I^2C บัสบนระบบควบคุมตัวโปรเซสเซอร์

A I^2C BUS –BASED ON EMBEDDED – PROCESSOR SYSTEM

นายนิวัฒน์ ลำพึงกาล
นายรัฐพล มณฑา

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมการวัดคุม
ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

A I^2C BUS –BASED ON EMBEDDED – PROCESSOR SYSTEM

NIWAT LAMPUENGKAL

RATTHAPHOL MONTHA

A THESIS SUBMITTED IN PATIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMENT OF INSTURMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2001

หัวข้อปริญญานิพนธ์ I^2C บัสบนระบบควบคุมตัวโปรเซสเซอร์

A I^2C BUS –BASED ON EMBEDDED – PROCESSOR SYSTEM

นักศึกษาผู้จัดทำ นายนิวัฒน์ ลำพึงกาล **รหัสประจำตัว** 42015392

นายรัฐพล มณฑา **รหัสประจำตัว** 42015397

ปริญญา วิศวกรรมศาสตรบัณฑิต

สาขาวิชา วิศวกรรมการวัดคุม

ปีการศึกษา 2544

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รศ.พิพัฒน์ เลาหสงคราม	

วัน/เดือน/ปี ที่สอบ วันที่ 27 เดือน มีนาคม พ.ศ. 2545

สถานที่สอบ ณ. ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

ภาควิชารับรองแล้ว

(ผศ.ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์	I^2C บัสบนระบบควบคุมตัวโปรเซสเซอร์
	A I^2C BUS –BASED ON EMBEDDED – PROCESSOR SYSTEM
นักศึกษาผู้จัดทำ	นายนิวัฒน์ ลำพึงกาล
	นายรัฐพล มณฑา
อาจารย์ที่ปรึกษา	รศ.พิพัฒน์ เลาหสงคราม
ปีการศึกษา	2544

บทคัดย่อ

ปริญญานิพนธ์นี้เสนอแนวทางการศึกษาการใช้ I^2C Bus สำหรับ MCS-51 ซิงเกิลชิฟ การออกแบบใช้ I^2C Bus เสมือนในตัวมาสเตอร์กับตัวสเลฟต่างๆ ของระบบควบคุมโปรเซสเซอร์ร่วมกับการใช้ I^2C Bus จริงในระบบ โดยจะใช้งานทั้งสองระบบได้ดีเท่ากัน

Thesis Title	A I^2C Bus – based on Embedded – processor System
Authors	Niwat Lampuengkal
	Ratthaphol Montha
Thesis Advisor	Assoc.Prof.Phiphat Laohasongkram
Year	2001

ABSTRACT

This thesis is presented the studying on the I^2C Bus for the MCS-51 single chip. The virtual I^2C Bus is design by using the microcontroller as the master and many specific slaves in the embedded-processor system included the real I^2C Bus is used in system .These two I^2C Buses is going well in this thesis.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำปรึกษาแนะนำและเสนอแนะแนวทางการดำเนินงานเป็นอย่างดี ข้าพเจ้าผู้จัดทำขอมอบความดีให้แก่บุคคลผู้ให้ความอนุเคราะห์ ตลอดจนแนะนำในด้านต่างๆ ต่อผู้จัดทำดังนี้

รศ.พิพัฒน์ เลาหสงคราม อาจารย์ที่ปรึกษา และให้คำแนะนำในการค้นคว้า มาโดยตลอด และ คณะอาจารย์ในภาควิชาวิศวกรรมการวัดคุม ที่ให้คำแนะนำและข้อมูลเพิ่มเติม

ตลอดจนพี่ๆ เพื่อนๆ ผู้ให้คำแนะนำและช่วยเหลือในด้านต่างๆและขาดเสียมิได้คือบุพการี ผู้ให้ความช่วยเหลือในด้านการเงินและกำลังใจด้วยดีตลอดมา

คณะผู้จัดทำ

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญรูป VI

สารบัญตาราง VIII

บทที่ 1 บทนำ 1

บทที่ 2 การศึกษาทฤษฎีต่างๆ ที่เกี่ยวข้องกับปริญญานิพนธ์ 4

2.1 ความรู้เบื้องต้นของระบบบัส I^2C 4

2.2 คุณสมบัติทั่วไปของบัส I^2C 4

2.3 หลักการของบัส 5

2.3.1 ข้อกำหนดสำคัญของการติดต่อบนบัส 5

2.3.2 สภาวะที่เกิดขึ้นบนบัส 5

2.4 การทำงานบนบัส I^2C 9

2.4.1 การอ้างถึงแบบ 7 บิต 10

2.4.2 การอ้างถึง แบบ 10 บิต 11

บทที่ 3 ไมโครคอนโทรลเลอร์ MCS-51 และอุปกรณ์ร่วมของระบบฐานบัส I^2C 12

3.1 ไมโครคอนโทรลเลอร์ MCS-51 12

3.2 โมดูลแสดงผลแบบผลึกเหลว (LCD) 19

3.3 ไอซีขยายพอร์ตเบอร์ PCF 8574A 29

3.4 ไอซี ADC/DAC เบอร์ PCF 8591 33

3.5 ไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก (RTC) เบอร์ DS 1307 41

สารบัญ(ต่อ)

หน้า

บทที่ 4 การทดลองและผลการทดลอง 49

การทดลองที่ 1

การแสดงผลที่โมดูลแสดงผลแบบผลึกเหลว(LCD)โดยผ่านบัส I^2C 49

การทดลองที่ 2

การขยายพอร์ตอินพุตเอาต์พุตด้วยไอซีขยายพอร์ตเบอร์ PCF 8574A บนระบบบัส I^2C 54

การทดลองที่ 3

การขยายพอร์ตอินพุตเอาต์พุตด้วยไอซีไมโครคอนโทรลเลอร์ MCS-51 บนระบบบัส I^2C 56

การทดลองที่ 4

การใช้งานไอซี ADC/DAC บนระบบบัส I^2C เบอร์ PCF 8591 58

การทดลองที่ 5

การใช้งานไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก (RTC) เบอร์ DS 1307 61

การทดลองที่ 6

การใช้งานระบบฐานบัส I^2C ของตัวควบคุมไมโครคอนโทรลเลอร์ 65

บทที่ 5 สรุปหลักการทำงานและการวิจารณ์ 68

บรรณานุกรม 69

ภาคผนวก

สารบัญรูป

รูปที่ หน้า

1.1 แสดงขอบเขตทางด้านฮาร์ดแวร์ของปริญญานิพนธ์ 1

2.1 แสดงโครงสร้างของวงจรที่ต่อเข้ากับบัส I^2C 4

2.2 แสดงสภาวะช่วงที่บัส I^2C ว่างอยู่ 5

2.3 แสดงสภาวะจุดเริ่มต้น START 6

2.4 แสดงสภาวะที่ข้อมูลที่ถูกส่งไม่มีการเปลี่ยนแปลงกับช่วงที่เปลี่ยนแปลง 6

2.5 แสดงสภาวะการรับรู้ที่เกิดจากตัวมาสเตอร์กับอุปกรณ์สเลฟ 7

2.6 แสดงถึงสภาวะของการรับข้อมูลเกิดการผิดพลาด 8

2.7 แสดงสภาวะที่ข้อมูลถูกส่งครบแล้วจึงมีการสร้างบิตหยุด STOP 8

2.8 แสดงการติดต่อของมาสเตอร์ในลักษณะการเขียนการอ่านหรือทั้งอ่านเขียนกับอุปกรณ์สเลฟ. 9

2.9 แสดงถึงโครงสร้างของแอดเดรสอุปกรณ์สเลฟที่ตัวมาสเตอร์ต้องระบุเพื่อการถ่ายทอดข้อมูล 10

2.10 รูปแบบข้อมูลในการอ้างถึงแบบ 7 บิต 11

2.11 รูปแบบข้อมูลในการอ้างถึงแบบ 10 บิต 11

3.1 แสดงโครงสร้างภายในชิปไมโครคอนโทรลเลอร์ 13

3.2 แสดงไดอะแกรมการทำงานของโมดูล LCD แบบอักขระ 20

3.3 แสดงรูปร่างและการจัดขาโมดูล LCD แบบอักขระ 22

3.4 แสดงโฟลวชาร์ตและตัวอย่างโปรแกรมย่อยการอินนิเชียลโมดูล LCD 27

3.5 แสดงโฟลวชาร์ตและตัวอย่างโปรแกรมย่อยส่งพัลส์เอ็นเอเบิลให้แก่โมดูล LCD 28

3.6 แสดงโครงสร้างภายในไอซีขยายพอร์ตเบอร์ PCF 8574A 29

3.7 ไดอะแกรมแสดงการทำงานของวงจร ADC แบบซักเซสซีฟแอปพร็อกซิเมชั่น. 33

3.8 การจัดขาของไอซี ADC/DAC ขนาด 8 บิต ผ่านบัส I^2C เบอร์ PCF8591 36

3.9 รายละเอียดข้อมูลควบคุมที่เขียนลงในรีจิสเตอร์ควบคุมภายในไอซี PCF8591 38

3.10 การจัดขาของไอซี DS 1307 ไอซีสร้างฐานเวลาจริง (RTC) 41

3.11 โครงสร้างภายในของไอซีรีลไทม์คล็อกเบอร์ DS1307 43

สารบัญรูป (ต่อ)

รูปที่ หน้า

3.12 การจัดหน่วยความจำแรมภายใน DS1307 และรายละเอียดของรีจิสเตอร์เก็บค่าของเวลาและรีจิสเตอร์ควบคุมของ DS1307 44

3.13 รูปแบบของข้อมูลสำหรับติดต่อกับ DS 1307 ในโหมดการเขียนข้อมูล 45

3.14 รูปแบบของข้อมูลสำหรับติดต่อกับ DS 1307 ในโหมดการอ่านข้อมูล 46

4.1 แสดงวงจรการทดลองที่ 1 50

4.2 แสดงโฟลวชาร์ตโปรแกรมย่อยสถานะเริ่มต้น สถานะหยุด และโปรแกรมย่อยกำเนิดสัญญาณนาฬิกาบัส I^2C 51

4.3 แสดงโฟลวชาร์ตโปรแกรมย่อยการติดต่ออุปกรณ์สเลฟ และการเขียนข้อมูลไปยังอุปกรณ์สเลฟ 52

4.4 แสดงโฟลวชาร์ตโปรแกรมย่อยการอ่านข้อมูลจากอุปกรณ์สเลฟ 53

4.5 แสดงวงจรการทดลองที่ 2 55

4.6 แสดงวงจรการทดลองที่ 3 57

4.7 แสดงวงจรการทดลองที่ 4 59

4.8 แสดงโฟลวชาร์ตโปรแกรมย่อยการอ่านข้อมูลจาก PCF8591และการเขียนให้แก่ PCF8591 . 60

4.9 แสดงวงจรการทดลองที่ 5 62

4.10 แสดงโฟลวชาร์ตโปรแกรมย่อยการอ่านค่า RTC 63

4.11 แสดงโฟลวชาร์ตโปรแกรมย่อยการเขียนค่า RTC 64

4.12 แสดงวงจรการทดลองที่ 6 66

สารบัญตาราง

ตารางที่ หน้า

3.1 Serial Port Control Register..... 15

3.2 แสดงโหมดต่างๆ ของการรับส่งแบบอนุกรม..... 16

3.3 แสดงค่าในรีจิสเตอร์TMOD ต่างๆ สำหรับ Timer 0 17

3.4 แสดงค่าในรีจิสเตอร์TMOD ต่างๆ สำหรับ Counter 0 17

3.5 แสดงค่าในรีจิสเตอร์TMOD ต่างๆ สำหรับ Timer 1 18

3.6 แสดงค่าในรีจิสเตอร์TMOD ต่างๆ สำหรับ Counter1 18

3.7 รีจิสเตอร์ใช้เฉพาะ PSW (Program Status Word) เข้าถึงข้อมูลได้ในระดับบิต 19

3.8 แสดงความสัมพันธ์ในการทำงานของขา RS,R/W และ E ของโมดูล LCD แบบ อักขระ..... 21

3.9 แสดงการควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผล..... 24

3.10 รายละเอียดหน้าที่การทำงานของแต่ละขาของไอซี PCF8574/ PCF8574A 31

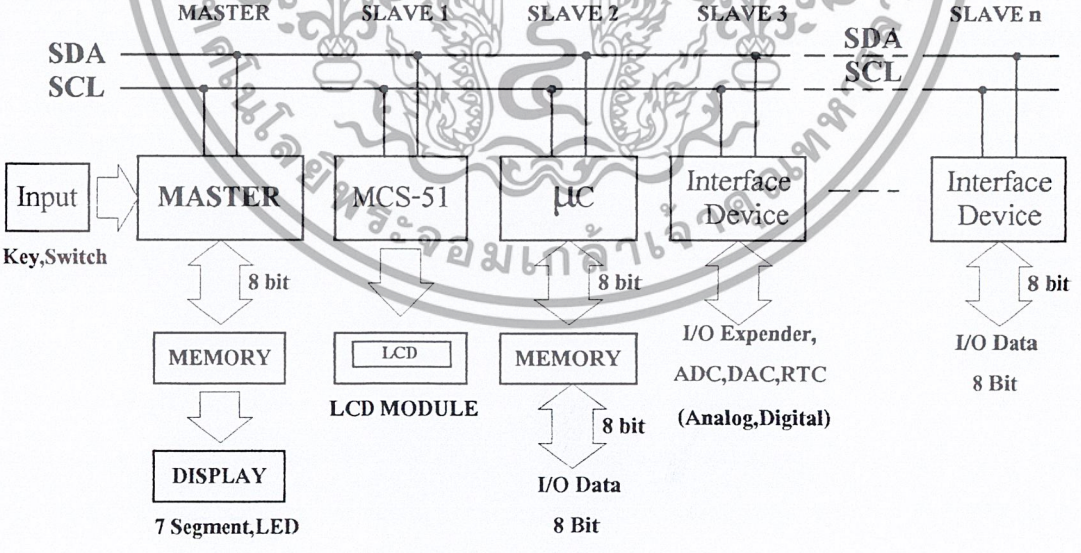
บทที่ 1
บทนำ

เนื่องจากปัจจุบันได้มีการใช้มาตรฐานการเชื่อมต่อ แบบบัส I^2C กันอย่างแพร่หลาย แต่ตัวอุปกรณ์ที่เป็นไมโครคอนโทรลเลอร์กลับยังไม่มีการเพิ่มฮาร์ดแวร์ที่มีโปรโตคอลแบบบัส I^2C ลงในตัวไมโครคอนโทรลเลอร์ ทำให้ไมโครคอนโทรลเลอร์ประเภทนี้จึงมีราคาสูงกว่าไมโครคอนโทรลเลอร์ทั่วไป เราจึงให้ความสนใจในการนำรูปแบบโปรโตคอล มาพัฒนาในรูปแบบซอฟแวร์เพื่อลดต้นทุนและรองรับความต้องการที่จะใช้มาตรฐานนี้ หรือสามารถจะใช้มาตรฐานการเชื่อมต่อแบบบัส I^2C นี้นำมาพัฒนาและใช้งานกับวงจรอื่นๆ ได้

1.1 วัตถุประสงค์

1. เพื่อศึกษามาตรฐานการเชื่อมต่อแบบบัส I^2C
2. เพื่อนำมาตรฐานการเชื่อมต่อแบบ I^2C มาประยุกต์และพัฒนาใช้ในการรับส่งข้อมูลบนบัส I^2C ระหว่างตัวมาสเตอร์กับอุปกรณ์สเลฟ

1.2 ขอบเขตของปริญญานิพนธ์



รูปที่ 1.1 แสดงขอบเขตทางด้านฮาร์ดแวร์ของโครงงาน

- **ขอบเขตทางด้านฮาร์ดแวร์**

1. สร้างบอร์ดตัวมาสเตอร์โดยใช้ไมโครคอนโทรลเลอร์เบอร์AT89C52
2. สร้างบอร์ดตัวสเลฟแสดงผลโดยใช้ไมโครคอนโทรลเลอร์เบอร์AT89C52 ขับ LCD
3. สร้างบอร์ดตัวสเลฟโดยใช้ไอซีขยายพอร์ตอินพุตเอาท์พุตเบอร์ PCF 8574A
4. สร้างบอร์ดตัวสเลฟโดยใช้ไอซี ADC/DAC เบอร์ PCF 8591
5. สร้างบอร์ดตัวสเลฟโดยใช้ไอซีฐานเวลาเบอร์ DS1307
6. สร้างบอร์ดตัวสเลฟขยายพอร์ตอินพุตเอาต์พุตโดยใช้ไมโครคอนโทรลเลอร์

- **ขอบเขตทางด้านซอฟแวร์**

จะเขียนโปรแกรมการติดต่อระหว่างตัวมาสเตอร์กับอุปกรณ์สเลฟแต่ละตัว แล้วทำการรวมอุปกรณ์ทุกตัวไว้บนบัสเดียวกัน และทดสอบโปรแกรมโดยรวม ซึ่งการเขียนโปรแกรมใช้ภาษาแอสแซมบลี้ของ MCS-51

1.3 ขั้นตอนและวิธีการดำเนินงาน

1. ศึกษาทฤษฎีการสื่อสารข้อมูลมาตรฐานแบบบัส I^2C
2. ศึกษาการทำงานของ MCS-51 และทดลองเขียนโปรแกรมแอสแซมบลี้ลงใน MCS-51
3. ศึกษาการทำงานของไอซีขยายพอร์ตเบอร์ PCF 8574A
4. ศึกษาการทำงานของโมดูลแสดงผลแบบผลึกเหลว (LCD MODULE)
5. ศึกษาการทำงานของไอซี ADC/DAC บนระบบบัส I^2C เบอร์ PCF 8591
6. ศึกษาการทำงานของไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อกเบอร์ DS 1307
7. สร้างไอซีขยายพอร์ตแบบบัส I^2C โดยใช้ไมโครคอนโทรลเลอร์เบอร์ AT 89C52
8. ทดลองการรับส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์กับไมโครคอนโทรลเลอร์รับส่งข้อมูลผ่านบัส I^2C
9. ทดลองส่งข้อมูลผ่านบัส I^2C ไปแสดงผลที่ LCD MODULE
10. ทดลองรับส่งข้อมูลระหว่างอุปกรณ์มาสเตอร์กับไอซีขยายพอร์ตเบอร์ PCF 8574A
11. ทดลองรับส่งข้อมูลระหว่างอุปกรณ์มาสเตอร์กับไอซี ADC/DAC บนระบบบัส I^2C เบอร์ PCF 8591
12. ทดลองรับส่งข้อมูลระหว่างอุปกรณ์มาสเตอร์ กับไอซีสร้างฐานเวลาจริงหรือรีลไทม์-คล็อกเบอร์ DS 1307
13. ทดลองรับส่งข้อมูลกับระหว่างอุปกรณ์มาสเตอร์กับอุปกรณ์สเลฟทั้งหมด
14. สรุปการทดลองและปัญหาการทำงานพร้อมทั้งข้อเสนอแนะ

1.4 ประโยชน์ที่คาดว่าจะได้รับ

เพื่อเป็นข้อมูลแก่ผู้ที่สนใจนำไปพัฒนาในการควบคุมกระบวนการ และต้องการใช้ประโยชน์ของโครงงานนี้ที่เกี่ยวกับการสื่อสารข้อมูลโดยมาตรฐานการเชื่อมต่อแบบบัส I^2C ซึ่งในปัจจุบันมีการนำโปรโตคอลนี้มาใช้อย่างแพร่หลาย

บทที่ 2
การศึกษาทฤษฎีต่างๆที่เกี่ยวข้องกับปริญญานิพนธ์

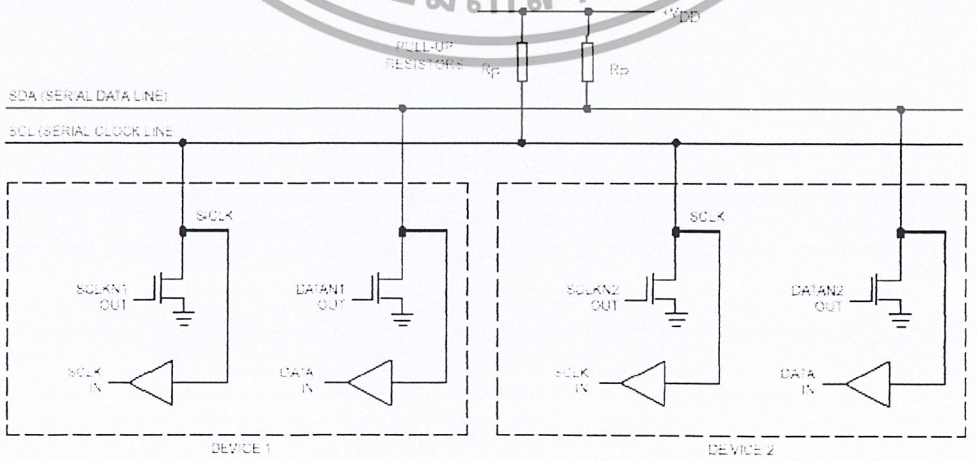
2.1 ความรู้เบื้องต้นของระบบบัส I^2C

I^2C ย่อมาจาก Inter - IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยบัส I^2C ได้รับการพัฒนาขึ้นโดยฟิลิปส์(Philips) ด้วยจุดมุ่งหมายหลัก คือต้องการให้ไอซีหรือโมดูลสามารถติดต่อ สั่งงาน และควบคุมภายใต้สายสัญญาณเพียงสองเส้นคือ สายข้อมูล อีกเส้นหนึ่งคือสายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานกันไปส่วนการกำหนดแอดเดรสสำหรับติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสข้อมูลร่วมกับการกำหนดสภาวะลอจิกที่ขาแอดเดรส

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

2.2 คุณสมบัติทั่วไปของบัส I^2C

สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (bi-directional line) ต้องมีการต่อตัวต้านทานพูลอัปกับแรงดัน +5 V ไว้ตลอดเวลา เพื่อให้สายสัญญาณมีสถานะลอจิกสูงในขณะที่ไม่มีการต่อใช้งาน ทั้งยังช่วยในการป้องกัน สัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาท์พุตของอุปกรณ์ที่ต่ออยู่บนบัส I^2C ต้องมีลักษณะ OPEN-DRAIN หรือ OPEN-COLLECTOR



รูปที่ 2.1 แสดงโครงสร้างของวงจรที่ต่อเข้ากับบัส I^2C

อัตราการถ่ายเทข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติ (Standard mode) และสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง (Fast mode) อุปกรณ์ที่ต่อร่วมอยู่บนบัส I^2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400 pF การเข้าถึงอุปกรณ์บนบัส I^2C ใช้ข้อมูลการเข้าถึง 2 ค่าคือ 7 บิต (7-bit Addressing) และ 10 บิต (10-bit Addressing)

2.3 หลักการของบัส

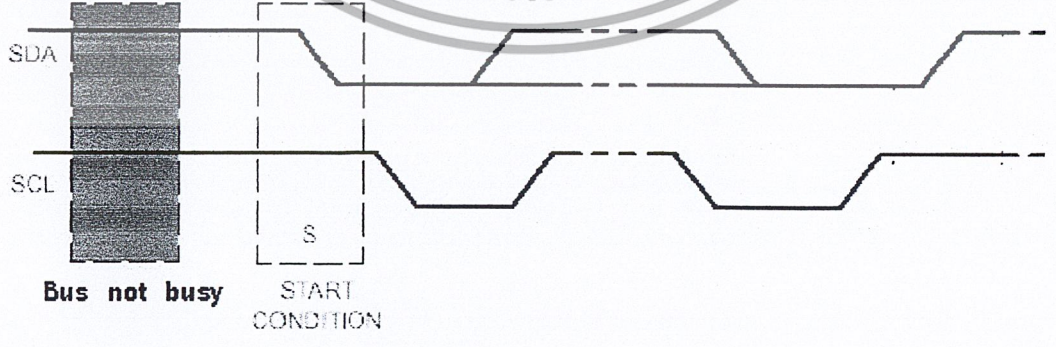
บัสประกอบด้วยสายสัญญาณ 2 เส้น คือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัส I^2C สามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส หรือ เรียกว่า โปรโตคอล (PROTOCOL) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดต่อกันอยู่ และอุปกรณ์ใดเป็นตัวส่งหรือตัวรับ

2.3.1 ข้อกำหนดสำคัญของการติดต่อบนบัส คือ

1. การถ่ายทอดข้อมูลที่จะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น
2. ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้อย่าให้เกิดการเปลี่ยนแปลงเด็ดขาด มิฉะนั้นสัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

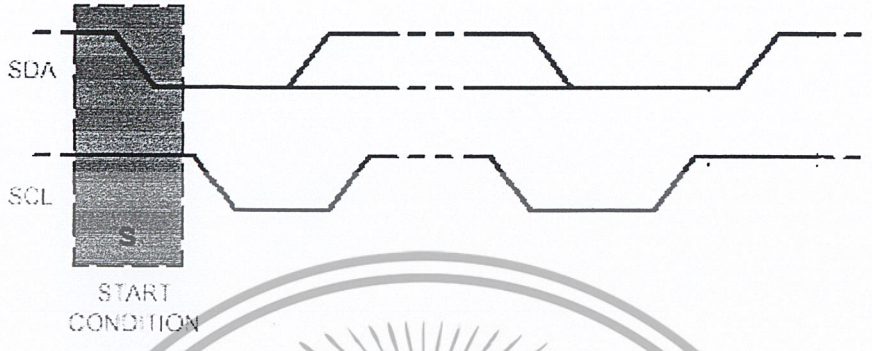
2.3.2 สภาวะที่เกิดขึ้นบนบัส

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



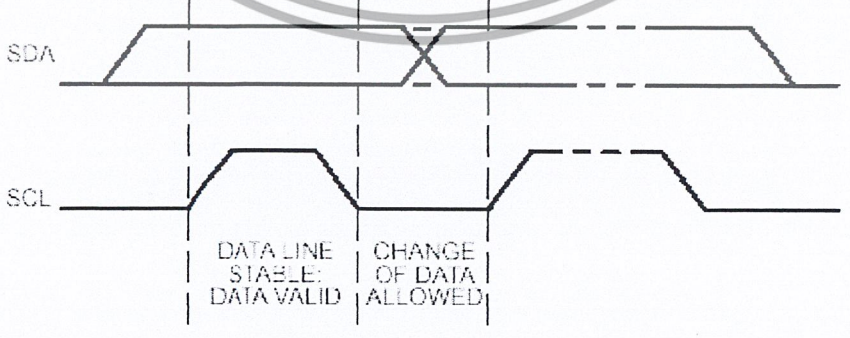
รูปที่ 2.2 แสดงสภาวะช่วงที่บัส I^2C ว่างอยู่

เริ่มต้นการถ่ายเทข้อมูล (start data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกสูงไปต่ำในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะเริ่มต้น (START)



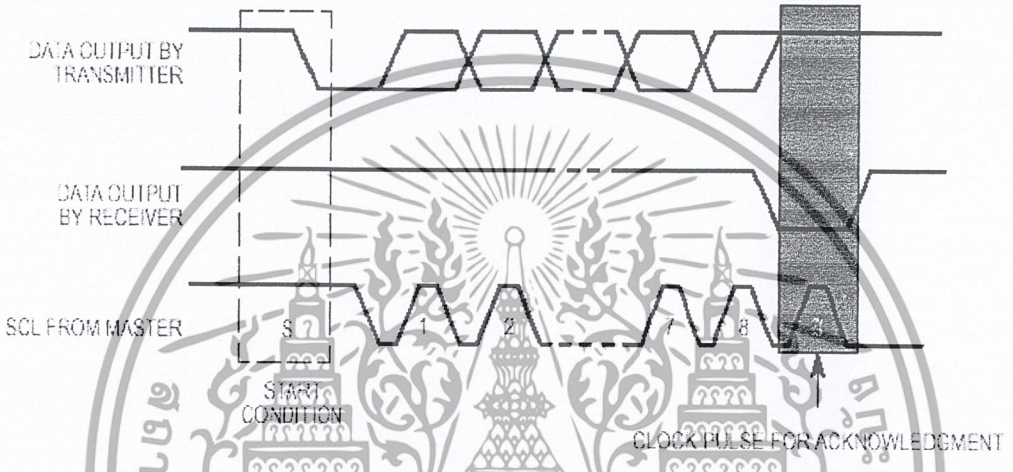
รูปที่ 2.3 แสดงสภาวะจุดเริ่มต้น START

ข้อมูลดำรงอยู่บนบัส (datavalid) สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะ ลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่า เป็น "0" หรือ "1" ข้อมูลที่อาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้มีการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูงหากเกิดการเปลี่ยนแปลงสถานะลอจิกที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุด หรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลการถ่ายทอดนั้นเกิดความผิดพลาดขึ้น



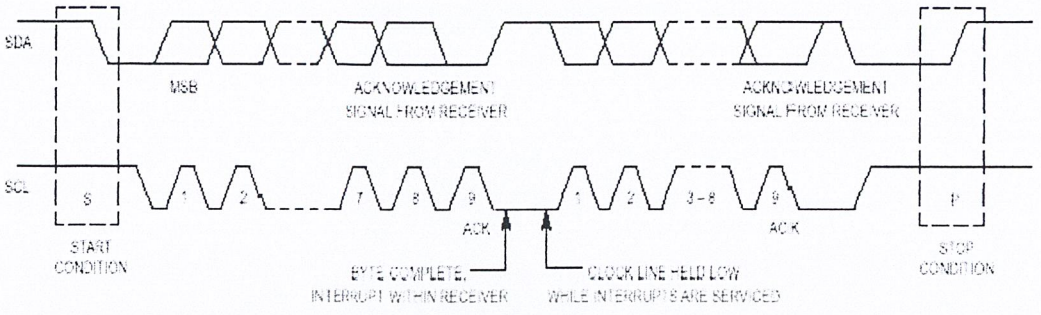
รูปที่ 2.4 แสดงสภาวะที่ข้อมูลที่ถูกส่งไม่มีการเปลี่ยนแปลงกับช่วงที่เปลี่ยนแปลง

รับรู้ข้อมูล (acknowledge) เกิดขึ้นหลังจากการถ่ายทอดข้อมูล จากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิต เรียกว่า บิตรับรู้ (acknowledge bit) มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์ จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อ หรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้ที่มีสถานะลอจิกต่ำ เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว



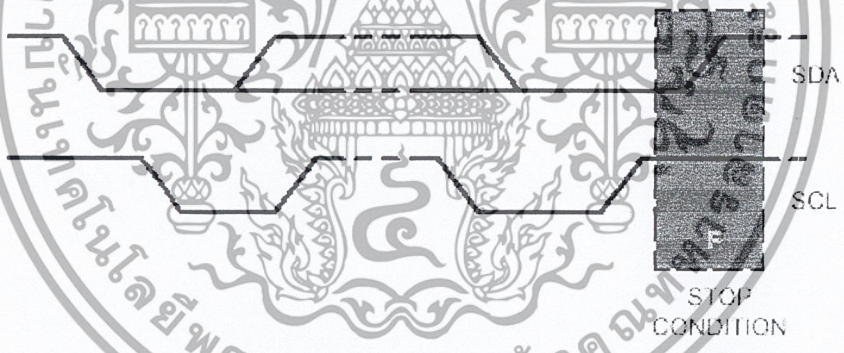
รูปที่ 2.5 แสดงสภาวะการรับรู้ที่เกิดจากตัวมาสเตอร์กับอุปกรณ์สเลฟ

แต่ว่าทุกไบต์ที่ผ่านเส้น SDA จะต้องมีขนาด 8 บิต จำนวนไบต์ที่ถูกส่งไปแต่ละครั้งจะไม่มีจำกัด แต่ในแต่ละไบต์จะต้องมีสัญญาณตอบรับ(Acknowledge) หนึ่งบิตเสมอ โดยที่ในหนึ่งไบต์ข้อมูลบิตแรกจะเป็นบิต MSB ของไบต์นั้นๆ ดูรูปที่ 2.5 ถ้าอุปกรณ์ตัวรับไม่สามารถจะรับขนาดไบต์ได้อย่างสมบูรณ์ มันจะต้องสร้างสัญญาณรับ ข้อมูลที่ผิดพลาดให้กับตัว MASTER รู้ด้วยการสร้างอินเตอร์รัพต์ภายในด้วยการกดสัญญาณบนเส้น SCL ให้มีสถานะต่ำ เพื่อให้การส่งข้อมูลเข้าสู่สภาวะการรอคอย ดังนั้นข้อมูลตัวต่อมาสามารถที่จะส่งต่อไปได้เมื่อตัวรับพร้อมที่จะรับไบต์ตัวต่อไป และเกิดสัญญาณนาฬิกาบนเส้น SCL ต่อไป



รูปที่ 2.6 แสดงถึงสภาวะของการรับข้อมูลเกิดการผิดพลาด

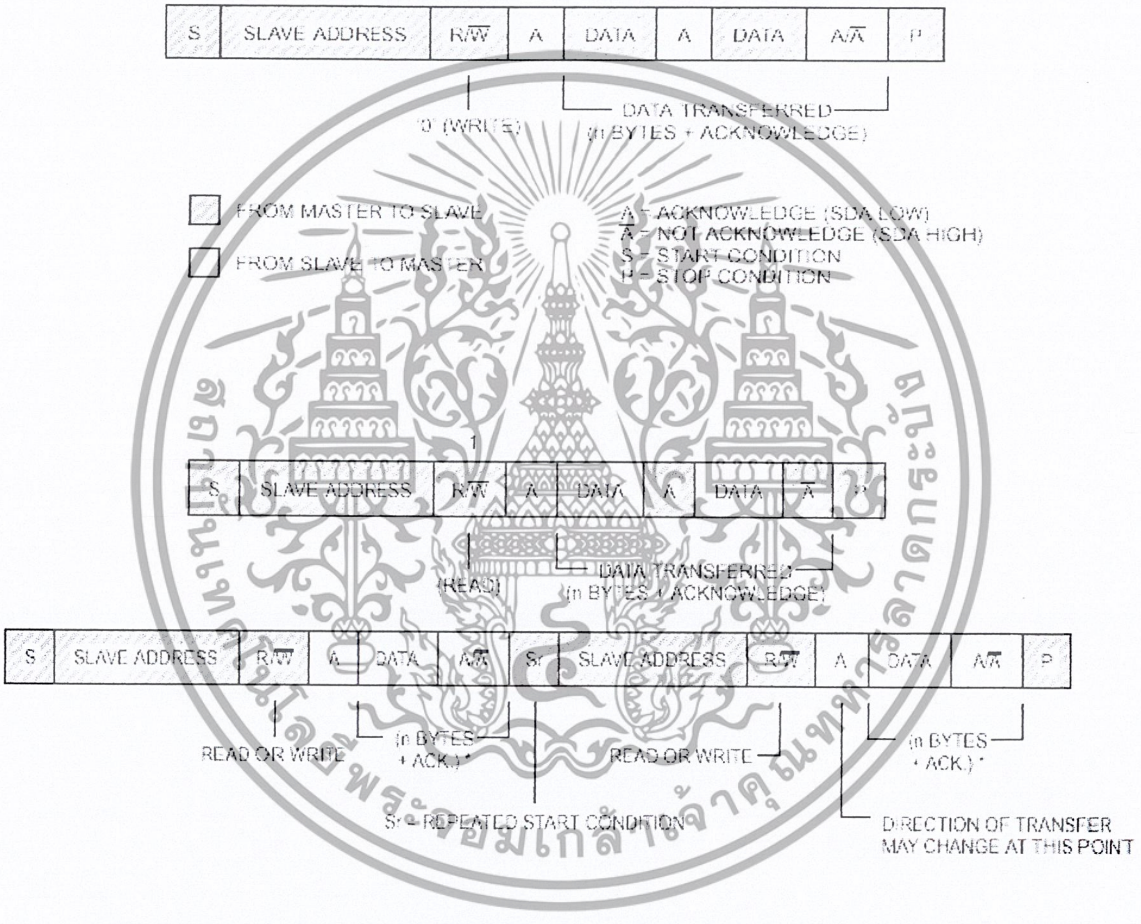
หยุดการถ่ายทอดข้อมูล (Stop Data Transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะหยุด (STOP)



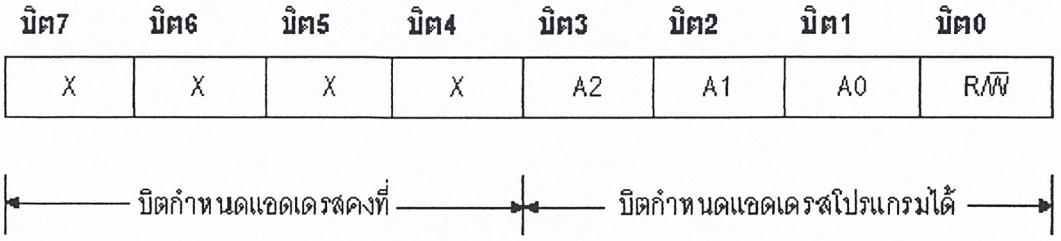
รูปที่ 2.7 แสดงสภาวะที่ข้อมูลถูกส่งครบแล้วจึงมีการสร้างบิตหยุด STOP

2.4 การทำงานบนบัส I^2C

ก่อนที่จะเริ่มต้นการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่างๆที่ต่ออยู่บนบัส ต้องมีการอ้างถึงอุปกรณ์เสียก่อน โดยการอ้างถึงอุปกรณ์บนบัส I^2C นั้นจะมีการอ้างถึงแบบ 7 บิตหรือ 10 บิต ในกรณีที่มีอุปกรณ์ที่ต่ออยู่บนบัสไม่มาก ใช้การอ้างถึงแบบ 7 บิตก็เพียงพอ แต่ถ้ามีอุปกรณ์ต่ออยู่บนบัสมากกว่า 127 แอดเดรส จำเป็นต้องใช้การอ้างถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้ว ก็จะเริ่มต้นการถ่ายทอดข้อมูลกันต่อไป



รูปที่ 2.8 แสดงการติดต่อของมาสเตอร์ในลักษณะการเขียน การอ่าน หรือทั้งอ่านเขียน กับอุปกรณ์สเลฟ



รูปที่ 2.9 แสดงถึงโครงสร้างของแอดเดรสอุปกรณ์สเลฟที่ตัวมาสเตอร์ต้องระบุเพื่อการถ่ายทอด ข้อมูล

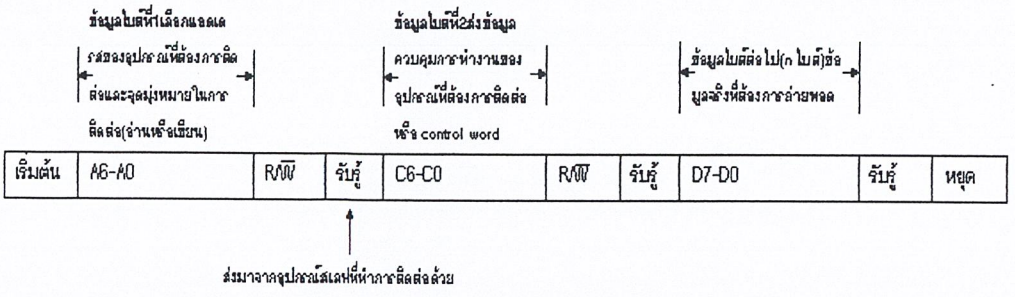
2.4.1 การอ้างถึงแบบ 7 บิต (7-bit addressing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อ โดยมีรูปแบบแสดงในรูปที่ 2.10 ใน 7 บิตบนรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อโดยแบ่งเป็น บิตกำหนดแอดเดรสคงที่ (fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิตเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I^2C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือการเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้นๆ หากบิต LSB เป็น "0" หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น "1" จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ

ข้อมูลไบต์ต่อมาคือ ข้อมูลควบคุม (control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลที่แตกต่างกันไป ยกตัวอย่าง ไอซีขยายพอร์ทมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุต บิตใดเป็นเอาต์พุต ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลในไบต์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง (data)

หลังจากได้มีการถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง

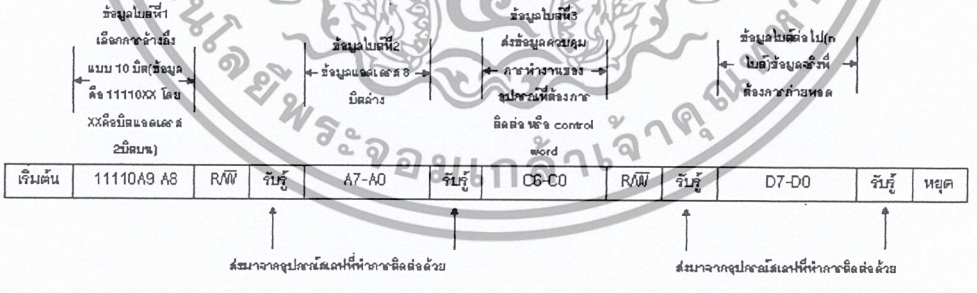


รูปที่ **2.10** รูปแบบข้อมูลในการอ้างถึงแบบ 7 บิต

2.4.2 การอ้างถึง แบบ 10 บิต

จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิตแอดเดรสของอุปกรณ์สเลฟตัวที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นจึงเป็นข้อมูลควบคุม ข้อมูลหลังจากนั้นจึงเป็นข้อมูลจริงที่ใช้ในการติดต่อ

เช่นเดียวกันกับการอ้างแบบ 7 บิต หลังการถ่ายทอดข้อมูลครบทุกไบต์ ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้



รูปที่ 2.11 รูปแบบข้อมูลในการอ้างถึงแบบ 10 บิต

บทที่ 3
ไมโครคอนโทรลเลอร์ และอุปกรณ์ร่วมของระบบฐานบัส I^2C

3.1 ไมโครคอนโทรลเลอร์ MCS-51

MCS-51 เป็นไมโครคอนโทรลเลอร์แบบชิปเดี่ยวที่มีข้อดีเมื่อเทียบกับไมโครโปรเซสเซอร์ขนาด 8 บิตตระกูลอื่น ดังนี้

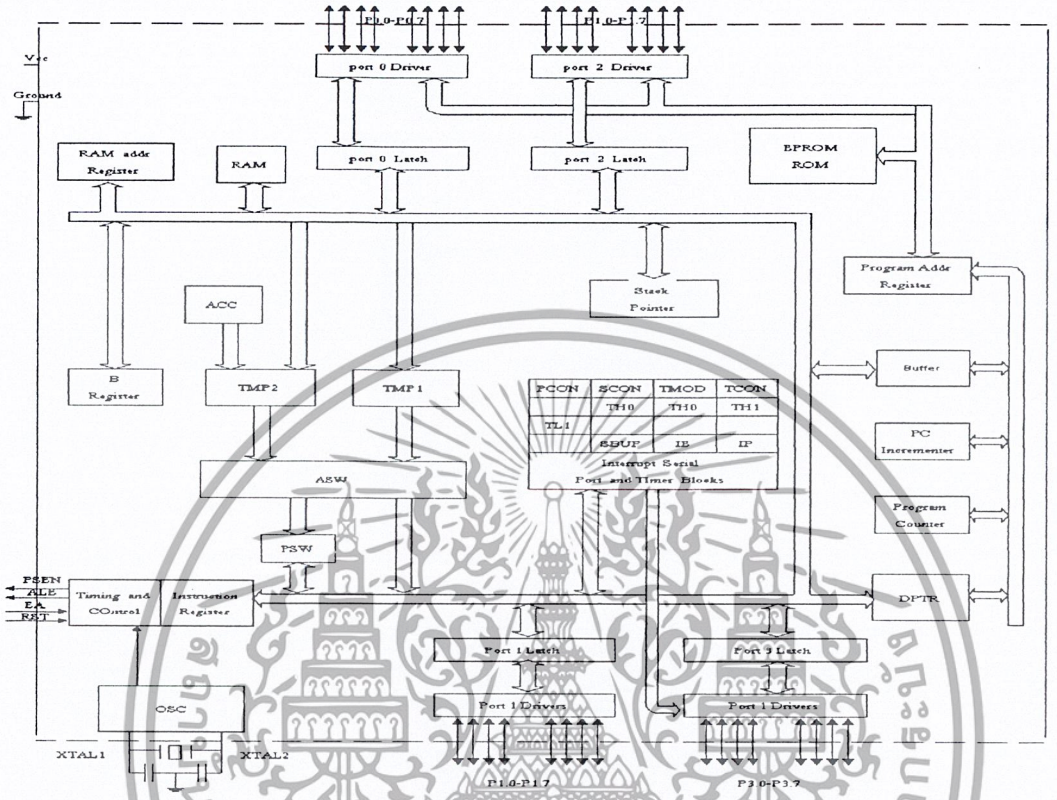
- มีหน่วยความจำสำหรับเก็บข้อมูลทั่วไป (RAM) บรรจุไว้ภายใน 128 – 256 ไบต์
- มีหน่วยความจำสำหรับเก็บโปรแกรมควบคุมการทำงานอยู่ภายในจำนวน4 กิโลไบต์
- มีวงจรตั้งเวลาวงจรนับขนาด 16 บิต 2 ตัว อยู่ภายใน
- มีวงจรรับส่งข้อมูลอนุกรม ได้ 2 ทิศทาง
- มีสัญญาณนาฬิกาภายในตัว
- มีพอร์ทที่สามารถรับหรือส่งข้อมูลได้ 2 ทิศทางจำนวน 4 พอร์ท ๆ ละ 8 บิต

นอกจากนี้ MCS – 51 ยังมีคุณสมบัติอื่น ๆ ที่น่าสนใจ คือ

- ต้องการแหล่งจ่ายไฟ 5 โวลท์เพียงชุดเดียว
- มีหน่วยความจำสำหรับเก็บ โปรแกรมควบคุมการทำงานอยู่ภายในชิป
- สามารถใช้หน่วยความจำ สำหรับโปรแกรม และข้อมูลที่อยู่ภายในชิปได้อย่างละ 64 กิโลไบต์
- มีคำสั่งคูณและหารเลขขนาด 8 บิตในตัวเอง
- จัดลำดับความสำคัญของสัญญาณอินเตอร์รัปต์ได้ 2 ระดับ
- รับและส่งข้อมูลแบบอนุกรมได้ในตัว โดยสามารถกำหนดอัตราเร็วในการรับและส่งข้อมูลได้ตั้งแต่ 300 – 375 กิโลบิตต่อวินาที
- สามารถประมวลผลแบบบูลีนเพื่อใช้ในงานควบคุมโดยเฉพาะ
- มีรีจิสเตอร์สำหรับใช้งานเป็นไทม์เมอร์หรือเคาน์เตอร์เพื่อนับจำนวนสัญญาณนาฬิกาภายในชิปหรือ นับการเปลี่ยนสถานะของสัญญาณภายนอกขนาด 16 บิต จำนวน 2 ตัว เพื่อใช้สำหรับนับจำนวนพัลส์ วัดความกว้างของพัลส์หรือใช้วัดช่วงเวลา
- หน่วยความจำสำหรับเก็บข้อมูลภายในบางส่วนสามารถเข้าถึงข้อมูลได้ทั้งระดับไบต์และระดับบิตเพื่อให้การออกแบบโปรแกรมและการควบคุมระบบงานทำได้ง่ายขึ้น

สถาปัตยกรรมของไมโครคอนโทรลเลอร์ 8051

โครงสร้างภายในชิปไมโครคอนโทรลเลอร์ 8051 ชิปเดี่ยวแสดงดังรูปที่ 3.1 ซึ่งอธิบายถึงส่วนย่อย ๆ ภายใน 8051



รูปที่ 3.1 แสดงโครงสร้างภายในชิปไมโครคอนโทรลเลอร์

ขาใช้งานต่าง ๆ ของไมโครคอนโทรลเลอร์มีดังนี้

- V_{CC} **ขา 40** เป็นขาที่ต้องป้อนไฟเลี้ยง +5 โวลท์เข้าไปเพื่อให้วงจรสามารถทำงานได้

- V_{SS} **ขา 20** เป็นขาที่ต้องต่อกับกราวด์ของแหล่งจ่ายไฟ

- **RST ขา 9** ขารีเซตนี้จะรีเซตการทำงานของ 8051 ถ้าป้อนสัญญาณที่มีสภาวะลอจิก 1 ที่ขานี้จะเป็นการรีเซตการทำงาน กลับไปเริ่มการทำงานจากคำสั่งที่อยู่ในหน่วยความตำแหน่ง 0000

- **ALE ขา 30** ใช้เป็นขาส่งสัญญาณออกไปภายนอก เพื่อควบคุมการแลตช์ค่าตำแหน่งไบต์ต่ำจากพอร์ท 0 ในระหว่างการติดต่อหน่วยความจำสำหรับเก็บโปรแกรมหรือข้อมูลภายนอก

- **PSEN ขา 29**ใช้ส่งสัญญาณเพื่ออ่านคำสั่ง จากโปรแกรมที่เก็บไว้ในหน่วยความจำภายนอกชิฟ

- **XTAL1 ขา 19** ใช้ต่อคริสตอลภายนอก โดยเป็นอินพุตเข้าสู่วงจรออสซิเลเตอร์

- **XTAL2 ขา 18** ใช้ต่อคริสตอลภายนอก โดยเป็นเอาท์พุตออกจากวงจรออสซิเลเตอร์

- PORT 0 เป็นพอร์ทขนาน8 บิตอยู่ที่ขา39 ถึง32 เริ่มจากบิต 0 ถึงบิต 7 ตามลำดับ พอร์ท 0 นี้ ใช้ได้ทั้งการรับส่งตำแหน่งและข้อมูลกับหน่วยความจำหรือใช้รับส่งข้อมูลก็ได้ นอกจากนี้ยังใช้งานได้หลายอย่างดังนี้

1. ใช้สำหรับส่งค่าตำแหน่งหน่วยความจำที่ต้องการติดต่อด้วย โดย 8 บิตล่างถูกส่งออกไปทางพอร์ท 0 และ 8 บิตบนถูกส่งออกทางพอร์ท 2
2. ใช้รับส่งข้อมูลกับ Data Memory หรือใช้รับข้อมูลจาก Program Memory
3. ใช้รับส่งข้อมูลออกจากพอร์ทโดยตรง

- PORT 1 เป็นพอร์ทขนานขนาด 8 บิต อยู่ที่ขา 1 ถึง 8 เริ่มจากบิต 0 ถึง บิต 7 ตามลำดับใช้ทำหน้าที่เป็นตัวรับส่งข้อมูลเท่านั้น ไม่สามารถส่งตำแหน่งได้

- PORT 2 เป็นพอร์ทขนานขนาด 8 บิต อยู่ที่ขา 21 ถึง 28 เริ่มจากบิต 0ถึงบิต 7 ตามลำดับใช้งานเพียง 2 ลักษณะ คือ

1. ใช้ส่งค่าตำแหน่งหน่วยความจำภายนอก ที่ต้องการติดต่อทำงานร่วมกับพอร์ท 0
2. ใช้เป็นพอร์ทรับส่งข้อมูลกับภายนอก

- PORT 3 เป็นพอร์ทขนานขนาด 8 บิตอยู่ที่ขา 10 ถึง 17 เริ่มจากบิต 0 ถึงบิต 7 ตามลำดับ นอกจากจะใช้วานเหมือนพอร์ทอื่น ๆ แล้วยังใช้งานอื่นโดยใช้คำสั่งควบคุม ดังนี้

P3.0 (RxD)	เป็นขาที่ใช้รับข้อมูลแบบอนุกรม
P3.1 (TxD)	เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม
P3.2 (INT0)	ใช้รับสัญญาณขัดจังหวะจากภายนอก
P3.3 (INT1)	ใช้รับสัญญาณขัดจังหวะจากภายใน
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 ในการรับส่งข้อมูลโดยขาทั้งสองจะอยู่ในพอร์ท 3 คือ P3.1 หรือขา 11 เป็น TxD และ P3.0 หรือขา 10 เป็น RxD พอร์ทอนุกรม MCS – 51 สามารถทำงานเป็นแบบ Full Duplex ได้ คือสามารถรับและส่งข้อมูลในเวลาเดียวกันได้ โดยมีการรับและส่งข้อมูลแบบบัฟเฟอร์สำหรับเก็บข้อมูลให้ใช้

รีจิสเตอร์ที่สำคัญในการรับส่งข้อมูลคือ SBUF และ SCON ซึ่งเป็นรีจิสเตอร์ที่อยู่ใน Special Function Register โดยรีจิสเตอร์ Serisl Port Buffer (SBUF) จะอยู่ในตำแหน่ง 99H ถ้าเขียนข้อมูลลงไปในตำแหน่งนี้จะเป็นการส่งข้อมูลออกทางพอร์ทอนุกรม และถ้าอ่านข้อมูลจาก

ตำแหน่งนี้จะเป็นการรับข้อมูลจากพอร์ทอนุกรม โดยใน SBUF จะประกอบด้วยบัฟเฟอร์ 2 ตัว สำหรับส่งและรับข้อมูล

สำหรับ Serial Port Control Register (SCON) ซึ่งอยู่ที่ตำแหน่ง 98 H จะเป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ รีจิสเตอร์นี้จะทำหน้าที่ควบคุม และบอกสถานะต่าง ๆ ของการรับส่งข้อมูลแบบอนุกรม

สำหรับความเร็วของการส่งข้อมูล (Baud Rate) สามารถหาได้จากการหาสัญญาณนาฬิกาที่ใช้กับ MCS – 51

ตารางที่ 3.1 Serial Port Control Register

บิต	ชื่อ	ตำแหน่ง	ความหมาย
SCON.7	SM0	9FH	บิตเลือกโหมดการทำงานบิต0
SCON.6	SM1	9EH	บิตเลือกโหมดการทำงานบิต1
SCON.5	SM2	9DH	บิตเลือกโหมดการทำงานบิต2
SCON.4	REN	9CH	บิตแฟลกกำหนดยอมให้มีการรับข้อมูล
SCON.3	TB8	9BH	ค่าของบิต 9 สำหรับส่งข้อมูลในโหมด 2 และ 3 สามารถเซต และเคลียร์ได้โดยซอฟแวร์
SCON.2	RB8	9AH	ค่าของบิต9 เมื่อรับข้อมูลเข้ามา
SCON.1	TI	99H	บิตแฟลกแสดงการอินเตอร์รัพท์ภายหลังการส่งข้อมูลออกไป โดยจะเซตเมื่อส่งข้อมูลออกไปหมดแล้ว และสามารถเคลียร์ได้โดยซอฟแวร์
SCON.0	RI	98H	แฟลกแสดงการอินเตอร์รัพท์ภายหลังรับข้อมูลเข้ามา สามารถเคลียร์ได้โดยซอฟแวร์

ตารางที่ 3.2 แสดงโหมดต่างๆ ของการรับส่งแบบอนุกรม

SM0	SM1	MODE	ความหมาย	Band Rate
0	0	0	Shift Register	เปลี่ยนแปลงไม่ได้(OscillatorFrequency)
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

ไทม์เมอร์/เคาน์เตอร์

ไมโครคอนโทรลเลอร์ในตระกูล มีรีจิสเตอร์พิเศษที่สามารถเลือกใช้งานเป็นไทม์เมอร์หรือเคาน์เตอร์อย่างใดอย่างหนึ่ง รีจิสเตอร์ประเภทนี้มีอยู่ด้วยกัน 2 ตัวแต่ละตัวขนาด 16 บิต เรียกไทม์เมอร์ 0 และ ไทม์เมอร์ 1 ตามลำดับ

-ไทม์เมอร์นั้นค่าในรีจิสเตอร์ที่ใช้เป็นไทม์เมอร์ที่ถูกเลือกใช้งานจะถูกเพิ่มขึ้นทุกแมชชีน-ไซเคิล

-เคาน์เตอร์นั้นค่าในรีจิสเตอร์ที่ใช้เป็นเคาน์เตอร์ที่ถูกเลือกใช้งานจะถูกเพิ่มค่าทีละ 1 เมื่อมีการเปลี่ยนสถานะ

ไทม์เมอร์ 0 และ ไทม์เมอร์ 1

สามารถเลือกการทำงานให้เป็นไทม์เมอร์ หรือเคาน์เตอร์ได้ โดยการกำหนดค่าบิตในรีจิสเตอร์ใช้งานเฉพาะ โดยหากบิตนี้มีค่าเป็น 0 หมายถึงเลือกใช้งานเป็นไทม์เมอร์ถ้าบิตนี้เป็น 1 หมายถึงเลือกใช้งานเป็นเคาน์เตอร์

นอกจากจะเลือกการทำงานของรีจิสเตอร์ให้เป็นไทม์เมอร์ หรือเคาน์เตอร์ได้แล้วในแต่ละการทำงานยังมีการทำงานย่อยอยู่อีก 4 แบบ ตามความเหมาะสมของการใช้งาน

โหมด 0 จะใช้รีจิสเตอร์ขนาด 8 บิตเป็นตัวนับโดยมีการเพิ่มค่าครั้งละ 1 ทุกครั้งนับสัญญาณได้ครบ 32 ครั้ง โดยในโหมดนี้รีจิสเตอร์ที่ใช้นับเพียง 13 บิต (8 บิตในรีจิสเตอร์ TLx รวมกับ 5 บิตใน THx)

โหมด 1 การทำงานเหมือนกับโหมด 0 เว้นแต่ค่าในรีจิสเตอร์ถูกใช้งานครบทั้ง 16 บิตนั่นเองคือไทม์เมอร์หรือเคาน์เตอร์ในโหมดนี้มีขนาด 16 บิต

โหมด 2 ในโหมดนี้จะกำหนดรีจิสเตอร์ใช้งานในการนับเพียง 8 บิต (จากรีจิสเตอร์ TLx) ที่มีการโหลดค่าด้วยค่าในรีจิสเตอร์ THx การใช้งานโหมดนี้มีไว้เพื่อสร้างสัญญาณอินเตอร์รัปต์ที่มีคาบเวลาคงที่

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

ตารางที่ 3.3 แสดงค่าในรีจิสเตอร์ TMOD ค่าต่างๆ สำหรับ 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

ตารางที่ 3.4 แสดงค่าในรีจิสเตอร์ 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 3.5 แสดงค่าในรีจิสเตอร์ TMOD ค่าต่างๆ สำหรับ 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

ตารางที่ 3.6 แสดงค่าในรีจิสเตอร์ TMOD ค่าต่างๆ สำหรับ Counter 1

โหมด	ฟังก์ชั่นเคาน์เมอร์ 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	_ _	_ _

รีจิสเตอร์สำหรับใช้งานทั่วไปใน MCS – 51

รีจิสเตอร์ A,B และรีจิสเตอร์ใช้งานทั่วไป R0 – R7 ซึ่งอยู่ในหน่วยความจำสำหรับเก็บข้อมูลทั่วไปภายในชิฟบริเวณ 128 ไบต์แรก รีจิสเตอร์ใช้งานทั่วไป R0 – R7 ใน MCS – 51 มีอยู่ด้วยกันทั้งบหมด 4 กลุ่ม แต่ละกลุ่มประกอบด้วยรีจิสเตอร์จำนวน 8 ตัว (R0 – R7) ซึ่งมีชื่อเรียกเหมือนกัน ดังนั้นรีจิสเตอร์ใช้งานทั่วไป R0 – R7 ใน MCS – 51 จึงมีทั้งหมด 32 ตัว ในการทำงานขณะใด ๆ รีจิสเตอร์ทั้ง 4 กลุ่ม (R0 – R7) จะถูกเลือกใช้งานเพียงกลุ่มเดียวเท่านั้น การเลือกใช้งานรีจิสเตอร์ R0 – R7 กลุ่มใดกลุ่มหนึ่งใน 4 กลุ่มกระทำโดยการเซตหรือเคลียร์บิต RS0,RS1 ในรีจิสเตอร์ใช้งานเฉพาะ PSW

รีจิสเตอร์ฟังก์ชั่นพิเศษใน MCS – 51

ใน MCS–51 รีจิสเตอร์จะใช้หน่วยความจำ RAM ภายในชิป โดยส่วนหนึ่งเป็นรีจิสเตอร์พิเศษ (Special Function Register : SFR) ซึ่งมีทั้งหมด 21 ตัว โดยรีจิสเตอร์พิเศษต่าง ๆ จะเริ่มที่หน่วยความจำตั้งแต่ 80H ถึง FFH ซึ่งมีทั้งหมด 128 ตำแหน่ง แต่จะเป็นรีจิสเตอร์ฟังก์ชั่นพิเศษเพียง 21 ตำแหน่ง แต่ถ้าเป็น 8032 / 8051 จะใช้ 26 ตำแหน่งหรือมี SFR 26 ตัว

ตารางที่ 3.7 รีจิสเตอร์ใช้เฉพาะ PSW (Program Status Word) เข้าถึงข้อมูลได้ในระดับบิต

บิต	ชื่อบิต	ตำแหน่ง	ความหมาย
PSW.7	CY	D7H	Carry Flag
PSW.6	AC	D6H	Auxiliary Flag
PSW.5	F0	D5H	Flag 0
PSW.4	RS1	D4H	บิตสำหรับเลือก รีจิสเตอร์ แบงค์ 1
PSW.3	RS0	D3H	บิตสำหรับเลือก รีจิสเตอร์แบงค์ 0
			00 = bank 0; Address 00H – 07H 01 = Bank 1; Address 08H – 0FH 10 = Bank 2; Address 10H – 17H 11 = Bank 3; Address 18H – 1FH
PSW.2	OV	D2H	Overflow Flag
PSW.1	-	D1H	Reserved
PSW.0	P	D0H	Even Parity Flag

3.2 โมดูลแสดงผลแบบผลึกเหลว(LCD)

ในโมดูลLCDจะมีส่วนประกอบหลักๆ 3 ส่วน ดังนี้

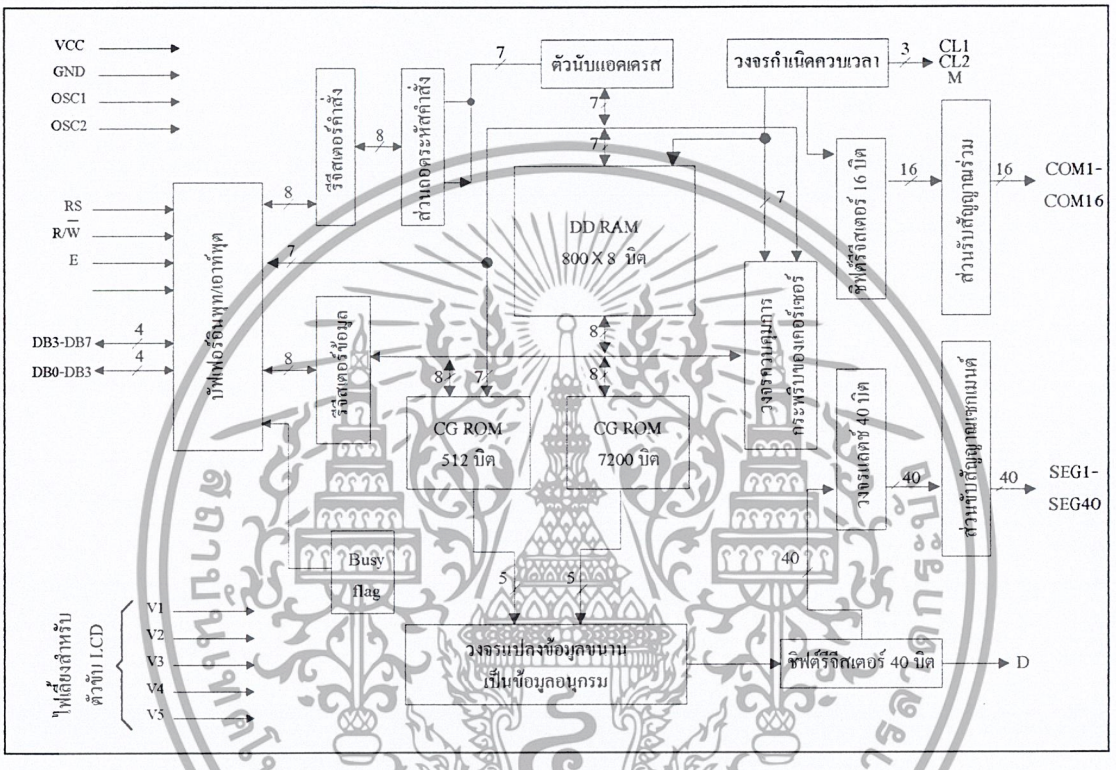
ตัวแสดงผล (display) ภายในเป็นผลึกเหลวที่สามารถแสดงผลให้เห็น โดยอาศัยแสงจากภายนอก ดังนั้นจึงต้องมีมุมในการมองข้อมูลที่แสดงผลบนจอ LCD

ตัวควบคุม (controller) เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาควบคุมการทำงานของโมดูล LCDเช่น ลบจอภาพ แสดงตัวอักษร หรือเลื่อนเคอร์เซอร์ เป็นต้น ตัวควบคุมนี้ใช้ชิปที่นิยมใช้ คือ เบอร์ HD44780 และ เบอร์ HD61830 โดยเบอร์ HD44780 จะใช้ควบคุม LCD แบบอักขระ ส่วน HD61830 ใช้ควบคุม LCD แบบกราฟิก

ตัวขับ (driver) เป็นตัวรับสัญญาณจากตัวควบคุมมาขับให้ตัวแสดงผลแสดงข้อมูลตามที่กำหนด ชิปที่ใช้ทำหน้าที่เป็นตัวขับนี้ได้แก่ เบอร์ HD44100H และ MSM5259 เป็นต้น

โครงสร้างภายในของตัวควบคุมโมดูล LCD

ในการใช้งานโมดูล LCD จำเป็นต้องทำความเข้าใจเกี่ยวกับโครงสร้างและคำสั่งที่ใช้ในการควบคุมให้ดีเสียก่อนในที่นี้จะอ้างเกี่ยวกับโมดูล LCD แบบอักขระ เพราะสามารถเข้าใจได้ง่าย รูปที่ 3.2 เป็นบล็อกไดอะแกรมภายในชิปควบคุม LCD เบอร์ HD44780 ซึ่งใช้ในโมดูล LCD แบบอักขระ ประกอบด้วย



รูปที่ 3.2 แสดงไดอะแกรมการทำงานของโมดูล LCD แบบอักขระ

บัฟเฟอร์อินพุทเอาต์พุต เป็นส่วนที่ใช้ในการติดต่อรับส่งข้อมูลกับอุปกรณ์ เพื่อที่จะถ่ายทอดข้อมูลเข้าออกภายในตัวควบคุม

รีจิสเตอร์คำสั่ง (Instruction Register :IR) เป็นรีจิสเตอร์ใช้รับคำสั่งจากอุปกรณ์ภายนอก เพื่อนำไปควบคุมการแสดงผล

รีจิสเตอร์ข้อมูล (Data Register :DR) เป็นรีจิสเตอร์ใช้รับข้อมูลคำสั่งจากอุปกรณ์ภายนอก เพื่อถ่ายทอดต่อไปยังหน่วยความจำที่ทำหน้าที่เก็บข้อมูลแสดงผล หรือนำข้อมูลไปสร้างตัวอักษรเพิ่มเติมในแรมเก็บตัวอักษร

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

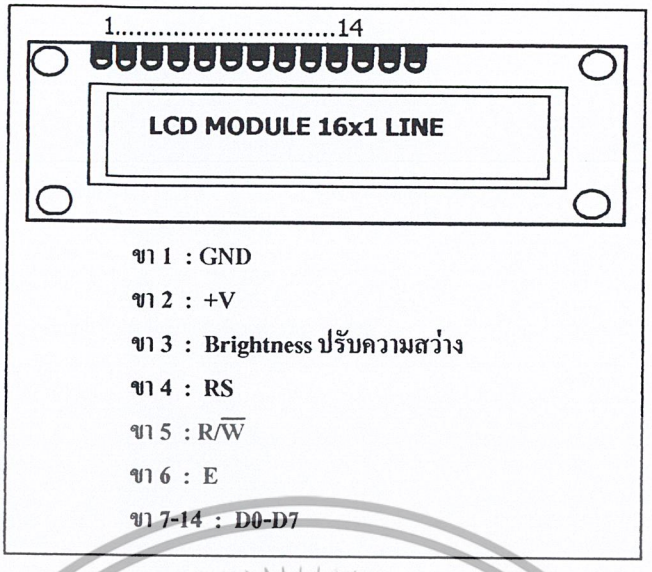
รอมเก็บตัวอักษร (Charater Generator ROM :CGROM) เป็นหน่วยความจำรอมที่ใช้เก็บข้อมูลตัวอักษรหรือสัญลักษณ์ที่สามารถอ่านออกไปแสดงที่ตัวแสดงผลได้ มีขนาด 7,200 บิต โดยจะถูกอ่านด้วยค่าของข้อมูลใน DDRAM

แรมเก็บตัวอักษร (Charater Generator RAM :CGRAM) เป็นหน่วยความจำแรมที่ใช้เก็บอักษรที่มีการสร้างเพิ่มเติมขึ้นใหม่ ในกรณีที่ตัวอักษรในCGROM ไม่เพียงพอ มีขนาด 512 บิต การเขียนและอ่านค่าไปใช้นั้นทำได้เช่นเดียวกับ CGROM คือ เขียนข้อมูลลงใน DDRAM แล้วตัวควบคุมจะมาอ่านค่าจาก CGRAM เอง

แฟลก BUSYเป็นส่วนที่ทำหน้าที่แจ้งสถานะการทำงานของตัวควบคุมให้อุปกรณ์ภายนอกทราบว่าตัวควบคุมพร้อมที่จะรับข้อมูลหรือคำสั่งหรือไม่ ดังนั้นก่อนการส่งข้อมูลหรือคำสั่งมายังตัวควบคุมต้องตรวจสอบสถานะของแฟลก BUSY นี้เสียก่อน

ตารางที่ 3.8 แสดงความสัมพันธ์ในการทำงานของขา 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



รูปที่ 3.3 แสดงรูปร่างและการจัดขาโมดูล LCD แบบอักขระ

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

สำหรับโมดูล LCD ที่ยกมาใช้ในการทดลองเป็นขนาด 16 ตัวอักษร 1 บรรทัด เนื่องจากราคาถูก หาง่าย และเป็นโมดูล LCD ที่มีโครงสร้างเป็นมาตรฐาน มีผู้ผลิตหลายราย และมีการระบุเบอร์แตกต่างกันออกไปตามผู้ผลิต อาทิ LM020L ของฮิตาชิ, DMC-16117A ของคอปเทร็กซ์ (Optrex) เป็นต้น แต่อย่างไรก็ตามคอนโทรลเลอร์ที่ใช้คือเบอร์เดียวกันนั่นคือ เบอร์ HD44750 ของฮิตาชิ

โมดูล LCD ขนาด 16X1 มีขาต่อใช้งานทั้งสิ้น 14 ขามีการจัดขาดังรูปที่ 3.3 สำหรับรายละเอียดการทำงานของแต่ละขามีดังนี้

V_{SS} **(ขา1)** : ต่อลงกราวด์

V_{DD} **(ขา2)** : ต่อไฟเลี้ยง +5 โวลต์

V_O **(ขา3)** : เป็นขาอินพุตรับแรงดันเพื่อปรับความเข้มของการแสดงผล

RS (ขา4) : เป็นขาอินพุตใช้ในการแยกชนิดของข้อมูลที่ทำการประมวลผลในขณะนั้นว่าเป็นคำสั่งสำหรับรีจีสเตอร์ IR หรือเป็นข้อมูลสำหรับรีจีสเตอร์ DR โดยที่ขานี้เป็น “0” ข้อมูลที่ส่งมาจะเป็นคำสั่ง แต่ถ้าขานี้เป็น “1” ข้อมูลที่ส่งมาจะเป็นข้อมูลสำหรับการแสดงผล

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

E (ขา6) : เป็นขาสำหรับรับสัญญาณพัลส์เอ็นเอเบิลโมดูล LCD ให้ทำงาน

D0 – D7 (ขา7) : เป็นขาที่ใช้เป็นทางผ่านของข้อมูลระหว่าง LCD กับอุปกรณ์ภายนอก ขนาด 8 บิต

อนึ่ง RS, R/W และ E จะใช้งานร่วมกัน โดยมีความสัมพันธ์แสดงในตารางที่ 3.8

คำสั่งควบคุมโมดูล LCD

ในการเขียนคำสั่งลงในตัวควบคุม แน่นอนว่าจะต้องกำหนดให้ขา RS และ R/W เป็น “0” แล้วเขียนคำสั่งตามไป คำสั่งควบคุมโมดูล LCDของชิปควบคุม HD44780 ที่สำคัญมี 10 คำสั่งดังนี้

1. คำสั่งเคลียร์ตัวแสดงผล (clear display)

มีข้อมูลเป็น 01H เป็นคำสั่งใช้เขียนข้อมูลช่องว่าง หรือ space เข้าไปใน DDRAM ทั้งหมด เมื่อตัวควบคุมเอ็กซิคิวต์คำสั่งนี้ จะทำการกำหนดแอดเดรสของ DDRAM เป็น “0” เคอร์เซอร์จะกลับไปอยู่ตำแหน่งซ้ายมือสุดของจอแสดงผล แล้วเซตบิต I/D ให้เป็น “1”

2. คำสั่ง return home

ต้องกำหนดให้บิต 1 ของข้อมูลเป็น “1” เป็นคำสั่งให้เคอร์เซอร์เคลื่อนที่กลับไปยังตำแหน่งซ้ายสุดของจอแสดงผล แต่ข้อมูลบนจอแสดงผลไม่เปลี่ยนแปลง นั่นคือ ข้อมูลคำสั่งของคำสั่งนี้จะเป็น 02 Hหรือ 03 H ก็ได้

3. คำสั่งเลือกโหมดการป้อนข้อมูล (Entry mode Set)

มี รายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

บิต S เป็นบิตที่ใช้ในการกำหนดลักกษณะการแสดงผล เมื่อมีการป้อนข้อมูลแต่ถ้าหากบิต S เป็น “1” เมื่อเกิดข้อมูลใหม่บนจอแสดงผล ตัวเคอร์เซอร์จะอยู่กับที่ แต่ตัวอักษรข้อมูลเดิมจะถูกดันไปทางด้านซ้ายมือ แต่หากบิตSนี้เป็น “0” เมื่อเกิดข้อมูลใหม่ตัวเคอร์เซอร์จะเลื่อนไปทางด้านขวามือ

บิต I/D เป็นบิตที่ใช้กำหนดว่า เมื่อเขียนหรืออ่านข้อมูลแล้ว แอดเดรสของ DDRAM เพิ่มขึ้นหรือลดลงหนึ่งแอดเดรส โดยถ้าบิตนี้เป็น “1” แอดเดรสของ DRRAM จะเพิ่มขึ้น แต่ถ้าเป็น “0” แอดเดรสจะลดลง

ดังนั้นข้อมูลคำสั่งที่เกิดขึ้นสำหรับคำสั่งนี้ได้แก่ 04H – 07H (4 ข้อมูลคำสั่ง) และที่ใช้บ่อยคือ 06H หมายถึง กำหนดให้เมื่อเกิดข้อมูลใหม่ เคอร์เซอร์จะเลื่อนไปทางด้านขวามือ และแอดเดรสของ DDRAMเพิ่มขึ้น

4. คำสั่งควบคุมการแสดงผล

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

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

บิต C ใช้ควบคุมการแสดงตัวเคอร์เซอร์บนจอแสดงผล ถ้าต้องการให้มีตัวเคอร์เซอร์แสดงผลบนจอแสดงผลต้องกำหนดบิตนี้ให้เป็น "1" ถ้ากำหนดให้เป็น"0" จะเป็นการปิดเคอร์เซอร์หรือไม่แสดงเคอร์เซอร์

บิต B ใช้ควบคุมการกระพริบของเคอร์เซอร์ ถ้าบิตนี้เป็น "1" เคอร์เซอร์จะกระพริบ

ดังนั้นจะมีข้อมูลคำสั่งได้ตั้งแต่ 08H –0FH (8 รูปแบบคำสั่ง)ที่ใช้บ่อยคือ 0CH เป็นการสั่งให้เปิดจอแสดงผลแต่ไม่แสดงเคอร์เซอร์ และ 0FH เป็นการสั่งให้เปิดจอแสดงผล แสดงเคอร์เซอร์และสั่งให้เคอร์เซอร์กระพริบ

5. คำสั่งควบคุมการเลื่อนของเคอร์เซอร์และข้อมูลตัวอักษร

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

การควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผล ขึ้นอยู่กับการกำหนดบิต S/C และ R/L ซึ่งสามารถสรุปได้ดังนี้

ตารางที่ 3.9 แสดงการควบคุมการเลื่อนเคอร์เซอร์และตัวอักษรบนจอแสดงผล

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

6. คำสั่งกำหนดฟังก์ชั่นการทำงาน

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนี้

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

บิต DL ใช้กำหนดจำนวนบิตที่ใช้ติดต่อส่งผ่านข้อมูล ถ้าบิตนี้เป็น “0” จะเป็นการติดต่อแบบ 4 บิต แต่ถ้าบิตนี้เป็น “1” จะเป็นการติดต่อแบบ 8 บิต

บิต N ใช้กำหนดจำนวนบรรทัดของการแสดงผล ถ้าเป็น “0” จะแสดงผล 1 บรรทัด ถ้าเป็น “1” จะแสดงผล 2 บรรทัด ในกรณีที่จอแสดงผลสามารถแสดงได้มากกว่า2บรรทัด และต้องการแสดงผลให้ได้มากกว่า2บรรทัดก็กำหนดบิต N นี้ให้เป็น “1” จุดที่น่าสังเกตคือโมดูลLCDแบบ 16 ตัวอักษร1บรรทัด แม้จะมีบรรทัดการแสดงผลเพียง1บรรทัด แต่จะต้องกำหนด N นี้ให้เป็น “1” เนื่องจากแอดเดรสของ DDRAM แบ่งเป็น 2 ช่องคือ 00H และ 40H

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

ข้อมูลคำสั่งที่ใช้บ่อยคือ 38H เป็นการกำหนดให้โมดูล LCD ทำงานในแบบ 8 บิตแสดงผล 2 บรรทัดและเลือกความละเอียดเป็น 5x7จุด

7. คำสั่งเลือกแอดเดรสของ CGRAM

เมื่อต้องการกำหนดแอดเดรสของต้องกำหนดให้บิต7เป็น “0” บิต 6 เป็น “1” ส่วนอีก 6 บิตที่เหลือจะแทนด้วยค่าแอดเดรสของ CGRAM จะต้องทำการกำหนดแอดเดรสด้วยคำสั่งนี้ ก่อนที่จะอ่านหรือเขียนข้อมูลให้ CGRAM โดยแอดเดรสของ CGRAM อยู่ระหว่าง 00H-3FH

8. คำสั่งเลือกแอดเดรสของ DDRAM

ใช้ในการเลือกแอดเดรสของ DDRAM ก่อนที่จะทำการอ่านหรือเขียนข้อมูล โดยบิต 7 ต้องเป็น “1” และข้อมูลอีก 7 บิต ที่เหลือจะเป็นค่าแอดเดรสของ DDRAM ซึ่งแอดเดรสของ DDRAM จะอยู่ระหว่าง 8CH-0FFH ทั้งนี้จำนวนแอดเดรสยังขึ้นกับการกำหนดสถานะที่บิต N ด้วย หากบิตด้วย N หากบิต N เป็น “0” แอดเดรสของ DDRAM จะอยู่ระหว่าง 80H-0CFH และถ้าบิต N เป็น “1” แอดเดรสของ DDRAM จะมี 2 ช่วงคือ 8CH-87H และ 0COH-0C7H

9. คำสั่งอ่านแฟลก BUSY และแอดเดรส

มีรายละเอียดของรูปแบบข้อมูลคำสั่งดังนั้น

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

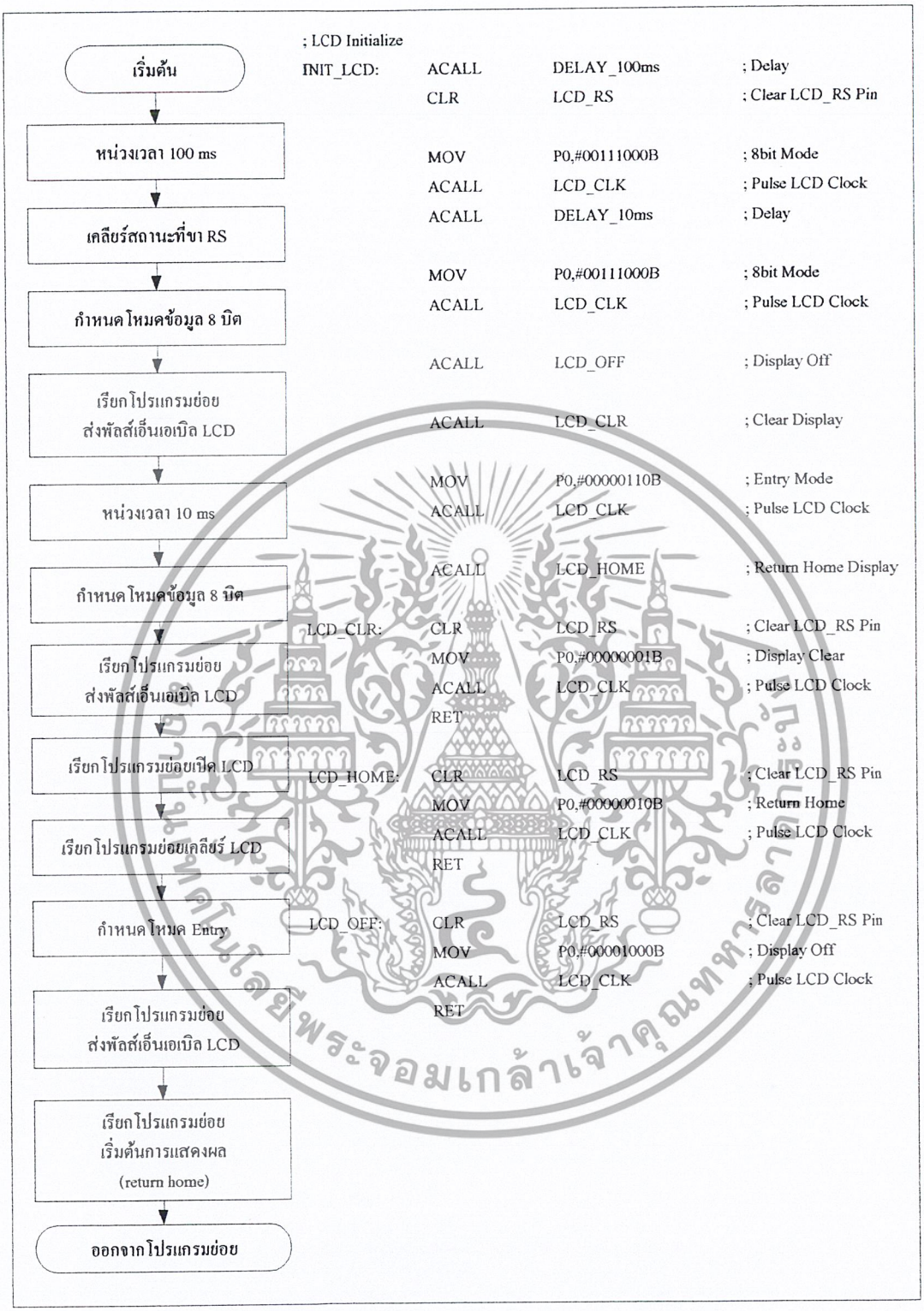
เป็นคำสั่งที่ใช้อ่านแฟลก BUSY (BF) โดยแฟลกนี้จะตัวบอกสถานะของตัวควบคุม LCD ว่าพร้อมจะรับข้อมูลอยู่หรือไม่ ถ้าหากบิต BF เป็น “0” แสดงว่าตัวควบคุม LCD พร้อมรับข้อมูลหรือคำสั่ง แต่ถ้าเป็น “1” แสดงว่า ขณะนี้ตัวควบคุม LCD ยังอยู่ในกระบวนการทำงานภายในหรือกำลังประมวลข้อมูลอยู่ ยังไม่พร้อมรับข้อมูลหรือคำสั่ง เมื่อต้องการอ่านแฟลกต้องกำหนดให้ขา R/W เป็น “1” ด้วย แต่สัญญาณที่ RS ยังต้องเป็น “0” อยู่เพราะข้อมูลนี้เป็นข้อมูลคำสั่ง นอกจากนี้ยังใช้เป็นคำสั่งอ่านข้อมูลแอดเดรสของ DGRAM และ DDRAM ด้วย โดยบิต 0 ถึง บิต 6 เป็นค่าข้อมูลของแอดเดรสที่ต้องการอ่าน

การเขียนคำสั่งและข้อมูลให้แก่โมดูล LCD

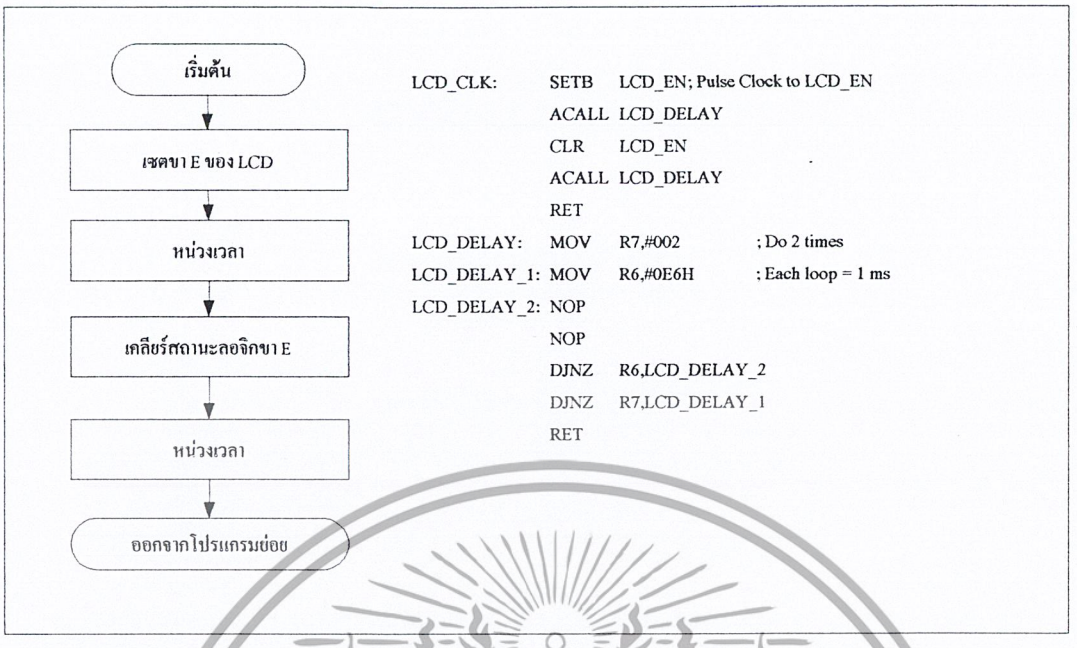
ในการเขียนข้อมูลเพื่อควบคุมให้โมดูล LCD แสดงผลตามที่ผู้ใช้งานต้องการ ต้องส่งคำสั่ง (instruction) แล้วกำหนดโหมดการทำงานให้แก่โมดูล LCD ก่อน จากนั้นจึงค่อยส่งข้อมูล (data) ที่ต้องการแสดงผล เนื่องจากบัสข้อมูลของโมดูล LCD มี 8 เส้นคือ D0-D7 และใช้เป็นทางผ่านของทั้งคำสั่งและข้อมูล ดังนั้นในการส่งคำสั่งและข้อมูลจึงต้องอาศัยการกำหนดสัญญาณลอจิกที่ขา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

เมื่อต้องการเขียนหรืออ่านข้อมูลใน CGRAM และ DDRAM เริ่มต้นต้องกำหนดแอดเดรสที่ต้องการอ่านหรือเขียนก่อน โดยใช้คำสั่งเลือกแอดเดรส จากนั้นกำหนดให้ขา RS เป็น “1” เพื่อแจ้งให้ตัวควบคุมภายใน โมดูล LCD ทราบว่าข้อมูลที่ปรากฏต่อไปนี้เป็นข้อมูลปกติไม่ใช่คำสั่ง ในกรณีที่ต้องการอ่านข้อมูลต้องกำหนดให้ขา R/W เป็น “1” ข้อมูลขนาด 8 บิต (หรือ 4 บิต) ก็จะปรากฏบนบัสข้อมูล โดยข้อมูลที่อ่านออกมาได้จะเป็นข้อมูลจากแอดเดรสของ CGRAM หรือ DDRAM ตามที่ต้องการ

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



รูปที่ 3.4 แสดงโฟลวชาร์ตและตัวอย่างโปรแกรมย่อยการอินนิเชียลโมดูล LCD



รูปที่ 3.5 แสดงโฟลวชาร์ตและตัวอย่างโปรแกรมย่อยส่งพัลส์เอ็นเอเบิลให้แก่โมดูล LCD

จังหวะการทำงานของ LCD โมดูล

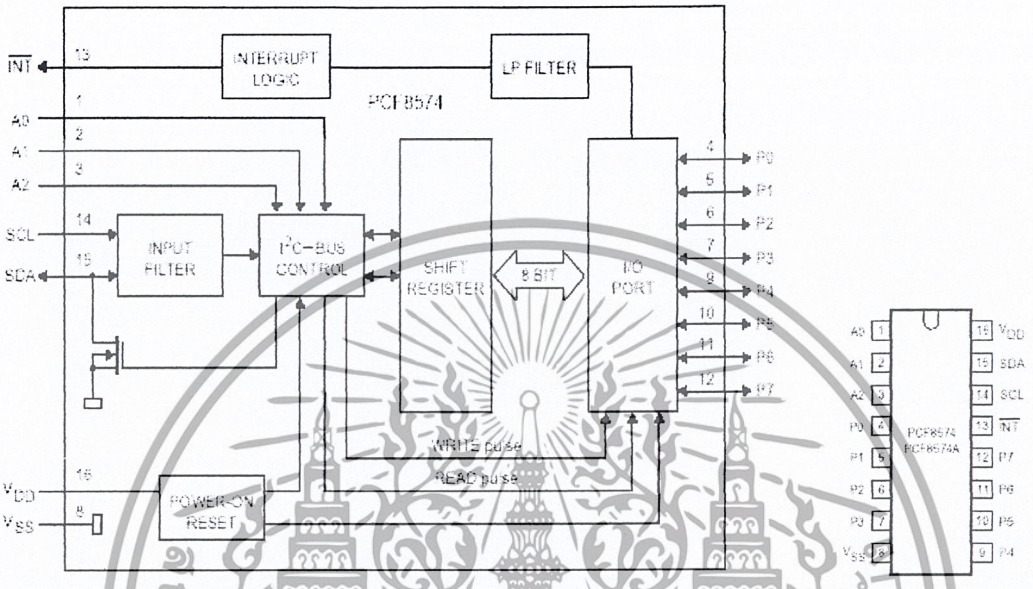
ในการติดต่อกับโมดูล LCD จะต้องมีการหน่วงเวลาหลังจากที่ทำการส่งรหัสคำสั่งหรือข้อมูล เนื่องจากต้องรอให้คอนโทรลเลอร์ภายใน LCD โมดูล แปลความหมายของรหัสคำสั่งและทำงานตามคำสั่งให้เรียบร้อยก่อนจากนั้นจึงจะรับข้อมูลหรือคำเนินการต่อไป

ดังนั้น ในการใช้งานโมดูล LCD ผู้เขียนโปรแกรมต้องมีโปรแกรมเพื่อหน่วงเวลารอให้โมดูล LCD พร้อมทำงานด้วย โดยเมื่อเริ่มจ่ายไฟให้แก่โมดูล LCD ต้องประมาณ 10 มิลลิวินาที เพื่อให้โมดูล LCD ทำการเตรียมความพร้อมหรืออินิเชียล (initial) หลังจากนั้นก็จะกำหนดลอจิกให้แก่ขา RS ของโมดูล LCD แล้วต้องหน่วงเวลาอีกประมาณ 2 มิลลิวินาที เพื่อให้คอนโทรลเลอร์ในโมดูล LCD แปลความหมายของลอจิกที่ขา RS ว่า ข้อมูลต่อไปที่จะได้รับนั้นเป็นรหัสคำสั่งหรือเป็นข้อมูลที่ต้องการแสดงผล จากนั้นจะเป็นการส่งข้อมูลมารอที่บัสข้อมูล D0-D7 (กรณีทำงานในโหมด 8 บิต) ขั้นตอนต่อไปจะเป็นการส่งสัญญาณพัลส์ไปที่ขา E เพื่อเอ็นเอเบิลโมดูล LCDให้รับข้อมูลจากบัสข้อมูลเข้าไป โดยพัลส์ที่ป้อนเข้าที่ขา E ของ โมดูล LCD ต้องเป็นพัลส์ขอบขาขึ้น จากนั้นทำการหน่วงเวลา 2 มิลลิวินาที

ทั้งหมดที่กล่าวมาคือขั้นตอนและจังหวะในหารทำงาน 1 รอบของโมดูล LCD จะเห็นได้ว่ามีโปรแกรมย่อยที่สำคัญอยู่ 3 โปรแกรมย่อยคือ โปรแกรมอินิเชียล LCD,โปรแกรมหน่วงเวลา และโปรแกรมย่อยการส่งพัลส์เพื่อเอ็นเอเบิลโมดูล LCD โปรแกรมตัวอย่างของโปรแกรมย่อยทั้งสามพร้อมทั้งโฟลวชาร์ตแสดงไว้แล้วในรูปที่ 3.4 และ 3.5

3.3 ไอซีขยายพอร์ตเบอร์ PCF 8574A

คุณลักษณะของ PCF8574A



รูปที่ 3.6 แสดงโครงสร้างภายในไอซีขยายพอร์ตเบอร์ PCF 8574A

ข้อมูลเบื้องต้นของ PCF8574A

ไอซี PCF 8574A มีคุณสมบัติดังนี้

- ทำงานที่ระดับแรงดันตั้งแต่ 2.5 V ถึง 6 V
- กินกระแสในสภาวะสแตนด์บายต่ำเพียง 10 mA
- ใช้การเชื่อมต่อแบบบัส I^2C
- มีเอาต์พุตอินเตอร์รัปต์แบบเดรนเปิด
- เอาต์พุตสามารถแลตช์หรือคงค่าไว้ได้ และขับ LED ได้ โดยต่อวงจรในลักษณะกระแสซิงก์ หรือทำงานด้วยลอจิก “0”
- สามารถกำหนดแอดเดรสของไอซีแต่ละตัวได้ทางฮาร์ดแวร์ ด้วยขา A0 - A2 ทำให้สามารถต่อพ่วงกันได้ถึง 8 ตัวต่อเบอร์
- ในอนุกรมของ PCF8574A ยังมีเบอร์ PCF8574A (ไม่ลงท้ายด้วย A) ทำให้สามารถต่อพ่วงทั้ง PCF8574 และ PCF8574A ได้ส่งผลให้สามารถขยายพอร์ตได้มากถึง 128 บิต

- ขาพอร์ตทั้ง 8 ขาของ PCF8574A สามารถกำหนดให้เป็น อินพุต หรือเอาต์พุต ได้โดยอิสระโดยไม่จำเป็นต้องใช้คำสั่งควบคุมเพื่อเลือกให้เป็นขา เอาต์พุต หรืออินพุต
- มีกระแสในขณะลอจิก "1" เพียง 100 μA เท่านั้น จึงไม่เหมาะที่จะใช้ไอซี PCF 8574A ในการขับกระแสซอร์ส แต่จะเหมาะกับการนำไปขับกระแสซิงก์มากกว่า

เมื่อต้องการให้ขาพอร์ตเหล่านี้ ทำหน้าที่เป็นอินพุตจะต้องส่งสัญญาณให้ขาเหล่านี้มีลอจิก "1" เสียก่อน เมื่อขาอินพุตได้รับสัญญาณจากภายนอกป้อนเข้ามา ไอซี PCF8574A จะสร้างสัญญาณอินเตอร์รัปต์ (INT) ป้อนให้ไมโครคอนโทรลเลอร์ หรือคอมพิวเตอร์รับรู้แทนการต้องคอยตรวจสอบขาอินพุตต่ออยู่ตลอดเวลา สัญญาณอินเตอร์รัปต์นี้จะถูกรีเซตเมื่อมีการอ่านข้อมูลหรือมีการเปลี่ยนค่าของอินพุตไปสู่ค่าเดิม

การเขียนโปรแกรมควบคุม PCF8574A

เนื่องจาก PCF8574A มีการเชื่อมต่อแบบบัส I^2C ดังนั้นการติดต่อจึงสามารถใช้โปรแกรมย่อยการติดต่อกับอุปกรณ์สเลฟ, การสร้างสภาวะเริ่มต้น, สภาวะหยุด, การอ่านและเขียนข้อมูลที่แสดงในรูปที่ 4.2 ส่วนที่สามารถเปลี่ยนแปลงได้สำหรับ PCF8574A คือข้อมูลกำหนดแอดเดรส โดยข้อมูลของ PCF8574A มีรูปแบบดังนี้

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	1	1	1	A2	A1	A0	R/$\overline{W}$

บิต A0, A1, A2 ใช้ในการระบุ PCF8574A ที่ใช้บนบอร์ดในกรณีที่มีการต่อ PCF8574 มากกว่า 1 ตัวโดยค่าของ A0–A2 จะมีความแตกต่างกันไปในแต่ละตัว สามารถกำหนดได้ทางฮาร์ดแวร์โดยการต่อขา A0–A2 เข้าไปกับไฟเลี้ยง + 5 โวลต์ เพื่อกำหนดเป็นลอจิก "1" หรือกราวด์เพื่อกำหนดเป็นลอจิก "0" ดังนั้นค่าแอดเดรสของ PCF8574A 1 ตัวจึงต้องนำสถานะที่กำหนดทางฮาร์ดแวร์ที่ขา A0-A2 มารวมด้วยจึงจะสมบูรณ์ ส่วนบิต R/W ใช้ PCF8574A กำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับไอซี PCF8574A ยกตัวอย่าง ถ้าหากกำหนดขา A0-A2 ลงกราวด์ทั้งหมดและต้องการอ่านข้อมูลจาก PCF8574A ข้อมูลกำหนดแอดเดรสที่ต้องการส่งให้แก่ PCF8574A คือ 01110001B เป็นต้น

ไอซี PCF8574A ยังมีอีกเบอร์หนึ่งในอนุกรมเดียวกัน นั่นคือ PCF8574 ซึ่งมีข้อมูลกำหนดแอดเดรสที่แตกต่างกับ PCF8574A แต่ฟังก์ชั่นการทำงานเหมือนกันทุกประการ โดยข้อมูลกำหนดแอดเดรสของ PCF8574 มีดังนี้

ตารางที่ 3.10 รายละเอียดหน้าที่การทำงานของแต่ละขาของไอซี PCF8574/ PCF8574A

ชื่อ	ตำแหน่งขา	หน้าที่
A0	1	อินพุตแอดเดรสตัวที่1
A1	2	อินพุตแอดเดรสตัวที่2
A2	3	อินพุตแอดเดรสตัวที่3
P0	4	พอร์ตอินพุตเอาต์พุต2ทิศทางบิต0
P1	5	พอร์ตอินพุตเอาต์พุต2ทิศทางบิต1
P2	6	พอร์ตอินพุตเอาต์พุต2ทิศทางบิต2
P3	7	พอร์ตอินพุตเอาต์พุต2ทิศทางบิต3
V_{SS}	8	กราวด์
P4	9	พอร์ตอินพุตเอาต์พุต2ทิศทางบิต4
P5	10	พอร์ตอินพุตเอาต์พุต2ทิศทางบิต5
P6	11	พอร์ตอินพุตเอาต์พุต2ทิศทางบิต6
P7	12	พอร์ตอินพุตเอาต์พุต2ทิศทางบิต7
INT	13	ขาเอาต์พุตอินเตอร์รัปต์(ทำงานที่ลอจิก 0)
SCL	14	ขาสัญญาณนาฬิกาสำหรับ I^2Cบัส
SDA	15	ขาข้อมูลสำหรับ I^2Cบัส
V_{DD}	16	ไฟเลี้ยง

บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
0	1	0	0	A2	A1	A0	R/$\overline{W}$

ดังนั้น จึงสามารถต่อพ่วงไอซีในอนุกรม PCF8574X ได้สูงถึง16 ตัว

การส่งหรือเขียนข้อมูลไปยัง PCF8574A โดยมีลำดับขั้นตอนดังนี้

1. เตรียมข้อมูลกำหนดแอดเดรสของ PCF8574A
2. เรียกโปรแกรมย่อยการติดต่อกับอุปกรณ์สเลฟ
3. รอรับการตอบรับจาก PCF8574A
4. ส่งข้อมูลไปยัง PCF8574A
5. เรียกโปรแกรมย่อยสภาวะหยุด

สามารถเขียนโปรแกรมได้ดังนี้

```
PCF8574_WR:   MOV   I2C_ADDR,#PCF8574_ID   ; Set PCF8574 as I2C Write Slave
              ACALL I2C_SLAVE              ; Connect Slave

              MOV   I2C_DATA,IO_DATA       ; Write IO_DATA to Slave
              ACALL I2C_DATA_WR            ; Write Data to Slave

              ACALL I2C_STOP               ; Send Stop Condition
              RET
```

การอ่านข้อมูลจาก PCF8574Aก็มีลักษณะใกล้เคียงกันคือมีรูปแบบดังนี้

1. เตรียมข้อมูลกำหนดแอดเดรสของ PCF8574A
2. เรียกโปรแกรมย่อยการติดต่อกับอุปกรณ์สเลฟ
3. รอรับการตอบกลับจาก PCF8574A
4. อ่านข้อมูลจาก PCF8574A โดยใช้โปรแกรมย่อยการอ่านข้อมูลจากอุปกรณ์สเลฟ
5. เรียกโปรแกรมย่อยสภาวะหยุด

สามารถเขียนโปรแกรมได้ดังนี้

```
PCF8574_RD:   MOV   I2C_ADDR,#PCF8574_ID+1 ; Set PCF8574 as I2C Read Slave
              ACALL I2C_SLAVE              ; Connect Slave

              ACALL I2C_DATA_RD            ; Read Data from Slave
              MOV   IO_DATA,I2C_DATA       ; Read Data to IO_DATA
              ACALL I2C_NACK_BIT           ; Send Not Acknowledge

              ACALL I2C_STOP               ; Send Stop Condition
              RET
```

3.4 ไอซี ADC/DAC เบอร์ PCF 8591

การแปลงสัญญาณอะนาลอกดิจิตอลแบบซักเซสซีฟแอปพร็อกซิเมชั่น (Successive Approximation ADC)

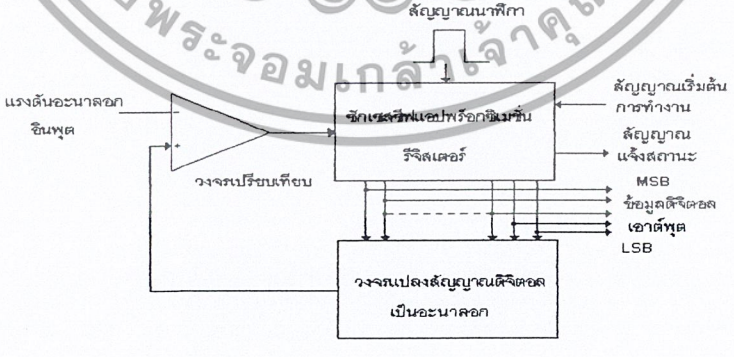
การแปลงสัญญาณอะนาลอกเป็นดิจิตอล (ADC) ที่ได้รับความนิยมสูงและมีประสิทธิภาพดี คือ การแปลงแบบซัสซีฟแอปพร็อกซิเมชั่น ไอซี ADC ที่เลือกมาทำการทดลองนี้ก็ใช้วงจร ADC แบบเดียวกันนี้ จึงต้องทำความเข้าใจพื้นฐานการทำงานของวงจร ADC แบบนี้ก่อน

ถ้าจะแปลงเป็นไทยอาจเรียกกระบวนการ ADC แบบซัสเซสซีฟแอปพร็อกซิเมชั่นนี้ว่าเป็นการแปลงแบบประมาณค่าใกล้เคียง ไดอะแกรมการทำงานแสดงในรูปที่ 3.7 ส่วนสำคัญหลัก คือ วงจรเปรียบเทียบแรงดันวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอกหรือ DAC, สัญญาณนาฬิกา และส่วนควบคุมลอจิก

วงจร ADC แบบซัสเซสซีฟแอปพร็อกซิเมชั่นนี้จะใช้รีจิสเตอร์เลขฐานสองหรือไบนารีรีจิสเตอร์ในการส่งข้อมูลดิจิตอลของวงจร DAC ภายในแต่ละบิตของรีจิสเตอร์จะเซตและรีเซต โดยการควบคุมจากวงจรควบคุมต่อไปนี้จะอธิบายการทำงานของ ADC แบบนี้ไปทีละขั้น

กำหนดให้แรงดันอะนาลอกอินพุต (Vin) มีค่า 13.5 V

1. ส่งสัญญาณเริ่มต้นการทำงาน (Start converter) มายังซัสเซสซีฟพร็อกซิเมชั่นรีจิสเตอร์
2. ขณะนี้สถานะของรีจิสเตอร์จะไม่ว่าง (busy) สัญญาณนาฬิกาลูกแรกถูกส่งเข้ามาเพื่อกำหนดให้ค่าของรีจิสเตอร์เท่ากับ 0000



รูปที่ 3.7 ไดอะแกรมแสดงการทำงานของวงจรADC แบบซักเซสซีฟแอปพร็อกซิเมชั่น

3. เอาต์พุตของ DAC จะเป็น 0V ส่งไปในวงจรเปรียบเทียบ เพื่อเปรียบเทียบกับ Vin ในขณะนี้จะได้เอาต์พุตเท่ากับ -5V กำหนดเป็นลอจิก “0”

4. เมื่อสัญญาณนาฬิกาลูกต่อไปเข้ามา จะทำการเซตบิต MSB ของรีจิสเตอร์เป็น “1”

5. ในกรณีนี้เป็น ADC ขนาด 4 บิต ดังนั้นการที่บิต MSB เซต จะทำให้วงจร DAC แปลงค่าเป็นแรงดัน 8V นำไปเปรียบเทียบที่วงจรเปรียบเทียบแรงดัน แต่ก็ยังน้อยกว่า Vin ดังนั้นเอาต์พุตของวงจรเปรียบเทียบยังคงเป็น “0” ทำให้รีจิสเตอร์ยังคงค่าบิต MSB ให้เป็น “1” ต่อไป

6. ต่อมาบิต B2 (ถัดจากบิต MSB 1 บิต เนื่องจากมี 4 บิต กำหนดบิต MSB=B3) จะเซตซึ่งมีค่าเท่ากับ 4V นำไปรวมกับค่าของบิต MSB ที่มีอยู่ 8V เช่น 12V นำไปเปรียบเทียบกับ Vin ก็ยังน้อยกว่ารีจิสเตอร์จึงยังคงค่า B2 ไว้ที่ “ 1 ” เช่นกัน

7. ต่อมาบิต B1 จะเซตทำให้แรงดันเอาต์พุตมา DAC กลายเป็น 8+4+2 =14V ซึ่งมากกว่า Vin ทำให้วงจรเปรียบเทียบเกิดการเปลี่ยนสถานะเป็น “ 1 ” ซึ่งจะส่งสัญญาณมาควบคุมให้ B1 กลายเป็น “0”

8. เมื่อบิต LSB ถูกเซต จะมีค่าแรงดัน 1 V เข้ามารวมกับค่าของ B3,B2 และ B1 เป็น 8+4+0+1=13V นำไปเปรียบเทียบกับ Vin ปรากฏว่าน้อยกว่า Vin ทำให้ที่บิต B0 หรือ LBS มีค่าเป็น “1”

9. ขณะนี้ทุกบิตในรีจิสเตอร์ถูกนำมาแปลงค่าเรียบร้อยแล้ว ทำให้สถานะของรีจิสเตอร์กลับมาเป็น พร้อมทำงาน(ready)

10. ข้อมูลดิจิตอลที่ได้จากการ ADC แบบนี้ จะมีค่า 1101_2 หรือ 13V ซึ่งใกล้เคียงกับ Vin 13.5V มากที่สุด ถ้าหากรีจิสเตอร์มีจำนวนบิตมากกว่านี้ ความละเอียดของข้อมูลที่แปลงได้จะมีความใกล้เคียงมากขึ้น ช่วงเวลาของการแปลงสัญญาณจะเริ่มต้นขึ้นตั้งแต่สัญญาณนาฬิกาลูกแรกถูกส่งเข้าไปเตรียมระบบไปจนถึงเมื่อสภาวะของรีจิสเตอร์กลับมาเป็น “พร้อมทำงาน” อีกครั้งหนึ่ง ซึ่งจะต้องใช้จำนวนสัญญาณนาฬิกาเท่ากับ n+1 พัลส์ โดย n เท่ากับจำนวนบิตของรีจิสเตอร์

ดังนั้นถ้าหาก ADC แบบซักเซสซีฟแอปพร็อกซิเมชั่นขนาด 4 บิต ตามตัวอย่างที่อธิบายมานี้ใช้สัญญาณนาฬิกาความถี่ 50 kHz เวลาที่ใช้ทั้งหมดในการแปลงสัญญาณจะคำนวณได้ดังนี้

(1) คำนวณคาบเวลาของสัญญาณนาฬิกา

$f_{clk} = 50\ kHz = 50 \times 10^3$

$T = 1/50 \times 10^3 = 20$ มิลลิวินาที

(2) จำนวนสัญญาณนาฬิกาทั้งหมดที่ใช้ในการแปลงเท่ากับ n+1, n มีค่าเท่ากับ 4 เนื่องจากมีจำนวน 4 บิต ดังนั้นจำนวนสัญญาณนาฬิกาที่ใช้ทั้งหมดจึงเท่ากับ 4+1 = 5

(3) เวลาทั้งหมดที่ใช้เท่ากับ 5 x 20 = 100 มิลลิวินาที

จะเห็นว่าวงจร ADC แบบซักเซสซีฟแอปพร็อกซิเมชั่น มีความเร็วในการทำงานสูงพอสมควรเหมาะอย่างยิ่งในการนำใช้กับไมโครคอนโทรลเลอร์ขนาดกลางอย่าง MCS-51

ความเที่ยงตรงของวงจร ADC

เป็นการเปรียบเทียบแรงดันอะนาลอก ของวงจร ADC กับแรงดันที่ควรจะเกิดขึ้นจริง ยกตัวอย่างที่ข้อมูลดิจิตอลสูงสุดของวงจร ADC ขนาด 8 บิตเมื่อเทียบเป็นแรงดันอะนาลอกควรมีค่าเท่ากับ 5.00 V แต่จากการคำนวณในตัวอย่างก่อนหน้านี้ได้ค่าแรงดัน 4.9804 V นั่นคือเกิดความผิดพลาดไป 0.0195 V หรือ 19.5 mV แต่การบอกค่าความเที่ยงตรงของวงจร ADC มักระบุเป็นจำนวนที่ เทียบกับ VLSB ดังนั้นในวงจร ADC ขนาด 8 บิต ที่ยกเป็นตัวอย่างนี้จึงมีค่าความเที่ยงตรง (หรือบางทีเรียกเป็นค่าความผิดพลาด) เป็น ±1/2LSB

ค่าเวลาในการแปลงสัญญาณ (conversion time)

เป็นค่าของเวลาทั้งหมดที่วงจร ADC แบบวงจรนับแรมป์ และแบบซักเซสซีฟแอปพร็อก-ซิเมชั่นใช้ในการแปลงสัญญาณอะนาลอกเป็นดิจิตอลจนเสร็จสิ้นพารามิเตอร์ตัวนี้มักจะปรากฏในคุณสมบัติของไอซีที่ทำงานเป็นวงจร ADC เมื่อไอซีแปลงสัญญาณเสร็จสิ้นลง จะส่งสัญญาณที่เรียกว่า EOC (End of conversion) ออกมา

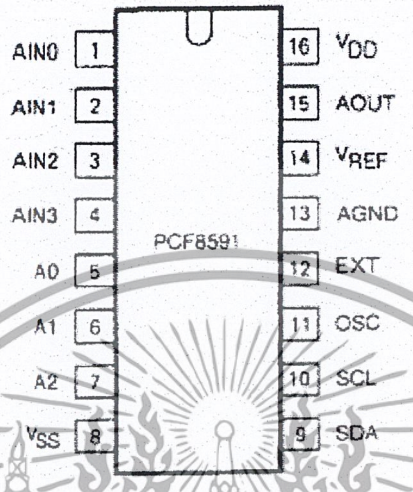
ค่าเวลาในการแปลงสัญญาณของวงจร ADC จะขึ้นอยู่กับจำนวนบิตของวงจร, ค่าความถี่ของสัญญาณนาฬิกาที่ใช้ในการแปลงสัญญาณและขนาดของสัญญาณอะนาลอกอินพุต

ข้อมูลเบื้องต้นของ PCF8591

ในการทดลองนี้จะใช้ไอซี ADC ที่ความสามารถสูงเบอร์ PCF8591 เนื่องจากในตัวมันมีวงจร ADC แบบซักเซสซีฟแอปพร็อกซิเมชั่นขนาด 8 บิตสูงถึง 4 ช่อง ทั้งยังมีวงจร DAC อีก 1 ช่องด้วย ระบบการเชื่อมต่อเป็นแบบบัส I^2C ทำให้สายสัญญาณเพียง 2 เส้น ทั้งยังสามารถต่อพ่วงกันได้สูงสุด 8 ตัว ทำให้ได้วงจร ADC รวมสูงถึง 32 ช่อง และวงจร DAC รวม 4 ช่อง สามารถนำไปประยุกต์ใช้งานได้อย่างกว้างขวางมีรายละเอียดคุณสมบัติทางเทคนิคดังนี้

- ทำงานโดยใช้แหล่งจ่ายไฟชุดเดียว
- ทำงานที่แรงดัน 2.5V ถึง 6V
- กินกระแสขณะอยู่ในสภาวะสแตนบายด์ต่ำ
- ติดต่อกับไมโครคอมพิวเตอร์หรือไมโครคอนโทรลเลอร์ผ่านระบบบัส I^2C
- เลือกตำแหน่งแอดเดรสทางฮาร์ดแวร์จากขา A0,A1,A2 ทำให้สามารถต่อพ่วงกนได้สูงสุดถึง 8 ตัว
- อัตราการสุ่มข้อมูล (samping) ขึ้นอยู่กับความเร็วของสัญญาณนาฬิกาบนบัส I^2C

- วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลเป็น แบบซักเซสซีฟแอปพร็อกซิเมชั่น ขนาด 8 บิต
- มีวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอกขนาด 8 บิต 1 ช่อง



รูปที่ 3.8 การจัดขาของไอซี ADC/DAC ขนาด 8 บิต ผ่านบัส I^2C เบอร์ PCF8591

PCF8591 สามารถทำหน้าที่เป็นไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลขนาด 8 บิต 4 ช่องและทำหน้าที่เป็นไอซีแปลงดิจิตอลเป็นอะนาลอกได้ในคราวเดียวกัน ด้วยการควบคุมผ่านระบบบัส I^2C ทำให้สามารถพ่วงไอซี PCF8591 ได้สูงสุดถึง 8 ตัว รองรับการอ่านค่าสัญญาณอะนาลอกอินพุตได้สูงสุดถึง 32 ช่อง และสามารถส่งสัญญาณอะนาลอกเอาต์พุตสูงสุดถึง 8 ช่อง ด้วยการกำหนดแอดเดรสจากขา A0 ,A1และA2 การจัดขาของ PCF8591 แสดงในรูปที่ 3.8 ส่วนรายละเอียดขาต่างๆมีดังนี้

ขาAN0-AN3(ขา1-4) เป็นขาอินพุตสำหรับป้อนสัญญาณอะนาลอกที่ต้องการแปลงค่า

ขาA0-A2 (ขา 5-7) เป็นขาสำหรับกำหนดข้อมูลแอดเดรสทางฮาร์ดแวร์ ปกติต่อลงกราวด์ แต่ถ้ามีการใช้งาน PCF8591 มากกว่า 1 ตัว ต้องกำหนดการต่อขา A0 -A2 ของ PCF8591 ให้ไม่ตรงกัน จึงสามารถต่อใช้งานร่วมกันได้ถึง 8 ตัว

ขา VSS(ขา 8) เป็นขาต่อกราวด์

ขา SDA,SCL (ขา 9 และ 10) เป็นขาเชื่อมต่อระบบบัส I^2C

ขาOSC (ขา 11) เป็นขาสำหรับต่อสัญญาณนาฬิกาภายนอก เมื่อขาEXT ต่อ กับไฟ +5V และจะทำงานเป็นขาเอาต์พุตสัญญาณนาฬิกาถ้าขา EXT ต่อลงกราวด์

ขา EXT (ขา 12) เป็นขาสำหรับเลือกแหล่งกำเนิดสัญญาณนาฬิกา ถ้าต่อไฟ +5 V จะเป็นการเลือกใช้สัญญาณนาฬิกาจากภายนอก โดยต่อสัญญาณนาฬิกาเข้าที่ขา OSC ถ้าต่อขานี้ลงกราวด์จะเป็นการเลือกใช้สัญญาณนาฬิกาจากภายใน

ขาAGND (ขา 13) เป็นขากราวด์ของแรงดันอ้างอิง ปกติต่อลงกราวด์

ขาV_{REF} (ขา 14) เป็นขาสำหรับป้อนแรงดัน ปกติต่อเข้ากับไฟเลี้ยง +5 V

ขา AOUT (ขา 15) เป็นขาเอาต์พุตของวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอก

ขาV_{DD} (ขา 16) เป็นขาต่อไฟเลี้ยง จ่ายได้ตั้งแต่ +2 ถึง +6 V ปกติใช้ +5 V

รายละเอียดฟังก์ชั่นต่างๆของ PCF8591

ตำแหน่งแอดเดรส

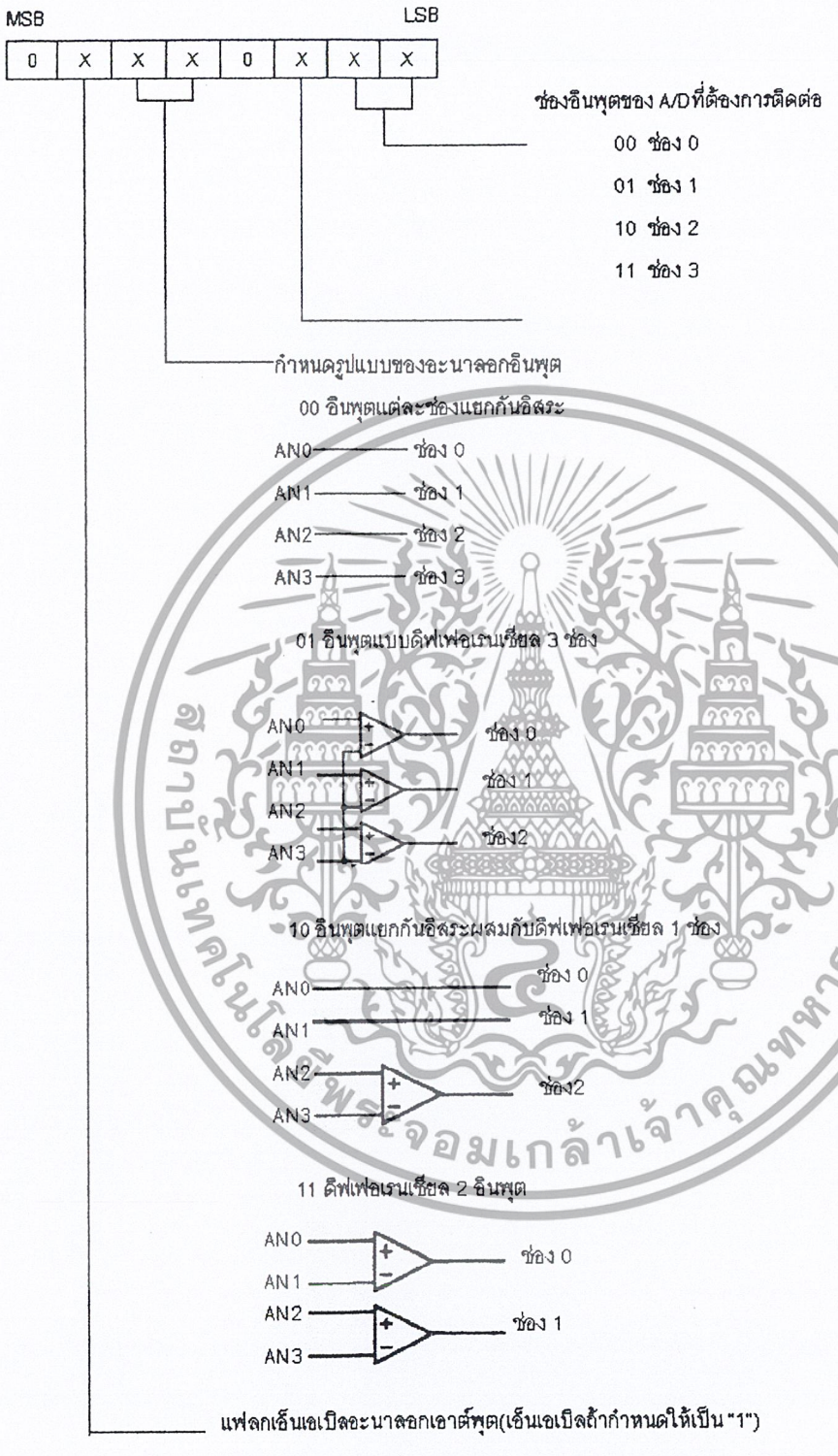
ในระบบบัส การติดต่อกับอุปกรณ์แต่ละตัวต้องระบุแอดเดรสของอุปกรณ์เหล่านั้นอย่างชัดเจนถ้าเป็นการอ้างถึงแบบ 7 บิต ข้อมูลกำหนดแอดเดรส 4 บิตบนจะเป็นค่าแอดเดรสเฉพาะของอุปกรณ์ตัวนั้นๆที่กำหนดมาจากผู้ผลิต ผู้ใช้งานไม่สามารถเปลี่ยนแปลงได้ สำหรับไอซี PCF8591 จะมีค่าเท่ากับ 1001(ฐานสอง) ข้อมูล 3 บิตถัดมาจะเป็นค่าของแอดเดรสผู้ใช้งานสามารถกำหนดได้ทางฮาร์ดแวร์เพื่อเลือกไอซี PCF8591 ที่ต้องการติดต่อด้วยในกรณีที่การต่อใช้งาน PCF8591 มากกว่า 1 ตัว ส่วนบิต LSB ใช้กำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับไอซีตัวนั้นๆ

ข้อมูลควบคุม

หลังจากส่งข้อมูลกำหนดแอดเดรสให้แก่ PCF8591 แล้ว ต้องส่งข้อมูลควบคุมตามไปด้วยเพื่อกำหนดคุณสมบัติของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล และวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอก ภายใน PCF8591 โดยมีรายละเอียดของข้อมูลในแต่ละบิต ดังรูปที่ 3.9

บิต 6 ของข้อมูลควบคุมใช้สำหรับการเอ็นเอเบิล ขาอะนาลอกเอาต์พุต เมื่อต้องการเอ็นเอเบิลต้องกำหนดให้ขานี้เป็น “1”

บิต 5 และบิต 4 ของข้อมูลใช้สำหรับการกำหนดรูปแบบของสัญญาณอะนาลอกอินพุตที่ป้อนให้แก่ PCF8591



รูปที่ 3.9 รายละเอียดข้อมูลควบคุมที่เขียนลงในรีจิสเตอร์ควบคุมภายในไอซี PCF8591

บิต 2 ใช้สำหรับเลือกรูปแบบการอ่านข้อมูลจากขาอินพุตอะนาลอกที่ต้องการอ่านจากเพียงอินพุตเดียวหรืออ่านแบบเรียงลำดับอินพุต ถ้าต้องการเลือกให้อ่านแบบเรียงลำดับต้องกำหนดให้บิตนี้เป็น “1”

บิต 1 และบิต 0 ใช้สำหรับกำหนดช่องของอินพุตอะนาลอกที่ต้องการอ่าน ถ้ากำหนดให้บิตสองเป็น “1” หลังจากอ่านค่าบิต 0 และบิต 1 แล้ว ในการอ่านครั้งต่อไปเป็นการอ่านอินพุตช่องที่ 1

ข้อมูลควบคุมทั้งหมดจะถูกเก็บไว้ในรีจีสเตอร์ควบคุมภายใน PCF8591

เมื่อจ่ายไฟให้แก่ PCF8591 ครั้งแรกบิตต่างๆของข้อมูลภายในรีจีสเตอร์ควบคุมจะเป็น “0”

ออสซิลเลเตอร์

วงจรออสซิลเลเตอร์ภายใน PCF8591 จะสร้างสัญญาณนาฬิกา สำหรับการแปลงสัญญาณอะนาลอกเป็นดิจิตอล เมื่อต้องการใช้วงจรออสซิลเลเตอร์ภายในขา EXT ต้องต่อลงกราวด์ ถ้าต้องการใช้ออสซิลเลเตอร์จากภายนอก ขาEXT ต้องต่อเข้ากับไฟบวก และป้อนสัญญาณนาฬิกาเข้าที่ขา OSC ของ PCF8591 โดยความถี่ของสัญญาณนาฬิกาสูงสุดที่ป้อนให้กับออสซิลเลเตอร์เท่ากับ 1.25 MHz

การอ่านค่าข้อมูลอินพุตอะนาลอกของ PCF8591

มีลำดับขั้นตอนดังนี้

1. เตรียมข้อมูลกำหนดแอดเดรส โดยในที่นี้กำหนดแอดเดรสของ PCF8591 ไว้ที่ 000 (ขา A0, A1, A2 ต่อลงกราวด์ทั้งหมด) และให้ทำงานในโหมดเขียนข้อมูล (ป้อนข้อมูลลอจิก “0” ให้แก่บิต R/W)
2. เรียกโปรแกรมย่อยการติดต่อกับอุปกรณ์สเลฟ
3. ส่งข้อมูลควบคุมไปยัง PCF8591
4. ส่งสัญญาณ STOP
5. เรียกโปรแกรมย่อยการติดต่อกับอุปกรณ์สเลฟ
6. ส่งข้อมูลกำหนดแอดเดรสอีกครั้งโดยครั้งนี้กำหนดให้เป็นโหมดอ่านข้อมูล (ส่งลอจิก “1” ให้แก่บิต R/W) เพื่อเริ่มต้นอ่านค่าข้อมูลจากช่องสัญญาณอะนาลอกอินพุต
7. อ่านค่าจากขาอินพุตของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลช่องที่ 1
8. หากต้องการอ่านค่าในช่องต่อไปก็ให้เริ่มต้นการติดต่อใหม่ ดังนั้นในการเขียนโปรแกรมเพื่ออ่านค่าต่อเนื่องทั้ง 4 ช่อง หรือมากกว่าจึงต้องเขียนโปรแกรมลูปเพื่อกำหนดรอบการทำงาน 4 รอบ หรือมากกว่า ก็จะสามารถอ่านค่าได้ครบทุกช่อง

การเขียนข้อมูลไปยังวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอกของ PCF8591

การเขียนข้อมูลไปยังขาอะนาลอกเอาต์พุตมีข้อแตกต่างจากการอ่านข้อมูลดังนี้

1. เรียกโปรแกรมย่อยการติดต่อกับอุปกรณ์สเลฟ
2. ส่งข้อมูลกำหนดแอดเดรสโดยให้ทำงานในโหมดเขียนข้อมูล (บิต R/W เป็นลอจิก “0”)
3. ส่งข้อมูลควบคุม 40H ไปยัง PCF8591 เพื่อเอ็นเอเบิลอะนาลอกเอาต์พุต
4. ส่งข้อมูลไปยังเอาต์พุตอะนาลอก โดยค่าที่ส่งออกไปจะต้องมีค่าอยู่ระหว่าง 0-255
5. ส่งสภาวะหยุด

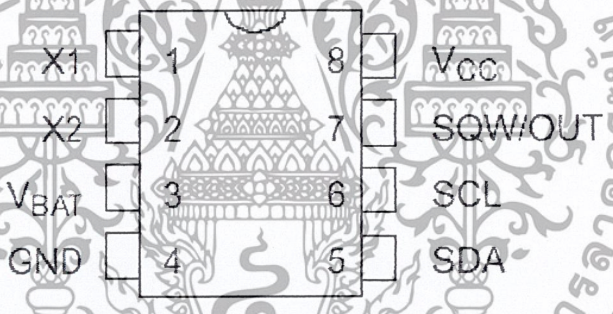
การเชื่อมต่อ PCF8591 กับไมโครคอนโทรลเลอร์ MCS-51

มีตัวอย่างวงจรแสดงในรูปที่ 4.7 จะเห็นได้ว่ามีลักษณะการต่อเหมือนกับ PCF8574A ทุกประการและสามารถที่จะต่อ ไอซี ทั้ง 2 เบอร์ร่วมกันบนสาย SDA และ SCL ได้ ตรงนี้เองที่แสดงให้เห็นถึงความสามารถของบัส I²C ผู้ใช้งานสามารถเชื่อมต่ออุปกรณ์ที่มีความต่างกันในหน้าที่การทำงานบนสายสัญญาณเดียวกันได้

3.5 ไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก (RTC) เบอร์ DS1307

เป็นไอซีสร้างฐานเวลาหรือรีลไทม์คล็อกมีหน้าที่สร้างฐานเวลาให้กับไมโครคอนโทรลเลอร์ โดย DS1307 จะให้ข้อมูลเกี่ยวกับเวลาทั้งหมด ไม่ว่าจะเป็นค่าของเวลาที่ละเอียดถึงหลักวินาที, นาที, ชั่วโมง, วันที่(date), เดือน และปี โดยสามารถปรับวันเดือนปีให้ตรงตามปฏิทินได้อย่างถูกต้อง รวมถึงการกำหนดวันในปีอธิกสุรทินด้วย คุณสมบัติทางเทคนิคที่สำคัญดังนี้

- เป็นไอซีรีลไทม์คล็อก ให้ข้อมูลตั้งแต่วินาทีจนถึงปี รวมถึงการปรับวันในปีอธิกสุรทินด้วย สามารถให้ข้อมูลเวลาได้อย่างเที่ยงตรงถึงคริสตศักราช 2100
- มีหน่วยความจำจำนวนโวลไทม์แรม 56 ไบต์อยู่ภายในสามารถใช้เก็บข้อมูลทั่วไปได้
- ใช้การเชื่อมต่อแบบระบบบัส I^2C
- มีวงจรตรวจจับไฟเลี้ยงต่ำหรือหายไปอย่างอัตโนมัติ และสามารถรักษาข้อมูลเวลาไว้ได้แม้ไม่มีไฟเลี้ยงไอซี



รูปที่ 3.10 การจัดขาของไอซี DS 1307 ไอซีสร้างฐานเวลาจริง (RTC)

รายละเอียดขาการต่อใช้งานของ DS1307

ในรูปที่ 3.10 แสดงการจัดขาของ DS1307 แต่ละขามีหน้าที่และการใช้งานดังนี้

- V_{CC}, **GND** (ขา 8 และ4) ต่อกับไฟเลี้ยง +5 V

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

- **SDA,SCL** (ขา 5 และ6) เป็นขาสำหรับเชื่อมต่อกับไมโครคอนโทรลเลอร์ บนบัส I^2C

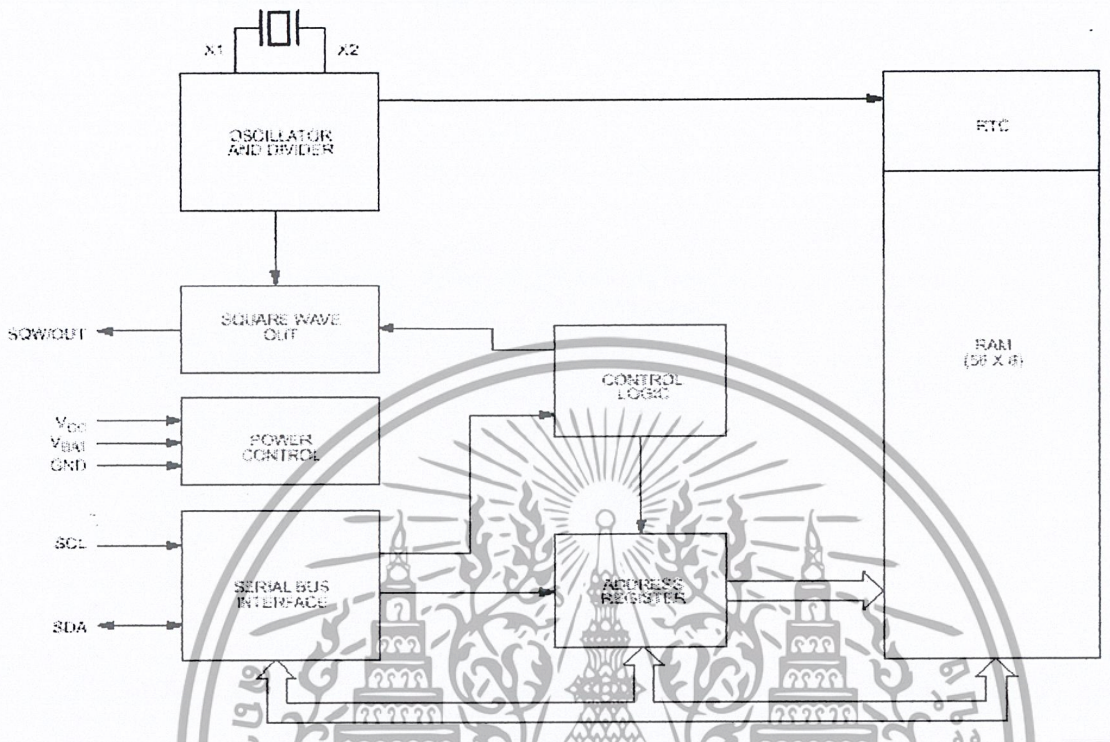
- **SQW/OUT** (ขา 7) ที่ขานี้จะมีสัญญาณรูปสี่เหลี่ยมส่งออกมา โดยสามารถเลือกความถี่ได้ 1Hz , 4.096 kHz , 8.192 kHz และ 32 kHz ในการใช้งานต้องต่อตัวต้านทาน 1k พลูอัปที่ขานี้ด้วย
- **X1,X2** (ขา 1 และ 2) ใช้ต่อคริสตอลความถี่มาตรฐาน 32.768k Hz เพื่อใช้เป็นฐานเวลาในการสร้างค่าเวลาจริง ในการใช้งานต้องต่อคริสตอลเข้ากับขาทั้งสองนี้ และที่แต่ละขาต้องต่อตัวเก็บประจุค่าต่ำๆ ประมาณ 15 pF คร่อมกับขากราวด์ด้วย

การทำงานของDS1307

ไอซี DS1307 จัดการเชื่อมต่อในแบบบัส I^2C โดยจะทำงานเป็นอุปกรณ์สเลฟเสมอ ดังนั้นการติดต่อเพื่อใช้งานจึงต้อง กำหนดรูปแบบตามที่กำหนดไว้ในการติดต่อแบบ I^2C (มีรายละเอียดเพิ่มเติมในการทดลองที่ 5) ในรูปที่ 3.11 แสดงส่วนประกอบหลักที่สำคัญ และไดอะแกรมการทำงานของ DS1307 วงจรออสซิลเลเตอร์ ถือเป็นหัวใจหลักของไอซี เนื่องจากเป็นจุดเริ่มต้นของการสร้างข้อมูลเวลาจริงในขณะที่ DS1307 ทำงานที่ขา SQW/OUT จะมีสัญญาณพัลส์สี่เหลี่ยมส่งออกมาตลอดเวลา ในกรณีที่มีการอีนาเบิลวงจรกำเนิดสัญญาณพัลส์ที่รีจิสเตอร์ควบคุม ค่าความถี่ของสัญญาณนี้สามารถเลือกได้ 4 ค่า คือ 1Hz, 4.096kHz, 8.192Khz และ 32kHz พร้อมกันนั้นก็มีการเก็บค่าของเวลาไว้ในหน่วยความจำนอนโวลาไทล์แรม ซึ่งมีขนาดรวม 64 ไบต์ แต่จัดสรรให้ใช้เก็บข้อมูลเวลา 8 ไบต์ และเป็นหน่วยความจำเก็บข้อมูลทั่วไปสำหรับผู้ใช้งานอีก 56 ไบต์

วงจรควบคุมพลังงานไฟฟ้าจะคอยตรวจสอบสภาวะของไฟเลี้ยงไอซี หากไฟเลี้ยงต่ำกว่า $1.25 \times V_{BAT}$ ก็จะควบคุมให้ DS1307 หยุดการทำงาน รีเซตค่าตัวนับแอดเดรสภายใน ทำให้ไม่สามารถติดต่อกับ DS1307 ได้ ดังนั้นในการใช้งาน DS1307 ต้องระมัดระวังอย่าให้ไฟเลี้ยงตกต่ำกว่า $1.25 \times V_{BAT}$ หรือ 3075 V ในกรณีที่ใช้ V_{BAT} เท่ากับ 3 ถ้าหากไฟเลี้ยงต่ำกว่า V_{BAT} ไอซี DS1307 จะเข้าสู่โหมดสำรองข้อมูลกระแสต่ำทันที จะไม่มีการส่งสัญญาณพัลส์ออกมาที่ขา SQW/OUT แต่วงจรสร้างฐานเวลายังคงทำงานเพื่อให้ค่าของเวลาเดินไปอย่างไม่ผิดพลาด เมื่อมีไฟเลี้ยงปรากฏขึ้นออกครั้ง DS1307 ก็จะสามารถให้ค่าของเวลาที่เป็นจริงแก่ผู้ใช้งานต่อไป

วงจรสื่อสารอนุกรมภายใน DS1307 ได้รับการกำหนดให้ทำงานตามรูปแบบของบัส I^2C เป็นช่องทางการสื่อสารระหว่าง DS1307 กับอุปกรณ์มาสเตอร์ ผู้ใช้งานสามารุเข้าถึงหน่วยความจำที่ใช้เก็บค่าเวลา และหน่วยความจำใช้งานทั่วไปได้โดยการเขียนข้อมูลตามรูปแบบที่กำหนดในระบบบัส I^2C



รูปที่ 3.11 โครงสร้างภายในของไอซีรีลไทม์คล็อกเบอร์ DS1307

การจัดสรรหน่วยความจำใน DS1307

ในรูปที่ 3.12 แสดงการจัดสรรพื้นที่ของหน่วยความจำภายใน DS1307 พื้นที่ 7 ไบต์แรกตั้งแต่แอดเดรส 00H – 06H เป็นพื้นที่ของรีจิสเตอร์ค่าเวลาใช้ในการเก็บข้อมูลเกี่ยวกับเวลา ไบต์ต่อมาที่แอดเดรส 07H เป็นพื้นที่ของรีจิสเตอร์ควบคุมการทำงานของ DS1307 ในรูปที่ 3.12 แสดงรายละเอียดของรีจิสเตอร์ค่าเวลาและรีจิสเตอร์ควบคุมของ DS1307

ด้วยการจัดสรรพื้นที่แบบนี้ ทำให้ผู้ใช้งานสามารถเรียกข้อมูลเวลาออกมาได้ตามที่ต้องการ โดยไม่จำเป็นต้องอ่านออกมาทั้งหมดก็ได้ ค่าของเวลาทั้งหมดจะอยู่ในรูปของเลขฐานสิบ สำหรับการแสดงเวลาในรูปของชั่วโมงสามารถเลือกได้ว่าต้องการแบบ 12 หรือ 24 ชั่วโมง โดยกำหนดที่บิต 6 ของแอดเดรส 02H และเมื่อเลือกแบบ 12 ชั่วโมง ที่บิต 5 ในแอดเดรสเดียวกันจะใช้ในการแสดงค่า AM/PM โดยถ้าบิตนี้เป็น “1” หมายถึง ค่าชั่วโมงในขณะนี้เป็นช่วงเวลาหลังเที่ยงวัน ในกรณีที่เป็นแบบ 24 ชั่วโมง บิตนี้จะใช้ในการแสดงค่า 2 ของหลักสิบในหน่วยชั่วโมง

รีจิสเตอร์ควบคุม

มีแอดเดรสอยู่ที่ 07H มีรายละเอียดของแต่ละบิตดังนี้

OUT (Output control) : ใช้ในการควบคุมระดับลอจิกที่ขา SQW OUT ในกรณีที่ดิสเอเบิลการกำเนิดสัญญาณสี่เหลี่ยม โดยถ้าบิตนี้เป็น "1" ที่ขา SQW OUT ก็จะเป็น "1" ถ้าบิตนี้เป็น "0" ที่ขา SQW OUT ก็จะเป็น "0"

00H	วินาที									
	นาที									
	ชั่วโมง	บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0	ค่าของข้อมูล
		CH	ข้อมูลวินาที(หลักสิบ)			ข้อมูลวินาที(หลักหน่วย)				00-59
	วัน	X	ข้อมูลนาที(หลักสิบ)			ข้อมูลนาที(หลักหน่วย)				00-59
	เดือน	X	12 ชั่วโมง	ชั่วโมงผหลักสิบ)	ข้อมูลชั่วโมง (หลักสิบ)	ข้อมูลชั่วโมง(หลักหน่วย)				01-12
			24 ชั่วโมง	AM/PM						00-23
	ปี	X	X	X	X	X	ข้อมูลวันในสัปดาห์			1-7
07H	รีจิสเตอร์ควบคุม	X	X	ข้อมูลวันที่(หลักสิบ)		ข้อมูลวันที่(หลักหน่วย)				01-28/29 01-30 01-31
08H	แรม 56 ไบต์	X	X	X	ข้อมูลเดือน (หลักสิบ)	ข้อมูลเดือน(หลักหน่วย)				01-12
		ข้อมูลปี (หลักสิบ)				ข้อมูลปี(หลักหน่วย)				00-99
3FH		OUT	X	X	SQWE	X	X	RS1	RS0	

รูปที่ 3.12 การจัดหน่วยความจำแรมภายใน DS1307 และรายละเอียดของรีจิสเตอร์เก็บค่าของเวลาและรีจิสเตอร์ควบคุมของ DS1307

SQWE (Square Wave Enable) : ใช้ในการเอ็นเอเบิลวงจรกำเนิดสัญญาณสี่เหลี่ยมที่ขา SQW OUT ถ้าต้องการให้มีสัญญาณสี่เหลี่ยมออกให้กำหนดบิตนี้เป็น "1"

RS1, RS0 (Rate Select) : ใช้ในการเลือกความถี่ของสัญญาณสี่เหลี่ยมที่ออกจากขา SQW OUT ดังมีรายละเอียดดังนี้

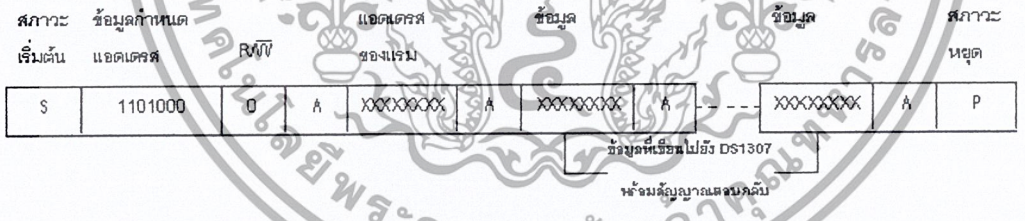
RS1	RS0	ค่าความถี่ของสัญญาณสี่เหลี่ยม
0	0	1Hz
0	1	4.096kHz
1	0	8.192kHz
1	1	32.768kHz

โหมดการทำงานของ DS1307

มีด้วยกัน 2 โหมดคือ โหมดเขียนข้อมูลและโหมดอ่านข้อมูล ในการใช้งาน DS1307 ตามปกติจะใช้งานเฉพาะโหมดอ่านข้อมูลเท่านั้น เนื่องจากไมโครคอนโทรลเลอร์จะติดต่อกับ DS1307 เพื่ออ่านข้อมูลของเวลาไปใช้งาน โหมดการเขียนข้อมูลจะถูกใช้งานก็ต่อเมื่อต้องการตั้งค่าเวลาใหม่และต้องการเขียนข้อมูลลงในหน่วยความจำใช้งานทั่วไป อย่างไรก็ตามเมื่อเริ่มต้นติดต่อกับ DS1307 จำเป็นอย่างยิ่งที่จะต้องเข้าสู่โหมดการเขียนข้อมูลก่อนเพื่อกำหนดแอดเดรสที่ต้องการอ่านข้อมูล จากนั้นจึงเปลี่ยนโหมดทำงานมาเป็นโหมดการอ่านข้อมูล

โหมดการเขียนข้อมูล

มีรูปแบบดังในรูปที่ 3.13 เริ่มต้นเมื่อไมโครคอนโทรลเลอร์ทำการกำหนดสภาวะเริ่มต้น (START : S) จากนั้นส่งข้อมูลกำหนดแอดเดรส 1101000 ตามด้วยข้อมูลเลือกการเขียน นั่นคือค่า 0 จากนั้นจะรอการตอบรับจาก DS1307 ขั้นตอนต่อมาคือ ส่งข้อมูลเพื่อเลือกแอดเดรสที่ต้องการเขียน จากนั้นรอการตอบรับจาก DS1307 เมื่อมีการตอบรับมาเรียบร้อย ก็เริ่มทยอยเขียนข้อมูลลงไปครั้งละแอดเดรส หลังจากเขียนข้อมูลในแต่ละแอดเดรส จะต้องหยุดรอการตอบรับจาก DS1307 ทุกครั้ง จึงจะสามารถเขียนข้อมูลต่อไปได้ เมื่อเขียนเรียบร้อยแล้วให้ส่งสภาวะหยุด (STOP : P) เป็นอันสิ้นสุดกระบวนการเขียนข้อมูล

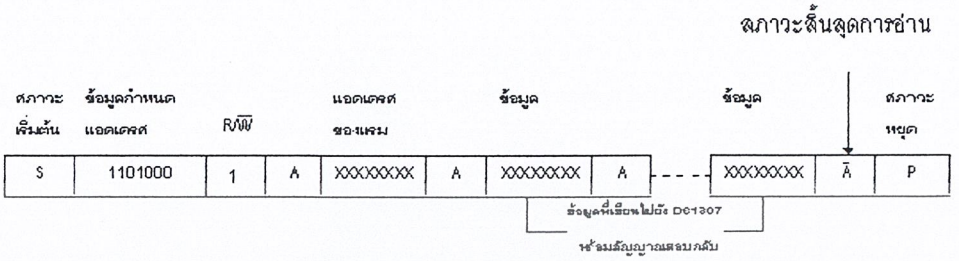


รูปที่ 3.13 รูปแบบของข้อมูลสำหรับติดต่อกับ DS 1307 ในโหมดการเขียนข้อมูล

โหมดการอ่านข้อมูล

มีรูปแบบแสดงในรูปที่ 3.14 เริ่มต้นการทำงานเหมือนกับโหมดการเขียนข้อมูลคือ ไมโครคอนโทรลเลอร์กำหนดสภาวะเริ่มต้นแล้วส่งข้อมูลกำหนดแอดเดรส ตามด้วยข้อมูลเลือกการอ่าน ซึ่งเท่ากับ 1 จากนั้นรอการตอบรับจาก DS1307 เมื่อตอบรับเรียบร้อย DS1307 จะทยอยส่งข้อมูลออกมาให้ไมโครคอนโทรลเลอร์คราวละ 1 แอดเดรสหรือ 1 ไบต์ โดยแอดเดรสที่เลือกอ่านข้อมูลจะต้องมีการกำหนดมาก่อนล่วงหน้าด้วยโหมดการเขียนข้อมูลวิธีง่าย ๆ คือ เข้าสู่โหมดการเขียนข้อมูลก่อน เมื่อถึงจังหวะที่ต้องเขียนข้อมูล ให้ทำการสร้างสภาวะเริ่มต้นและส่งข้อมูล

กำหนดแอดเดรสใหม่อีกครั้ง ตามด้วยเลือกโหมดการอ่านข้อมูล ข้อมูลที่ออกมาจาก DS1307 ก็จะเป็นข้อมูลจากแอดเดรสที่กำหนดไว้ก่อนหน้านี้



รูปที่ 3.14 รูปแบบของข้อมูลสำหรับติดต่อกับ DS 1307 ในโหมดการอ่านข้อมูล

การเชื่อมต่อ DS13071 กับไมโครคอนโทรลเลอร์ MCS – 51

มีตัวอย่างวงจรแสดงในรูปที่ 4.5 จะเห็นได้ว่ามีลักษณะการต่อเหมือนกับอุปกรณ์บนระบบบัส I^2C ตัวอื่น ๆ ทุกประการ และสามารถที่จะต่อไอซีทั้งหมดร่วมกันบนสาย SDA และ SCL ได้ เป็นการย้ำให้เห็นถึงความสามารถพิเศษของระบบบัส I^2C ที่ผู้ใช้งานสามารถเชื่อมต่ออุปกรณ์ที่มีความต่างกันในหน้าที่การทำงานบนสายสัญญาณเดียวกันได้ ถึงการทดลองนี้ผู้ใช้งานสามารถเชื่อมต่ออุปกรณ์ระบบบัส I^2C ได้ถึง 3 ตัว 3 ลักษณะการทำงาน โดยใช้สายสัญญาณเพียง 2 เส้น

จากวงจรในรูปที่ 4.5 ไอซี DS1307 จำเป็นจะต้องต่อแบตเตอรี่ไว้ตลอดเวลาไม่ว่าจะใช้งานหรือไม่ทั้งนี้เพื่อรักษาการทำงานของวงจรภายใน DS1307 ให้ยังคงทำงานต่อเนื่องไป เมื่อใดที่ไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูล ก็จะได้ข้อมูลเวลาที่เป็นจริงตลอดเวลา

โปรแกรมย่อยเพื่อติดต่อกับ DS1307

เช่นเดียวกับอุปกรณ์บนระบบบัส I^2C ตัวอื่น ๆ ต้องมีการกำหนดสภาวะเริ่มต้น แต่สำหรับ DS1307 อาจมีรูปแบบที่แตกต่างกับ PCF8574 และ PCF8591 อยู่บ้าง จึงขอแยกการเขียนโปรแกรมเพื่อใช้งาน DS1307 ออกเป็นโปรแกรมย่อย 2 โปรแกรมคือ โปรแกรมย่อยการอ่านข้อมูลจาก DS1307 และโปรแกรมย่อยการเขียนข้อมูลไปยัง DS1307 ดังมีโฟลวชาร์ตและรายละเอียดโปรแกรมแสดงในรูปที่ 4.10 และ 4.11

โปรแกรมย่อยการนำข้อมูลเลขฐานสิบไปแสดงผลที่โมดูล LCD

ในการทดลองนี้ยังมีโปรแกรมย่อยที่น่าสนใจอีกหนึ่งโปรแกรมคือ โปรแกรมย่อยการนำข้อมูลเลขฐานสิบที่อ่านได้จากไอซี DS1307 ไปแสดงผลที่โมดูล LCD เนื่องจากโมดูล LCD ต้องการข้อมูลรหัสแอสกี้ในการแสดงผล โปรแกรมย่อยนี้จึงทำหน้าที่ในการแปลงค่าเลขฐานสิบให้กลายเป็นรหัสแอสกี้เพื่อส่งให้โมดูล LCD ทำการแสดงผลต่อไป ดังมีรายละเอียดของโปรแกรมย่อยดังต่อไปนี้

```
BCK2LCD:  PUSH    ACC            ; Push ACC.  To  Stack
          PUSH    B              ; Push B to Stack
          MOV     A,LCD _ DATA   ; Get input data value
          MOV     B,A            ; Copy to B
          ANL     A,# 11110000B  ; Get higher 4 bit
          SWAP    A              ; Swap nibble
          ADD     A,#030H        ; Convert to ASCII
          MOV     LCD_DATA,A     ; Write LCD
          ACALL   WRCHAR_LCD     ;
          MOV     A,B            ;Restore value
          ANL     A,#00001111B   ;Get lower 4 bit
          ADD     A,#030H        ; Convert to ASCII
          MOV     LCD_DATA,A     ; Write LCD
          ACALL   WRCHAR_LCD     ;
          POP     B              ;Pop B from Stack
          POP     ACC            ; Pop ACC.  From Stack
          RET                    ; Return
```

การทำงานเริ่มจากเก็บข้อมูลเดิมของแอกคิวมูเลเตอร์และรีจิสเตอร์ B ไปไว้ในสแต็ก จากนั้นนำข้อมูลเลขฐานสิบที่เก็บอยู่ใน LCD_DATA มาเก็บไว้ในแอกคิวมูเลเตอร์ พร้อมทั้งคัดลอกไว้ที่รีจิสเตอร์ B ด้วย จากนั้นทำการแอนด์ 4 บิตล่างด้วย 0000H เพื่อให้เหลือเฉพาะข้อมูล 4 บิตบน โดยข้อมูล 4 บิตบนนี้จะกลายเป็นตัวเลขหลักสิบเมื่อนำไปแสดงผลบนโมดูล LCD จากนั้นทำการสลับข้อมูล 4 บิตบนและ 4 บิตล่างด้วยคำสั่ง AWAP A แล้วบวกเข้ากับ 30H เพื่อให้ได้ค่าข้อมูลรหัสแอสกี้ของตัวเลขฐานสิบที่หลักสิบ ก่อนส่งไปยังโปรแกรมย่อยการแสดงผลของโมดูล LCD สำหรับตัวเลขหลักหน่วยก็ใช้วิธีการเดียวกัน แต่จะไม่มีการสลับค่า 4 บิตบนและล่าง

ยกตัวอย่าง หากข้อมูลที่ต้องการแปลงมีค่า 40 ด้วยการใช้โปรแกรมย่อยข้างต้น จะได้ค่าข้อมูลรหัสแอสกี้ของเลข 4 เป็น 34H และเลข 0 เป็น 30H ส่งไปให้โมดูล LCD เพื่อแสดงผลเป็น 40 ต่อไป

บทที่ 4

การทดลองและผลการทดลอง

การทดลองที่ 1

การแสดงผลที่โมดูลแสดงผลแบบผลึกเหลว(LCD)โดยผ่านบัส I^2C

จุดประสงค์

1. สามารถเขียนโปรแกรมเพื่อนำข้อมูลออกแสดงผลที่โมดูลแสดงผลแบบผลึกเหลว หรือ LCD ได้
2. เพื่อทดลองการส่งข้อมูลควบคุมและข้อมูลแสดงผลบนบัส I^2C และสามารถนำข้อมูลที่ได้ออกแสดงผล

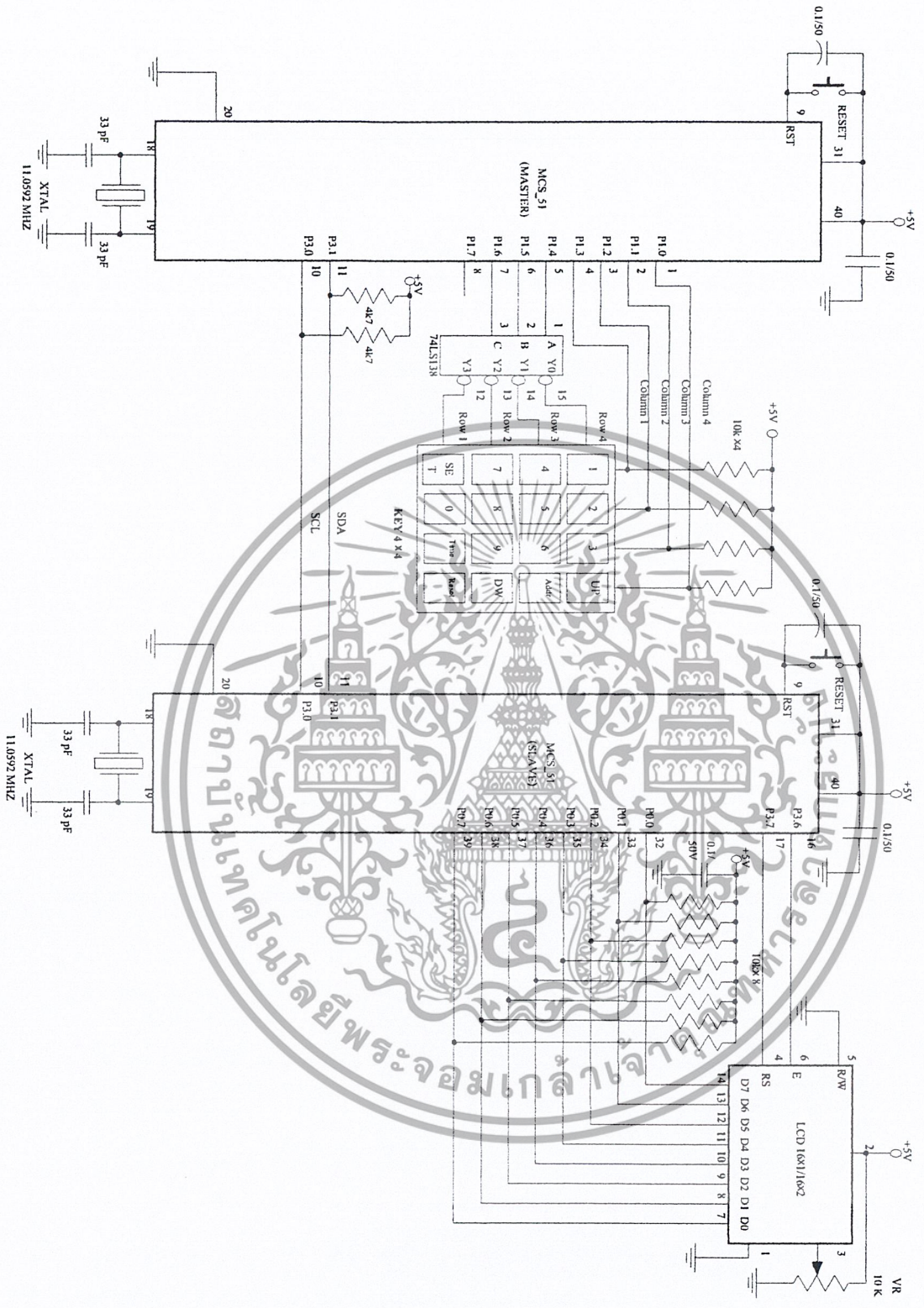
เครื่องมือและอุปกรณ์

1. เครื่องโปรแกรมไมโครคอนโทรลเลอร์ พร้อมสายต่อคอมพิวเตอร์
2. บอร์ดไมโครคอนโทรลเลอร์ (ตัวมาสเตอร์) กับบอร์ดไมโครคอนโทรลเลอร์ (อุปกรณ์สเลฟ)
3. สายต่อบัส I^2C
4. โมดูล LCD ขนาด 16 ตัวอักษร 1 บรรทัด

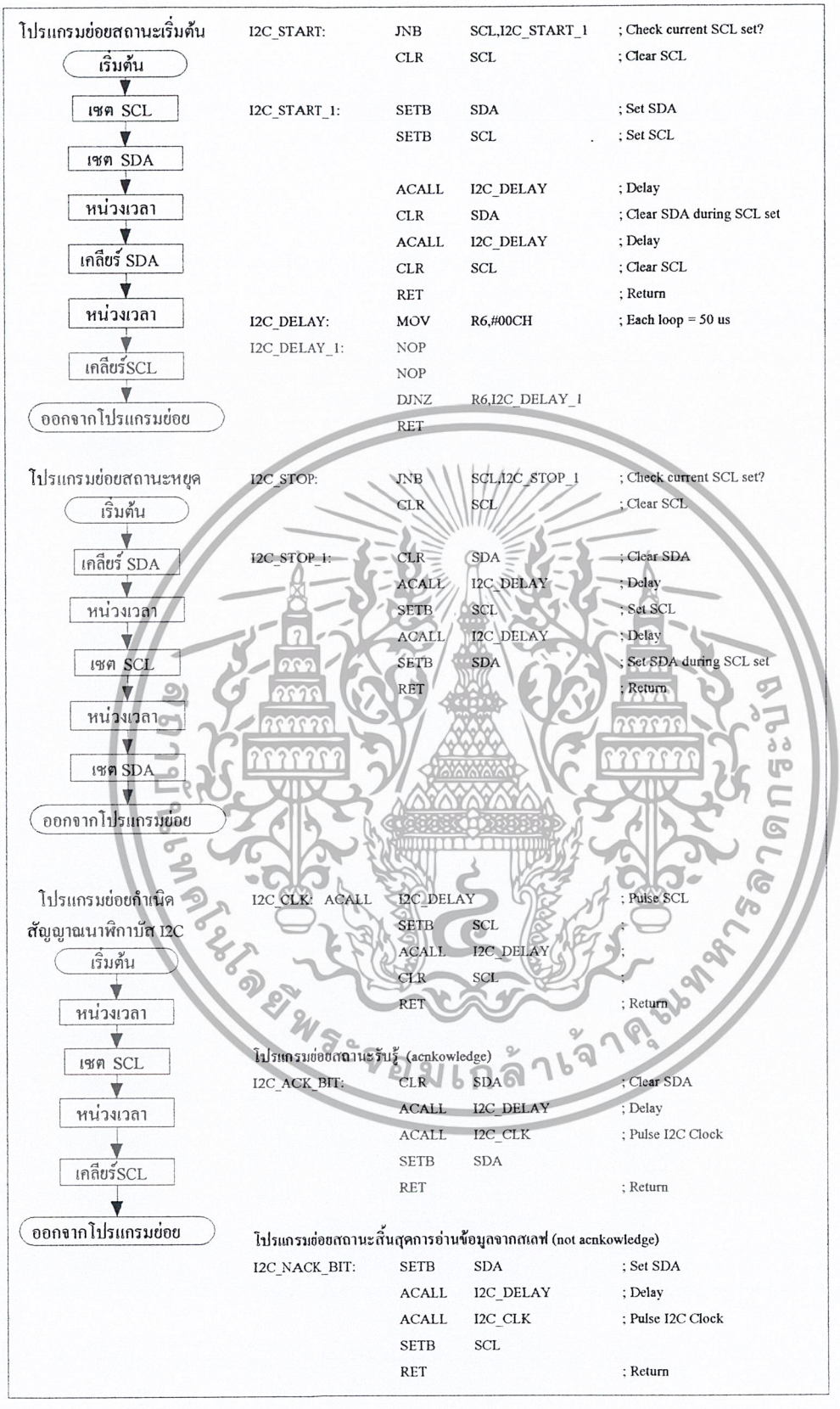
การทดลอง

การทดลองเพื่อแสดงข้อความบนโมดูล LCD ขนาด 16 ตัวอักษร 1 บรรทัด

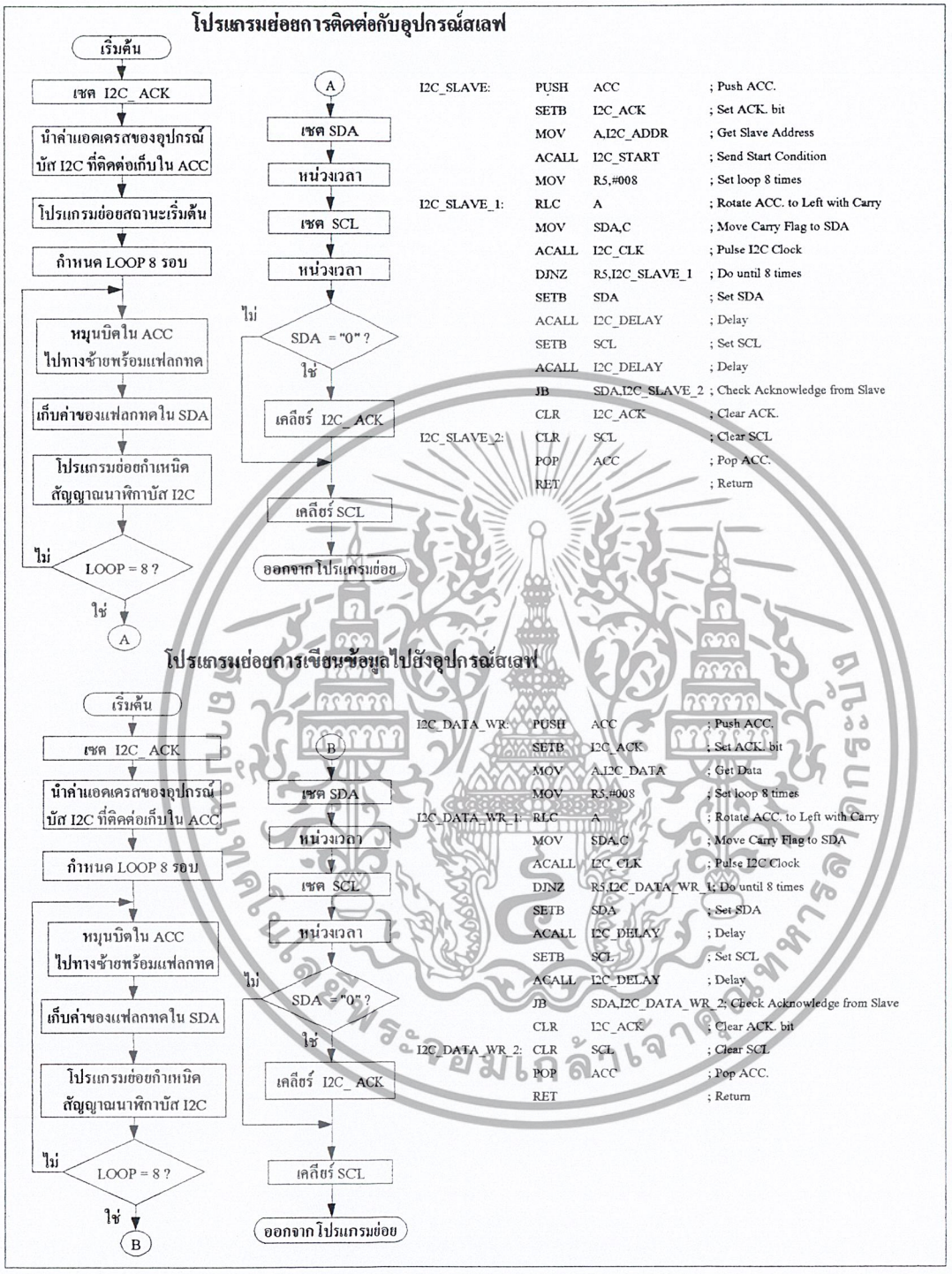
1. ศึกษาโฟลวชาร์ตในรูปที่ 4.2,4.3,4.4 โปรแกรมที่ 1 และโปรแกรมที่ 2 ร่วมกับวงจรที่ใช้ในการทดลองตามรูปที่ 4.1 ของวงจรการทดลองที่ 1
2. เขียนโปรแกรมที่ 1 และโปรแกรมที่ 2 ทำการแอสแซมเบลอร์แล้วทำการโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ทั้งสองซึ่ง โปรแกรมที่ 1 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวมาสเตอร์และโปรแกรมที่ 2 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวสเลฟ
3. ต่อวงจรตามรูปวงจรการทดลองที่ 1
4. จ่ายไฟให้แก่วงจร รันโปรแกรมโดยกดสวิตซ์ RESET
5. ป้อนคำส่งเข้าไปในคีย์บอร์ด (KEY)



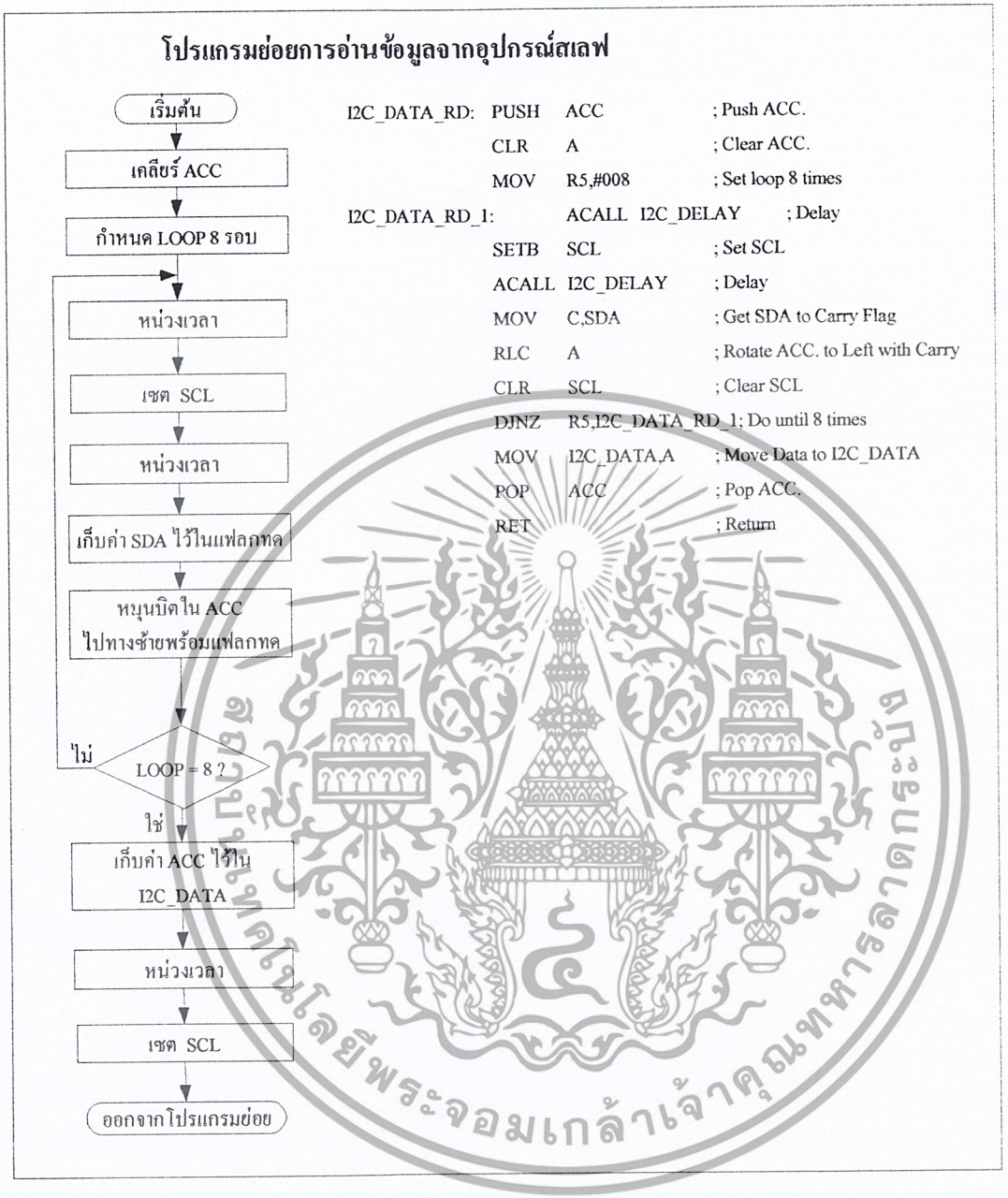
รูปที่ 4.1 แสดงวงจรการทดลองที่ 1



รูปที่ 4.2 แสดงโฟลวชาร์ตโปรแกรมย่อยสถานะเริ่มต้น สถานะหยุด และโปรแกรมย่อยกำเนิดสัญญาณนาฬิกาบัส I^2C



รูปที่ 4.3 แสดงโฟลวชาร์ตโปรแกรมย่อยการติต่ออุปกรณ์สเลฟและการเขียนข้อมูลไปยังอุปกรณ์สเลฟ



รูปที่ 4.4 แสดงโฟลวชาร์ตโปรแกรมย่อยการอ่านข้อมูลจากอุปกรณ์สเลฟ

การทดลองที่ 2

การขยายพอร์ตอินพุตเอาต์พุตด้วยไอซีขยายพอร์ตเบอร์ PCF 8574A บนระบบบัส I^2C

จุดประสงค์

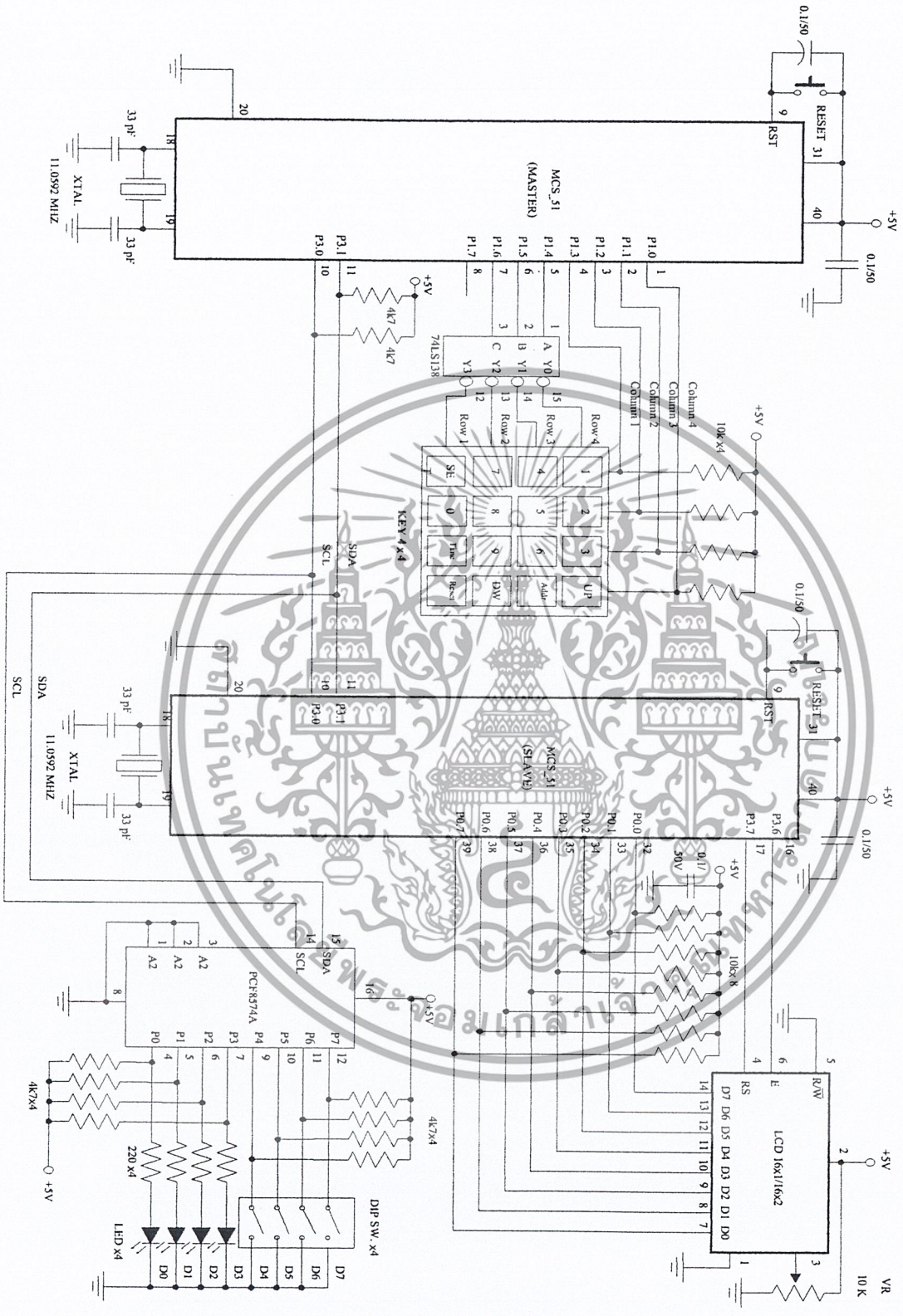
1. เพื่อศึกษาการรับส่งข้อมูลบนบัส I^2C โดยอุปกรณ์อินเตอร์เฟส PCF 8574A
2. เพื่อเป็นแนวทางการนำอุปกรณ์อินเตอร์เฟสมาใช้ในการขยายพอร์ตอินพุตเอาต์พุตเพิ่มเติมได้
3. สามารถเขียนโปรแกรมเพื่อติดต่อกับอุปกรณ์ที่ใช้การเชื่อมต่อบนระบบบัส I^2C ได้

เครื่องมือและอุปกรณ์

1. เครื่องโปรแกรมไมโครคอนโทรลเลอร์ พร้อมสายต่อคอมพิวเตอร์และอะแดปเตอร์ 1 ชุด
2. บอร์ดไมโครคอนโทรลเลอร์(ตัวมาสเตอร์) กับบอร์ดขยายพอร์ตอินพุตเอาต์พุต PCF 8574A (อุปกรณ์สเลฟ)
3. สายต่อบัส I^2C

การทดลอง

1. ศึกษาโฟลวชาร์ตในรูปที่ 4.2,4.3,4.4 โปรแกรมที่ 1 และโปรแกรมที่ 2 ร่วมกับวงจรที่ใช้ในการทดลองตามรูปที่ 4.5 ของวงจรการทดลองที่ 2
2. เขียนโปรแกรมที่ 1 และโปรแกรมที่ 2 ทำการแอสแซมเบลอร์แล้วทำการโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ทั้งสองชิ้น โปรแกรมที่ 1 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวมาสเตอร์ และโปรแกรมที่ 2 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวสเลฟ
3. ต่อวงจรตามรูปที่ 4.5
4. จ่ายไฟให้แก่วงจร รันโปรแกรมโดยกดสวิตซ์ RESET
5. ป้อนคำส่งเข้าไปในคีย์บอร์ด (KEY)
6. สังเกตการทำงานของ LED ที่ส่วนของบอร์ดขยายพอร์ตอินพุตเอาต์พุต PCF 8574A และ LCD ที่ตัวอุปกรณ์สเลฟ (LCD)



รูปที่ 4.5 แสดงวงจรการทดลองที่ 2

การทดลองที่ 3

การขยายพอร์ตอินพุตเอาต์พุตด้วยไอซีไมโครคอนโทรลเลอร์ MCS-51 บนระบบบัสI^2C

จุดประสงค์

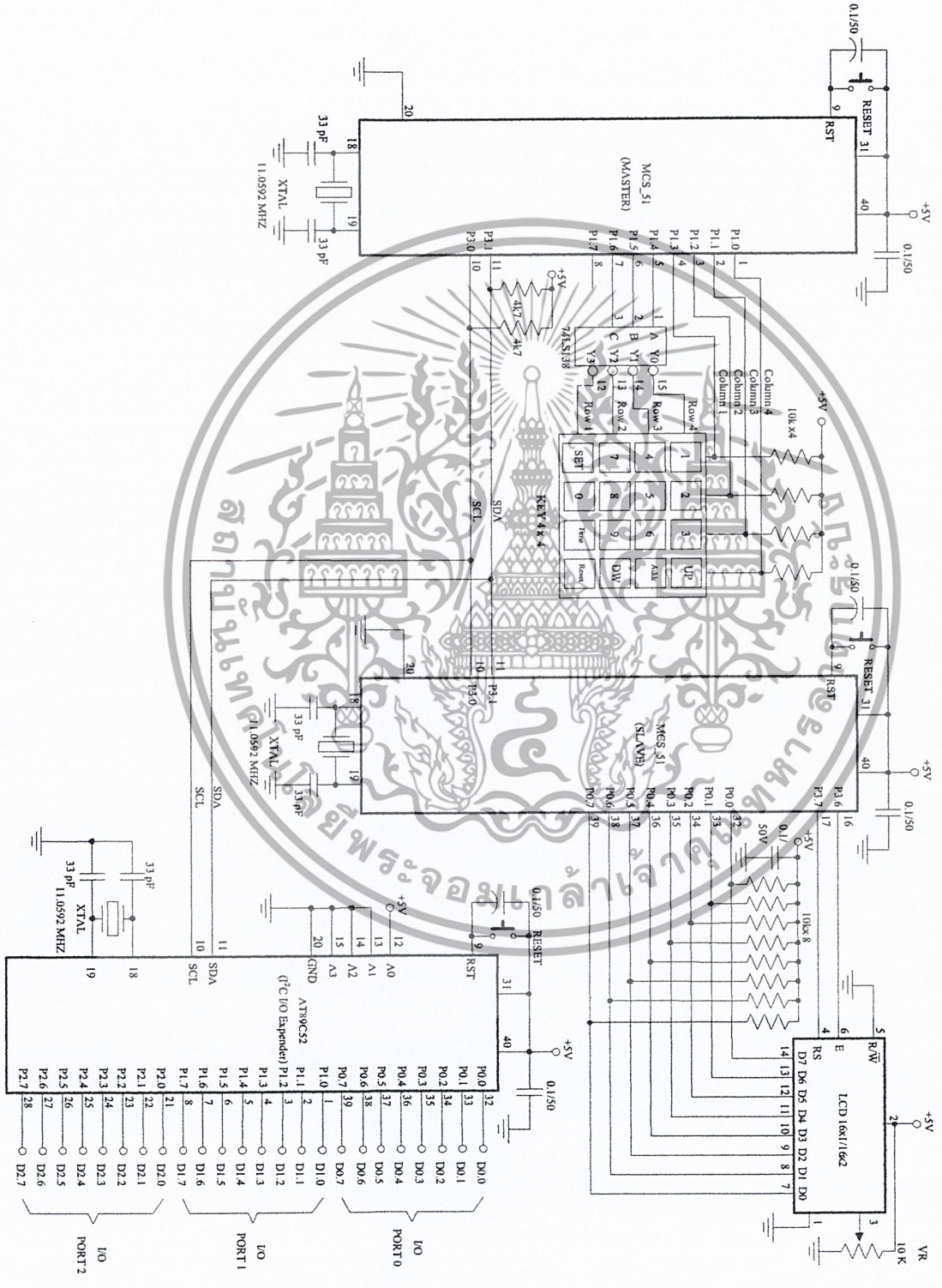
1. เพื่อศึกษาการรับส่งข้อมูลบนบัส I^2C โดยอุปกรณ์อินเตอร์เฟส ไมโครคอนโทรลเลอร์
2. เพื่อเป็นแนวทางการนำไมโครคอนโทรลเลอร์มาใช้ในการขยายพอร์ตอินพุตเอาต์พุตเพิ่มเติมและควบคุมงานได้
3. สามารถเขียนโปรแกรมเพื่อติดต่อกับอุปกรณ์ที่ใช้การเชื่อมต่อบนระบบบัส I^2C ได้

เครื่องมือและอุปกรณ์

1. เครื่องโปรแกรมไมโครคอนโทรลเลอร์ พร้อมสายต่อคอมพิวเตอร์และอะแดปเตอร์ 1 ชุด
2. บอร์ดไมโครคอนโทรลเลอร์ (ตัวมาสเตอร์) กับบอร์ด ขยายพอร์ตอินพุตเอาต์พุตไมโครคอนโทรลเลอร์ (อุปกรณ์สเลฟ)
3. สายต่อบัส I^2C

การทดลอง

1. ศึกษาโฟลวชาร์ตในรูปที่ 4.2,4.3,4.4 โปรแกรมที่ 1 และโปรแกรมที่ 2 ร่วมกับวงจรที่ใช้ในการทดลองตามรูปที่ 4.6 ของวงจรการทดลองที่ 3
2. เขียนโปรแกรมที่ 1 โปรแกรมที่ 2 และโปรแกรมที่ 3 ทำการแอสแซมเบลอร์แล้วทำการโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ทั้งสองซึ่ง โปรแกรมที่ 1 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวมาสเตอร์ โปรแกรมที่ 2 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวสเลฟ(LCD) และ โปรแกรมที่ 3 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวสเลฟที่ 2
3. ต่อวงจรตามรูปที่ 4.6
4. จ่ายไฟให้แก่วงจร รันโปรแกรมโดยกดสวิตซ์ RESET
5. ป้อนคำส่งเข้าไปในคีย์บอร์ด (KEY)
6. สังเกตการทำงานของ LED ที่ส่วนของบอร์ดขยายพอร์ตอินพุตเอาต์พุตไมโครคอนโทรลเลอร์ และตัวอุปกรณ์สเลฟ (LCD)



รูปที่ 4.6 วงจรการทดลองที่ 3

การทดลองที่ 4

การใช้งานไอซี ADC/DAC บนระบบบัส I^2C เบอร์ PCF 8591

จุดประสงค์

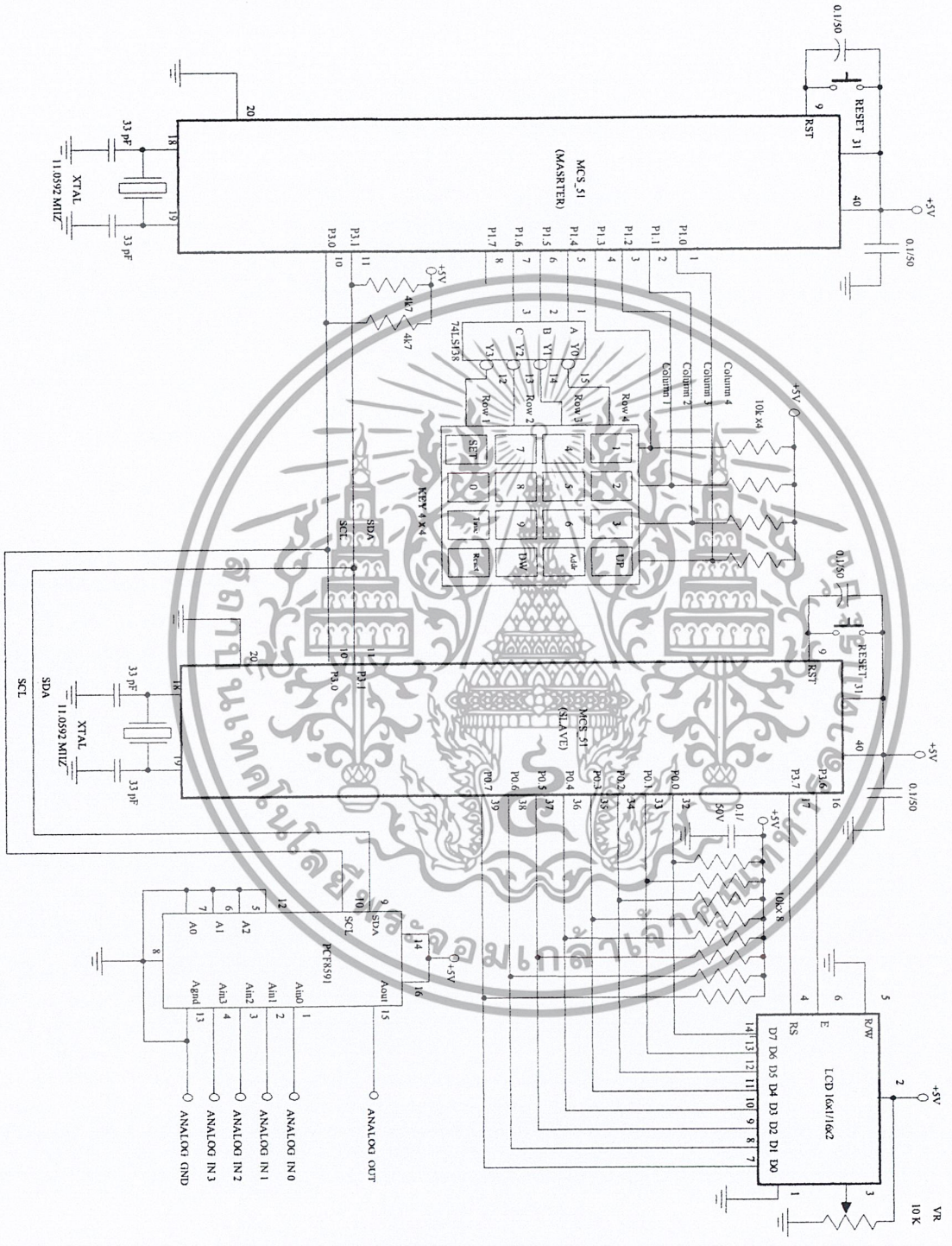
1. เพื่อศึกษาการรับส่งข้อมูลบนบัส I^2C โดยอุปกรณ์อินเตอร์เฟส ไอซี ADC/DAC บนระบบบัส I^2C เบอร์ PCF 8591(อุปกรณ์สเลฟ)
2. สามารถที่จะแปลงสัญญาณอะนาลอกเป็นดิจิตอลและแปลงดิจิตอลเป็นอะนาลอก บนระบบบัส I^2C ได้
3. สามารถเขียนโปรแกรม เพื่อติดต่อกับอุปกรณ์ที่ใช้การเชื่อมต่อบนระบบบัส I^2C (ซอฟแวร์) ได้

เครื่องมือและอุปกรณ์

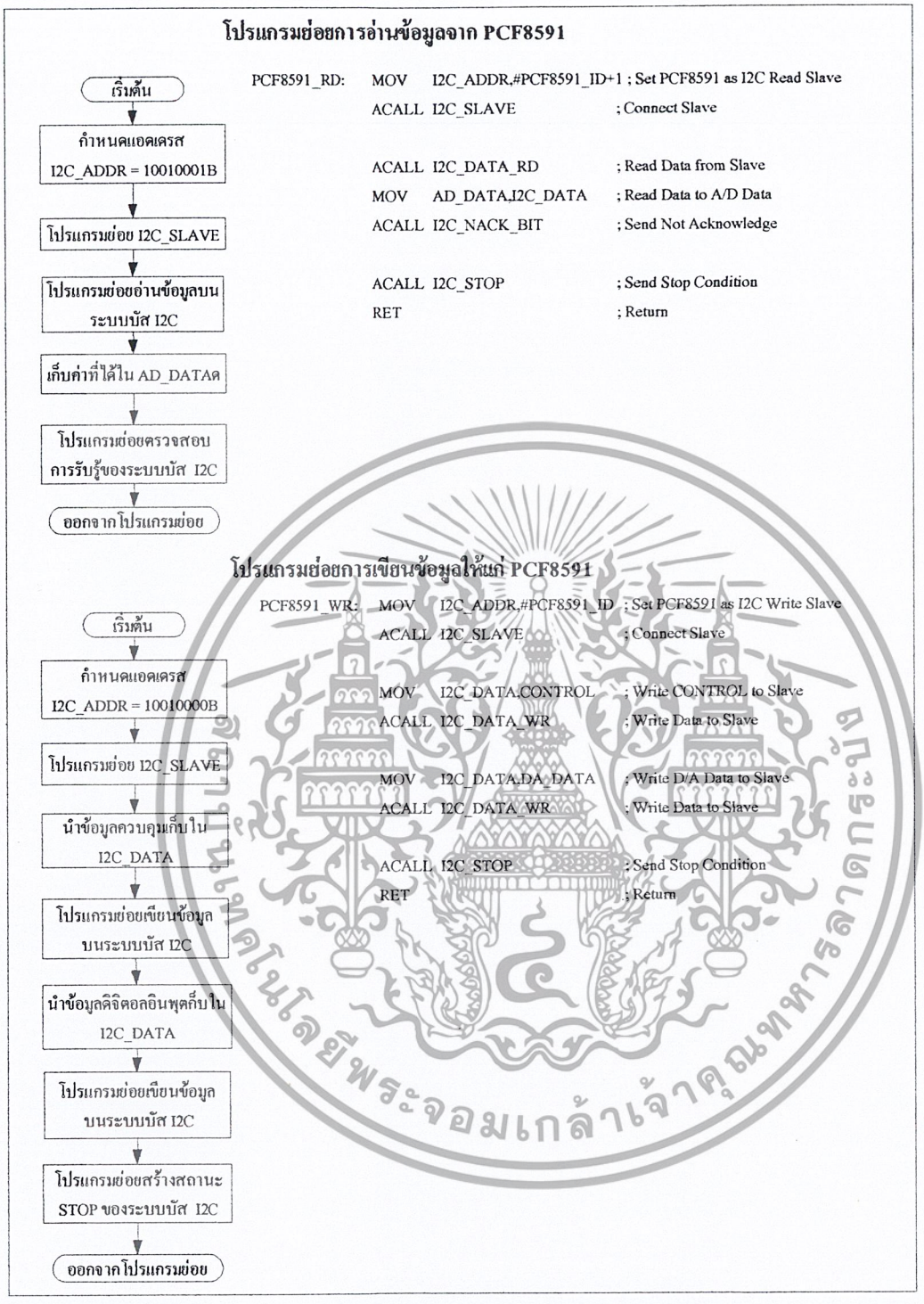
1. เครื่องโปรแกรมไมโครคอนโทรลเลอร์ พร้อมสายต่อคอมพิวเตอร์
2. บอร์ดไมโครคอนโทรลเลอร์ (ตัวมาสเตอร์) กับบอร์ด ADC/DAC ไอซีเบอร์ PCF 8591 (อุปกรณ์สเลฟ)
3. สายต่อบัส I^2C

การทดลอง

1. ศึกษาโฟลวชาร์ตในรูปที่ 4.8 โปรแกรมที่ 1 และโปรแกรมที่ 2 ร่วมกับวงจรที่ใช้ในการทดลองตามรูปที่ 4.7 ของวงจรการทดลองที่ 4
2. เขียนโปรแกรมที่ 1 และโปรแกรมที่ 2 ทำการแอสแซมเบลอร์แล้วทำการโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ทั้งสองซึ่ง โปรแกรมที่ 1 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวมาสเตอร์ และ โปรแกรมที่ 2 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวสเลฟ(LCD)
3. ต่อวงจรตามรูปที่ 4.7
4. จ่ายไฟให้แก่วงจร รันโปรแกรมโดยกดสวิตซ์ RESET
5. จ่ายสัญญาณอะนาล็อกเข้าไปในบอร์ด ADC/DAC ไอซีเบอร์PCF 8591(อุปกรณ์สเลฟ)
2. สังเกตการแสดงผลของ LCD ที่ส่วนของบอร์ดตัวอุปกรณ์สเลฟ (LCD)



รูปที่ 4.7 แสดงวงจรการทดลองที่ 4



รูปที่ 4.8 แสดงโฟลวชาร์ตโปรแกรมย่อยการอ่านข้อมูลจาก PCF 8591 และการเขียนให้แก่ PCF 8591

การทดลองที่ 5

การใช้งานไอซีสร้างฐานเวลาจริงหรือรีลไทม์คล็อก (RTC) เบอร์ DS 1307

จุดประสงค์

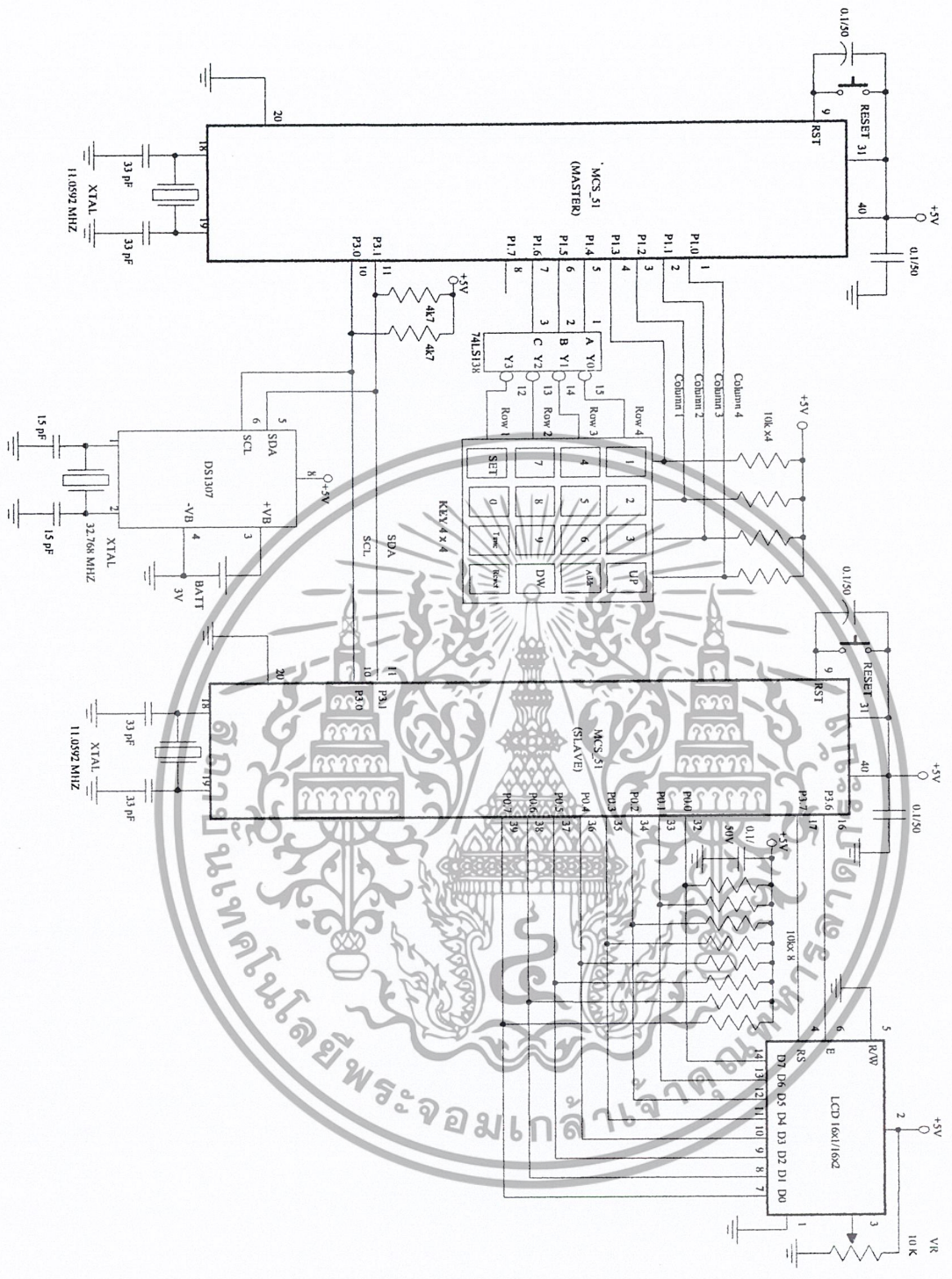
1. เพื่อศึกษาการรับส่งข้อมูลบนบัส I^2C โดยอุปกรณ์อินเตอร์เฟสรีลไทม์คล็อก (RTC) เบอร์ DS 1307
2. สามารถที่จะทำการควบคุม และอ่านค่าข้อมูลของเวลา RTC บนระบบบัส I^2C มาแสดงผลที่ตัวมาสเตอร์ได้ ได้
3. สามารถเขียนโปรแกรมเพื่อติดต่อกับอุปกรณ์ที่ใช้การเชื่อมต่อบนระบบบัส I^2C ได้

เครื่องมือและอุปกรณ์

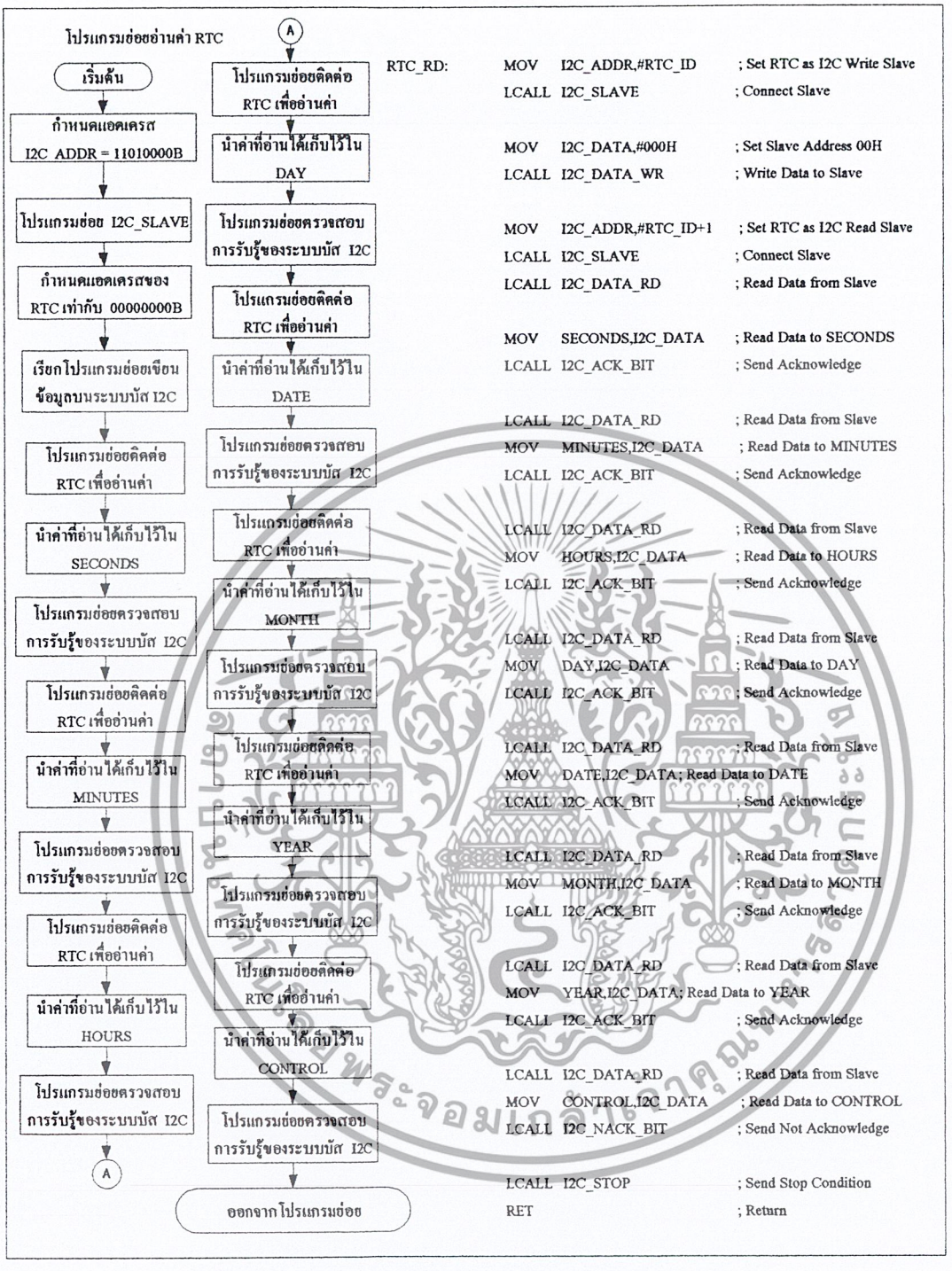
1. เครื่องโปรแกรมไมโครคอนโทรลเลอร์ พร้อมสายต่อคอมพิวเตอร์ และอะแดปเตอร์ 1 ชุด
2. บอร์ดไมโครคอนโทรลเลอร์(ตัวมาสเตอร์)กับบอร์ดรีลไทม์คล็อก (RTC) เบอร์ DS1307 (อุปกรณ์สเลฟ)
3. สายต่อบัส I^2C

การทดลอง

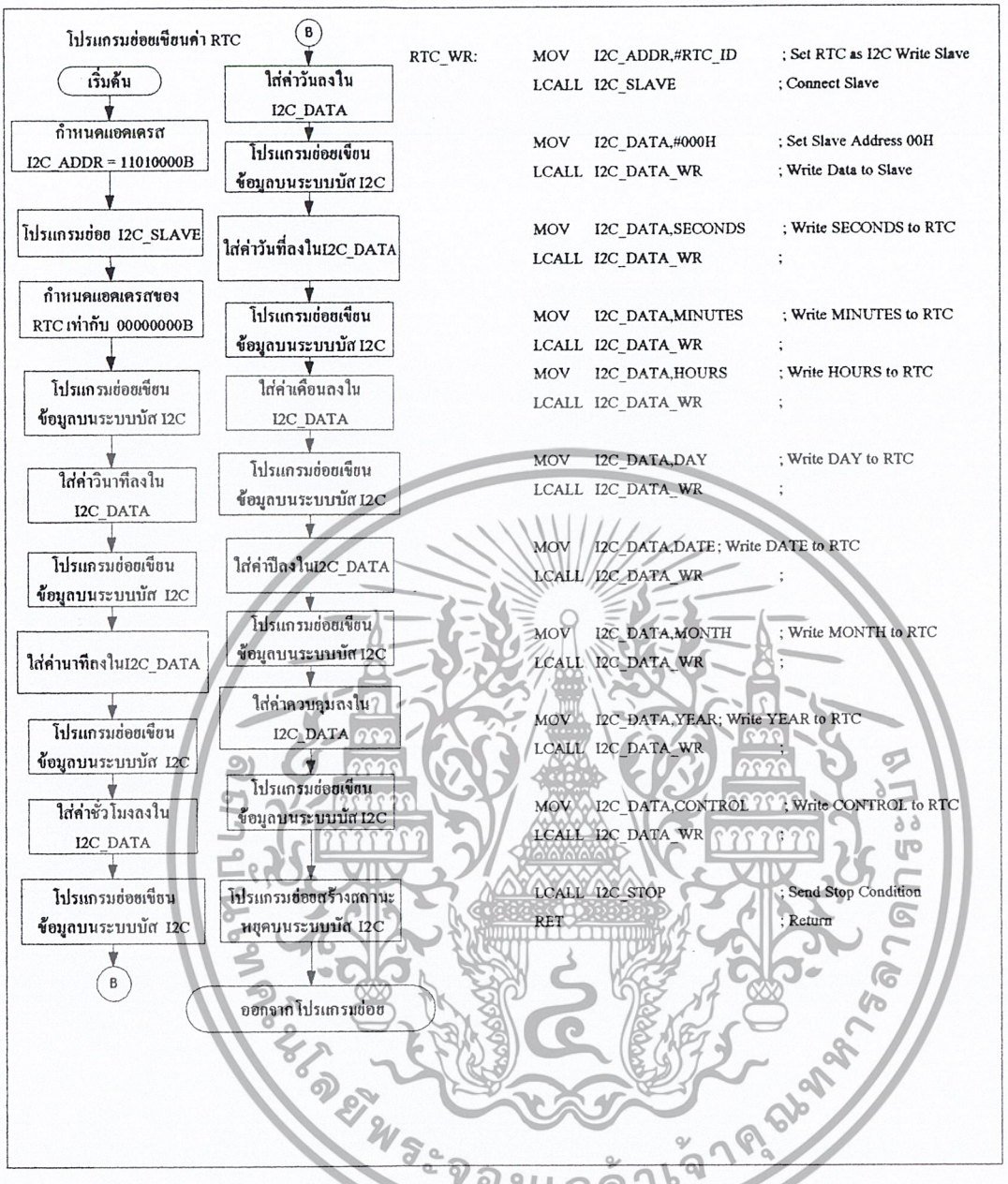
1. ศึกษาโฟลวชาร์ตในรูปที่ 4.10, 4.11 โปรแกรมที่ 1 และโปรแกรมที่ 2 ร่วมกับวงจรที่ใช้ในการทดลองตามรูปที่ 4.9 ของวงจรการทดลองที่ 5
2. เขียนโปรแกรมที่ 1 และโปรแกรมที่ 2 ทำการแอสแซมเบลอร์แล้วทำการโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ทั้งสองซึ่ง โปรแกรมที่ 1 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวมาสเตอร์ และ โปรแกรมที่ 2 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวสเลฟ (LCD)
3. ต่อวงจรตามรูปที่ 4.9
4. จ่ายไฟให้แก่วงจร รันโปรแกรมโดยกดสวิตซ์ RESET
5. สังเกตการแสดงผลของเวลาจาก LCD ที่ส่วนของบอร์ดตัวอุปกรณ์สเลฟ (LCD)



รูปที่ 4.9 แสดงวงจรการทดลองที่ 5



รูปที่ 4.10 แสดงโฟลวชาร์ตโปรแกรมย่อยการอ่านค่า RTC



รูปที่ 4.11 แสดงโฟลวชาร์ตโปรแกรมย่อยการเขียนค่า RTC

การทดลองที่ 6

การใช้งานระบบฐานบัส I^2C ของตัวควบคุมไมโครคอนโทรลเลอร์

จุดประสงค์

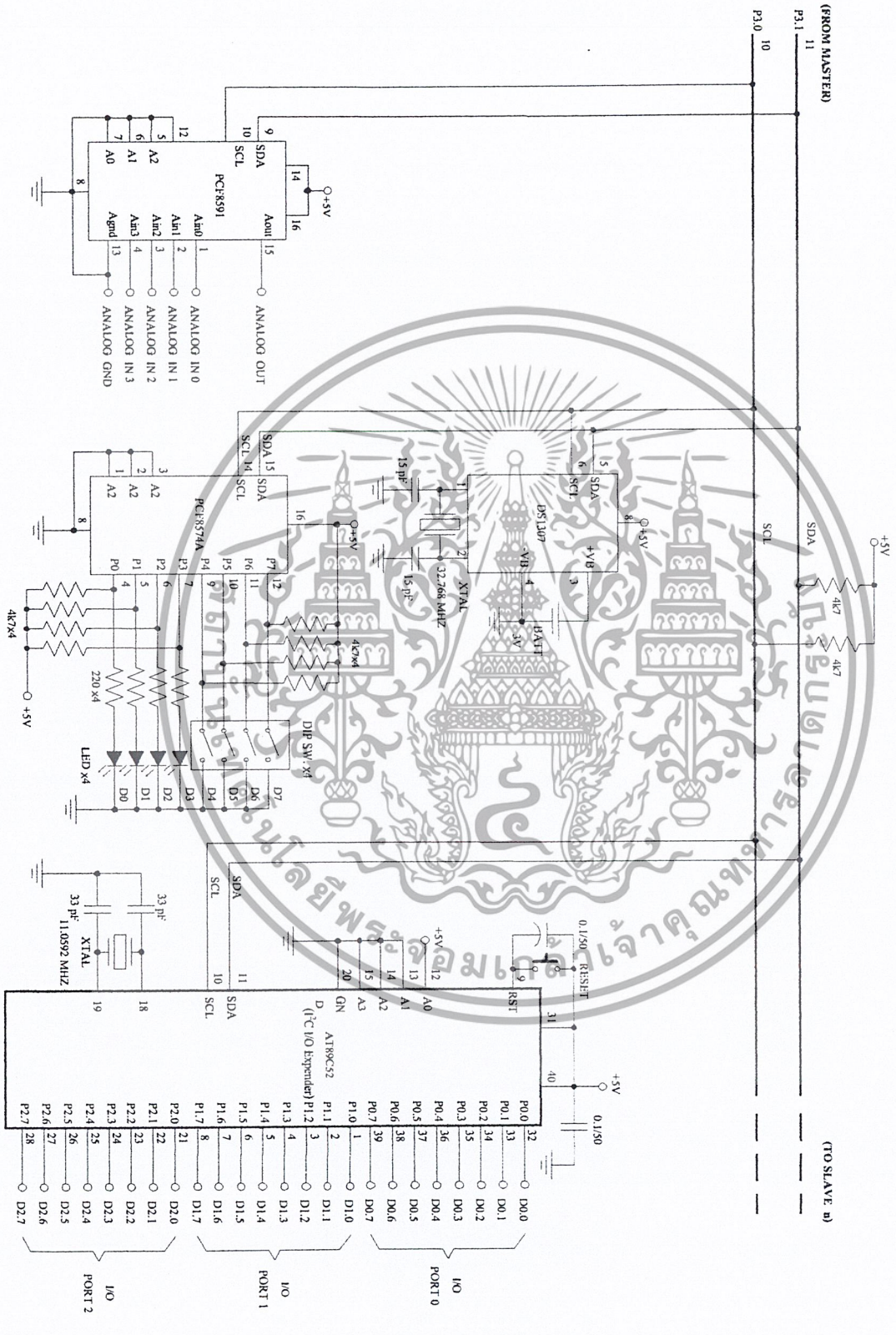
1. เพื่อศึกษาการรับส่งข้อมูลบนบัส I^2C โดยอุปกรณ์อินเตอร์เฟส ทุกตัวบนบัส
2. สามารถที่จะเลือกการติดต่อแต่ละตัวได้และทำการรับหรือส่งข้อมูลบนระบบบัส I^2C ได้
3. สามารถเขียนโปรแกรม เพื่อติดต่อกับอุปกรณ์ที่ใช้ การเชื่อมต่อบนระบบบัส I^2C (ซอฟแวร์และฮาร์ดแวร์) ได้

เครื่องมือและอุปกรณ์

1. เครื่องโปรแกรมไมโครคอนโทรลเลอร์ พร้อมสายต่อคอมพิวเตอร์ และอะแดปเตอร์ 1 ชุด
2. บอร์ดไมโครคอนโทรลเลอร์ (ตัวมาสเตอร์) กับบอร์ด อินเตอร์เฟสทุกตัว (อุปกรณ์สเลฟ)
3. สายต่อบัส I^2C

การทดลอง

1. ศึกษาโปรแกรมที่ 1 โปรแกรมที่ 2 โปรแกรมที่ 3 ร่วมกับวงจรที่ใช้ในการทดลองตามรูปที่ 4.12 ของวงจรการทดลองที่ 6
2. เขียนโปรแกรมที่ 1 โปรแกรมที่ 2 และโปรแกรมที่ 3 ทำการแอสแซมเบลอร์แล้วทำการโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ทั้งสองซึ่ง โปรแกรมที่ 1 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวมาสเตอร์ โปรแกรมที่ 2 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวสเลฟ(LCD) และ โปรแกรมที่ 3 จะโปรแกรมลงในตัวไมโครคอนโทรลเลอร์ตัวสเลฟที่ 2
3. ต่อวงจรตามรูปที่ 4.12
4. จ่ายไฟให้แก่วงจร รันโปรแกรมโดยกดสวิตซ์ RESET
5. ป้อนคำส่งเข้าไปในคีย์บอร์ด (KEY)
6. สังเกตการแสดงผลของเวลาจาก LCD ที่ส่วนของบอร์ดตัวอุปกรณ์สเลฟ (LCD) และ LED ส่วนของอุปกรณ์สเลฟทั้งหมด



รูปที่ 4.12 แสดงวงจรการทดลองที่ 6

สรุปผลการทดลอง

จากการทดลองทั้ง 6 การทดลอง จะทำให้เราเข้าใจถึงโครงสร้างการทำงานของคำสั่งในการติดต่ออุปกรณ์บนบัส I^2C โดยที่ไอซีอินเตอร์เฟสแต่ละตัวที่อยู่บนบัสจะทำหน้าที่แตกต่างกัน เช่น PCF 8574A เป็นไอซีขยายพอร์ตอินพุตเอาต์พุต ,PCF 8591 เป็นไอซี ADC/DAC ในการแปลงค่าอะนาลอกเป็นดิจิตอลหรือแปลงค่าดิจิตอลเป็นอะนาลอก ,DS 1307 เป็นไอซีสำหรับสร้างฐานเวลา แต่การติดต่อหรือควบคุมจะมีลักษณะเหมือนกันคือ การสร้างบิต start ,stop และบิตสัญญาณ acknowledge และที่ต่างกันคือ ค่าของแอดเดรสของอุปกรณ์แต่ละตัว

การสร้างตัวอุปกรณ์ขยายพอร์ตจากไมโครคอนโทรลเลอร์ MCS-51 โดยการกำหนดให้มีโปรโตคอลตรงตามมาตรฐานของ I^2C บัส จะเห็นได้ว่าเราสามารถเพิ่มข้อดีให้แก่อุปกรณ์ขยายพอร์ตตัวนี้มากกว่า PCF 8574A อีกทั้งยังสามารถที่นำข้อมูลที่ได้มาประมวลผลในการควบคุมได้

บทที่ 5
สรุปหลักการทำงานและการวิจารณ์

5.1 สรุปหลักการทำงานของระบบฐานบัส I^2C

เมื่อทำการต่ออุปกรณ์อินเตอร์เฟสบนฐานบัส I^2C เรียบร้อยแล้ว การที่เราจะรับส่งข้อมูลหรือควบคุมนั้นจะต้องมีตัวอุปกรณ์ที่เรียกว่าตัวมาสเตอร์เป็นตัวกำหนดสภาวะของบัสเริ่มต้น และสร้างสัญญาณนาฬิกาเพื่อตรวจสอบและควบคุมข้อมูลในการรับส่ง สภาวะของบัสเริ่มต้นนั้นเป็นสภาวะที่อุปกรณ์สเลฟตรวจสอบการรับส่งข้อมูลว่าบัสนั้นมีสภาวะว่างหรือไม่ เพื่อเป็นการป้องกันการชนกันของข้อมูลที่อาจจะเกิดขึ้นบนบัส I^2C จากนั้นตัวมาสเตอร์จะสร้างบิต START และทำการส่งค่าของแอดเดรสที่ต้องการจะติดต่อด้วยตามด้วยบิต อ่านหรือเขียน เมื่ออุปกรณ์ปลายทางที่ถูกติดต่อด้วยจะทำการส่งบิตตอบรับกลับมายังตัวมาสเตอร์ จากนั้นตัวมาสเตอร์จะส่งข้อมูลหรือรับข้อมูลไปเรื่อยๆ จนกว่ามาสเตอร์สร้างบิต STOP ซึ่งเป็นการสิ้นสุดการรับส่งข้อมูลในหนึ่งๆ ชุด และตัวมาสเตอร์จะสร้างสภาวะบัสว่างขึ้นมาอีก ซึ่งข้อมูลแต่ละชุดจะมีบิตตอบรับจากตัวสเลฟตามมาด้วย

5.2 วิจารณ์

การรับส่งข้อมูลผิดพลาดไปจากส่งข้อมูลที่ส่งแท้จริง โดยสาเหตุเกิดจากทางด้านฮาร์ดแวร์ของอุปกรณ์มาสเตอร์และอุปกรณ์สเลฟ โดยแต่ละอุปกรณ์อินเตอร์เฟสอาจจะมีแรงดันทาง ด้านอินพุตเอาต์พุตต่างกัน ขึ้นอยู่กับคุณสมบัติของอุปกรณ์อินเตอร์เฟสแต่ละตัว อาจจะเป็นชนิด NMOS, CMOS ซึ่งจะทำให้เกิดการผิดเพี้ยนของข้อมูลได้ วิธีการแก้ปัญหานี้โดยต่อค่าความต้านทานจากแหล่งจากเข้าขา SDA และ SCL ทั้งคู่ เพื่อให้ได้แรงดันอินพุตเอาต์พุตเท่ากัน

ความยุ่งยากในการเขียนโปรแกรมให้ตรงกับมาตรฐานบัส I^2C ที่มีมาตรฐานตามทฤษฎีที่กล่าวมาแล้ว

บรรณานุกรม

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

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

ภาคผนวก

โปรแกรมที่ 1 ส่วนของตัว MASTER

```
;*******************************
;PROGRAM OF A I2C – BASED
;   EMBEDDED - PROCESSOR
;            SYSTEM
;          (MASTER)
;*******************************
;         Define Port&Pin Name
;*******************************
SDA             BIT     P3.1
SCL             BIT     P3.0
;*******************************
;         Define User Registor
;*******************************
FLAG            EQU     2FH
I2C_ACK         BIT     FLAG.0
KEY_UP          BIT     FLAG.1
KEY_DW          BIT     FLAG.2
KEY_SHIFT       BIT     FLAG.3
KEY_TIME        BIT     FLAG.4
KEY_SET         BIT     FLAG.5
KEY_ADDR        BIT     FLAG.6
;*******************************
;         Define User Registor
;*******************************
D_ADDR          EQU     26H
I2C_ADDR        EQU     27H
I2C_DATA        EQU     28H
DATA_KEY        EQU     29H
IO_KEY          EQU     2AH
DATA_WR         EQU     2BH
DATA_READ       EQU     2CH
CONTROL_9       EQU     2DH
INT_ADDR        EQU     2EH
LCD_ADDR        EQU     30H
LCD_DATA        EQU     31H
```

(โปรแกรม MASTER ต่อ)

```
LCD_PTR         EQU     32H
DATA_SPEED      EQU     33H
BUFFER          EQU     34H
LCD_SET         EQU     35H
IO_DISP1        EQU     36H
IO_DISP2        EQU     37H
IO_DISP3        EQU     38H
DATA_DISPD      EQU     39H
SECONDS         EQU     40H
MINUTS          EQU     41H
HOURS           EQU     42H
DAY             EQU     43H
DATE            EQU     44H
MONTH           EQU     45H
YEAR            EQU     46H
CONTROL_RTC     EQU     47H
DATA_WR1        EQU     49H
;*********************************
;       Define I2C Slave Address
;*********************************
RTC_ID          EQU     11010000B
LCD_ID          EQU     00000100B
;*********************************
;    **** Main Program ****
;*********************************
                ORG     0000H
                MOV     P1,#11111111B
                MOV     P2,#11111111B
                MOV     P3,#11111111B
                MOV     CONTROL_91,#01000000B
MAIN:           LCALL   START_PROGRAM
                LCALL   CLRDISP
                MOV     D_ADDR,#00111000B
                MOV     DATA_WR,#0
                LCALL   I2C_PCF8574_WR
```

(โปรแกรม MASTER ต่อ)

```
        MOV   LCD_SET,#00
        LCALL LCD_WRITE
        MOV   R0,#50H
CLR_RAM: MOV  @R0,#00
        INC   R0
        MOV   A,R0
        CJNE  A,#0FFH,CLR_RAM
MAIN_1: MOV   DATA_WR,#0
        MOV   DATA_WR1,#0
        MOV   DATA_READ,#0
        MOV   D_ADDR,#0
        MOV   INT_ADDR,#0
;*****************************************
ADDR_SLAVE: ACALL WAIT_KEY
ADDR_SLAVE_1: MOV LCD_SET,#01H
        LCALL LCD_WRITE
        MOV   IO_DISP1,#0FFH
        MOV   IO_DISP2,INT_ADDR
        MOV   IO_DISP3,D_ADDR
        LCALL SAVE_DISP
        LCALL SCAN_DISP
        LCALL SCANKEY
        JC    ADDR_SLAVE_1
        LCALL CHECK_KEY
        JBC   KEY_ADDR,MAIN_1
        JBC   KEY_SET,ADDR_SLAVE_2
        JBC   KEY_TIME,ADDR_SLAVE_3
        JB    KEY_UP,ADDR_SLAVE_4
        JB    KEY_DW,ADDR_SLAVE_5
        JB    KEY_SHIFT,ADDR_SLAVE
        MOV   INT_ADDR,DATA_WR
        MOV   D_ADDR,DATA_WR1
        AJMP  ADDR_SLAVE
ADDR_SLAVE_2: AJMP ADDR_READ
ADDR_SLAVE_3: AJMP RTC_READ
```

(โปรแกรม MASTER ต่อ)

```
ADDR_SLAVE_4: MOV  DATA_SPEED,INT_ADDR
        ACALL SPEED_INC2
        MOV   INT_ADDR,DATA_SPEED
        AJMP  ADDR_SLAVE
ADDR_SLAVE_5: MOV  DATA_SPEED, INT_ADDR
        ACALL SPEED_DEC2
        MOV   INT_ADDR,DATA_SPEED
        AJMP  ADDR_SLAVE
;*****************************************
;*****************************************
ADDR_READ:  ACALL WAIT_KEY
ADDR_READ_1: LCALL I2C_READ
        LCALL SAVE_DISP
        MOV   IO_DISP1,DATA_READ
        MOV   IO_DISP2,INT_ADDR
        MOV   IO_DISP3,D_ADDR
        MOV   LCD_SET,#02H
        LCALL LCD_WRITE
        LCALL SCAN_DISP
        LCALL SCANKEY
        JC    ADDR_READ_1
        LCALL CHECK_KEY
        JBC   KEY_TIME,ADDR_READ_3
        JB    KEY_UP,ADDR_READ_4
        JB    KEY_DW,ADDR_READ_5
        JBC   KEY_ADDR,ADDR_READ_6
        MOV   DATA_WR,DATA_READ
;*****************************************
ADDR_WRITE: ACALL WAIT_KEY
ADDR_WRITE_1: MOV  IO_DISP1,DATA_WR
        MOV   IO_DISP2,INT_ADDR
        MOV   IO_DISP3,D_ADDR
        MOV   LCD_SET,#03H
        LCALL LCD_WRITE
        LCALL SAVE_DISP
```

(โปรแกรม MASTER ต่อ)

```
                LCALL   SCAN_DISP
                LCALL   SCANKEY
                JC      ADDR_WRITE_1
                LCALL   CHECK_KEY
                JBC     KEY_SET,ADDR_WRITE_2
                JBC     KEY_TIME,ADDR_READ_3
                JB      KEY_UP,ADDR_READ_4
                JB      KEY_DW,ADDR_READ_5
                JBC     KEY_ADDR,ADDR_READ_6
                AJMP    ADDR_WRITE
ADDR_WRITE_2:   ACALL   I2C_WRITE
                AJMP    ADDR_READ
;************************************************
ADDR_READ_3:    AJMP    RTC_READ
ADDR_READ_4:    MOV     DATA_SPEED,INT_ADDR
                ACALL   SPEED_INC2
                MOV     INT_ADDR,DATA_SPEED
                AJMP    ADDR_READ
ADDR_READ_5:    MOV     DATA_SPEED,INT_ADDR
                ACALL   SPEED_DEC2
                MOV     INT_ADDR,DATA_SPEED
                AJMP    ADDR_READ
ADDR_READ_6:    AJMP    ADDR_SLAVE
;************************************************
RTC_READ:       ACALL   WAIT_KEY
RTC_READ_1:     LCALL   I2C_RTC_RD
                MOV     LCD_SET,#04H
                LCALL   LCD_WRITE
                MOV     IO_DISP1,SECONDS
                MOV     IO_DISP2,MINUTS
                MOV     IO_DISP3,HOURS
                LCALL   SAVE_DISP
                LCALL   SCAN_DISP
                LCALL   SCANKEY
                JC      RTC_READ_1
```

(โปรแกรม MASTER ต่อ)

```
                LCALL   CHECK_KEY
                JBC     KEY_ADDR,RTC_READ_2
                JBC     KEY_TIME,RTC_READ_3
                JBC     KEY_SET,RTC_READ_4
                AJMP    RTC_READ
RTC_READ_2:     AJMP    ADDR_SLAVE
RTC_READ_3:     AJMP    ADDR_READ
RTC_READ_4:     MOV     DATA_WR,MINUTS
;************************************************
RTC_SETM:       ACALL   WAIT_KEY
RTC_SETM_1:     MOV     LCD_SET,#05H
                LCALL   LCD_WRITE
                MOV     IO_DISP1,#0FFH
                MOV     IO_DISP2,MINUTS
                MOV     IO_DISP3,#0FFH
                LCALL   SAVE_DISP
                LCALL   SCAN_DISP
                LCALL   SCANKEY
                JC      RTC_SETM_1
                LCALL   CHECK_KEY
                JBC     KEY_TIME,RTC_SETM_2
                JB      KEY_UP,RTC_SETM_3
                JB      KEY_DW,RTC_SETM_4
                JBC     KEY_SET,RTC_SETM_5
                MOV     MINUTS,DATA_WR
                AJMP    RTC_SETM
RTC_SETM_2:     LCALL   I2C_RTC_WR
                AJMP    RTC_READ
RTC_SETM_3:     MOV     DATA_SPEED,MINUTS
                ACALL   SPEED_INC2
                MOV     MINUTS,DATA_SPEED
                AJMP    RTC_SETM
RTC_SETM_4:     MOV     DATA_SPEED,MINUTS
                AJMP    SPEED_DEC2
                MOV     MINUTS,DATA_SPEED
```

(โปรแกรม MASTER ต่อ)

```
        AJMP    RTC_SETM
RTC_SETM_5: MOV DATA_WR,HOURS
;***********************************************
;***********************************************
RTC_SETH:   ACALL WAIT_KEY
RTC_SETH_1: MOV LCD_SET,#06H
        LCALL LCD_WRITE
        MOV IO_DISP1,#0FFH
        MOV IO_DISP2,#0FFH
        MOV IO_DISP3,HOURS
        LCALL SAVE_DISP
        LCALL SCAN_DISP
        LCALL SCANKEY
        JC RTC_SETH_1
        LCALL CHECK_KEY
        JBC KEY_SET,RTC_SETM
        JBC KEY_TIME,RTC_SETH_3
        JB KEY_UP,RTC_SETH_4
        JB KEY_DW,RTC_SETH_5
        MOV HOURS,DATA_WR
        AJMP RTC_SETH
RTC_SETH_3: LCALL I2C_RTC_WR
        AJMP RTC_READ
RTC_SETH_4: MOV DATA_SPEED,HOURS
        ACALL SPEED_INC3
        MOV HOURS,DATA_SPEED
        AJMP RTC_SETH
RTC_SETH_5: MOV DATA_SPEED,HOURS
        AJMP SPEED_DEC3
        MOV HOURS,DATA_SPEED
        AJMP RTC_SETH
```

(โปรแกรม MASTER ต่อ)

```
;*****************************************
;       DATA_SPEED2
;       I/P : DATA_SPEED2
;       O/P: DATA_SPEED
;*****************************************
SPEED_INC2: INC DATA_SPEED
        MOV R5,#40H
SPEED_INC_21: LCALL LCD_WRITE
        MOV IO_DISP1,#0FFH
        MOV IO_DISP2,DATA_SPEED
        MOV IO_DISP3,#0FFH
        LCALL SAVE_DISP
        LCALL SCAN_DISP
        LCALL SCANKEY
        JC SPEED_RET2
        LCALL CHECK_KEY
        JNB KEY_UP,SPEED_RET2
        DJNZ R5,SPEED_INC_21
SPEED_INC_22: MOV R5,#06H
SPEED_INC_23: LCALL LCD_WRITE
        MOV IO_DISP1,#0FFH
        MOV IO_DISP2,DATA_SPEED
        MOV IO_DISP3,#0FFH
        LCALL SAVE_DISP
        LCALL SCAN_DISP
        LCALL SCANKEY
        JC SPEED_RET2
        LCALL CHECK_KEY
        JNB KEY_UP,SPEED_RET2
        DJNZ R5,SPEED_INC_23
        INC DATA_SPEED
        AJMP SPEED_INC_22
;*************************************
SPEED_DEC2: DEC DATA_SPEED
        MOV R5,#40H
```

(โปรแกรม MASTER ต่อ)

```
SPEED_DEC_21: LCALL  LCD_WRITE
              MOV    IO_DISP1,#0FFH
              MOV    IO_DISP2,DATA_SPEED
              MOV    IO_DISP3,#0FFH
              LCALL  SAVE_DISP
              LCALL  SCAN_DISP
              LCALL  SCANKEY
              JC     SPEED_RET2
              LCALL  CHECK_KEY
              JNB    KEY_DW,SPEED_RET2
              DJNZ   R5,SPEED_DEC_21
SPEED_DEC_22: MOV    R5,#06H
SPEED_DEC_23: LCALL  LCD_WRITE
              MOV    IO_DISP1,#0FFH
              MOV    IO_DISP2,DATA_SPEED
              MOV    IO_DISP3,#0FFH
              LCALL  SAVE_DISP
              LCALL  SCAN_DISP
              LCALL  SCANKEY
              JC     SPEED_RET2
              LCALL  CHECK_KEY
              JNB    KEY_DW,SPEED_RET2
              DJNZ   R5,SPEED_DEC_23
              DEC    DATA_SPEED
              AJMP   SPEED_DEC_22
SPEED_RET2:   RET
;*****************************************
;       DATA_SPEED3
;       I/P : DATA_SPEED3
;       O/P: DATA_SPEED
;*****************************************
SPEED_INC3:   INC    DATA_SPEED
              MOV    R5,#40H
SPEED_INC_31: LCALL  LCD_WRITE
              MOV    IO_DISP1,#0FFH
```

(โปรแกรม MASTER ต่อ)

```
              MOV    IO_DISP2,#0FFH
              MOV    IO_DISP3,DATA_SPEED
              LCALL  SAVE_DISP
              LCALL  SCAN_DISP
              LCALL  SCANKEY
              JC     SPEED_RET3
              LCALL  CHECK_KEY
              JNB    KEY_UP,SPEED_RET3
              DJNZ   R5,SPEED_INC_31
SPEED_INC_32: MOV    R5,#06H
SPEED_INC_33: LCALL  LCD_WRITE
              MOV    IO_DISP1,#0FFH
              MOV    IO_DISP2,#0FFH
              MOV    IO_DISP3,DATA_SPEED
              LCALL  SAVE_DISP
              LCALL  SCAN_DISP
              LCALL  SCANKEY
              JC     SPEED_RET3
              LCALL  CHECK_KEY
              JNB    KEY_UP,SPEED_RET3
              DJNZ   R5,SPEED_INC_33
              INC    DATA_SPEED
              AJMP   SPEED_INC_32
;*****************************************
SPEED_DEC3:   DEC    DATA_SPEED
              MOV    R5,#40H
SPEED_DEC_31: LCALL  LCD_WRITE
              MOV    IO_DISP1,#0FFH
              MOV    IO_DISP2,#0FFH
              MOV    IO_DISP3,DATA_SPEED
              LCALL  SAVE_DISP
              LCALL  SCAN_DISP
              LCALL  SCANKEY
              JC     SPEED_RET3
              LCALL  CHECK_KEY
```

(โปรแกรม MASTER ต่อ)

```
                JNB     KEY_DW,SPEED_RET3
                DJNZ    R5,SPEED_DEC_31
SPEED_DEC_32:   MOV     R5,#06H
SPEED_DEC_33:   LCALL   LCD_WRITE
                MOV     IO_DISP1,#0FFH
                MOV     IO_DISP2,#0FFH
                MOV     IO_DISP3,DATA_SPEED
                LCALL   SAVE_DISP
                LCALL   SCAN_DISP
                LCALL   SCANKEY
                JC      SPEED_RET3
                LCALL   CHECK_KEY
                JNB     KEY_DW,SPEED_RET3
                DJNZ    R5,SPEED_DEC_33
                DEC     DATA_SPEED
                AJMP    SPEED_DEC_32
SPEED_RET3:     RET
;*********************************
;       Store Data  Display
;   Input 1         =  IO_DISP1
;   Input 2         =  IO_DISP2
;   Input3          =  IO_DISP3
;   Output          =  DATA_DISPD(39H)
;*********************************
SAVE_DISP:      MOV     R0,#DATA_DISPD
                MOV     A,IO_DISP1
                ANL     A,#0FH
                MOV     @R0,A
                INC     R0
                MOV     A,IO_DISP1
                ANL     A,#0F0H
                SWAP    A
                MOV     @R0,A
                INC     R0
                MOV     A,IO_DISP2
```

(โปรแกรม MASTER ต่อ)

```
                ANL     A,#0FH
                MOV     @R0,A
                INC     R0
                MOV     A,IO_DISP2
                ANL     A,#0F0H
                SWAP    A
                MOV     @R0,A
                INC     R0
                MOV     A,IO_DISP3
                ANL     A,#0FH
                MOV     @R0,A
                INC     R0
                MOV     A,IO_DISP3
                ANL     A,#0F0H
                SWAP    A
                MOV     @R0,A
                RET
;*********************************
;       **** START PROGRAM ****
;*********************************
START_PROGRAM:  ACALL   CLRDISP
                MOV     R5,#6
                MOV     R1,#DATA_DISPD
ST_PRO:         MOV     @R1,#8
                MOV     R6,#86H
ST_PRO_1:       ACALL   SCAN_DISP
                DJNZ    R6,ST_PRO_1
                INC     R1
                DJNZ    R5,ST_PRO
                DEC     R1
                MOV     R5,#6
ST_PRO2:        MOV     @R1,#0FFH
                MOV     R6,#86H
ST_PRO_2:       ACALL   SCAN_DISP
                DJNZ    R6,ST_PRO_2
```

(โปรแกรม MASTER ต่อ)

```
        DEC     R1
        DJNZ    R5,ST_PRO2
        RET
;*********************************
;     **** CLEAR  DISPLAY****
;*********************************
CLRDISP:        MOV     R1,#6
        MOV     R0,#DATA_DISPD
CLRD:   MOV     @R0,#0
        INC     R0
        DJNZ    R1,CLRD
        RET
;*****************************************
;*********************************
;          **** WAIT_KEY ****
;*********************************
WAIT_KEY:   LCALL  SCAN_DISP
        LCALL  SCANKEY
        JNC     WAIT_KEY
        LCALL  SCAN_DISP
        LCALL  SCANKEY
        JNC     WAIT_KEY
        RET
;*******************************
;          *** SCANKEY ****
;  Input      =  FROM KEY (PORT 1)
;  Output     =  DATA_KEY
;*******************************
SCANKEY:    MOV     R0,#4
        MOV     R3,#0F8H
LOOP:       PUSH    ACC
        PUSH    03H
        XCH     A,R3
        SWAP    A
        MOV     P1,A
```

(โปรแกรม MASTER ต่อ)

```
        POP     03H
        POP     ACC
        MOV     A,P1
        MOV     R2,#4
LOOP2:  RRC     A
        JNC     ONKEY
        DJNZ    R2,LOOP2
        INC     R3
        DJNZ    R0,LOOP
        SETB    C
        MOV     P1,#11111111B
        RET
;************************
ONKEY:      MOV     A,R3
        ANL     A,#7H
        SWAP    A
        ORL     A,R2
        CLR     C
        MOV     DATA_KEY,A
        RET
;*******************************
;          ***  SCANDISPLAY  ***
;  Input      =  DATA_DISPD..(80H-85H)
;  Output     =  DISPLAY...(PORT 2)...
;*******************************
SCAN_DISP:  MOV     R2,#6
        MOV     R0,#DATA_DISPD
        MOV     R3,#0F8H
SCANDLP:    ACALL  SCAND1
        INC     R3
        INC     R0
        DJNZ    R2,SCANDLP
        MOV     P1,#11111111B
        RET
```

(โปรแกรม MASTER ต่อ)

```
SCAND1:     MOV   P2,@R0
            PUSH  ACC
            PUSH  03H
            XCH   A,R3
            SWAP  A
            MOV   P1,A
            POP   03H
            POP   ACC
            ACALL DELAY
            MOV   P2,#0FFH
            RET
      ;.....DELAY.....
DELAY:      MOV   R4,#00
            DJNZ  R4,$
            MOV   R4,#00
            DJNZ  R4,$
            RET
;*********************************
;        *** CHECK KEY ***
;    Input     =  DATA_KEY
;    Output    =  DATA_WR
;*********************************
CHECK_KEY:  CLR   KEY_UP
            CLR   KEY_DW
            MOV   A,DATA_KEY
            CJNE  A,#31H,CHK_1
            SETB  KEY_UP
            AJMP  RET_2
CHK_1:      CJNE  A,#11H,CHK_3
            SETB  KEY_DW
            AJMP  RET_2
CHK_3:      CJNE  A,#01H,CHK_5
            SETB  KEY_SET
            AJMP  RET_2
```

(โปรแกรม MASTER ต่อ)

```
CHK_5:      CJNE  A,#03H,CHK_6
            MOV   IO_KEY,#0H
CHK_6:      CJNE  A,#34H,CHK_7
            JBC   KEY_SHIFT,CHK_61
            MOV   IO_KEY,#01H
            JMP   CHK_7
CHK_61:     MOV   IO_KEY,#0AH
CHK_7:      CJNE  A,#33H,CHK_8
            JBC   KEY_SHIFT,CHK_71
            MOV   IO_KEY,#02H
            JMP   CHK_8
CHK_71:     MOV   IO_KEY,#0BH
CHK_8:      CJNE  A,#32H,CHK_9
            JBC   KEY_SHIFT,CHK_81
            MOV   IO_KEY,#03H
            JMP   CHK_9
CHK_81:     MOV   IO_KEY,#0CH
CHK_9:      CJNE  A,#24H,CHK_10
            JBC   KEY_SHIFT,CHK_91
            MOV   IO_KEY,#04H
            JMP   CHK_10
CHK_91:     MOV   IO_KEY,#0DH
CHK_10:     CJNE  A,#23H,CHK_11
            JBC   KEY_SHIFT,CHK_101
            MOV   IO_KEY,#05H          JMP
            CHK_11
CHK_101:    MOV   IO_KEY,#0EH
CHK_11:     CJNE  A,#22H,CHK_12
            JBC   KEY_SHIFT,CHK_111
            MOV   IO_KEY,#06H
            JMP   CHK_12
CHK_111:    MOV   IO_KEY,#0FH
CHK_12:     CJNE  A,#14H,CHK_13
            MOV   IO_KEY,#07H
CHK_13:     CJNE  A,#13H,CHK_14
```

(โปรแกรม MASTER ต่อ)

```
        MOV   IO_KEY,#08H
CHK_14: CJNE  A,#12H,CHK_15
        MOV   IO_KEY,#09H
CHK_15: CJNE  A,#21H,CHK_16
        SETB  KEY_ADDR
        AJMP  RET_2
CHK_16: CJNE  A,#02H,CHK_17
        SETB  KEY_TIME
        AJMP  RET_2
CHK_17: CJNE  A,#04H,RET_1
        JBC   KEY_SHIFT,CHK_171
        SETB  KEY_SHIFT
CHK_171: AJMP RET_2
RET_1:  MOV   A,DATA_WR
        SWAP  A
        MOV   R5,A
        ANL   A,#0F0H
        ORL   A,IO_KEY
        MOV   DATA_WR,A
        MOV   A,DATA_WR1
        SWAP  A
        ANL   A,#0F0H
        PUSH  ACC
        MOV   A,R5
        ANL   A,#0FH
        MOV   R5,A
        POP   ACC
        ORL   A,R5
        MOV   DATA_WR1,A
RET_2:  RET
;*******************************
```

(โปรแกรม MASTER ต่อ)

```
;*******************************
;       *** PROGRAM  I2C ***
;*******************************
;*******************************
;       ****READ DATA I2C*****
;*******************************
I2C_READ:     MOV   A,D_ADDR
              CJNE  A,#00111000B,I2C_READ_1
              ACALL I2C_PCF8574_RD
              MOV   DATA_READ,I2C_DATA
              MOV   INT_ADDR,#0
              RET
I2C_READ_1:   CJNE  A,#01001000B,I2C_READ_2
              ACALL I2C_PCF8591_RD
              MOV   DATA_READ,I2C_DATA
              MOV   INT_ADDR,#0
              RET
I2C_READ_2:   CJNE  A,#01101000B,I2C_READ_3
              ACALL I2C_RTC_RD_1
              RET
I2C_READ_3:   CJNE  A,#00000010B,I2C_READ_4
              RET
I2C_READ_4:   CJNE  A,#0,I2C_READ_5
              LCALL I2C_RTC_RD
              MOV   R0,INT_ADDR
              MOV   DATA_READ,@R0
              RET
I2C_READ_5:   ACALL I2C_MCS51_RD
              MOV   DATA_READ,I2C_DATA
              RET
;*****************************************
```

(โปรแกรม MASTER ต่อ)

```
;*************************************
;          ****WRITE DATA I2C*****
;*************************************
I2C_WRITE:      MOV     A,D_ADDR
                CJNE    A,#00111000B,I2C_WRITE_1
                ACALL   I2C_PCF8574_WR
                RET
I2C_WRITE_1:    CJNE    A,#01001000B,I2C_WRITE_2
                RET
I2C_WRITE_2:    CJNE    A,#01001000B,I2C_WRITE_3
                ACALL   I2C_RTC_WR_1
                RET
I2C_WRITE_3:    CJNE    A,#00000010B,I2C_WRITE_4
                RET
I2C_WRITE_4:    CJNE    A,#0,I2C_WRITE_5
                MOV     R0,INT_ADDR
                MOV     A,INT_ADDR
                CLR     C
                SUBB    A,#50H
                JC      I2C_WRITE_41
                MOV     @R0,DATA_WR
I2C_WRITE_41:   RET
I2C_WRITE_5:    ACALL   I2C_MCS51_WR
                RET
;*********************************
```

(โปรแกรม MASTER ต่อ)

```
;*********************************
;        *** I2C MCS51 Write ***
;*********************************
I2C_MCS51_WR:   CLR     C
                MOV     A,D_ADDR
                RLC     A
                MOV     I2C_ADDR,A
                ACALL   I2C_SLAVE
                MOV     I2C_DATA,INT_ADDR
                ACALL   I2C_DATA_WR
                MOV     I2C_DATA,DATA_WR
                ACALL   I2C_DATA_WR
                ACALL   I2C_STOP
                RET
;*********************************
;        *** I2C PCF8591 Write ***
;*********************************
I2C_PCF8591_WR:     CLR     C
                MOV     A,D_ADDR
                RLC     A
                MOV     I2C_ADDR,A
                ACALL   I2C_SLAVE
                MOV     I2C_DATA,CONTROL_91
                ACALL   I2C_DATA_WR
                MOV     I2C_DATA,DATA_WR
                ACALL   I2C_DATA_WR
                ACALL   I2C_STOP
                RET
;*********************************
```

(โปรแกรม MASTER ต่อ)

```
;*******************************
;       *** I2C RTC Write ***
;*******************************
I2C_RTC_WR:   MOV   I2C_ADDR,#RTC_ID
              ACALL I2C_SLAVE
              MOV   I2C_DATA,#000
              ACALL I2C_DATA_WR
              MOV   I2C_DATA,SECONDS
              ACALL I2C_DATA_WR
              MOV   I2C_DATA,MINUTS
              ACALL I2C_DATA_WR
              MOV   I2C_DATA,HOURS
              ACALL I2C_DATA_WR
              ACALL I2C_STOP
              RET
      ;*************************
I2C_RTC_WR_1: MOV   I2C_ADDR,#RTC_ID
              ACALL I2C_SLAVE
              MOV   I2C_DATA,INT_ADDR
              ACALL I2C_DATA_WR
              MOV   I2C_DATA,DATA_WR
              ACALL I2C_DATA_WR
              ACALL I2C_STOP
              RET
;*******************************
;       *** I2C LCD Write ***
;*******************************
LCD_WRITE:    MOV   I2C_ADDR,#LCD_ID
              ACALL I2C_SLAVE
              MOV   I2C_DATA,LCD_SET
              ACALL I2C_DATA_WR
              MOV   I2C_DATA,D_ADDR
              ACALL I2C_DATA_WR
              MOV   I2C_DATA,INT_ADDR
              ACALL I2C_DATA_WR
```

(โปรแกรม MASTER ต่อ)

```
              MOV   I2C_DATA,DATA_WR
              ACALL I2C_DATA_WR
              MOV   I2C_DATA,DATA_READ
              ACALL I2C_DATA_WR
              MOV   I2C_DATA,SECONDS
              ACALL I2C_DATA_WR
              MOV   I2C_DATA,MINUTS
              ACALL I2C_DATA_WR
              MOV   I2C_DATA,HOURS
              ACALL I2C_DATA_WR
              ACALL I2C_STOP
              RET
;*******************************
;       *** I2C PCF8574 Write ***
;*******************************
I2C_PCF8574_WR:   CLR   C
                  MOV   A,D_ADDR
                  RLC   A
                  MOV   I2C_ADDR,A
                  ACALL I2C_SLAVE
                  MOV   I2C_DATA,DATA_WR
                  ACALL I2C_DATA_WR
                  ACALL I2C_STOP
                  RET
;*******************************
;       *** I2C MCS51 Read ***
;*******************************
I2C_MCS51_RD: SETB  C
              MOV   A,D_ADDR
              RLC   A
              MOV   I2C_ADDR,A
              ACALL I2C_SLAVE
              MOV   I2C_DATA,INT_ADDR
              ACALL I2C_DATA_WR
```

(โปรแกรม MASTER ต่อ)

```
                ACALL  I2C_DATA_RD
                ACALL  I2C_NACK_BIT
                ACALL  I2C_STOP
                RET
;*******************************
;       *** I2C MCS51(2) Read  ***
;*******************************
I2C_MCS51_RD_2:         SETB    C
                MOV     A,D_ADDR
                RLC     A
                MOV     I2C_ADDR,A
                ACALL  I2C_SLAVE
                ACALL  I2C_DATA_RD
                ACALL  I2C_NACK_BIT
                ACALL  I2C_STOP
                RET
;*******************************
;*******************************
;       *** I2C  PCF8591   Read ***
;*******************************
I2C_PCF8591_RD:         ACALL  I2C_PCF8591_WR
                ACALL  I2C_MCS51_RD_2
                RET
;*******************************
;       *** I2C  RTC Read ***
;*******************************
I2C_RTC_RD:     MOV     I2C_ADDR,#RTC_ID
                ACALL  I2C_SLAVE
                MOV     I2C_DATA,#000
                ACALL  I2C_DATA_WR
                MOV     I2C_ADDR,#RTC_ID+1
                ACALL  I2C_SLAVE
                ACALL  I2C_DATA_RD
                MOV     SECONDS,I2C_DATA
                LCALL  I2C_ACK_BIT
```

(โปรแกรม MASTER ต่อ)

```
                ACALL  I2C_DATA_RD
                MOV     MINUTS,I2C_DATA
                LCALL  I2C_ACK_BIT
                ACALL  I2C_DATA_RD
                MOV     HOURS,I2C_DATA
                LCALL  I2C_ACK_BIT
                ACALL  I2C_DATA_RD
                MOV     DAY,I2C_DATA
                LCALL  I2C_ACK_BIT
                ACALL  I2C_DATA_RD
                MOV     DATE,I2C_DATA
                LCALL  I2C_ACK_BIT
                ACALL  I2C_DATA_RD
                MOV     MONTH,I2C_DATA
                LCALL  I2C_ACK_BIT
                ACALL  I2C_DATA_RD
                MOV     YEAR,I2C_DATA
                LCALL  I2C_NACK_BIT
                ACALL  I2C_STOP
                RET
;*************************
I2C_RTC_RD_1:   MOV     I2C_ADDR,#RTC_ID
                ACALL  I2C_SLAVE
                MOV     I2C_DATA,INT_ADDR
                ACALL  I2C_DATA_WR
                MOV     I2C_ADDR,#RTC_ID+1
                ACALL  I2C_SLAVE
                ACALL  I2C_DATA_RD
                MOV     DATA_READ,I2C_DATA
                LCALL  I2C_NACK_BIT
                ACALL  I2C_STOP
                RET
;*******************************
```

(โปรแกรม MASTER ต่อ)

```
;*******************************
;     *** I2C  PCF8574  Read ***
;*******************************
I2C_PCF8574_RD:     SETB   C
                    MOV    A,D_ADDR
                    RLC    A
                    MOV    I2C_ADDR,A
                    ACALL  I2C_SLAVE
                    ACALL  I2C_DATA_RD
                    ACALL  I2C_NACK_BIT
                    ACALL  I2C_STOP
                    RET
;*******************************
;     *** I2C Slave Connect ***
;         Address = I2C_ADDR
;*******************************
I2C_SLAVE:          PUSH   ACC
                    SETB   I2C_ACK
                    LCALL  I2C_START
                    MOV    A,I2C_ADDR
                    MOV    R1,#08H
I2C_SLAVE_1:        RLC    A
                    MOV    SDA,C
                    ACALL  I2C_CLK
                    DJNZ   R1,I2C_SLAVE_1
                    SETB   SDA
                    ACALL  I2C_DELAY
                    SETB   SCL
                    ACALL  I2C_DELAY
                    JB     SDA,I2C_SLAVE_2
                    CLR    I2C_ACK
I2C_SLAVE_2:        CLR    SCL
                    POP    ACC
                    RET
;*******************************
```

(โปรแกรม MASTER ต่อ)

```
;*******************************
;     *** I2C   Data  Write ***
;         Data  I/P = I2C_DATA
;*******************************
I2C_DATA_WR:        PUSH   ACC
                    SETB   I2C_ACK
                    MOV    A,I2C_DATA
                    MOV    R1,#08
I2C_DATA_WR_1:      RLC    A
                    MOV    SDA,C
                    ACALL  I2C_CLK
                    DJNZ   R1,I2C_DATA_WR_1
                    SETB   SDA
                    ACALL  I2C_DELAY
                    SETB   SCL
                    ACALL  I2C_DELAY
                    JB     SDA,I2C_DATA_WR_2
                    CLR    I2C_ACK
I2C_DATA_WR_2:      CLR    SCL
                    POP    ACC
                    RET
;*******************************
;*******************************
;     ***I2C  Data  Read ***
;       Data  O/P =  I2C_DATA
;*******************************
I2C_DATA_RD:        PUSH   ACC
                    SETB   I2C_ACK
                    CLR    A
                    MOV    R1,#08
I2C_DATA_RD_1:      ACALL  I2C_DELAY
                    SETB   SCL
                    ACALL  I2C_DELAY
                    MOV    C,SDA
                    RLC    A
```

(โปรแกรม MASTER ต่อ)

```
                CLR     SCL
                DJNZ    R1,I2C_DATA_RD_1
                MOV     I2C_DATA,A
                POP     ACC
                RET
;************************************
;       *** I2C  Start Condition ***
;************************************
I2C_START:      JB      SCL,I2C_START_1
                SETB    SCL
I2C_START_1:    SETB    SDA
                ACALL   I2C_DELAY
                CLR     SDA
                ACALL   I2C_DELAY
                CLR     SCL
                RET
;*******************************
;           I2C  Stop Condition
;*******************************
I2C_STOP:       JNB     SCL,I2C_STOP_1
                CLR     SCL
I2C_STOP_1:     CLR     SDA
                ACALL   I2C_DELAY
                SETB    SCL
                ACALL   I2C_DELAY
                SETB    SDA
                RET
;*******************************
;*******************************
;           I2C  Acknowledge
;*******************************
I2C_ACK_BIT:    CLR     SDA
                ACALL   I2C_CLK
                SETB    SDA
                RET
```

(โปรแกรม MASTER ต่อ)

```
;*******************************
;           I2C  Not  Acknowledge
;*******************************
I2C_NACK_BIT:   SETB    SDA
                ACALL   I2C_CLK
                SETB    SDA
                RET
;*******************************
;           I2C  CLOCK (SCL)
;*******************************
I2C_CLK:        ACALL   I2C_DELAY
                SETB    SCL
                ACALL   I2C_DELAY
                CLR     SCL
                RET
;*******************************
;*************************************************
; Dummy  Delay  time I2C_DELAY,  10ms, 100ms , ls
;*************************************************
I2C_DELAY:      MOV     R2,#00CH
I2C_DELAY_1:    NOP
                NOP
                DJNZ R2,I2C_DELAY_1
                RET

DELAY_10ms:     MOV     R2,#10
DELAY_10ms_1:   MOV     R4,#0E6H
DELAY_10ms_2:   NOP
                NOP
                DJNZ    R4,DELAY_10ms_2
                DJNZ    R2,DELAY_10ms_1
                RET

DELAY_100ms:    MOV     R2,#100
DELAY_100ms_1:          MOV  R4,#0E6H
```

(โปรแกรม MASTER ต่อ)

```
DELAY_100ms_2:      NOP
                    NOP
                DJNZ    R4,DELAY_100ms_2
                DJNZ    R2,DELAY_100ms_1
                RET
;*******************************
                END
```

โปรแกรมที่ 2 ส่วนของตัว SLAVE (LCD)

```
;*******************************
;PROGRAM OF A I2C – BASED
;   EMBEDDED - PROCESSOR
;            SYSTEM
;          (SLAVE LCD)
;*******************************
;         Define Port&Pin Name
;*******************************
SDA             BIT     P3.1
SCL             BIT     P3.0
LCD_EN          BIT     P3.6
LCD_RS          BIT     P3.7
;*******************************
;         Define User Registor
;*******************************
FLAG            EQU     2FH
FLAG1           EQU     2EH
I2C_ACK         BIT     FLAG.0
ERR_BIT         BIT     FLAG.1
LCD_BIT0        BIT     FLAG1.0
LCD_BIT1        BIT     FLAG1.1
LCD_BIT2        BIT     FLAG1.2
LCD_BIT3        BIT     FLAG1.3
LCD_BIT4        BIT     FLAG1.4
LCD_BIT5        BIT     FLAG1.5
LCD_BIT6        BIT     FLAG1.6
LCD_ADDR        EQU     030H
LCD_DATA        EQU     031H
;*******************************
;         Define User Registor
;*******************************
I2C_ADDR        EQU     26H
I2C_DATA        EQU     27H
DATA_WR         EQU     28H
DATA_READ       EQU     29H
```

(โปรแกรม SLAVE LCD ต่อ)

```
INT_ADDR        EQU     2AH
LCD_SET         EQU     2BH
LCD_PTR         EQU     32H
BUFFER          EQU     33H
DATA_ADDR       EQU     34H
SECONDS         EQU     40H
MINUTES         EQU     41H
HOURS           EQU     42H
DAY             EQU     43H
DATE            EQU     44H
MONTH           EQU     45H
YEAR            EQU     46H
CONTROL_RTC     EQU     47H
;*******************************
;         Define I2C Slave Address
;*******************************
SLAVE_ID_R      EQU     00000101B
SLAVE_ID_W      EQU     00000100B
;*******************************
;*******************************
;            **** Main Program ****
;*******************************
                ORG 0000H
MAIN_0:         MOV     FLAG1,#0
                MOV     P0,#00000000B
                MOV     P1,#11111111B
                MOV     P3,#11111111B
                MOV     P2,#10000001B
                CLR     LCD_EN
                CLR     LCD_RS
                ACALL   INIT_LCD
                MOV     LCD_ADDR,#000H
                ACALL   SET_ADDR_LCD
                MOV     DPTR,#TITLE_1
```

(โปรแกรม SLAVE LCD ต่อ)

```
                ACALL  WRLINE_LCD
                ACALL  DELAY_LS
                ACALL  DELAY_LS
                ACALL  DELAY_LS
;***********************************************
 MAIN:          SETB   SDA
                 SETB  SCL
;***********************************************
START_BIT:      JNB    SDA,$
START_BIT_1:    JNB    SCL,START_BIT
                JB     SDA,START_BIT_1
                JNB    SCL,START_BIT
                ACALL  I2C_CLK_LOW
                JNB    ERR_BIT,START_BIT
                JB     SDA,START_BIT
                ACALL  ADDR_MASTER
                MOV    A,I2C_ADDR
                CJNE   A,#SLAVE_ID_R,ADDRESS_1
                AJMP   MAIN
ADDRESS_1:      CJNE   A,#SLAVE_ID_W,MAIN
                 ACALL I2C_ACK_BIT
                ACALL  MASTER_WRITE
                ACALL  I2C_ACK_BIT
                MOV    LCD_SET,I2C_DATA
                ACALL  MASTER_WRITE
                ACALL  I2C_ACK_BIT
                MOV    DATA_ADDR,I2C_DATA
                ACALL  MASTER_WRITE
                ACALL  I2C_ACK_BIT
                MOV    INT_ADDR,I2C_DATA
                ACALL  MASTER_WRITE
                ACALL  I2C_ACK_BIT
                MOV    DATA_WR,I2C_DATA
                ACALL  MASTER_WRITE
                ACALL  I2C_ACK_BIT
```

(โปรแกรม SLAVE LCD ต่อ)

```
                MOV    DATA_READ,I2C_DATA
                ACALL  MASTER_WRITE
                ACALL  I2C_ACK_BIT
                MOV    SECONDS,I2C_DATA
                ACALL  MASTER_WRITE
                ACALL  I2C_ACK_BIT
                MOV    MINUTES,I2C_DATA
                ACALL  MASTER_WRITE
                ACALL  I2C_NACK_BIT
                MOV    HOURS,I2C_DATA
STOP_BIT:       ACALL  I2C_CLK_HIGH
                JB     SDA,STOP_BIT3
                MOV    R2,#1AH
STOP_BIT1:      JB     SDA,STOP_BIT2
                DJNZ   R2,STOP_BIT1
STOP_BIT2:      SETB   I2C_ACK
STOP_BIT3:      AJMP   DISPLAY
;***********************************************
DISPLAY:        MOV    A,LCD_SET
                CJNE   A,#01H,DISPLAY_1
                AJMP   DISP_ADDR
DISPLAY_1:      CJNE   A,#02H,DISPLAY_2
                AJMP   DISP_READ
DISPLAY_2:      CJNE   A,#03H,DISPLAY_3
                AJMP   DISP_WRITE
DISPLAY_3:      CJNE   A,#04H,DISPLAY_4
                AJMP   DISP_TIME
DISPLAY_4:      CJNE   A,#05H,DISPLAY_5
                AJMP   DISP_SETM
DISPLAY_5:      CJNE   A,#06H,DISPLAY_6
                AJMP   DISP_SETH
DISPLAY_6:      AJMP   MAIN
;***********************************************
```

(โปรแกรม SLAVE LCD ต่อ)

```
;*********************************************
DISP_ADDR:    JB      LCD_BIT1,DISP_ADDR1
              MOV     FLAG1,#00000010B
              MOV     LCD_ADDR,#000H
              ACALL   SET_ADDR_LCD
              MOV     DPTR,#SCR_ADDR
              ACALL   WRLINE_LCD
DISP_ADDR1:   MOV     LCD_ADDR,#043H
              ACALL   SET_ADDR_LCD
              MOV     LCD_DATA,DATA_ADDR
              ACALL   HEX2LCD
              MOV     LCD_ADDR,#045H
              ACALL   SET_ADDR_LCD
              MOV     LCD_DATA,INT_ADDR
              ACALL   HEX2LCD
              LJMP    MAIN
DISP_READ:    JB      LCD_BIT2,DISP_READ1
              MOV     FLAG1,#00000100B
              MOV     LCD_ADDR,#000H
              ACALL   SET_ADDR_LCD
              MOV     DPTR,#SCR_READ
              ACALL   WRLINE_LCD
DISP_READ1:   MOV     LCD_ADDR,#006H
              ACALL   SET_ADDR_LCD
              MOV     LCD_DATA,DATA_ADDR
              ACALL   HEX2LCD
              MOV     LCD_ADDR,#040H
              ACALL   SET_ADDR_LCD
              MOV     LCD_DATA,INT_ADDR
              ACALL   HEX2LCD
              MOV     LCD_ADDR,#045H
              ACALL   SET_ADDR_LCD
              MOV     LCD_DATA,DATA_READ
              ACALL   HEX2LCD
              LJMP    MAIN
```

(โปรแกรม SLAVE LCD ต่อ)

```
DISP_WRITE:   JB      LCD_BIT3,DISP_WRITE1
              MOV     FLAG1,#00001000B
              MOV     LCD_ADDR,#000H
              ACALL   SET_ADDR_LCD
              MOV     DPTR,#SCR_WRITE
              ACALL   WRLINE_LCD
              MOV     LCD_ADDR,#006H
              ACALL   SET_ADDR_LCD
              MOV     LCD_DATA,DATA_ADDR
              ACALL   HEX2LCD
              MOV     LCD_ADDR,#040H
              ACALL   SET_ADDR_LCD
              MOV     LCD_DATA,INT_ADDR
              ACALL   HEX2LCD
DISP_WRITE1:  MOV     LCD_ADDR,#045H
              ACALL   SET_ADDR_LCD
              MOV     LCD_DATA,DATA_WR
              ACALL   HEX2LCD
              LJMP    MAIN
DISP_TIME:    JB      LCD_BIT4,DISP_TIME1
              MOV     FLAG1,#00010000B
              MOV     LCD_ADDR,#000H
              ACALL   SET_ADDR_LCD
              MOV     DPTR,#SCR_TIME
              ACALL   WRLINE_LCD
DISP_TIME1:   MOV     LCD_ADDR,#040H
              ACALL   SET_ADDR_LCD
              MOV     LCD_DATA,HOURS
              ACALL   HEX2LCD
              MOV     LCD_ADDR,#043H
              ACALL   SET_ADDR_LCD
              MOV     LCD_DATA,MINUTES
              ACALL   HEX2LCD
              MOV     LCD_ADDR,#046H
              ACALL   SET_ADDR_LCD
```

(โปรแกรม SLAVE LCD ต่อ)

```
        MOV   LCD_DATA,SECONDS
        ACALL HEX2LCD
        AJMP  MAIN
DISP_SETM:  JB    LCD_BIT5,DISP_SETM1
        MOV   FLAG1,#00100000B
        MOV   LCD_ADDR,#000H
        ACALL SET_ADDR_LCD
        MOV   DPTR,#SCR_SETM
        ACALL WRLINE_LCD
DISP_SETM1: MOV   LCD_ADDR,#045H
        ACALL  SET_ADDR_LCD
        MOV   LCD_DATA,DATA_WR
        ACALL HEX2LCD
        LJMP  MAIN
DISP_SETH:  JB    LCD_BIT6,DISP_SETH1
        MOV   FLAG1,#01000000B
        MOV   LCD_ADDR,#000H
        ACALL SET_ADDR_LCD
        MOV   DPTR,#SCR_SETH
        ACALL WRLINE_LCD
DISP_SETH1: MOV   LCD_ADDR,#045H
        ACALL SET_ADDR_LCD
        MOV   LCD_DATA,DATA_WR
        ACALL HEX2LCD
        LJMP  MAIN
;*************************************
;*************************************
ADDR_MASTER: PUSH  ACC
        MOV   A,#0
        MOV   R1,#8
ADDR_MASTER_1:  ACALL I2C_CLK_HIGH
        MOV   C,SDA
        RLC   A
        ACALL I2C_CLK_LOW
        DJNZ  R1,ADDR_MASTER_1
```

(โปรแกรม SLAVE LCD ต่อ)

```
        MOV   I2C_ADDR,A
        POP   ACC
        RET
;*************************************
MASTER_READ: PUSH  ACC
        SETB  I2C_ACK
        MOV   P1,#0FFH
        MOV   A,P1
        MOV   R1,#8
MASTER_READ_1:  RLC   A
        MOV   SDA,C
        ACALL I2C_CLK_HIGH
        ACALL I2C_CLK_LOW
        DJNZ  R1, MASTER_READ_1
        SETB  SDA
        ACALL I2C_CLK_HIGH
        JB    SDA,MASTER_READ_2
        CLR   I2C_ACK
MASTER_READ_2:  ACALL I2C_CLK_LOW
        POP   ACC
        RET
;*************************************
MASTER_WRITE:   PUSH  ACC
        MOV   A,#0
        MOV   R1,#8
MASTER_WRITE_1: ACALL I2C_CLK_HIGH
        MOV   C,SDA
        RLC   A
        ACALL I2C_CLK_LOW
        DJNZ  R1,MASTER_WRITE_1
        MOV   I2C_DATA,A
        POP   ACC
        RET
;*************************************
```

(โปรแกรม SLAVE LCD ต่อ)

```
;*******************************
;          I2C  Acknowledge
;*******************************
I2C_ACK_BIT:   ACALL  I2C_CLK_HIGH
               CLR    SDA
               ACALL  I2C_CLK_LOW
               SETB   SDA
               RET
;*******************************
;          I2C  Not Acknowledge
;*******************************
I2C_NACK_BIT:  ACALL  I2C_CLK_HIGH
               SETB   SDA
               ACALL  I2C_CLK_LOW
               RET
;************************************
I2C_CLK_HIGH:  SETB   ERR_BIT
               MOV    R2,#1AH
I2C_CLK_HIGH_1: JB    SCL, I2C_CLK_HIGH_2
               DJNZ   R2, I2C_CLK_HIGH_1
               CLR    ERR_BIT
I2C_CLK_HIGH_2:       RET
;************************************
I2C_CLK_LOW:   SETB    ERR_BIT
               MOV     R2,#1AH
I2C_CLK_LOW_1: JNB     SCL,I2C_CLK_LOW_2
               DJNZ    R2,I2C_CLK_LOW_1
               CLR     ERR_BIT
I2C_CLK_LOW_2:         RET
;************************************
I2C_DELAY:     MOV     R2,#10H
I2C_DELAY_1:           NOP
                       NOP
               DJNZ    R2, I2C_DELAY_1
               RET
```

(โปรแกรม SLAVE LCD ต่อ)

```
;************************************
;          ***    LCD   ***
;************************************
;          HEX Code to show LCD
;          I/P:      LCD_DATA
;************************************
HEX2LCD:       PUSH    ACC
               MOV     A,LCD_DATA
               MOV     B,#16
               DIV     AB
               ADD     A,#030H
               MOV     LCD_DATA,A
               ACALL   HEX_CHK
               ACALL   WRCHAR_LCD
               MOV     A,B
               ADD     A,#030H
               MOV     LCD_DATA,A
               ACALL   HEX_CHK
               ACALL   WRCHAR_LCD
               POP     ACC
               RET
HEX_CHK:       MOV     A,LCD_DATA
               CJNE    A,#03AH,CHK_OTHER
CHK_OTHER:     JNC     CONV_2_ALPHA
               RET
CONV_2_ALPHA:          ADD     A,#7
               MOV     LCD_DATA,A
               RET
;************************************
;          LCD Initialize
;************************************
INIT_LCD:      ACALL   DELAY_100ms
               CLR     LCD_RS
               MOV     P0,#00111000B
               ACALL   LCD_CLK
```

(โปรแกรม SLAVE LCD ต่อ)

```
        ACALL  DELAY_10ms
        MOV    P0,#00111000B
        ACALL  LCD_CLK
        ACALL  LCD_OFF
        ACALL  LCD_CLR
        MOV    P0,#00000110B
        ACALL  LCD_CLK
        ACALL  LCD_HOME
;****************************************
;       LCD  Clear Display
;****************************************
LCD_CLR:    CLR    LCD_RS
            MOV    P0,#00000001B
            ACALL  LCD_CLK
            RET
;****************************************
;       LCD  Return Home
;****************************************
LCD_HOME:   CLR    LCD_RS
            MOV    P0,#00000010B
            ACALL  LCD_CLK
            RET
;****************************************
;       LCD Display Off
;****************************************
LCD_OFF:    CLR    LCD_RS
            MOV    P0,#00001000B
            ACALL  LCD_CLK
            RET
;****************************************
;       LCD Clk
;****************************************
LCD_CLK:    SETB   LCD_EN
            ACALL  LCD_DELAY
            CLR    LCD_EN
```

(โปรแกรม SLAVE LCD ต่อ)

```
            ACALL  LCD_DELAY
            RET
;****************************************
;       LCD Display On
;****************************************
LCD_ON:     CLR    LCD_RS
            MOV    P0,#00001100B
            ACALL  LCD_CLK
            RET
;****************************************
;       LCD Cursor  On
;****************************************
LCD_BLINK:  CLR    LCD_RS
            MOV    P0,#00001111B
            ACALL  LCD_CLK
            RET
;****************************************
;       LCD Left Shift Display
;****************************************
LCD_LSHF:   CLR    LCD_RS
            MOV    0,#00011000B
            ACALL  LCD_CLK
            RET
;****************************************
;       LCD Right Shift Display
;****************************************
LCD_RSHF:   CLR    LCD_RS
            MOV    P0,#00011100B
            ACALL  LCD_CLK
            RET
;****************************************
;   Set  LCD  Address
; I/P :  LCD_ADDR
;****************************************
SET_ADDR_LCD:       CLR    LCD_RS
```

(โปรแกรม SLAVE LCD ต่อ)

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

        MOV     LCD_ADDR,#040H
        ACALL   SET_ADDR_LCD

WRLINE_LCD_2: SETB  LCD_RS
        CLR     A
        MOVC    A,@A+DPTR
        MOV     P0,A
```

(โปรแกรม SLAVE LCD ต่อ)

```
        ACALL   LCD_CLK
        INC     DPTR
        INC     R0
        CJNE    R0,#16,WRLINE_LCD_2
        ACALL   LCD_ON
        RET
;************************************
;       Write 3 Character from Rom
;       I/P:    LCD_PTR
;************************************
WR3CHAR_LCD:    MOV     R0,#0
        MOV     A,LCD_PTR
        DEC     A
        MOV     B,#3
        MUL     AB
WR3CHAR_LCD_1:  SETB    LCD_RS
        MOVC    A,@A+DPTR
        MOV     P0,A
        ACALL   LCD_CLK
        INC     DPTR
        INC     R0
        MOV     A,LCD_PTR
        DEC     A
        MOV     B,#3
        MUL     AB
        CJNE    R0,#8,WRLINE_LCD_1
        LCALL   LCD_ON
        RET
;********************************
;Dummy Delay time LCD_DELAY,   10ms, 100ms, ls
;********************************
LCD_DELAY:    MOV   R7,#002
LCD_DELAY_1:  MOV   R6,#0E6H
LCD_DELAY_2:  NOP
        NOP
```

(โปรแกรม SLAVE LCD ต่อ)

```
                DJNZ    R6,LCD_DELAY_2
                DJNZ    R7,LCD_DELAY_1
                RET
DELAY_10ms:     MOV     R7,#010
DELAY_10ms_1:   MOV     R6,#0E6H
DELAY_10ms_2:   NOP
                NOP
                DJNZ    R6,DELAY_10ms_2
                DJNZ    R7,DELAY_10ms_1
                RET
DELAY_100ms:    MOV     R7,#100
DELAY_100ms_1:          MOV     R6,#0E6H
DELAY_100ms_2:          NOP
                NOP
                DJNZ    R6,DELAY_100ms_2
                DJNZ    R7,DELAY_100ms_1
                RET
DELAY_LS:       MOV     R5,#100
DELAY_ls_1:     ACALL   DELAY_10ms
                DJNZ    R5,DELAY_ls_1
                RET
;*****************************************
TITLE_1:                DB      'I/O I2C Display.'
SCR_READ:       DB      'Read @   H : H'
SCR_TIME:       DB      'TIME:   : : '
SCR_WRITE:      DB      'Set  @   H : H'
SCR_ADDR:       DB      'SLAVE ID:    H'
SCR_SETM:       DB      'Set Minutes?:  H'
SCR_SETH:       DB      'Set Hours? :  H'
;*****************************************
                END
```

โปรแกรมที่ 3 ส่วนของตัว SLAVE 2

(I/O EXPENDER MCS-51)

```
;*******************************
;PROGRAM  OF A  I2C – BASED
;   EMBEDDED - PROCESSOR
;            SYSTEM
;   (SLAVE I/O EXPENDER)
;*******************************
;          Define Port&Pin Name
;*******************************
SDA             BIT     P3.1
SCL             BIT     P3.0
OUT_INT         BIT     P3.7
OUT_W           BIT     P3.6
;*******************************
;          Define User Registor
;*******************************
FLAG            EQU     2FH
I2C_ACK         BIT     FLAG.0
ERR_BIT         BIT     FLAG.1
I2C_ADDR        EQU     26H
I2C_DATA        EQU     27H
POINTER         EQU     28H
SLAVE_ID_R      EQU     29H
SLAVE_ID_W      EQU     2AH
;***********************************************
        ORG 0000H
                MOV     P0,#11111111B
                MOV     P1,#11111111B
                MOV     P2,#10000001B
                MOV     P3,#11111111B
                MOV     POINTER,#030H
                ;************************
                MOV     A,P3
                RR      A
```

(โปรแกรม SLAVE I/O EXPENDER ต่อ)

```
                ANL     A,#0FH
                CLR     ACC.0
                MOV     SLAVE_ID_W,A
                SETB    ACC.0
                MOV     SLAVE_ID_R,A
                ;************************
MAIN:           CLR     OUT_W
                SETB    SDA
                SETB    SCL
;*********************************************
START_BIT:      JNB     SDA,$
START_BIT_1:    JNB     SCL,START_BIT
                JB      SDA,START_BIT_1
                JNB     SCL,START_BIT
                ACALL   I2C_CLK_LOW
                JNB     ERR_BIT,START_BIT
                JB      SDA,START_BIT
                ACALL   ADDR_MASTER
                MOV     A,I2C_ADDR
                CJNE    A,SLAVE_ID_R,ADDRESS_1
                ACALL   I2C_ACK_BIT
                ACALL   MASTER_WRITE
                ACALL   I2C_ACK_BIT
                MOV     POINTER,I2C_DATA
                MOV     30H,P0
                MOV     31H,P1
                MOV     32H,P2
                MOV     R0,POINTER
                MOV     I2C_DATA,@R0
                ACALL   MASTER_READ
                AJMP    STOP_BIT
ADDRESS_1:      CJNE    A,SLAVE_ID_W,MAIN
                ACALL   I2C_ACK_BIT
                ACALL   MASTER_WRITE
                ACALL   I2C_ACK_BIT
```

(โปรแกรม SLAVE I/O EXPENDER ต่อ)

```
        MOV   POINTER,I2C_DATA
        ACALL MASTER_WRITE
        SETB  OUT_W
        ACALL I2C_NACK_BIT
        MOV   R0,POINTER
        MOV   A,POINTER
        CLR   C
        SUBB  A,#30H
        JC    STOP_BIT
        MOV   A,POINTER
        CJNE  A,#30H,PORT_1
        MOV   P0,I2C_DATA
        MOV   @R0,P0
        AJMP  STOP_BIT
PORT_1: CJNE  A,#31H,PORT_2
        MOV   P1,I2C_DATA
        MOV   @R0,P1
        AJMP  STOP_BIT
PORT_2: CJNE  A,#32H,MEMORY_1
        MOV   P2,I2C_DATA
        MOV   @R0,P2
        AJMP  STOP_BIT
MEMORY_1: MOV @R0,I2C_DATA
;*************************************
STOP_BIT: CLR   OUT_W
        ACALL I2C_CLK_HIGH
        JB    SDA,STOP_BIT3
        MOV   R2,#1AH
STOP_BIT1: JB   SDA,STOP_BIT2
        DJNZ  R2,STOP_BIT1
STOP_BIT2: SETB I2C_ACK
STOP_BIT3: AJMP MAIN
;*************************************
```

(โปรแกรม SLAVE I/O EXPENDER ต่อ)

```
;*************************************
ADDR_MASTER: PUSH ACC
        MOV   A,#0
        MOV   R1,#8
ADDR_MASTER_1: ACALL I2C_CLK_HIGH
        MOV   C,SDA
        RLC   A
        ACALL I2C_CLK_LOW
        DJNZ  R1,ADDR_MASTER_1
        MOV   I2C_ADDR,A
        POP   ACC
        RET
;*************************************
MASTER_READ: PUSH ACC
        MOV   A,I2C_DATA
        SETB  I2C_ACK
        MOV   R1,#8
MASTER_READ_1: RLC A
        MOV   SDA,C
        ACALL I2C_CLK_HIGH
        ACALL I2C_CLK_LOW
        DJNZ  R1, MASTER_READ_1
        SETB  SDA
        ACALL I2C_CLK_HIGH
        JB    SDA,MASTER_READ_2
        CLR   I2C_ACK
MASTER_READ_2: ACALL I2C_CLK_LOW
        POP   ACC
        RET
;*************************************
MASTER_WRITE: PUSH ACC
        MOV   A,#0
        MOV   R1,#8
MASTER_WRITE_1: ACALL I2C_CLK_HIGH
        MOV   C,SDA
```

(โปรแกรม SLAVE I/O EXPENDER ต่อ)

```
                RLC     A
                ACALL   I2C_CLK_LOW
                DJNZ    R1,MASTER_WRITE_1
                MOV     I2C_DATA,A
                POP     ACC
                RET
;*******************************
;       I2C  Acknowledge
;*******************************
I2C_ACK_BIT:    ACALL   I2C_CLK_HIGH
                CLR     SDA
                ACALL   I2C_CLK_LOW
                SETB    SDA
                RET
;*******************************
;       I2C  Not Acknowledge
;*******************************
I2C_NACK_BIT:   ACALL   I2C_CLK_HIGH
                SETB    SDA
                ACALL   I2C_CLK_LOW
                RET
;*********************************
I2C_CLK_HIGH:   SETB    ERR_BIT
                MOV     R2,#1AH
I2C_CLK_HIGH_1: JB      SCL, I2C_CLK_HIGH_2
                DJNZ    R2, I2C_CLK_HIGH_1
                CLR     ERR_BIT
I2C_CLK_HIGH_2:         RET
;*********************************
I2C_CLK_LOW:    SETB    ERR_BIT
                MOV     R2,#1AH
I2C_CLK_LOW_1:  JNB     SCL,I2C_CLK_LOW_2
                DJNZ    R2,I2C_CLK_LOW_1
                CLR     ERR_BIT
I2C_CLK_LOW_2:          RET
```

(โปรแกรม SLAVE I/O EXPENDER ต่อ)

```
;*********************************
I2C_DELAY:      MOV     R2,#10H
I2C_DELAY_1:    NOP
                NOP
                DJNZ R2, I2C_DELAY_1
                RET
;*********************************
                END
```