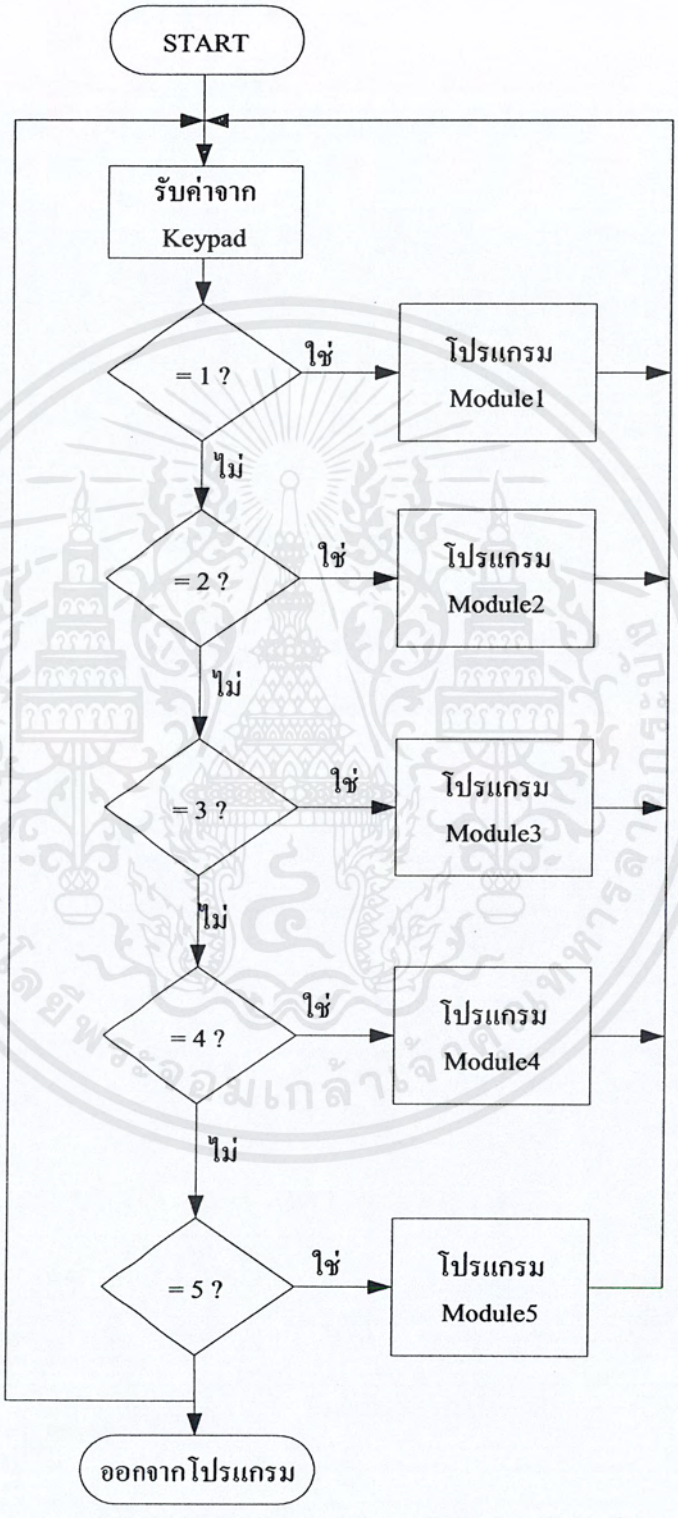
จากภาพที่ 3.4 แสดงหน้าต่างของการแสดงผลอุณหภูมิที่รับมาจากไมโครคอนโทรลเลอร์โดยผ่านทาง Serial Port ซึ่งจะเป็นการรับค่าอุณหภูมิที่ได้มาจาก DS1820 ซึ่งเป็นไอซีตรวจวัดอุณหภูมิโดยนำค่าอุณหภูมิที่ได้นั้นมาแสดงที่จอ LCD ของตัวสเลฟและนำมาแสดงที่คอมพิวเตอร์โดยผ่านทางพอร์ตอนุกรม

ซึ่งการทำงานของส่วนแสดงผลคือ จะต้องป้อนค่าอุณหภูมิสูงสุดที่ต้องการให้ Relay ตัดหรือหยุดจ่ายแรงดันไฟฟ้าให้แก่ตัวให้ความร้อนชนิดใดๆ (ซึ่งแล้วแต่จะนำมาใช้งาน)ในที่นี้ขอยกตัวอย่างเป็นการให้ความร้อนในตู้ฟักไข่ โดยใช้หลอดไฟ ซึ่งหากในตู้ฟักไข่นั้นมีอุณหภูมิเกินกว่าที่เรากำหนดโปรแกรมของตัวแสดงผลนี้ก็จะทำการสั่งให้ RelayOff ทันที

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5 การออกแบบโปรแกรมโดยใช้โฟลวชาร์ต

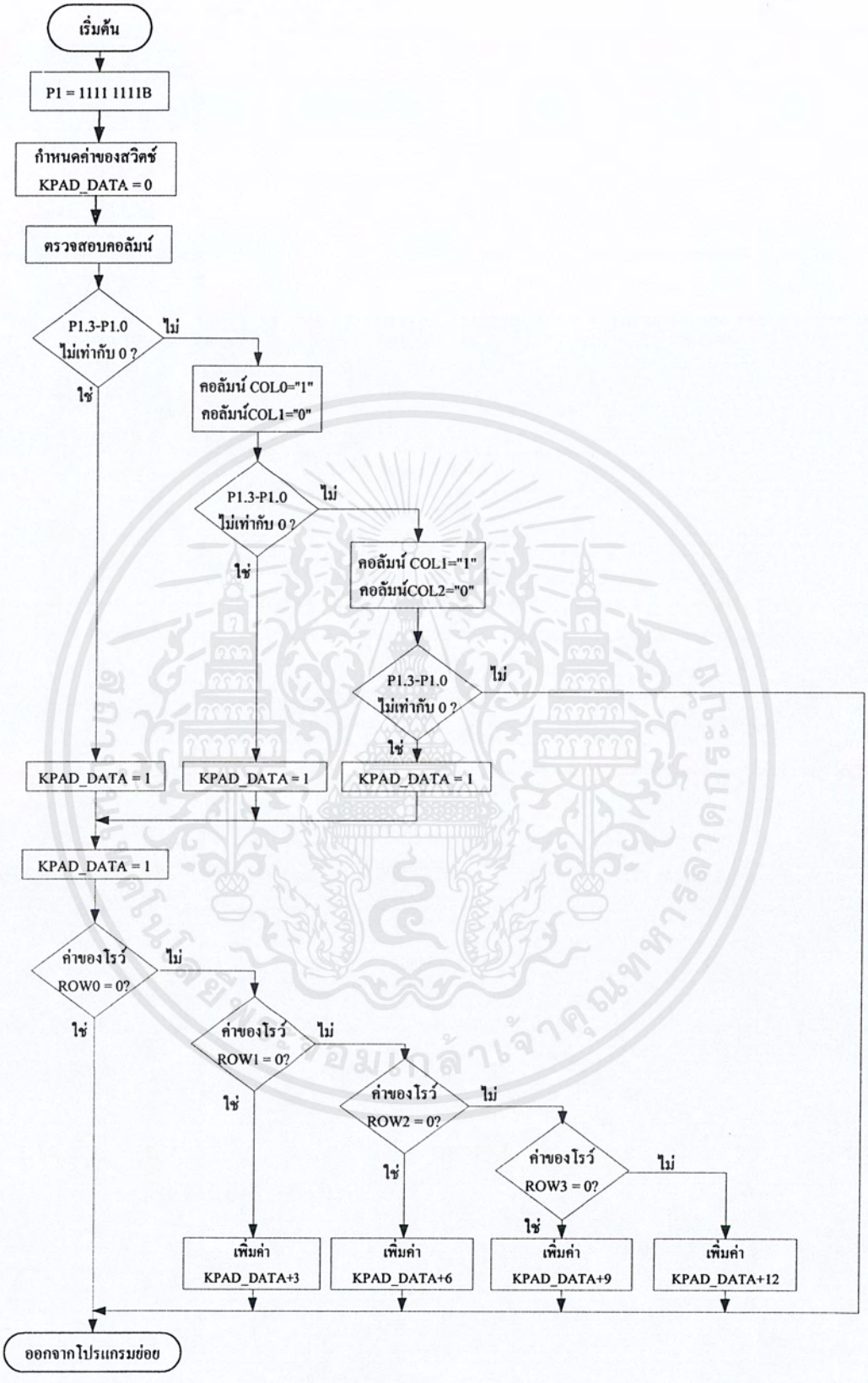
3.5.1 โฟลวชาร์ตโปรแกรมเมน



ภาพที่ 3.5 โฟลวชาร์ตโปรแกรมเมน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

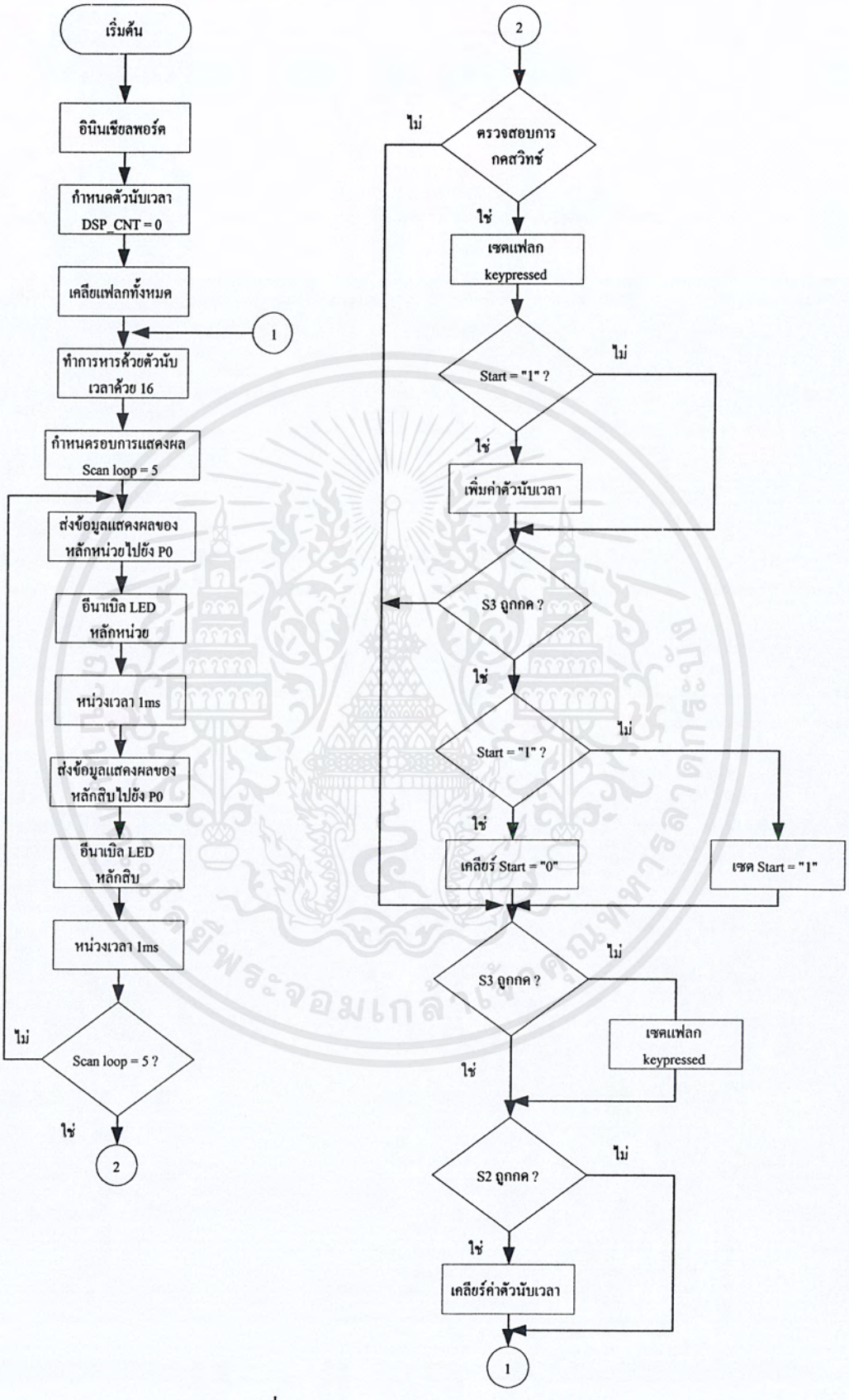
3.5.2 โฟลวชาร์ตโมดูล Keypad



ภาพที่ 3.6 โฟลวชาร์ตโมดูล Keypad

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

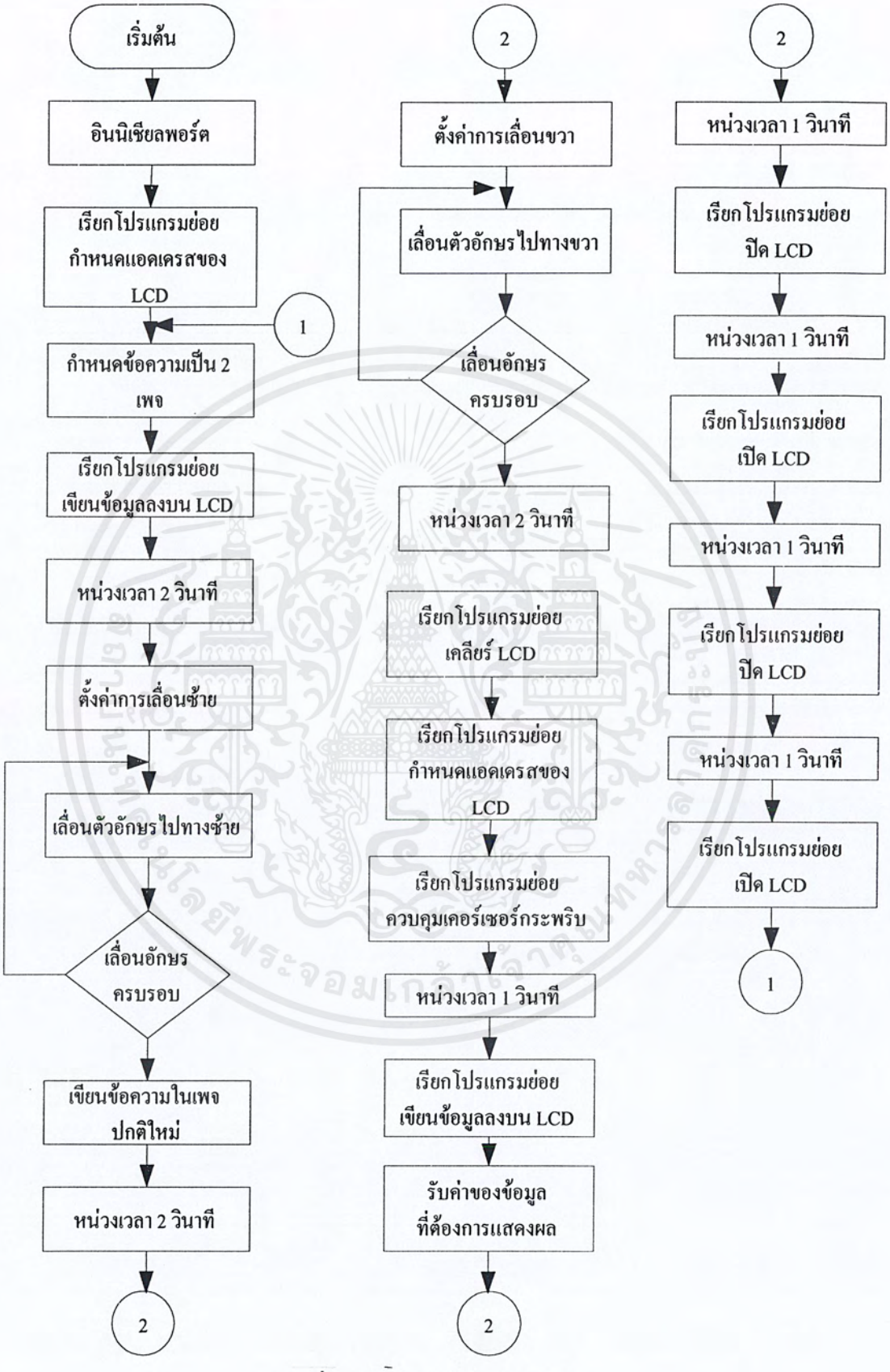
3.5.3 โฟลวชาร์ตโมดูล 7-Segment



ภาพที่ 3.7 โฟลวชาร์ตโมดูล 7- Segment

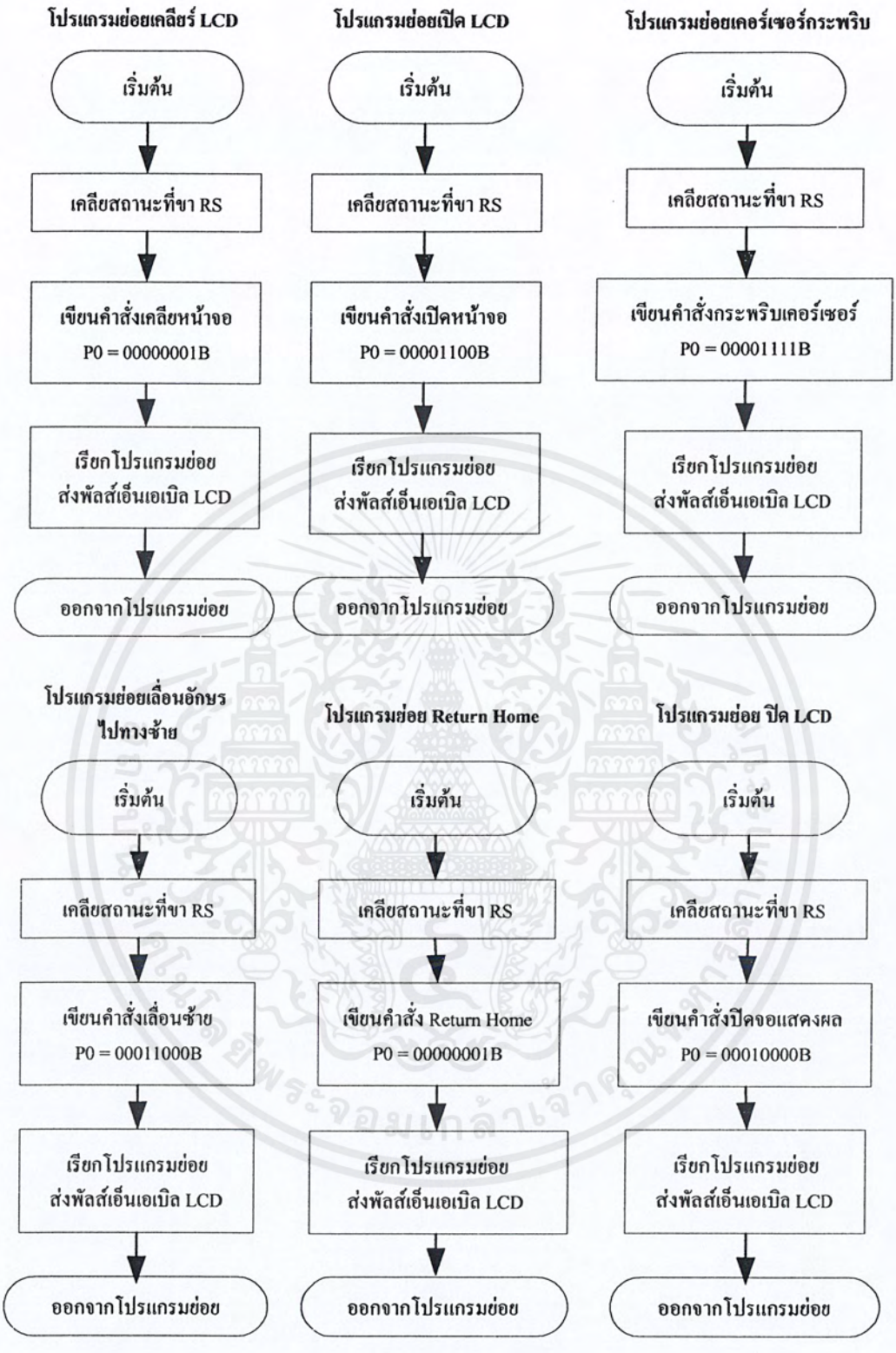
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5.4 โฟลวชาร์ตโมดูล LCD



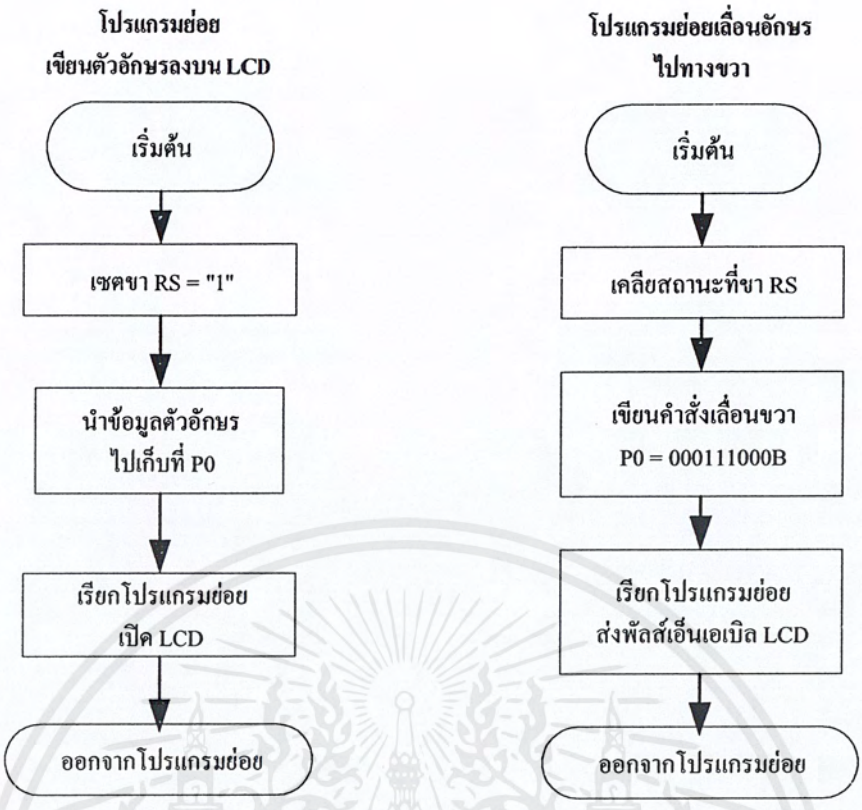
ภาพที่ 3.8 โฟลวชาร์ตโมดูล LCD

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.8(ต่อ) โฟลวชาร์ตโมดูล LCD

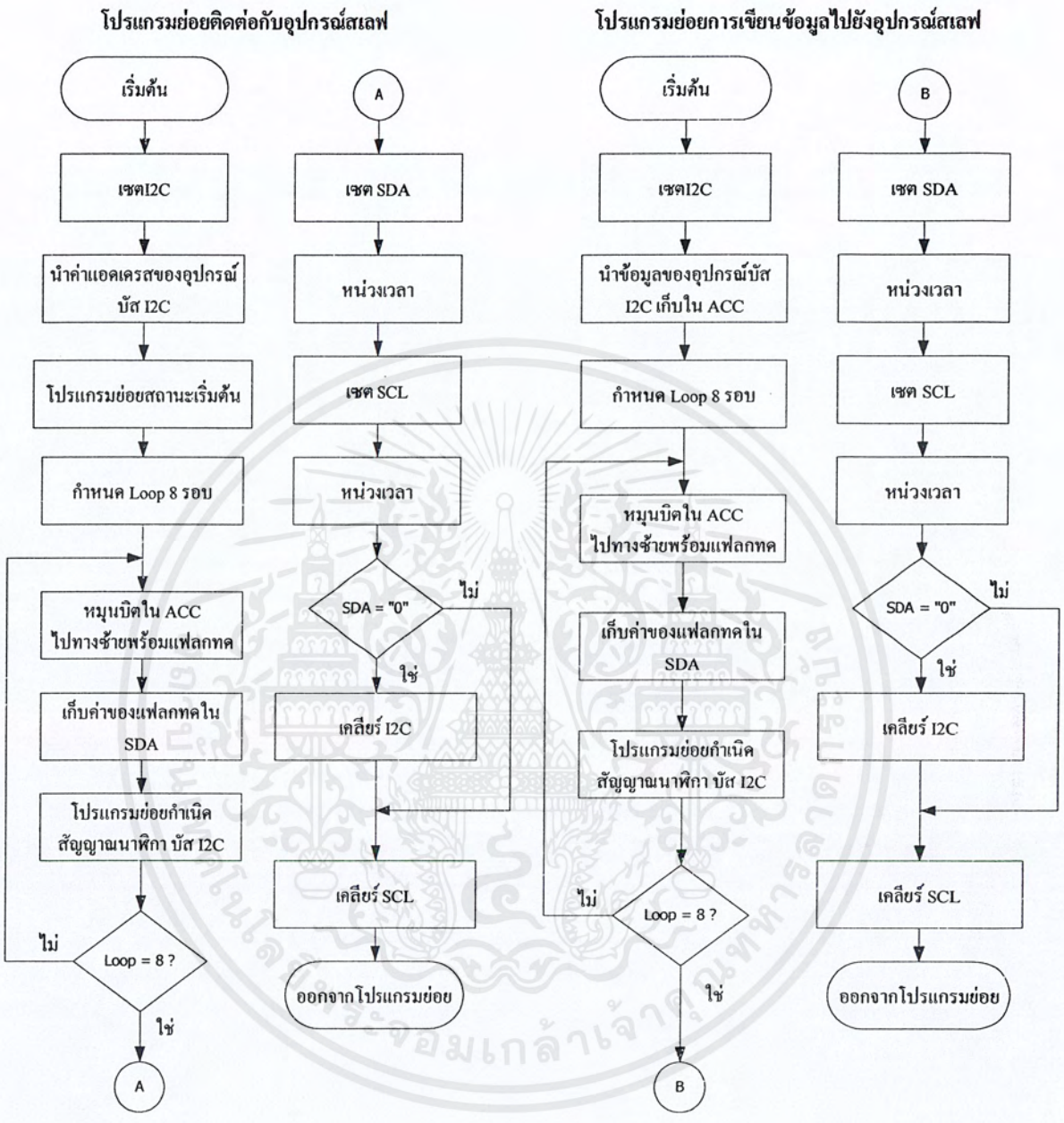
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.8(ต่อ) โฟลวชาร์ตโมดูล LCD

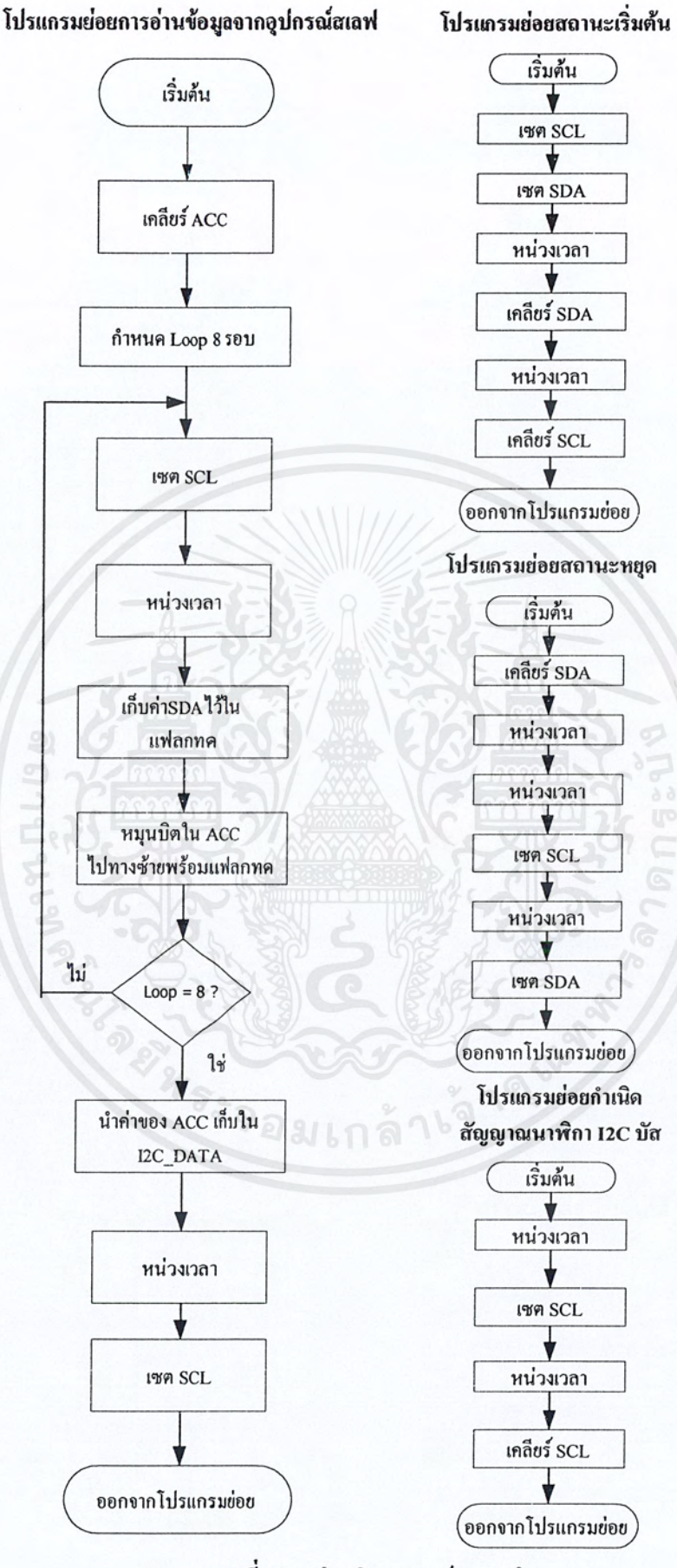
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5.5 โฟลวชาร์ตบัส I^2C



ภาพที่ 3.9 โฟลวชาร์ตบัส I^2C

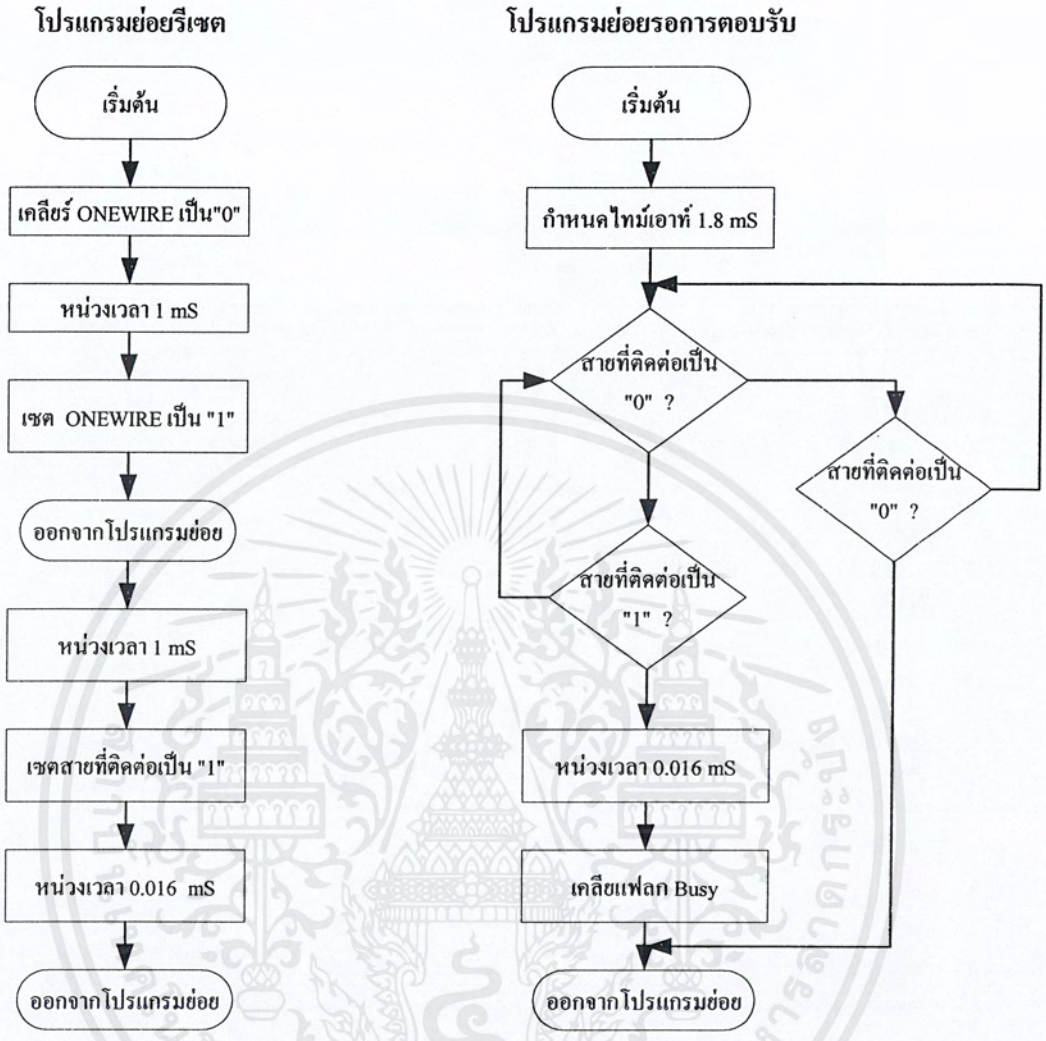
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 3.9 (ต่อ) โฟลวชาร์ตบัส I^2C

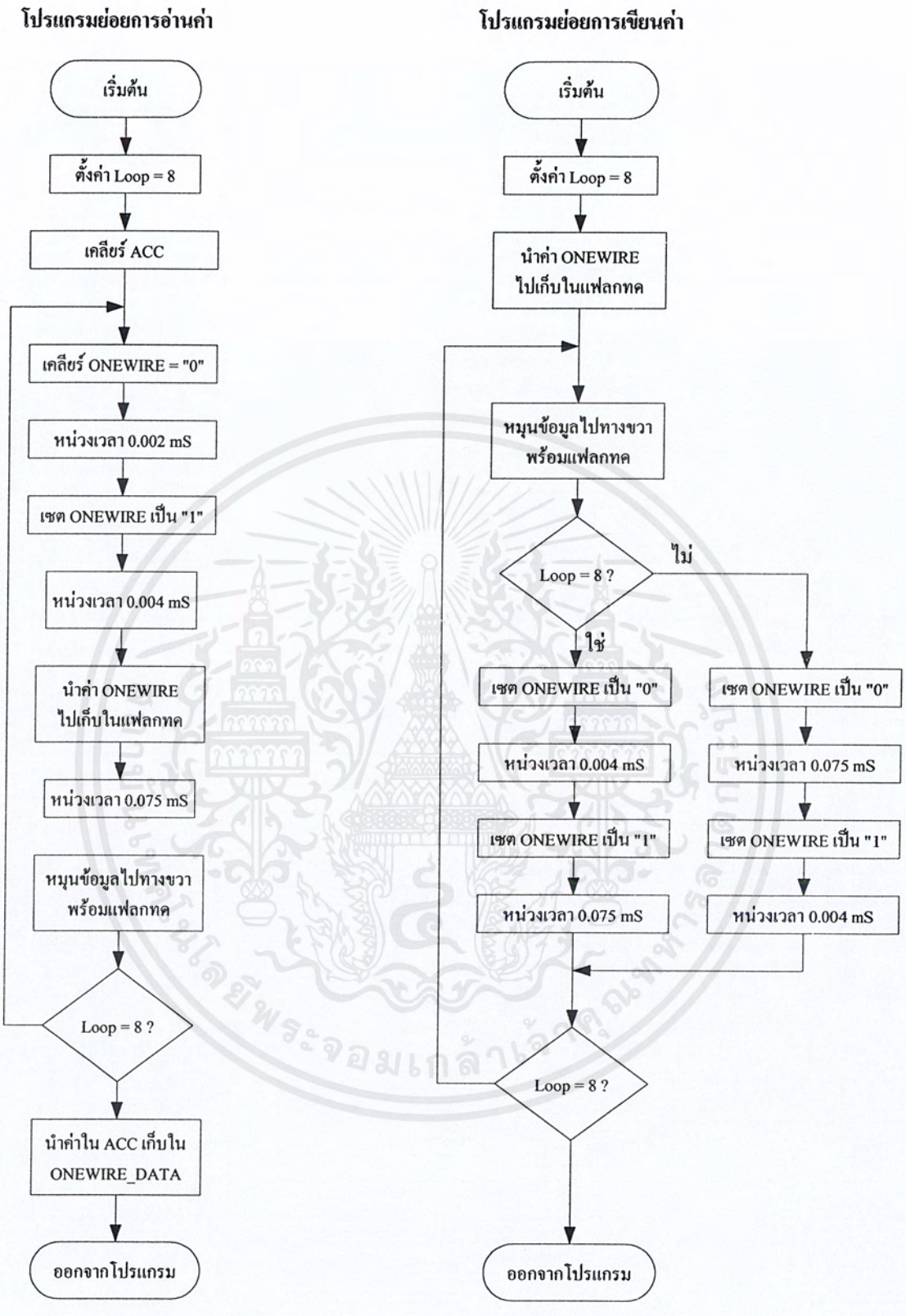
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5.6 โฟลวชาร์ต สำหรับการติดต่อแบบ 1-Wire™ Serial



ภาที่ 3.10 โฟลวชาร์ต สำหรับการติดต่อแบบ 1-Wire™ Serial

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่3.10(ต่อ) โฟลวชาร์ต สำหรับการติดต่อแบบ 1-Wire™ Serial

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
การเขียนโปรแกรมและการทดลอง

จะขออธิบายในส่วนของโปรแกรมแต่ละโมดูลซึ่งโมดูลทั้งหมดที่สร้างไว้มีดังนี้
โดยโมดูลทั้งหมดจะเขียนโปรแกรมเป็นแบบ Linking\Location

4.1 โมดูล ScanKey

จากโปรแกรมจะเป็นการ scankey แบบ 4 แถว 4หลัก โดยใช้หลักการสแกนไปทีละแถวและทีละหลักสลับกันไป ค่าที่ได้จากการสแกนคีย์จะเก็บใน KeyData ถ้าไม่มีการกดคีย์ใดๆ KeyData จะเท่ากับศูนย์ NAME MODULESANKEY

PUBLIC STARTKEYPAD ; ประกาศเพื่อให้โปรแกรมอื่นๆเรียกไปใช้ได้

```
ROW0        BIT   P1.0
ROW1        BIT   P1.1
ROW2        BIT   P1.2
ROW3        BIT   P1.3
COL3        BIT   P1.7
COL2        BIT   P1.6
COL1        BIT   P1.5
COL0        BIT   P1.4
KEYDATA     EQU   032H     ; เก็บค่าการกดคีย์แต่ละครั้งที่มีการสแกน
SCAN SEGMENT  CODE
RSEG  SCAN


STARTKEYPAD:  MOV     P1,#0FFH
              MOV     KEYDATA,#0
CHKCOL0:      CLR     Col0
              MOV     A,P1
              ANL     A,#00FH
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                CJNE    A,#00FH,COL0_DETECT
                AJMP    CHKCOL1
COL0_DETECT:    MOV     KEYDATA,#01
                AJMP    GETROW
CHKCOL1:        SETB    COL0
                CLR     COL1
                MOV     A,P1
                ANL     A,#00FH
                CJNE    A,#00FH,COL1_DETECT
                AJMP    CHKCOL2
COL1_DETECT:    MOV     KEYDATA,#02
                AJMP    GETROW
CHKCOL2:        SETB    COL1
                CLR     COL2
                MOV     A,P1
                ANL     A,#00FH
                CJNE    A,#00FH,COL2_DETECT
                AJMP    CHKCOL3
COL2_DETECT:    MOV     KEYDATA,#03
                AJMP    GETROW
CHKCOL3:        SETB    COL2
                CLR     COL3
                MOV     A,P1
                ANL     A,#00FH
                CJNE    A,#00FH,COL3_DETECT
                RET
COL3_DETECT:    MOV     KEYDATA,#04
                AJMP    GETROW
GETROW:         CLR     COL0
                CLR     COL1
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                CLR     COL2
                CLR     COL3
                JB      ROW0,CHKROW1
                RET
CHKROW1:        JB      ROW1,CHKROW2
                MOV     A,KEYDATA
                ADD     A,#4
                MOV     KEYDATA,A
                RET
CHKROW2:        JB      ROW2,CHKROW3
                MOV     A,KEYDATA
                ADD     A,#8
                MOV     KEYDATA,A
                RET
CHKROW3:        JB      ROW3,RETURN
                MOV     A,KEYDATA
                ADD     A,#12
                MOV     KEYDATA,A
RETURN:         RET
                END
```

4.2 โมดูล Segment

จากโปรแกรมจะเป็นการนำค่าข้อมูลออกสู่ 7-Segment โดยจะแบ่งออกเป็น 2 ส่วน คือ MainShow ซึ่งจะใช้แสดงข้อมูลบน 7-Segment แบบหลายหลักพร้อมกันโดยหลักการคือ หน่วงเวลาสลับกันที่ละหลักประมาณ1mS ซึ่งจะเร็วจนสายตามองไม่เห็นการกระพริบ Show1Digit ใช้แสดงข้อมูลบน 7-Segment ทีละหลัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
NAME    MODULESEGMENT
EXTRN   CODE(STARTKEYPAD,DELAY1_MS,CHEND);ประกาศส่วนของ โปรแกรมย่อยที่
                                          จะเรียกใช้จากโมดูลอื่นๆ
PUBLIC  MAINSHOW,SHOW_DSP,SHOW1DIGIT

PROGSEG  SEGMENT CODE
TABLESEG SEGMENT CODe

DSP1            BIT     P3.3
DSP2            BIT     P3.2
FLAG            EQU     02FH
KEYPRESSED      BIT     FLAG.0
R_SHF           BIT     FLAG.1
CHECKEND        EQU     035H
DSP1_BUFFER     EQU     033H
DSP2_BUFFER     EQU     034H
KEYDATA         EQU     032H
SEGDATA         EQU     03BH
                RSEG    PROGSEG
MAINSHOW:       MOV     P0,#00000000B
                SETB    DSP1
                SETB    DSP2
                MOV     P1,#11111111B

MAIN:           MOV     DSP1_BUFFER,#0
                MOV     DSP2_BUFFER,#0
                MOV     FLAG,#0
LOOP:           ACALL   STARTKEYPAD
                MOV     A,KEYDATA
                JZ      NEXT
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                JB      KEYPRESSED,SHOW
                SETB    KEYPRESSED
LEFT_SHIFT:     MOV     DSP2_BUFFER,DSP1_BUFFER
                MOV     DSP1_BUFFER,KEYDATA
                AJMP    SHOW
NEXT:           CLR     KEYPRESSED
SHOW:           ACALL   SHOW_DSP
                CLR     R_SHF
                ACALL   CHEND
                MOV     R2,CHECKEND
                CJNE    R2,#01,LOOP
                RET
SHOW_DSP:       MOV     R4,#5
SCAN_DSP_LOOP:  MOV     A,DSP1_BUFFER
                MOV     DPTR,#Num0
                MOVC    A,@A+DPTR
                MOV     P0,A
                CLR     DSP1
                ACALL   Delay1_ms
                SETB    DSP1
                MOV     A,DSP2_BUFFER
                MOV     DPTR,#Num0
                MOVC    A,@A+DPTR
                MOV     P0,A
                CLR     DSP2
                ACALL   Delay1_ms
                SETB    DSP2
                DJNZ    R4,SCAN_DSP_LOOP
                RET
SHOW1DIGIT:     CLR _   DSP1
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        SETB    DSP2
        MOV     DPTR,#Num0
        MOV     A,SEGData
        MOVC    A,@A+DPTR
        MOV     P0,A
        RET
    RSEG TABLESEG
Num0:   DB      00111111B
Num1:   DB      00000110B
Num2:   DB      01011011B
Num3:   DB      01001111B
Num4:   DB      01100110B
Num5:   DB      01101101B
Num6:   DB      01111101B
Num7:   DB      00000111B
Num8:   DB      01111111B
Num9:   DB      01101111B
ChaA:   DB      01110111B
ChaB:   DB      01111100B
ChaC:   DB      00111001B
ChaD:   DB      01011110B
ChaE:   DB      01111001B
ChaF:   DB      01110001B
        END
```

4.3 โมดูล LCD

บอร์ดการทดลองเราจะใช้ LCD 2 ชุด คือ 2 แถว 20 หลักของตัวมาสเตอร์และ 1 แถว 16 หลักของตัวสเลฟ ซึ่งรายละเอียดของแต่ละโปรแกรมย่อยจะอธิบายไว้ในตัวโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
NAME      MODULELCD
EXTRN     CODE(Delay_100ms,Delay10_ms)
PUBLIC    INILCD,LCDCLR,LCDHOME,LCDOFF,LCDCLK
PUBLIC    WRLINE_LCD1L,W1LINELCD,W2LINELCD
PUBLIC    LcdOn,LcdBlink,LcdLShf,LcdRShf,SetAddrLcd,WrCharLcd,WR3CHAR_LCD
PROGLCD   SEGMENT  CODE
LcdEn          BIT       P3.4  ;Slave Use P3.6
LcdRs          BIT       P3.5  ;Slave Use P3.7
LcdAddr        EQU       030H
LcdData        EQU       031H
LCD_PTR        EQU       049H

RSEG      ProgLcd
IniLcd:        ACALL     Delay_100ms ;เตรียมLcdให้พร้อมสำหรับการทำงาน
               CLR       LcdRs
               MOV       P2,#00111000B
               ACALL     LcdClk
               ACALL     Delay10_ms
               MOV       P2,#00111000B
               ACALL     LcdClk
               ACALL     LcdOff
               ACALL     LcdClr
               MOV       P2,#00000110B
               ACALL     LcdClk
               ACALL     LcdHome

LcdClr:        CLR       LcdRs       ;Clear ค่าใน DDRam
               MOV       P2,#00000001B
               ACALL     LcdClk
               RET
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
LcdHome:    CLR     LcdRs       ;Clear DDRam=00H และ Cursor อยู่ซ้ายสุด)
            MOV     P2,#00000010B
            ACALL   LcdClk
            RET

LcdOff:     CLR     LcdRs       ;ปิดการแสดงผลแต่ข้อมูลยังคงอยู่
            MOV     P2,#00001000B
            ACALL   LcdClk
            RET

LcdClk:     SETB    LcdEn
            ACALL   LcdDelay
            CLR     LcdEn
            ACALL   LcdDelay
            RET
LcdOn:      CLR     LcdRs       ;เปิดการแสดงผล
            MOV     P2,#00001100B
            ACALL   LcdClk
            RET

LcdBlink:   CLR     LcdRs       ;ให้ Cursor กระพริบ
            MOV     P2,#00001111B
            ACALL   LcdClk
            RET

LcdLShf:    CLR     LcdRs       ;เลื่อนตำแหน่งการแสดงผล
            MOV     P2,#00011000B
            ACALL   LcdClk
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                RET
SetAddrLcd:     CLR     LcdRs       ;กำหนด Address การแสดงผล
                MOV     A,LcdAddr
                SETB    Acc.7
                MOV     P2,A
                ACALL   LcdClk
                RET

WrCharLcd:      SETB    LcdRs       ;เขียนตัวอักษรในตำแหน่งAddressที่กำหนด
                MOV     P2,LcdData
                ACALL   LcdClk
                ACALL   LcdOn
                RET

WRLINE_LCD1L:   MOV     R0,#0       ;เขียนตัวอักษรทั้งบันทัดของLcd 1แถว 16 หลัก

WRLINE_LCD_1:   SETB    LcdRs
                CLR     A
                MOVC    A,@A+DPTR
                MOV     P2,A
                ACALL   LcdClk
                INC     DPTR
                INC     R0
                CJNE    R0,#8,WRLINE_LCD_1
                MOV     LcdAddr,#040H
                ACALL   SetAddrLcd


WRLINE_LCD_2:   SETB    LcdRs
                CLR     A
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                MOVC    A,@A+DPTR
                MOV     P2,A
                ACALL   LcdClk
                INC     DPTR
                INC     R0
                CJNE    R0,#16,WRLINE_LCD_2
                ACALL   LcdOn
                RET

W1LINELCD:      MOV     R0,#0       ;เขียนตัวอักษรบนบันทัดแรกของLcd 2 แถว 20
หลัก
W1LINE1L:       SETB    LcdRs
                CLR     A
                MOVC    A,@A+DPTR
                MOV     P2,A
                ACALL   LcdClk
                INC     DPTR
                INC     R0
                CJNE    R0,#8,W1LINE1L
                MOV     LcdAddr,#008H
                ACALL   SetAddrLcd

W1LINE2L:       SETB    LcdRs
                CLR     A
                MOVC    A,@A+DPTR
                MOV     P2,A
                ACALL   LcdClk
                INC     DPTR
                INC     R0
                CJNE    R0,#19,W1LINE2L
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ACALL   LcdOn
                RET

W2LINELCD:      MOV     R0,#0   ;เขียนตัวอักษรบนบันทัดสองของLcd 2 แถว 20
หลัก
W2LINE2L:       SETB    LcdRs
                CLR     A
                MOVC    A,@A+DPTR
                MOV     P2,A
                ACALL   LcdClk
                INC     DPTR
                INC     R0
                CJNE    R0,#8,W2LINE2L
                MOV     LcdAddr,#048H
                ACALL   SetAddrLcd

W2LINEL:        SETB    LcdRs
                CLR     A
                MOVC    A,@A+DPTR
                MOV     P2,A
                ACALL   LcdClk
                INC     DPTR
                INC     R0
                CJNE    R0,#19,W2LINEL
                ACALL   LcdOn
                RET

WR3CHAR_LCD:    MOV     R0,#0   ;เขียนตัวอักษรครั้งละ3ตัวอักษร
                MOV     A,LCD_PTR
                DEC     A
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    MOV     B,#3
                    MUL     AB
WR3CHAR_LCD_1:      SETB    LcdRs
                    MOVC    A,@A+DPTR
                    MOV     P2,A
                    ACALL   LcdClk
                    INC     DPTR
                    INC     R0
                    MOV     A,LCD_PTR
                    DEC     A
                    MOV     B,#3
                    MUL     AB
                    CJNE    R0,#3,WR3CHAR_LCD_1
                    LCALL   LcdOn
                    RET

LcdDelay:           MOV     R7,#002
LcdDelay1:          MOV     R6,#0E6H
LcdDelay2:          NOP
                    NOP
                    DJNZ    R6,LcdDelay2
                    DJNZ    R7,LcdDelay1
                    RET
                    END
```

4.4 โมดูล I2C

การสื่อสารของแต่ละส่วนในการทดลองซึ่งได้แก่ PCF8574(ขยายพอร์ท), PCF8591(AtoD-DToA), DS1307(RealTime),MCS-51Slave ทุกส่วนจะสื่อสารกันบนบัส I2Cทั้งหมด รายละเอียดของการติดต่อกับอุปกรณ์ต่างๆจะอธิบายไว้ในตัวโปรแกรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
NAME  ModuleI2CBus
PUBLIC Pcf8574Rd,Pcf8574Wr,PCF8591_RD,PCF8591_WR,RTC_RD,RTC_WR

   ProgI2C  SEGMENT   Code
SDA             BIT     P3.6
SCL             BIT     P3.7
Flag            EQU     02FH
I2C_Ack         BIT     Flag.2
I2C_Addr        EQU     036H
I2CData         EQU     037H
IOData          EQU     038H
OutData         EQU     039H
InData          EQU     03AH
CONTROL         EQU     03BH
DA_DATA         EQU     03CH
AD_DATA         EQU     03EH
SECONDS         EQU     040H
MINUTES         EQU     041H
HOURS           EQU     042H
DAY             EQU     043H
DATE            EQU     044H
MONTH           EQU     045H
YEAR            EQU     046H
CONTROLR        EQU     047H        ;กำหนด Address ของตัวอุปกรณ์ Slave
PCF8574ID       EQU     01110000B
PCF8591ID       EQU     10010000B
RTC_ID          EQU     11010000B
SlaveR          EQU     00000010B
SlaveW          EQU     00000001B
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                Rseg        ProgI2C
RTC_RD:         MOV         I2C_Addr,#RTC_ID    ;อ่านค่าข้อมูลจากICRealTime
                                                (Ds1307)
                LCALL       I2C_Slave
                MOV         I2CData,#000H
                LCALL       I2CData_Wr
                MOV         I2C_Addr,#RTC_ID+1
                LCALL       I2C_Slave
                LCALL       I2CData_Rd
                MOV         SECONDS,I2CData
                LCALL       I2C_Ack_Bit
                LCALL       I2CData_Rd
                MOV         MINUTES,I2CData
                LCALL       I2C_Ack_Bit
                LCALL       I2CData_Rd
                MOV         HOURS,I2CData
                LCALL       I2C_Ack_Bit
                LCALL       I2CData_Rd
                MOV         DAY,I2CData
                LCALL       I2C_Ack_Bit
                LCALL       I2CData_Rd
                MOV         DATE,I2CData
                LCALL       I2C_Ack_Bit
                LCALL       I2CData_Rd
                MOV         MONTH,I2CData
                LCALL       I2C_Ack_Bit
                LCALL       I2CData_Rd
                MOV         YEAR,I2CData
                LCALL       I2C_Ack_Bit
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                LCALL   I2CData_Rd
                MOV     CONTROLR,I2CData
                LCALL   I2C_Nack_Bit
                LCALL   I2CStop
                RET

RTC_WR:         MOV     I2C_Addr,#RTC_ID; เขียนค่าข้อมูลจากICRealTime
                                          (Ds1307)
                LCALL   I2C_Slave
                MOV     I2CData,#000H
                LCALL   I2CData_Wr
                MOV     I2CData,SECONDS
                LCALL   I2CData_Wr
                MOV     I2CData,MINUTES
                LCALL   I2CData_Wr
                MOV     I2CData,HOURS
                LCALL   I2CData_Wr
                MOV     I2CData,DAY
                LCALL   I2CData_Wr

                MOV     I2CData,DATE
                LCALL   I2CData_Wr
                MOV     I2CData,MONTH
                LCALL   I2CData_Wr
                MOV     I2CData,YEAR
                LCALL   I2CData_Wr
                MOV     I2CData,CONTROLR
                LCALL   I2CData_Wr
                LCALL   I2CStop
                RET
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
PCF8591_RD:     MOV     I2C_Addr,#PCF8591ID+1; อ่านค่าข้อมูลจาก
                                              IC AtoD-DtoA(Pcf8591)
                ACALL   I2C_Slave
                ACALL   I2CData_Rd
                MOV     AD_DATA,I2CData
                ACALL   I2C_Nack_Bit
                ACALL   I2CStop
                RET
PCF8591_WR:     MOV     I2C_Addr,#PCF8591ID ;เขียนค่าข้อมูลจากIC
                                              AtoD-DtoA(Pcf8591)
                ACALL   I2C_Slave
                MOV     I2CData,CONTROL
                ACALL   I2CData_Wr
                MOV     I2CData,DA_DATA
                ACALL   I2CData_Wr
                ACALL   I2CStop
                RET
Pcf8574Rd:      MOV     I2C_Addr,#PCF8574ID+1     ; อ่านค่าข้อมูลจากIC
                                                  ขยายพอร์ท(Pcf8574)
                ACALL   I2C_Slave
                ACALL   I2CData_Rd
                MOV     INData,I2CData
                ACALL   I2C_Nack_Bit
                ACALL   I2CStop
                RET

Pcf8574Wr:      MOV     I2C_Addr,#PCF8574ID ; เขียนค่าข้อมูลจากIC ขยาย
                                              พอร์ท(Pcf8574)
                ACALL   I2C_Slave
                MOV     I2CData,IOData
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ACALL   I2CData_Wr
                ACALL   I2CStop
                RET

I2CData_Wr:     PUSH    Acc             ; เขียนข้อมูลไปยังอุปกรณ์Slave
                                          ต่างๆ
                SETB    I2C_Ack
                MOV     A,I2CData
                MOV     R5,#008
I2CWr1:         RLC     A
                MOV     Sda,C
                ACALL   I2CClk
                DJNZ    R5,I2CWr1
                SETB    Sda
                ACALL   I2CDelay
                SETB    Scl
                ACALL   I2CDelay
                JB      Sda,I2CWr2
                CLR     I2C_Ack
I2CWr2:         CLR     Scl
                POP     Acc
                RET

I2CData_Rd:     PUSH    Acc             ; อ่านข้อมูลมาจากอุปกรณ์Slaveต่างๆ
                CLR     A
                MOV     R5,#008
I2CRd1:         ACALL   I2CDelay
                SETB    Scl
                ACALL   I2CDelay
                MOV     C,Sda
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                RLC     A
                CLR     Scl
                DJNZ    R5,I2CRd1
                MOV     I2CData,A
                POP     Acc
                RET

I2C_Slave:      PUSH    Acc     ;แจ้งอุปกรณ์Slaveให้ทราบว่าจะมีการติดต่อ
                SETB    I2C_Ack
                MOV     A,I2C_Addr
                ACALL   I2CStart
                MOV     R5,#008
I2CSlave1:      RLC     A
                MOV     Sda,C
                ACALL   I2CClk
                DJNZ    R5,I2CSlave1
                SETB    Sda
                ACALL   I2CDelay
                SETB    Scl
                ACALL   I2CDelay
                JB      Sda,I2CSlave2
                CLR     I2C_Ack

I2CSlave2:      CLR     Scl
                POP     Acc
                RET

I2CStart:       JNB     Scl,I2CStart1           ;สภาวะStart
                CLR     Scl
I2CStart1:      SETB    Sda
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                SETB    Scl
                ACALL   I2CDelay
                CLR     Sda
                ACALL   I2CDelay
                CLR     Scl
                RET

I2CStop:        JNB     Scl,I2CStop1        ;สภาวะStop
                CLR     Scl
I2CStop1:       CLR     Sda
                ACALL   I2CDelay
                SETB    Scl
                ACALL   I2CDelay
                SETB    Sda
                RET

I2CClk:         ACALL   I2Cdelay            ;I2CClock
                SETB    Scl
                ACALL   I2CDelay
                CLR     Scl
                RET

I2C_Ack_Bit:    CLR     Sda;I2C Acknowledge
                ACALL   I2CDelay
                ACALL   I2CClk
                SETB    Sda
                RET
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
I2C_Nack_bit:   SETB    Sda     ;I2C Not AcKnowledge
                ACALL   I2CDelay
                ACALL   I2CClk
                SETB    Scl
                RET

I2CDelay:       MOV     R6,#00CH
I2CDelay1:      NOP
                NOP
                DJNZ    R6,I2CDelay1
                RET
                END
```

4.5 โมดูล A To D และ D To A

ติดต่อกับมาสเตอร์บนบัส I2C และนำค่าที่อ่านได้แสดงยังLCDของตัวมาสเตอร์

```
NAME            AtoDAndDtoA
EXTRN           Code(IniLcd,PCF8591_WR,PCF8591_RD,SetAddrLcd,W2LINELCD)
EXTRN           Code(Delay_1S,WrCharLcd,ChEND)
PUBLIC          MainAdTDa
MainPro SEGMENT Code
CONTROL         EQU   03BH
DA_DATA         EQU   03CH
CHANNEL         EQU   03DH
AD_DATA         EQU   03EH
BUFFER          EQU   03FH
LcdAddr         EQU   030H
LcdData         EQU   031H
CheckEnd        EQU   035H
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                RSEG    MainPro
MainAdTDa:      MOV     DA_DATA,#0
MAIN_LOOP:      MOV     CHANNEL,#0
                MOV     R1,#BUFFER
                MOV     R4,#4

CONVERSION_LOOP: MOV    A,CHANNEL
                ADD     A,#01000000B
                MOV     CONTROL,A
                ACALL   PCF8591_WR
                ACALL   PCF8591_RD
                ACALL   PCF8591_RD
                MOV     @R1,AD_DATA
                INC     R1
                INC     CHANNEL
                DJNZ    R4,CONVERSION_LOOP
                MOV     LcdAddr,#040H
                ACALL   SetAddrLcd
                MOV     DPTR,#SCR_DA
                ACALL   W2LINELCD
                MOV     LcdAddr,#04CH
                ACALL   SetAddrLcd
                MOV     LcdData,DA_DATA
                ACALL   HEX2LCD
                INC     DA_DATA
                ACALL   Delay_1S
                ACALL   Delay_1S
                MOV     R1,#BUFFER
                MOV     LcdAddr,#040H
                ACALL   SetAddrLcd
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
MOV     DPTR,#SCR_AD01
ACALL   W2LINELCD
MOV     LcdAddr,#044H
ACALL   SetAddrLcd
MOV     LcdData,@R1
ACALL   HEX2LCD

INC     R1
MOV     LcdAddr,#04CH
ACALL   SetAddrLcd
MOV     LcdData,@R1
ACALL   HEX2LCD
ACALL   Delay_1S
ACALL   Delay_1S
INC     R1
MOV     LcdAddr,#040H
ACALL   SetAddrLcd
MOV     DPTR,#SCR_AD23
ACALL   W2LINELCD

MOV     LcdAddr,#044H
ACALL   SetAddrLcd
MOV     LcdData,@R1
ACALL   HEX2LCD
INC     R1
MOV     LcdAddr,#04CH
ACALL   SetAddrLcd
MOV     LcdData,@R1
ACALL   HEX2LCD
ACALL   Delay_1S
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ACALL   Delay_1S
                ACALL   ChEND
                MOV     R2,CheckEnd
                CJNE    R2,#01,MAIN_LOOP
                RET

HEX2LCD:        PUSH    ACC
                MOV     A,LcdData
                MOV     B,#16
                DIV     AB
                ADD     A,#030H
                MOV     LcdData,A
                ACALL   HEX_CHK
                ACALL   WrCharLcd
                MOV     A,B
                ADD     A,#030H
                MOV     LcdData,A
                ACALL   HEX_CHK
                ACALL   WrCharLcd
                POP     ACC
                RET
HEX_CHK:        MOV     A,LcdData
                CJNE    A,#03AH,CHK_OTHER
CHK_OTHER:      JNC     CONV_2_ALPHA
                RET
CONV_2_ALPHA:   ADD     A,#7
                MOV     LcdData,A
                RET

SCR_AD01:       DB      'Ch0: H Ch1: H   '
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SCR_AD23:       DB      'Ch2: H Ch3: H   '
SCR_DA:         DB      'Analog Out :  H  '
                END
```

4.6 โมดูล RealTime

ติดต่อกับมาสเตอร์บนบัสI2Cและนำค่าวันที่และเวลาที่อ่านได้แสดงยังLCDของตัวมาสเตอร์และยังสามารถตั้งค่าวันที่และเวลาได้

```
NAME        RealTime

EXTRN       Code(RTC_RD,SetAddrLcd,WrCharLcd,WR3CHAR_LCD,StartKeypad)
EXTRN       Code(LcdClr,LcdBlink,W2LINELCD,RTC_WR)
PUBLIC      LOOPRT

        ProgReal  SEGMENT  Code

LcdAddr         EQU     030H
LcdData         EQU     031H
I2C_Addr        EQU     036H
I2C_Data        EQU     037H
KeyData         EQU     032H
BUFFERT         EQU     048H
LCD_PTR         EQU     049H
SECONDS         EQU     040H
MINUTES         EQU     041H
HOURS           EQU     042H
DAY             EQU     043H
DATE            EQU     044H
MONTH           EQU     045H
YEAR            EQU     046H
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
CONTROLR        EQU     047H
        RSEG    ProgReal
LOOPRT:         ACALL   RTC_RD
                MOV     LcdAddr,#040H
                ACALL   SetAddrLcd
                MOV     LcdData,#' '
                ACALL   WrCharLcd
                MOV     LcdData,DATE
                ACALL   BCD2LCD
                MOV     LcdData,#' '
                ACALL   WrCharLcd
                MOV     A,MONTH
                CJNE    A,#010H,WR_CHK_MONTH_1
                MOV     A,#00AH
                AJMP    WRITE_MONTH_NX
WR_CHK_MONTH_1: CJNE    พอร์ท(Pcf8574)A,#011H,WR_CHK_MONTH_2
                MOV     A,#00BH
                AJMP    WRITE_MONTH_NX
WR_CHK_MONTH_2: CJNE    A,#012H,WRITE_MONTH_NX
                MOV     A,#00CH
WRITE_MONTH_NX: MOV     LCD_PTR,A
                MOV     DPTR,#MONTH_JAN
                ACALL   WR3CHAR_LCD
                MOV     LcdData,#' '
                ACALL   WrCharLcd
                MOV     LcdData,YEAR
                ACALL   BCD2LCD
                MOV     LcdData,#' '
                ACALL   WrCharLcd
                MOV     A,HOURS
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                    ANL     A,#00110000B
                    JZ      WRITE_TIME_HN
                    SWAP    A
                    ADD     A,#030H
                    AJMP    WRITE_TIME_HH
WRITE_TIME_HN:      MOV     A,#' '
WRITE_TIME_HH:      MOV     LcdData,A
                    ACALL   WrCharLcd
                    MOV     A,HOURS
                    ANL     A,#00001111B
                    ADD     A,#030H
                    MOV     LcdData,A
                    ACALL   WrCharLcd
                    MOV     A,SECONDS
                    ANL     A,#001H
                    JNZ     WRITE_SPACE
                    MOV     LcdData,#':'
                    ACALL   WrCharLcd
                    JMP     WRITE_MINUTES
WRITE_SPACE:        MOV     LcdData,#' '
                    ACALL   WrCharLcd

WRITE_MINUTES:      MOV     LcdData,MINUTES
                    ACALL   BCD2LCD
                    ACALL   StartKeypad
                    MOV     A,KeyData
                    CJNE    A,#0DH,CheckSetD
                    ACALL   SET_TIME_1

CheckSetD:          CJNE    A,#0EH,ChkEd
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ACALL   SET_DATE

ChkEd:          CJNE    A,#10H,BackLRT
                JMP     TurnToMain
BackLRT:        AJMP    LOOPRT
TurnToMain:     RET

SET_TIME_1:     AJMP    SET_TIME
SET_DATE:       Acall   LcdClr
                MOV     LcdAddr,#040H
                ACALL   SetAddrLcd
                MOV     DPTR,#SCR_SET_DATE
                ACALL   W2LINELCD
                MOV     LcdAddr,#048H
                ACALL   SetAddrLcd
                ACALL   LcdBlink
                ACALL   WAIT_KEYPRESSED
                MOV     BUFFERT,KeyData
                ACALL   KPAD2LCD
                ACALL   LcdBlink
                ACALL   WAIT_KEY
                ACALL   WAIT_KEYPRESSED
                MOV     BUFFERT+1,KeyData
                ACALL   KPAD2LCD
                ACALL   WAIT_KEY
                ACALL   BUFFER2ACC
                MOV     DATE,A
                MOV     LcdAddr,#04BH
                ACALL   SetAddrLcd
                ACALL   LcdBlink
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
ACALL     WAIT_KEYPRESSED
MOV       BUFFERT,KeyData
ACALL     KPAD2LCD
ACALL     LcdBlink
ACALL     WAIT_KEY

ACALL     WAIT_KEYPRESSED
MOV       BUFFERT+1,KeyData
ACALL     KPAD2LCD
ACALL     WAIT_KEY

ACALL     BUFFER2ACC
MOV       MONTH,A
MOV       LcdAddr,#04EH
ACALL     SetAddrLcd
ACALL     LcdBlink
ACALL     WAIT_KEYPRESSED
MOV       BUFFERT,KeyData
ACALL     KPAD2LCD
ACALL     LcdBlink
ACALL     WAIT_KEY

ACALL     WAIT_KEYPRESSED
MOV       BUFFERT+1,KeyData
ACALL     KPAD2LCD
ACALL     WAIT_KEY

ACALL     BUFFER2ACC
MOV       YEAR,A
ACALL     RTC_WR
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ACALL    LcdClr
                AJMP     LOOPRT

SET_TIME:       ACALL    LcdClr
                MOV      LcdAddr,#040H
                ACALL    SetAddrLcd
                MOV      DPTR,#SCR_SET_TIME
                ACALL    W2LINELCD
                MOV      LcdAddr,#048H
                ACALL    SetAddrLcd
                ACALL    LcdBlink
                ACALL    WAIT_KEYPRESSED
                MOV      BUFFERT,KeyData
                ACALL    KPAD2LCD
                ACALL    LcdBlink
                ACALL    WAIT_KEY
                ACALL    WAIT_KEYPRESSED
                MOV      BUFFERT+1,KeyData
                ACALL    KPAD2LCD
                ACALL    WAIT_KEY
                ACALL    BUFFER2ACC
                MOV      HOURS,A
                MOV      LcdAddr,#04BH
                ACALL    SetAddrLcd
                ACALL    LcdBlink
                ACALL    WAIT_KEYPRESSED
                MOV      BUFFERT,KeyData
                ACALL    KPAD2LCD
                ACALL    LcdBlink
                ACALL    WAIT_KEY
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ACALL   WAIT_KEYPRESSED
                MOV     BUFFERT+1,KeyData
                ACALL   KPAD2LCD
                ACALL   WAIT_KEY
                ACALL   BUFFER2ACC
                MOV     MINUTES,A
                MOV     LcdAddr,#04EH
                ACALL   SetAddrLcd
                ACALL   LcdBlink
                ACALL   WAIT_KEYPRESSED
                MOV     BUFFERT,KeyData
                ACALL   KPAD2LCD
                ACALL   LcdBlink
                ACALL   WAIT_KEY
                ACALL   WAIT_KEYPRESSED
                MOV     BUFFERT+1,KeyData
                ACALL   KPAD2LCD
                ACALL   WAIT_KEY
                ACALL   BUFFER2ACC
                MOV     SECONDS,A
                ACALL   RTC_WR
                ACALL   LcdClr
                AJMP    LOOPRT


BUFFER2ACC:     MOV     A,BUFFERT
                ANL     A,#00FH
                SWAP    A
                MOV     B,A
                MOV     A,BUFFERT+1
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ANL     A,#00FH
                ADD     A,B
                RET

KPAD2LCD:       MOV     A,KeyData
                ADD     A,#030H
                MOV     LcdData,A
                ACALL   WrCharLcd
                RET

BCD2LCD:        PUSH    ACC
                PUSH    B
                MOV     A,LcdData
                MOV     B,A
                ANL     A,#11110000B
                SWAP    A
                ADD     A,#030H
                MOV     LcdData,A
                ACALL   WrCharLcd
                MOV     A,B
                ANL     A,#00001111B
                ADD     A,#030H
                MOV     LcdData,A
                ACALL   WrCharLcd
                POP     B
                POP     ACC
                RET

WAIT_KEY:       MOV     A,P1
                ANL     A,#00FH
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                CJNE    A,#00FH,WAIT_KEY
                RET

WAIT_KEYPRESSED: ACALL  StartKeypad
                MOV     A,KeyData
                CJNE    A,#0,CHK_KEY_NEXT
                AJMP    WAIT_KEYPRESSED
CHK_KEY_NEXT:   CJNE    A,#10,CHK_KEY_0
                JMP     WAIT_KEYPRESSED
CHK_KEY_0:      CJNE    A,#11,CHK_VALID_KEY
                MOV     KeyData,#0
                RET
CHK_VALID_KEY:  JNC     WAIT_KEYPRESSED
                RET
SCR_SET_DATE:   DB      ' Date : dd/mm/yy  '
SCR_SET_TIME:   DB      ' Time : hh/mm/ss  '

MONTH_JAN:      DB      'Jan'
MONTH_FEB:      DB      'Feb'
MONTH_MAR:      DB      'Mar'
MONTH_APR:      DB      'Apr'
MONTH_MAY:      DB      'May'
MONTH_JUN:      DB      'Jun'
MONTH_JUL:      DB      'Jul'
MONTH_AUG:      DB      'Aug'
MONTH_SEP:      DB      'Sep'
MONTH_OCT:      DB      'Oct'
MONTH_NOV:      DB      'Nov'
MONTH_DEC:      DB      'Dec'
                END
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.7 โมดูล Temperature

สาเหตุที่เขียนโมดูลนี้ขึ้นมาก็เพื่อนำไปประยุกต์ใช้กับโมดูล serial เพื่อส่งค่าอุณหภูมิที่อ่านได้ออกมาแสดงยังเครื่องคอมพิวเตอร์ซึ่งการติดต่อกับตัววัดอุณหภูมิจะเป็นแบบ 1-Wire

```
Name   Temperature
Extrn          Code(IniLcd,SetAddrLcd,WRLINE_LCD1L,Delay_1S,WrCharLcd,Delay1_ms)
Public         MainTemp
ProgTemp       Segment  Code
ONEWIRE              BIT        P3.3
FLAG                 EQU        02FH
BUSY                 BIT        FLAG.0
LcdAddr              EQU        030H
LcdData              EQU        031H
ONEWIRE_DATA         EQU        032H
      TEMP           EQU        033H
Rseg   ProgTemp
Start:               SETB       ONEWIRE
MainTemp:            ACALL      IniLcd
                     MOV        LcdAddr,#000H
                     ACALL      SetAddrLcd
                     MOV        DPTR,#TITLE_1
                     ACALL      WRLINE_LCD1L
                     ACALL      Delay_1S
                     ACALL      Delay_1S
                     MOV        LcdAddr,#000H
                     ACALL      SetAddrLcd
                     MOV        DPTR,#SCR_TEMP
                     ACALL      WRLINE_LCD1L
LOOP:                ACALL      DS1820_RST
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ACALL   DS1820_PRES
                MOV     ONEWIRE_DATA,#0CCH
                ACALL   DS1820_WR
                MOV     ONEWIRE_DATA,#044H
                ACALL   DS1820_WR
                SETB    BUSY

PRES_CHK_LOOP:  ACALL   DS1820_RST
                ACALL   DS1820_PRES
                JB      BUSY,PRES_CHK_LOOP
                NOP
                NOP
                NOP
                NOP

                ACALL   DS1820_RST
                ACALL   DS1820_PRES
                MOV     ONEWIRE_DATA,#0CCH
                ACALL   DS1820_WR
                MOV     ONEWIRE_DATA,#0BEH
                ACALL   DS1820_WR

                ACALL   DS1820_RD
                MOV     TEMP,ONEWIRE_DATA
                ACALL   DS1820_RST
                ACALL   DS1820_PRES
                MOV     LcdAddr,#040H
                ACALL   SetAddrLcd
                MOV     A,TEMP
                CLR     C
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                RRC     A
                MOV     LcdData,A
                ACALL   HEX2LCD
                MOV     LcdAddr,#044H
                ACALL   SetAddrLcd
                MOV     A,TEMP
                JNB     ACC.0,WRITE_0C
                MOV     LcdData,#'5'
                AJMP    WRITE_NEXT
WRITE_0C:       MOV     LcdData,#'0'
WRITE_NEXT:     ACALL   WrCharLcd
                ACALL   Delay_1S
                AJMP    LOOP
                Ret

HEX2LCD:        PUSH    ACC
                MOV     A,LcdData
                MOV     B,#100
                DIV     AB
                ADD     A,#030H
                CJNE    A,#030H,HEX2_LCD_NX
                MOV     A,#' '

HEX2_LCD_NX:    MOV     LcdData,A
                ACALL   WrCharLcd
                MOV     A,B
                MOV     B,#10
                DIV     AB
                ADD     A,#030H
                MOV     LcdData,A
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ACALL   WrCharLcd
                MOV     A,B
                ADD     A,#030H
                MOV     LcdData,A
                ACALL   WrCharLcd
                POP     ACC
                RET

DS1820_RD:      MOV     R4,#8
                CLR     A

DS1820_RD_LOOP: CLR     ONEWIRE
                NOP
                NOP
                SETB    ONEWIRE
                NOP
                NOP
                NOP
                NOP
                MOV     C,ONEWIRE
                ACALL   ONEWIRE_DELAY
                RRC     A
                DJNZ    R4,DS1820_RD_LOOP
                MOV     ONEWIRE_DATA,A
                RET

DS1820_WR:      MOV     R4,#8
                MOV     A,ONEWIRE_DATA

DS1820_WR_LOOP: RRC     A
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                JNC         DS1820_WR_L
                CLR         ONEWIRE
                NOP
                NOP
                NOP
                NOP
                SETB        ONEWIRE
                ACALL       ONEWIRE_DELAY
                AJMP        DS1820_WR_NX

DS1820_WR_L:    CLR         ONEWIRE
                ACALL       ONEWIRE_DELAY
                SETB        ONEWIRE
                NOP
                NOP
                NOP
                NOP

DS1820_WR_NX:   DJNZ        R4,DS1820_WR_LOOP
                RET

DS1820_RST:     CLR         ONEWIRE
                ACALL       Delay1_ms
                SETB        ONEWIRE
                MOV         R4,#8
                DJNZ        R4,$
                RET

DS1820_PRES:    MOV         R4,#8
DS1820_PRES_1:  MOV         R3,#0
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
DS1820_PRES_2:      JNB     ONEWIRE,DS1820_PRES_3
                    DJNZ    R3,DS1820_PRES_2
                    DJNZ    R4,DS1820_PRES_1
                    RET
DS1820_PRES_3:      JNB     ONEWIRE,$
                    MOV     R4,#8
                    DJNZ    R4,$
                    CLR     BUSY
                    RET

ONEWIRE_DELAY:      MOV     R6,#012H
ONEWIRE_DELAY_1:    NOP
                    NOP
                    DJNZ    R6,ONEWIRE_DELAY_1
                    RET

DELAY_50us:         MOV     R6,#00CH
DELAY_50us_1:       NOP
                    NOP
                    DJNZ    R6,DELAY_50us_1
                    RET

DELAY_100us:        MOV     R6,#017H
DELAY_100us_1:      NOP
                    NOP
                    DJNZ    R6,DELAY_100us_1
                    RET
TITLE_1:            DB      ' 1-Wire  DS1820 '
SCR_TEMP:           DB      ' Temp :   . ',0DFH,'C '
                    End
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.8 โมดูล Serial

ใช้เพื่อสื่อสารระหว่างคอนโทรลเลอร์กับคอนโทรลเลอร์และคอนโทรลเลอร์กับคอมพิวเตอร์ ซึ่งจะแบ่งเป็นสองส่วนหลักคือรับและส่งข้อมูล

```
NAME        ModuleSerial
PUBLIC      InitialSerial,Recive,Send
ProgSerial  SEGMENT  Code
BUFFER      EQU      050H          ; Buffer
DataSend    EQU      05FH
            Rseg     ProgSerial

InitialSerial:
            MOV      TMOD,#021H
            MOV      TH1,#0FDH
            MOV      TL1,#0FDH
            SETB     TR1
            MOV      SCON,#040H
            MOV      R0,#BUFFER
            SETB     REN
            Ret

Recive:
            JNB      RI,$
            CLR      RI
            MOV      A,SBUF
            MOV      @R0,A
            CJNE     A,#00DH,RX_NEXT
            Ret

Send:
            CLR      TI
            MOV      A,Datasend
            MOV      SBUF,A
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                JNB     TI,$
                CLR     TI
                Ret

RX_NEXT:        INC     R0          ; Increase pointer
                AJMP    Recive      ; Back to do loop
                End
```

การทดลอง

การทำการทดลองจริงๆนั้นควรจะเริ่มจากการใช้ตัวโปรแกรมไมโครวิชั่นโดยเริ่มจากการสร้างโปรแกรมหลักขึ้นมาเพื่อเรียกใช้โมดูลที่ได้กล่าวไว้ในข้างต้น

ในส่วนนี้ทางผู้จัดทำได้สร้าง Main โปรแกรมขึ้นมาโดยจะแบ่งเป็นสองส่วนคือ Main ของมาสเตอร์และMainของสเลฟและAddโมดูลต่างๆที่ต้องการเข้ามา

โปรแกรมเมนของตัวมาสเตอร์

```
Name        MainPro

Extrn       Code(IniLcd,WrCharLcd,LcdLShf,LcdRShf,LcdHome,SetAddrLcd)
Extrn       Code(LcdClr,W1LINELCD,W2LINELCD,StartKeypad)
Extrn       Code(MainShow,Pcf8574Wr,Pcf8574Rd,Show1Digit)
Extrn       Code(Delay_1s,MainAdTDa,LOOPRT)
Public      ChEND
Prog        Segment  Code

LcdEn       BIT     P3.4
LcdRs       BIT     P3.5
CheckEnd    EQU     035H
LcdAddr     EQU     030H
LcdData     EQU     031H
KeyData     EQU     032H
I2C_Addr    EQU     036H
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
IOData          EQU     038H
InData          EQU     03AH
SegData         QU      03BH

;Main ProGram
                JMP     Start
                RSEG    Prog
Start:          MOV     P2,#00000000B
                CLR     LcdEn
                CLR     LcdRs
                CALL    IniLcd
Main:           MOV     LcdAddr,#000H
                ACALL   SetAddrLcd
                MOV     DPTR,#Letter1
                ACALL   W1LINELCD
                MOV     LcdAddr,#040H
                CALL    SetAddrLcd
                MOV     DPTR,#Letter2
                ACALL   W2LINELCD
                MOV     CheckEnd,#00
                ACALL   StartKeypad
                MOV     A,KeyData       ;Select Module
                CJNE    A,#01H,Module2
                ACALL   LcdClr
                MOV     LcdAddr,#000H
                ACALL   SetAddrLcd
                MOV     DPTR,#Letter3
                ACALL   W1LINELCD
                ACALL   MainShow        ; Call ModuleSegment
                JMP     Main
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Module2:    CJNE    A,#02H,Module3
            ACALL   LcdClr
            MOV     LcdAddr,#000H
            ACALL   SetAddrLcd
            MOV     DPTR,#Letter4
            ACALL   W1LINELCD
            ACALL   Show8574I2C;Call ModuleI2C(Pcf8574)
            JMP     Main

Module3:    CJNE    A,#03H,Module4
            ACALL   LcdClr
            MOV     LcdAddr,#000H
            ACALL   SetAddrLcd
            MOV     DPTR,#Letter5
            ACALL   W1LINELCD
            ACALL   MainAdTDa ;Call ModuleAtoD_DtoA
            JMP     Main
Module4:    CJNE    A,#04H,Module5
            ACALL   LcdClr
            MOV     LcdAddr,#000H
            ACALL   SetAddrLcd
            MOV     DPTR,#Letter6
            ACALL   W1LINELCD
            ACALL   LOOPRT
            JMP     Main
Module5:    CJNE    A,#05H,Module6
            ACALL   LcdClr
            MOV     LcdAddr,#000H
            ACALL   SetAddrLcd
            MOV     DPTR,#Letter7
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                ACALL   W1LINELCD
                ACALL   Delay_1s
                ACALL   Delay_1s
                JMP     Main
Back:           AJMP    Main
ChEND:          ACALL   StartKeypad
                MOV     A,KeyData
                CJNE    A,#10H,Ch1
                MOV     CheckEnd,#01 ;End Module
                JMP     Turn
Ch1:            MOV     CheckEnd,#00 ;Module Active
Turn:           RET
  ;Call Pcf8574 to Show
Show8574I2C:    ACALL   StartKeyPad
                MOV     A,KeyData
                CJNE    A,#00H,I2CShow
                ACALL   OutSeg
                ACALL   ChEND
                MOV     R2,CheckEnd
                CJNE    R2,#01,Show8574I2C
                JMP     Main

I2CShow:        ANL     A,#00001111B
                MOV     IOData,A
                ACALL   Pcf8574Wr
                ORL     A,#11110000B
                MOV     IOData,A
                ACALL   Pcf8574Wr
                ACALL   OutSeg
                AJMP    Show8574I2C
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
OutSeg:     MOV     R3,#04H
            ACALL   Pcf8574Rd
            MOV     A,INData
            ANL     A,#11110000B
Rotate:     RR      A
            DJNZ    R3,Rotate
            MOV     SegData,A
            ACALL   Show1Digit
            RET

Letter1:    DB      '  Select Module  '
Letter2:    DB      '  Press 1 TO 5   '
Letter3:    DB      ' ModuleShowSeg Run '
Letter4:    DB      ' ModulePCF8574 Run '
Letter5:    DB      ' ModulePCF8591 Run '
Letter6:    DB      ' ModuleRealTime Run'
Letter7:    DB      '     Slave Run      '
            End
```

โปรแกรมเมนของตัวสลาฟ

```
            NAME  MainSlave
            EXTRN  Code(MainTemp,I2CRecive)
            ProgMSlave  Segment  Code
            RSEG   ProgMSlave
            JMP    Start
Start:      MOV          P0,#00000000B
            ACALL        I2CRecive
            ACALL        MainTemp
            END
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อทำการ Built Project ไมโครวิชั่นก็จะสร้าง Hex File และนำไป Flash ลงในไมโครคอนโทรลเลอร์เบอร์ AT89S8252(Atmel) หรือเบอร์อื่นๆซึ่งในกรณีที่ใช้เบอร์อื่นจะต้องกำหนดเบอร์ของคอนโทรลเลอร์ในไมโครวิชั่นให้ถูกต้องด้วย

เมื่อประกอบวงจรทั้งหมดดังรูปวงจร และทำการรันโปรแกรม จะปรากฏข้อความบน LCD ของไมโครคอนโทรลเลอร์ Select Module Press 1 To 5 เมื่อกด

1.จะเป็นการนำโมดูล ScanKey และ Segment มาใช้ร่วมกันโดยค่าที่กดจาก Keypad จะไปแสดงยัง 7-Segment

2.จะเป็นการนำโมดูล I^2C,Segment,Scankey ร่วมกันเพื่อติดต่อกับ PCF8574 ในการขยายพอร์ทอินพุทเอาพุทบนบัส I^2C โดยค่าจาก Keypad จะแสดงยัง LED ของพอร์ทPCF8574 ฝั่ ทล่าง ส่วนดิฟสวิตจะต่ออยู่กับ 2 บิทบน เพื่อรับค่าเข้ามาแสดงยัง 7-Segment

3.จะนำโมดูล I^2C, AtoD-DtoA,Lcd ใช้ร่วมกันโดยค่าAnalog in และDigital Out ที่อ่านได้แสดงบน LCD ของตัวมาสเตอร์

4. จะนำโมดูล I2C, RealTime,ScanKey,Lcd ใช้ร่วมกันโดยค่าวันที่และเวลาจะแสดงบน LCD ของตัวมาสเตอร์ และสามารถกำหนดค่าวันเวลาใหม่ได้ด้วย Keypad

5. จะเป็นการสั่งงานจากตัวมาสเตอร์เพื่อไปสั่งงานให้คอนโทรลเลอร์สลาฟทำงานโดย ในส่วนของมาสเตอร์ใช้โมดูล I^2C, ScanKey ส่วนตัว สเลฟใช้โมดูล Serial,Temperature เพื่อแสดงค่าอุณหภูมิที่ LCD ของ สเลฟ และคอมพิวเตอร์ นอกจากนี้ยังมีการประยุกต์ใช้นำค่าอุณหภูมิที่อ่านได้มาเปรียบเทียบเพื่อใช้ในการตัดต่อ Relay ควบคุมซึ่งสามารถนำไปประยุกต์ใช้ในงานควบคุมอื่นๆได้ ทุกโปรแกรมสามารถสั่งหยุดการทำงานได้จาก Keypad

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุปหลักการทำงานและการวิจารณ์

5.1 สรุปหลักการทำงานการสร้างโปรแกรมควบคุมโมดูลมาตรฐานสำหรับ MCS – 51

ในการใช้งานเราจะสร้างโมดูลมาตรฐานต่างๆ ขึ้นมา โดยใช้ลักษณะการเขียนโปแกรมแบบ Linker/Linking Location และจะใช้ความสามารถของตัว Software Microvision เข้ามาช่วยด้วย โดยเมื่อเราต้องการจะใช้โมดูลใดเราก็จะ Add เข้าไปใน Project ที่เราสร้างไว้ เช่นจากตัวอย่างโปรเจ็กของเราต้องการ ใช้ Module Scankeypad ,Module Delay และ Module ShowSegment เราก็จะ Add Module ทั้ง3 Module นี้เข้าไปในโปรเจ็คและทำการ Build Project เพื่อสร้าง Hex File.Hex ออกมา และนำ Hex File ตัวนี้ไป Flash บน Eprom เพื่อนำไปใช้งาน และจากบอร์ดการทดลองที่สร้างขึ้นจะเน้นการสื่อสารระบบบัส I^2C เป็นส่วนมาก

โดยเมื่อทำการต่ออุปกรณ์อินเตอร์เฟสบนฐานบัส I^2C เรียบร้อยแล้ว การที่เราจะรับส่งข้อมูลหรือควบคุมนั้นจะต้องมีอุปกรณ์ที่เรียกว่าอุปกรณ์มาสเตอร์เป็นตัวกำหนดสภาวะของบัสเริ่มต้นและสร้างสัญญาณนาฬิกาเพื่อตรวจสอบและควบคุมข้อมูลในการส่ง สภาวะของบัสเริ่มต้นนั้นเป็นสภาวะที่อุปกรณ์สเลฟตรวจสอบการรับส่งข้อมูลว่าบัสนั้นมีสภาวะว่างหรือไม่ เพื่อเป็นการป้องกันการชนกันของข้อมูลที่อาจจะเกิดขึ้นบนบัส I^2C จากนั้นตัวมาสเตอร์จะสร้างบิต Start และทำการส่งค่าของแอดเดรสที่ต้องการจะติดต่อด้วยตามด้วยบิต อ่านหรือเขียน เมื่ออุปกรณ์ปลายทางที่ถูกติดต่อด้วยจะทำการส่งบิตตอบรับกลับมายังตัวมาสเตอร์ จากนั้นตัวมาสเตอร์จะส่งข้อมูลหรือรับข้อมูลไปเรื่อยๆ จนกว่าตัวมาสเตอร์สร้างบิต Stop ซึ่งเป็นการสิ้นสุดการรับส่งข้อมูลในหนึ่งๆ ชุด และตัวมาสเตอร์จะสร้างสภาวะบัสว่างขึ้นมาอีก ซึ่งข้อมูลแต่ละชุดจะมีบิตตอบรับจากตัวสเลฟตามมาด้วย

5.2 วิจารณ์

สาเหตุการ Link ระหว่างโมดูลต่างๆที่ทำให้เกิดการผิดพลาดนั้น เกิดจากการกำหนด 7-Segment ไม่ถูกต้อง หรือมักจะเกิดจากการเขียนโครงสร้างของโปรแกรมผิดพลาด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.3 ข้อดีและข้อเสียของโปรแกรมแบบ Module

ข้อดี

1.ง่ายต่อการตรวจสอบเพราะทุกส่วนแบ่งแยกเป็นอิสระต่อกัน

2.ช่วยประหยัดเนื้อที่หน่วยความจำในกรณีที่มี Sub Program เดียวกันที่ต้องการใช้ใน Project เดียวกัน ก็สามารถดึงมาใช้ร่วมกันได้เลย

ข้อเสีย

1.ในช่วงเริ่มต้นการเขียนโปรแกรม อาจมีความสับสนในโครงสร้างการเขียนโปรแกรมแบบ Link ได้

2.ค่าตัวแปรต่างๆ ที่ใช้งานร่วมกันจะต้องประกาศให้ทราบในทุกโมดูล

3.ปัญหาที่เกิดขึ้นจากการเขียนโปรแกรมติดต่อระหว่างไมโครคอนโทรลเลอร์ ตัวมาสเตอร์และสเลฟ ซึ่งใช้การติดต่อสื่อสารกับแบบบัส I^2C ซึ่งเราจะต้องกำหนดให้ตรงตามมาตรฐานของบัส I^2C ซึ่งบางครั้งในการติดต่อสื่อสารนั้นยังมีความผิดพลาดอยู่บาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

ธีรวัฒน์ ประกอบผล, **"การประยุกต์ใช้งานไมโครคอนโทรลเลอร์"** สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) ,2541

วรพจน์ กรแก้ววัฒนกุล,ชัยวัฒน์ ลิ้มพรจิตรวิไล, **" เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบเฟลช "** ,อินโนเวตีฟ เอ็กเพอริเมนต์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้