ปริญญานิพนธ์

ชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458

PIC18F458 MICROCONTROLLER EXPERIMENT SETS

นายคมสันต์ แวงวรรณ
นายฉาวร แสงใส
นายอภิวัธน์ ศรีบรรเทา

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์
ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2546

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ ชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458

PIC18F458 Microcontroller Experiment Sets

ชื่อนักศึกษา

1. นายคมสันต์ แวงวรรณ รหัสประจำตัว 45035333
2. นายถาวร แสงใส รหัสประจำตัว 45035340
3. นายอภิวัธน์ ศรีบรรเทา รหัสประจำตัว 45035370

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** อิเล็กทรอนิกส์และคอมพิวเตอร์

อาจารย์ที่ปรึกษา อาจารย์สุรพงษ์ สิริพงศ์ดี

อาจารย์ที่ปรึกษาร่วม ผศ.วิสุทธิ์ อธิพรธรรม

คณะกรรมการสอบปริญญานิพนธ์	ลายมือชื่อ
1. อาจารย์สุระชัย พิมพ์สาลี	
2. อาจารย์สมชาย หมื่นสายญาติ	
3. อาจารย์สุรพงษ์ สิริพงศ์ดี	
4. อาจารย์อมรชัย ชัยชนะ	
5. อาจารย์ปิยะ ศุภวราสุวัฒน์	

วัน/เดือน/ปีที่สอบ วันพุธที่ 31 มีนาคม พ.ศ. 2547 เวลา 18.00 น.

สถานที่สอบ ห้อง ค.311 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.สุรสิทธิ์ ราตรี)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่.....เดือน.....พ.ศ.....

ปริญญานิพนธ์

เรื่อง ชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458

PIC18F458 Microcontroller Experiment Sets

วัตถุประสงค์

1. เพื่อศึกษาระบบการทำงานของไมโครคอนโทรลเลอร์ PIC18F458
2. เพื่อออกแบบวงจรในชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458
3. เพื่อสร้างใบงานในการทดลองของชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458
4. เพื่อทดลองเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ PIC18F458
5. เพื่อนำชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 ไปใช้เป็นสื่อการเรียนการสอน

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รู้ระบบการทำงานของไมโครคอนโทรลเลอร์ PIC18F458
2. ได้วงจรการใช้งานในชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458
3. ได้ใบงานที่ใช้ในการทดลองในชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458
4. ได้โปรแกรมการทดลองในชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458
5. ได้ชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 เป็นสื่อการเรียนการสอน

ชื่อหัวข้อ	ชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458
นักศึกษา	นายคมสันต์ แวงวรรณ
	นายถาวร แสงใส
	นายอภิวัธน์ ศรีบรรเทา
อาจารย์ที่ปรึกษา	อาจารย์สุรพงษ์ สิริพงศ์ดี
อาจารย์ที่ปรึกษาร่วม	ผศ.วิสุทธ์ อธิพรธรรม
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชา	อิเล็กทรอนิกส์และคอมพิวเตอร์
ปีการศึกษา	2546

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้นำเสนอการออกแบบและการสร้างชุดทดลอง ไมโครคอนโทรลเลอร์ PIC18F458 ชุดทดลองแบ่งเป็น 3 ส่วน คือ ส่วนที่หนึ่ง เป็นส่วนของ ชุดอุปกรณ์ทดลองต่างๆ โดยแบ่งแยกออกเป็นโมดูล 9 โมดูล คือ โมดูลหลัก PIC-ICSP โมดูลสวิตช์พื้นฐาน โมดูลแสดงผลแอลอีดี โมดูลแสดงผลแอลอีดีตัวเลขเจ็ดส่วน โมดูลแสดงผลแอลซีดี โมดูลสื่อสารข้อมูลอนุกรม โมดูลเชื่อมต่ออุปกรณ์ RTC/SPEAKER โมดูลขับสเตปปิ้งมอเตอร์ โมดูลพัลส์เจเนอร์เรเตอร์ ส่วนที่สอง เป็นส่วนของใบงานที่ใช้ประกอบการทดลองและ ส่วนที่สาม เป็นส่วนของคู่มือการใช้งานโปรแกรมประกอบการทดลอง คือ คู่มือการใช้งานโปรแกรม MPLAB และโปรแกรม Epic Win ชุดทดลองนี้จึงช่วยให้ เข้าใจหลักการทำงานของไมโครคอนโทรลเลอร์ตระกูล PIC และ PIC18F458 ได้ง่ายยิ่งขึ้น และ สามารถประยุกต์ใช้งานในระบบควบคุมทางอุตสาหกรรมด้านต่างๆ ได้นอกจากนี้แล้วยังสามารถนำ ชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 ไปใช้เป็นสื่อการเรียนการสอนสำหรับนักเรียน นักศึกษา และบุคคลทั่วไปที่สนใจได้

Thesis Title	PIC18F458 Microcontroller Experiment Sets
Students	Mr.Komsan Wangwan
	Mr.Tavorn Saengsai
	Mr.Apiwat Sribantao
Advisor	Mr.Surapong Siripongdee
Co-Advisor	Assist.Prof.Wisuit Atiporntum
Education Level	Bachelor of Science in Industrial Education
Program in	Electronics and Computer
Academic Year	2003

ABSTRACT

This thesis this presents a design and implementation of Microcontroller PIC18F458 Experiment Sets. The project can be divided 3 part : First is equipment testing which has separated 9 module, main module (PIC-ICSP), Basic Swicth Module, LED Display Module, LCD Display Module, Seven Segment Display Module, RTC/Speaker Interfacing Module, Serial Interfacing Module, Stepping Motor Driver Module and Pulse Generator Module. The second part is sheer working. And third is manual program for testing which had MPLAB manual program and Epic win program that they are more easily for working and can bring use to control with an industries. Besides of them can take microcontroller PIC18F458 for learning that suitable for student any general people intersted.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ลุล่วงไปได้ด้วยดีอันเนื่องมาจากความร่วมมือของสมาชิกในกลุ่มทุกท่าน ขอขอบคุณอาจารย์สุรพงษ์ สิริพงศ์ดี และคณาจารย์ภาควิชาครุศาสตร์วิศวกรรมทุกท่านที่ให้ความอนุเคราะห์เครื่องมือ และอุปกรณ์ รวมทั้งให้คำแนะนำ แนวความคิด ความรู้ต่างๆ แนวทางแก้ไขปัญหาในการจัดทำปริญญานิพนธ์ ขอขอบคุณห้องสมุดคณะครุศาสตร์อุตสาหกรรม สำนักหอสมุดกลาง ที่ช่วยอำนวยความสะดวกและเอื้อเฟื้อสถานที่ในการค้นคว้าข้อมูล สุดท้ายที่ควรระลึกถึงอย่างยิ่ง บิดาและมารดาที่เป็นผู้ให้การสนับสนุนด้านการศึกษา และเป็นผู้ให้กำลังใจด้วยดีตลอดมา ตั้งแต่อดีตจนถึงปัจจุบัน

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ไมโครคอนโทรลเลอร์ (Microcontroller) ได้เข้ามามีบทบาทเป็นอย่างมากในระบบควบคุมทางอุตสาหกรรม และระบบอิเล็กทรอนิกส์ ไมโครคอนโทรลเลอร์ PIC (Perpheral Interface Contorller) ก็เป็นไมโครคอนโทรลเลอร์อีกตระกูลหนึ่ง ที่ได้รับความนิยมเป็นอย่างมากในปัจจุบัน ที่นำมาสร้างและพัฒนาระบบควบคุมทางอุตสาหกรรม เพื่อพัฒนาผลิตภัณฑ์หรือสินค้าในด้านอุตสาหกรรมต่างๆ ให้ช่วยเพิ่มผลผลิตจำนวนมากๆ ไมโครคอนโทรลเลอร์ PIC18F458 เป็นไมโครคอนโทรลเลอร์อีกเบอร์หนึ่งของไมโครคอนโทรลเลอร์ตระกูล PIC ที่มีความเร็วในการทำงานที่สูงถึง 40 เมกะเฮิรตซ์ (MHz) โดยเป็นไมโครคอนโทรลเลอร์ที่มีสถาปัตยกรรมและโครงสร้างชุดคำสั่งเป็นแบบ 16 บิต ซึ่งมีฟังก์ชั่น (Function) และโมดูล (Module) การใช้งานต่างๆ อยู่ภายในตัวทำให้เหมาะต่อการใช้งาน จึงถูกเลือกมาเพื่อศึกษาเรียนรู้และประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูลนี้

ดังนั้นคณะผู้จัดทำจึงได้จัดทำชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 ขึ้นเพื่อใช้ในการศึกษาระบบการทำงานของไมโครคอนโทรลเลอร์ตระกูล PIC ซึ่งชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 นี้จะช่วยให้นักศึกษาและผู้ที่สนใจเข้าใจหลักการทำงานของไมโครคอนโทรลเลอร์ ตระกูล PIC ได้ง่ายขึ้น

1.2 ขีดความสามารถของโครงงาน

โครงงานนี้มีขีดความสามารถดังนี้

1. มีใบงานการทดลอง 17 ใบงาน
2. มีเฉลยใบงานทั้ง 17 ใบงาน
3. มีคู่มือประกอบการทดลอง ดังนี้
   - 3.1 คู่มือการใช้โปรแกรม MPLAB
   - 3.2 คู่มือการใช้โปรแกรม Epic Win
   - 3.3 คู่มือการใช้งานโมดูลการทดลอง

4. ชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 สามารถใช้ได้กับไมโครคอนโทรลเลอร์ตระกูล PIC ในรุ่น 40 ขาในแบบตัวถังแบบ PDIP ได้

5. สามารถดาวน์โหลดซอร์สโค้ดลงในไมโครคอนโทรลเลอร์ได้โดยตรงบนบอร์ดทดลองได้

6. สามารถนำชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 ไปใช้เป็นสื่อการเรียนการสอนวิชาไมโครคอนโทรลเลอร์ได้

7. ไมโครคอนโทรลเลอร์ PIC18F458 สามารถโปรแกรมข้อมูลได้นับพันครั้ง

1.3 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้แบ่งออกเป็นบทต่างๆ เพื่อความสะดวกต่อการศึกษาและทำความเข้าใจ ในแต่ละบทประกอบด้วยเนื้อหาดังต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปริญญานิพนธ์ขีดความสามารถของโครงงานและเนื้อหาในบทต่างๆโดยสังเขป

บทที่ 2 ทฤษฎีและหลักการ ประกอบด้วยเนื้อหาดังต่อไปนี้ คือ ความรู้เบื้องต้นเกี่ยวกับไมโครคอนโทรลเลอร์ PIC18F458 การจัดขาของ PIC18F458 สถาปัตยกรรมและโครงสร้างของไมโครคอนโทรลเลอร์ PIC18F458 โหมดสัญญาณนาฬิกา การรีเซตของไมโครคอนโทรลเลอร์ PIC18F458 จังหวะและไซเคิลการทำงานของไมโครคอนโทรลเลอร์ PIC18F458 หน่วยความจำ รีจิสเตอร์หลักของ PIC18F458 ชุดคำสั่งของ PIC18F458 และการเข้าถึงรีจิสเตอร์

บทที่ 3 การออกแบบ การสร้าง และการทำงาน กล่าวถึงเนื้อหาเกี่ยวกับ การออกแบบทางด้านฮาร์ดแวร์ ผังวงจรต่างๆ ที่ใช้ในโครงงาน ซึ่งแบ่งออกเป็น การออกแบบโมดูลหลัก PIC- ICSP โมดูลแสดงผลแอลอีดี โมดูลแสดงผลตัวเลขเจ็ดส่วน โมดูลแปลงสัญญาณดิจิตอลเป็นแอนะลอกและแอนะลอกเป็นดิจิตอล โมดูลสวิตช์อินพุตและเมตริกซ์สวิตช์ โมดูลสื่อสารข้อมูลอนุกรม โมดูลขับสเตปปิ้งมอเตอร์ โมดูลเชื่อมต่ออุปกรณ์ RTC และ SPEAKER โมดูลแสดงผลแอลซีดี โมดูลพัลส์เจเนอเรเตอร์

บทที่ 4 การทดลองและผลการทดลอง ประกอบด้วย การทดลองและผลการทดลองใช้งานของโมดูลต่างๆ คือ โมดูลหลัก PIC- ICSP โมดูลแสดงผลแอลอีดี โมดูลแสดงผลตัวเลขเจ็ดส่วน โมดูลแปลงสัญญาณดิจิตอลเป็นแอนะลอกและแอนะลอกเป็นดิจิตอล โมดูลสวิตช์อินพุตและเมตริกซ์สวิตช์ โมดูลสื่อสารข้อมูลอนุกรม โมดูลขับสเตปปิ้งมอเตอร์ โมดูลเชื่อมต่ออุปกรณ์ RTC และ SPEAKER โมดูลแสดงผลแอลซีดี โมดูลพัลส์เจเนอเรเตอร์ โมดูลแสดงผลแอลอีดีเมตริกซ์

บทที่ 5 บทสรุป ปัญหา แนวทางแก้ไขและพัฒนา ขั้นการสรุปผล ในการจัดทำโครงงาน ปัญหาที่เกิดขึ้น และได้เสนอแนะแนวทาง ในการพัฒนาให้มีประสิทธิภาพมากยิ่งขึ้น

ภาคผนวก ก เครื่องต้นแบบ

ภาคผนวก ข วงจร และแผ่นวงจรพิมพ์

ภาคผนวก ค รายการอุปกรณ์

ภาคผนวก ง แผนผังการทำงานและรหัสต้นฉบับของโปรแกรม

ภาคผนวก จ ใบงานการทดลอง

ภาคผนวก ฉ คู่มือการใช้งาน

ภาคผนวก ช รายละเอียดและคุณสมบัติของอุปกรณ์

ประวัติผู้แต่ง

บทที่ 2
ทฤษฎีและหลักการ

2.1 ความรู้เบื้องต้นเกี่ยวกับไมโครคอนโทรลเลอร์ PIC18F458

ไมโครคอนโทรลเลอร์ PIC18F458 เป็นไมโครคอนโทรลเลอร์อีกเบอร์หนึ่งของตระกูล PIC (Peripherral Interface Contorller) ในปัจจุบันซึ่งมีศักยภาพในการทำงานสูงและในคุณสมบัติของไมโครคอนโทรลเลอร์เบอร์นี้คือเพียบพร้อมไปด้วยทรัพยากรณ์หรือฟังก์ชั่น (Function) การใช้งานต่างๆ ไว้ภายในตัวมันเอง เช่น มีโมดูล (Module) แปลงสัญญาณแอนะลอกเป็นดิจิตอล (Analog To Digital Converter), USART, SPI, I^2C, PWM อื่นๆ มีโมดูลที่เพิ่มเติมขึ้นมาใหม่คือ Can Modle, ECCP ซึ่งเป็นคุณสมบัติใหม่ของไมโครคอนโทรลเลอร์ตระกูล PIC โดยมีในไมโครคอนโทรลเลอร์ PIC18F458 และเบอร์อื่นๆ ของ PIC18FXXX และยังเหมาะต่อการใช้งานตรงที่หน่วยความจำโปรแกรมเป็นหน่วยความจำแบบเฟลช (Flash Program Memory) ซึ่งสามารถเขียนและลบข้อมูลได้ด้วยสัญญาณไฟฟ้าได้นับหลายพันครั้งข้อเด่นอีกประการหนึ่งของไมโครคอนโทรลเลอร์เบอร์นี้ในเรื่องของความเร็ว PIC18F458 สามารถทำงานได้ที่ความถี่สัญญาณนาฬิกาสูงถึง 40 เมกะเฮิรตซ์ (MHz) และมีวงจร PLL (Phase Lock Loop) ซึ่งเป็นวงจรคูณความถี่อยู่ภายในโดยสามารถเลือกโดยการโปรแกรมทางซอฟต์แวร์ ซึ่งสามารถคูณค่าความถี่ที่รับเข้ามาได้ถึง 4 เท่าของสัญญาณนาฬิกาจากภายนอกทั้งยังทำงานในลักษณะไปป์ไลน์ (Pipe Line) ทำให้มีความเร็วในการทำงานมากกว่าซีพียูทั่วไปที่ค่าความถี่เดียวกันโดยในลักษณะการทำงานจะใช้สัญญาณนาฬิกาเพียง 1 หรือ 2 ไซเคิล (Cycle) ต่อคำสั่งเท่านั้นและหน่วยความจำไม่ถูกแบ่งเป็นเพจ (Page) อีกต่อไปการเขียนโปรแกรมจึงง่ายโดยไม่ต้องเลือกแบงก์ (Bank)

2.1.1 คุณสมบัติของไมโครคอนโทรลเลอร์ PIC18F458

สามารถสรุปคุณสมบัติของไมโครคอนโทรลเลอร์ PIC18F458 ได้ดังต่อไปนี้

1. มีชุดคำสั่ง 75 คำสั่ง
2. ซีพียูเป็นแบบ RISC (Reduce Insturction Set Computer)
3. เป็นซีพียู 16 บิต
4. หน่วยความจำ SRAM 1536 ไบต์ (Byte)
5. หน่วยความจำโปรแกรม 32 กิโลไบต์ (KByte)
6. หน่วยความจำอีอีพรอม 256 ไบต์
7. รับความถี่สัญญาณนาฬิกาตั้งแต่ไฟตรงถึง 40 เมกะเฮิรตซ์

8. มีวงจร PLL (Phase Lock Loop) ภายในคูณค่าความถี่ของสัญญาณนาฬิกา 4 เท่าของค่าสัญญาณนาฬิกาอินพุต

9. ตอบสนองการอินเตอร์รัพต์ (Interrupt) ได้ถึง 21 แหล่ง

10. มีขารับสัญญาณอินเตอร์รัพต์จากภายนอก 3 ขา คือ RB0/INT0, RB1/INT1, RB2 /INT2

11. เลือกลำดับความสำคัญของการอินเตอร์รัพต์จากอุปกรณ์ต่อพ่วงได้

12. กระแสซิงก์ (Synchronous) และซอร์ส (Source) ของพอร์ต (Port) สูงสุด 25 มิลลิแอมป์

13. มีโมดูลไทเมอร์ 4 ตัวดังนี้

13.1 ไทเมอร์ 0 ขนาด 8/16 บิต เป็นไทเมอร์/เคาน์เตอร์ พร้อมปรีสเกลเลอร์ 8 บิต

13.2 ไทเมอร์ 1 ขนาด 16 บิต/เป็นไทเมอร์/เคาน์เตอร์ พร้อมปรีสเกลเลอร์ 8 บิต

13.3 ไทเมอร์ 2 ขนาด 8 บิต มีปรีสเกลเลอร์ โพสต์สเกลเลอร์ และ รีจิสเตอร์คาบเวลา (Period Register) เป็นตัวเปรียบค่ากับไทเมอร์ 2 อยู่ภายในตัว

13.4 ไทเมอร์ 3 ขนาด 16 บิต เป็นไทเมอร์ / เคาน์เตอร์

14. มีโมดูล CCP (Capture/Compare/PWM) 1 ชุด ส่วนตรวจจับสัญญาณ (Capture) ขนาด 16 บิต ความระเอียดสูงสุด 6.25 นาโนวินาที ส่วนเปรียบเทียบสัญญาณ (Compare) ความระเอียดสูงสุด 100 นาโนวินาที และส่วนวงจรมอดูเลชั่นทางความกว้างของพัลส์ (Pulse Width Modulation : PWM) ความระเอียดสูงสุด 10 บิต

15. มีโมดูล ECCP (Enhanced Capture/Compare/PWM) 1 ชุด ทำงานคล้ายกับโมดูล CCP แต่จะต่างกันตรงที่จะใช้งานในการควบคุมมอเตอร์

16. มีโมดูล MSSP (Master Syncronouse Serial Port) ใช้งานเป็นวงจรเชื่อมต่ออุปกรณ์อนุกรมทำงานได้ 2 โหมด คือ SPI และ I2C

17. มีโมดูลการสื่อสารอนุกรม USART (Universal Syncronous Asyncronous Receiver Transmitter)

18. มีโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอล (Analog to Digital Converter) อยู่ภายในความระเอียด 10 บิต 8 แชนแนล

19. มีโมดูลเปรียบเทียบแรงดันแอนะลอก (Comparater Voltage Referrent Module) ภายใน

20. สามารถเลือกโหมดการป้องกันข้อมูลได้ (Code Protection)

21. ทำงานที่ไฟเลี้ยง 2.0 V ถึง 5.5 V

22. สามารถโปรแกรมด้วยแรงดันไฟต่ำได้ LVP (Low Voltage Programing)

23. ฟังก์ชั่นการโปรแกรมเป็นแบบ ICSP (In Circuit Serial Programing)

24. มีเพาเวอร์ออนรีเซต (Power On Reset : POR) เพาเวอร์อัปไทเมอร์ (Power Up Timer : PWRT) และ ออสซิลเลเตอร์สตาร์ตอัปไทเมอร์ (Oscillator Start Up Timer : OST)

25. มีวอตช์ด็อกไทเมอร์ (Watch Dog Timer : WDT) ทำให้มีความเชื่อมั่นในการทำงานสูง

26. มีโหมดในการประหยัดพลังงาน (Sleep Mode)

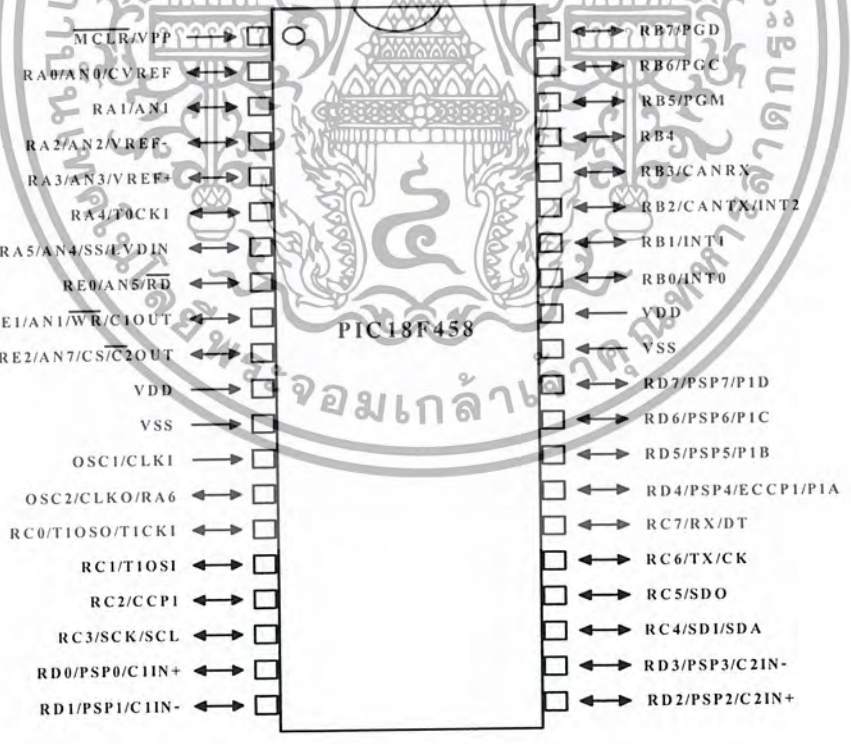
27. มีฟังก์ชั่นการตรวจสอบแรงดันไฟเลี้ยง (Brown Out Reset : BOR)

28. มีสแต็ก (Stack) 31 ระดับ

29. มีพอร์ตอินพุตเอาต์พุต 5 พอร์ตคือ A, B, C, D และ E รวมแล้วพอร์ตการใช้งานทั้งหมดคือ 34 บิต โดยพอร์ต A มีจำนวน 7 บิต คือ RA0-RA6 พอร์ต B มีจำนวน 8 บิต คือ RB0-RB7 พอร์ต C และ D มีจำนวน 8 บิต คือ RC0-RC7 และ RD0-RD7 พอร์ต E มีจำนวน 3 บิต คือ RE0-RE2

2.1.2 การจัดขาของ PIC18F458

PIC18F458 มีขาทั้งหมด 40 ขาในตัวถังแบบ PDIP และ 44 ขาตัวถังแบบ TQFP ซึ่งแต่ละขาจะมีหน้าที่การใช้งานแตกต่างกันออกไปและมีขาอินพุตเอาต์พุตทั้งหมด 34 ขา ซึ่งทั้ง 34 ขาสามารถเลือกใช้งานเป็นอินพุตเอาต์พุตได้ทั้งหมด ดังรูปที่ 2.1 แสดงการจัดขาตัวถังแบบ PDIP



รูปที่ 2.1 การจัดขาของ PIC18F458 ในตัวถังแบบ PDIP

ตารางที่ 2.1 คุณสมบัติและรายละเอียดขาต่างๆของ PIC18F458

ขาสัญญาณ	ขาที่	ชนิดของขา	ชนิดของบัฟเฟอร์	รายละเอียด
OSC1 / CLKI	9	I	ST	ขารับสัญญาณนาฬิกาหลักจากภายนอก
OSC2 / CLK OUT	14	O	ST	เป็นขาของสัญญาณนาฬิกา (1 / 4 ของ OSC1)ใช้ต่อร่วมกับขา OSC1 ในโหมด XT ถ้าใช้โหมดสัญญาณนาฬิกาแบบ RC ขานี้ปล่อยลอยไว้
MCLR / VPP	1	I / P	ST	ขารีเซตหลัก / ขารับสัญญาณในการโปรแกรม
RAO / ANO / CVREF	2	I / O	TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาอินพุตวงจรเปรียบเทียบแรงดันแอนะลอกช่อง AN0 / เอาต์พุตแรง ดันอ้างอิงของโมดูลแรงดันอ้างอิง
RA1 / AN1	3	I / O	TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาอินพุตวงจรเปรียบเทียบแรงดันแอนะลอกช่อง AN1
RA2 / AN2 / VREF-	4	I / O	TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาอินพุตวงจรเปรียบเทียบแรงดันแอนะลอกช่อง AN2 / ขาสัญญาณ
RA3 / AN3 / VREF+	5	I / O	TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาอินพุตวงจรเปรียบเทียบแรงดันแอนะลอกช่อง AN3 / ขาสัญญาณ
RA4 / TOCKI	6	I / O	TTL	ขาพอร์ตอินพุตเอาต์พุต / สัญญาณนาฬิกาของไทเมอร์ 0
RA5 / AN4 / SS / LVDIN	7	I / O	TTL	ขาพอร์ตอินพุตเอาต์พุต/ ขาอินพุตวงจรเปรียบเทียบแรงดันแอนะลอกช่อง AN4 / ขาสัญญาณ Slave Select ในการสื่อสารแบบซิงโครนัส
RB0 / INT0	34	I / O	TTL / ST	ขาพอร์ตอินพุตเอาต์พุต / ขารับสัญญาณอินเตอร์-รัพต์จากภายนอก INT 0
RB1 / INT1	35	I / O	TTL	ขาพอร์ตอินพุตเอาต์พุต / ขารับสัญญาณอินเตอร์-รัพต์จากภายนอก INT 1
RB2 / CANTX / INT2	36	I / O	TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้ในการส่งข้อมูลใน Can Module / ขารับสัญญาณอินเตอร์รัพต์จากภายนอก INT 2
RB3 / CANRX	37	I / O	TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้ในการรับข้อมูลในระบบ Can Module
RB5 / PGM	38	I / O	TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาสัญญาณการโปรแกรมด้วยแรงดันไฟต่ำ

ตารางที่ 2.1 (ต่อ) คุณสมบัติและรายละเอียดขาต่างๆ PIC18F458

ขาสัญญาณ	ขาที่	ชนิดของขา	ชนิดของบัฟเฟอร์	รายละเอียด
RB6 / PGC	39	I / O	TTL / ST	ขาพอร์ตอินพุตเอาต์พุต / ขาสัญญาณอินเตอร์รัพต์จากการเปลี่ยนแปลงสถานะของขาสัญญาณ / ขาสัญญาณนาฬิกาในการโปรแกรม
RB7 / PGD	40	I / O	TTL / ST	ขาพอร์ตอินพุตเอาต์พุต / ขาสัญญาณอินเตอร์รัพต์จากการเปลี่ยนแปลงสถานะของขาสัญญาณ / ขาสัญญาณข้อมูลในการโปรแกรม
RC0 / T1OSO / T1CKI	15	I / O	ST	ขาพอร์ตอินพุตเอาต์พุต / ขาเอาต์พุตออสซิลเลเตอร์ของไทเมอร์ 1 / ขาอินพุตสัญญาณนาฬิกาของไทเมอร์ 1
RC1 / T1OSI	16	I / O	ST	ขาพอร์ตอินพุตเอาต์พุต / ขาอินพุตออสซิลเลเตอร์ไทเมอร์ 1
RC2 / CCP1	17	I / O	ST	ขาพอร์ตอินพุตเอาต์พุต / ขาสัญญาณ (Capture 1 Input / Compare 1 Input / PWM 1 Input)
RC3 / SCK / SCL	18	I / O	ST	ขาพอร์ตอินพุตเอาต์พุต / ขาอินพุตสัญญาณนาฬิกาในการสื่อสารแบบซิงโครนัส / ขาสัญญาณนาฬิกาในโหมด I^2C และ SPI
RC4 / SDI / SDA	23	I / O	ST	ขาพอร์ตอินพุตเอาต์พุต / ขาอินพุตสัญญาณข้อมูลในโหมด SPI / เอาต์พุตสัญญาณข้อมูลในโหมด I^2C
RC5 / SDO	24	I / O	ST	ขาพอร์ตอินพุตเอาต์พุต / ขาสัญญาณเอาต์พุตในโหมด SPI
RC6 / TX / CK	25	I / O	ST	ขาพอร์ตอินพุตเอาต์พุต / ขาสัญญาณส่งข้อมูลในการสื่อสาร USART / ขาสัญญาณนาฬิกาในโหมดการสื่อสารแบบซิงโครนัส
RC7 / RX / DT	26	I / O	ST	ขาพอร์ตอินพุตเอาต์พุต / ขาสัญญาณรับข้อมูลในการสื่อสาร USART / ขาสัญญาณข้อมูลโหมดการสื่อสารแบบซิงโครนัส
RD0 / PSP0 / C1IN+	19	I / O	ST	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนานติดต่อกับระบบบัสไมโครโปรเซสเซอร์
RD1 / PSP1 / C1IN	20	I / O	ST	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนานติดต่อกับระบบบัสไมโครโปรเซสเซอร์
RD2 / PSP2 / C2IN+	21	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนานติดต่อกับระบบบัสไมโครโปรเซสเซอร์

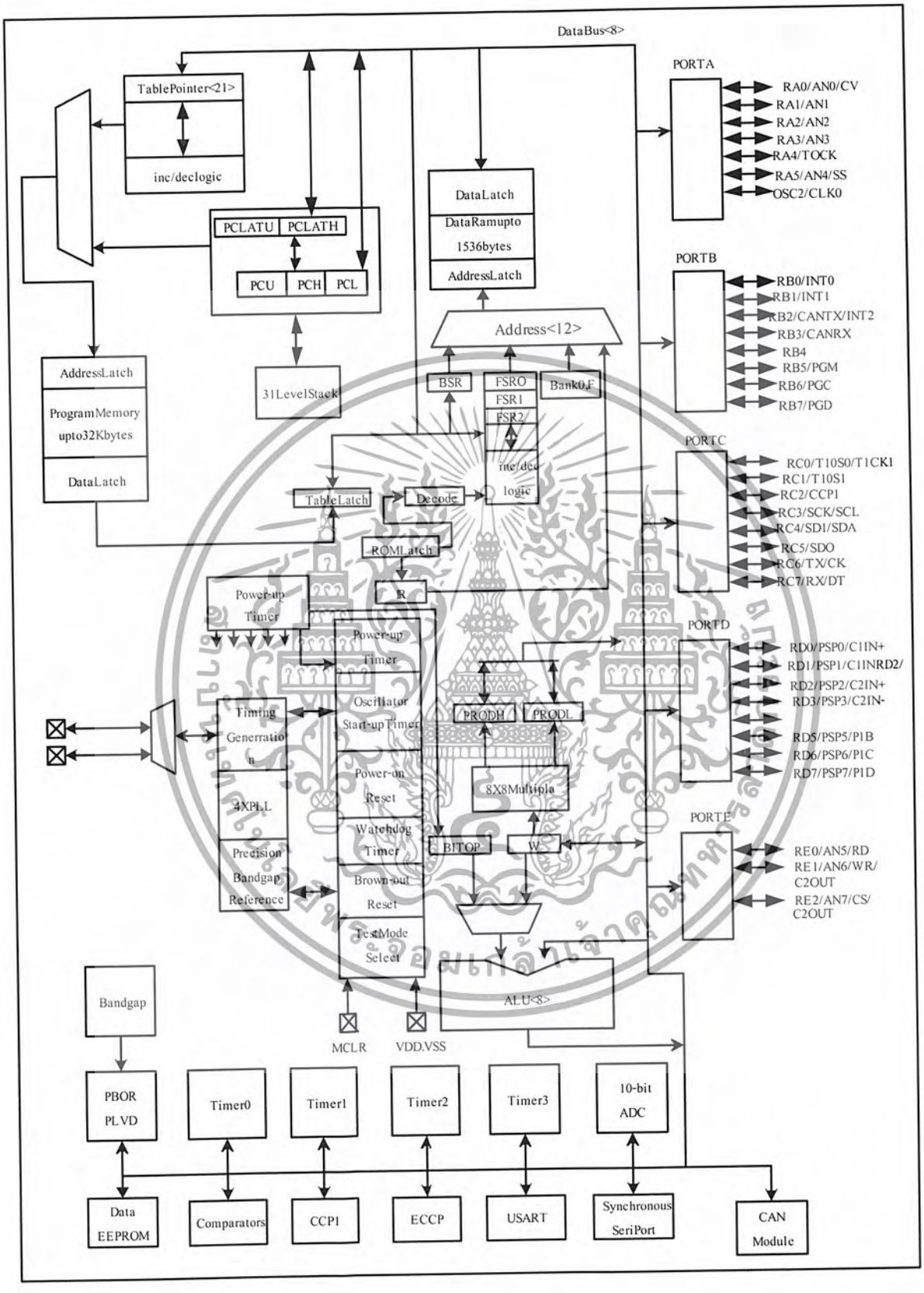
ตารางที่ 2.1 (ต่อ) คุณสมบัติขาสัญญาณของ PIC18F458

ขาสัญญาณ	ขาที่	ชนิดของขา	ชนิดของบัฟเฟอร์	รายละเอียด
RD0 / PSP0 / C1IN+	19	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนาน (Parallel Slave Port : PSP) / ขาอินพุต C1IN+ ของโมดูลเปรียบเทียบแรงดันแอนะลอก
RD1 / PSP1 / C1IN-	20	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนาน ขาอินพุต C1IN- ของโมดูลเปรียบเทียบแรงดันแอนะลอก
RD2 / PSP2 / C2IN+	21	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนาน ขาอินพุต C2IN+ ของโมดูลเปรียบเทียบแรงดันแอนะลอก
RD3 / PSP3 / C2IN-	22	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนาน ขาอินพุต C2IN- ของโมดูลเปรียบเทียบแรงดันแอนะลอก
RD4 / PSP4 / ECCP1 / P1A	27	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนาน ขาสัญญาณของโมดูล ECCP ช่องที่ P1A
RD5 / PSP5 / P1B	28	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนาน ขาสัญญาณของโมดูล ECCP ช่องที่ P1B
RD6 / PSP6 / P1C	29	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนาน ขาสัญญาณของโมดูล ECCP ช่องที่ P1C
RD7 / PSP7 / P1D	30	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาใช้งานเป็นพอร์ตขนาน ขาสัญญาณของโมดูล ECCP ช่องที่ P1D
RE0 / AN5 / RD	8	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาอินพุตแอนะลอกช่องที่ AN5 / ขาควบคุมการอ่านในโหมด PSP
RE1 / AN6 / WR / C1OUT	9	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาอินพุตแอนะลอกช่องที่ AN6 / ขาควบคุมการเขียนในโหมด PSP
RE2 / AN7 / CS / C2OUT	10	I / O	ST / TTL	ขาพอร์ตอินพุตเอาต์พุต / ขาอินพุตแอนะลอกช่องที่ AN7 / ขาควบคุมการเขียนข้อมูลในโหมด PSP
Vcc, Vss	11, 12	-	-	ขาต่อไฟเลี้ยงบวก 2-6 V, ขา 12 ต่อกับกราวด์
หมายเหตุ : I = อินพุต O = เอาต์พุต		I / P = อินพุต / เอาต์พุต TTL = วงจรทีทีแอล		ST = วงจรชมิตต์ทริกเกอร์ P = เพาเวอร์

2.1.3 สถาปัตยกรรมและโครงสร้างของไมโครคอนโทรลเลอร์ PIC18F458

ไมโครคอนโทรลเลอร์ PIC18F458 เป็นไมโครคอนโทรลเลอร์ที่พัฒนาสถาปัตยกรรมมาจากเบอร์อื่นๆ ของตระกูล PIC เช่น PIC16F877 ซึ่งถือว่ามีคุณสมบัติที่ใกล้เคียงกันกับ PIC18F458 แต่จะต่างกันตรงที่มีฟังก์ชั่นและโมดูลต่างๆที่เพิ่มเติมเข้ามาและที่เห็นได้ชัดคือ PIC18F458 นั้นเป็นไมโครคอนโทรลเลอร์ 16 บิต ที่มีชุดคำสั่งทั้งหมด 75 คำสั่ง ซึ่งยังไม่รวมคำสั่งไดเร็กทีฟที่มากับโปรแกรม MPLAB ไมโครคอนโทรลเลอร์ PIC18F458 เป็นไมโครคอนโทรลเลอร์ที่มีสถาปัตยกรรมแบบฮาร์วาร์ด (Harvard Architecture) โดยลักษณะของสถาปัตยกรรมแบบนี้จะเป็นการแยกหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลออกจากกันทำให้การทำงานได้เร็วขึ้น และนอกจากการจัดสถาปัตยกรรมแบบนี้แล้วไมโครไมโครคอนโทรลเลอร์ตระกูล PIC ยังมีลักษณะเด่นอีกอย่างคือมีการทำงานเป็นแบบไปป์ไลน์ (Pipe Line) ซึ่งลักษณะการทำงานคือสามารถเฟตช์ (Fetch) คำสั่งถัดไปได้ในขณะที่กำลังเอ็กซิคิวต์ (Execute) อยู่ จึงทำให้การทำงานของไมโครคอนโทรลเลอร์ PIC18F458 มีการทำงานได้เร็วขึ้นโครงสร้างจะมีบางส่วนที่คล้ายคลึงกับ PIC16F877 สิ่งที่เพิ่มเติมเข้ามาก็คือระบบ Can Module, ECCP และยังมีไทเมอร์ เพิ่มขึ้นมาอีก 1 ตัวมีขนาด 16 บิต คือไทเมอร์ 3 รวมแล้วเป็นจำนวน 4 ตัว และหน่วยความจำก็ได้เพิ่มขนาดขึ้นด้วยโดย PIC16F877 มีหน่วยความจำโปรแกรมเพียง 8 กิโลไบต์ หน่วยความจำข้อมูล 368 กิโลไบต์ ส่วน PIC18F458 มีมากถึง 32 กิโลไบต์ และมีหน่วยความจำข้อมูล 1536 ไบต์ สำหรับหน่วยความจำอีอีพรอมยังคงเท่าเดิมคือ 256 กิโลไบต์ สถาปัตยกรรมและการทำงานของไมโครคอนโทรลเลอร์ตระกูล PIC โดยรวมแล้วจะมีส่วนการทำงานพื้นฐานที่เหมือนกัน รูปที่ 2.2 ได้แสดงสถาปัตยกรรมของไมโครคอนโทรลเลอร์ PIC18F458

การทำงานของไมโครคอนโทรลเลอร์เริ่มจากการป้อนไฟเลี้ยงและป้อนสัญญาณนาฬิกาให้แก่ตัวมันจากนั้นซีพียูจะติดต่อกับหน่วยความจำโปรแกรม เพื่อที่จะอ่านข้อมูลคำสั่งแล้วทำงานตามคำสั่งที่บรรจุอยู่ในหน่วยความจำโปรแกรม เมื่อไมโครคอนโทรลเลอร์ทำงานตามคำสั่งที่กำหนดให้ข้อมูลของชุดคำสั่งจะถูกนำไปเก็บไว้ในรีจิสเตอร์คำสั่ง (Instruction Register) จากนั้นจะถูกส่งไปยังวงจรถอดรหัสเพื่อทำการควบคุมไทเมอร์ทั้งหมดภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ยังส่งไปควบคุมหน่วยคำนวณทางคณิตศาสตร์ (Arithmetic Logic Unit : ALU) โดยผ่านทางด้านวงจรมัลติเพล็กซ์ (Multiplex) ด้วยการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์ PIC18F458 เป็นหน้าที่ของส่วนกำเนิดจังหวะการทำงาน (Timing Generation) ซึ่งจะทำงานสัมพันธ์กับไทเมอร์ 3 ตัว คือ ออสซิลเลเตอร์สตาร์ตอัปไทเมอร์ วอตช์ด็อกไทเมอร์ และ เพาเวอร์อัปไทเมอร์



รูปที่ 2.2 สถาปัตยกรรมของไมโครคอนโทรลเลอร์ PIC18F458

2.1.4 โหมดสัญญาณนาฬิกา

ในไมโครคอนโทรลเลอร์ PIC18F458 สามารถเลือกแหล่งกำเนิดสัญญาณนาฬิกาที่ป้อนให้แก่ซีพียูได้ 8 โหมด การกำหนดให้เป็นโหมดต่างๆนั้นสามารถกระทำได้ที่รีจิสเตอร์กำหนดค่าการทำงาน (Configuration Bit) โดยกำหนดที่บิต FOSC2, FOSC1 และ FOSC0

LP : ใช้สัญญาณนาฬิกาจากคริสตอลกำลังงานต่ำ

XT :ใช้สัญญาณนาฬิกาจากคริสตอลแบบเรโซเนเตอร์

HS :ใช้สัญญาณนาฬิกาจากคริสตอลความเร็วสูง

HS4 :ใช้วงจร PLL ภายในคูณค่าความถี่ของสัญญาณนาฬิกา จากความถี่สัญญาณนาฬิกาอินพุต 4 เท่าโดยใช้ร่วมกับคริสตอลความเร็วสูง

RC :ใช้สัญญาณนาฬิกาตัวต้านทานและตัวเก็บประจุต่อภายนอก

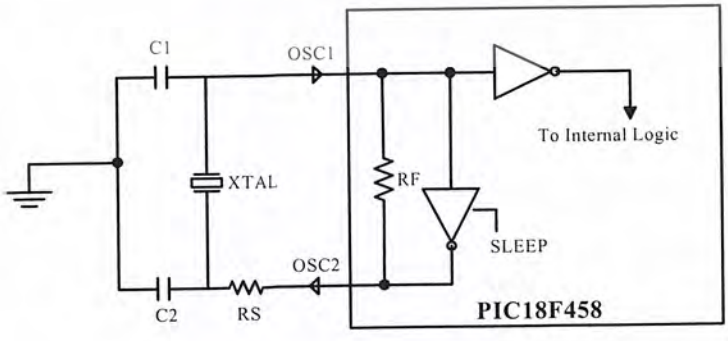
EC :ใช้สัญญาณนาฬิกาจากภายนอก

RCIO :ใช้สัญญาณนาฬิกาตัวต้านทานและตัวเก็บประจุต่อภายนอกแต่โหมดนี้จะสามารถใช้งานขา OSC2 เป็นขาอินพุตเอาต์พุตได้

ECIO : ใช้สัญญาณนาฬิกาจากภายนอกป้อนที่ขา OSC1 แต่โหมดนี้จะสามารถใช้งานขา OSC2 เป็นขาอินพุตเอาต์พุตได้

1) การป้อนสัญญาณนาฬิกาโดยใช้คริสตอล

โหมดสัญญาณนาฬิกา LP, XT, HS และ HS4 จะมีลักษณะของการต่อเหมือนกันแต่จะต่างกันตรงที่ชนิดของคริสตอลที่ใช้ ส่วนการต่อใช้งานแบบนี้จะไม่สามารถนำขา OSC2 ไปใช้งานเป็นขาอินพุตเอาต์พุตได้ถ้าหากต้องการนำขา OSC2 ไปใช้งานจะต้องใช้คริสตอลที่มีโมดูล คือ มีตัวเก็บประจุและตัวต้านทานอยู่ภายใน ซึ่งการใช้งานเพียงต่อไฟเลี้ยงให้เพียงเท่านั้น แล้วก็สามารถนำขา OSC2 ไปใช้งานได้โดยที่ขานี้จะมีความถี่ออกมาเป็นค่าความถี่ของ OSC1/4 ประโยชน์ก็คือสามารถนำขา OSC2 ไปใช้งานเป็นสัญญาณนาฬิกาให้กับอุปกรณ์อื่นๆ ได้



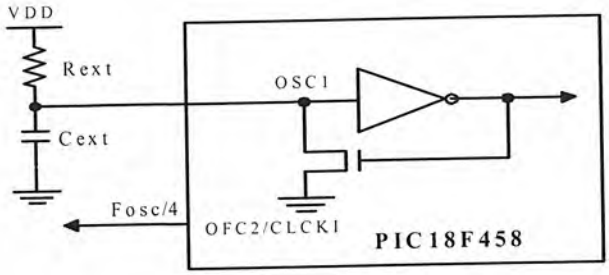
รูปที่ 2.3 การต่อคริสตอลแบบเรโซเนเตอร์

ตารางที่ 2.2 ค่าของตัวเก็บประจุที่ใช้กับคริสตอลออสซิลเลเตอร์แบบต่างๆ

ชนิดคริสตอลที่ใช้	ความถี่	C1	C2
LP	32.0 KHz	33 pF	33 pF
	200 KHz	15 pF	15 pF
XT	200 KHz	47-68 pF	47-68 pF
	1.0 MHz	15 pF	15 pF
	4.0 MHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0 MHz	15-33 pF	15-33 pF
	20.0 MHz	15-33 pF	15-33 pF
	25.0 MHz	15-33 pF	15-33 pF

2) โหมดการป้อนสัญญาณนาฬิกาโดยใช้ตัวต้านทานและตัวเก็บประจุ RC และ RCIO

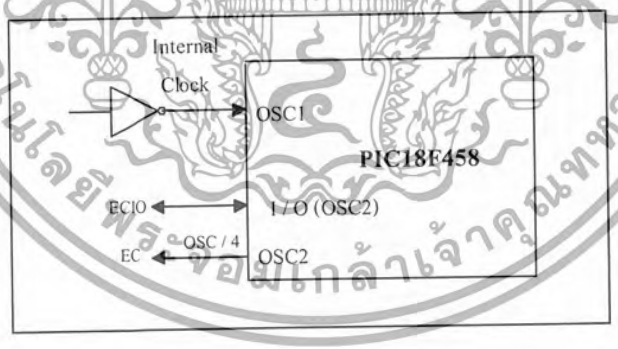
การต่อออสซิลเลเตอร์ในโหมดนี้จะต่อใช้งานเพียงขาเดียวคือขา OSC1 ซึ่งการต่อใช้งานในโหมดนี้จะไม่เป็นที่นิยมเท่าไหร่ เนื่องด้วยการใช้งานโหมดนี้มีข้อเสียในเรื่องของอุณหภูมิซึ่งเมื่อไมโครคอนโทรลเลอร์ทำงานเป็นเวลานานๆ อาจทำให้ไมโครคอนโทรลเลอร์ทำงานผิดพลาดได้ โดยเฉพาะในงานที่ต้องการความเที่ยงตรงของเวลา การใช้ออสซิลเลเตอร์แบบนี้จะทำให้เกิดความคลาดเคลื่อนได้ แต่ข้อดีของการใช้ออสซิลเลเตอร์แบบนี้คือประหยัดหาซื้อง่าย ส่วนในโหมดของ RCIO จะสามารถนำขา RA6 ไปใช้งานเป็นขาอินพุตเอาต์พุตปกติได้ การต่อใช้งานแสดงดังรูปที่ 2.4 ซึ่งจะมีตัวต้านทานและตัวเก็บประจุทำหน้าที่กำหนดค่าความถี่ โดยค่าตัวต้านทาน Rext ที่นำมาใช้นั้นจะอยู่ระหว่าง 4 -100 กิโลโอห์มตัวเก็บประจุ Cext ที่ใช้งานได้จะต้องมากกว่า 20 pF ขึ้นไป



รูปที่ 2.4 การต่อออสซิลเลเตอร์แบบ RC และ RCIO

3) โหมดการป้อนสัญญาณนาฬิกาแบบ EC และ ECIO

การทำงานในโหมดนี้จะเป็นการรับสัญญาณนาฬิกาจากภายนอกผ่านทางขา OSC1 ซึ่งขา OSC2 จะปล่อยลอยไว้ส่วนขา RA6/OSC2 จะมีสัญญาณนาฬิกาออกมาที่ขานี้เท่ากับความถี่สัญญาณนาฬิกาอินพุตหารด้วย 4 สามารถนำสัญญาณนาฬิกาจากขา OSC2 ไปใช้งานได้ประโยชน์ได้ เช่น ใช้ในการสื่อสารซิงโครนัส (Synchronize) และอุปกรณ์ต่อพ่วงอื่นๆ แต่โหมด ECIO นั้นจะต่างกันตรงที่จะสามารถนำขา RA6/OSC2 ไปใช้เป็นขาอินพุตเอาต์พุตปกติได้ รูปที่ 2.5 แสดงการเชื่อมต่อออสซิลเลเตอร์แบบ EC และ ECIO

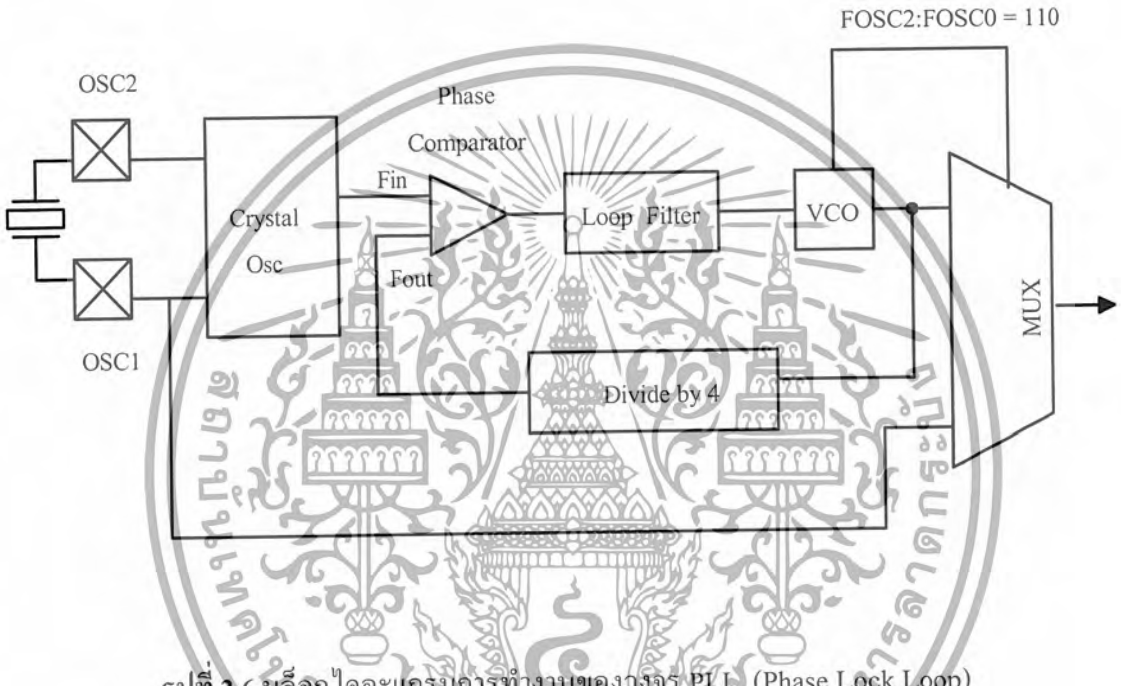


รูปที่ 2.5 การต่อออสซิลเลเตอร์แบบ EC และ ECIO

4) โหมดการป้อนสัญญาณนาฬิกาโดยใช้วงจร PLL (Phase Lock Loop)

โหมดนี้เพิ่มเติมเข้ามาใหม่ในไมโครคอนโทรลเลอร์ของตระกูล PIC บางเบอร์เท่านั้นซึ่งมีในไมโครคอนโทรลเลอร์ PIC18F458 การใช้งานในโหมดนี้เริ่มจากการป้อนสัญญาณนาฬิกาให้แก่ซีพียูโดยใช้วิธีต่อแบบ คริสตรอลออสซิลเลเตอร์จากกำหนดค่าให้โหมด PLL (Phase Lock Loop) ทำงานโดยการกำหนด Configuration Bit ที่บิต OSC1-OSC2 และในซอฟต์แวร์โปรแกรมซีพียู เช่น

โปรแกรม Epic Win ในฟังก์ชั่น HS4 และที่แกรมโดยซีพียูเองจะมีวงจร PLL อยู่ภายในทำการคูณความถี่ให้ โดยมีค่าเท่ากับสัญญาณนาฬิกาที่อินพุตคูณด้วย 4 เช่นป้อนสัญญาณนาฬิกามีค่าเท่ากับ 10 เมกะเฮิรตซ์ เมื่อผ่านวงจรเฟสล็อกลูปก็จะได้ค่าความถี่ออกมาเป็น 40 เมกะเฮิรตซ์ ซึ่งคุณสมบัติแบบนี้จะไม่มีใน PIC16F877 ซึ่งการทำงานของซีพียูจะอาศัยสัญญาณนาฬิการะบบ (Buss Clock) เป็นจุดอ้างอิงในการทำงานโดย ของไมโครคอนโทรลเลอร์จะมีค่าเท่ากับ 1 หาร 4 ของสัญญาณนาฬิกาอินพุต



รูปที่ 2.6 บล็อกไดอะแกรมการทำงานของวงจร PLL (Phase Lock Loop)

2.1.5 การรีเซตของไมโครคอนโทรลเลอร์ PIC18F458

การรีเซตของไมโครคอนโทรลเลอร์ PIC 18F458 สามารถเกิดได้หลายแหล่งดังต่อไปนี้

- การรีเซตที่ขึ้นเมื่อเริ่มจ่ายไฟให้ซีพียู (Power On Reset : POR)
- การรีเซตจากขารีเซตหลัก (MCLR)
- การรีเซตจากขา MCLR ในระหว่างการทำงานในโหมดประหยัดพลังงาน (Sleep Mode)
- การรีเซตจากวอตซ์ด็อกไทม์เมอร์ (Wacthdog Timer : WDT)
- การรีเซตของวอตซ์ด็อกไทม์เมอร์ ที่ทำให้เกิดเวกอัป (Wake Up) ในขณะอยู่ในโหมดพลังงาน (Sleep Mode)
- การรีเซตที่เกิดจากวงจรตรวจสอบแรงดันต่ำ (Brown Out Reset : BOR)

1) เพาเวอร์ออนรีเซต (Power On Reset : POR)

การรีเซตแบบนี้จะเกิดขึ้นเมื่อเริ่มจ่ายไฟให้กับซีพียูโดยจะเกิดรีเซตขึ้นหากพบว่ามีการจ่ายไฟให้กับตัวซีพียูอยู่ในระดับที่ซีพียูไม่สามารถทำงานได้ โดยปกติแล้วจะอยู่ในช่วง 1.2 ถึง 1.7 โวลต์ โดยในการรีเซตแบบนี้ถือว่ามีความสำคัญสูงสุดเนื่องจากเป็นการรีเซตลำดับแรกสุดถ้าหากไม่มีการจ่ายไฟให้แก่ซีพียูก็จะไม่เกิดการรีเซตแบบนี้ขึ้น

2) เพาเวอร์อัปไทเมอร์ (Power Up Timer : TWRT)

เมื่อเกิดพาวเวอร์ออนรีเซตจะมีไทเมอร์ตัวหนึ่งภายในทำงานไทเมอร์ตัวนั้นก็คือเพาเวอร์-อัปไทเมอร์โดยเพาเวอร์อัปไทเมอร์จะเป็นตัวกำหนดคาบเวลาการเกิดไทม์เอาต์ไว้ที่ 72 มิลลิวินาที หลังการเกิดเพาเวอร์ออนรีเซตทั้งนี้ก็เพื่อที่จะให้ไมโครคอนโทรลเลอร์รอให้ส่วนต่างๆของซีพียูพร้อมที่จะทำงาน ก่อนที่จะมีการเกิดรีเซตเกิดขึ้น สำหรับการปิดการทำงานของเพาเวอร์อัปไทเมอร์ตัวนี้สามารถกระทำได้ที่รีจิสเตอร์ Configuration ที่บิต Pwrte

3) ออสซิลเลเตอร์สตาร์ตอัปไทเมอร์ (Oscillator Start Up Timer : OST)

ออสซิลเลเตอร์แบบนี้จะทำหน้าที่หน่วงเวลาในการทำงานของไมโครคอนโทรลเลอร์ไปอีก 1024 ไซเคิล จากอินพุตที่ขาสัญญาณนาฬิกา OSC1 โดยจะเกิดขึ้นหลังในส่วนการเกิดเพาเวอร์-อัปไทเมอร์ (หากมีการเปิดการทำงานไว้) ทั้งนี้เพื่อที่จะให้แน่ใจว่าส่วนกำเนิดสัญญาณนาฬิกา หรือวงจรออสซิลเลเตอร์นั้นทำงานได้คงที่หรือมีเสถียรภาพพอแล้ว โดยการทำงานในส่วนนี้จะเกิดขึ้นได้ก็ต่อเมื่อเราเลือกใช้โหมดสัญญาณนาฬิกาเป็นแบบ LP XT และ HS และเมื่อการใช้งานเพาเวอร์-ออนรีเซต หรือเมื่อเกิดการเวกอัป (Wake Up) จากการทำงานในโหมดสลีป (Sleep Mode)

4) บราวเอาต์รีเซต (Brown Out Reset : BOR)

บราวเอาต์รีเซตนี้จะทำหน้าที่รีเซตซีพียูในขณะที่ไฟเลี้ยงจ่ายให้ซีพียูตกลงจนต่ำแรงดันที่กำหนดไว้ (VBOR) โดยปกติแล้วจะอยู่ที่ 4 โวลต์ โดยหากมีการตรวจพบว่าแรงดันที่ไฟเลี้ยง VDD มีขนาดลดต่ำลงกว่าแรงดันที่ VBOR เป็นเวลานานเกิน 100 นาโนวินาที ก็จะการส่งสัญญาณไปทำการรีเซตซีพียูแต่ถ้าหากแรงดัน VDD ลดต่ำกว่า VBOR ในระยะเวลาไม่เกิน 100 นาโนวินาที กระบวนการรีเซตของซีพียูก็จะไม่เกิดขึ้น สำหรับประโยชน์ของการรีเซตแบบนี้ คือจะช่วยแก้ปัญหาการทำงานที่ผิดพลาดของซีพียูที่เกิดจากแรงดันที่ไม่คงที่เมื่อเกิดการรีเซตจากบราวเอาต์รีเซตสภาวะการรีเซตนี้จะยังคงค้างอยู่จนกว่าแรงดันที่ไฟเลี้ยง VDD จะมีค่าสูงกว่าแรงดัน VBOR อีกครั้งซึ่งหลังจากแรงดันที่ VDD กลับมามีค่ามากกว่าแรงดัน VBOR แล้วจะเกิดการหน่วงเวลาไปอีก 72 มิลลิวินาที จากวงจรเพาเวอร์อัปไทเมอร์ หากมีการเปิดการทำงานของเพาเวอร์อัปไทเมอร์เอาไว้โดยปกติสามารถกระทำได้โดยผ่านทางซอฟต์แวร์ก่อนการโปรแกรม สำหรับโปรแกรมดังกล่าวที่มีฟังก์ชั่นนี้ก็เช่นโปรแกรม Epic Win ซึ่งในช่วงระหว่างการหน่วงเวลา

72 มิลลิวินาที ของเพาเวอร์อัปไทเมอร์ นี้หากแรงดัน VDD ตกลงต่ำกว่า VBOR อีก การทำงานของบราวเอาต์รีเซตนี้ ก็จะถูกทำการรีเซตกลับไปเริ่มใหม่ทันที

2.1.6 ลำดับการเกิดไทม์เอาต์ (Time Out Sequence)

จากการทำงานในส่วนต่างๆ กว่าที่ซีพียูจะเกิดรีเซตได้นั้นจะต้องผ่านขั้นตอนต่างๆ หลายขั้นตอนนั้นก็เพื่อให้การทำงานของซีพียูถูกต้องแม่นยำและมีเสถียรภาพในการทำงานมากขึ้น ในตารางที่ 2.3 แสดงค่าเวลาที่เกิดขึ้นจากการรีเซตของซีพียูในแต่ละแบบ โดยหลังจากการเกิดเพาเวอร์ออนรีเซต หากทำการปิดการทำงานของเพาเวอร์อัปไทเมอร์สัญญาณรีเซตจะถูกหน่วงเวลาไปอีกเป็นเวลา 72 มิลลิวินาที หลังจากนั้นภาคออสซิลเลเตอร์สตาร์ตอัปไทเมอร์จะหน่วงเวลาของออสซิลเล-เตอร์ออกไปอีกจำนวน 1024 ไซเคิล และเมื่อเสร็จสิ้นกระบวนการดังกล่าวตามที่ได้กล่าวมาแล้วนี้กระบวนการรีเซตของซีพียูจึงจะเกิดขึ้น ส่วนบราวเอาต์รีเซตจะมีลักษณะเดียวกันกับการรีเซตของเพาเวอร์ออนรีเซต ส่วนการเกิดเวกอัป (Wake Up) จะมีการหน่วงเวลาการทำงานของซีพียูไปเป็น 1024 ไซเคิล เพื่อรอให้ระดับกระแสและแรงดันที่จ่ายให้ซีพียูคงที่หรือมีเสถียรภาพมากขึ้น

ตารางที่ 2.3 ค่าเวลาต่างๆ ของแต่ละส่วนก่อนการเกิด ไทม์เอาต์ (Time Out)

ออสซิลเลเตอร์	เพาเวอร์อัปไทเมอร์		เพาเวอร์ออนรีเซต	การเวกอัปจากโหมดสลีป
	PWRTEN = 0	PWRTEN = 1		
HS4	72 มิลลิวินาที + 1024 ไซเคิล + 2 มิลลิวินาที	1024 ไซเคิล + 2 มิลลิวินาที	72 มิลลิวินาที + 1024 ไซเคิล + 2 มิลลิวินาที	1024 ไซเคิล + 2 มิลลิวินาที
HS, XT, LP	72 มิลลิวินาที + 1024 ไซเคิล	1024 ไซเคิล	72 มิลลิวินาที + 1024 ไซเคิล	1024 ไซเคิล
EC	72 มิลลิวินาที	-	72 มิลลิวินาที	-
External RC	72 มิลลิวินาที	-	72 มิลลิวินาที	-

2.2 หน่วยความจำ

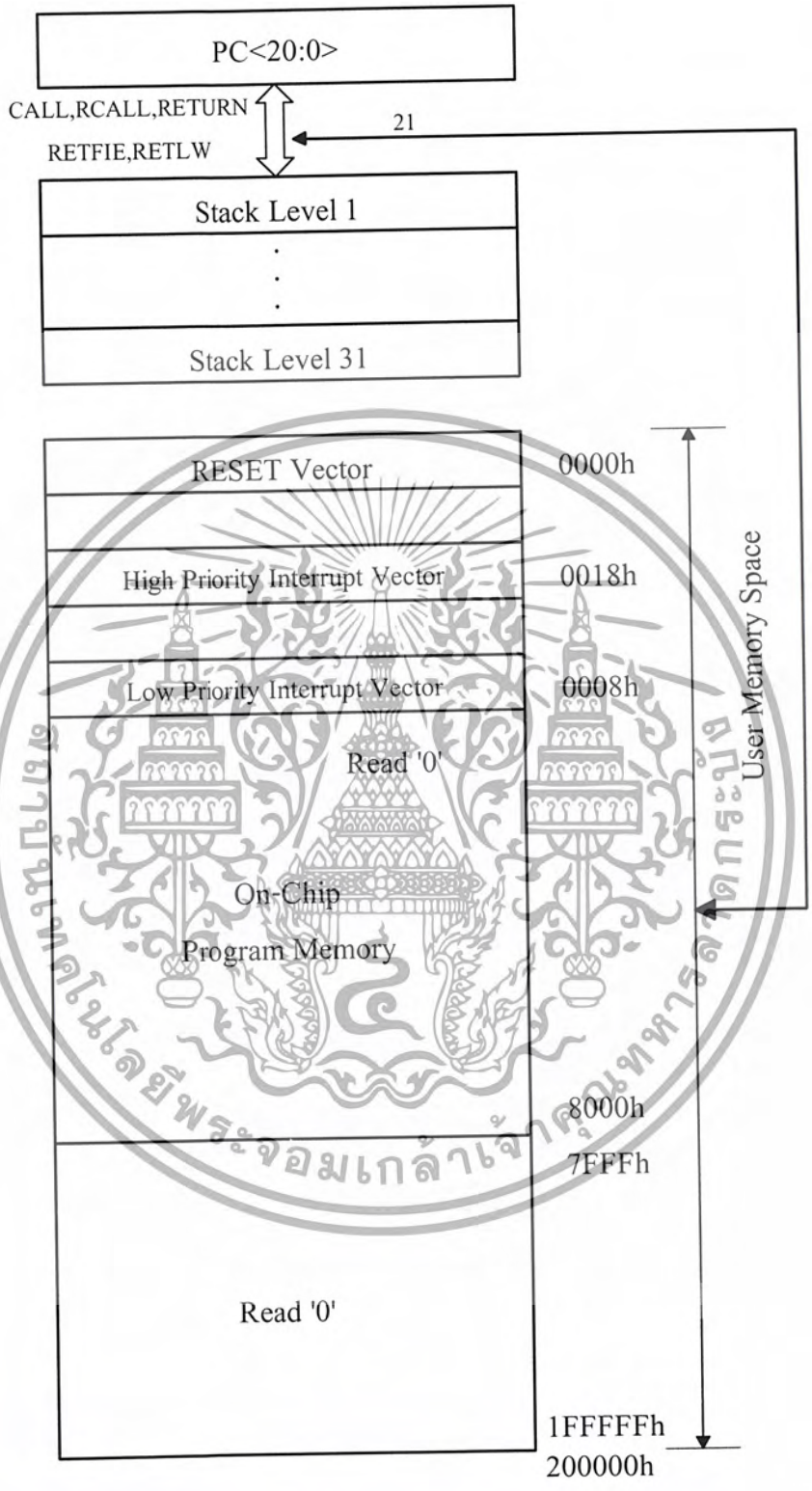
หน่วยความจำเป็นส่วนที่เก็บข้อมูลและโค๊ดโปรแกรมต่างๆ หน่วยความจำของไมโครคอนโทรลเลอร์ PIC18F458 มีอยู่ด้วยกัน 3 ส่วน ดังนี้

- หน่วยความจำโปรแกรม (Program Memory)
- หน่วยความจำข้อมูล (Data Memory)
- หน่วยความจำอีอีพรอม (EEPROM)

2.2.1 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมเป็นส่วนที่ใช้เก็บซอร์สโค๊ดของโปรแกรม โดยหน่วยความจำโปรแกรมของ PIC18F458 จะเป็นแบบเฟลช (Fash Memory) ที่สามารถทำการเขียนและลบข้อมูลได้หลายพันครั้งโดยโครงสร้างชุดคำสั่งของ PIC18F458 จะเป็น 16 บิต ซึ่งลักษณะการอ้างถึงข้อมูลจะอ้างได้ถึง 16 บิต เช่นกันหน่วยความจำโปรแกรมของ PIC18F458 มีขนาด 32 กิโลไบต์ ซึ่งถือว่าเพียงพอต่อการใช้งานแล้วหลังจากที่ทำการเขียนในขั้นตอนของการโปรแกรมแล้วก็จะมีไว้สำหรับอ่านออกมาได้เพียงอย่างเดียวเท่านั้น รูปที่ 2.7 ได้แสดงพื้นที่การจัดสรรหน่วยความจำโปรแกรมของ PIC18F458 และ PIC18F258 ซึ่งมีโครงสร้างเหมือนกัน โดยการจัดสรรนั้นได้ถูกจัดวางไว้อยู่ที่ตำแหน่ง 0000h-1FFFFFh สำหรับแอดเดรส 0000h-07FFFh จะสงวนไว้สำหรับเก็บค่ารีเซตเว็กเตอร์ (Reset Vector) และที่ตำแหน่ง 0008h และ 0018h จะมีไว้เก็บค่าอินเตอร์รัพต์เวกเตอร์ (Interrupt Vector) ไบต์สูงและต่ำ ดังนั้นในการเขียนโปรแกรมจะต้องมาเริ่มต้นที่แอดเดรส 0020h จึงจะเหมาะสมที่สุดแต่ถ้าหากผู้เขียนโปรแกรมคาดว่าไม่มีการใช้งานการอินเตอร์รัพต์แล้วก็สามารถที่จะละเลยในส่วนของการสงวนนี้ได้

ไมโครคอนโทรลเลอร์ PIC18F458 ยังมีพื้นที่หน่วยความจำพิเศษสำหรับเก็บค่าของโปรแกรมเคาน์เตอร์ชั่วคราวเรียกว่าสแต็ก (Stack) ซึ่งจะมีบทบาทมากในการกระโดดไปทำงานยังโปรแกรมย่อยโดยหากเมื่อมีการกระทำคำสั่งให้กระโดดไปทำงานยังโปรแกรมย่อยซีพียูจะทำการเก็บค่าโปรแกรมเคาน์เตอร์ (Program Counter : PC) ในขณะนั้นไว้ในสแต็กจากนั้นจึงกระโดดไปทำงานยังโปรแกรมย่อยและเมื่อทำงานเรียบร้อยแล้วซีพียูก็จะไปอ่านค่าโปรแกรมเคาน์เตอร์ที่อยู่ในสแต็กกลับคืนมา แล้วไปทำงานยังโปรแกรมหลักต่อไป สำหรับสแต็กภายใน PIC18F458 นี้จะสามารถเก็บค่าได้มากถึง 31 ระดับ ซึ่งมีค่ามากกว่า PIC16F877 ที่มีขนาดเพียง 8 ระดับเท่านั้น และในส่วนของโปรแกรมเคาน์เตอร์ของ PIC18F458 จะมีขนาด 21 บิต ซึ่งใน PIC16F877 จะมีขนาดเพียง 13 บิต อาจกล่าวได้ว่าถ้าสแต็กมีค่ามากก็จะสามารถเก็บค่าโปรแกรมเคาน์เตอร์ได้มากนั่นเอง



รูปที่ 2.7 การจัดสรรหน่วยความจำโปรแกรมของ PIC18F458

2.2.2 สแต็ก (Stack)

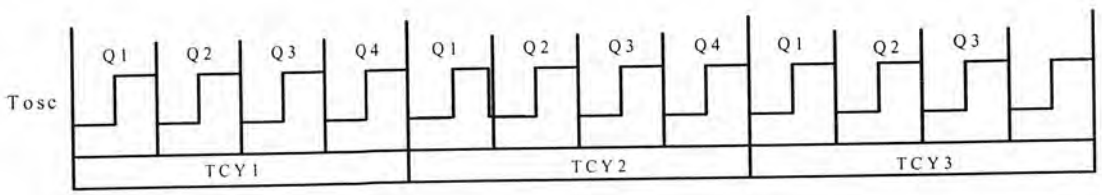
ในไมโครคอนโทรเลอร์ PIC18F458 ได้จัดสรรหน่วยความจำที่เรียกว่า สแต็กไว้เก็บค่าของโปรแกรมเคาน์เตอร์ ชั่วคราวได้ 31 ระดับโดยในแต่ละระดับจะมีขนาด 21 บิต ถ้าหากมีการพุช (Push) 1 ครั้งสแต็กก็จะทำการเก็บค่าของโปรแกรมเคาน์เตอร์ ไว้ 1 ค่าโดยในไมโครคอนโทรลเลอร์ PIC18F458 จะสามารถเก็บค่าหรือรองรับได้ 31 ครั้งหากมีการพุช (Push) เป็นครั้งที่ 32 ซีพียูก็จะนำค่าของโปรแกรมเคาน์เตอร์เก็บลงในสแต็กในครั้งที่ 1 สำหรับการพุช (Push) หมายถึง การที่ซีพียูกระทำคำสั่งตามปกติแล้วพบคำสั่ง CALL หรือเกิดมีการอินเตอร์รัพต์เกิดขึ้น ดังนั้นก่อนที่ซีพียูจะกระโดดไปทำคำสั่งถัดไปซีพียูก็จะทำการเก็บค่าของโปรแกรมเคาน์เตอร์ไว้ในหน่วยความจำชั่วคราว ซึ่งหน่วยความจำชั่วคราวนั้นคือสแต็ก (Stack) นั่นเองเมื่อซีพียูกระทำคำสั่งในโปรแกรมย่อยเรียบร้อยแล้วซีพียูก็จะไปทำการอ่านค่าที่โปรแกรมเคาน์เตอร์ที่เก็บไว้ในสแต็กกลับมากระทำคำสั่งในแอดเดรสถัดไป

2.2.3 รีจิสเตอร์โปรแกรมเคาน์เตอร์ PCL, PCLATH, และ PCLATL

สำหรับโปรแกรมเคาน์เตอร์ (Program Counter : PC) ในซีพียู PIC18F458 มีขนาด 21 บิต โดยจะแบ่งออกเป็นสองส่วนคือ PCL และ PCH ไบต์ต่ำคือ PCL ไบต์สูงคือ PCH โดยสามารถอ่านและเขียนค่าได้ในรีจิสเตอร์ PCL ส่วน PCH ในรีจิสเตอร์ PC บิตที่ 15 และ 8 จะไม่สามารถอ่านและเขียนค่าได้โดยตรงซึ่งหากต้องการที่จะอ่านและเขียนข้อมูลนั้นจะต้องกระทำผ่านรีจิสเตอร์ PCLATH และเมื่อเกิดรีเซตทุกครั้งรีจิสเตอร์ PC จะถูกเคลียร์ค่าเป็นศูนย์ทั้งหมดและมีผลทำให้ตำแหน่งเริ่มต้นที่ซีพียูทำงานนั้นไปอยู่ที่ตำแหน่ง 0x000

2.2.4 จังหวะและไซเคิลการทำงานของ PIC18F458

สัญญาณนาฬิกาหลักภายใน ของไมโครคอนโทรลเลอร์ PIC18F458 จะประกอบไปด้วยสัญญาณนาฬิกาจากภายนอกจำนวน 4 ไซเคิล (OSC/4) ประกอบไปด้วย Q1, Q2, Q3 และ Q4 โดยสัญญาณนาฬิกาหลักภายใน ของซีพียูจะมีค่าเท่ากับ 1/4 ของสัญญาณนาฬิกาจากภายนอกที่รับเข้ามา โดย Tosc คือ คาบเวลาของสัญญาณนาฬิกาภายนอก และ TCY คือ คาบเวลาของสัญญาณนาฬิกาหลักภายใน



รูปที่ 2.8 จังหวะและไซเคิลการทำงานของ PIC18F458

โดยในแต่ละช่วงการทำงานจะมีลักษณะดังต่อไปนี้

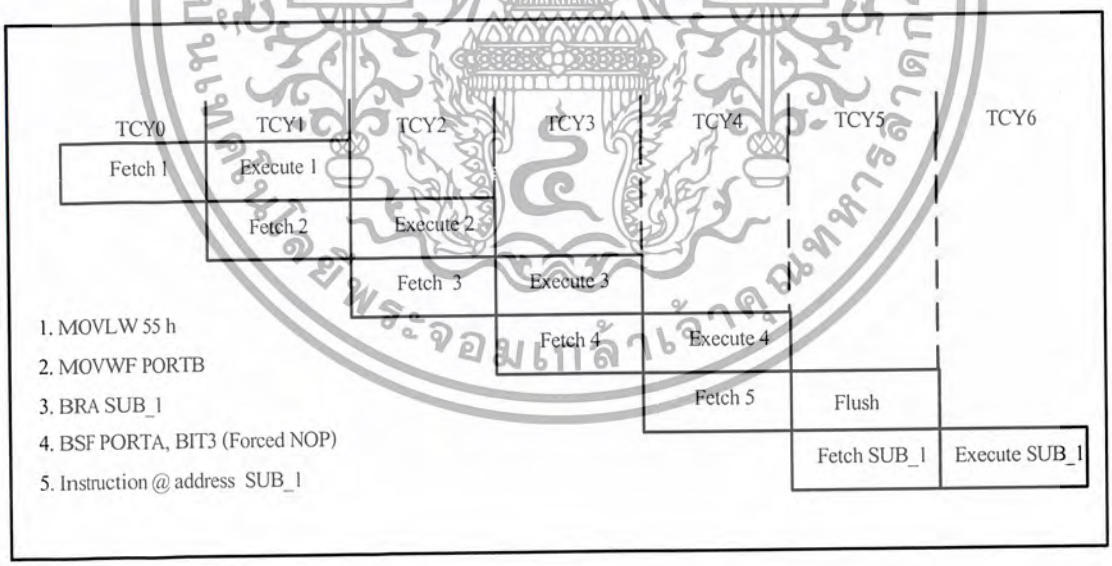
Q1 : ถอดรหัสคำสั่ง (Decode) หรืออยู่ในสภาวะ No Operation

Q2 : อ่านคำสั่งข้อมูล (Read Data Cycle) หรืออยู่ในสภาวะ No Operation

Q3 : ประมวลผลข้อมูล (Process The Data)

Q4 : เขียนรหัสข้อมูล (Write Data Cycle)

รีจิสเตอร์ PC (Program Counter) จะเพิ่มขึ้นทีละ 1 ค่าทุกๆ ไซเคิลการทำงานที่ 1 (Q1) และจะมีการอ่านข้อมูลคำสั่ง (Source Code) หรือเรียกสภาวะการเฟตช์ (Fetch) ข้อมูลจากหน่วยความจำโปรแกรมมาเก็บไว้ในรีจิสเตอร์คำสั่ง ในไซเคิลการทำงานที่ 4 (Q4) ซึ่งขบวนการในการถอดรหัสคำสั่ง และการปฏิบัติคำสั่ง (Execute) จะเกิดขึ้นอยู่ในช่วงการทำงานที่ 1 ถึง 4 ดังรูปที่ 2.8 ดังนั้นในการประมวลผลของไมโครคอนโทรลเลอร์ PIC18F458 สามารถประมวลผล 1 Clock ต่อ 1 คำสั่ง นั้นหมายถึงจะเป็นลักษณะ Clock ที่อยู่ภายในไม่ใช่ Clock ที่อยู่ภายนอกซึ่ง Clock ภายในประกอบด้วยสัญญาณนาฬิกาจากภายนอก 4 ไซเคิล และเหตุที่ซีพียูในตระกูล PIC นี้สามารถประมวลผลสัญญาณ 1 ไซเคิล ต่อคำสั่งได้เพราะรูปแบบการประมวลผลนั้นจะเป็นแบบไปป์ไลน์ (Pipe line) คือ จะมีลักษณะการทำงานที่เหลื่อมล้ำกัน ดังรูปที่ 2.9



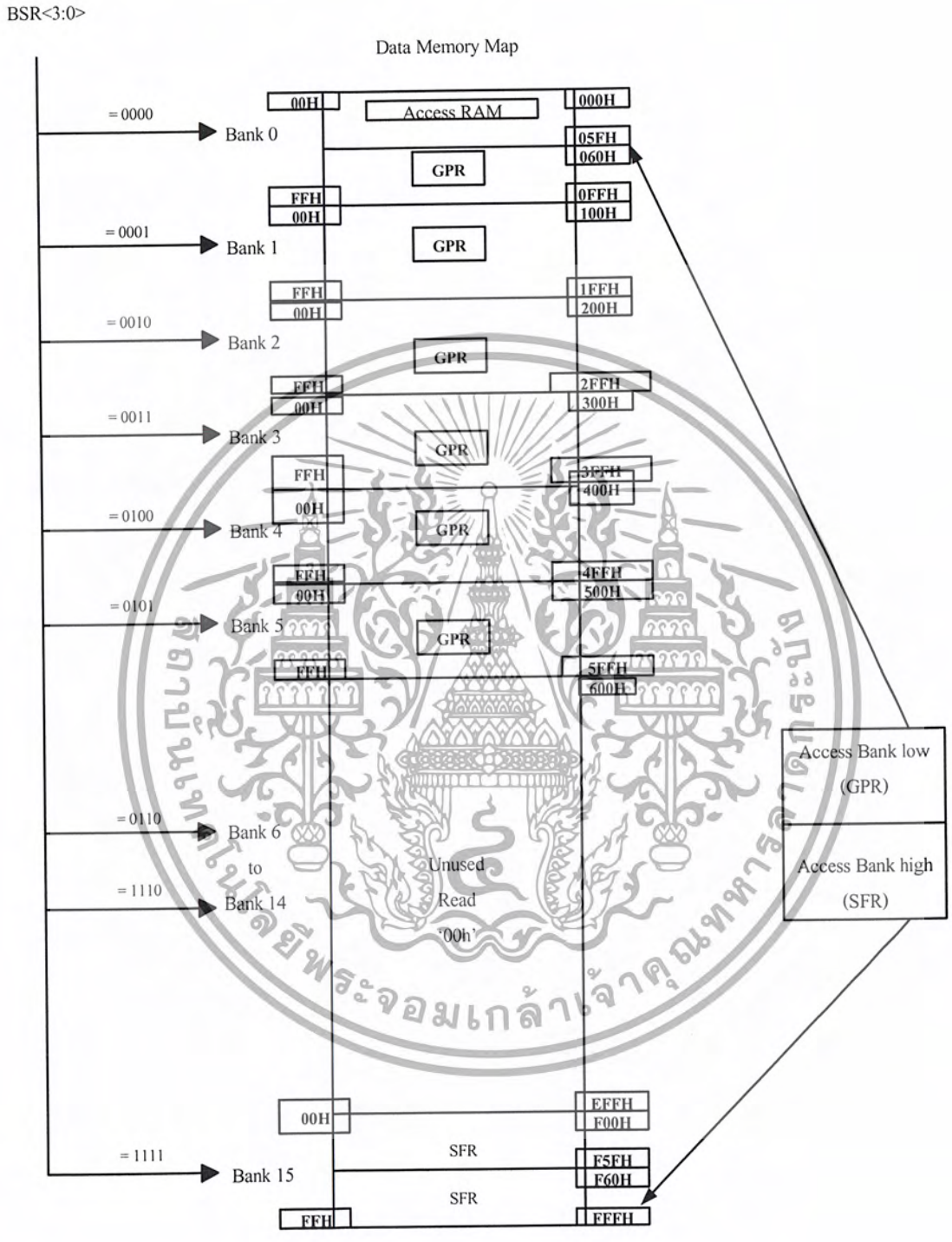
รูปที่ 2.9 การทำงานแบบไปป์ไลน์ของไมโครคอนโทรลเลอร์ PIC18F458

2.2.5 ลักษณะการทำงานแบบไปป์ไลน์ (Pipe line)

ในการประมวณผลคำสั่งของซีพียูนั้นหนึ่งคำสั่งจะประกอบไปด้วย 2 ขั้นตอนหลักๆ ก็คือ การเฟตช์ (Fetch) และเอ็กซิคิวต์ (Execute) จากรูปที่ 2.9 ลักษณะการทำงาน คือ เมื่อซีพียูทำการเฟตช์ คือคำสั่งที่ 1 ต่อจากนั้นซีพียูจะทำการเอ็กซิคิวต์คำสั่งที่ 1 นี้พร้อมกับการเฟตช์ เอาคำสั่งต่อไปคือคำสั่งที่ 2 ออกมาและในขณะที่ทำการเอ็กซิคิวต์คำสั่งที่ 2 ก็จะทำการเฟตช์ เอาคำสั่งที่ 3 ออกมาโดยจะเป็นไปแบบนี้เรื่อยๆ ทำให้การประมวลผลที่ดูเหมือนกับใช้เวลา 2 ไซเคิล ดังรูปที่ 2.9 ในบรรทัดที่ 3 เป็นคำสั่ง BRA SUB_1 ซึ่งเป็นคำสั่งที่ใช้กระโดดไปทำงานยังโปรแกรมย่อยที่กำหนดโดยไม่มีเงื่อนไข การในลักษณะนี้คือซีพียูจะทำการเอ็กซิคิวต์คำสั่งที่ BRA SUB_1 พร้อมกับการเฟตช์เอาคำสั่งที่ 4 คือ BSF PORTA, BIT 3 และเก็บค่าแอดเดรสของคำสั่งที่ 4 จะไม่เกิดขึ้น แต่ซีพียูจะไปทำการเฟตช์คำสั่งในส่วนของโปรแกรมย่อย SUB_1 ออกมาและทำการเอ็กซิคิวต์คำสั่งของโปรแกรมย่อยในไซเคิลที่ 5 (TCY5) ส่วนคำสั่งที่ 4 คือ (BSF PORT A, BIT 3) ที่ได้เฟตช์ค้างไว้นั้นจะได้ทำการเอ็กซิคิวต์ ก็ต่อเมื่อซีพียูออกจากโปรแกรมย่อยแล้วกลับไปทำงานในตำแหน่งนั้นอีกครั้ง ด้วยกระบวนการต่างๆเหล่านี้ทำให้ขบวนการที่เป็นคำสั่งการกระโดดนี้จะใช้เวลาในการทำงานถึง 2 ไซเคิล

2.2.6 หน่วยความจำข้อมูล (Data Memory)

หน่วยความจำข้อมูลของ PIC18F458 เป็นหน่วยจำชนิด Static Ram ซึ่งสามารถเปลี่ยนแปลงข้อมูลได้ตลอดเวลา หน่วยความจำข้อมูลของ PIC18F458 มีขนาด 1536 ไบต์ (Byte) การจัดแบ่งพื้นที่การใช้งานต่างๆ จะแบ่งเป็นพื้นที่ ที่เป็นรีจิสเตอร์ที่ใช้งานทั่วไป (General Purpose Register : GPR) ซึ่งภายใน PIC18F458 จะมีทั้งหมด 6 แบงก์ (Bank) ของพื้นที่แบงก์ข้อมูลทั้งหมดคือ 15 แบงก์ โดยใน 6 แบงก์ ของ GPR นั้นจะมีขนาดแบงก์ละ 96 ไบต์ เท่ากันหมดโดยแอดเดรสเริ่มต้นคือ 000h ถึง 5FFh สำหรับรีจิสเตอร์ฟังก์ชั่นพิเศษ (Special Function Regiter : FSR) มีพื้นที่ทั้งหมด 160 ไบต์ โดยจะอยู่ที่แบงก์ 15 ข้อดีอีกอย่างหนึ่งของ PIC18F458 คือในการเขียนโปรแกรมผู้เขียนไม่จำเป็นที่จะเลือกแบงก์หรือสลับแบงก์ไปมาเหมือนไมโครคอนโทรลเลอร์เบอร์อื่นๆ เช่น PIC16F87X หรือเบอร์อื่นๆ ที่ไม่มีคุณสมบัติดังกล่าว เนื่องจากว่าหน่วยความจำของไมโครคอนโทรลเลอร์ PIC18F458 ไม่ถูกแบ่งเป็นเพจ (Page) คือในการจัดสรรแบงก์ข้อมูลจะอยู่ในเพจเดียวกัน ไม่เหมือน PIC16F87X ที่จะต้องทำการเลือกแบงก์ข้อมูลซึ่งกระทำผ่านรีจิสเตอร์ STATUS ในบิต RP0 และ RP1 ซึ่งในส่วนของ PIC18F458 จะไม่มีบิตดังกล่าวในรีจิสเตอร์ STATUS สำหรับการเลือกแบงก์จะมีรีจิสเตอร์ที่ทำหน้าที่นี้คือรีจิสเตอร์ BSR ซึ่งการเลือกแบงก์นั้นจะใช้บิตที่ 0 ถึงบิตที่ 3 ของรีจิสเตอร์ BSR เช่น ถ้าต้องการเลือกแบงก์ 0 ก็เขียนข้อมูลไปยังรีจิสเตอร์ BSR เป็น 0000h และถ้าต้องการเลือกแบงก์ 1 ก็เขียนข้อมูลไปยัง รีจิสเตอร์ BSR เป็น 0001h เป็นต้น



รูปที่ 2.10 การจัดสรรหน่วยความจำข้อมูลของ PIC18F458

2.3 รีจิสเตอร์หลักของ PIC18F458

2.3.1 รีจิสเตอร์ STATUS

รีจิสเตอร์ STATUS เป็นรีจิสเตอร์ที่บรรจุบิตสถานะทางคณิตศาสตร์ต่างๆ ค่าข้อมูลที่อยู่ในรีจิสเตอร์ STATUS จะเกิดจากการใช้คำสั่งในการเข้าถึงต่างๆ และในบางบิตจะเกิดการเปลี่ยนแปลงโดยอัตโนมัติ ตามผลลัพธ์ของการกระทำทางคณิตศาสตร์ต่างๆ ซึ่งได้แก่บิต Z, DC, OV, N และ C โดยรีจิสเตอร์ STATUS มีแอดแดรสอยู่ที่ FD8h ตารางที่ 2.4 แสดงรายระเอียดบิตต่างๆ ของรีจิส-เตอร์ STATUS

ตารางที่ 2.4 รายระเอียดบิตต่างๆ ของรีจิสเตอร์ STATUS

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
-	-	-	N	OV	Z	DC	C
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
R : อ่านค่าได้, W : เขียนค่าได้, U : ไม่ใช้งานอ่านค่าได้เป็นศูนย์, - : ค่าที่เกิดขึ้นหลังเกิดเพาเวอร์ออนรีเซต							

บิต 7-5 : ไม่ใช้งานอ่านค่าได้เท่ากับศูนย์

บิต 4 N : (Negative Bit) บิตนี้ใช้แสดงสถานะทางคณิตศาสตร์ที่ได้ผลลัพธ์จากการทำ 2's Complement

1 = ผลลัพธ์เป็นบวก

0 = ผลลัพธ์เป็นลบ

บิต 3 : OV (Overflow Bit) บิตแสดงสถานการณ์เกิดโอเวอร์โฟลว์ (Overflow)

1 = มีการเกิดโอเวอร์โฟลว์

0 = ไม่มีการเกิดโอเวอร์โฟลว์

บิต 2 : Z (Zero Bit) บิตแสดงสถานะทางคณิตศาสตร์

1 = เมื่อผลลัพธ์จากการกระทำทางคณิตศาสตร์ และลอจิกมีค่าเท่ากับศูนย์

0 = เมื่อผลลัพธ์จากการกระทำทางคณิตศาสตร์ และลอจิกมีค่าไม่เท่ากับศูนย์

บิต 1 : DC (Digit Carry / Borrow Bit) บิตทดหรือยืมระหว่างหลักเป็นบิตที่แสดงสถานะทางคณิตศาสตร์กรณีที่กระทำคำสั่ง (ADDWF, ADDLW, SUBLW, SUBWF)

1 = ผลลัพธ์การกระทำคำสั่งทางคณิตศาสตร์มีการทดเข้ามาจากบิต 3 มายังบิต 4

0 = ผลลัพธ์การกระทำคำสั่งทางคณิตศาสตร์ไม่มีการทดข้ามจากบิต 3 มายังบิต 4

บิต 0 : C (Carry / Borrow Bit) บิตทดหรือยืมเป็นบิตที่แสดงสถานะทางคณิตศาสตร์กรณีที่กระทำคำสั่ง (ADDWF, ADDLW, SUBLW, SUBWF)

1 = ผลลัพธ์จากการกระทำคำสั่งทางคณิตศาสตร์มีการทดหรือมีการยืมในบิต 7

0 = ผลลัพธ์จากการกระทำคำสั่งทางคณิตศาสตร์ไม่มีการทดหรือมีการยืมในบิต 7

2.3.2 รีจิสเตอร์ RCON

รีจิสเตอร์ RCON เป็นรีจิสเตอร์ที่บรรจุบิตสถานะต่างๆ เกี่ยวกับการเกิดรีเซตแบบต่างๆ โดย รีจิสเตอร์ RCON สามารถเซตและเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์

ตารางที่ 2.5 รายระเอียดบิตต่างๆ ของรีจิสเตอร์ RCON

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
IPEN	-	-	RI	TO	PD	POR	BOR
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
R : อ่านค่าได้, W : เขียนค่าได้, U : ไม่ใช้งานอ่านค่าได้เป็นศูนย์, -n : ค่าที่เกิดขึ้นหลังเกิดเพาเวอร์ออนรีเซต							

บิต 7 : IPEN (Interrupt Priority Enable Bit) บิตเอ็นเอเบิล (Enable) การเลือกลับดับการเกิดอินเตอร์รัพต์

1 = เอ็นเอเบิลการเลือกลำดับการเกิดอินเตอร์รัพต์

0 = ไม่มีการเลือกลำดับการเกิดอินเตอร์รัพต์

บิต 6-5 : ไม่ใช้อ่านค่าได้เป็นศูนย์

บิต 4 : RI (Reset Instruction Flag Bit) บิตแสดงสถานะการรีเซตจากคำสั่งทางซอฟต์แวร์

1 = มีการเกิดรีเซต

0 = ไม่มีการเกิดรีเซต

บิต 3 :TO (Watchdog Time Out Flag Bit) บิตแสดงสถานะเกิดไทม์เอาต์ของวอตซ์ด็อกไทเมอร์

1 = หลังจากเกิดเพาเวอร์อัปไทเมอร์ เมื่อใช้คำสั่ง CLRWDT และ SLEEP

0 = เมื่อเกิดไทม์เอาต์ของวอตซ์ด็อกไทเมอร์

บิต 2 : PD (Power Down Detection Flag Bit) บิตแสดงสถานะการเกิด Power Down

1 = หลังจากเกิด เพาเวอร์อัปไทเมอร์ หรือ มีการใช้คำสั่ง CLRWDT

0 = เมื่อกระทำคำสั่ง SLEEP

บิต 1 : POR (Power On Reset Status Bit) บิตแสดงสถานะของการเกิดเพาเวอร์ออนรีเซต

1 = ไม่มีการเกิดเพาเวอร์ออนรีเซต

0 = เกิดเพาเวอร์ออนรีเซต (ควรจะเซตด้วยซอฟต์แวร์)

บิต 0 : BOR (Brown Out Reset Status Bit) บิตแสดงสถานะการเกิดบราวเอาต์รีเซต

1 = ไม่มีการเกิดบราวเอาต์รีเซต

0 = มีการเกิดบราวเอาต์รีเซต (ควรจะเซตด้วยซอฟต์แวร์)

2.3.3 รีจิสเตอร์ INTCON

เป็นรีจิสเตอร์ที่มีนัยสำคัญสูงสุดสำหรับการอินเตอร์รัพต์ สามารถอ่านและเขียนได้ทุกบิต มีแอดเดรสอยู่ที่ FF2h ใช้ในการเปิดการทำงานอินเตอร์รัพต์รวมและการอินเตอร์รัพต์พื้นฐานของไมโครคอนโทรลเลอร์ PIC มีรายละเอียดบิตต่างๆต่อไปนี้

ตารางที่ 2.6 รายระเอียดบิตต่างๆ ของรีจิสเตอร์ INTCON

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
GIE/GIE	PEIE/GIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
R : อ่านค่าได้, W : เขียนค่าได้, U : ไม่ใช้งานอ่านค่าได้เป็นศูนย์, -n : ค่าที่เกิดขึ้นหลังเกิดเพาเวอร์ออนรีเซต							

บิต 7 : GIE / GIEH (Global Interrupt Enable Bit) บิตเอ็นเอเบิล (Enable) การอินเตอร์-รัพต์รวม และบิตเอ็นเอเบิลการอินเตอร์รัพต์รวมไบต์สูง

1 = เลือกให้มีการอินเตอร์รัพต์เกิดขึ้นทั้งหมด

0 = เลือกไม่ให้มีการอินเตอร์รัพต์เกิดขึ้นทั้งหมด

บิต 6 : PEIE / GIEL (Peripheral Interrupt Enable Bit) บิตเอ็นเอเบิลการอินเตอร์รัพต์จากอุปกรณ์ต่อพ่วง และบิตเอ็นเอเบิลการอินเตอร์รัพต์รวมไบต์ต่ำ

1 = เอ็นเอเบิลการอินเตอร์รัพต์แบบนี้

0 = ดิสเอเบิลการอินเตอร์รัพต์แบบนี้

บิต 5 : TMROIE (TMRO Overflow Interrupt Enable Bit) บิตเอ็นเอเบิลการอินเตอร์รัพต์จากการเกิดโอเวอร์โฟลว์ของไทเมอร์ 0

1 = เอ็นเอเบิลการอินเตอร์รัพต์แบบนี้

0 = ดิสเอเบิลการอินเตอร์รัพต์แบบนี้

บิต 4 : INTOIE (INT0 External Interrupt Enable Bit) บิตเอ็นเอเบิลการอินเตอร์รัพต์จากภายนอกที่ขา RB0 / INT

1 = เอ็นเอเบิลการอินเตอร์รัพต์แบบนี้

0 = ดิสเอเบิลการอินเตอร์รัพต์แบบนี้

บิต 3 : RBIE (Port B Change Interrupt Enable) บิตเอ็นเอเบิลการอินเตอร์รัพต์จากการเปลี่ยนแปลงระดับลอจิกที่พอร์ต B ขา RB4-RB7

บิต 2 : TOIF (TMRO Overflow Interrupt Flag Bit) บิตแสดงการโอเวอร์โฟลว์ของ TMR0

1 = ไทเมอร์ 0 โอเวอร์โฟลว์ ทำให้เกิดอินเตอร์รัพต์หากเอ็นเอเบิลไว้

0 = ไทเมอร์ 0 ไม่เกิดโอเวอร์โฟลว์

บิต 1 : INT0F (RBO / INT External Interrupt Flag Bit) บิตแสดงการอินเตอร์รัพต์จากภายนอกที่ขา RBO / INT

1 = มีสัญญาณอินเตอร์รัพต์จากภายนอกเกิดขึ้นที่ขา RBO / INT

0 = ไม่มีสัญญาณอินเตอร์รัพต์จากภายนอกเกิดขึ้นที่ขา RBO / INT

บิต 0 : RBIF (Port B Chang Interrupt Flag Bit) บิตแสดงการเปลี่ยนแปลงระดับลอจิกที่ขา RB4-RB7

1 = มีการเปลี่ยนแปลงเกิดขึ้นที่ขา RB4-RB7

0 = ไม่มีการเปลี่ยนแปลงเกิดขึ้นที่ขา RB4-RB7

2.3.4 รีจิสเตอร์ W

เป็นรีจิสเตอร์ที่มีบทบาทสำคัญมากที่สุดของไมโครคอนโทรลเลอร์ตระกูล PIC เนื่องจากในการการประมวณผลทางคณิตศาสตร์ ลอจิก การเพิ่มหรือลดค่าต่างๆ ต้องกระทำผ่านรีจิสเตอร์ตัวนี้ทั้งสิ้นหรืออาจกล่าวได้ว่ารีจิสเตอร์ W คือแอกคิวมอดูเลเตอร์ (Accumulator) สำหรับในการเรียกใช้งานรีจิสเตอร์ตัวนี้สามารถกระทำได้ทันทีและตลอดเวลาโดยผ่านการกระทำคำสั่งที่เกี่ยวข้องกับรีจิสเตอร์ W ซึ่งสามารถสังเกตได้จากชื่อคำสั่งที่มีคำว่า W อยู่ด้วย เช่น MOVLW 0x10, MOVWF PORTB การทำงานคือ เมื่อกระทำคำสั่ง MOVLW 0x10 นั้นจะมีการเปลี่ยนค่าในรีจิสเตอร์ W คือจะเป็นการนำค่า 0x10 ที่เป็นเลขฐานสิบหกมาเก็บไว้ยังรีจิสเตอร์ W โดยเมื่อกระทำคำสั่งนี้แล้วค่าในรีจิสเตอร์ W จะมีค่าเท่ากับ “00000001” ส่วนคำสั่ง MOVWF PORTB นั้นจะเป็นการส่งค่าข้อมูลที่อยู่ในรีจิสเตอร์ W ออกไปยัง PORTB เช่นเมื่อกระทำคำสั่งนี้แล้วจะมีข้อมูลออกไปยังพอร์ต เป็น “00000001” หากมีการต่อแอลอีดีไว้ที่ขา RB0 ก็จะทำให้แอลอีดีที่ขานี้ติดหรืออาจกล่าวได้ว่าในการนำข้อมูลไปแสดงผลหรือส่งออกไปยังพอร์ตจะต้องกระทำผ่านรีจิสเตอร์ W เสมอ

2.4 ชุดคำสั่งของ PIC 18F458 และการเข้าถึงรีจิสเตอร์

โครงสร้างชุดคำสั่งของไมโครคอนโทรลเลอร์ PIC 18F458 นั้นจะเป็นแบบ 16 บิต มีคำสั่งทั้งหมด 75 คำสั่งซึ่งเป็นไมโครคอนโทรลเลอร์ที่พัฒนามาจาก PIC16F87X และเบอร์อื่นๆในแบบ 8 บิต ซึ่งไมโครชิปเทคโนโลยี ผู้ผลิตไมโครคอนโทลเลอร์ตระกูล PIC ได้จัดแบ่งชุดคำสั่งที่ใช้กับไมโครคอนโทรลเลอร์ PIC18F458 ออกเป็น 4 กลุ่ม คือ

2.4.1 กลุ่มคำสั่งการจัดการข้อมูลระดับไบต์กับรีจิสเตอร์ไฟล์ (Byte Operiented File Register Operations)

เป็นกลุ่มคำสั่งที่กระทำกับรีจิสเตอร์ไฟล์โดยตรงหรือโดยอ้อม โดยขนาดของข้อมูลอยู่ในระดับไบต์ มีทั้งสิ้น 31 คำสั่ง

2.4.2 กลุ่มคำสั่งการจัดการข้อมูลระดับบิตกับรีจิสเตอร์ไฟล์ (Bit Opriented File Register Operations)

เป็นกลุ่มคำสั่งที่ต้องเกี่ยวข้องกระทำกับรีจิสเตอร์ไฟล์โดยตรงหรือโดยอ้อม โดยขนาดของข้อมูลอยู่ในระดับบิต มีทั้งสิ้น 5 คำสั่ง

2.4.3 กลุ่มคำสั่งจัดการกับค่าคงที่ (Literal Operations)

เป็นกลุ่มคำสั่งที่ผสมกันระหว่างการประมวลผลกับค่าคงที่ คำสั่งเกี่ยวกับการกระโดดไปยังโปรแกรมย่อย และคำสั่งกำหนดโหมดในการทำงานมีทั้งสิ้น 10 คำสั่ง

2.4.4 กลุ่มคำสั่งควบคุมการทำงาน (Control Operations)

เป็นกลุ่มคำสั่งที่เกี่ยวกับการควบคุมลำดับการทำงานของโปรแกรมเช่นคำสั่งการกระโดด คำสั่งเรียกโปรแกรมย่อย คำสั่งออกจากโปรแกรมย่อยเป็นต้น มีทั้งสิ้น 23 คำสั่งนอกจากกลุ่มคำสั่งข้างต้นแล้วยังมีคำสั่งที่เพิ่มเติมเข้ามาอีกคือเป็นคำสั่งที่เกี่ยวข้องกับการอ่านและเขียนหน่วยความจำ มีทั้งสิ้น 10 คำสั่ง

2.4.5 ความหมายของตัวแปรที่ควรทราบ

สามารถสรุปรายละเอียดความหมายของตัวแปรทั้งหมดได้ดังนี้

f หมายถึง ค่าแอดเดรสของรีจิสเตอร์ไฟล์มีค่า 0 – 127 หรือ 0x00 – 0x7F

d หมายถึง ที่เลือกเก็บผลลัพธ์ของการกระทำคำสั่งนั้นๆมีค่าได้ 2 ค่าคือ “0” และ “1”

d = “0” หลังจากกระทำคำสั่งแล้วผลลัพธ์ เก็บไว้ในรีจิสเตอร์ W

d = “1” หลังจากกระทำคำสั่งแล้วผลลัพธ์เก็บไว้ในรีจิสเตอร์ไฟล์คำสั่งนั้นๆ

k หมายถึง ค่าคงที่ใดๆ ในกรณีใช้กับคำสั่งประมวลผลทางคณิตศาสตร์และลอจิก

bbb หมายถึง ข้อมูลระดับบิตหรือบิตภายในรีจิสเตอร์ไฟล์ขนาด 8 บิต

w หมายถึง รีจิสเตอร์ w (แอกคิวมูเลเตอร์)

x หมายถึง ตำแหน่งที่ไม่สนใจเมื่อใช้กับโปรแกรม MPASM จะกำหนดค่า x นี้เท่ากับ "0" โดยอัตโนมัติ

a หมายถึง บิตข้อมูลแรม
a = "0" หน่วยความจำแรมไม่ถูกเลือกแบงก์โดยรีจิสเตอร์ BSR
a = "1" หน่วยความจำแรมถูกเลือกแบงก์โดยรีจิสเตอร์ BSR

* หมายถึง ไม่มีการเปลี่ยนแปลงค่าในรีจิสเตอร์ (กรณีใช้กับคำสั่ง TBLPTR ในการอ่านและเขียนข้อมูลในหน่วยความจำ)

*+ หมายถึง โพสต์ค่าแล้วเพิ่มค่าในรีจิสเตอร์ขึ้น (กรณีใช้กับคำสั่ง TBLPTR ในการอ่านและเขียนข้อมูลในหน่วยความจำ)

*- หมายถึง โพสต์ค่าแล้วลดค่าในรีจิสเตอร์ลง (กรณีใช้กับคำสั่ง TBLPTR ในการอ่านและเขียนข้อมูลในหน่วยความจำ)

+* หมายถึง เพิ่มค่าในรีจิสเตอร์ก่อนแล้วทำการโพสต์ค่า (กรณีใช้กับคำสั่ง TBLPTR ในการอ่านและเขียนข้อมูลในหน่วยความจำ)

label หมายถึง ตำแหน่งของแอดเดรสที่ต้องการกระโดดไปทำงาน

<b> หมายถึง ค่าในระดับบิต

TOS หมายถึง ตำแหน่งบนสุดของสแต็ก (Top Of Stack)

PC หมายถึง โปรแกรมเคาน์เตอร์

PCH หมายถึง โปรแกรมเคาน์เตอร์ไบต์สูง

PCL หมายถึง โปรแกรมเคาน์เตอร์ไบต์ต่ำ

[ ] หมายถึง ออปชั่น หรือส่วนเพิ่มเติม

( ) หมายถึง ค่าของตัวแปรหรือข้อมูล

สำหรับค่าคงที่ของข้อมูลเลขฐานสิบหกที่เกี่ยวข้องในการเขียนโปรแกรมของไมโครคอนโทลเลอร์ PIC จะเขียนเป็น 0x00 ตัวเลข 2 หลักท้ายจะเป็นค่าของเลขฐานสิบหก ส่วน x โดยปกติจะเป็นค่า "0" การเขียนในลักษณะนี้ได้รับการกำหนดมาจากโปรแกรมแอสแซมเบลอร์ MPASM ของไมโครชิป จะเห็นได้ว่าไม่มีตัวอักษร H ต่อท้าย แต่ถ้าเป็นข้อมูลเลขฐานสิบ จะต้องเขียนตัวอักษร d ต่อท้ายด้วย

2.4.5 สรุปคำสั่งทั้งหมดของไมโครคอนโทลเลอร์ PIC18F458

ตารางที่ 2.7 เป็นตารางสรุปกลุ่มคำสั่งทั้งหมดของไมโครคอนโทรลเลอร์ PIC ทั้งนี้ยังไม่รวมคำสั่งเทียมที่ใช้ประกอบในการเขียนโปรแกรม ซึ่งแตกต่างกันขึ้นอยู่กับการใช้ซอฟต์แวร์แอส-แซมเบลอร์สำหรับไมโครคอนโทลเลอร์ PIC ควรใช้โปรแกรม MPASM ซึ่งมีมาพร้อมกับโปรแกรม MPLAB สามารถดาวน์โหลดได้ทางอินเตอร์เน็ตได้ฟรี

ตารางที่ 2.7 สรุปกลุ่มคำสั่งทั้งหมดของไมโครคอนโทลเลอร์ PIC18F458

กลุ่มคำสั่งการจัดการข้อมูลระดับไบต์ของรีจิสเตอร์ไฟล์						
นีโมนิก	โอเปอร์แรนด์	รายระเอียด	ไซเคิล	ออปโค้ด 16 บิต Msb Lsb	บิตสถานะที่เกี่ยวข้อง	หมายเหตุ
ADDWF	f, d, a	บวกค่าใน W ด้วยค่าใน F	1	0010 01da ffff ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	บวกค่าใน W ด้วยค่าใน F และ C	1	0010 00da ffff ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	แอนด์ค่าใน W กับ F	1	0001 01da ffff ffff	Z, N	1, 2
GLRF	f, a	เคลียร์ค่าใน F	1	0110 011a ffff ffff	Z	2
COMF	f, d, a	คอมพลีเมนต์ F	1	0001 11da ffff ffff	Z, N	1, 2
CPFSEQ	f, a	เปรียบเทียบค่าของ F กับ W ข้ามหนึ่งแอดเดรสถ้าค่าเท่ากัน	1 (2 or 3)	0110 001a ffff ffff	None	4
CPFSGT	f, a	เปรียบเทียบค่าของ F กับ W ข้ามหนึ่งแอดเดรสถ้าค่า F มากกว่า W	1 (2 or 3)	0110 01da ffff ffff	None	4
CPFSLT	f, a	เปรียบเทียบค่าของ F กับ W ข้ามหนึ่งแอดเดรสถ้าค่า F น้อยกว่า W	1 (2 or 3)	0010 01da ffff ffff	None	1,2
DECF	f, d, a	ลดค่าใน F ลงหนึ่ง	1	0000 01da ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	ลดค่า F ข้ามหนึ่งแอดเดรสถ้าค่าเป็นศูนย์	1 (2 or 3)	0010 11da ffff ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	ลดค่า F ข้ามหนึ่งแอดเดรสถ้าค่าไม่เป็น ศูนย์	1 (2 or 3)	0100 11da ffff ffff	None	1, 2
INCF	f, d, a	เพิ่มค่า F ขึ้นหนึ่งค่า	1	0010 10da ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	เพิ่มค่า F ข้ามหนึ่งแอดเดรสถ้าค่าเป็นศูนย์	1 (2 or 3)	0011 11da ffff ffff	None	4

ตารางที่ 2.7 (ต่อ) สรุปกลุ่มคำสั่งทั้งหมดของไมโครคอนโทรลเลอร์ PIC18F458

กลุ่มคำสั่งการจัดการข้อมูลระดับไบต์ของรีจิสเตอร์ไฟล์						
นีโมนิก	โอเปอร์แรนด์	รายระเอียด	ไซเคิล	ออปโค้ด 16 บิต Msb Lsb	บิตสถานะที่เกี่ยวข้อง	หมายเหตุ
INFSNZ	f, d	ลดค่า F, ข้ามหนึ่งแอดเดรสถ้าค่าไม่เป็น ศูนย์	1 (2 or 3)	0100 10da ffff ffff	None	1, 2
IORWF	f, d, a	ออร์ค่าใน W กับค่าใน F	1	0001 00da ffff ffff	C, DC, Z, OV, N	1,2
MOVF	f, d, a	โอนย้ายข้อมูลของ F ไปยัง d	1	0101 00da ffff ffff	C, DC, Z, OV, N	1
MOVFF	f, d, a	โอนย้ายข้อมูลของ F ไปยัง F	2	1100 ffff ffff ffff	Z, N	1,2
MOVWF	f, a	โอนย้ายข้อมูลของ W ไปยัง F	1	0110 011a ffff ffff	None	
กลุ่มคำสั่งจัดการข้อมูลระดับบิตของรีจิสเตอร์ไฟล์						
นีโมนิก	โอเปอร์แรนด์	รายระเอียด	ไซเคิล	ออปโค้ด 16 บิต Msb Lsb	บิตสถานะที่เกี่ยวข้อง	หมายเหตุ
BCF	f, b, a	เคลียร์ค่าบิตของ F	1	0001 00da ffff ffff	None	1, 2
BSF	f, b, a	เซตค่าบิตของ F	1	0101 00da ffff ffff	None	1, 2
BTFSC	f, b, a	ทดสอบบิตที่ F ข้ามหนึ่งแอดเดรสถ้าเป็น 0 ถ้าเป็น 1 กระทำบรรทัดต่อไป	1 (2)	1100 ffff ffff ffff	None	3, 4
BTFSS	f, b, a	ทดสอบบิตที่ F ข้ามหนึ่งแอดเดรสถ้าเป็น 1 ถ้าเป็น 0 กระทำบรรทัดต่อไป	1 (2)	0110 011a ffff ffff	None	3, 4
BGT	f, b, a	กลับค่าข้อมูลบิตที่อยู่ใน F	1	0000 001a ffff ffff	None	1, 2

ตารางที่ 2.7 (ต่อ) สรุปกลุ่มคำสั่งทั้งหมดของไมโครคอนโทรลเลอร์ PIC18F458

กลุ่มคำสั่งควบคุมลำดับการทำงาน						
นีโมนิก	โอเปอร์แรนด์	รายระเอียด	ไซเคิล	ออปโค้ด 16 บิต Msb Lsb	บิตสถานะที่เกี่ยวข้อง	หมายเหตุ
BC	n	กระโดดไปยังแอดเดรสที่กำหนดหลังจากตรวจสอบพบว่าแฟล็กทดเป็น1	1 (2)	1110 0010 nnnn nnnn	None	
BN	n	กระโดดไปยังแอดเดรสที่กำหนดหลังจากตรวจสอบพบว่ามีค่ามากกว่า	1 (2)	1110 0110 nnnn nnnn	None	
BNC	n	กระโดดไปยังแอดเดรสที่กำหนดหลังจากตรวจสอบพบว่าแฟล็กทดเป็น 0	1 (2)	1110 0011 nnnn nnnn	None	
BNN	n	กระโดดไปยังแอดเดรสที่กำหนดหลังจากตรวจสอบว่ามีค่าน้อยกว่า	1 (2)	1110 0111 nnnn nnnn	None	
BNOV	n	กระโดดไปยังแอดเดรสที่กำหนดหลังจากตรวจสอบว่าไม่เกิดโอ-เวอร์โฟล์	1 (2)	1110 0101 nnnn nnnn	None	
BNZ	n	กระโดดไปยังแอดเดรสที่กำหนดหลังจากตรวจสอบพบว่าแฟลกศูนย์เป็น 0	2	1110 0001 nnnn nnnn	None	
BOV	n	กระโดดไปยังแอดเดรสที่กำหนดหลังจากตรวจสอบว่าเกิดโอเวอร์-โฟล์	1 (2)	1110 0100 nnnn nnnn	None	
BRA	n	กระโดดไปยังแอดเดรสที่กำหนดโดยไม่มีเงื่อนไข	1 (2)	1101 0nnn nnnn nnnn	None	
BZ	n	กระโดดไปยังแอดเดรสที่กำหนดหลังจากตรวจสอบพบว่าแฟลกศูนย์เป็น 1	1 (2)	1110 0000 nnnn nnnn	None	
CALL	n, s	เรียกโปรแกรมย่อย	2	1110 110s kkkk kkkk 1111 kkkk kkkk kkkk	None	
CLRWDT	-	เคลียร์ค่าวอดซ์ด็อกไทเมอร์	1	0100 11da ffff ffff	TO, PD	
DAW	-	ลดค่าใน W	1	0010 10da ffff ffff	C	

ตารางที่ 2.7 (ต่อ) สรุปกลุ่มคำสั่งทั้งหมดของไมโครคอนโทรลเลอร์ PIC18F458

กลุ่มคำสั่งควบคุมลำดับการทำงาน						
นีโมนิก	โอเปอร์แรนด์	รายระเอียด	ไซเคิล	ออปโค้ด 16 บิต		บิตสถานะที่เกี่ยวข้อง	หมายเหตุ
				Msb	Lsb		
GOTO	n	กระโดดไปยังแอดเดรสที่กำหนด	2	0011 11da 0100 10da	ffff ffff ffff ffff	None	
NOP	-	ไม่มีการทำงาน	1	0000 0000	0000 0000	None	
NOP	-	ไม่มีการทำงาน	1	1111 xxxx	xxxx xxxx	None	
RESET		รีเซตไมโครคอนโทรเลอร์ด้วยคำสั่ง RESET	1	0000 0000	1111 1111	ALL	
RETFIE	s	ออกจากโปรแกรมย่อยพร้อมกับเอ็น-เอเบิลการอินเตอร์รัพต์	2	0000 0000	0001 0000	GIE, GIEH, PEI, GIEL	
RETLW	k	ออกจากโปรแกรมย่อยพร้อมทั้งค่าคงที่ใน W	2	0000 1100	kkkk kkkk	None	
RETURN	s	ออกจากโปรแกรมย่อย	2	0000 0000	0001 001s	None	
SLEEP	-	เข้าสู่โหมดสลีปหรือโหมดประหยัดพลังงาน	1	0000 0000	0000 001s	TO, PD	

2.4.7 การเข้าถึงรีจิสเตอร์และข้อมูลของไมโครคอนโทรลเลอร์ PIC

ไมโครคอนโทรลเลอร์ PIC18F458 มีกระบวนการในการเข้าถึงหน่วยความจำ และรีจิสเตอร์อยู่ 4 แบบ ดังนี้

1) การเข้าถึงแบบทันทีทันใด (Immediate Addressing Mode)

จะเป็นการเข้าถึงค่าคงที่โดยตรง ด้วยการใช้ชุดคำสั่งที่เกี่ยวข้องกับค่าคงที่ (Literal) ร่วมกับรีจิสเตอร์ w หรือ แอกคิวมูเลเตอร์ ตัวอย่างของคำสั่งที่แสดงให้เห็นถึงการเข้าถึงข้อมูลแบบทันทีทันใดได้แก่ Movlw k โดยที่ k คือค่าคงที่

2) การเข้าถึงข้อมูลแบบโดยตรง (Direct Addressing Mode)

เป็นการเข้าถึงข้อมูลหรือรีจิสเตอร์ด้วยการกำหนดแอดเดรสที่ต้องการเข้าถึงอย่างเจาะจง หรือระบุผ่านชื่อของรีจิสเตอร์ก็ได้รูปแบบ เช่น Clrf Temp

3) การเข้าถึงข้อมูลแบบโดยอ้อม (Indirect Addressing Mode)

ในกรณีที่ต้องติดต่อกับรีจิสเตอร์หลายๆตัวในคราวเดียวกันไมโครคอนโทรลเลอร์ PIC นั้น จะมีรีจิสเตอร์ FSR (File Select Register) และรีจิสเตอร์ INDF ช่วยในการติดต่อแบบโดยอ้อม โดย รีจิสเตอร์ FSR ทำหน้าที่เป็นตัวชี้แอดเดรสของหน่วยความจำที่ต้องการเข้าถึง แล้วอ่านข้อมูลนั้นๆ ออกมาจากรีจิสเตอร์ INDF หรืออาจกล่าวได้ว่าใช้รีจิสเตอร์ FSR เก็บค่าแอดเดรสและรีจิสเตอร์ INDF เก็บค่าข้อมูล

4) การเข้าถึงข้อมูลแบบสัมพัทธ์ (Relative Addressing Mode)

การเข้าถึงข้อมูลแบบสัมพัทธ์นี้ประกอบด้วยการคำนวณค่าสัมพัทธ์ของระยะห่างระหว่าง แอดเดรสที่เริ่มต้นทำงานกับแอดเดรสที่ต้องการเข้าถึง โดยมีการใช้ค่าของโปรแกรมเคาน์เตอร์เข้า มาช่วยโดยคำสั่งที่นำมาใช้ในการเข้าถึงข้อมูลแบบสัมพัทธ์ คือ RETLW โดยกระบวนการเริ่มด้วย การกำหนดค่าออฟเซตลงในรีจิสเตอร์ W แล้วนำไปรวมกับค่าของโปรแกรมเคาน์เตอร์ก็จะได้ค่า แอดเดรสของหน่วยความจำที่ต้องการเข้าถึงการเข้าถึงแบบนี้มีข้อจำกัด คือ จะต้องเข้าถึงข้อมูลเป็น ช่วงๆ ช่วงละ 256 ตำแหน่ง

ตัวอย่าง วิธีการเข้าถึงข้อมูลแบบสัมพัทธ์

```
PC          EQU     0x02            ; กำหนดแอดเดรสของ PC
INDEX       EQU     0x70            ; กำหนดแอดเดรสของ
            MOVLW   0x02            ; กำหนดค่าเพื่อที่จะชี้ตำแหน่งไปยัง W
            MOVWF   INDEX           ; นำค่าของตัวชี้ตำแหน่งเก็บที่ INDEX
            MOVF    INDEX, 0        ; นำค่าของ INDEX กลับมายัง W
            CALL    SEGMENT         ; เรียกโปรแกรมย่อย SEGMENT
SEGMENT     ADDWF   PCL, 1          ; บวกค่าตัวชี้ตำแหน่งที่เก็บอยู่ใน W
                                    ; ตำแหน่งกับ PC
```

บทที่ 3

การสร้างและการออกแบบ

3.1 กล่าวนำ

ชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 มีการออกแบบส่วนที่ใช้ในการทดลอง คือ การออกแบบทางด้านฮาร์ดแวร์ (Hard Ware) ซึ่งในชุดทดลองนี้จะแบ่งออกเป็นโมดูล (Module) ตามลักษณะการทดลองต่างๆ ซึ่งแต่ละโมดูลจะสามารถเชื่อมต่อในการทดลองโดยผ่านคอนเน็ตเตอร์ที่ได้ออกแบบไว้โดยเชื่อมต่อผ่านโมดูล PIC-ICSP ซึ่งเป็นโมดูลหลักในการทดลอง เมื่อต้องการใช้งานร่วมกับโมดูลใดก็สามารถเชื่อมต่อแล้วเขียนโปรแกรมทำการคอมไพล์ และดาวน์โหลดข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ และทำการรันผลการทดลองแล้วจึงเชื่อมต่อผ่านพอร์ตใช้งานนั้น

3.2 การสร้างและการออกแบบโมดูลชุดทดลอง

3.2.1 โมดูลหลัก PIC - ICSP (PIC In Circuit Serial Programing)

เป็นโมดูลที่มีความสำคัญมากที่สุด เนื่องจากเป็นโมดูลที่เชื่อมต่อกับโมดูลต่างๆ ให้ทำงานตามการสั่งงานของซีพียู โดยในการโปรแกรมข้อมูลบนตัวซีพียูนี้จะเป็นลักษณะ In Circuit Serial Programing จากพอร์ตขนานของคอมพิวเตอร์ ในโมดูลนี้สามารถที่จะใช้ร่วมกับไมโคร-คอนโทรลเลอร์ 40 ขา ใช้ตัวถังแบบ PDIP ของไมโครคอนโทรลเลอร์ตระกูล PIC ได้

1. คุณสมบัติของ PIC ICSP Module

1.1 สามารถเลือกโหมดสัญญาณนาฬิกาได้ 2 แบบ คือ แบบ XT และแบบ RC โดยการเซตที่จั๊มเปอร์ (Jumper)

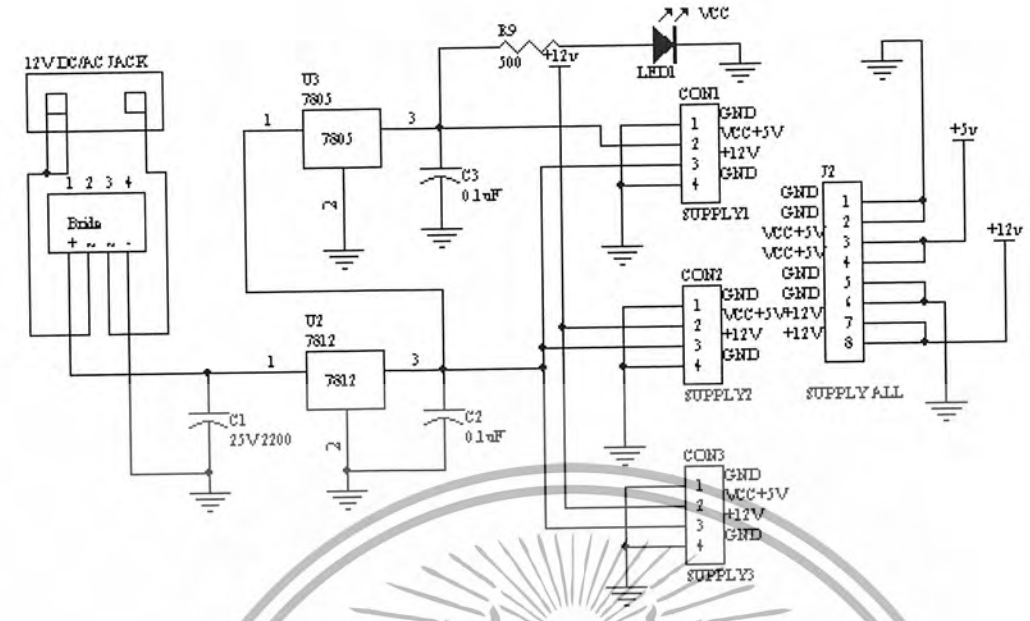
1.2 มีคอนเน็คเตอร์ที่ต่อร่วมกับพอร์ตทั้งหมดของไมโครคอนโทรลเลอร์เพื่อใช้ในการต่อทดลองต่างๆ

1.3 ออกแบบคอนเน็คเตอร์แยกพอร์ตขนาด 10 ขา 5 พอร์ต

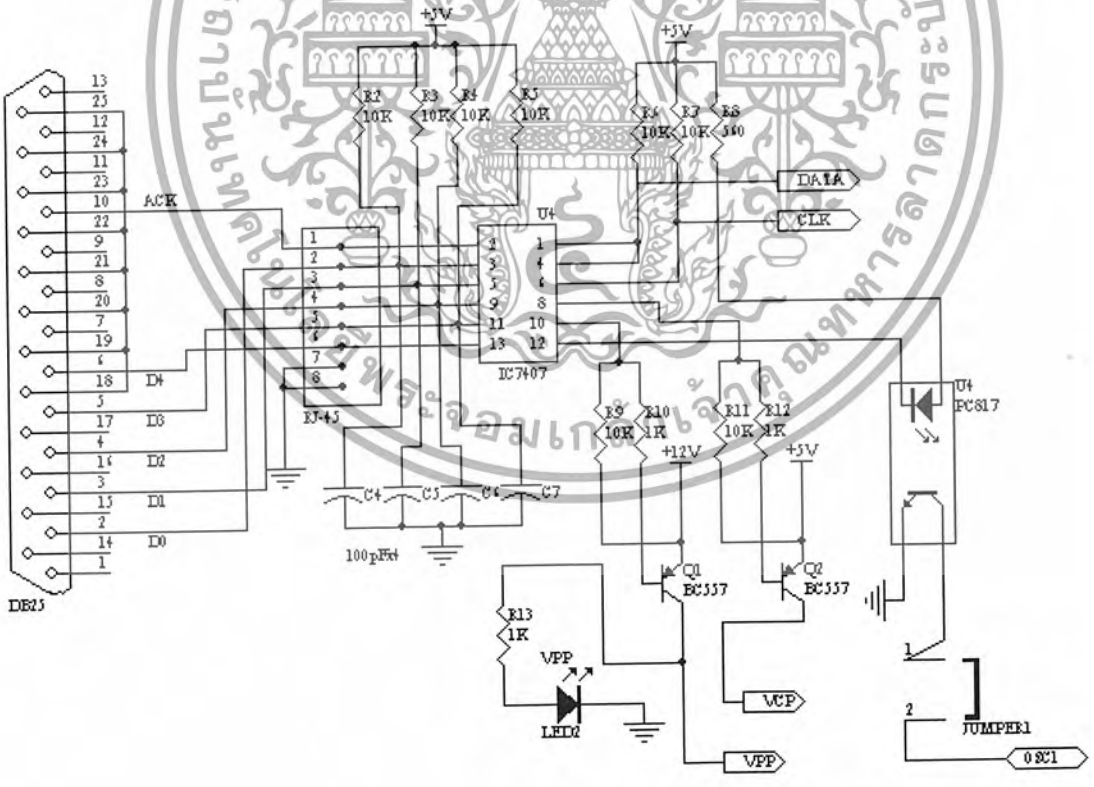
1.4 ออกแบบคอนเน็คเตอร์ All Port ขนาด 40 ขา ที่เป็นแบบเลือกบิตในการเชื่อมต่อได้

1.4 ในการรันโปรแกรมในการทดลองมี SW ที่ทำหน้าที่ตัดต่อระหว่างรัน (Run) และโปรแกรม (Program)

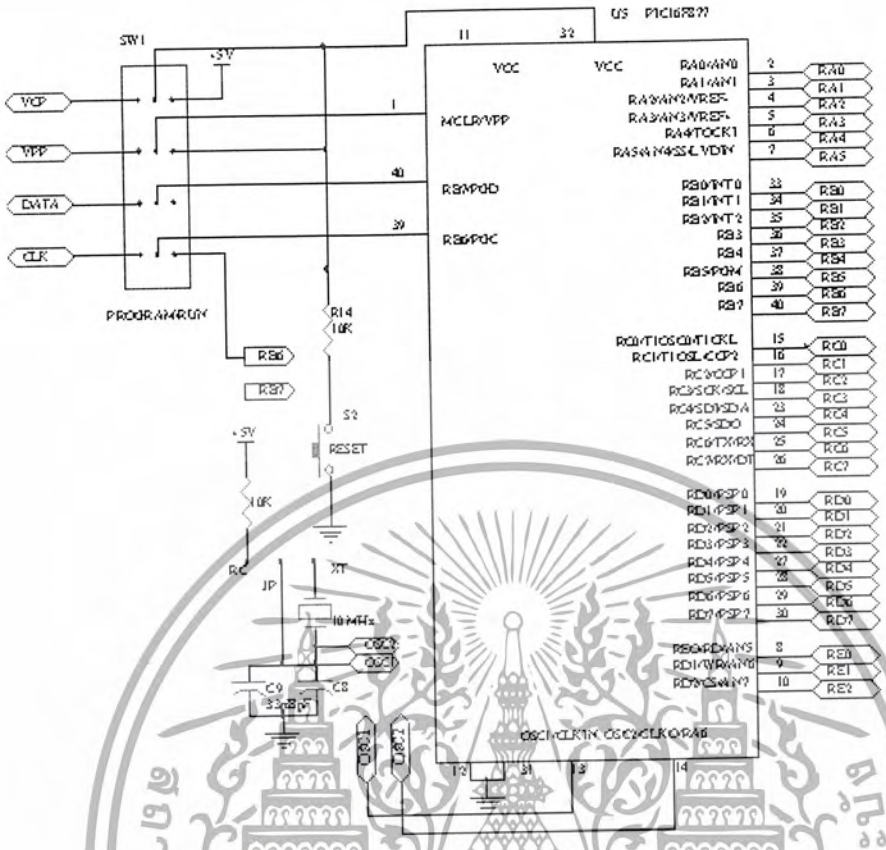
1.5 ออกแบบแหล่งจ่ายไฟตรง 5 โวลต์ และ 12 โวลต์ แบบคอนเน็คเตอร์ตัวผู้ 3 ชุด และตัวเมียที่เป็นแบบ All Supply จำนวน 1 ชุด



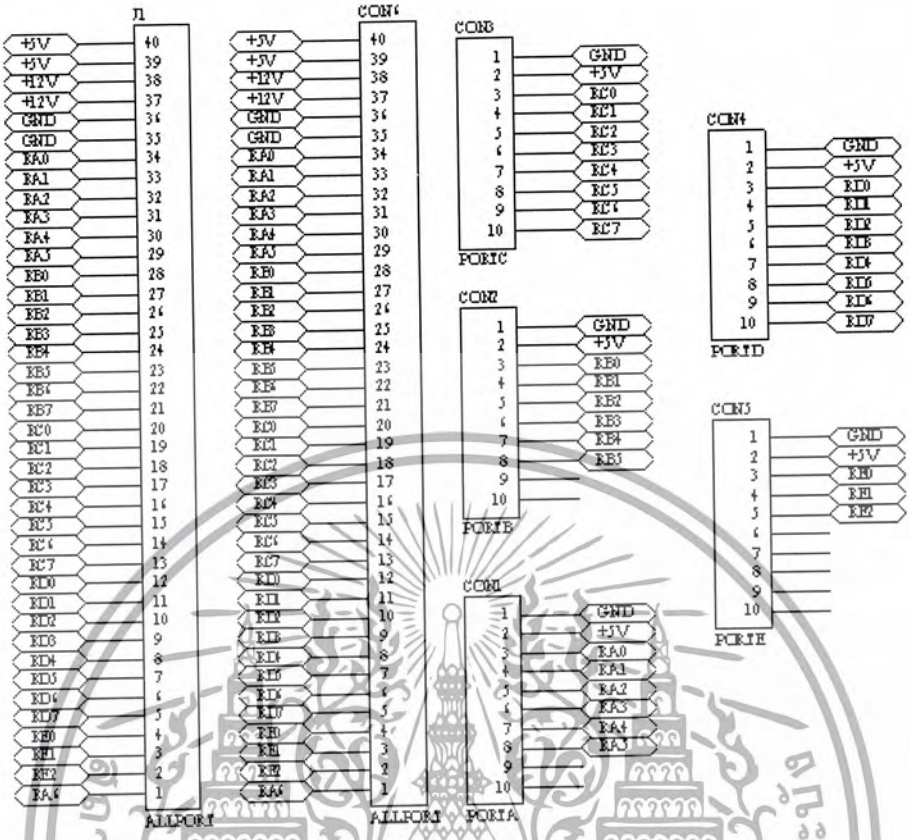
รูปที่ 3.1 วงจรภาคเพาเวอร์ซัพพลาย



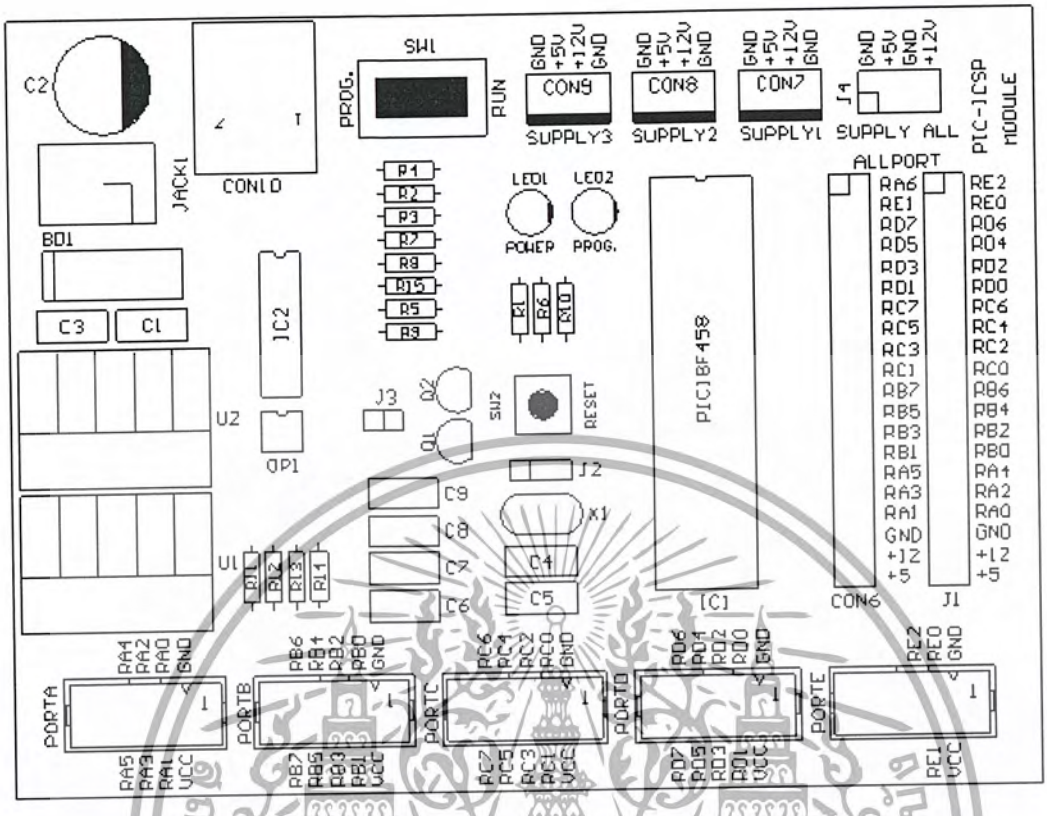
รูปที่ 3.2 วงจรภาค In Circuit Serial Programing



รูปที่ 3.3 ภาคกำเนิดสัญญาณนาฬิกาและซีพียู



รูปที่ 3.4 ภาคเชื่อมต่อพอร์ตใช้งาน

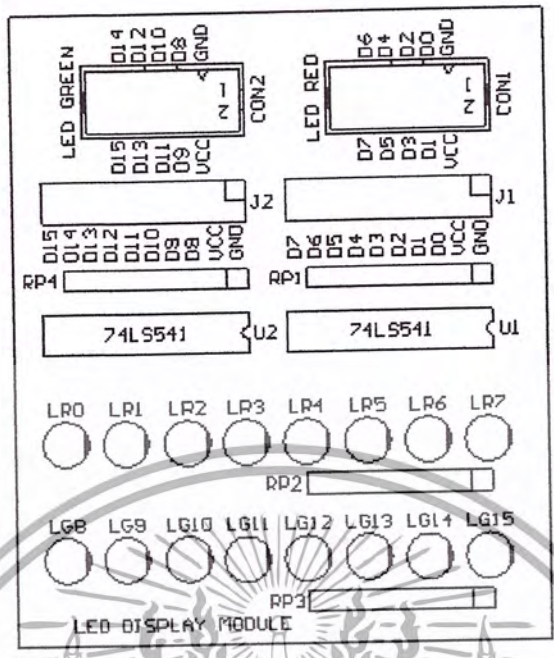


รูปที่ **3.5** แผนผังอุปกรณ์ของโมดูลหลัก PIC-ICSP

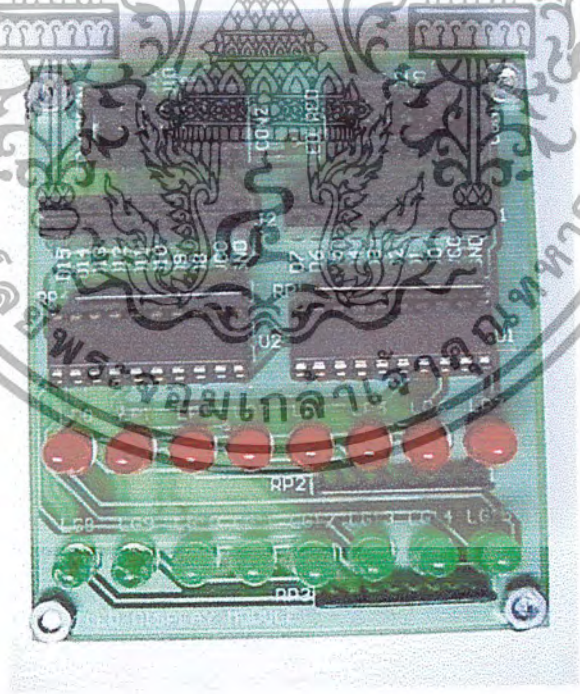
3.3.2 โมดูลแสดงผลแอลอีดี (LED Display Module)

โมดูลแสดงผลแอลอีดีเป็นโมดูลที่ทำหน้าที่แสดงผลการทำงานของส่วนต่างๆ ที่ต่อร่วมโดยแสดงผลออกมาเป็นแสงสีเขียว และสีแดงมีแอลอีดีทั้งหมด 16 ดวง แบ่งออกเป็นสีแดง 8 ดวง สีเขียว 8 ดวง และมีไอซี 74LS541 เป็นบัฟเฟอร์ให้แก่วงจร และใช้ตัวต้านทาน 10 กิโลโอห์ม ต่อพูลอัป (Pull Up) ไว้ด้วย ส่วนการใช้งานจะมีคอนเน็คเตอร์ตัวเมีย และตัวผู้รวมแล้ว 4 ตัว ใช้เชื่อมต่อเพื่อใช้งานรับข้อมูลเพื่อแสดงผล โดยคอนเน็คเตอร์ตัวผู้จะต่อใช้งานร่วมกับพอร์ตต่างๆ ของโมดูลหลักโดยตรง ส่วนตัวเมียจะใช้เชื่อมต่อแบบเลือกได้ว่าจะใช้บิตไหน ถ้าต้องการเชื่อมต่อกับโมดูลหลักเพื่อใช้งานพอร์ตต่างๆ ก็สามารถจั๊มสายไปยังบิตดังกล่าวได้ทันที

รูปที่ 3.6 วงจร โมดูลแสดงผลแอลอีดี



รูปที่ 3.5 แผนผังอุปกรณ์ของโมดูลแสดงผลแอลอีดี



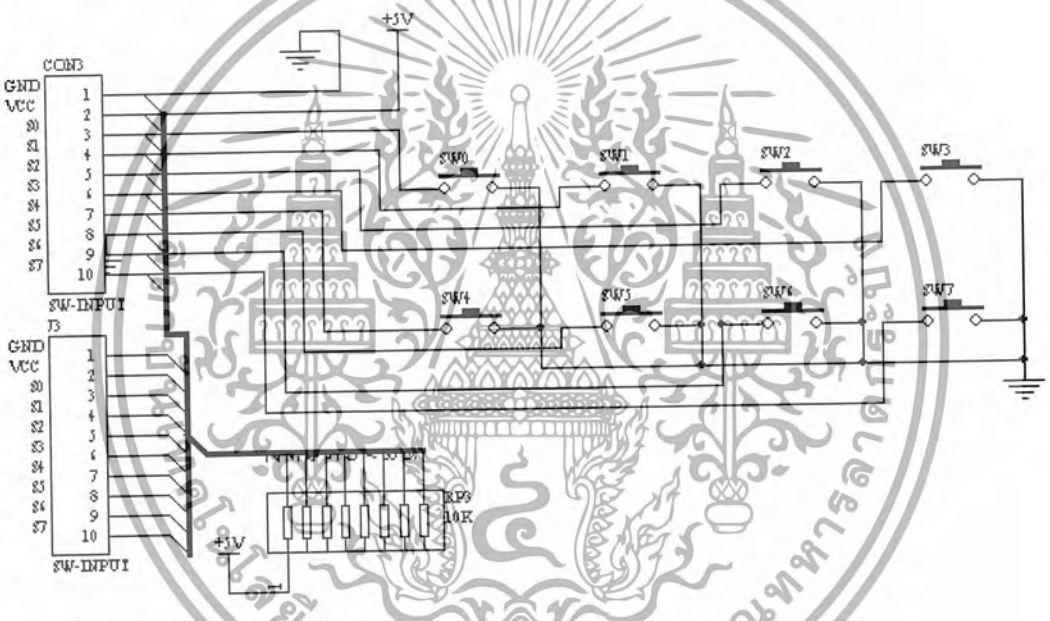
รูปที่ 3.6 โมดูลแสดงผลแอลอีดี

3.3.3 โมดูลสวิตช์พื้นฐาน (Basic Swith Module)

โมดูลสวิตช์พื้นฐาน เป็นโมดูลใช้งานทดลองร่วมกับสวิตช์พื้นฐานต่างๆ ดังนี้

1. สวิตช์อินพุต (Swith Input)

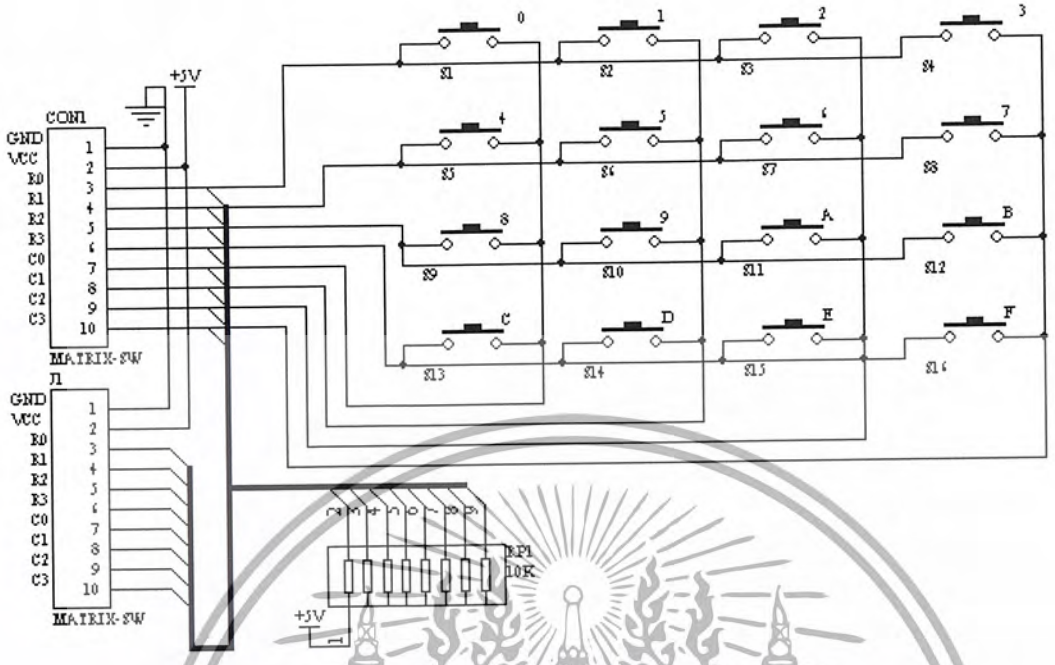
เป็นวงจรที่ต่อใช้งานเป็นอินพุตให้แก่ไมโครคอนโทรลเลอร์ มีสวิตช์กดติดปล่อยดับทั้งหมด 8 ตัวสามารถเลือกใช้งานบิตไหนก็ได้ คือ SW0 – SW7 ส่วน RP 3 ทำหน้าที่เป็นตัวต้านทานพูลอัปให้แก่วงจร ในการใช้งานจะมีคอนเน็คเตอร์ CON 3 ใช้งานในการเชื่อมต่อกับพอร์ตของไมโครคอนโทรลเลอร์ ส่วนคอนเน็คเตอร์ J3 จะใช้เชื่อมต่อกับพอร์ตของไมโครคอนโทรลเลอร์เช่นกัน แต่ในการใช้งานสามารถเลือกได้ว่าจะให้ทำงานในบิตไหน



รูปที่ 3.9 วงจรโมดูลสวิตช์อินพุต

2. เมตริกซ์สวิตช์ (Matrix Switch)

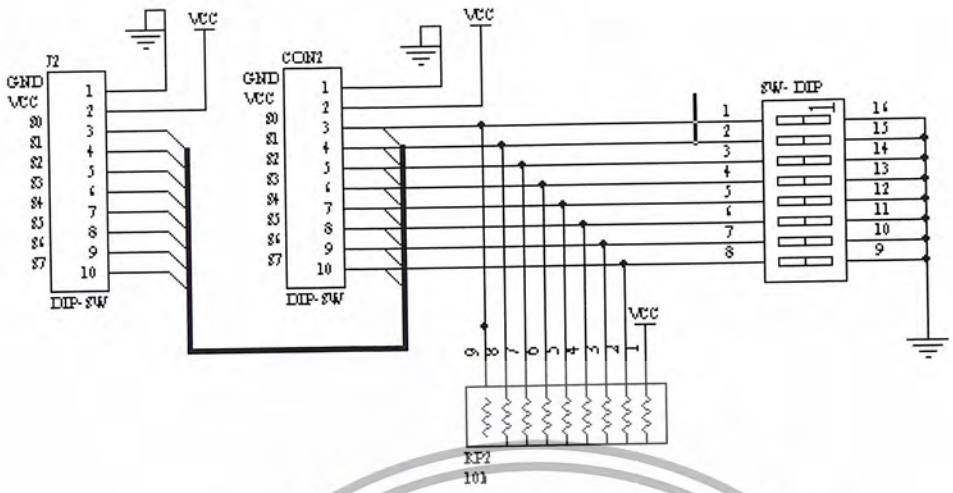
เป็นวงจรที่ใช้เป็นสวิตช์พื้นฐานเช่นกัน โดยจะออกแบบสวิตช์ในลักษณะต่อกันเป็นแบบเมตริกซ์ (Matrix) มีจำนวนแถว 4 แถว และจำนวนหลัก 4 หลัก ในการต่อลักษณะนี้ก็เพื่อลดจำนวนลงเมื่อต้องการใช้งานสวิตช์จำนวนมากๆ โดยจะใช้หลักการตรวจสอบการกดสวิตช์ที่ตำแหน่งต่างๆ จากนั้นก็จะอ่านค่าออกมายังพอร์ตเพื่อใช้งาน ในวงจรมีตัวต้านทาน 10 กิโลโอห์ม ทำหน้าที่เป็นตัวต้านทานพูลอัปให้แก่วงจร และคอนเน็คเตอร์ใช้เชื่อมต่อร่วมกับโมดูลหลักติดต่อกับพอร์ตต่างๆ ของไมโครคอนโทรลเลอร์



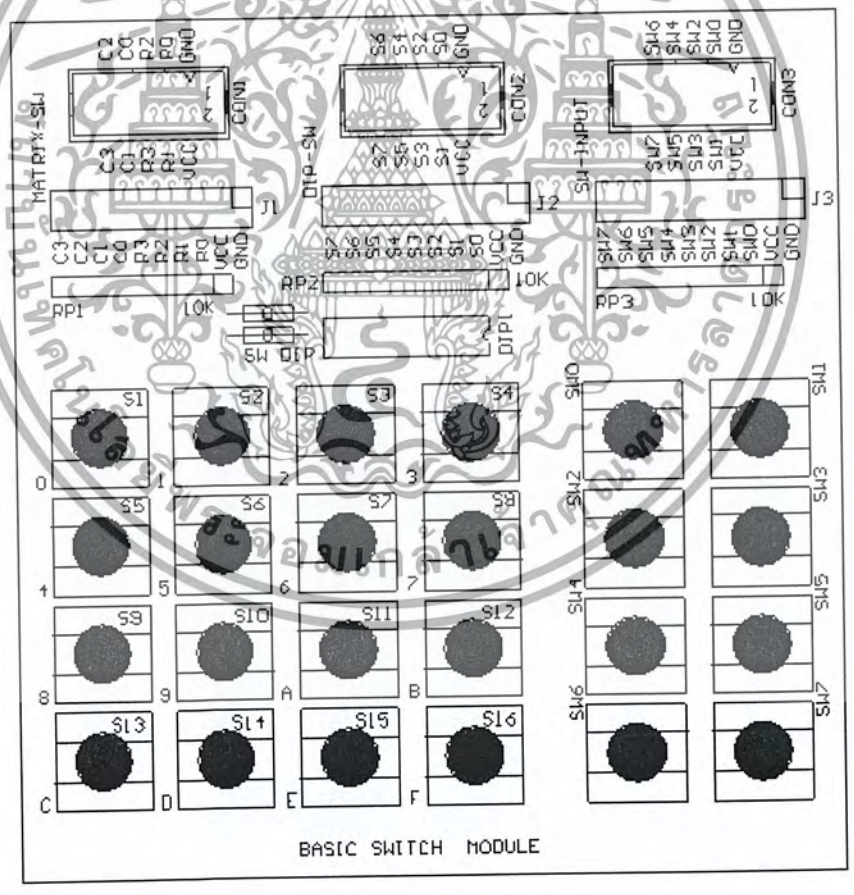
รูปที่ 3.10 วงจรแมตริกซ์สวิตช์ 4x4

3. ดิฟสวิตซ์ (Dip Switch)

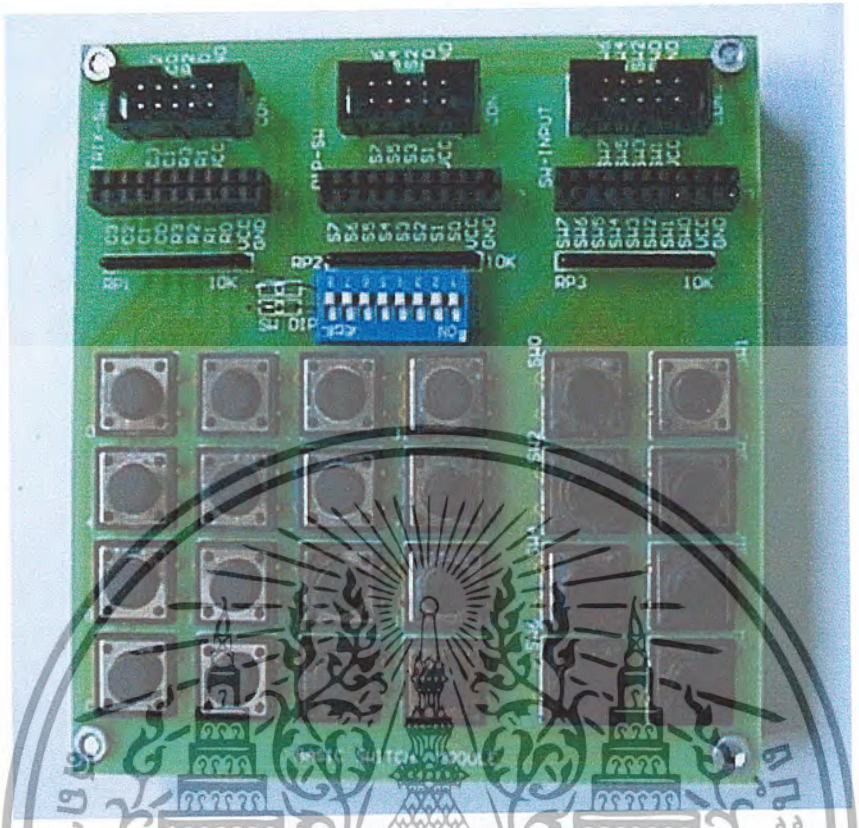
วงจรดิฟสวิตช์ใช้สำหรับการทดลองใช้งานเป็นอินพุตให้แก่ไมโครคอนโทรลเลอร์โดยจะใช้ดิฟสวิตช์ที่มีจำนวน 8 บิต และในวงจรมีตัวต้านทานต่อพูลอัป 10 กิโลโอห์ม โดยมีคอนเน็คเตอร์ CON 2 ใช้เชื่อมต่อกับพอร์ตไมโครคอนโทรลเลอร์โดยเชื่อมต่อทั้ง 8 บิต ส่วน J2 จะใช้เชื่อมต่อกับพอร์ตของไมโครคอนโทรลเลอร์แบบเลือกบิตได้



รูปที่ 3.11 วงจรดิพสวิตช์



รูปที่ 3.12 แผนผังอุปกรณ์ของโมดูลสวิตช์พื้นฐาน

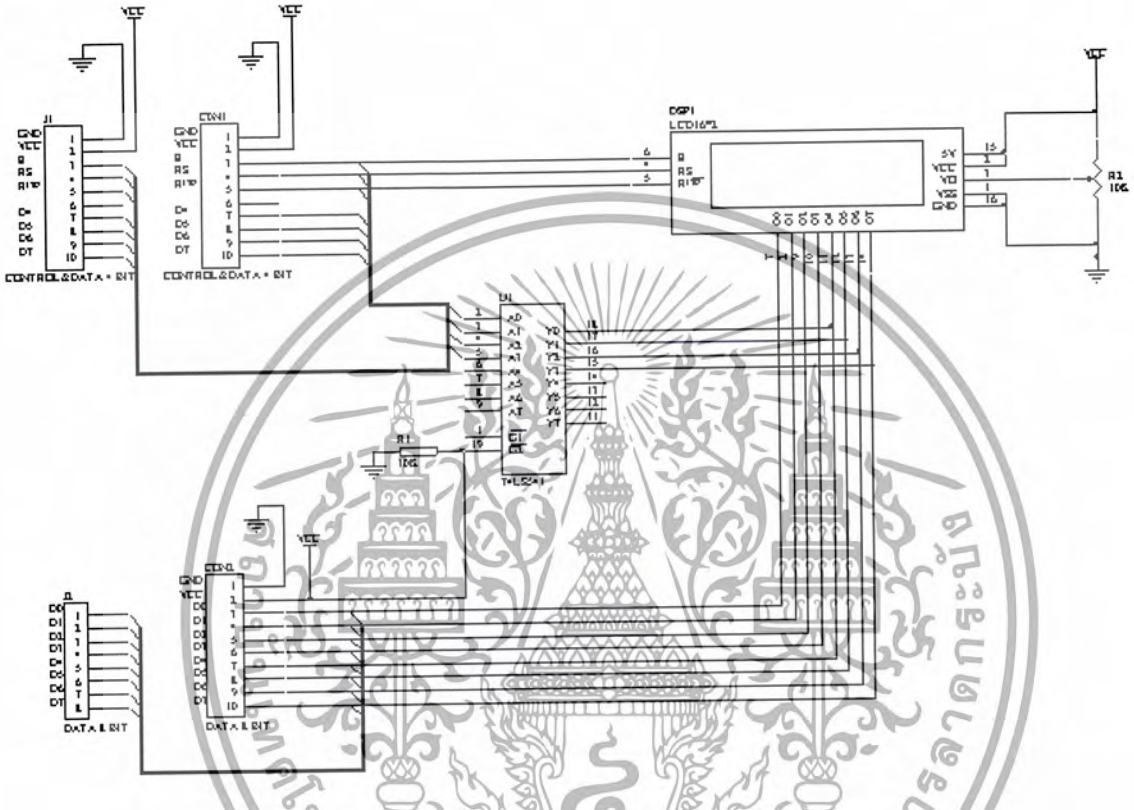


รูปที่ 3.13 โมดูลสวิตช์พื้นฐาน

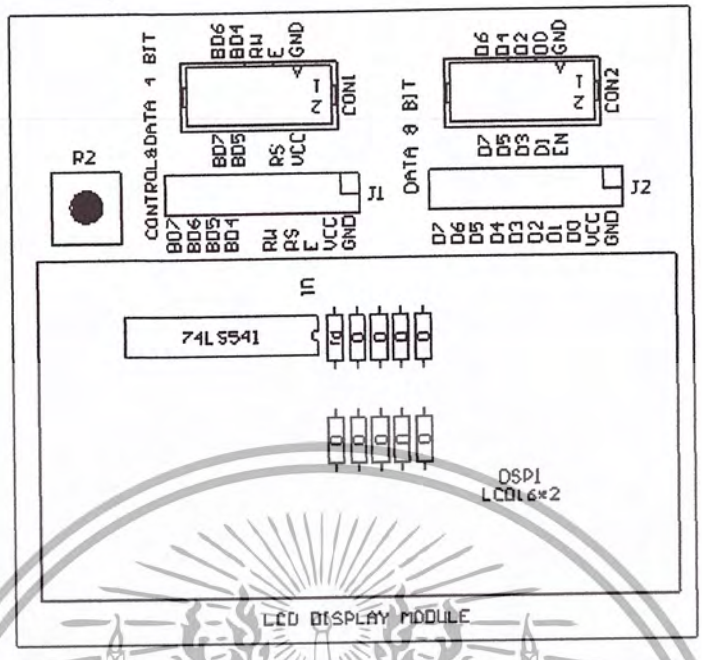
3.3.4 โมดูลแสดงผลแอลซีดี (LCD Display Module)

โมดูลแสดงผลแอลซีดีเป็นโมดูลที่ใช้แอลซีดีแสดงผลเป็นแบบ 16 ตัวอักษร 2 บรรทัดมีขาทั้งหมด 14 ขาโดยแบ่งออกเป็นขาข้อมูล 8 เส้น และขาสัญญาณ 4 เส้นตัวต้านทานปรับค่าได้ 10 กิโลโอห์ม ทำหน้าที่ปรับความสว่างของจอแสดงผล ในมาตรฐานการเชื่อมต่อเพื่อเขียนข้อมูลไปยังแอลซีดีจะเป็นแบบ 8 บิต และ 4 บิต โดยเมื่อใช้งาน 8 บิตก็ใช้ขาสัญญาณ D0 – D7 ในการรับส่งข้อมูลส่วนแบบ 4 บิต จะใช้เพียง D0 – D3 ให้ต่อลงกราวน์ในการออกแบบวงจรแสดงผลแอลซีดีจะมี IC 74LS541 เป็นตัวทำหน้าที่ปิดและเปิดการทำงานแบบ 8 บิต และ 4 บิตโดยเมื่อต้องการใช้งานแบบ 4 บิตก็สามารถเชื่อมต่อคอนเน็คเตอร์ CON 1 เข้ากับพอร์ตของไมโครคอนโทรลเลอร์ และลอยคอนเน็คเตอร์ CON 2 ไว้ก็จะเป็นการเชื่อมต่อข้อมูลแบบ 4 บิต คือ D4 – D7 ผ่านเข้าไปยังบัฟเฟอร์ ส่วนในการใช้งานแบบ 8 บิตก็สามารถทำได้โดยเชื่อมต่อผ่านพอร์ต CON 2 ก็จะมีแรงดัน VCC

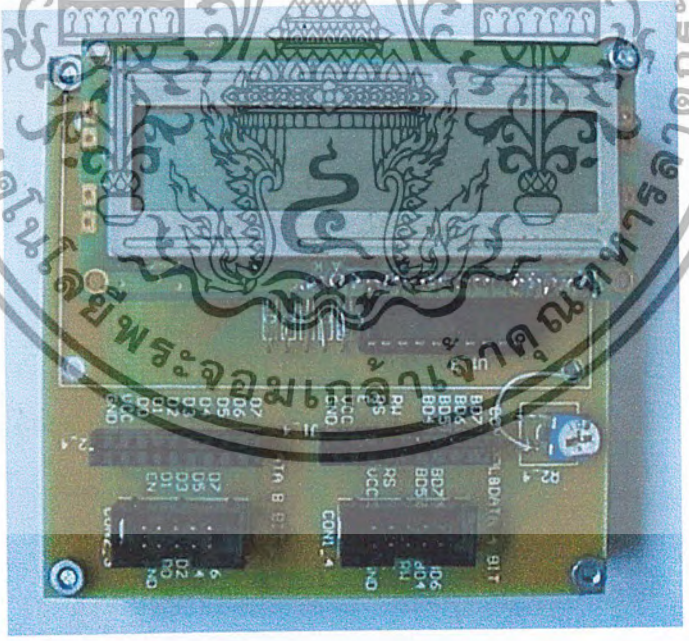
จากโมดูลหลักไปปิดการทำงานของบัพเฟอร์ทำให้เชื่อมต่อเป็นแบบ 8 บิต คือ D0 – D7 ส่วนคอนเน็คเตอร์ J1 และ J2 เป็นคอนเน็คเตอร์เชื่อมต่อสัญญาณควบคุม และบิตข้อมูลแบบเลือกการเชื่อมต่อได้



รูปที่ 3.14 วงจรแสดงผลแอลซีดี



รูปที่ 3.15 แผนผังอุปกรณ์ของโมดูลแสดงผลแอลซีดี



รูปที่ 3.16 โมดูลแสดงผลแอลซีดี

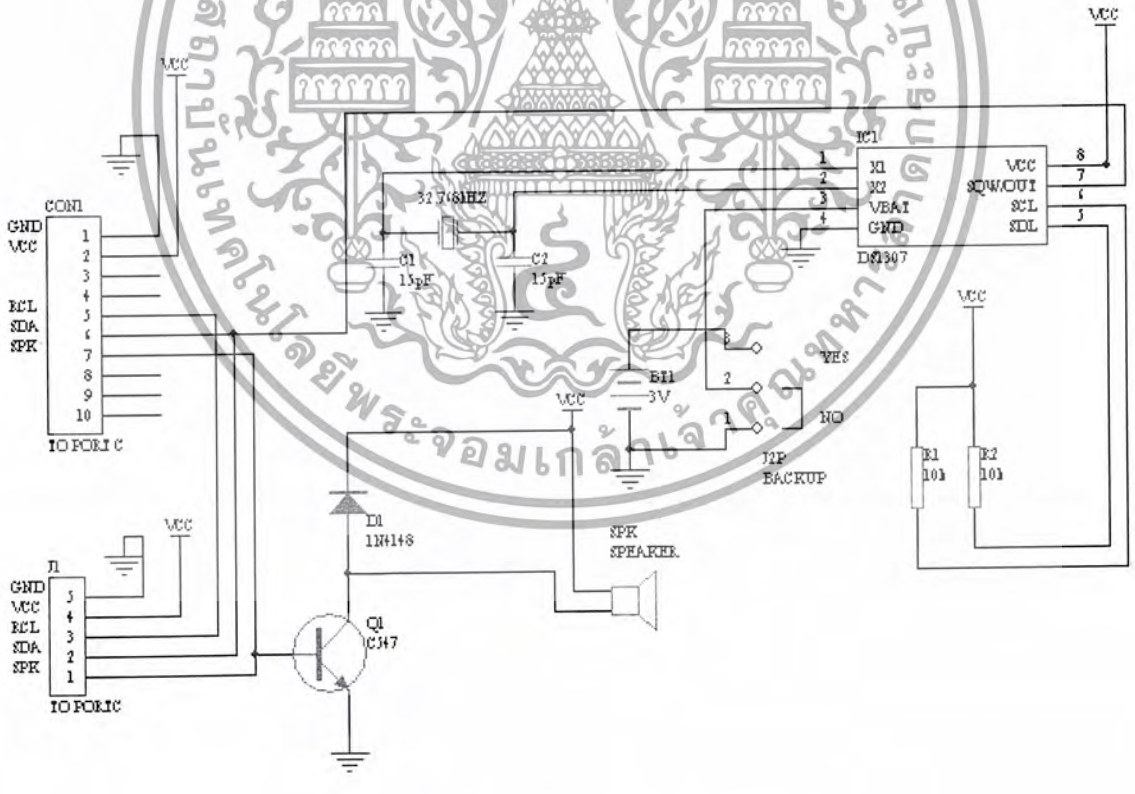
3.3.5 โมดูลเชื่อมต่ออุปกรณ์ RTC / Speaker (RTC / Speaker Interfacing Module)

1. RTC (Real Time Clock)

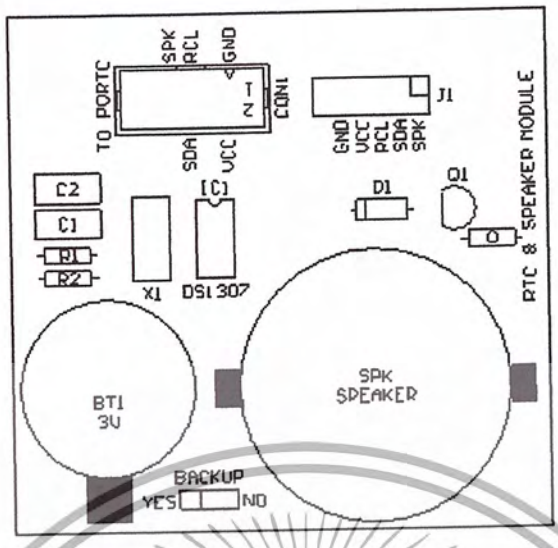
วงจรทดลอง RTC จะใช้ขาสัญญาณในการติดต่อกับไมโครคอนโทรลเลอร์ 2 ขา คือ SCL และ SDA เป็นขาเชื่อมต่อแบบ I^2C ส่วน SQW Out จะมีสัญญาณรูปสี่เหลี่ยมส่งออกมา โดยสามารถเลือกได้ คือ 1 Hz, 4.096 KHz, 8.192 KHz, และ 32 KHz สำหรับ JP1 ใช้สำหรับการเคลียร์ค่าข้อมูลใน DS 1307 คือเมื่อต้องการเริ่มค่าเวลาใหม่ให้ทำการเซตค่าไปยังตำแหน่ง Yes หรือปลดถ่านออก ในการเชื่อมต่อการใช้งานจะใช้ต่อร่วมกับพอร์ต C ที่บิต 3 และ บิต 4 ตามลำดับ

2. ลำโพง (SPEAKER)

วงจรเชื่อมต่อลำโพงจะใช้ร่วมกับไมโครคอนโทรลเลอร์โดยให้ไมโครคอนโทรลเลอร์สร้างพัลส์เพื่อกำหนดความถี่ให้กับลำโพงเปล่งเสียงออกมาตามความถี่ที่ป้อนให้ โดยในวงจรจะมีทรานซิสเตอร์ Q1 ทำหน้าที่ปิดเปิดการทำงานของลำโพงโดยจะมีพัลส์รับเข้ามาที่ขาเบสของ Q1 ส่วน D1 จะต่อไว้เพื่อป้องกันแรงดันย้อนกลับ



รูปที่ 3.17 วงจรเชื่อมต่ออุปกรณ์ RTC / SPEAKER



รูปที่ 3.18 แผนผังอุปกรณ์ของ โมดูลเชื่อมต่ออุปกรณ์ RTC / SPEAKER

รูปที่ 3.19 โมดูลเชื่อมต่ออุปกรณ์ RTC / SPEAKER

3.3.6 โมดูลสื่อสารข้อมูลอนุกรม (Serial Interfacing Module)

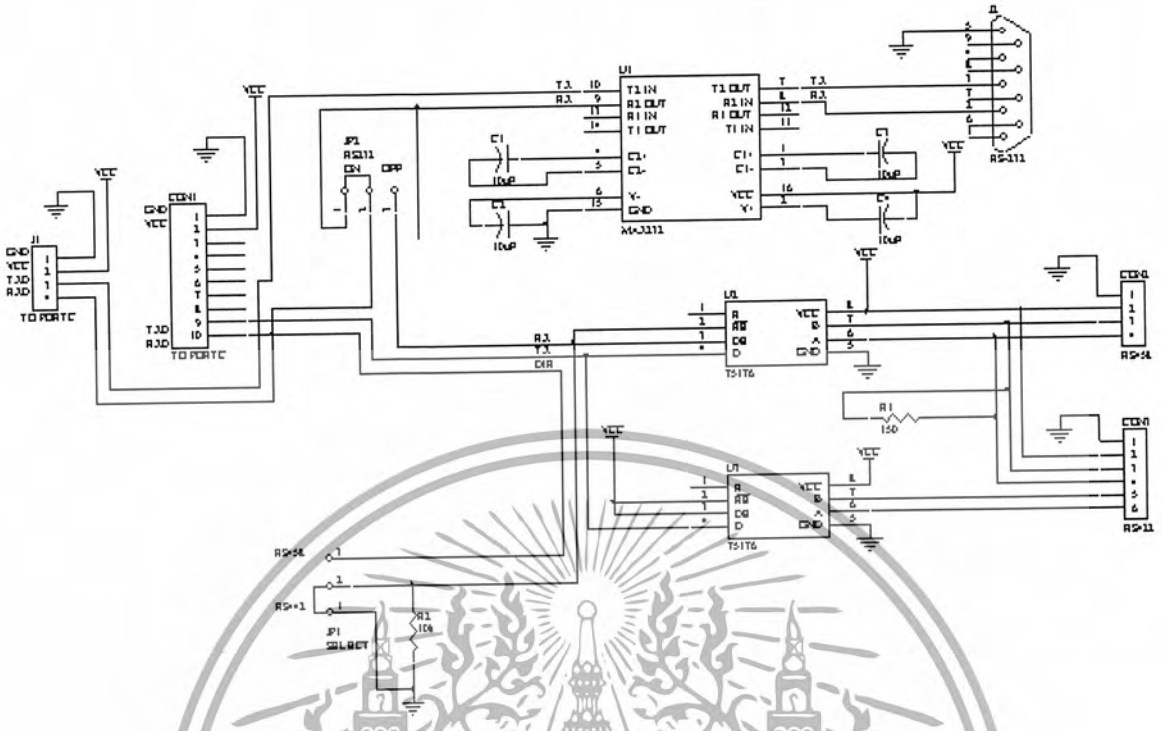
ในโมดูลสื่อสารข้อมูลอนุกรมจะประกอบไปด้วยวงจรการทดลองในมาตรฐาน RS - 232, RS - 485 และ RS -422

1. RS – 232 Interface

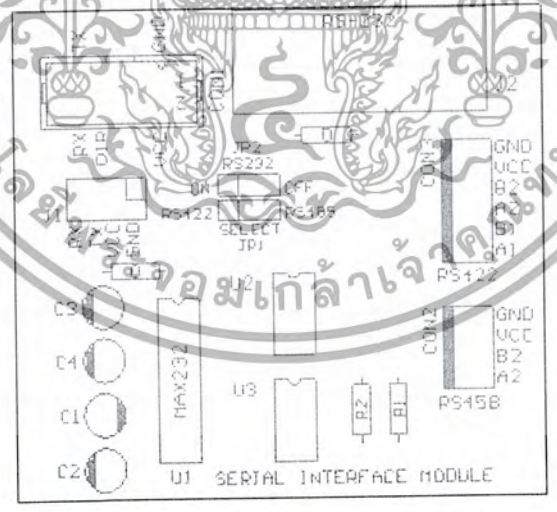
ในการสื่อสารแบบนี้จะใช้ IC Max 232 ในการแปลงสัญญาณจากไมโครคอนโทรลเลอร์ไปสู่มาตรฐาน RS – 232 โดยติดต่อระหว่างพอร์ต RC6 (TX) และ RC7 (RX) ของไมโครคอนโทรลเลอร์ ในการใช้งาน RS – 232 จะมีจั๊มเปอร์ให้เลือกตำแหน่ง โดยให้เลือกไปที่ตำแหน่ง On ก็จะเป็นการเชื่อมต่อการสื่อสารแบบนี้ไปยังคอนเน็คเตอร์ Con 1 และ J1 เพื่อใช้งานร่วมกับพอร์ต C ของไมโครคอนโทรลเลอร์บิต 6 และ บิต 7

2. RS - 485 / RS – 422 Interface

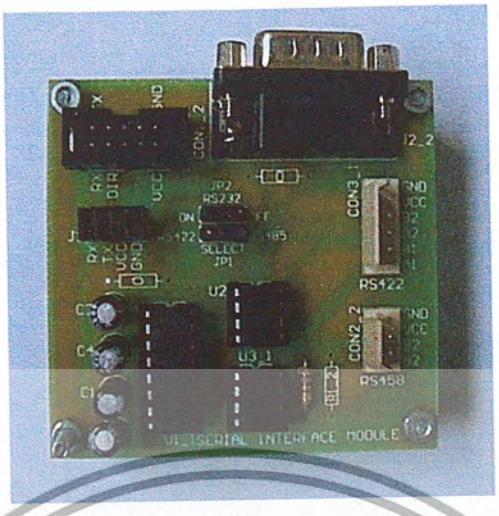
การสื่อสารแบบนี้จะใช้ IC SN 75176 (U2 และ U3) ของการเชื่อมต่อในมาตรฐาน RS-485 จะเป็นการเชื่อมต่อแบบ Half Duplex ส่วน RS-422 จะเป็นแบบ Full Duplex ในการเลือกใช้งานของมาตรฐานทั้ง 2 นี้จะต้องทำการเลือกจั๊มเปอร์ JP2 ไปยังตำแหน่ง Off ก่อนจึงจะสามารถเลือกใช้งาน RS-485 และ RS-422 ได้ และเมื่อต้องการเลือกใช้งานแบบไหนก็เลือก JP1 โดยเลือกจั๊มไปยังตำแหน่งมาตรฐานนั้นๆ



รูปที่ 3.20 วงจรโมดูลสื่อสารข้อมูลอนุกรม



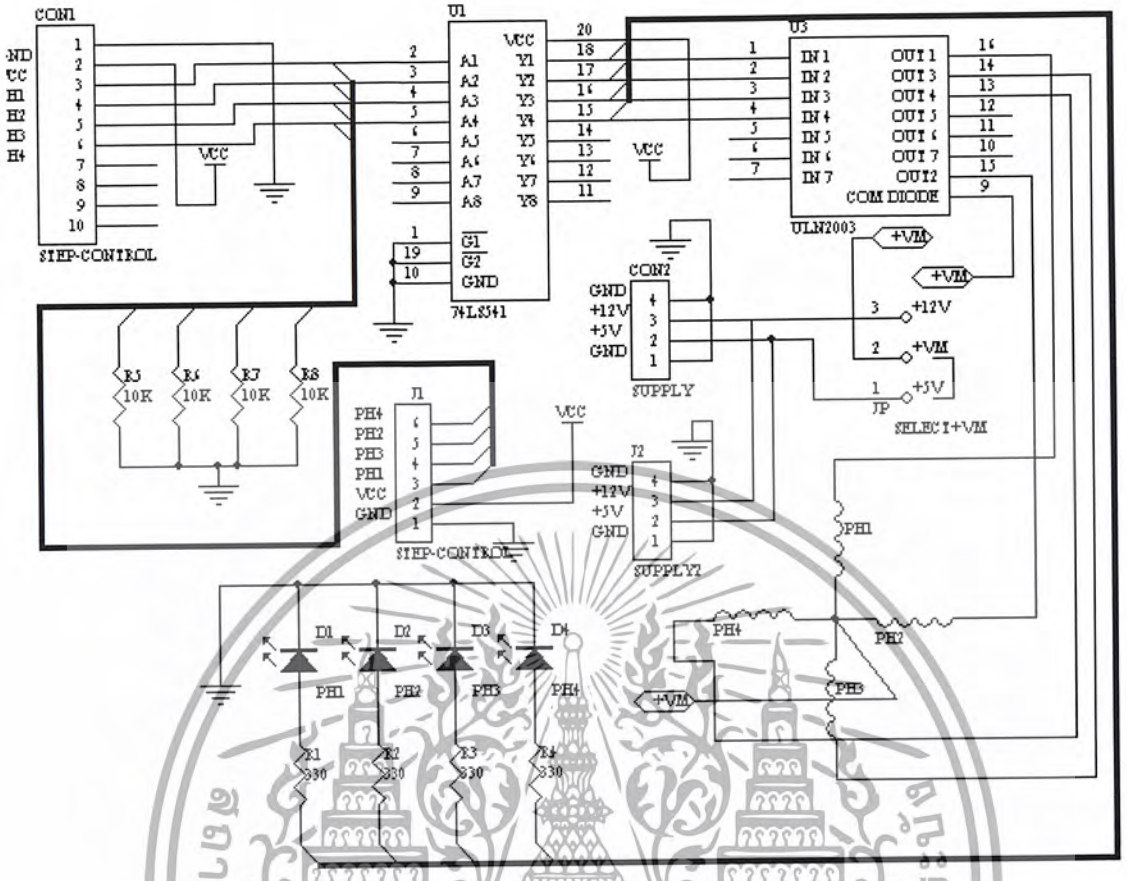
รูปที่ 3.21 แผนผังอุปกรณ์ของโมดูลสื่อสารข้อมูลอนุกรม



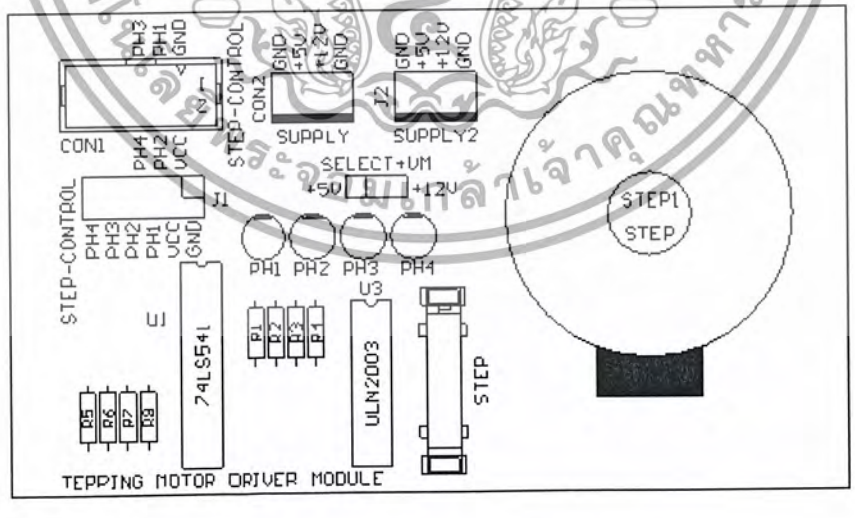
รูปที่ 3.22 โมดูลสื่อสารข้อมูลอนุกรม

3.3.7 โมดูลขับสเตปปิ้งมอเตอร์ (Stepping Motor Driver Module)

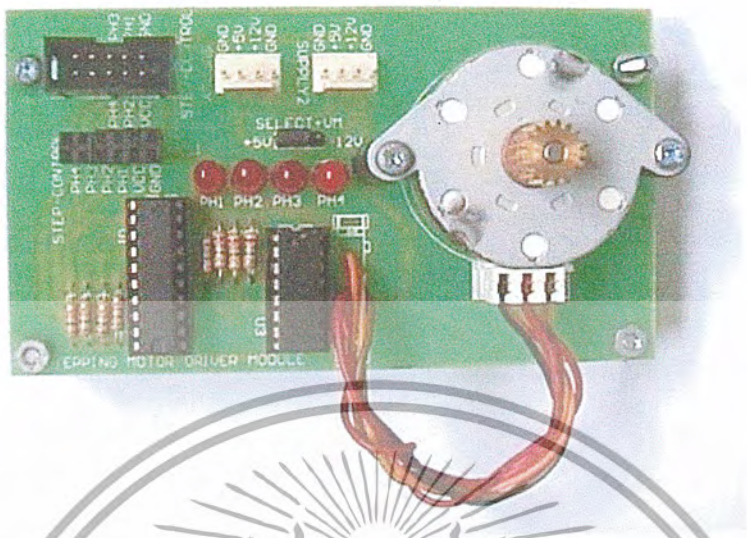
วงจรขับสเตปปิ้งมอเตอร์จะใช้ไอซี ULN 2003 ทำหน้าที่ขับกระแสให้แก่สเตปปิ้งมอเตอร์ โดยจะมีแอลอีดีทำหน้าที่แสดงสถานะการทำงานเป็นเฟสของสเตปปิ้งมอเตอร์ในแต่ละเฟสในส่วนของเพาเวอร์ซัพพลาย (Power Supply) ที่จ่ายให้สเตปปิ้งมอเตอร์จะที 2 ค่า คือ +12 V และ +5 V โดยสามารถเลือกได้จาก JP ส่วนคอนเน็คเตอร์ J1 จะใช้เชื่อมต่อร่วมกับพอร์ตของโมดูลหลักแบบเลือกบิตในการเชื่อมต่อได้



รูปที่ 3.23 วงจรขับสเตปปิ้งมอเตอร์



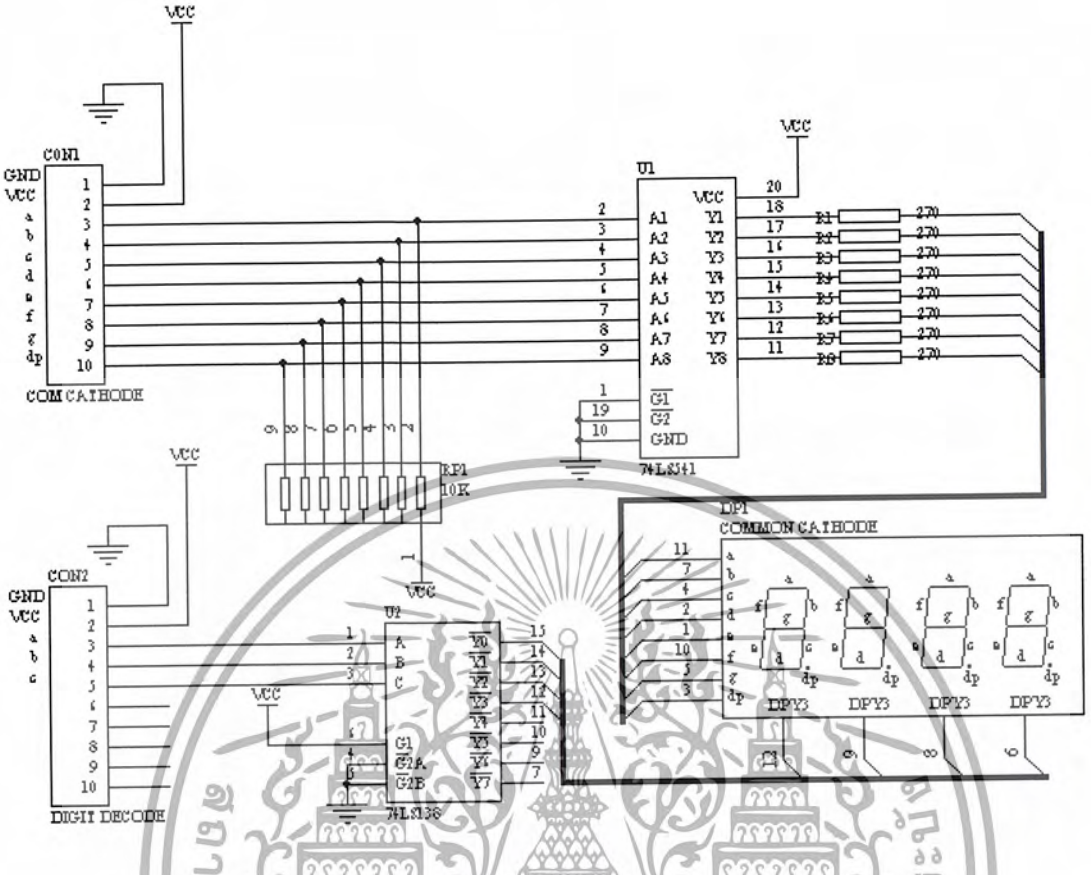
รูปที่ 3.24 แผนผังอุปกรณ์ของโมดูลขับสเตปปิ้งมอเตอร์



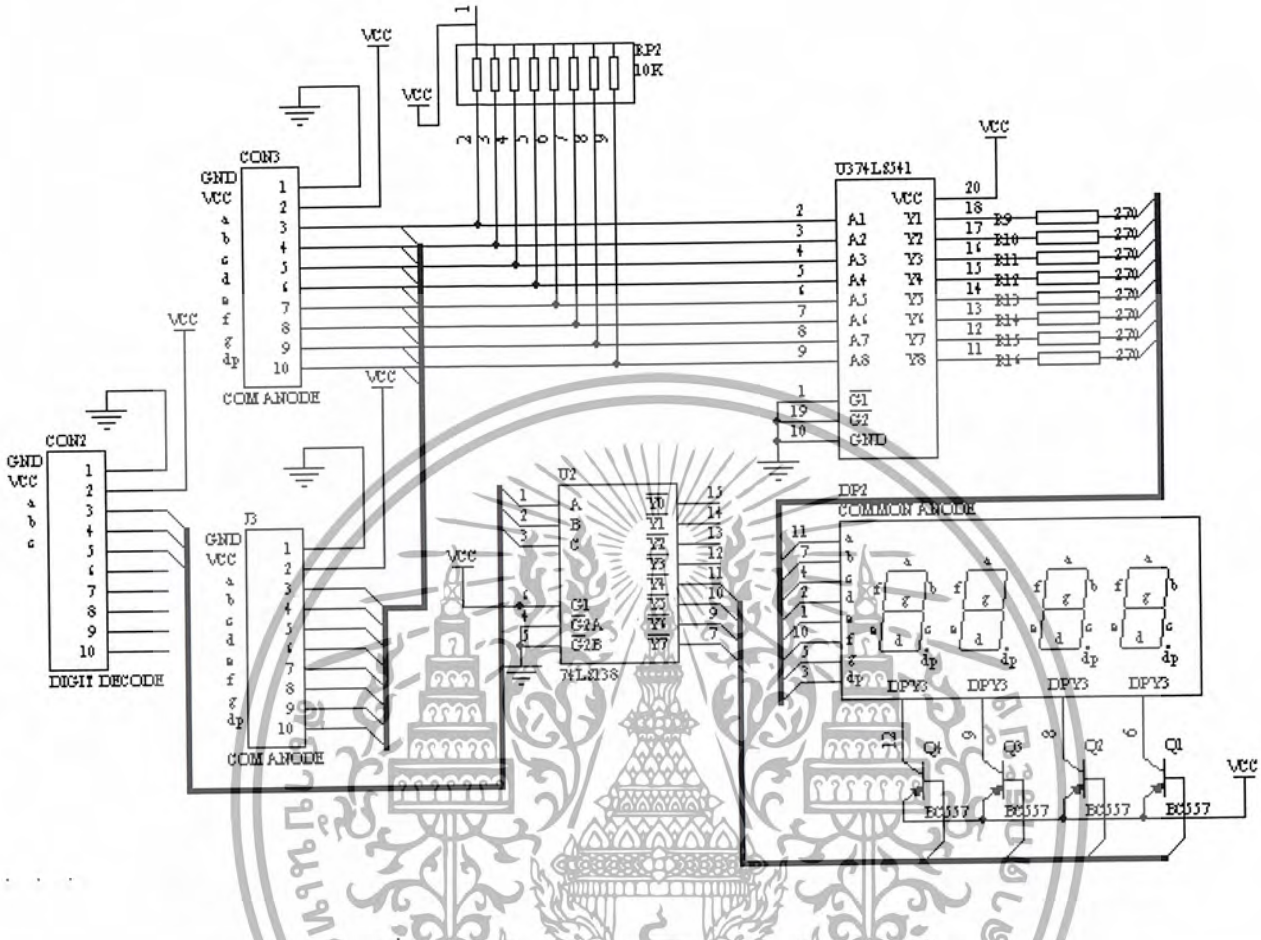
รูปที่ 3.25 โมดูลขับสเตปปิ้งมอเตอร์

3.3.7 โมดูลแสดงผลตัวเลขเจ็ดส่วน (Seven Segment Display Module)

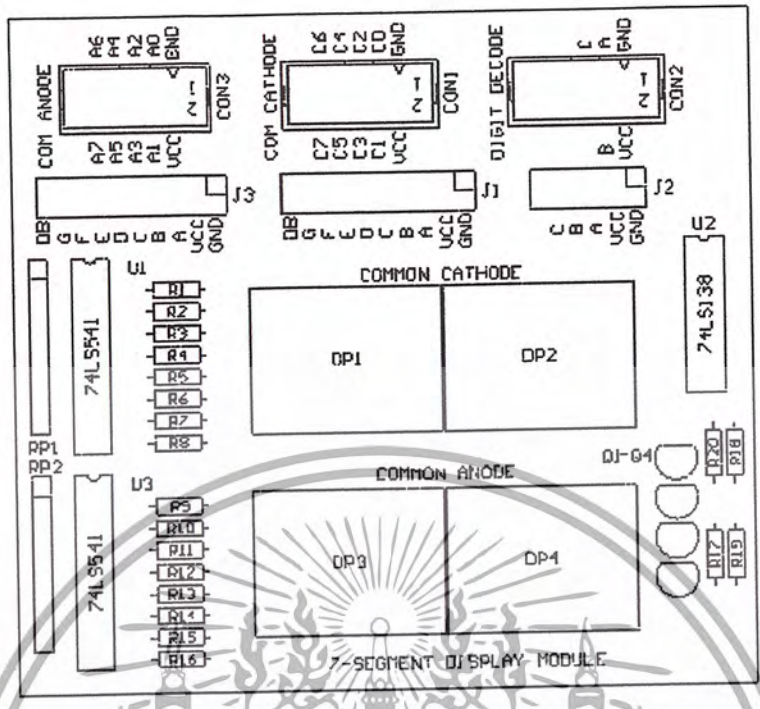
ในโมดูลแสดงผลตัวเลขเจ็ดส่วนจะแบ่งออกเป็น 2 ส่วนคือ ส่วนที่เป็นคอมมอนคาโถด และคอมมอนอาโนด โดยทั้งสองส่วนจะแสดงผลเป็นแบบ 4 หลัก และในการเลือกให้แต่ละหลักแสดงผลจะใช้ IC 74LS138 เป็นตัวถอดรหัส (Decode) เพื่อเลือกให้หลักต่างๆ ติด โดยที่ IC 74LS138 จะทำงาน ที่ลอจิกศูนย์ ในส่วนของคอมมอนอาโนดจะมีทรานซิสเตอร์ 4 ตัว คือ Q1-Q4 ทำหน้าที่กลับสภาวะลอจิกที่ป้อนให้แต่ละคอมมอนให้เป็น 1 ส่วน R17-R20 จะทำหน้าที่กำจัดกระแสให้กับทรานซิสเตอร์ ในการต่อใช้งานจะมีคอนเน็คเตอร์ Con1, Con3, J1 และ J3 เชื่อมต่อกับขาข้อมูลทั้ง 7 บิตของแอลอีดี คือ a, b, c, d, e, f, g ส่วนคอนเน็คเตอร์ J2 และ Con2 จะใช้ เชื่อมต่อขาแอลอีดีผ่าน IC 74LS138 เพื่อทำการเลือกให้คอมมอนติด



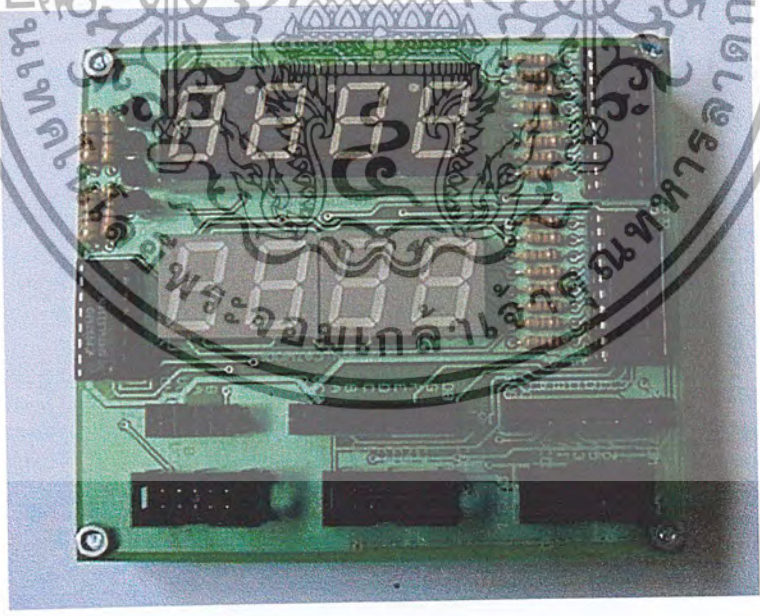
รูปที่ 3.26 วงจรแสดงผลตัวเลขเจ็ดส่วนคอมมอนคาโถด



รูปที่ 3.27 วงจรแสดงผลตัวเลขเจ็ดส่วนคอมมอนอาโนด



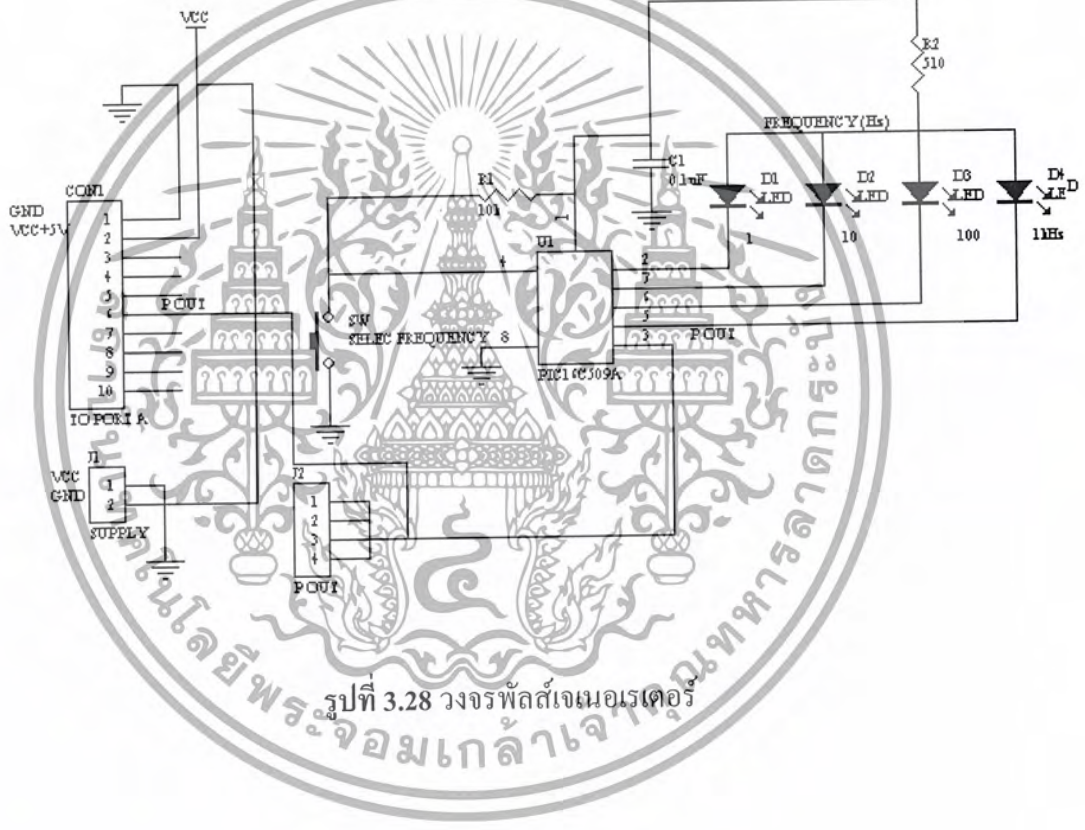
รูปที่ 3.28 แผนผังอุปกรณ์ของโมดูลแสดงผลตัวเลขเจ็ดส่วน



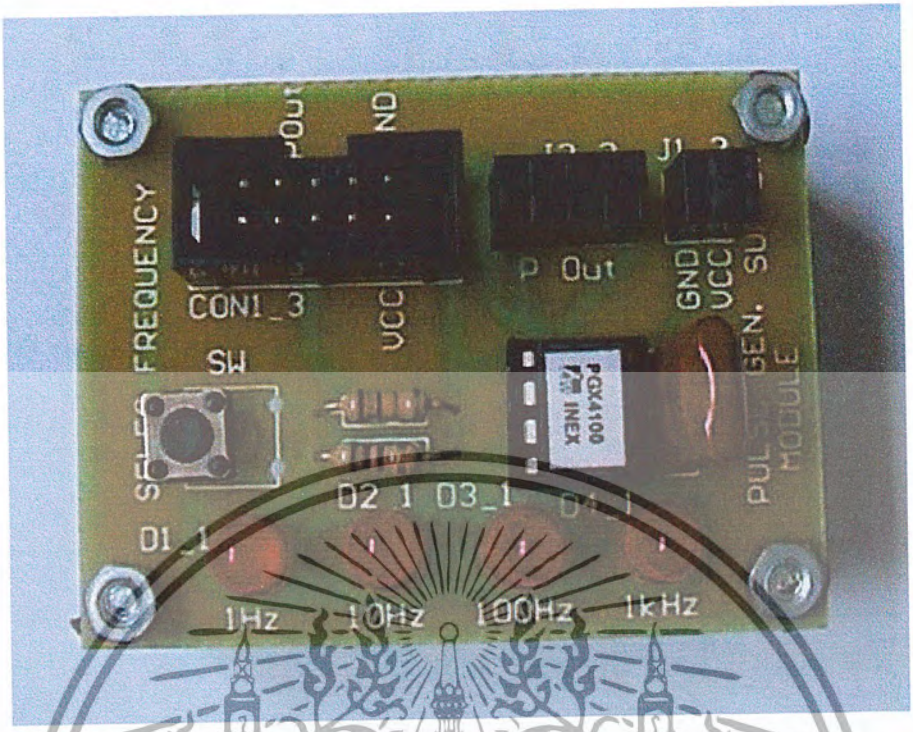
รูปที่ 3.29 โมดูลแสดงผลตัวเลขเจ็ดส่วน

3.3.8 โมดูลพัลส์เจเนอเรเตอร์ (Pulse Generator Module)

โมดูลพัลส์เจเนอเรเตอร์เป็นโมดูลที่สร้างสัญญาณพัลส์เพื่อใช้ในการทดลองไทเมอร์ เคาน์เตอร์หรืออื่นๆ ที่เกี่ยวข้องโดยได้ออกแบบให้มีค่าความถี่ออกมาเป็น 1 Hz, 10 Hz, 100 Hz, และ 1 KHz โดยใช้ไมโครคอนโทรลเลอร์ PIC16C509A เป็นตัวสร้างพัลส์สำหรับการเลือกค่าความถี่นั้น จะมีสวิตช์กดติดปล่อยดับใช้ในการเลือกค่าความถี่ระดับต่างๆ โดยการแสดงผลนั้นจะไปแสดงผลที่ แอลอีดีทั้ง 4 ดวงติดเป็นค่าความถี่ต่างๆ



รูปที่ 3.28 วงจรพัลส์เจเนอเรเตอร์



รูปที่ จ.29 โมดูลพัลส์เจเนอเรเตอร์

บทที่ 4

การทดลองและผลการทดลอง

4.1 กล่าวนำ

จากการได้ออกแบบทางด้านฮาร์ดแวร์ในบทที่ 3 ในบทนี้จะเป็นการทดลองของโมดูลต่างๆ ที่ได้ออกแบบไว้รวมไปถึงผลการทดลองที่เกิดขึ้นว่าเป็นตามที่ได้ออกแบบไว้หรือไม่หรือถูกต้องตามกระบวนการสร้างจริงหรือไม่ โดยในการทดลองจะทดลองแยกออกเป็นโมดูลต่างๆ ดังต่อไปนี้

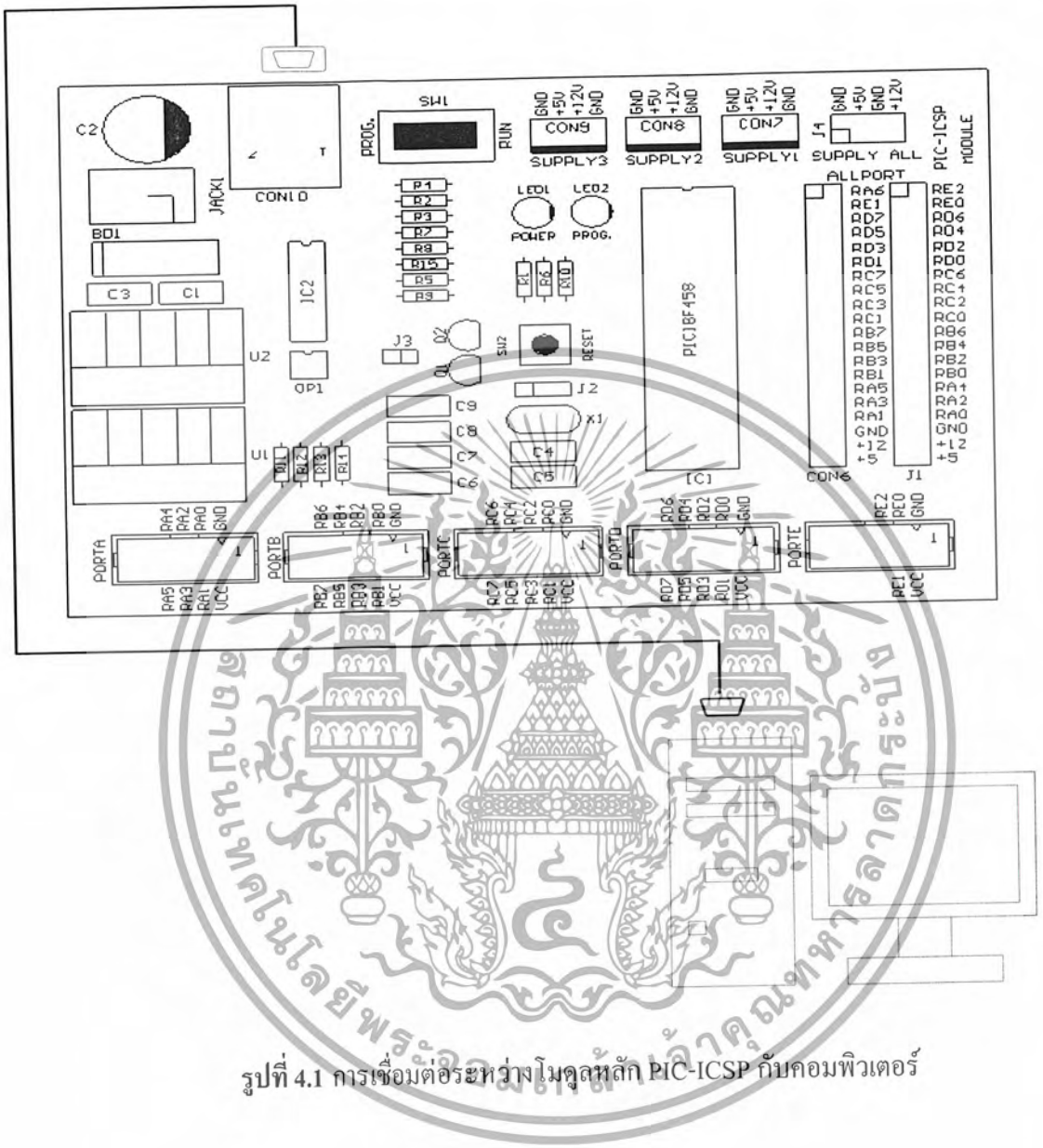
4.2 การทดลองและผลการทดลอง

4.2.1 การทดลองโมดูล PIC –ICSP

โมดูลนี้เป็นโมดูลที่มีความสำคัญมากที่สุด ถ้าผิดพลาดในการออกแบบและการสร้างก็จะไม่สามารถใช้งานร่วมกับโมดูลอื่นได้เลยหรืออาจจะทำให้เกิดการผิดพลาดในการสั่งงานได้ ในการทดลองโมดูลนี้จึงเป็นส่วนแรกที่สำคัญ ซึ่งมีวิธีการทดลองดังต่อไปนี้

ลำดับขั้นการทดลอง

1. เชื่อมต่อโมดูลหลักเข้ากับพอร์ตขนานของคอมพิวเตอร์ ดังรูปที่ 4.1
2. จ่ายไฟให้แก่โมดูลหลัก PIC-ICSP จากนั้นสังเกตแอลอีดีสีแดงที่แสดงสถานะการทำงานของโมดูลเมื่อมีไฟเลี้ยงให้แก่โมดูลก็จะทำให้แอลอีดีดวงนี้ติด
3. ทดสอบวัดแรงดันที่จุดต่างๆ ของโมดูลทดลองวัดแรงดันที่ ขา 11 และ 32 ของไมโครคอนโทรลเลอร์ ถ้าแรงดันมีค่าประมาณ 5 โวลต์ ในขณะที่สวิตช์ SW1 อยู่ที่ตำแหน่ง RUN แสดงว่าตอนนี้ไฟจ่ายให้กับซีพียูแล้ว และเมื่อเลื่อนสวิตช์มาทาง Program แรงดันที่ ขา 11 และ 32 จะเป็น 0 โวลต์ ตอนนี้สวิตช์จะอยู่ที่ตำแหน่งที่พร้อมจะรับแรงดันในการโปรแกรม
4. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมแล้วคอมไพล์ตรวจสอบความถูกต้องจากนั้นเปิดโปรแกรม Epic Win เปิดไฟล์เฮ็กซ์ (Hex) ที่คอมไพล์แล้วทำการโปรแกรมข้อมูลลงบนตัวไมโคร-คอนโทรลเลอร์



รูปที่ 4.1 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์

5. สังเกตผลที่เกิดขึ้นกับแอลอีดีสีแดงขณะโปรแกรมจะติดชั่วครู่หนึ่งและในขณะทำโปรแกรม ถ้าไม่มีการผิดพลาดของการโปรแกรม แอลอีดีก็จะดับลง จากนั้นรันโปรแกรมโดยเลื่อนสวิตช์ SW1 ไปที่ตำแหน่ง RUN ถ้าไม่มีการผิดพลาดเกิดขึ้นก็จะเป็นไปตามผลการทำงานของโปรแกรมที่เขียนขึ้น

ผลการทดลอง

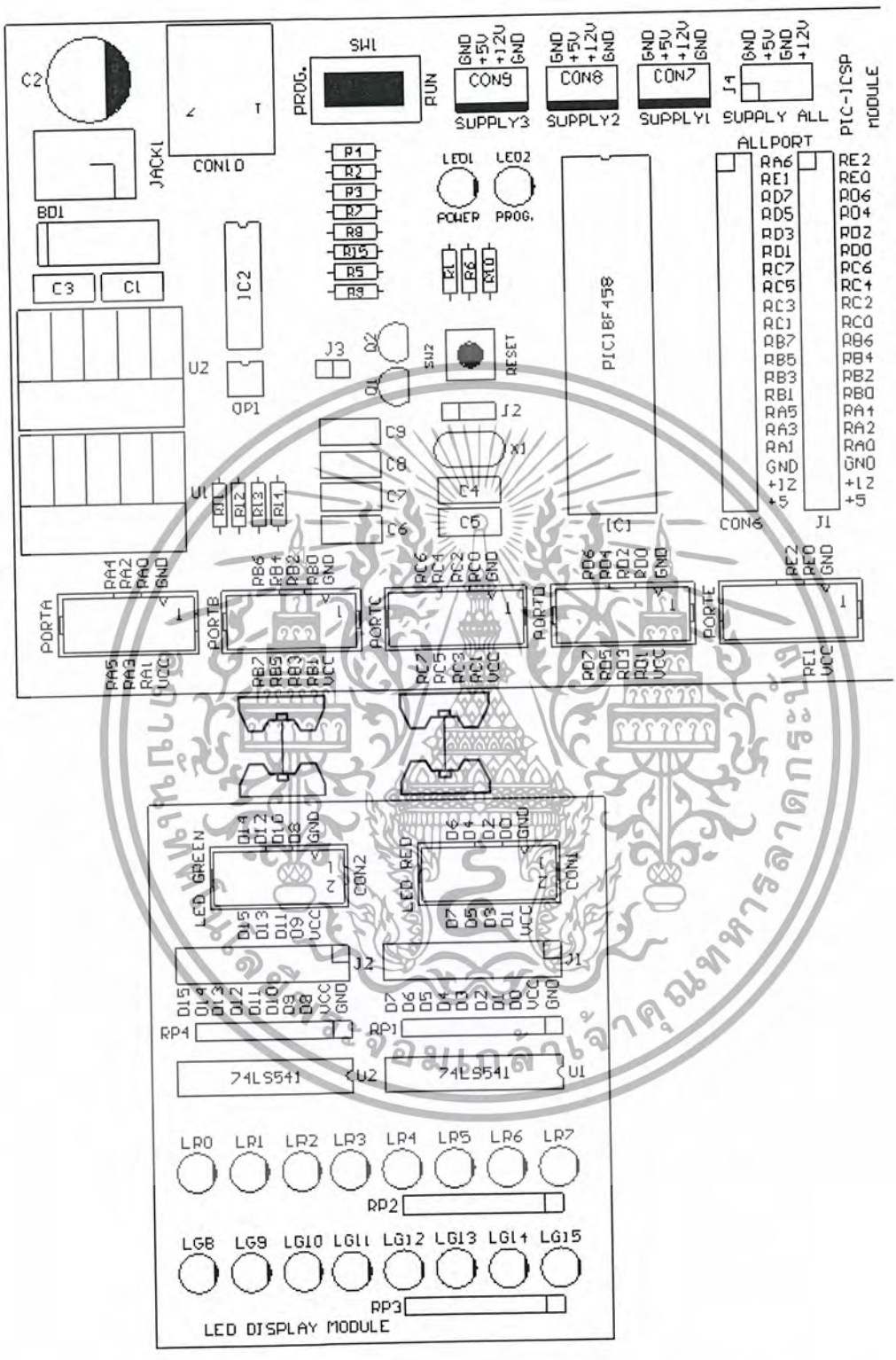
การทดลองเมื่อจ่ายไฟเลี้ยงให้โมดูลแล้วทำให้แอลอีดีสีแดงติดค้างและเมื่อวัดแรงดันที่จุดขา 11 และ 32 เป็นปกติ และทดลองดาวน์โหลดข้อมูลลงบนไมโครคอนโทรลเลอร์ สังเกตผลที่เกิดขึ้น แอลอีดีสีเหลืองติดกระพริบและไม่มีข้อความฟ้องการทำงานผิดพลาดแสดงว่าโมดูล PIC-ICSP ทำงานได้แล้ว

4.2.2 การทดลองใช้งานโมดูลแสดงผลแอลอีดี

ในการทดลองใช้งานโมดูลนี้จะตรวจสอบว่า แอลอีดีแสดงผลทั้ง 16 ดวงนั้นใช้งานได้หรือไม่ โดยเชื่อมต่อร่วมกับโมดูลหลักขับกระแสจากพอร์ตต่างๆ ของไมโครคอนโทรลเลอร์ไปแสดงผลไปยังแอลอีดีทั้ง 16 ดวง

ลำดับขั้นการทดลอง

1. เชื่อมต่อโมดูลหลัก PIC-ICSP เข้ากับโมดูลแสดงผลแอลอีดีตามรูปที่ 4.2
2. จากผังการทำงานที่ได้ออกแบบไว้ เขียนโปรแกรม ตามรูปที่ 4.3
3. รันโปรแกรม สังเกตผลที่เกิดขึ้นกับแอลอีดีทั้ง 16 ดวง ถ้าแอลอีดีทั้ง 16 ดวงติดทั้งหมด แสดงว่าโมดูล แสดงผลแอลซีดีใช้งานได้ตามที่ออกแบบไว้



รูปที่ 4.2 การเชื่อมระหว่างโมดูลหลัก PIC-ICSP กับโมดูลแสดงผลแอลอีดี

```
;********* TEST LED********
    LIST P=18F458
#include <P18F458.inc>
        ORG 0x0000
        CLRF    TRISC
        CLRF    TRISB
        CLRF    PORTC
        CLRF    PORTB
ST      SETF    PORTC
        SETF    PORTB
        GOTO    ST
        END
```

รูปที่ 4.3 โปรแกรมทดลองใช้งานโมดูลแสดงผลแอลอีดี

ผลการทดลอง

จากผลการทดลองเมื่อรันโปรแกรมปรากฏว่าแอลอีดีทั้ง 16 ดวงติดสว่างแสดงว่าโมดูลแสดงว่าแอลอีดีใช้งานได้ตามที่ได้ออกแบบไว้

4.2.3 การทดลองใช้งานโมดูลสวิตช์พื้นฐาน

ในการทดลองโมดูลนี้เป็นการทดสอบสวิตช์ทั้งหมด คือ สวิตซ์อินพุต ดิฟสวิตช์และเมตริกสวิตช์ซึ่งใช้ในการทดลองพื้นฐาน โดยการทดสอบสวิตช์อินพุตนั้นจะใช้โมดูลแสดงผลแอลอีดี แสดงสถานะการกดของสวิตช์เมื่อมีการกดสวิตช์ที่ตำแหน่งใดก็จะแสดงผลของการกดที่แอลอีดี

ลำดับขั้นการทดลอง

1. ต่อโมดูลแสดงผลแอลอีดีและสวิตช์อินพุตเข้ากับโมดูลหลักโดยเชื่อมต่อ พอร์ต B เข้ากับสวิตช์อินพุตและพอร์ต D ต่อเข้ากับแอลอีดีแสดงผล

2. จากผังการทำงานที่ได้ออกแบบไว้ เขียนโปรแกรมที่ใช้ในการทดลองการทำงานของโมดูลสวิตช์อินพุต ในรูปที่ 4.4

3. รันโปรแกรมสังเกตผลที่เกิดขึ้นกับแอลอีดีทั้ง 8 ดวง ทดลองกดสวิตซ์ S1-S7 สังเกตผลที่เกิดขึ้นกับโมดูลแสดงผลแอลอีดี

4. เปลี่ยนการเชื่อมต่อมายังตำแหน่งดิฟสวิตช์ทดลองกดสวิตช์ S0-S7 สังเกตผลการทดลองที่เกิดขึ้นกับโมดูลแสดงผลแอลอีดี

5. เปลี่ยนการเชื่อมต่อมายังตำแหน่งเมตริกซ์สวิตช์เขียนโปรแกรมตามรูปที่ 4.5

ผลการทดลอง

เมื่อกดสวิตช์ S0-S7 ในการทดลอง สวิตช์ อินพุต และดิฟสวิตช์ จะเห็นว่าแอลอีดีทั้ง 8 ดวง ดับตามตำแหน่งสวิตซ์ที่ถูกกด ในการทดลองเมตริกซ์สวิตช์ นั้น เมื่อกดสวิตช์ ที่ตำแหน่ง 0 ถึง F จะให้ผลแสดงที่แอลอีดีตามตำแหน่งที่ถูกกด และแสดงค่าที่ถูกต้อง แสดงว่าโมดูลสวิตช์ พื้นฐานใช้งานได้

```
;*************** Test sw ********************
 list p=18f458                    ; list directive to define processor
  #include <p18f458.inc>          ; processor specific variable definitions
   ORG 0x0000
   CLRF     TRISD                ; PORTD is output
   MOVLW    0xFF                 ; "11111111"
   MOVWF    TRISB                ; PORTB is inpu
LOOP MOVF   PORTB,W              ; W <-- (PORTB)
   MOVWF    PORTD                ; PORTD <-- (W)
   GOTO     LOOP
   END
```

รูปที่ 4.4 โปรแกรมทดลองโมดูลสวิตช์ พื้นฐาน

4.2.4 การทดลองใช้งานโมดูลขับสเตปปิ้งมอเตอร์

วงจรขับสเตปปิ้งมอเตอร์แบบยูนิโพลาร์ จะทดลองด้วยการควบคุมการหมุนแบบเวฟเพื่อทดลองการทำงานของโมดูล

ลำดับขั้นการทดลอง

1. เชื่อมต่อโมดูลหลัก PIC-ICSP เข้ากับโมดูลขับสเตปปิ้งมอเตอร์เพื่อทดลองการทำงานโดยใช้พอร์ต D ในการเชื่อมต่อ

2. จากผังการทำงานที่ได้ออกแบบไว้ เขียนโปรแกรมควบคุมสเตปปิ้งมอเตอร์แบบเวฟตาม โปรแกรมรูปที่ 4.4

3. โปรแกรมลงบนไมโครคอนโทรลเลอร์

```
;*****************TEST STEP <1 เฟส **********************
            list p=18f458          ; list directive to define processor
            #include <p18f458.inc>    ; processor specific variable definitions

 OFFSET   EQU    0x20
 COUNT    EQU    0x21
 COUNT1   EQU    0x22
 COUNT2   EQU    0x23

                ORG    0x0000
        CLRF    TRISD           ; PORTD is output
ST      CLRF    OFFSET          ; OFFSET = 0
        MOVLW   .4
        MOVWF   COUNT           ; COUNT = 4
START   MOVF    OFFSET,W        ; W <-- (OFFSET)
        CALL    TAB             ; Read data form Table
        MOVWF   PORTD           ; Send to PORTD
        CALL    DELAY           ; delay
        INCF    OFFSET,F
        INCF    OFFSET,F        ; Offset = Offset + 2
        DECFSZ  COUNT,F         ; COUNT = COUNT-1 and skip if COUNT =0
        GOTO    START           ; != 0
        GOTO    ST
;******** Delay loop ********
DELAY   MOVLW   0x00
        MOVWF   COUNT1
DEL0    CLRF    COUNT2
DEL1    DECFSZ  COUNT2,F
        GOTO    DEL1
        DECFSZ  COUNT1,F
        GOTO    DEL0
```

```
        RETURN
;********* Tebel of data **********************
TAB   ADDWF   PCL                    ; Move offset to PC lower
 DT     0x01,0x02,0x04,0x08          ; 0001 , 0010 , 0100 , 1000

        END
```

รูปที่ 4.6 โปรแกรมควบคุมสเตปปิ้งมอเตอร์แบบเวฟ

ผลการทดลอง

เมื่อรันผลการทำงานจะเห็นสเตปปิ้งมอเตอร์หมุนทีละสเตปสังเกตการหมุนของสเตปปิ้งมอเตอร์ได้จากแอลอีดีแสดงผลทั้ง 4 ดวง ถ้าแอลอีดีติดเป็นไฟวิ่งและมอเตอร์แสดงว่าโมดูลขับสเตปปิ้งมอเตอร์ใช้งานได้แล้ว

4.2.5 การทดลองใช้งานโมดูลพัลส์เจเนอเรเตอร์

โมดูลพัลส์เจเนอเรเตอร์จะใช้งานในการส่งพัลส์ให้โมดูลต่างๆ ภายในไมโครคอนโทรลเลอร์ ซึ่งในการตรวจสอบการทำงานนั้นจะต้องตรวจสอบว่าโมดูล สามารถสร้างพัลส์ออกมาได้จริงหรือไม่ โดยมีวิธีการดังต่อไปนี้

ลำดับขั้นการทดลอง

1. ต่อวงจรตามรูปที่ 4.6
2. จะสังเกตเห็นแอลอีดีที่โมดูลพัลส์เจเนอเรเตอร์ติดจากนั้นทดสองกดสวิตช์ S1
3. จะสังเกตเห็นแอลอีดีเปลี่ยนแปลงตามสภาวะของการกดการกด
4. ทดลองใช้มัลติมิเตอร์ (กรณีที่ไม่มีออสซิลโลสโคป) วัดค่าที่จุดคอนเน็ตเตอร์ J2 ซึ่งเป็นจุดเชื่อมต่อของสัญญาณพัลส์ทางเอาต์พุตสังเกตเข็มของมิเตอร์จะกระดิกขึ้นลง
5. ใช้ออสซิลโลสโคปวัดค่าตามการทดลองข้อ 4 และลองเปลี่ยนแปลงค่าความถี่ของโมดูลโดยการกดสวิตช์ S1

ผลการทดลอง

เมื่อใช้มัลติมิเตอร์วัดค่าสัญญาณพัลส์ที่จุดคอนเน็ตเตอร์ J2 และลองเปลี่ยนย่านวัดไปยังย่านต่างๆ และกดสวิตช์เปลี่ยนค่าความถี่ปรากฏว่าเข็มของมิเตอร์กระดิกเร็วและช้าตามจังหวะของสัญญาณพัลส์ค่านั้นๆ แสดงว่าโมดูลพัลส์เจเนอเรเตอร์ใช้งานได้แล้ว

4.2.6 การทดลองใช้งานโมดูลแสดงผลแอลซีดี

การทดลองนี้เป็นการทดลองใช้งานแอลซีดี 16 ตัวอักษร 2 บรรทัดโดยการแสดงผลนั้นมีใช้วิธีการเชื่อมต่อในแบบ 4 บิต ซึ่งเป็นวิธีการที่ถือว่าลดการใช้สายสัญญาณในการเชื่อมต่อในการทดลองนั้นมีวิธีการดังต่อไปนี้

ลำดับขั้นการทดลอง

1. ต่อวงจรตามรูปที่ 4.8
2. เขียนโปรแกรมตามรูปที่ 4.7
3. ทดลองรันโปรแกรมสังเกตผลที่เกิดขึ้นกับโมดูลแสดงผลแอลซีดี
4. ปรับค่าความต้านทาน 10 กิโลโอห์ม เพื่อปรับความสว่างของจอแสดงผลถ้าโมดูลแสดงผลแอลซีดีใช้งานได้จะต้องปรากฏข้อความ PIC18F458

```
list p=18f458
#include <p18f458.inc>
#DEFINE   RS      PORTC,0
#DEFINE   E       PORTC,1
COM       EQU     0x20
DAT       EQU     0x21
COUNT1    EQU     0x22
COUNT2    EQU     0x23
COUNT3    EQU     0x24

      ORG     0x0000
;************ initial ********************
     CLRF     TRISC
      CLRF     TRISD
      CALL     DELAY
      MOVLW    B'00110011'
      CALL     WR_INS
      MOVLW    B'00110010'
      CALL     WR_INS
      MOVLW    B'00101000'
      CALL     WR_INS
      MOVLW    B'00001100'
      CALL     WR_INS
      MOVLW    B'00000110'
      CALL     WR_INS
      MOVLW    B'00000001'
      CALL     WR_INS
;********** Data to display LCD ********
      MOVLW    "P"
      CALL     WR_DATA
      MOVLW    "I"
      CALL     WR_DATA
      MOVLW    "C"
      CALL     WR_DATA
      MOVLW    "1"
      CALL     WR_DATA
      MOVLW    "8"
```

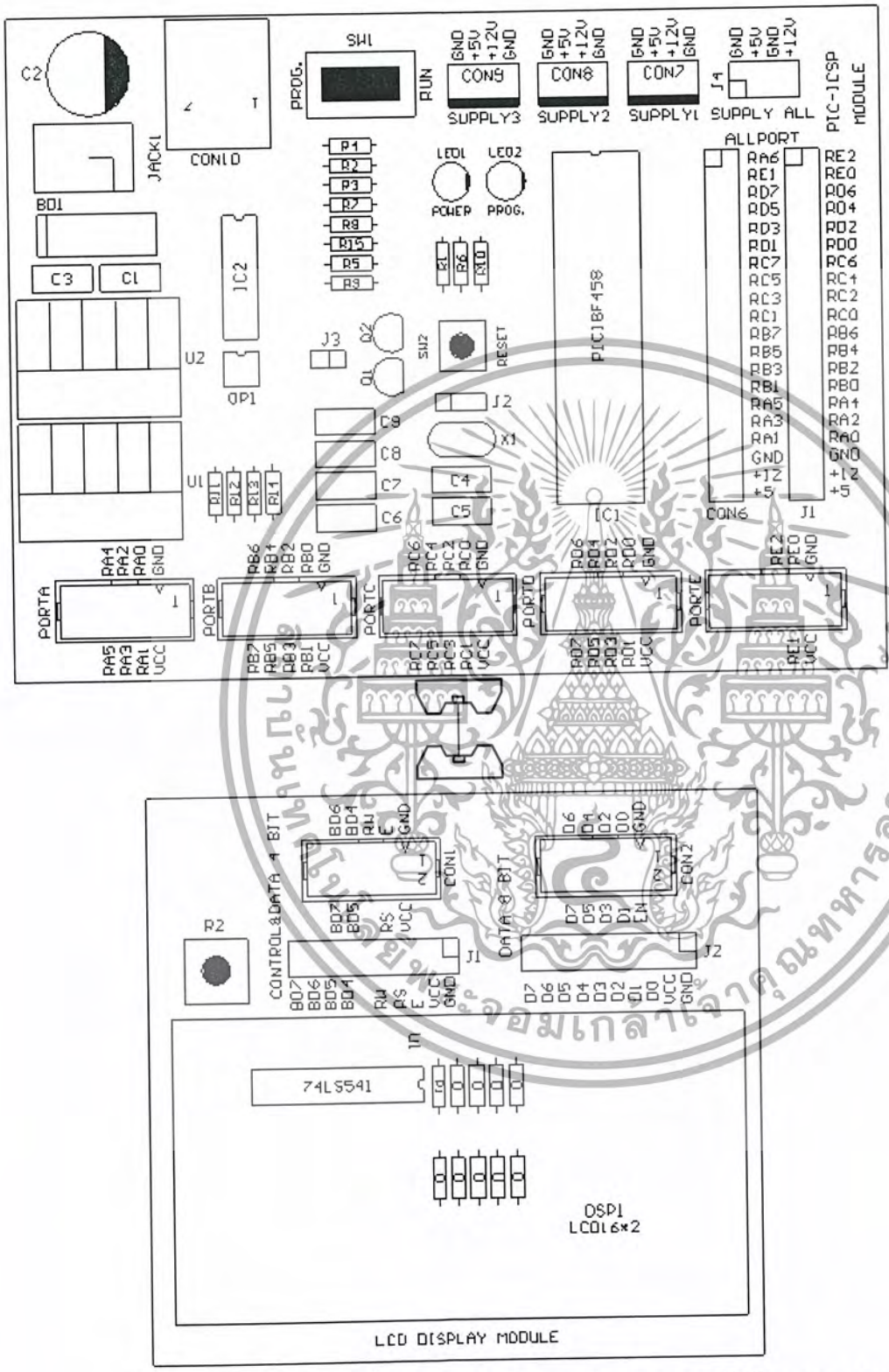
```
CALL    WR_DATA
        MOVLW   "F"
        CALL    WR_DATA
        MOVLW   "4"
        CALL    WR_DATA
        MOVLW   "5"
        CALL    WR_DATA
        MOVLW   "8"
        CALL    WR_DATA
        GOTO    $
;********* Write command to LCD *********
WR_INS  BCF     RS
        BSF     E
        MOVWF   COM
        ANDLW   0xF0
        MOVWF   PORTD
        CALL    PULSE
        SWAPF   COM,W
        ANDLW   0xF0
        MOVWF   PORTD
        CALL    PULSE
        RETURN

;********** Write data to LCD **********

WR_DATA BSF     RS
        BSF     E
        MOVWF   DAT
        ANDLW   0xF0
        MOVWF   PORTD
        CALL    PULSE
        SWAPF   DAT,W
        ANDLW   0xF0
        MOVWF   PORTD
        CALL    PULSE
        RETURN
```

```
;********* Generate pulse *******
            BCF         E
            CALL        DELAY
            BSF         E
            RETURN
;****** Delay time *********
DELAY       MOVLW       .50
            MOVWF       COUNT1
DEL1        CLRF        COUNT2
DEL2        DECFSZ      COUNT2
            GOTO        DEL2
            DECFSZ      COUNT1
            GOTO        DEL1
            RETURN
            E ND
```

รูปที่ 4.7 โปรแกรมการทดลองใช้งานโมดูลแสดงผลแอลซีดี

```
;********* Generate pulse *******
            BCF         E
            CALL        DELAY
            BSF         E
            RETURN
;****** Delay time *********
DELAY       MOVLW       .50
            MOVWF       COUNT1
DEL1        CLRF        COUNT2
DEL2        DECFSZ      COUNT2
            GOTO        DEL2
            DECFSZ      COUNT1
            GOTO        DEL1
            RETURN
            E ND
```

รูปที่ 4.7 โปรแกรมการทดลองใช้งานโมดูลแสดงผลแอลซีดี



รูปที่ 4.8 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับโมดูลแสดงผลแอลซีดี

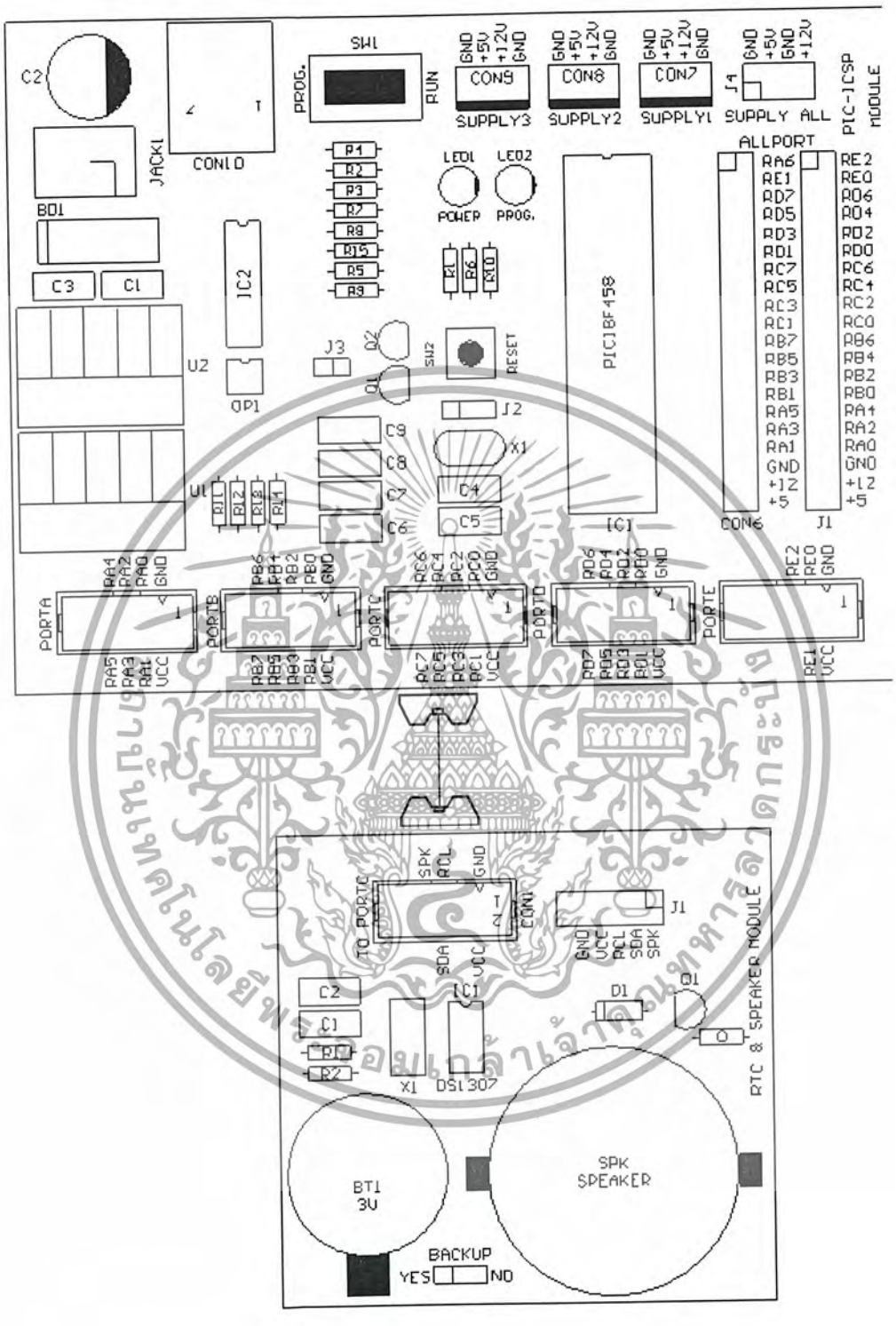
ผลการทดลอง

หลังจากรันโปรแกรมและทดลองปรับค่าความความสว่างของจอแสดงผลจอแสดงผลจะปรากฏข้อความ PIC18F458 บนจอแสดงผลแสดงว่าโมดูลแสดงผลแอลซีดีใช้งานได้แล้ว

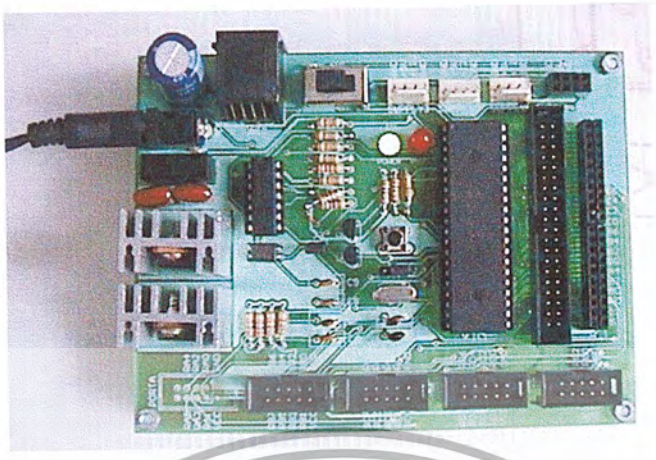
4.2.7 การทดลองใช้งานโมดูลเชื่อมต่ออุปกรณ์ SPEAKER/RTC

การทดสอบโมดูลนี้แบ่งออกเป็น 2 ส่วนคือส่วนที่เชื่อมต่ออุปกรณ์ SPEAKER และ RTC การทดสอบ SPEAKER นั้นจะส่งความถี่ให้แก่ลำโพงเพื่อให้ลำโพงเปล่งเสียงออกมาส่วน RTC จะเป็นการอ่านค่าเวลาจากตัว DS1307 มีวิธีการทดลองดังต่อไปนี้

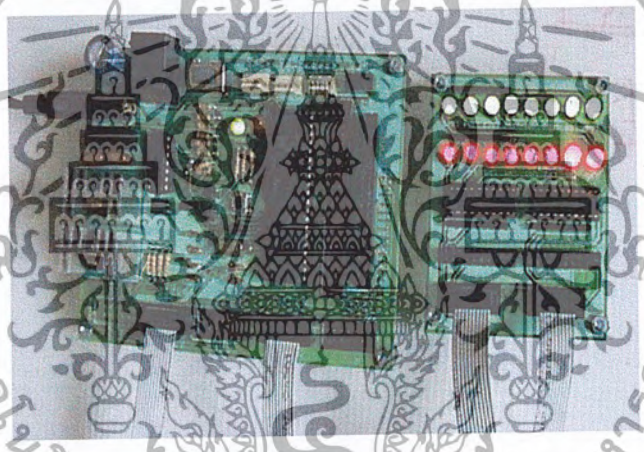
1. ต่อวงจรตามรูปที่ 4.9 เพื่อเชื่อมต่อทดลองใช้งานทดสอบลำโพง
2. เขียนโปรแกรมตามรูปที่ 4.10
3. ทดลองเปลี่ยนค่าของโปรแกรมหน่วงเวลาดูและโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้งลำโพงจะให้ระดับเสียงที่ระดับต่างกันหลังรันโปรแกรม
4. ทดลองเปลี่ยนค่าแรงดันที่ป้อนให้กับโมดูลโดยเลือกเชื่อมต่อที่คอนเน็คเตอร์ All Port โดยเลือกแรงดันที่ 12 โวลต์ต่อเข้ากับลำโพงผ่านคอนเน็คเตอร์ J1 ถ้าลำโพงสามารถเปล่งเสียงออกมาดังยิ่งขึ้นกว่าเดิมและเสียงไม่แสดงว่าลำโพงสามารถทำงานที่ค่าแรงดัน 12 โวลต์ได้แสดงว่าลำโพงยังใช้งานได้



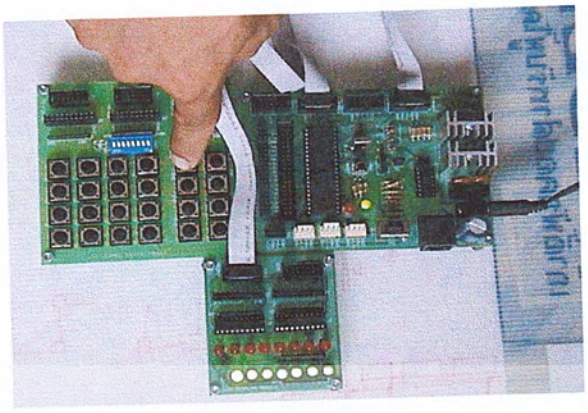
รูปที่ 4.9 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP เพื่อทดสอบลำโพง



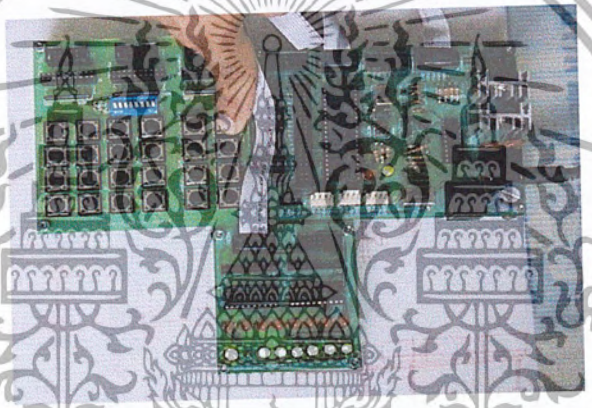
รูปที่ 4.10 ผลการทดลองโมดูลหลัก PIC-ICSP



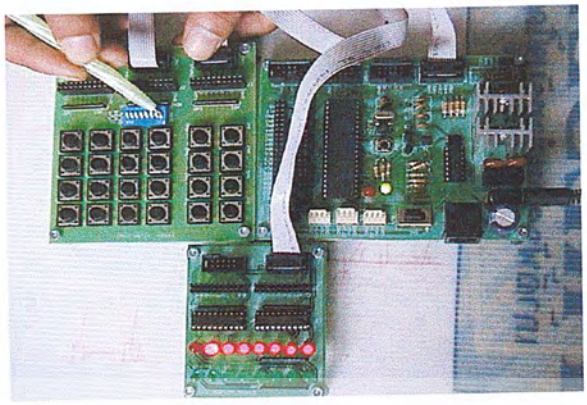
รูปที่ 4.11 ผลการทดลองโมดูลแสดงผลแอลอีดี



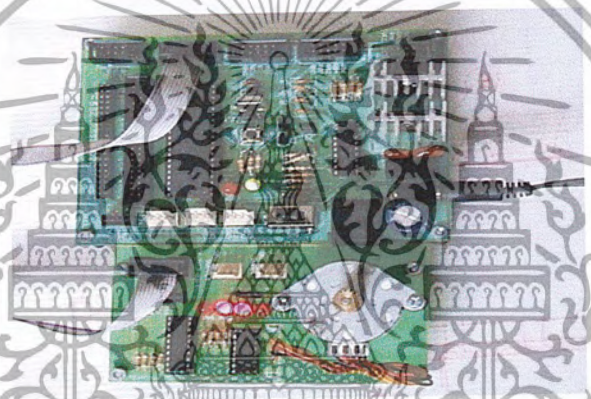
รูปที่ 4.12 ผลการทดลองสวิตช์อินพุต



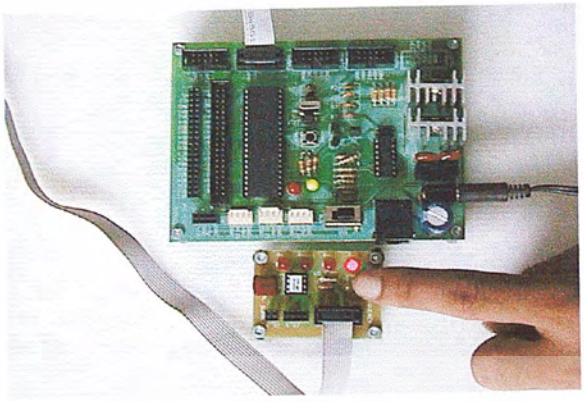
รูปที่ 4.13 (ต่อ) การทดลองสวิตช์อินพุต



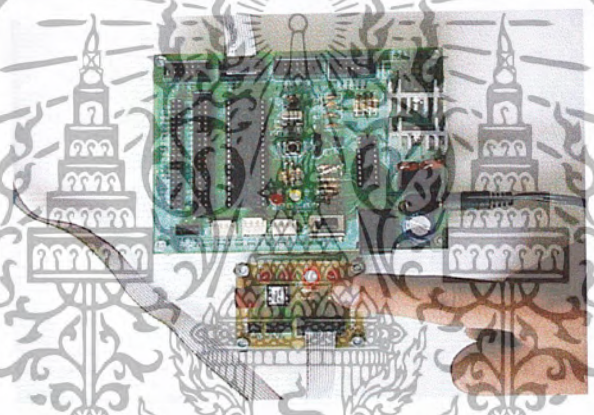
รูปที่ 4.14 ผลการทดลองดิฟสวิตช์



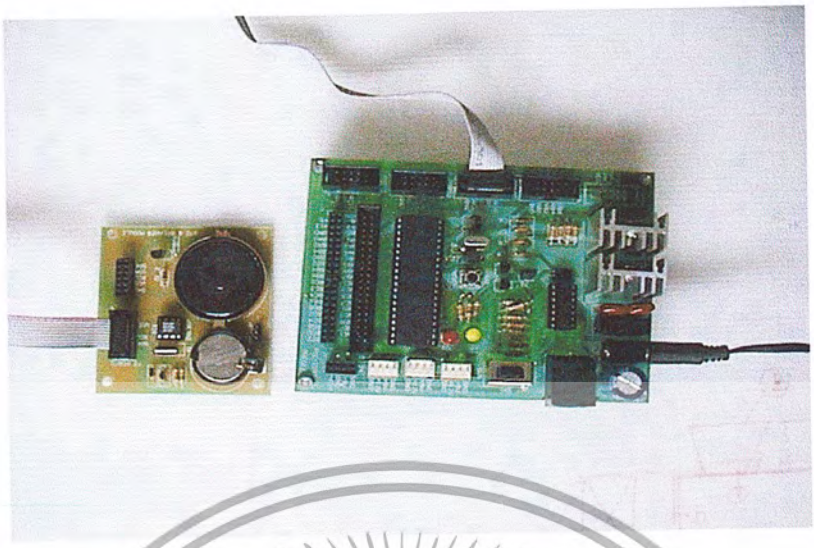
รูปที่ 4.15 ผลการทดลองสเต็ปปิ้งมอเตอร์



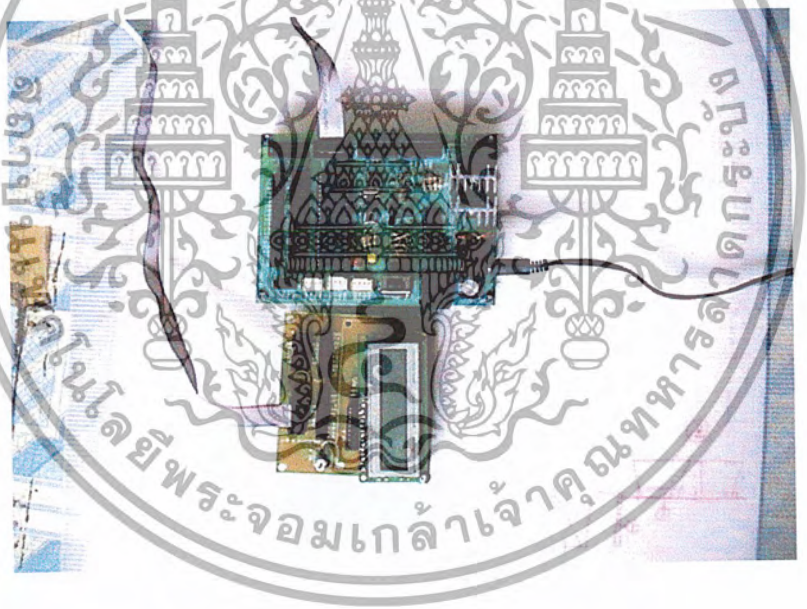
รูปที่ 4.16 ผลการทดลองพัลส์เจเนอเรเตอร์



รูปที่ 4.17 ผลการทดลองพัลส์เจเนอเรเตอร์



รูปที่ 4.18 ผลการทดลองโมดูลเชื่อมต่อ RTC/SPEAKER



รูปที่ 4.19 ผลการทดลองโมดูลแสดงผลแอลซีดี

บทที่ 5
บทสรุป

5.1 สรุป

ชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 สร้างขึ้นเพื่อศึกษาทฤษฎีและหลักการตลอดจนการนำไปใช้งาน ของชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 ชุดทดลองนี้แบ่งออกเป็น 3 ส่วน คือ ส่วนที่หนึ่ง เป็นส่วนของชุดอุปกรณ์ทดลองต่างๆ โดยแบ่งออกเป็น 9 โมดูล คือ โมดูลหลัก PIC-ICSP โมดูลสวิตช์พื้นฐาน โมดูลแสดงผลแอลอีดี โมดูลแสดงผลตัวเลขเจ็ดส่วน โมดูลแสดงผลแอลซีดี โมดูลสื่อสารข้อมูลอนุกรม โมดูลเชื่อมต่ออุปกรณ์ RTC/SPEAKER โมดูลขับสเตปปิ้งมอเตอร์ ส่วนที่สอง เป็นส่วนของใบงานที่ใช้ประกอบการทดลอง และส่วนที่สาม เป็นส่วนของคู่มือการใช้งานโปรแกรมการทดลอง คือ คู่มือการใช้งานชุดทดลอง คู่มือการใช้งานโปรแกรม Epic Win และคู่มือการใช้งานโปรแกรม MPLAB ชุดทดลองนี้จึงช่วยให้เข้าใจหลักการทำงานของไมโครคอนโทรลเลอร์ตระกูล PIC และ PIC18F458 ได้ง่ายยิ่งขึ้น และสามารถประยุกต์ใช้งานในระบบควบคุมทางอุตสาหกรรมด้านต่างๆ ได้ นอกจากนี้ยังสามารถนำชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458 ไปใช้เป็นสื่อการเรียนการสอนสำหรับนักเรียนนักศึกษาและบุคคลทั่วไปที่สนใจได้

5.2 ปัญหาและแนวทางแก้ไข

1. ปัญหา การศึกษาด้านไมโครคอนโทรลเลอร์ PIC18F458 ล่าช้าเพราะข้อมูลส่วนใหญ่นั้นเป็นภาษาต่างประเทศ ทำให้ทำความเข้าใจได้ยาก

แนวทางแก้ไข แปลและทำความเข้าใจทดลองเขียนโปรแกรม ศึกษาเปรียบเทียบกับไมโครคอนโทรลเลอร์ PIC เบอร์อื่นๆ เช่น PIC16F87X

2. ปัญหา การออกแบบทางด้านฮาร์ดแวร์ในโมดูลต่างๆ ให้เส้นเดินลายวงจรเล็กเกินไป ทำให้เวลาบัดกรี ลายวงจรขาดร่อนได้ง่าย

แนวทางแก้ไข ออกแบบให้ลายวงจร และจุดบัดกรีใหญ่ขึ้น

3. ปัญหา โมดูลหลัก PIC18F458 ในจุดเชื่อมต่อพอร์ต All Port นั้นมีเนื้อที่แคบเกินไป และชื่อที่แสดงรายละเอียดพอร์ตไปทับกันคอนเน็คเตอร์ ทำให้เห็นไม่ชัดเจน

แนวทางแก้ไข ควรออกแบบให้พอร์ตมีขนาดพื้นที่กว้างมากขึ้น และจัดวางตำแหน่งของคอนเน็คเตอร์ขยับออกมาอีก

5.3 แนวทางการพัฒนา

1. โมดูลทางฮาร์ดแวร์นั้นมีขนาดความยาวความกว้างที่ไม่เท่ากัน บางบอร์ดอาจเล็กจนเกินไป ควรออกแบบให้โมดูลมีขนาดเท่ากันเป็นดีที่สุด

2. ควรเพิ่มโมดูลให้มากขึ้นกว่านี้ เช่น โมดูลแสดงผลเมตริกซ์แอลอีดี โมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอลและดิจิตอลเป็นแอนาลอก โมดูลทดลองทางวงจรที่ใช้ทดลองภายนอกเพิ่มเติม

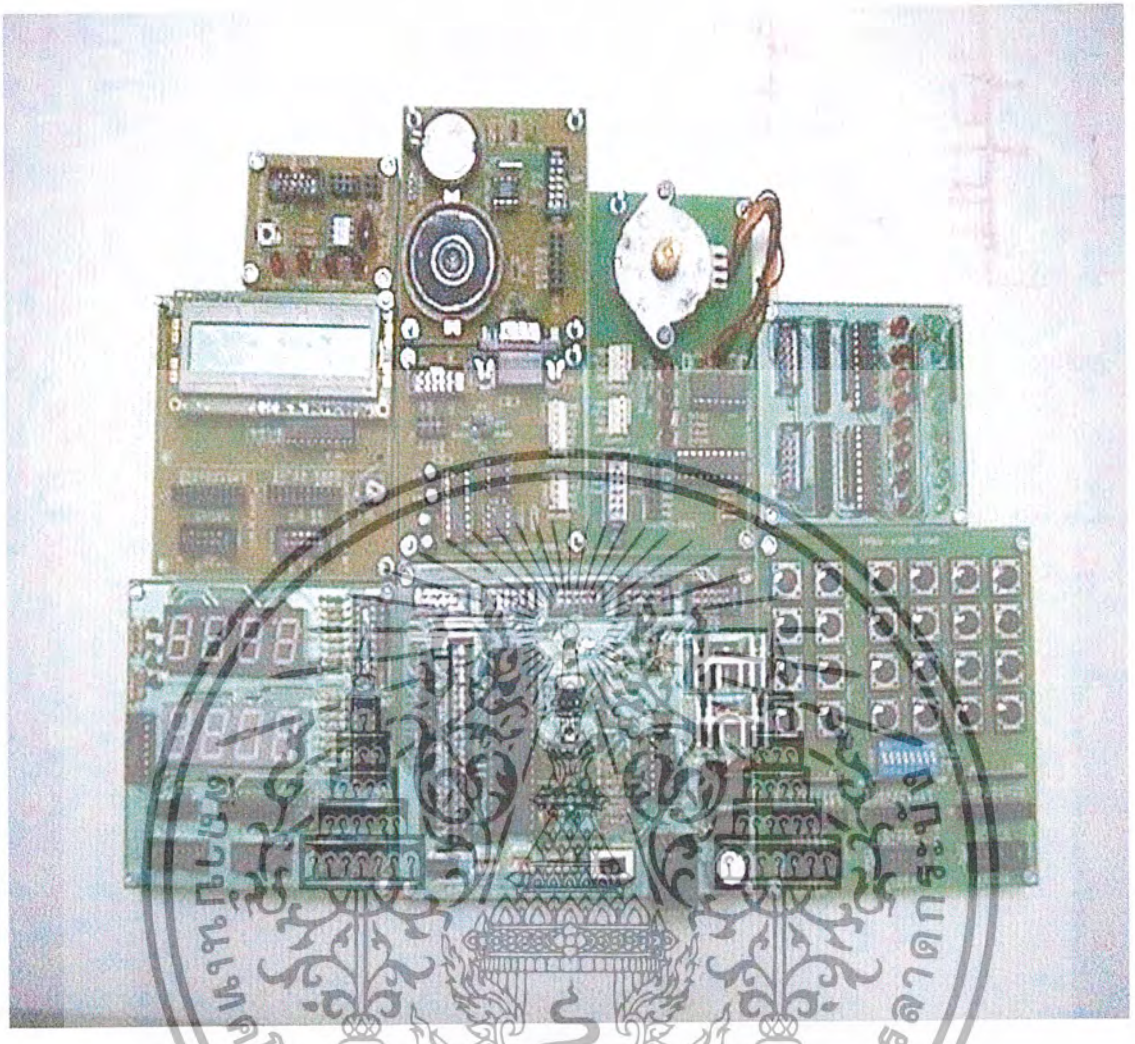
3. พัฒนาภาษาที่ใช้เขียนโปรแกรมทดลองเช่น ภาษาซี, เบสิก เพราะสามารถพัฒนาและเรียนรู้ได้เร็ว

บรรณานุกรม

กฤษฎา ใจเย็น และคณะ. **เรียนรู้ไมโครคอนโทรลเลอร์อย่างง่ายกับเบสิกแสแตมป์ 2**. กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอร์ริเมนต์. ม.ป.ป.

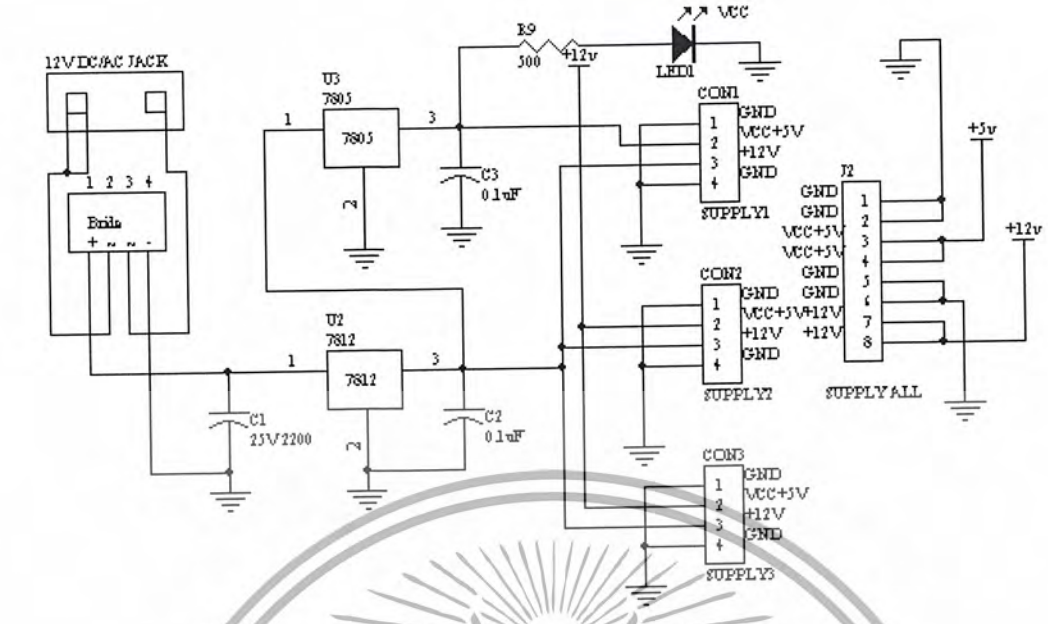
กฤษฎา ใจเย็น และชัยวัฒน์ ลิ้มพรจิตรวิไล. **เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ PIC16F84**. กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอร์ริเมนต์. ม.ป.ป.

ภาคผนวก ก
เครื่องต้นแบบ

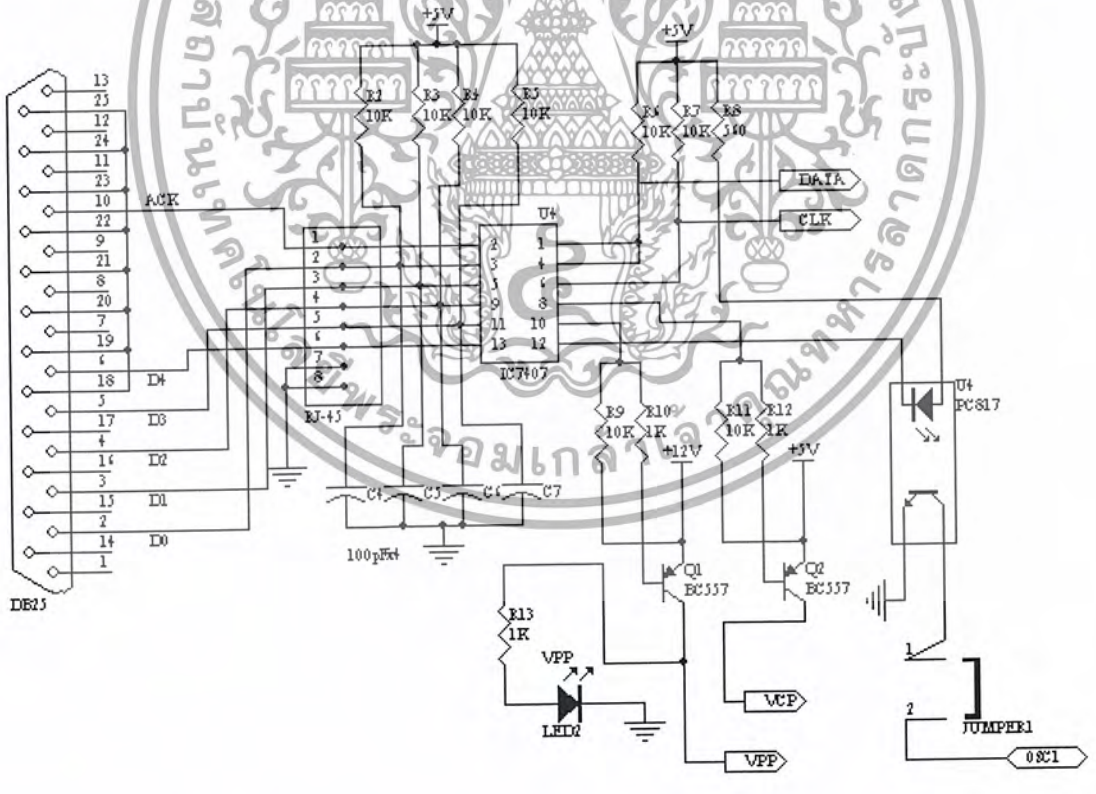


รูปที่ ก.1 เครื่องต้นแบบ

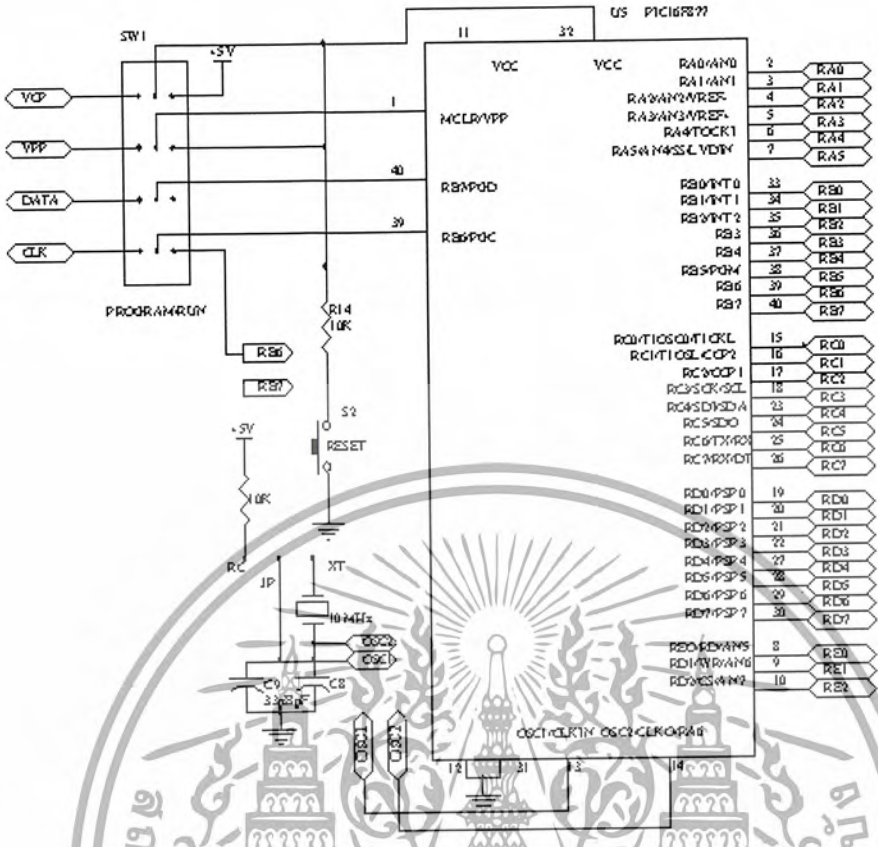
ภาคผนวก ข
วงจรและแผ่นวงจรพิมพ์



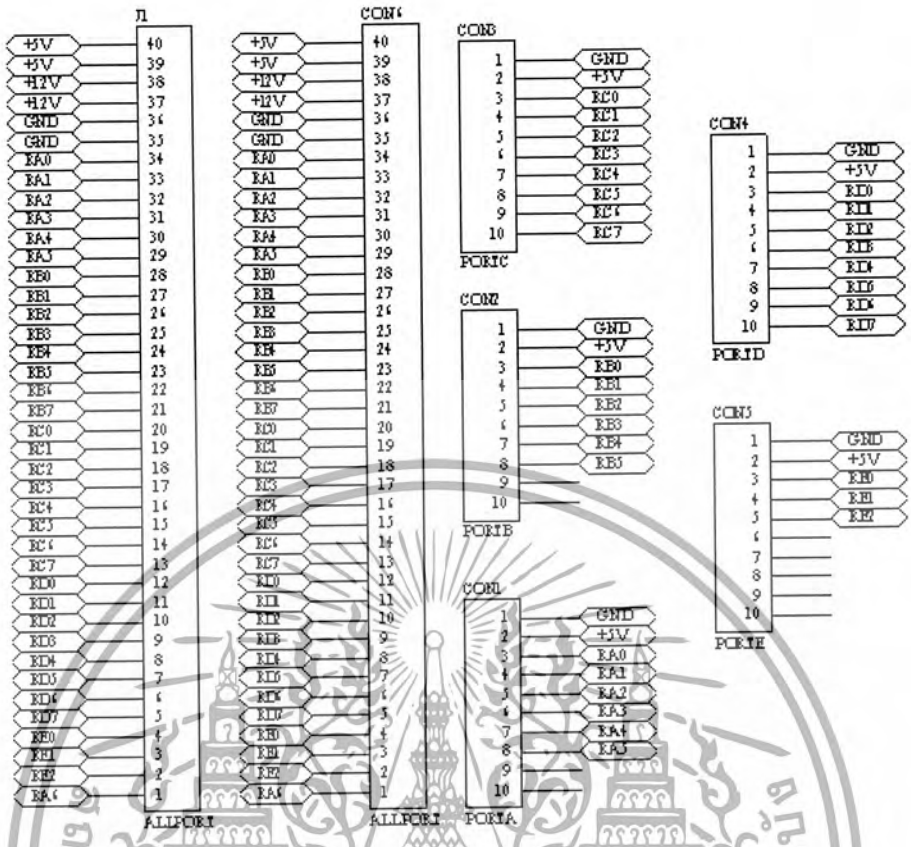
รูปที่ ข.1 วงจรโมดูลหลัก PIC-ICSP ภาคเพาเวอร์ซัพพลาย



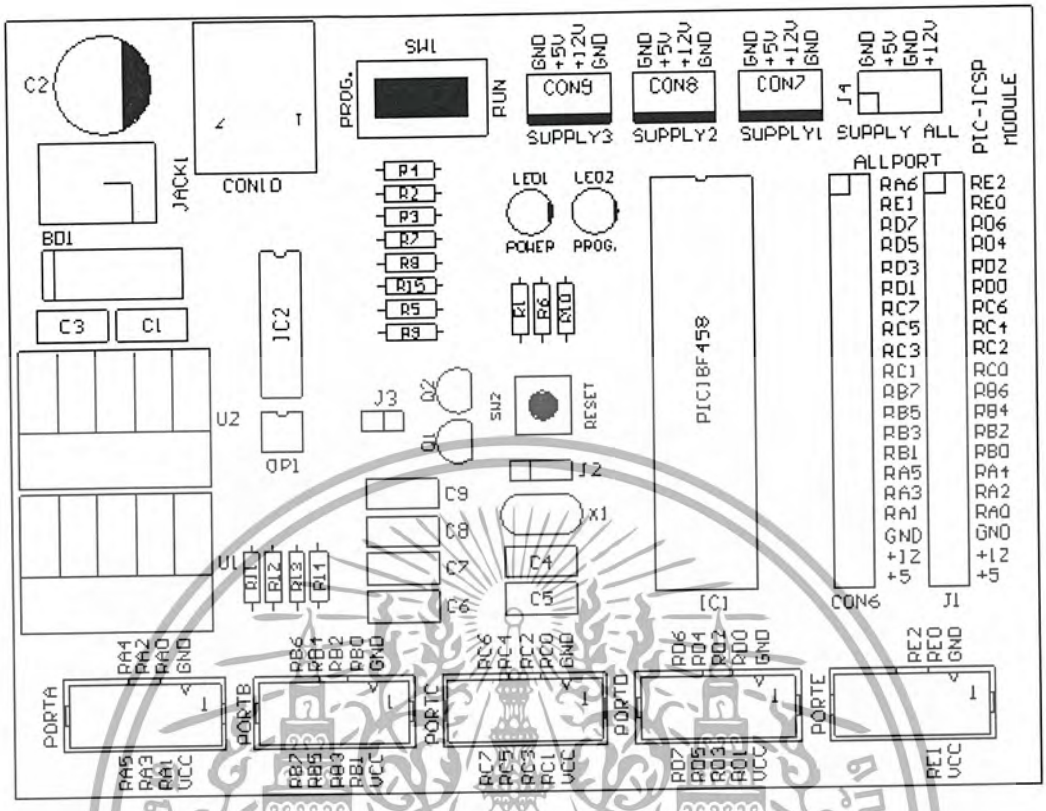
รูปที่ ข.2 วงจรโมดูลหลัก PIC-ICSP ภาค PIC In Circuit Serial Programing



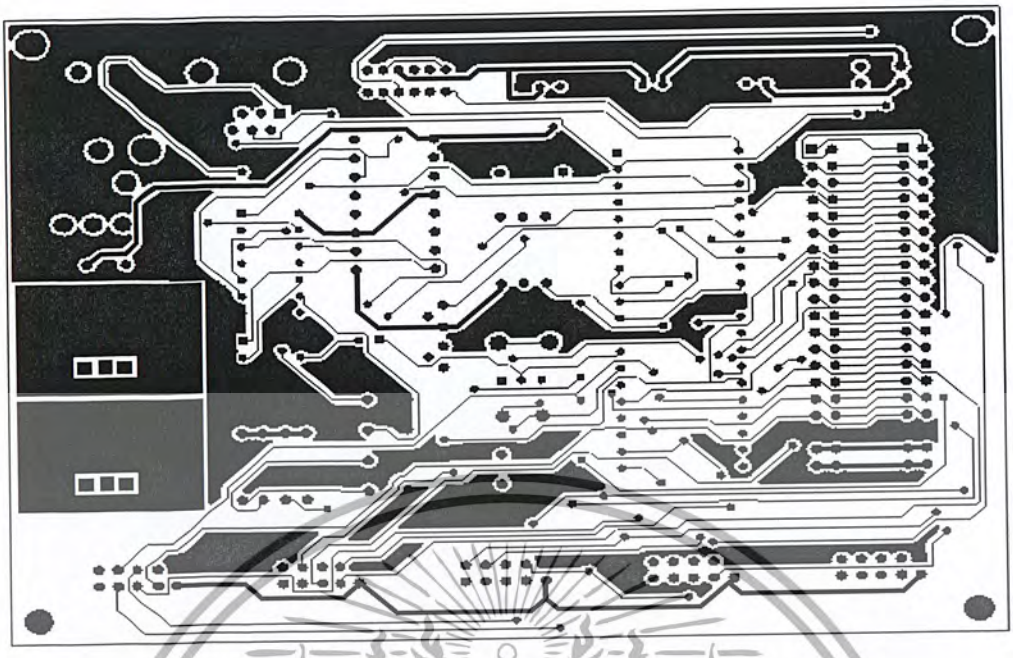
รูปที่ ข.3 วงจรโมดูลหลัก PIC-ICSP ภาคซีพียูและส่วนควบคุมการทำงาน



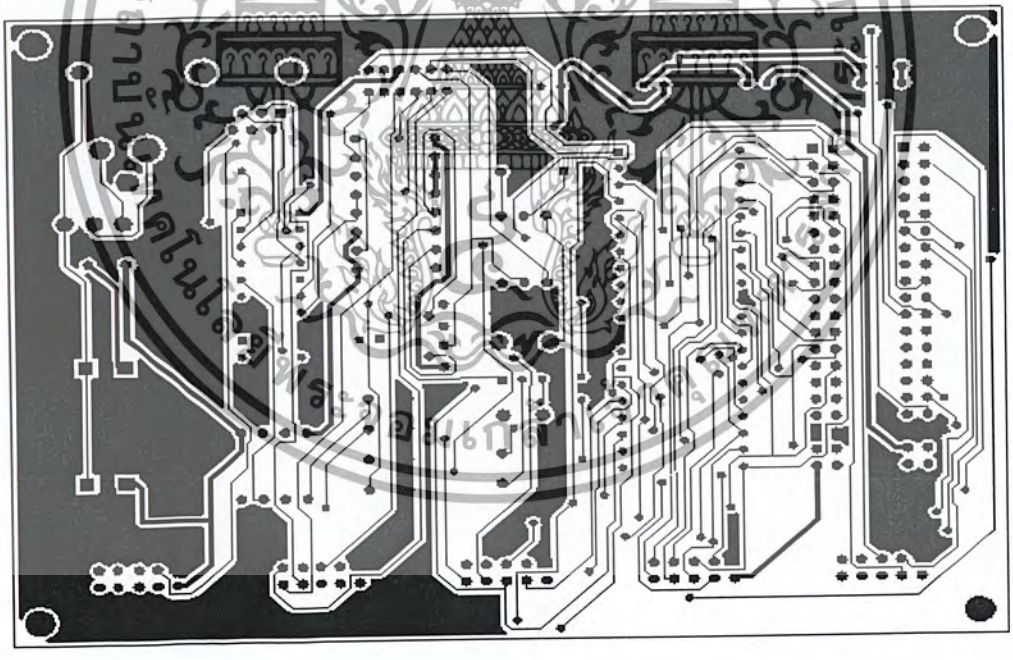
รูปที่ ข.4 วงจรโมดูลหลัก PIC-ICSP ส่วนเชื่อมต่อพอร์ตใช้งาน



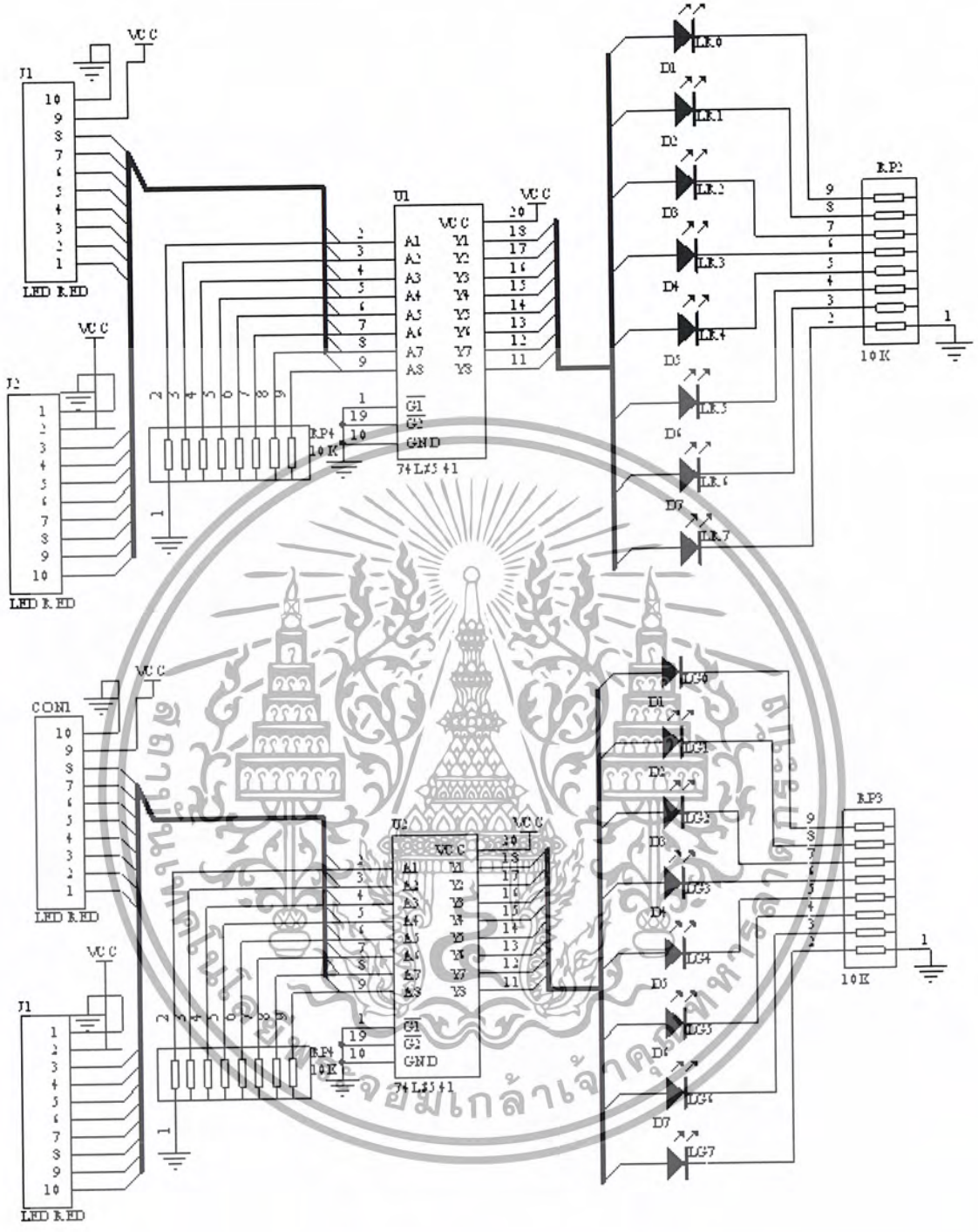
รูปที่ ข.5 ตำแหน่งการวางอุปกรณ์ของโมดูลหลัก PIC-ICSP



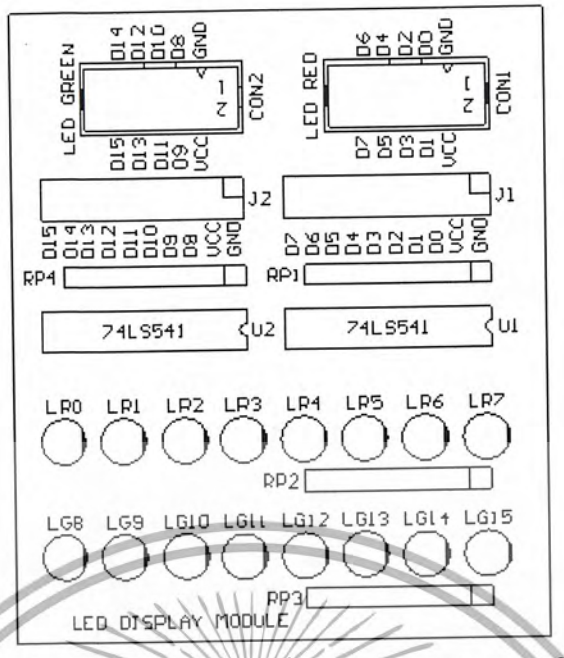
รูปที่ ข.6 ลายวงจงพิมพ์ด้านบนของโมดูลหลัก PIC-ICSP



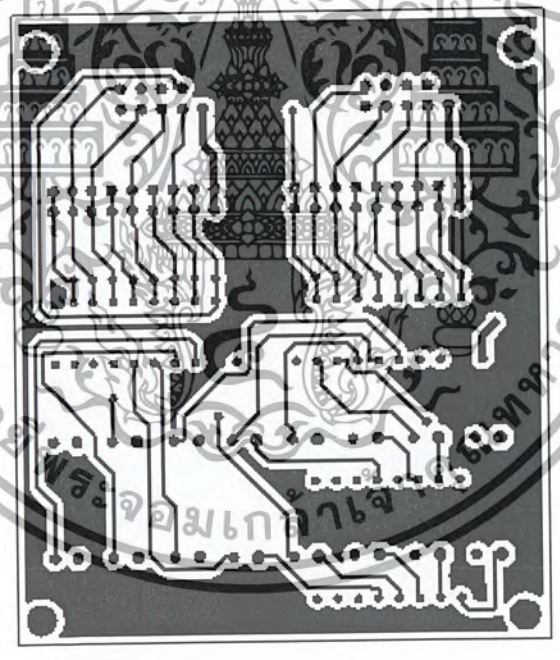
รูปที่ ข.7 ลายวงจงพิมพ์ด้านล่างของโมดูลหลัก PIC-ICSP



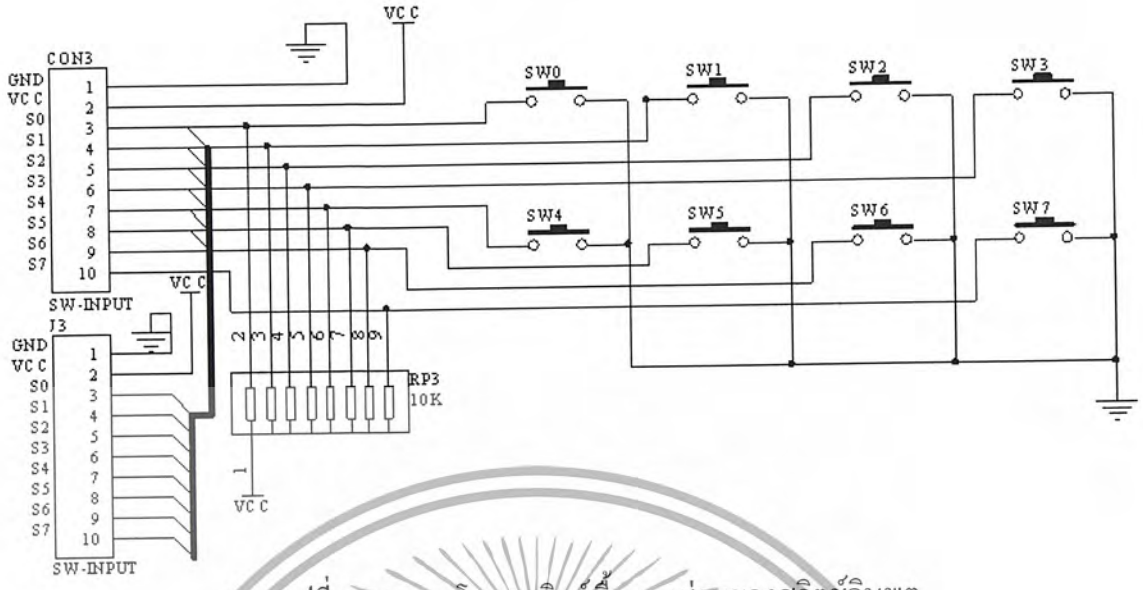
รูปที่ ข.8 วงจรโมดูลแสดงผลแอลอีดี



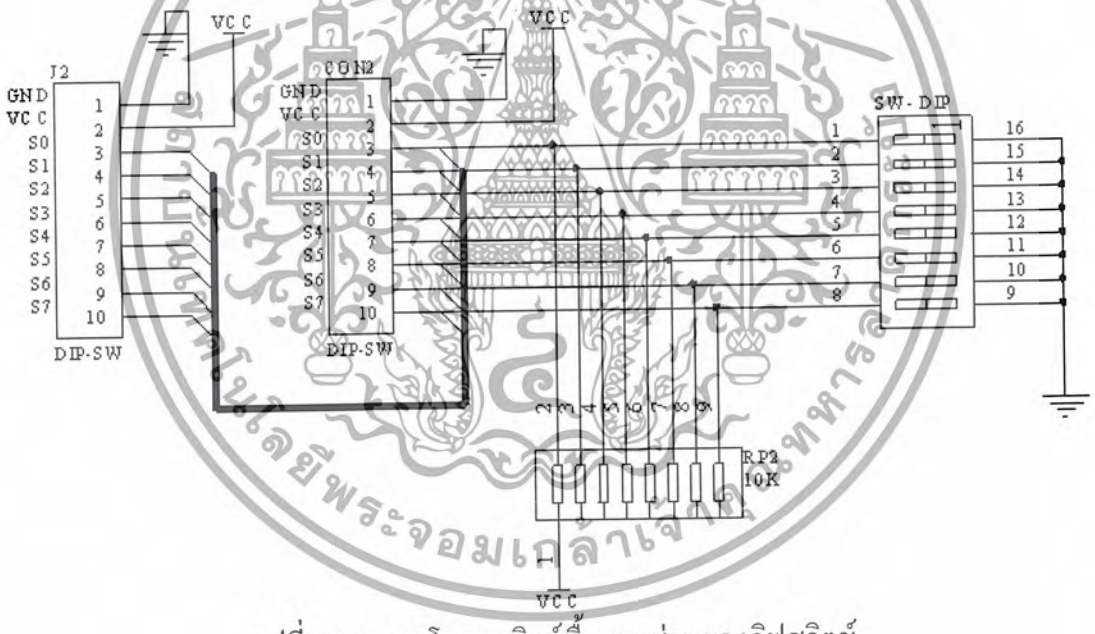
รูปที่ ข.9 ตำแหน่งการวางอุปกรณ์ของโมดูลแสดงผลแอลอีดี



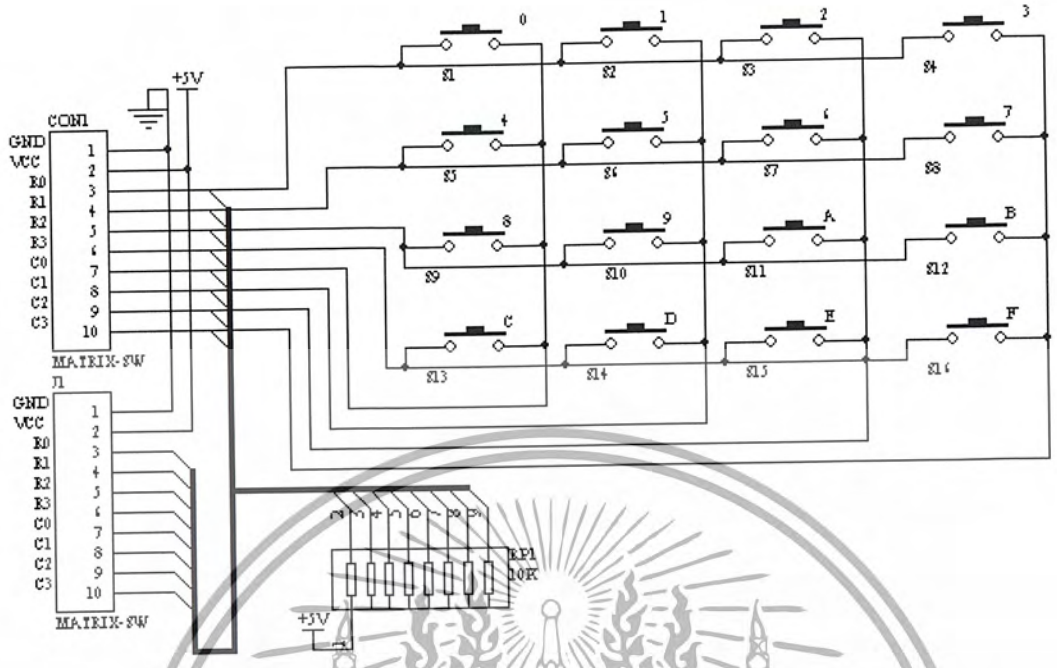
รูปที่ ข.10 ลายวงจงพิมพ์ด้านบนของโมดูลแสดงผลแอลอีดี



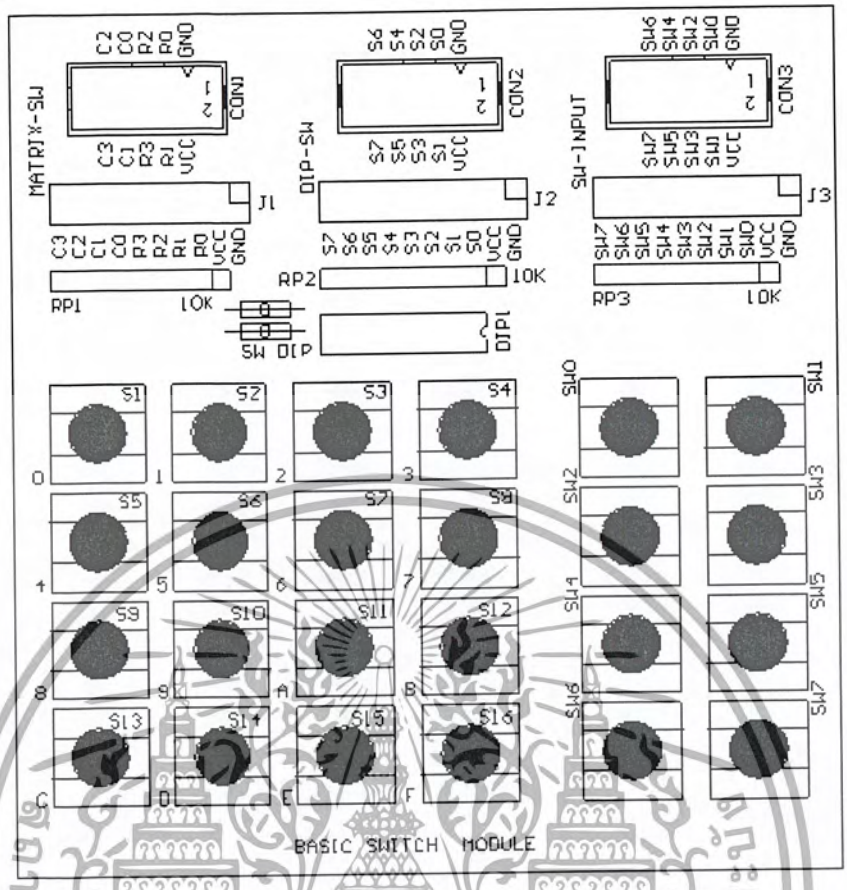
รูปที่ ข.11 วงจรโมดูลสวิตช์พื้นฐานส่วนของสวิตช์อินพุต



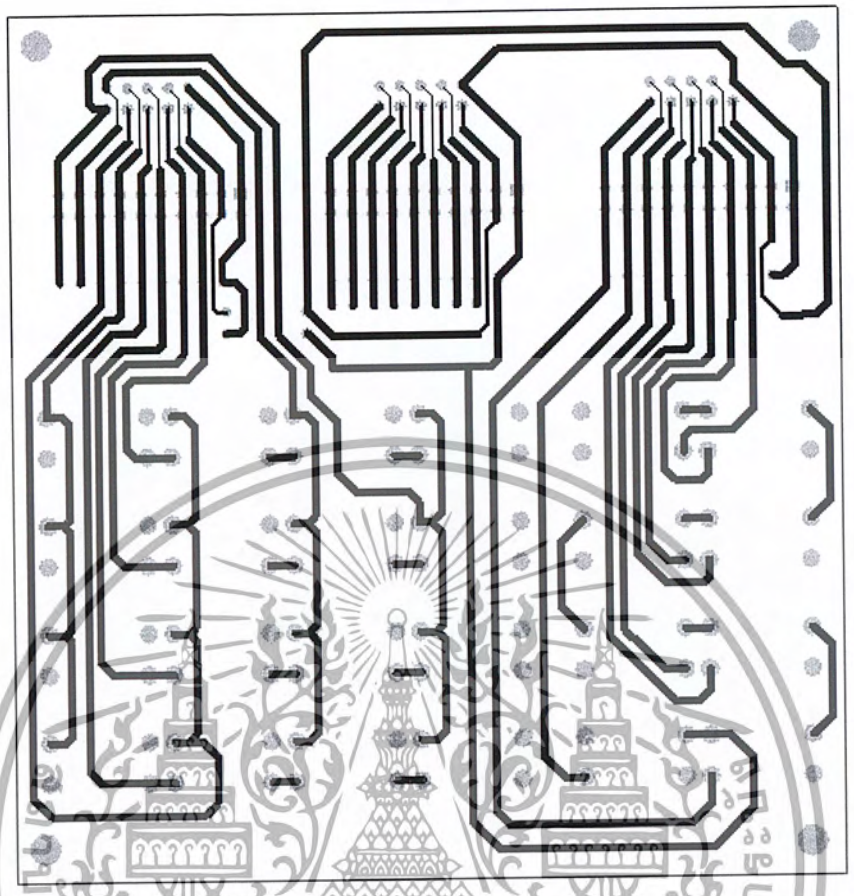
รูปที่ ข.12 วงจรโมดูลสวิตช์พื้นฐานส่วนของดิฟสวิตช์



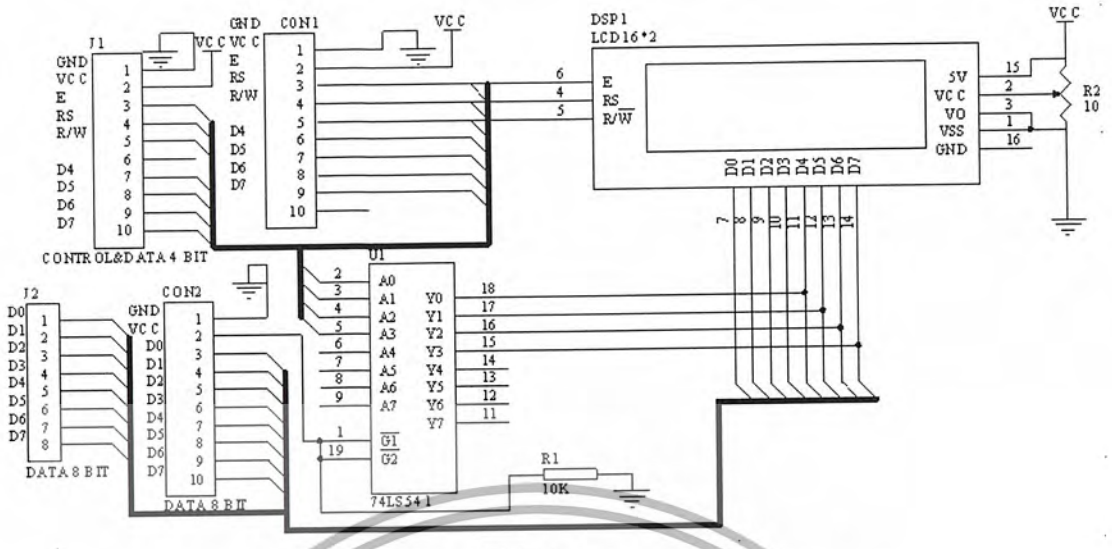
รูปที่ ข.13 วงจรโมดูลสวิตช์พื้นฐานส่วนของเมตริกซ์สวิตช์



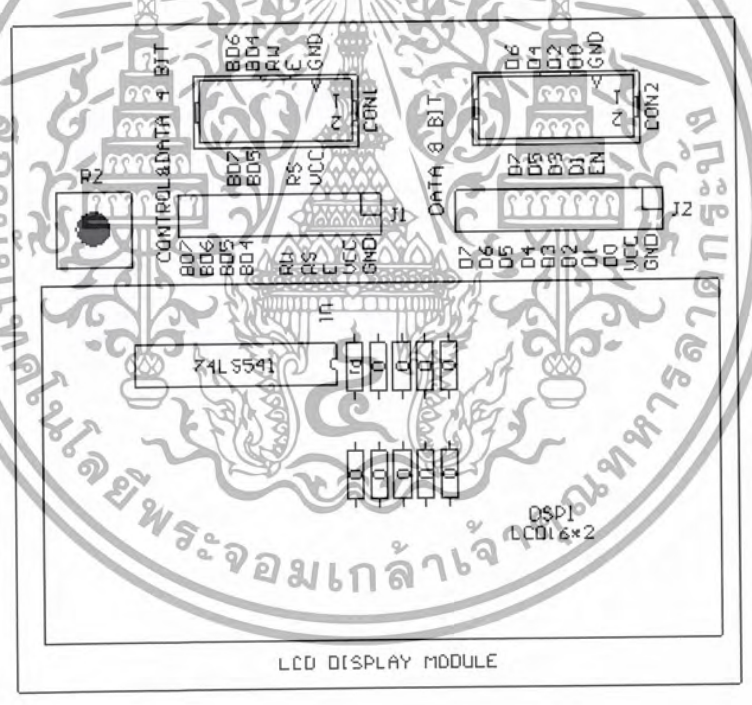
รูปที่ ข.14 ตำแหน่งการวางอุปกรณ์ของโมดูลสวิตช์พื้นฐาน



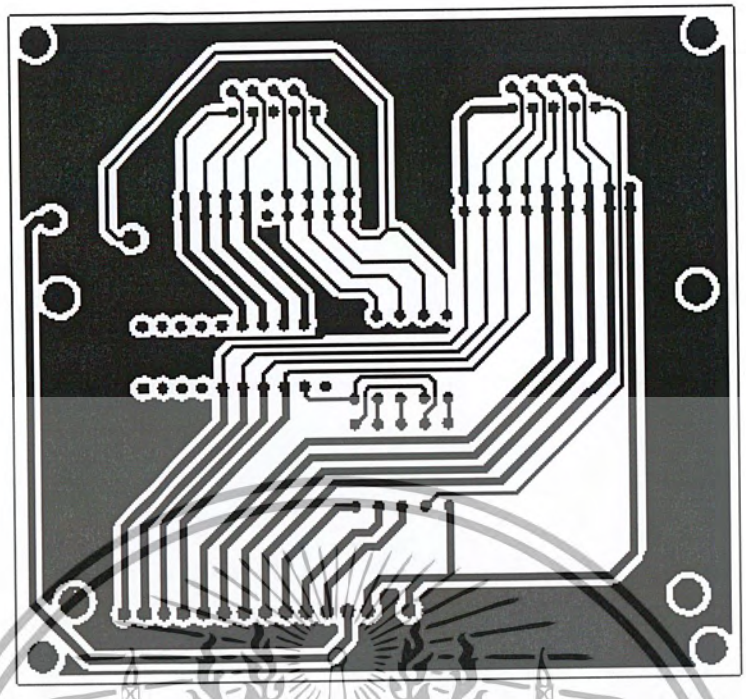
รูปที่ ข.15 ลายวงจงพิมพ์ของโมดูลสวิตช์พื้นฐาน



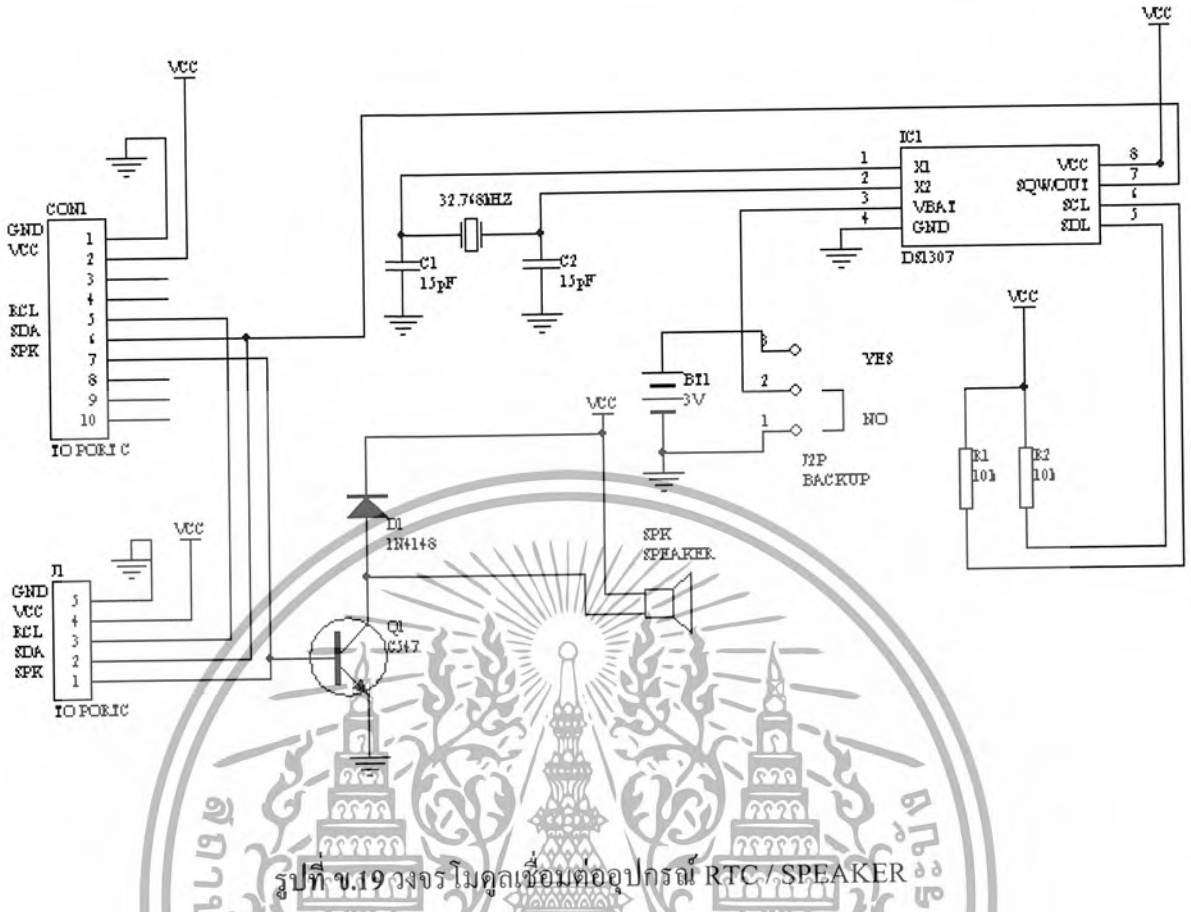
รูปที่ ข.16 วงจรโมดูลแสดงผลแอลซีดี



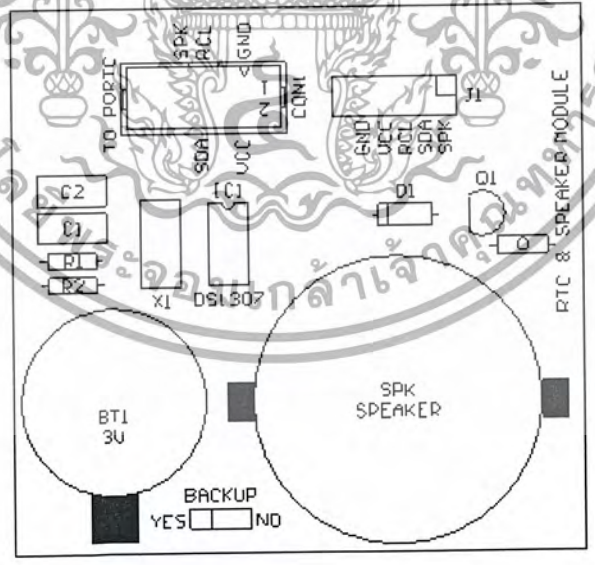
รูปที่ ข.17 ตำแหน่งการวางอุปกรณ์ของโมดูลแสดงผลแอลซีดี



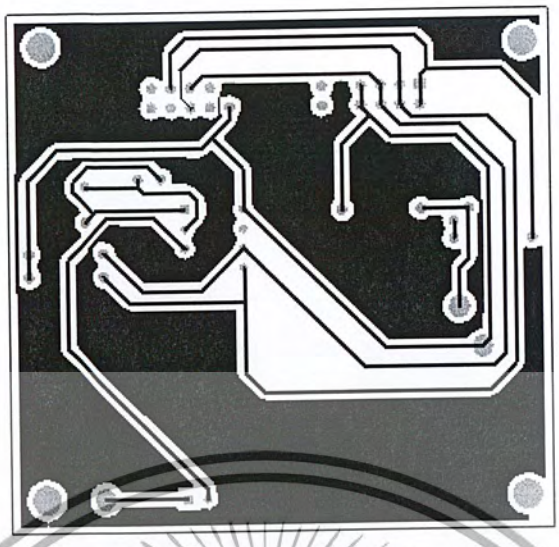
รูปที่ ข.18 ลายวงจงพิมพ์ของโมดูลแสดงผลแอลซีดี



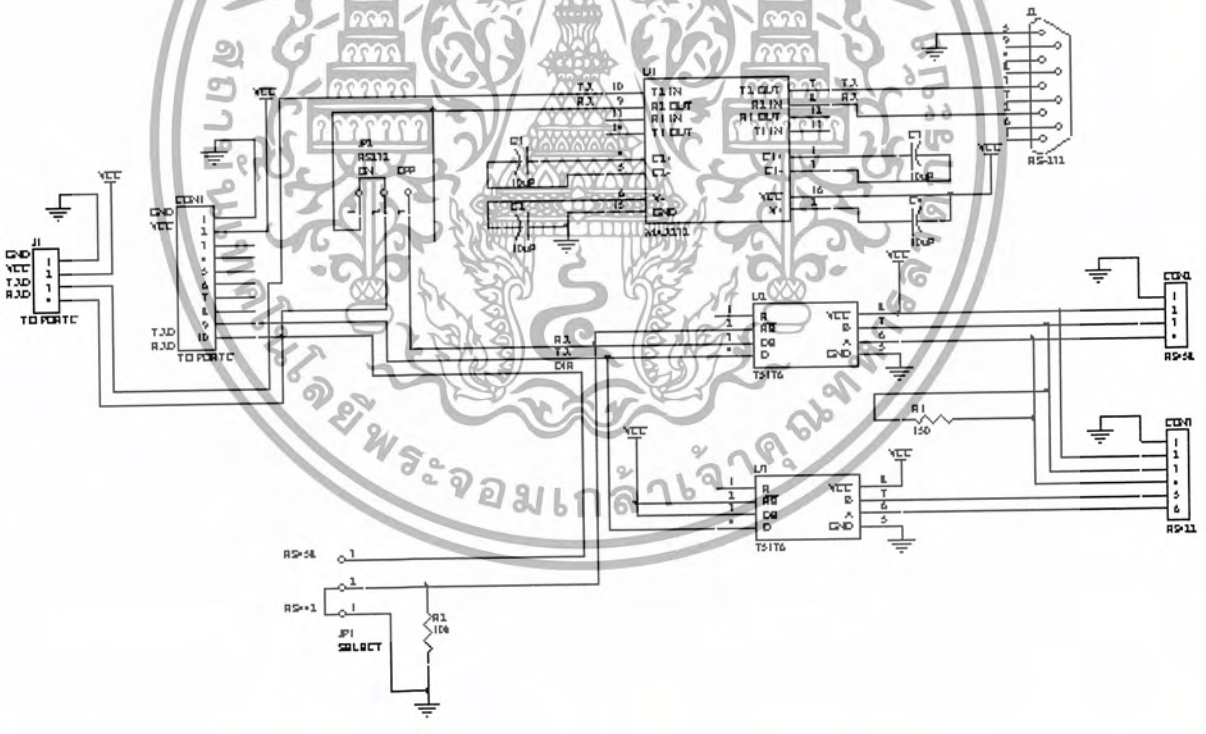
รูปที่ ข.19 วงจรโมดูลเชื่อมต่ออุปกรณ์ RTC / SPEAKER



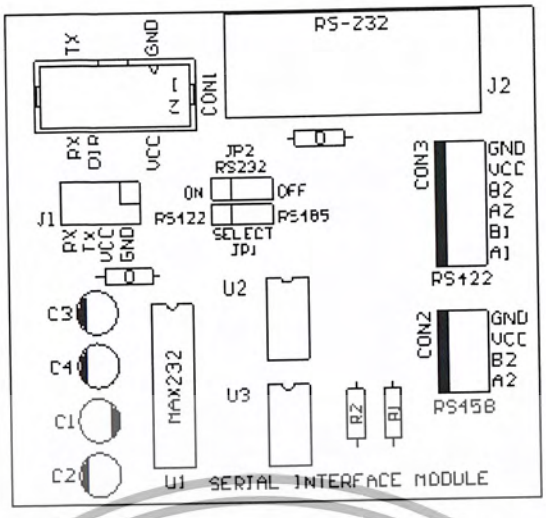
รูปที่ ข.20 ตำแหน่งการวางอุปกรณ์ของโมดูลเชื่อมต่ออุปกรณ์ RTC / SPEAKER



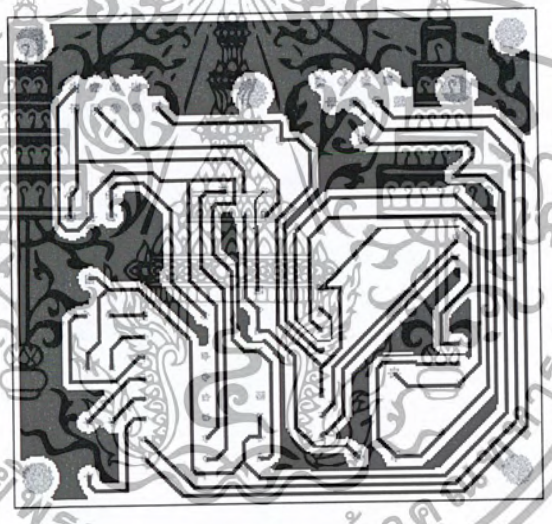
รูปที่ ข.21 ลายวงจรพิมพ์ ของโมดูลเชื่อมต่ออุปกรณ์ RTC / SPEAKER



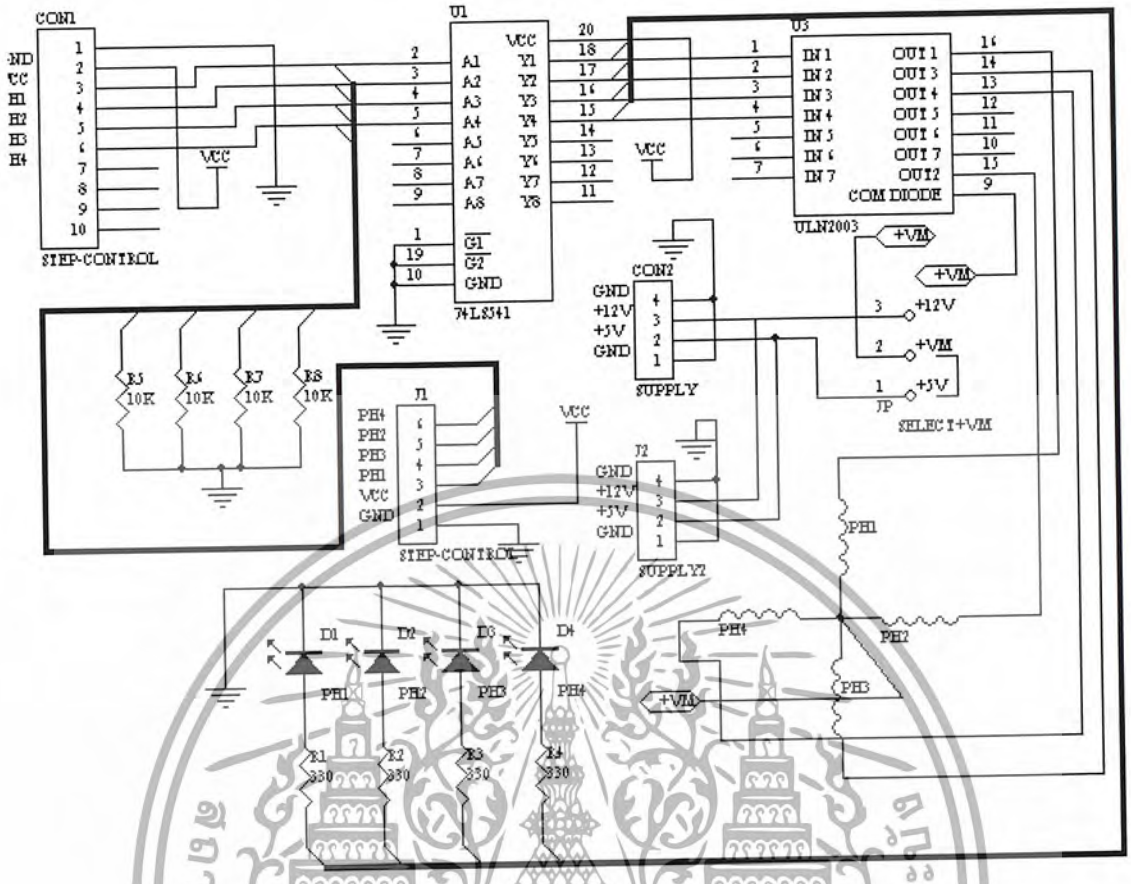
รูปที่ ข.22 วงจรโมดูลสื่อสารข้อมูลอนุกรม



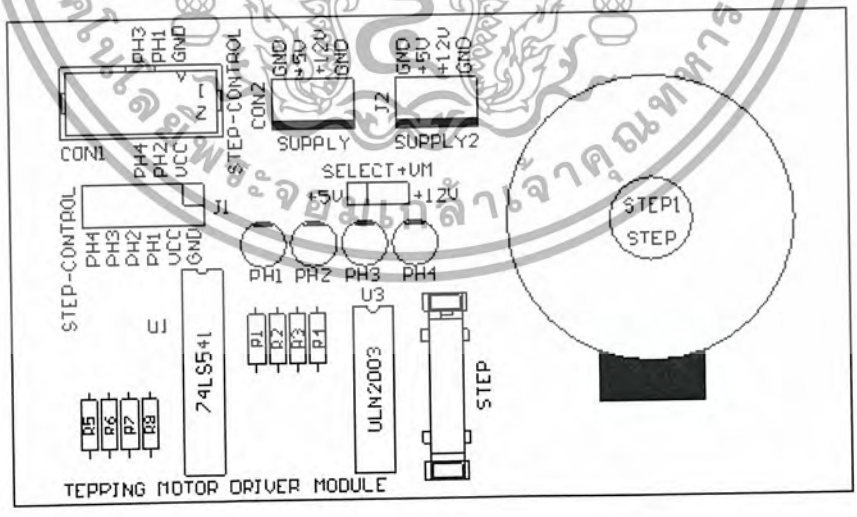
รูปที่ ข.23 ตำแหน่งการวางอุปกรณ์ของโมดูลสื่อสารอนุกรม



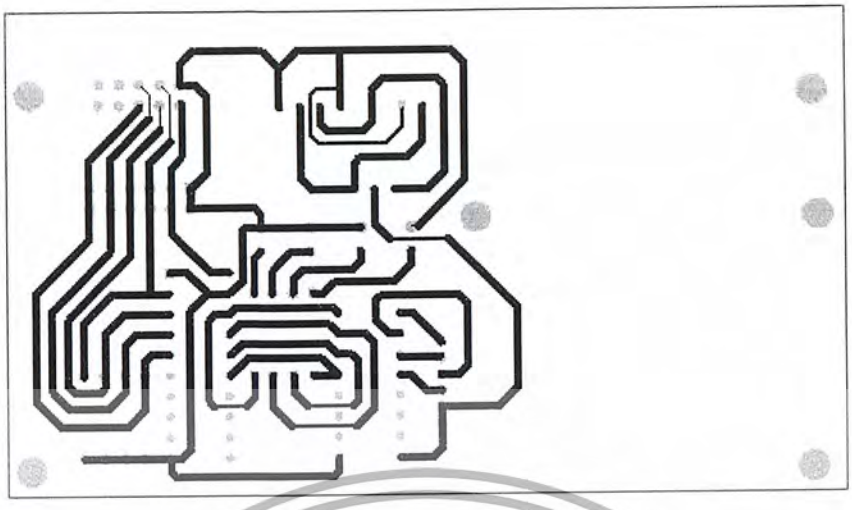
รูปที่ ข.24 ลายวงจรพิมพ์ของโมดูลสื่อสารอนุกรม



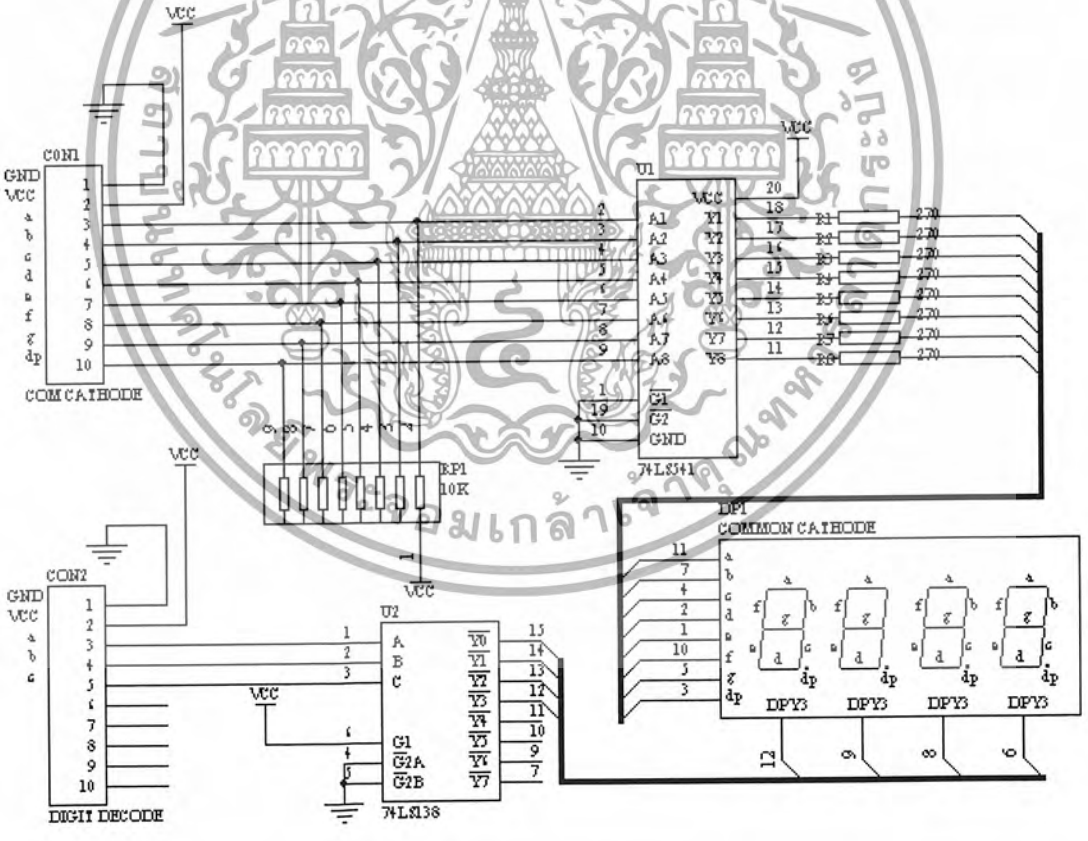
รูปที่ ข.25 วงจรขับสเตปปิ้งมอเตอร์



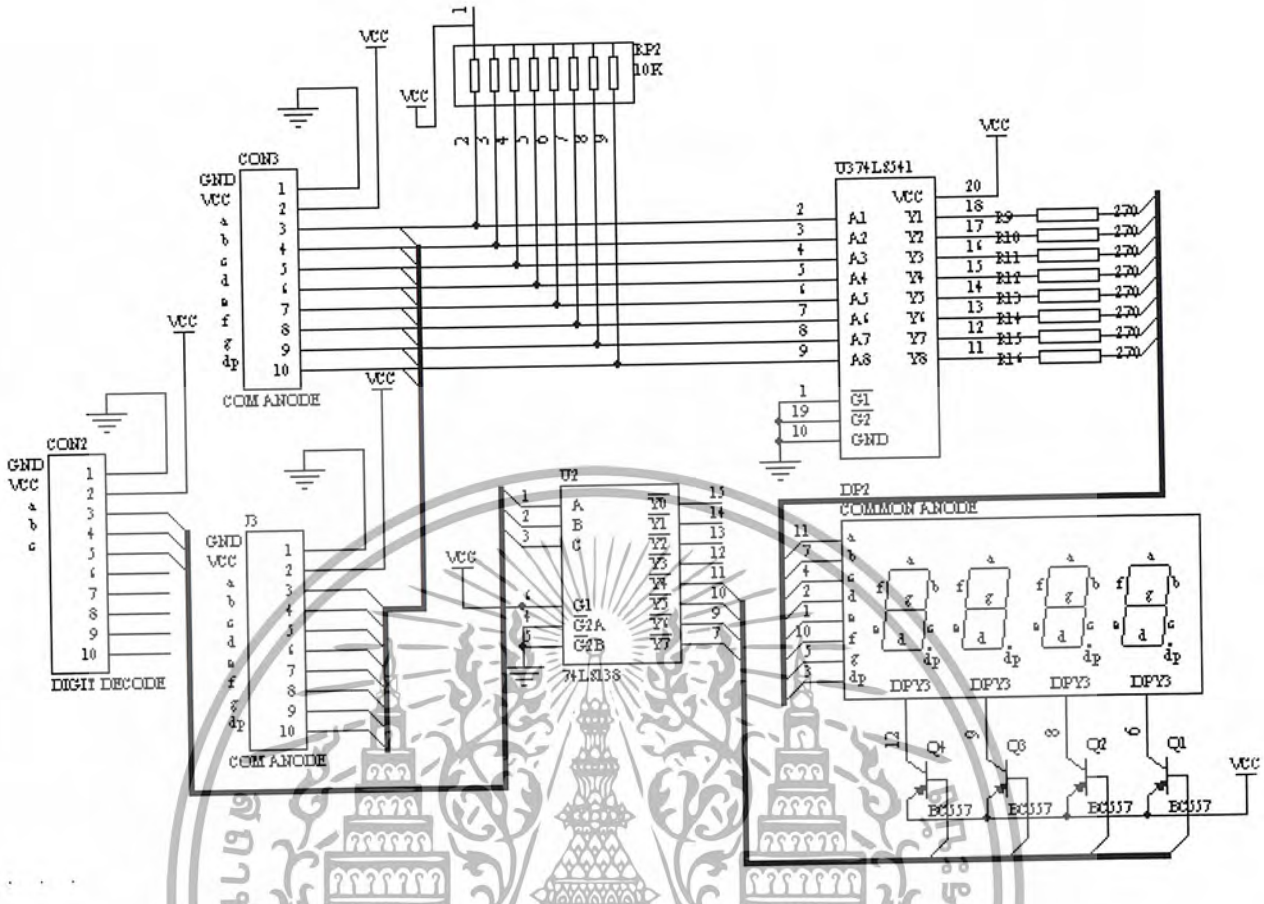
รูปที่ ข.26 ตำแหน่งการวางอุปกรณ์ของโมดูลขับสเตปปิ้งมอเตอร์



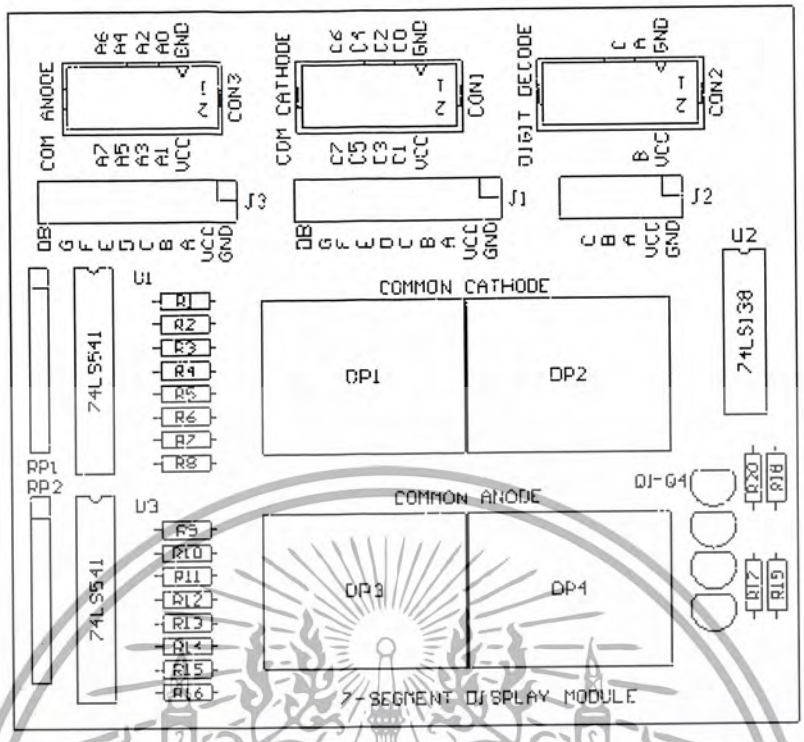
รูปที่ ข.27 ลายวงจรพิมพ์ของโมดูลขับสเตปปิ้งมอเตอร์



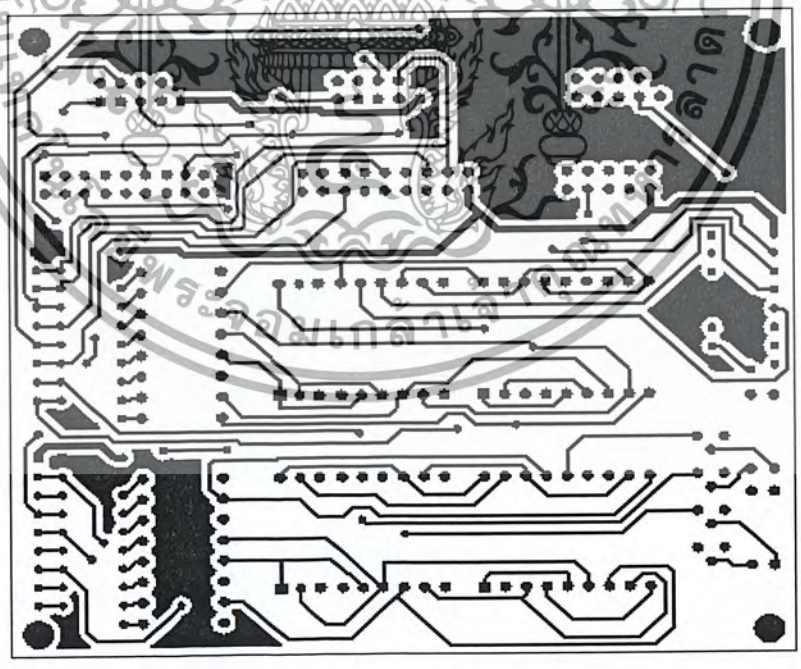
รูปที่ ข.28 วงจรแสดงผลตัวเลขเจ็ดส่วนคอมมอนคาโถด



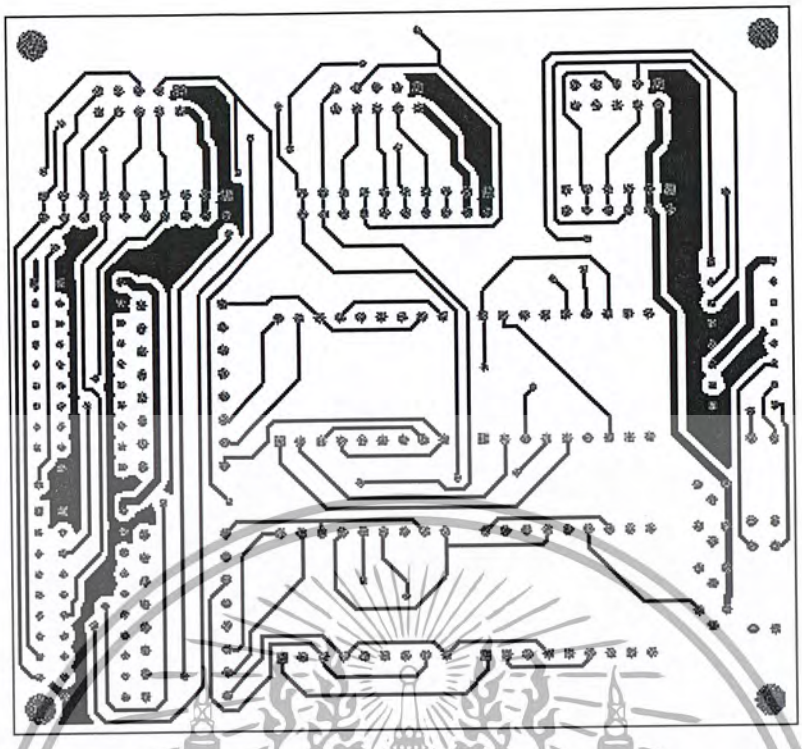
รูปที่ ข.29 วงจรแสดงผลตัวเลขเจ็ดส่วนคอมมอนอาโนด



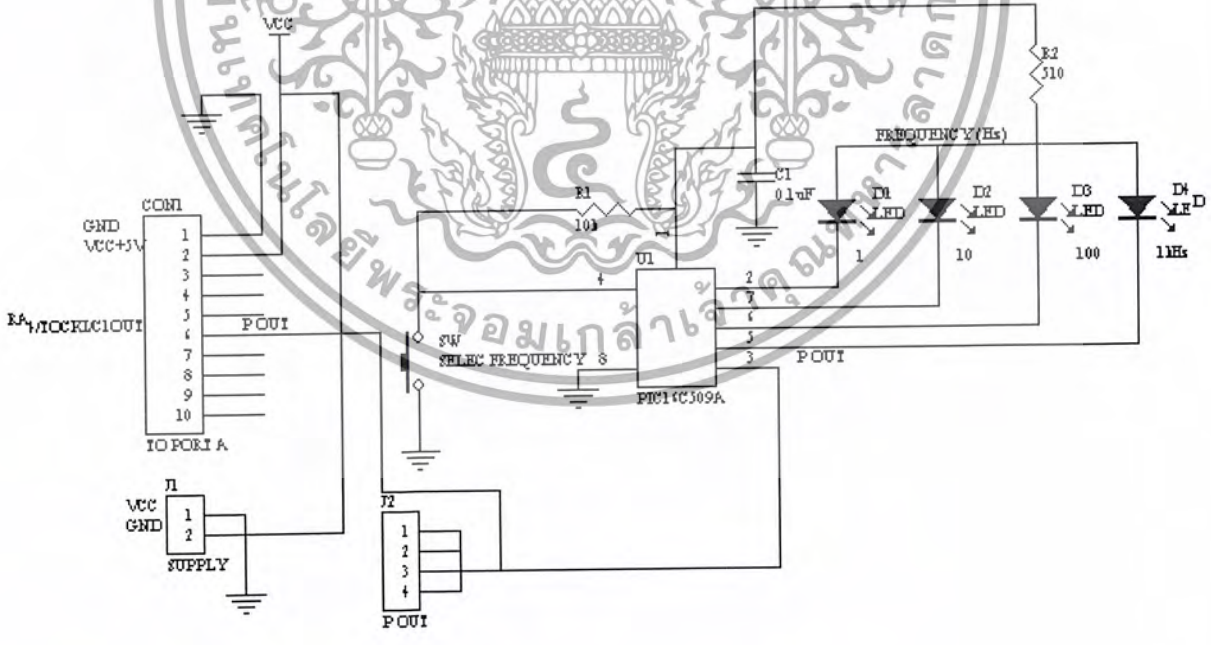
รูปที่ ข.30 ตำแหน่งการวางอุปกรณ์ของโมดูลแสดงผลตัวเลขเจ็ดส่วน



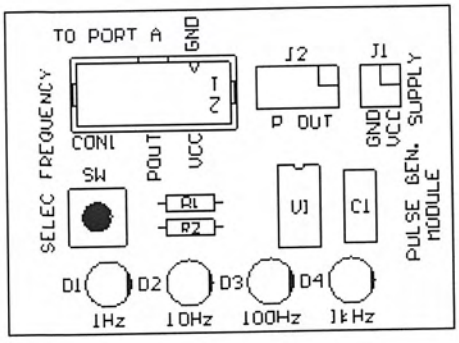
รูปที่ ข.31 ลายวงจรพิมพ์ด้านบนของโมดูลแสดงผลตัวเลขเจ็ดส่วน



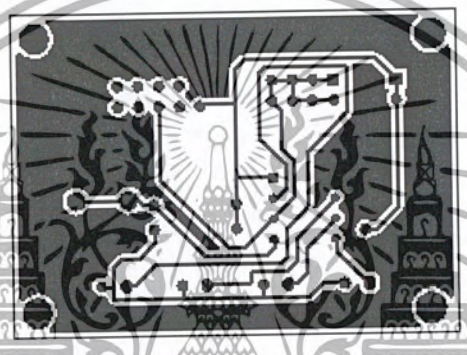
รูปที่ ข.32 ลายวงจรพิมพ์ด้านล่างของโมดูลแสดงผลตัวเลขเจ็ดส่วน



รูปที่ ข.33 วงจรโมดูลพัลส์เจเนอเรเตอร์



รูปที่ ข.34 ตำแหน่งการวางอุปกรณ์ของโมดูลพัลส์เจเนอเรเตอร์



รูปที่ ข.35 ลายวงจรพิมพ์ของโมดูลพัลส์เจเนอเรเตอร์

ภาคผนวก ค

รายการอุปกรณ์

รายการวัสดุอุปกรณ์ที่ใช้ในการทำชุดทดลอง

ไอซี (IC)

ลำดับที่	รายการ	จำนวน
1	74LS541	10 ตัว
2	74LS138	1 ตัว
3	715176	3 ตัว
4	74LS125	1 ตัว
5	74LS595	1 ตัว
6	LF351	1 ตัว
7	LF353N	1 ตัว
8	MAX232	1 ตัว
9	7812	1 ตัว
10	7805	1 ตัว
11	DS3107	1 ตัว
12	PIC16C509	1 ตัว

ตัวเก็บประจุ (Capacitor)

ลำดับที่	รายการ	จำนวน
1	15 pF	2 ตัว
2	22 pF	2 ตัว
3	33 pF	2 ตัว
4	220 pF	1 ตัว
5	10 nF	8 ตัว
6	0.1 uF	1 ตัว

7	1 uF	1 ตัว
8	10 uF	4 ตัว
9	100 uF	1 ตัว
10	2200 uF	2 ตัว

ตัวต้านทาน (Resistor)

ลำดับที่	รายการ	จำนวน
1	10 โอห์ม	2 ตัว
2	39 โอห์ม	8 ตัว
3	56 โอห์ม	8 ตัว
4	100 โอห์ม	15 ตัว
5	150 โอห์ม	1 ตัว
6	220 โอห์ม	18 ตัว
7	330 โอห์ม	18 ตัว
8	680 โอห์ม	24 ตัว
9	820 กิโลโอห์ม	2 ตัว
10	1 กิโลโอห์ม	1 ตัว
11	2 กิโลโอห์ม	11 ตัว
12	10 กิโลโอห์ม	20 ตัว
13	20 กิโลโอห์ม	20 ตัว
14	33 กิโลโอห์ม	17 ตัว
15	120 กิโลโอห์ม	3 ตัว
16	แบบแถว 330 โอห์ม	2 ตัว
17	แบบแถว 10 โอห์ม	8 ตัว
18	ปรับค่าได้ 10 กิโลโอห์ม	1 ตัว
19	ปรับค่าได้ 50 กิโลโอห์ม	3 ตัว

ไดโอดเปล่งแสง (LED)

ลำดับที่	รายการ	จำนวน
1	LED กลม ขนาด 3 mm สีแดง	17 ตัว
2	LED กลม ขนาด 3 mm สีเขียว	9 ตัว
3	7-Segment Commond Cathod 2 หลักคู่	2 ตัว
4	7-Segment Commond Anode 2 หลักคู่	2 ตัว

ทรานซิสเตอร์ (Transistor)

ลำดับที่	รายการ	จำนวน
1	BC547	4 ตัว
2	BC557	6 ตัว
3	BD139	2 ตัว
4	BD140	17 ตัว

คอนเนคเตอร์ (Connector)

ลำดับที่	รายการ	จำนวน
1	10 ขา	20 ตัว
2	20 ขา	6 ตัว
3	4 ขา	4 ตัว
4	6 ขา	4 ตัว
5	5 ขา	4 ตัว

สวิตช์ (Switch)

ลำดับที่	รายการ	จำนวน
1	Switch กดติดปล่อยดับ 5 mm	25 ตัว
2	DIP Switch 8 ตำแหน่ง	1 ตัว

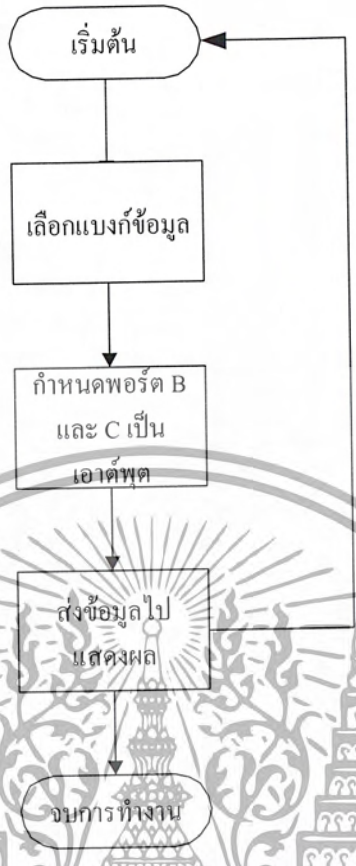
ไดโอด (Diode)

ลำดับที่	รายการ	จำนวน
1	1N4148	2 ตัว
2	Diode Bridge	1 ตัว

รายการอื่นๆ

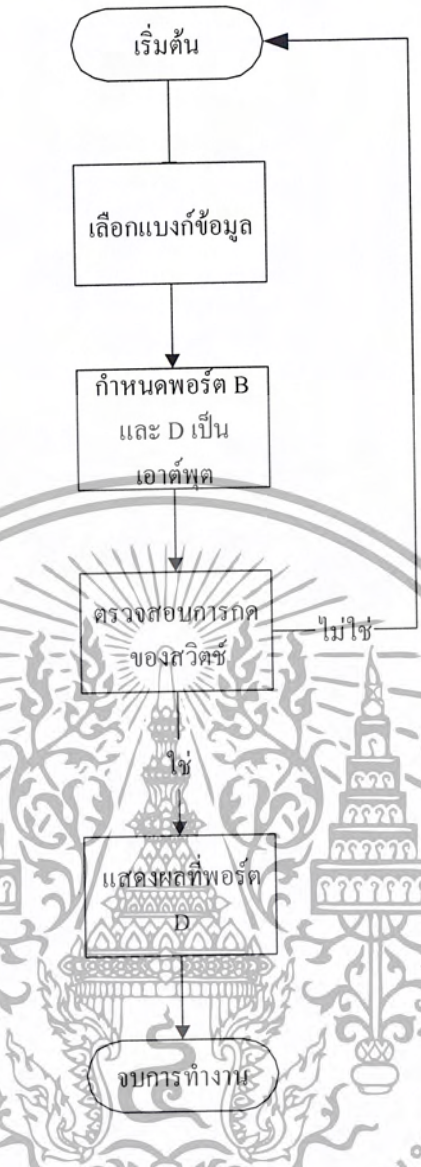
ลำดับที่	รายการ	จำนวน
1	คริสตอลความถี่ 10 เมกะเฮิตซ์	1 ตัว
2	LCD Character ขนาด 16 ตัวอักษร 2 บรรทัด	1 ตัว
3	ลำโพงบัซเซอร์	1 ตัว
4	ลำโพง 1 W	1 ตัว
5	สเตปปิ้งมอเตอร์	1 ตัว

ภาคผนวก ง
แผนผังการทำงานของโปรแกรม



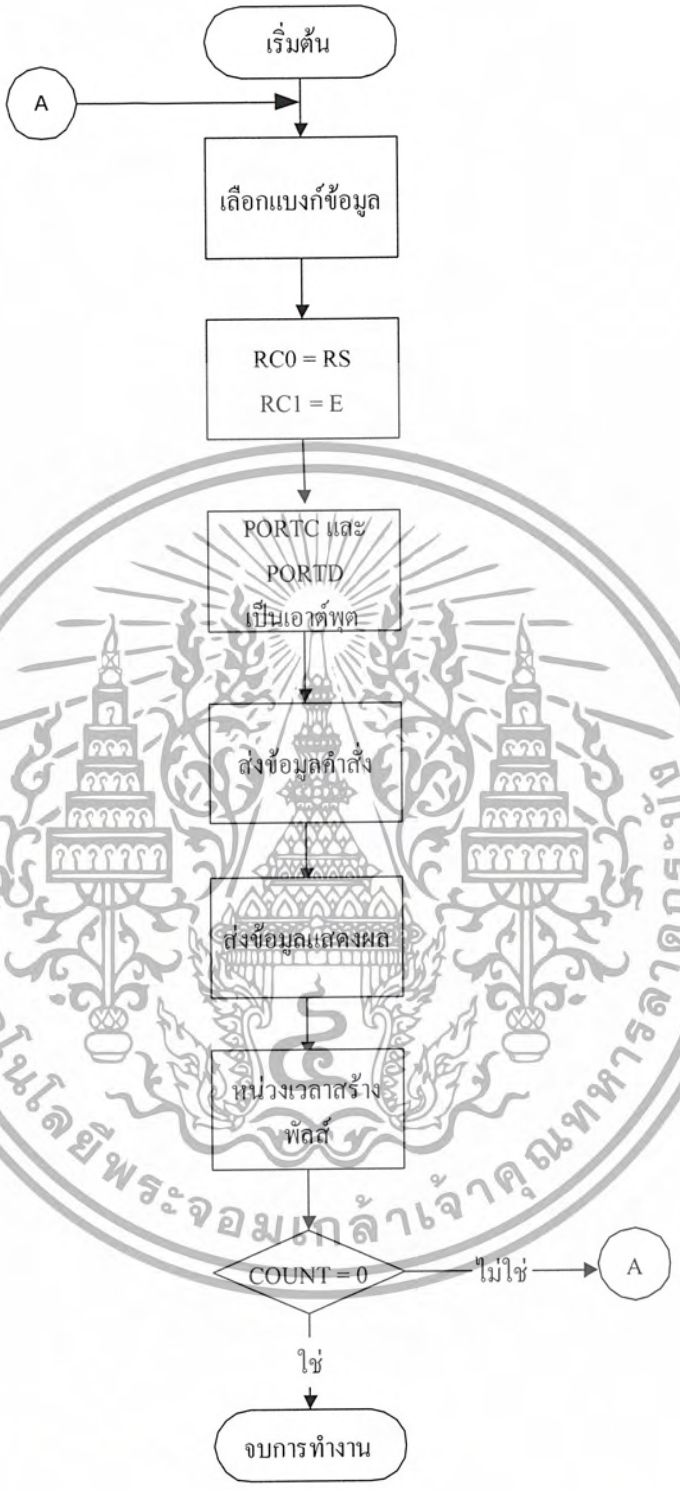
รูปที่ ง.1 ผังงานโปรแกรมการทดลองใช้งานโมดูลแสดงผลแอลอีดี

รูปที่ ง.2 ผังงาน โปรแกรมการทดลอง ใช้งานสวิตช์อินพุต



รูปที่ ง.3 ผังงานโปรแกรมการทดลองใช้งานดิฟสวิตช์

รูปที่ ง.4 ผังงานโปรแกรมการทดลองใช้งานโมดูลขับสเตปปิ้งมอเตอร์



รูปที่ ง.5 ผังงานโปรแกรมการทดลองใช้งานโมดูลแสดงผลแอลซีดี

ภาคผนวก จ

ใบงานการทดลอง

ใบงานที่ 1
การทดลองใช้งานโหมดสลีปของ PIC18F458

จุดประสงค์

1. เพื่อให้สามาถอธิบายการทำงานของพอร์ตทั้งหมดของ PIC18F458 ได้
2. เพื่อให้สามารถเขียนโปรแกรมใช้งานพอร์ตของ PIC18F458 เป็นเอาต์พุตได้
3. เพื่อให้สามารถประยุกต์แก้ไขโปรแกรมในการทดลองได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ Epic Win
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
5. โมดูลแสดงผลแอลอีดี

ทฤษฎีเบื้องต้น

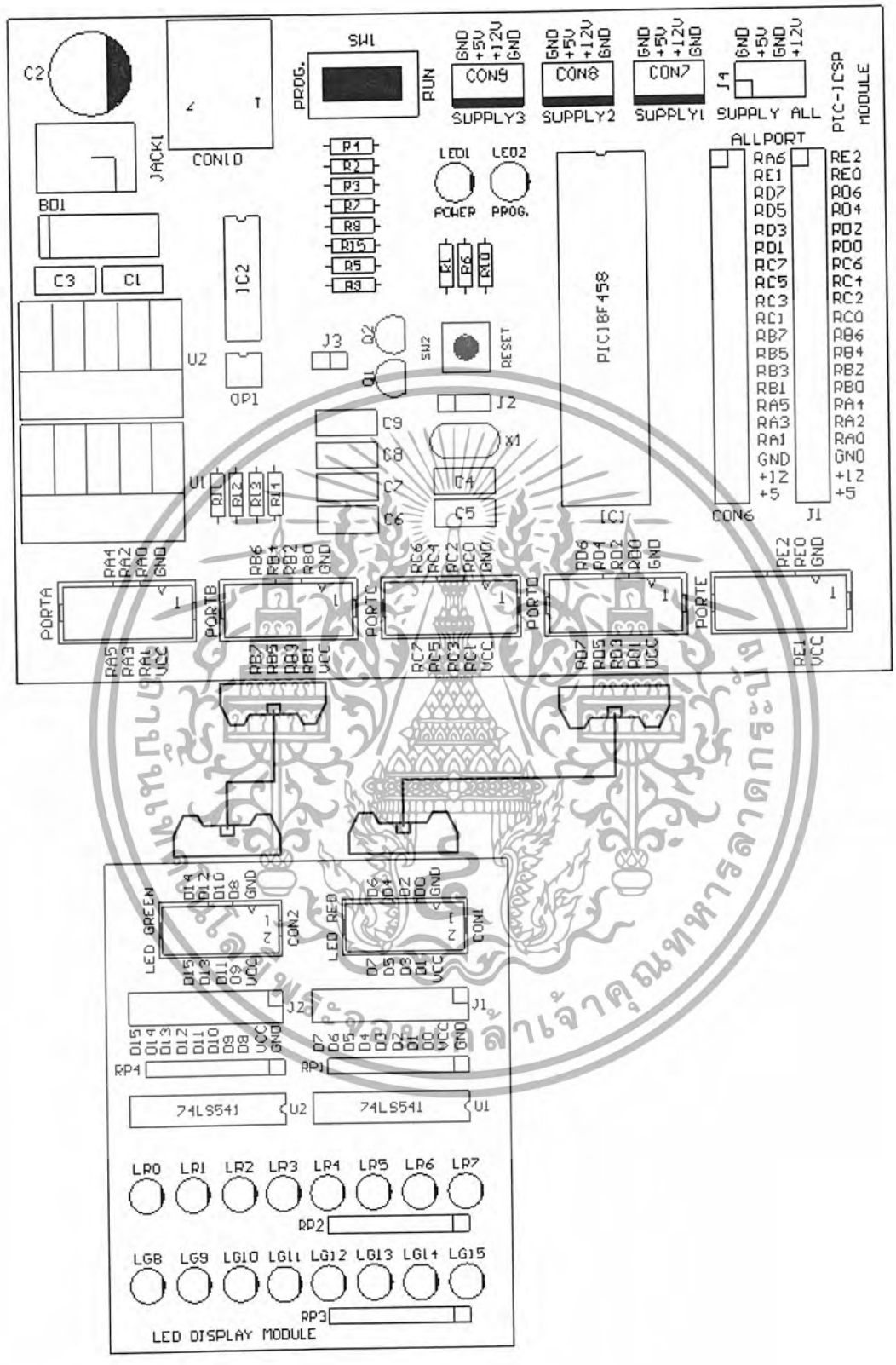
ไมโครคอนโทรลเลอร์ PIC18F458 มีพอร์ตใช้งาน 5 พอร์ต จำนวน 34 บิต และด้วยความสามารถของพอร์ตใน PIC18F458 ที่สามารถทำงานได้หลายอย่าง จึงจำเป็นอย่างยิ่งที่ผู้ใช้งานต้องทำความเข้าใจถึงโครงสร้างฮาร์ดแวร์และการกำหนดหรือเลือกฟังก์ชั่นการทำงานให้แก่ขาพอร์ต แต่ละขาด้วยกระบวนการทางซอฟต์แวร์ ทั้งนี้เพื่อสามารถใช้งานพอร์ตทั้งหมดของ PIC18F458 ได้อย่างมีประสิทธิภาพสูงสุดในส่วนนี้จะกล่าวถึงภาพรวมของพอร์ตทั้งหมด ตั้งแต่พอร์ต A ถึง E โดยจะเน้นไปที่โครงสร้างทางฮาร์ดแวร์และ ฟังก์ชั่นการทำงานในภาพรวม

พอร์ตของ PIC18F458 แบ่งออกเป็นพอร์ต A มีขนาด 7 บิต พอร์ต B, C และ D มีอย่างละ 8 บิต ส่วนพอร์ต E มีขนาด 3 บิต ในการเลือกใช้งานพอร์ต RA0-RA3, RA5 และพอร์ต E นั้นจะต้องเขียนค่าไปยังรีจิสเตอร์ ADCON1 เป็น '0000011x' ด้วยโดย x คือ เป็น 0 หรือ 1 ก็ได้ เพื่อทำการดิสเอเบิลโหมด PSP ถ้าไม่กระทำเช่นนี้แล้วจะไม่สามารถใช้งานเป็นพอร์ตอินพุตเอาต์พุตปกติได้ ส่วนพอร์ตอื่นๆ คือ B, C และ D สามารถที่จะใช้งานเป็น

พอร์ตอินพุตเอาต์พุตได้ทันทีโดยไม่ต้องทำการกำหนดค่าอย่างข้างต้นที่ได้กล่าวมาส่วนการกำหนดทิศทางทางขาพอร์ตนี้กระทำผ่านรีจิสเตอร์ TRISA, TRISB, TRISC, TRISD, TRISE ไล่เรียงลำดับหากต้องการกำหนดให้ขาพอร์ตในบิตใดเป็นอินพุตต้องเขียนข้อมูล “1” ไปยังบิตนั้น และในทางตรงข้ามหากต้องการกำหนดให้เป็นขาเอาต์พุตให้เขียนข้อมูล “0” ไปยังบิตที่ต้องการ ดังตัวอย่างส่วนการกำหนดพอร์ต B, C, E นั้นสามารถกระทำได้คล้ายกัน

```
;********ตัวอย่างการกำหนดพอร์ต A************
INIT    MOVLW   0X06        ; w = “00000110”
        MOVWF   ADCON1      ; เขียนค่า 0X06 ไปยัง ADCON1 เพื่อกำหนดให้เป็น I/O
        MOVLW   OX0F        ; ‘00001111’
        MOVWF   PORTA       ; RA0-RA3 เป็นอินพุต ,RA4-RA7 เป็นเอาต์พุต
        CLRF    PORTA       ;เคลียร์ค่าพอร์ต A
START   .....       ;.....

;******* ตัวอย่างการกำหนดพอร์ต B **************
  INIT  MOVLW   0XF0        ; w = “11110000”
        MOVWF   TRISB       ;RB0-RB4 เป็นอินพุต RB5-RB7 เป็นเอาต์พุต
        CLRF    PORTB       ;เคลียร์ค่าพอร์ต B
  START ....        ;....
;หมายเหตุ ในการกำหนดพอร์ต C,D กระทำคล้ายกับพอร์ต B
```

รูปที่ จ.1 ตัวอย่างการกำหนดพอร์ตของ PIC18F458



รูปที่ จ.2 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP และโมดูลแสดงผลแอลอีดี

```
;*****************************************************************
; Filename :            lab1.asm
; Descripton :          test I/O Port
; MCU :                 PIC18F458
;*****************************************************************
list        p=18f458
  #include <p18f458.inc>
ROUND   EQU     0x20
COUNT1  EQU     0x21    ; RAM
COUNT2  EQU     0x22    ; RAM
COUNT3  EQU     0x23    ; RAM
        ORG     0x0000
  ;     MOVLW   B'00000000'
        MOVLW   B'00000111'
        MOVWF   ADCON1
        CLRF    TRISA
        CLRF    TRISE
        CLRF    PORTA
        CLRF    PORTB
START   MOVLW   0XFF
        MOVWF   PORTA
        MOVLW   0XFF
        MOVWF   PORTE
        CLRF    PORTA
        GOTO    START
        END
```

รูปที่ จ.3 โปรแกรมทดลองใช้งานพอร์ตของ PIC18F458

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.1 ทำการคอมไพล์ให้เป็น Hex File
2. ไปที่เมนู OPTION เลือก Development Mode กำหนดโหมดเป็น MPLAB-SIM Simulator
3. เลือก Process เป็นรุ่น PIC18F458 จากนั้นกดปุ่ม Apply แล้วกด OK
4. ไปที่เมนู Project เลือก New Project ตั้งชื่อ Project Name แล้วกด OK

5. ไปที่เมนู Edit Project ให้ Remove Project File ออกให้หมดแล้วกด OK

6. ไปที่เมนู File เลือก New Souse เพื่อเขียนโปรแกรมดังรูปที่ จ.3 ลงไป

7. จากนั้น Save File เป็นนามสกุล . ASM จากนั้นทำการแอสเซมเบลอร์

8. ไปที่เมนู Project เลือก Edit Porject แล้วกำหนดชื่อไฟล์เป็น ชื่อที่เราบันทึกไว้ในข้อ 6. เพื่อ Add ไปยัง Project File

9. ไปที่เมนู Project เลือก Make Project เพื่อแอสเซมเบลอร์โปรแกรม

10. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

11. ต่อวงจรตามรูปที่ จ.2

12. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

.....

.....

.....

13. เปลี่ยนโปรแกรมจาก MOVLW 0xFF ในบรรทัดที่ ถัดจาก START เป็น MOVLW 0x01 ทำการคอมไพล์และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ใหม่อีกครั้ง บันทึกผลการทดลอง

.....

.....

.....

.....

.

13. ทดลองนำเอาเครื่องหมายที่อยู่หน้า MOVLW '00000000' ออกทำการคอมไพล์ใหม่อีกครั้งแล้วโปรแกรมลงในตัวไมโครคอนโทรลเลอร์บันทึกผลการทดลอง

.....

.....

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายการทำงานของพอร์ต A และ E มาพอเข้าใจ
2. จากโปรแกรมที่ จ.3 เมื่อต้องการเขียนโปรแกรมให้พอร์ต RA0 ดับส่วน RA2 และ RA3 ติดจะแทรกโปรแกรมที่ตำแหน่งไหนจงอธิบาย
3. เมื่อต้องการใช้งานพอร์ต A และ E ทำงานในโหมด PSP นั้นจะเขียนโปรแกรมอย่างไร
4. ถ้าต้องการกำหนดพอร์ต D เป็นอินพุตทั้งหมดจะเขียนโปรแกรมเพื่อกำหนดพอร์ตอย่างไร

ใบงานที่ 2
การทดลองใช้งานสัญญาณนาฬิกาของ PIC18F458

จุดประสงค์

1. เพื่อให้สามารถอธิบายวิธีป้อนสัญญาณนาฬิกาให้แก่ PIC18F458 ได้
2. เพื่อให้สามารถป้อนสัญญาณนาฬิกาให้แก่ไมโครคอนโทรลเลอร์ในโหมดต่าง ๆ
3. เพื่อให้สามารถแก้ไขโปรแกรมในการทดลองได้
4. เพื่อให้สามารถสรุปและวิจารณ์ผลการทดลองเทียบกับทฤษฎีได้

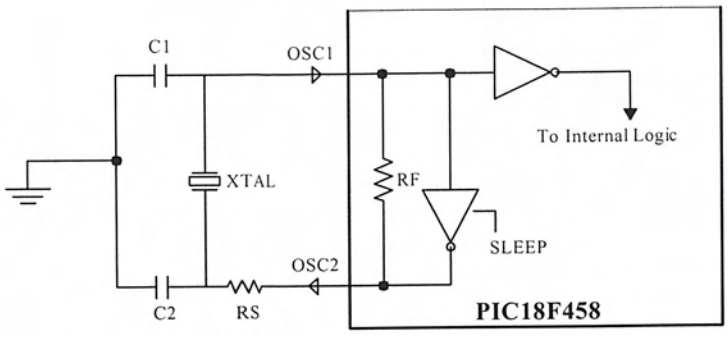
เครื่องมือและอุปกรณ์

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ EPIC win
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
5. โมดูลสวิตช์พื้นฐาน

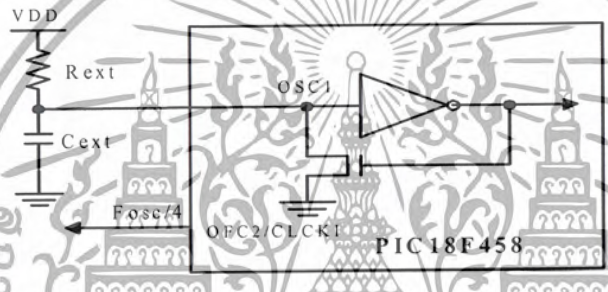
ทฤษฎีเบื้องต้น

การป้อนสัญญาณนาฬิกาโดยใช้คริสตอล

โหมดสัญญาณนาฬิกา LP, XT, HS และ HS4 จะมีลักษณะของการต่อเหมือนกัน แต่จะต่างกันตรงที่ชนิดของคริสตอลที่ใช้ ส่วนการต่อใช้งานแบบนี้จะไม่สามารถนำขา OSC2 ไปใช้งานเป็นขาอินพุตเอาต์พุตได้ถ้าหากต้องการนำขา OSC2 ไปใช้งานจะต้องใช้คริสตอลที่มีโมดูล คือ มีตัวเก็บประจุและตัวต้านทานอยู่ภายใน ซึ่งการใช้งานเพียงต่อไฟเลี้ยงให้เพียงเท่านั้นแล้วก็สามารถนำขา OSC2 ไปใช้งานได้โดยที่ขานี้จะมีความถี่ออกมาเป็นค่าความถี่ของ OSC1/4 ประโยชน์ก็คือสามารถนำขา OSC2 ไปใช้งานเป็นสัญญาณนาฬิกาให้กับอุปกรณ์อื่นๆ ได้



รูปที่ จ.4 การต่อคริสตอลแบบเรโซเนเตอร์



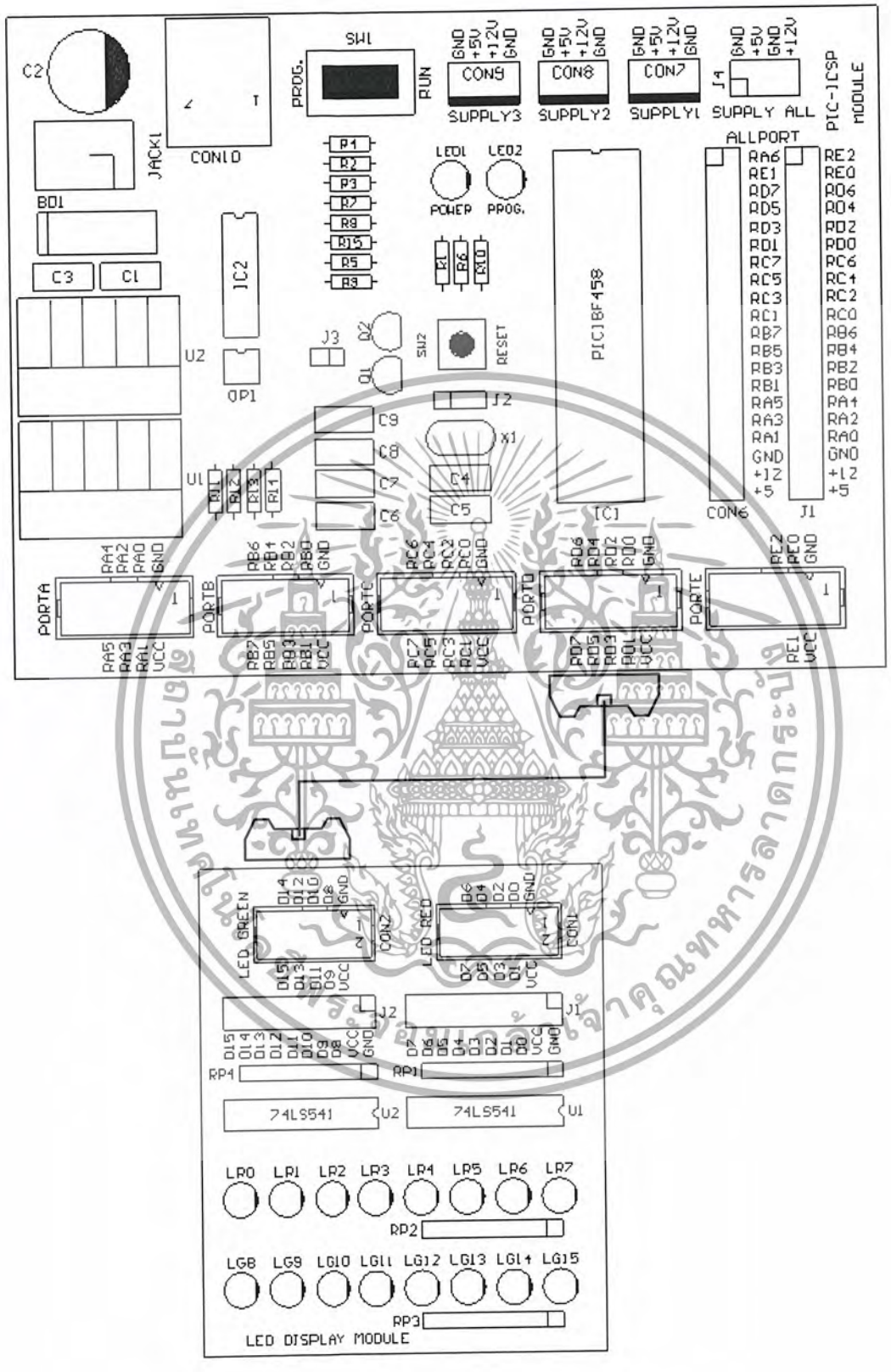
รูปที่ จ.5 การต่อออสซิลเลเตอร์แบบ RC

ตารางที่ จ.1 ค่าของตัวเก็บประจุที่ใช้กับคริสตอลออสซิลเลเตอร์แบบต่างๆ

ชนิดคริสตอลที่ใช้	ความถี่	C1	C2
LP	32.0 KHz	33 pF	33 pF
	200 KHz	15 pF	15 pF
XT	200 KHz	47-68 pF	47-68 pF
	1.0 MHz	15 pF	15 pF
	4.0 MHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0 MHz	15-33 pF	15-33 pF
	20.0 MHz	15-33 pF	15-33 pF
	25.0 MHz	15-33 pF	15-33 pF

โหมดการป้อนสัญญาณนาฬิกาโดยใช้ตัวต้านทานและตัวเก็บประจุ RC และ RCIO

การต่อออสซิลเลเตอร์ในโหมดนี้จะต่อใช้งานเพียงขาเดียวคือขา OSC1 ซึ่งการต่อใช้งานในโหมดนี้จะไม่เป็นที่นิยมเท่าไหร่ เนื่องด้วยการใช้งานโหมดนี้มีข้อเสียในเรื่องของอุณหภูมิซึ่งเมื่อไมโครคอนโทรลเลอร์ทำงานเป็นเวลานานๆ อาจทำให้ไมโครคอนโทรลเลอร์ทำงานผิดพลาดได้ โดยเฉพาะในงานที่ต้องการความเที่ยงตรงของเวลาการใช้ออสซิลเลเตอร์แบบนี้จะทำให้เกิดความคลาดเคลื่อนได้ แต่ข้อดีของการใช้งานออสซิลเลเตอร์แบบนี้คือประหยัดหาซื้อง่าย ส่วนในโหมดของ RCIO จะสามารถที่จะนำขา RA6 ไปใช้งานเป็นขาอินพุตเอาต์พุตปกติได้ การต่อใช้งานแสดงดังรูปที่ จ.4 ซึ่งจะมีตัวต้านทานและตัวเก็บประจุทำหน้าที่กำหนดค่าความถี่ โดยค่าตัวต้านทาน Rext ที่นำมาใช้นั้นจะอยู่ระหว่าง 4-100 กิโลโอห์มตัวเก็บประจุ Cext ที่ใช้งานได้จะต้องมากกว่า 20 pF



รูปที่ จ.5 การเชื่อมต่อระหว่างโมดูลหลักกับโมดูลแสดงผลแอลอีดี

```
;*****************************************************
; Program     : RC MODE
; Filename    : lab2.asm
; MCU  : PIC 18F458
;*****************************************************
            list p=18f458           ; list directive to define processor
      #include <p18f458.inc>    ; processor specific variable definitions

  OFFSET    EQU     0x20
  COUNT1    EQU     0x21
  COUNT2    EQU     0x22
  COUNT3    EQU     0x23
        ORG     0x0000

          CLRF      TRISD         ; PORTD is output
START     MOVLW  0X01
          MOVWF  OFFSET
LOOP      MOVFF   OFFSET,PORTD
          RLCF    OFFSET,F
          CALL    DELAY
          GOTO    LOOP

;******* Delay loop ***********
DELAY   MOVLW   0x03
        MOVWF   COUNT1
DEL     CLRF    COUNT2
DEL0    CLRF    COUNT3
DEL1    DECFSZ  COUNT3,F
        GOTO    DEL1
        DECFSZ  COUNT2,F
        GOTO    DEL0
        DECFSZ  COUNT1
        GOTO    DEL
        RETURN
        END
```

รูปที่ จ.6 โปรแกรมการทดลองใช้งานโหมดสัญญาณนาฬิกา PIC18F458

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ได้ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตาม รูปที่ จ.6 ทำการคอมไพล์ให้เป็น Hex File

2. ต่อสายเชื่อมต่อระหว่างโมดูลตามรูปที่ จ.5

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ เลือกโหมดสัญญาณนาฬิกาจากโปรแกรม Epic Win ให้เป็น RC

4. เลือกจัมเปอร์ที่ตำแหน่ง XT ให้เป็น RC ในโมดูลหลัก PIC-ICSP

5. รันโปรแกรมสังเกตผลที่เกิดขึ้นกับแอลอีดีทั้ง 8 ดวง บันทึกผลการทดลอง

.....

.....

.....

6. ใช้โปรแกรมในรูปที่ จ.6 คอมไพล์อีกครั้ง แต่ในการโปรแกรมเลือกโหมดสัญญาณนาฬิกาเป็น HS โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ อีกครั้ง

7. เลือกจัมเปอร์ที่ตำแหน่ง XT สังเกตผลที่เกิดขึ้นกับแอลอีดีทั้ง 8 ดวง บันทึกผลการทดลอง

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายวิธีการป้อนสัญญาณนาฬิกาแบบ RC และ XT ว่ามีวิธีการอย่างไร
2. จงบอกความแตกต่างของการต่อสัญญาณนาฬิกาแบบ RC และ XT
3. จงบอกข้อดีและข้อเสียของการป้อนสัญญาณนาฬิกาแต่ละแบบมาพอเข้าใจ

ใบงานที่ 3
การทดลองใช้งานหน่วยความจำข้อมูลอีอีพรอมภายใน PIC18F458

จุดประสงค์

1. เพื่อให้สามารถอธิบายหน้าที่ของรีจิสเตอร์ที่เกี่ยวข้องกับหน่วยความจำอีอีพรอมของ PIC18F458 ได้
2. เพื่อให้สามารถเขียนโปรแกรมใช้งานหน่วยความจำอีอีพรอมภายใน PIC18F458ได้
3. เพื่อให้สามารถแก้ไขดัดแปลงโปรแกรมจากการทดลองได้
4. เพื่อให้สามารถสรุปและวิจารณ์ผลการทดลองเทียบกับทฤษฎีได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ Epic Win
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
5. โมดูลแสดงผลแอลอีดี

ทฤษฎีเบื้องต้น

หน่วยความจำข้อมูลของ PIC18F458 มีขนาด 256 ไบต์ ผู้ใช้งานสามารถนำข้อมูลเข้าไปเก็บและสามารถอ่านค่าออกมาได้ในการติดต่อกับหน่วยความจำข้อมูลนั้นจะมีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 4 ตัวดังนี้

1. EECON1 รีจิเตอร์ควบคุมการเข้าถึงหน่วยความจำ
2. EECON2 รีจิสเตอร์จัดลำดับการเขียนข้อมูลในหน่วยความจำ
3. EEDATA รีจิสเตอร์บัฟเฟอร์ข้อมูลสำหรับการอ่านและเขียน
4. EEADR รีจิสเตอร์แอดเดรส

ในการติดต่อกับหน่วยความจำข้อมูลอีอีพรอมจะใช้รีจิสเตอร์ 4 ตัวคือ EECON1, EECON2, EEDATA และ EEADR โดยทำการเลือกแอดเดรสของหน่วยความจำที่ต้องการติดต่อ

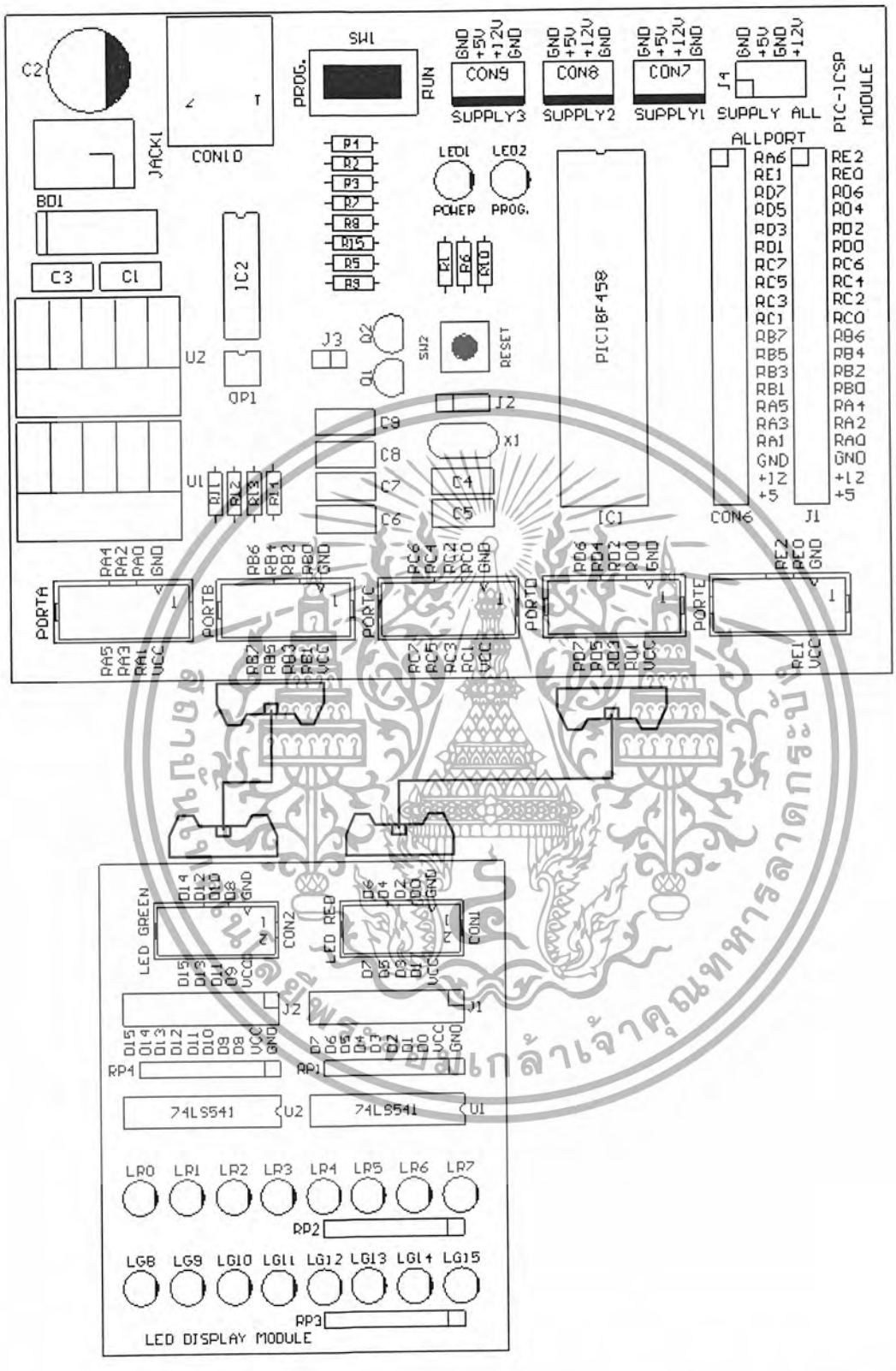
ผ่านทาง EEADR จากนั้นใช้ EECON1 และ EECON2 เพื่อกำหนดจุดประสงค์ในการติดต่อกับหน่วยความจำว่า ต้องการอ่านหรือเขียน โดยข้อมูลที่ทำการอ่านหรือเขียนจะบรรจุอยู่ในรีจิสเตอร์ EEDATA หน่วยความจำข้อมูลอีอีพรอมนี้สามารถทำการลบและเขียนใหม่ได้ถึง 100,000 รอบระยะเวลาในการเขียนข้อมูลลงในหน่วยความจำอีอีพรอมนั้นจะควบคุมโดยตัวตั้งเวลาหรือไทเมอร์ภายในไมโครคอนโทรลเลอร์ โดยระยะเวลาในการเขียนนั้นจะขึ้นอยู่กับแรงดันและอุณหภูมิ ในขณะนั้นในการใช้งานหน่วยความจำจะมี 2 รูปแบบ คือการอ่านและการเขียน

การอ่านหน่วยความจำข้อมูลอีอีพรอม

เริ่มต้นด้วยการกำหนดแอดเดรสที่ต้องการอ่านลงในรีจิสเตอร์ EEADR เคลียร์บิต EEPGD ในรีจิสเตอร์ EECON และเซตบิต RD ในรีจิสเตอร์ EECON1 ข้อมูลจากหน่วยความจำจะได้รับการอ่านออกมาในไซเคิลการทำงานของคำสั่งถัดไป โดยข้อมูลที่อ่านออกมานี้จะบรรจุอยู่ในรีจิสเตอร์ EEDATA และรักษาข้อมูลนี้ไว้จนกว่าจะเกิดการอ่านหรือเขียนข้อมูลใหม่ขึ้น

การเขียนข้อมูลลงในหน่วยความจำข้อมูลอีอีพรอม

มีกระบวนการคล้ายคลึงกับ PIC16F84 แตกต่างกันตรงที่ตำแหน่งแอดเดรสของรีจิสเตอร์ที่ใช้งาน เริ่มต้นกระบวนการเขียนด้วยการกำหนดแอดเดรสของหน่วยความจำไปยังรีจิสเตอร์ EEADR ต่อด้วยเขียนข้อมูลที่ต้องการไปยังรีจิสเตอร์ EEDATA เคลียร์บิต EEPGD ในรีจิสเตอร์ EECON1 เพื่อเลือกการติดต่อหน่วยความจำข้อมูลอีอีพรอม เซตบิต WREN เพื่ออีนาเบิลการเขียนข้อมูล ดิสเอเบิลการอินเตอร์รัพต์ทุกรูปแบบ



รูปที่ จ.7 การเชื่อมต่อโมดูลหลัก PIC-ICSP และโมดูลแสดงผลแอลอีดี

```
;*************************************************************
; Filename : lab3.asm
; Descripton :          EEPROM read & writing
; MCU :                 PIC18F458
;*************************************************************
          LIST      P=18F458
#include <p18F458.INC>
          ORG       0x000
;*********Init Port************
          CLRF      TRISD
          CLRF      PORTD
          MOVLW     0X01
          MOVWF     PORTB
          CLRF      PORTB
;***********WRITE**************
WRITE     MOVLW     0x20                ; Assign address
          MOVWF     EEADR
          MOVLW     0XFF                ; Assign data
          MOVWF     EEDATA
          BCF       EECON1,EEPGD        ; Data EEPROM section
          BSF       EECON1,WREN         ; Write enable
;******** Write sequence**********
          MOVLW     0x55
          MOVWF     EECON2
          MOVLW     0xAA
          MOVWF     EECON2
          BSF       EECON1,WR           ; Writing start
          BTFSC     EECON1,WR           ; Skip if completed
          GOTO      $-1         ; WAIT
          BCF       EECON1,WREN         ; Disable writing
;***********READ*****************
          BSF       PORTB,0   ; Turn-on LED
READ      CLRF      EEADR     ; Address = 0x00
          BCF       EECON1,EEPGD        ; Data EEPROM  section
          BSF       EECON1,RD           ; Read enable
          MOVF      EEDATA,W            ; Read data
          MOVWF     PORTD               ; Show data
          GOTO      READ
          RETURN
          END
```

รูปที่ จ.8 โปรแกรมการทดลองอ่านและเขียนหน่วยความจำข้อมูลอีอีพรอม

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ได้ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.8 ทำการคอมไพล์ให้เป็น Hex File

2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ด้วยโปรแกรม Epic Win

3. ต่อวงจรตามรูปที่ จ.7

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

.....

.....

.....

5. ลองเปลี่ยนโปรแกรมจาก MOVLW 0xFF เป็น MOVLW 0x03 จากนั้นทำการคอมไพล์และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้ง

6. จากการทดลองข้อ 5 เมื่อรันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลการทดลอง

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. รีจิสเตอร์ที่เกี่ยวข้องกับการอ่านและเขียนหน่วยความจำข้อมูลอีอีพอรมมีกี่ตัวแต่ละตัวมีหน้าที่อะไรบ้างจงอธิบายและการทำงานอย่างไรจงอธิบาย

2. เมื่อต้องการเขียนโปรแกรมให้เขียนข้อมูล 0x07 ไปยังหน่วยความจำตำแหน่ง 0x70 จากนั้นทำการอ่านออกมาแสดงผลยังแอลอีดีจะแทรกโปรแกรมตรงไหนหรือเขียนโปรแกรมอย่างไรจงอธิบาย

ใบงานที่ 4
การทดลองใช้งานโมดูลสร้างแรงดันอ้างอิงของ PIC18F458

จุดประสงค์การทดลอง

1. เพื่อให้สามารถอธิบายการหลักการสร้างแรงดันอ้างอิงภายใน PIC18F458
2. เพื่อให้สามารถเขียน โปรแกรมใช้งานการ โมดูลสร้างแรงดันภายใน PIC18F458
3. เพื่อให้สามารถแก้ไข โปรแกรมในการทดลองได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ EPIC win
2. สายเชื่อมต่อระหว่าง โมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่าง โมดูล
4. โมดูลแสดงผลแอลอีดี
5. โมดูลหลัก PIC-ICSP
6. โมดูลสวิตช์พื้นฐาน

ทฤษฎีเบื้องต้น

โมดูลสร้างแรงดันอ้างอิง

ในไมโครคอนโทรลเลอร์ PIC18F458 มีโมดูลที่ทำงานเกี่ยวกับสัญญาณแอนะลอก หนึ่งส่วนคือ โมดูลสร้างแรงดันอ้างอิง (Voltage Reference Module) โดยภายในโมดูลนี้ ประกอบด้วยตัวต้านทาน 2 ค่าที่ต่อร่วมกันในลักษณะคล้ายขั้นบันได (Ladder) 16 จุด ต่อโดยต่อเข้ากับ วงจรแอนะลอกมัลติเพล็กซ์แบบเข้า 16 ออก 1 ค่าของตัวต้านทานในโมดูลนี้มีความแตกต่างกัน 8 เท่า คือ R และ 8R ค่าตัวต้านทาน R กำหนดให้เท่ากับ 2 โอห์ม ค่าแรงดันเอาต์พุตและการทำงาน ทั้งหมดของโมดูลนี้ขึ้นอยู่กับรีจิสเตอร์ CVRCON แรงดันอ้างอิงที่สร้างขึ้นนี้จะส่งออกทางขาพอร์ต RA2 / CVREF สามารถเลือกได้ 2 ย่านคือ 0 ถึง $0.75CV_{RSRC}$ และ $0.25CV_{RSRC}$ ถึง $0.75CV_{RSRC}$ โดยที่ CVR_{RSRC} คือ แรงดันไฟเลี้ยงของส่วนอ้างอิงการเปรียบเทียบ (Comparator Reference Supply Voltage) ซึ่งต่อตรงเข้ากับ V_{DD} หรืออาจกล่าวได้ง่ายๆ ว่าแรงดันอ้างอิงสร้างขึ้นมีค่า 0-0.75

V_{DD} และ 0.25 V_{DD} - 0.75 V_D ขึ้นอยู่กับการกำหนดที่รีจิสเตอร์ CVRCON แรงดันอ้างอิงที่ได้จากโมดูลนี้สามารถนำไปเชื่อมต่อกับอุปกรณ์ภายนอกได้หรือใช้งานร่วมกับโมดูลเปรียบเทียบแรงดันแอนะลอกของ PIC18F458 เมื่อกำหนดให้ทำงานในโหมดวงจรเปรียบเทียบ 2 ชุดใช้ แรงดันอ้างอิงภายในร่วมกัน (CM2-CM0 : 110) CORCON รีจิสเตอร์ควบคุมการทำงานของโมดูลสร้างแรงดันอ้างอิงสำหรับโมดูลเปรียบเทียบแรงดันเป็นรีจิสเตอร์ขนาด 8 บิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต0
CVREN	CVROE	CVERR	-	CVR3	CVR2	CVR1	CVR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

รูปที่ จ.9 รีจิสเตอร์ CVRCON

CVREN (Comparator Voltage Reference Enable Bit : บิต 7) บิตเลือกการทำงานของโมดูลสร้างแรงดันอ้างอิง

"0" - ปิดการทำงานของโมดูลนี้

"1" – จ่ายแรงดันให้แก่โมดูลสร้างแรงดันหรือเอ็นเอเบิลให้โมดูลสร้างแรงดันอ้างอิงนี้ทำงาน

CVROE (Comparator V_{REF} Output Enable Bit : บิต 6) บิตกำหนดให้ส่งค่าแรงดันออกทางเอาต์พุต

"0" ปลดการติดต่อระหว่างโมดูลสร้างแรงดันอ้างอิงกับขาพอร์ต RA2

"1" กำหนดให้จ่ายแรงดันอ้างอิงออกทองขาพอร์ต RA2/AN2/V_{REF}/CV_{REF}

CVRR (Comparator V_{REF} Range Selection bit : บิต 5) บิตเลือกย่านแรงดันอ้างอิง

"0" - เลือกแรงดันอ้างอิงย่าน 0.25CV_{RSRC}-0.75 CV_{RSRC} มีระดับการเปลี่ยนแปลงค่าเท่ากับ CV_{RSRC} /32 ต่อระดับ

"1" - เลือกแรงดันอ้างอิงย่าน 0.075 CV_{RSRC} มีระดับการเปลี่ยนแปลงค่าเท่ากับ CV_{RSRC}/24 ต่อระดับ

บิต 4 : ไม่มีการใช้งาน อ่านค่าเป็น "0"

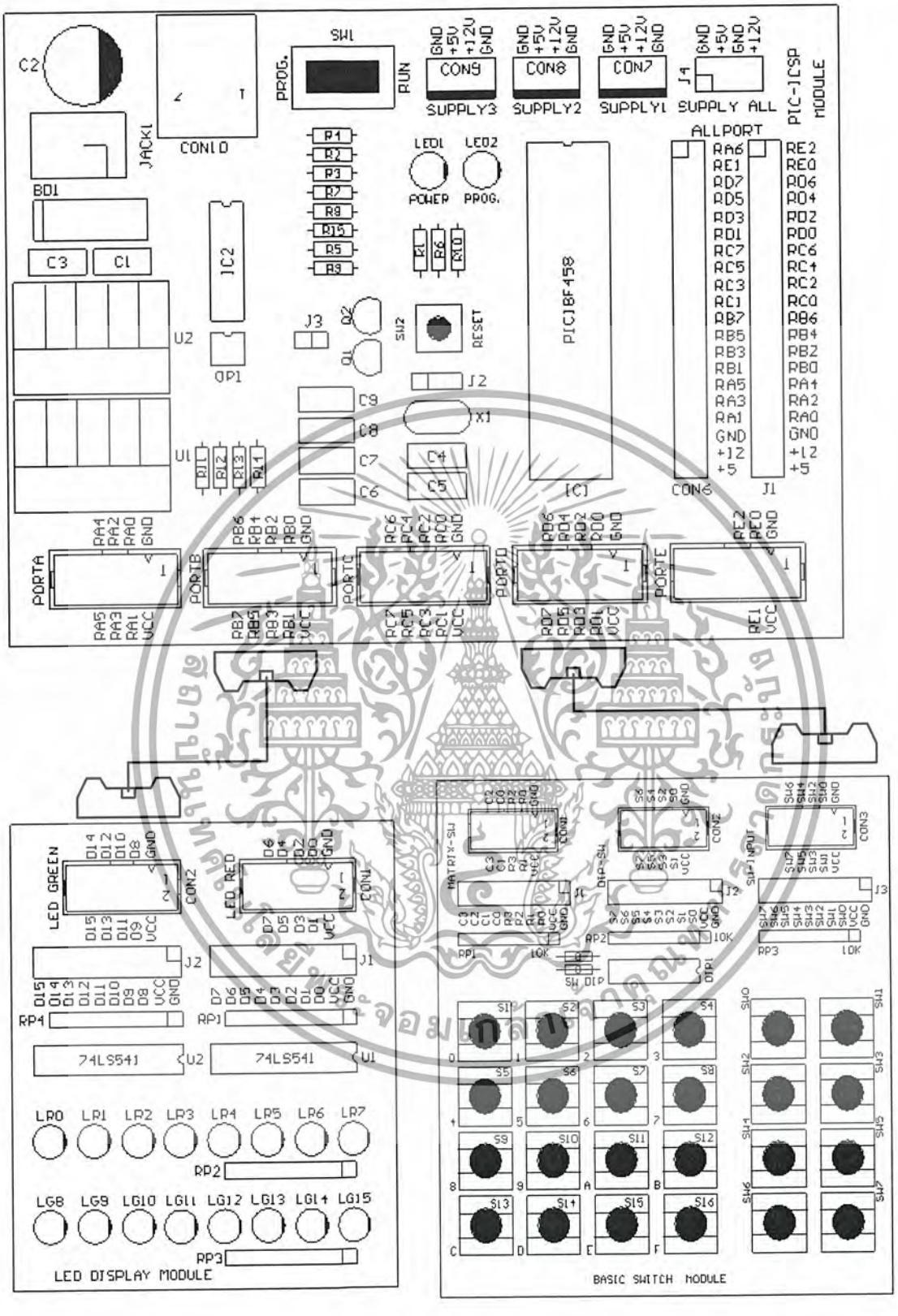
CVR3 : CVR0 (Comparator V_{REF} Value Selection : บิต 3-0) บิตเลือกค่าแรงดันอ้างอิง แรงดันอ้างอิงที่เกิดขึ้นของโมดูลนี้จะขึ้นอยู่กับการกำหนดค่าขอบบิต V_R ทั้ง 4 บิตร่วมกับค่าของบิต V_{RR} โดยค่าของบิต V_R ทั้ง 4 บิตจะเกิดเป็นค่าของเลขฐานสิบได้ตั้งแต่

0-15 (เกิดจาก 0000-1111 ในรูปแบบเลขฐานสอง) ดังนั้นค่าของแรงดันอ้างอิงที่เกิดขึ้นสามารถคำนวณได้จาก

(ก) ในกรณีบิต CVRR เท่ากับ "0" แรงดันอ้างอิงเท่ากับ $¼CV_{RSRC} + (VR_{10}/32xCV_{RSRC}$

(ข) ในการณีบิต CVRR เท่ากับ "1" แรงดันอ้างอิงเท่ากับ $VR_{10}/24xCV_{RSRC}$

โดยที่ VR_{10} คือ ค่าของบิต CVR3-CVR0 ในรูปของเลขฐานสิบ



รูปที่ จ.10 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับโมดูลสวิตช์พื้นฐานแสดงผลที่โมดูลแสดงผลแอลอีดี

```
;*************************************************************
; Filename :        lab4.asm
; Descripton :      Demonstrate voltage refernce in PIC18F458
; MCU :             18f458
;*************************************************************
        LIST    P=18f458
#INCLUDE        <P18f458.INC>
COUNT   EQU     0x20
        ORG     0x0000
        MOVLW   B'00001111'         ; RB7:RB4 = output,RD3:RD0 = input
        MOVWF   TRISD
        MOVLW   B'00001111'
        MOVWF   TRISB
        CLRF    TRISB
        CLRF    PORTD
        BSF     CVRCON,CVRR     ; Select low range
        CALL    VREF_1
SW_RANGE BTFSS  PORTB,0         ; Range 1
        CALL    VREF_1
        BTFSS   PORTD,1         ; Range 2
        CALL    VREF_2
        BTFSS   PORTD,2         ; Range 3
        CALL    VREF_3
        BTFSS   PORTD,3         ; Range 4
        CALL    VREF_4
        GOTO    SW_RANGE

VREF_1  MOVLW   B'00010000'
        MOVWF   PORTB
        MOVLW   B'11100110'
        GOTO    VREF_ENABLE

VREF_2  MOVLW   B'00100000'
        MOVWF   PORTB
        MOVLW   B'11101010'
        GOTO    VREF_ENABLE

VREF_3  MOVLW   B'01000000'
        MOVWF   PORTB
        MOVLW   B'11101101'
```

```
                GOTO    VREF_ENABLE
        VREF_4  MOVLW   B'10000000'
                MOVWF   PORTB
                MOVLW   B'11101111'
                GOTO    VREF_ENABLE
VREF_ENABLE     NOP
                MOVWF   CVRCON
DELAY10US       MOVLW   .17
                MOVWF   COUNT
                DECFSZ  COUNT,1
                GOTO    $-1
                RETURN

                END
```

รูปที่ จ.11 โปรแกรมใช้งานโมดูลสร้างแรงดันอ้างอิงของ PIC18F458

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.11 เสร็จแล้วทำการคอมไพล์ให้เป็น Hex file

2 ต่อสายเชื่อมต่อระหว่างโมดูลตามรูปที่ จ.10

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

4. รันโปรแกรมทดลองกด SW0-SW3 ที่โมดูลแอลอีดีสังเกตผลที่เกิดขึ้นกัลแอลอีดีทั้ง 4 ดวงที่ต่ออยู่กับ RD4-RD7

5. เลือกแรงดันเอาต์พุตจากสวิตช์ SW0-SW2 โดยทำการกดเลือกย่านความถี่ เมื่อกดสวิตช์ตามย่านความถี่ต่างๆ สังเกตผลที่เกิดขึ้นบันทึกผล

.....

.....

.....

.....

6. วัดค่าแรงดันที่ขา RA2/AN2 เทียบกับแรงดันเอาต์พุตที่เลือก บันทึกผลในตารางที่ จ.2

ตารางที่ จ.2 ผลการทดลองการกดสวิตช์ S1-S2

สวิตช์	แรงดันอ้างอิงของโมดูลสร้างแรงดันอ้างอิง	
	ค่าที่คำนวณได้	ค่าที่วัดได้
S1		
S2		
S3		
S4		

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงบอกรีจิสเตอร์ที่เกี่ยวข้องกับโมดูลสร้างแรงดันอ้างอิงภายใน PIC18F458 มีอะไรบ้าง พร้อมอธิบายหลักการทำงานมาพอเข้าใจ

2. จงบอกการนำไปใช้งานของโมดูลสร้างแรงดันอ้างอิงของภายใน PIC18F458

3. เมื่อต้องการใช้แรงดันอ้างอิงที่ขา RA2 เป็นศูนย์จะเขียนโปรแกรมอย่างไร

ใบงานที่ 5
การทดลองใช้งานโมดูลการแปลงสัญญาณแอนะลอกเป็นดิจิตอล ภายใน PIC18F458

จุดประสงค์

1. เพื่อให้สามารถอธิบายหลักการแปลงสัญญาณแอนะลอกเป็นดิจิตอลภายใน PIC18F458 ได้
2. เพื่อให้สามารถเขียนโปรแกรมใช้งานการแปลงสัญญาณแอนะลอกเป็นดิจิตอลภายใน PIC18F458 ได้
3. เพื่อให้สามารถแก้ไขโปรแกรมในการทดลองได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ Epic Win
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลแสดงผลแอลอีดี
5. ตัวต้านทานปรับค่าได้ 10 กิโลโอห์ม

ทฤษฎีเบื้องต้น

โมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอลของ PIC18F458

อีกหนึ่งโมดูลที่สำคัญที่ไมโครคอนโทรลเลอร์สมัยใหม่จะต้องมี คือ โมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอล และสำหรับไมโครคอนโทรลเลอร์ PIC18F458 มีความละเอียดที่ 10 บิต 8 ช่องอินพุตสำหรับรุ่น 28 ขา และ 8 ช่องอินพุตสำหรับรุ่น 40 ขา โดยขาพอร์ตที่ใช้งานร่วมด้วยคือ ขาพอร์ต RA0-RA3, RA5 และ RE-RE2 (เฉพาะในรุ่น 40 ขา) การทำงานเพื่อรองรับอินพุตจำนวนมากจะใช้วิธีการมัลติเพล็กซ์ ซึ่งควบคุมด้วยกระบวนการทางซอฟต์แวร์ สำหรับการทำงานของโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอลใน PIC18F458 เป็นแบบซักเซสซีฟแอปพร็อกซิเมชั่น รีจิสเตอร์ที่เกี่ยวข้อง

มีทั้งสิ้น 4 ตัว โดยแบ่งเป็นรีจิสเตอร์ควบคุมการทำงาน 2 ตัว คือ ADCON0 และ ADCON1 ส่วนอีก 2 ตัวคือรีจิสเตอร์ ADRESH และ ADRESL ซึ่งใช้ในการเก็บผลลัพธ์ของการแปลงสัญญาณ โดย ADRESH รีจิสเตอร์ทั้ง 2 มีขนาดตัวละ 8 บิต ต้องทำงานร่วมกันเพื่อรองรับข้อมูลดิจิตอล 10 บิต ที่ได้จากการแปลงสัญญาณแอนะลอกรีจิสเตอร์ ADCON0 มีขนาด 8 บิต เป็นรีจิสเตอร์หลักที่ใช้ในการควบคุมการทำงานของโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอล รายละเอียดการใช้งานมีดังนี้

ตารางที่ จ.3 รีจิสเตอร์ ADCON0

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	-	ADON
R/W –0	R/W –0	R/W –0	R/W –0	R/W –0	R/W –0	R/W -0	R/W –0

ADCS1, ADCS0 (A/D Conversion Clock Select Bites -บิต 7 และ บิต 6) : บิตเลือกความถี่ของสัญญาณนาฬิกาที่ใช้ในการแปลงสัญญาณแอนะลอกเป็นดิจิตอล

"00" - ความถี่สัญญาณนาฬิกาหาร 2 (FOSC/2)

"01" - ความถี่สัญญาณนาฬิกาหาร 8 (FOSC/8)

"10" - ความถี่สัญญาณนาฬิกาหาร 32 (FOSC/32)

"11" - ใช้ความถี่สัญญาณนาฬิกาจากวงจร RC (FRC)

นอกจากนี้ยังใช้ร่วมกับบิต ADCS2 (บิต 6 ในรีจิสเตอร์ ADCS1) เฉพาะในอนุกรม PIC18F458 เพื่อเลือกความถี่สัญญาณนาฬิกาได้มากขึ้น

CHS2, CHS1, CHS0 (Analog Channel Select Bits -บิต 5, 4 และ บิต 3) : บิตเลือกสัญญาณแอนะลอก

"000" - ช่อง 0 (AN0/AR0)

"001" - ช่อง 1 (AN1/AR1)

"010" - ช่อง 2 (AN2/AR2)

"011" - ช่อง 3 (AN3/AR3)

"100" - ช่อง 4 (AN4/AR5)

"101" - ช่อง 5 (AN5/AE0) (ไม่มีในรุ่น 28 ขา)

"110" - ช่อง 6 (AN6/AE1) (ไม่มีในรุ่น 28 ขา)

"111" - ช่อง 7 (AN7/AR2) (ไม่มีในรุ่น 28 ขา)

GO/DONE (A/D Conversion Status Bite -บิต 2) : บิตแสดงสถานการณ์แปลง ทำงานร่วมกับบิต ACON กรณีบิต ADON เป็น "1"

"0" - การแปลงสัญญาณเสร็จสมบูรณ์ หรือยังไม่เริ่มการแปลงสัญญาณ

"1" - ยังอยู่ในระหว่างการแปลงสัญญาณ

บิตนี้สามารถเคลียร์ด้วยกระบวนการทางฮาร์ดแวร์ 2 ลักษณะคือ เมื่อการเปลี่ยนแปลงเสร็จสมบูรณ์เราจะเคลียร์เองอัตโนมัติ และเคลียร์เนื่องจากการเกิด เพาเวอร์ออนรีเซต

บิต 1 ไม่ใช้งานกำหนดเป็น "0"

ADON (A/D On Bit - บิต 0) : บิตเปิดการทำงานของโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอล

"0" - ปิดการทำงาน

"1" - เปิดการทำงาน

รีจิสเตอร์ ADCON1

เป็นรีจิสเตอร์ควบคุมการทำงานของโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอลที่ต้องทำงานร่วมกับ ADCON0 มีขนาด 8 บิต โดยรีจิสเตอร์ตัวนี้ใช้กำหนดการทำงานของขาพอร์ตที่เกี่ยวข้องกับโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอล และเลือกใช้รูปแบบของข้อมูลผลลัพธ์ที่ได้จากการแปลงสัญญาณรายละเอียดการใช้งานมีดังนี้

ตารางที่ จ.4 รีจิสเตอร์ ADCON 1

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
ADFM	ADCS2		-	PCFG3	PCFG2	PCFG1	PCFG0
R/W –0				R/W –0	R/W –0	R/W -0	R/W –0

ADFM(A/D Result Format Select -บิต 7) : บิตเลือกรูปแบบผลลัพธ์ของการแปลงสัญญาณ

"0" - เลือกผลลัพธ์แบบชิดซ้าย

"1" - เลือกผลลัพธ์แบบชิดขวา มีรูปแบบข้อมูลดังนี้

ADCS2 (A/D Conversion Clock Select Bites -บิต 6) : บิตเลือกความถี่ของสัญญาณนาฬิกาที่ใช้ในการแปลงสัญญาณแอนะลอกเป็นดิจิตอล บิตนี้มีเฉพาะในอนุกรม PIC18F458 ต้องใช้ร่วมกับ ADCS1 และ ADCS0 ในรีจิสเตอร์ ADCON0

บิต 5 และ บิต 4 ไม่ใช้งานกำหนดเป็น "0"

PCFG3, CFG2, CFG1, CFG0 (A/D Port Configuration Control Bits - บิต 3, 2, 1 และ 0) : บิตกำหนดการทำงานของพอร์ตที่ใช้ในโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอล

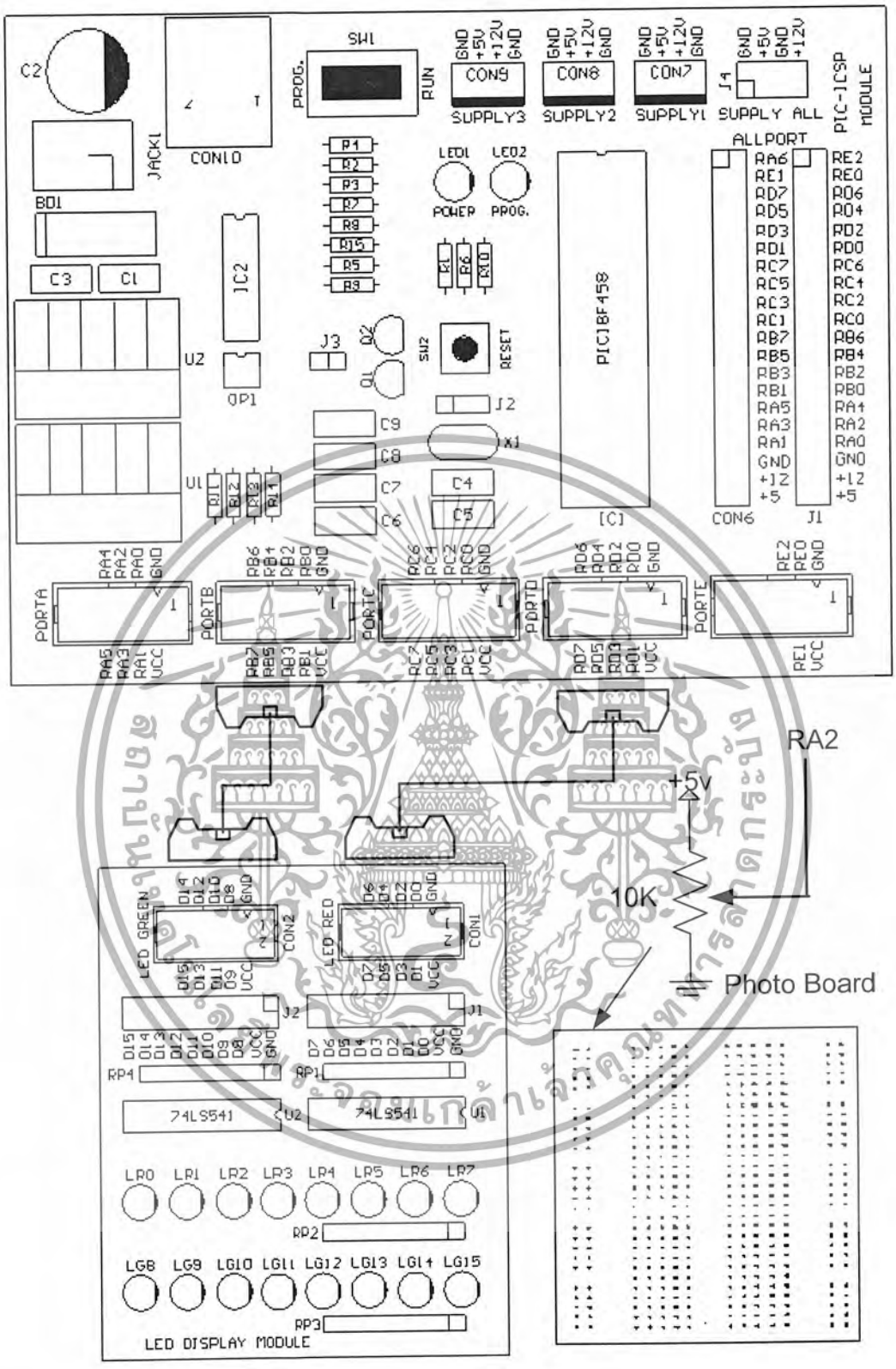
ตารางที่ จ.5 การกำหนดการทำงานของบิต ADCS2- ADCS1

ADCS2	ADCS1	ADCS1	ความถี่ของสัญญาณนาฬิกา
0	0	0	ความถี่สัญญาณนาฬิกาหาร 2 (FF_{OSC} /2)
0	0	1	ความถี่สัญญาณนาฬิกาหาร 8 (FF_{OSC} /8)
0	1	0	ความถี่สัญญาณนาฬิกาหาร 32 (FF_{OSC} /32)
0	1	1	ใช้ความถี่สัญญาณนาฬิกาจากวงจร RC (F_{RC})
1	0	0	ความถี่สัญญาณนาฬิกาหาร 4 (F_{OSC} /4)
1	0	1	ความถี่สัญญาณนาฬิกาหาร 16 (F_{OSC} /16)
1	1	0	ความถี่สัญญาณนาฬิกาหาร 64 (F_{OSC}/64)
1	1	1	ใช้ความถี่สัญญาณนาฬิกาจากวงจร RC (F_{RC})

ตารางที่ จ.6 การทำงานของพอร์ตที่ใช้ในโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอล

PCFG3 :PCFG 2	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	
0000	A	A	A	A	AN6	A	A	A	VDD	VSS	8:0
0001	A	A	A	A	VREF +	A	A	A	AN3	VSS	8:1
0010	D	D	D	A	AN6	A	A	A	VDD	VSS	8:0
0011	D	D	D	A	VREF +	D	A	A	AN3	VSS	8:1
0100	D	D	D	D	D	D	A	A	VDD	VSS	8:0
0101	D	D	D	D	VREF +	D	A	A	AN3	VSS	8:1
011x	D	D	D	D	D	D	D	D	-	-	8:0
1000	A	A	A	A	VREF +	VREF-	A	A	AN3	AN2	8:2
1001	D	D	A	A	A	A	A	A	VDD	VSS	8:0
1010	D	D	A	A	VREF +	VREF-	A	A	AN3	VSS	8:1
1011	D	D	A	A	VREF +	VREF-	A	A	AN3	AN2	8:2
1100	D	D	D	A	VREF +	VREF-	A	A	AN3	AN2	8:2
1101	D	D	D	D	VREF +	VREF-	A	A	AN3	AN2	8:2
1110	D	D	D	D	D	D	D	A	VDD	VSS	8:0
1111	D	D	D	D	VREF +	VREF-	D	A	VREF-	AN2	8:2

หมายเหตุ (A) = อินพุตแอนะลอก, (D) = อินพุตดิจิตอล, (VREF+) = แรงดันอ้างอิงขาบวก, (VREF-) = แรงดันอ้าอิงลบ , (AN2) = ขาอินพุตแอนะลอกช่อง 2 , (AN3) = ขาอินพุตแอนะลอกช่อง 3, (VDD) = ไฟเลี้ยง , VSS คือ กราวด์



รูปที่ จ.12 การเชื่อมต่อสายระหว่างโมดูลหลัก PIC-ICSP กับโมดูลแสดงผลแอลอีดี

```
;*************************************************************
; Filename :            lab5.asm
; Descripton :          A2D Convertor
; MCU :                 PIC18F458

;*************************************************************
            list        p=18F458
#include <p18F458.INC>
            ORG         0x0000
;*********Init Port************
;1          CLRF        ADCON1
            MOVLW       10000000
            MOVWF       ADCON1
            CLRF        TRISD
            CLRF        TRISB
7;          MOVLW       10000001
            MOVWF       ADCON0
START       BSF         ADCON0,2
WAIT        BTFSC       ADCON0,2
            GOTO        WAIT
            MOVF        ADRESL,W
            MOVWF       PORTD
            MOVF        ADRESH,W
            MOVWF       PORTB
            GOTO        START
            END
```

รูปที่ จ.13 โปรแกรมการทดลองใช้งานโมดูลการแปลงสัญญาณแอนะลอกเป็นดิจิตอลภายใน PIC18F458

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.13 เสร็จแล้วทำการคอมไพล์ให้เป็น Hex File

2. ต่อสายเชื่อมต่อระหว่างโมดูลตามรูปที่ จ.12

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

4. ทำการปรับค่าความต้านทานที่ต่ออยู่กับ RA5 สังเกตการเปลี่ยนแปลงที่เกิดขึ้น

5. จากข้อ 4 ผลการทดลองเป็นดังนี้

.....

.....

.....

6. ทำการเปลี่ยนโปรแกรมจาก movlw 10100001 เป็น 10000001 ทำการคอมไพล์และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ใหม่อีกครั้ง

7. รันโปรแกรม ทดลองปรับค่าความต้านทานที่ต่ออยู่กับ RA5/AN5 ผลที่ได้เป็นดังนี้

.....

.....

.....

8. ทดลองปรับค่าความต้านทานที่ต่ออยู่กับ RA2/AN2 สังเกตผลที่เกิดขึ้นกับโมดูลแอลอีดี บันทึกผลการทดลอง

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายหลักการในการแปลงสัญญาณแอนะลอกเป็นดิจิตอลของไมโคร-คอนโทรลเลอร์ PIC18F458 มาพอเข้าใจ

2. จงบอกรีจิสเตอร์ที่เกี่ยวข้องกับการแปลงสัญญาณแอนะลอกเป็นดิจิตอลของไมโคร-คอนโทรลเลอร์ PIC18F458 พร้อมอธิบายหน้าที่ในการทำงานมาพอเข้าใจ

3. เมื่อต้องการให้ขา RA3 รับสัญญาณแอนะลอกอินพุตจะเขียนโปรแกรมอย่างไร

ใบงานที่ 6
การทดลองใช้งานการอินเตอร์รัพต์ของ PIC18F458

วัตถุประสงค์

1. เพื่อให้สามารถอธิบายหลักการทำงานของการอินเตอร์รัพต์ของ PIC18F458 ได้
2. เพื่อให้สามารถสรุปและวิจารณ์ผลการทดลองเทียบกับทฤษฎีได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ EPICwin
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
6. โมดูลแสดงผลแอลอีดี

ทฤษฎีเบื้องต้น

การอินเตอร์รัพต์ของไมโครคอนโทรลเลอร์ PIC18F458

การอินเตอร์รัพต์ (Interrupt) หรือการขัดจังหวะการทำงานของซีพียูนับเป็นคุณสมบัติที่ต้องมีในไมโครคอนโทรลเลอร์สมัยใหม่ และเป็นคุณสมบัติที่มีบทบาทสำคัญอย่างมากเมื่อนำไมโครคอนโทรลเลอร์มาสร้างระบบควบคุมอัตโนมัติ สำหรับไมโครคอนโทรลเลอร์ PIC18F458 สามารถกำเนิดและตอบสนองการเกิดอินเตอร์รัพต์ได้ถึง 21 แหล่งพื้นที่ๆ ใช้เก็บค่าอินเตอร์รัพต์เวกเตอร์มีอยู่ 2 ส่วน คือ 0008h และ 0018h สำหรับส่วนนี้จะเป็นส่วนของพื้นที่ของการบริการการอินเตอร์รัพต์โดยการอินเตอร์รัพต์ระดับความสำคัญสูงจะอยู่ที่แอดเดรส 0008h และระดับการอินเตอร์รัพต์สำคัญต่ำจะอยู่ที่แอดเดรส 0018h โดยถ้าเงื่อนไขของการเกิดอินเตอร์รัพต์เป็นจริงจะมีการเซตแฟลกของการอินเตอร์รัพต์นั้นๆ ขึ้น (ชื่อของแฟลกจะลงท้ายด้วยตัวอักษร F โดยอาจกล่าวได้ว่าเป็นบิตร้องขอการอินเตอร์รัพต์) จากนั้นจะตรวจสอบว่า มีการเอ็นเอเบิลการอินเตอร์รัพต์นั้นหรือไม่โดยดูจากส่วนที่ลงท้ายด้วย E โดยอาจกล่าวได้ว่าเป็นส่วนที่ปิดเปิดการอินเตอร์รัพต์นั่นเอง

เช่นถ้ามี สัญญาณร้องขอเข้ามาที่ขา RB0/INT แฟลก INTF จะเซต และถ้าหากมีการเอ็นเอเบิลการอินเตอร์รัพต์แบบนี้ ซึ่งตรวจสอบจากบิต INTE และมีการเอ็นเอเบิลการอินเตอร์รัพต์รวมไว้ก็จะทำเกิดการอินเตอร์รัพต์ขึ้นในระบบโดยในส่วนของการเอ็นเอเบิลการอินเตอร์รัพต์รวมจะกระทำที่บิต GIE ของรีจิสเตอร์ INTCON บิตที่ 7 โดยการเซตค่าให้บิตนี้เป็น 1 หากต้องการให้มีการอินเตอร์รัพต์เกิดขึ้นต่อมาหากมีการให้เกิดการอินเตอร์รัพต์แหล่งต่างๆ ก็สามารถกระทำได้ในรีจิสเตอร์ PIE1, PIE2, และ PIE3 ตามลำดับซึ่งเป็นรีจิสเตอร์ที่เอ็นเอเบิลการอินเตอร์รัพต์จากอุปกรณ์ต่อพ่วงต่างๆ

รีจิสเตอร์ที่เกี่ยวข้องกับการเกิดอินเตอร์รัพต์

รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัพต์มีทั้งหมด 13 ตัวซึ่งจะทำหน้าที่ในการควบคุมการอินเตอร์รัพต์และแสดงสถานะการเกิดอินเตอร์รัพต์จากแหล่งต่างๆ โดยจะมีหน้าที่ของรีจิสเตอร์ของแต่ละตัวแตกต่างกันออกออกไป RCON, INTCON, INTCON2, INTCON3, PIR1, PIR2, PIR3, PIE1, PIE2, PIE3, IPR1, IPR2, IPR3 แต่จะขอกล่าวถึงเพียงรีจิสเตอร์ INTCON เท่านั้น

รีจิสเตอร์ INTCON

รีจิสเตอร์ INTCON สามารถอ่านและเขียนได้ทุกบิต มีแอดเดรสอยู่ที่ FF2H โดยจะใช้ในการเอ็นนาเบิลการอินเตอร์รัพต์รวมและการอินเตอร์รัพต์พื้นฐานและเป็นริจิสเตอร์ที่แสดงการเกิดจากแหล่งต่างๆ ด้วยโดยดูจากบิตที่ลงท้ายด้วย F ซึ่งรายละเอียดบิตต่างๆ ของรีจิสเตอร์ INTCON มีรายละเอียดแต่ละบิตดังนี้ดังนี้

ตารางที่ จ.7 รายละเอียดบิตต่างๆ ของรีจิสเตอร์ INTCON

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
GIE/GI	PEIE/GI	TOIE	INTE	RBIE	TOIF	INTF	RBIF
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
R : อ่านค่าได้, W : เขียนค่าได้, U : ไม่ใช้งานอ่านค่าได้เป็นศูนย์, -n : ค่าที่เกิดขึ้นหลังเกิดเพาเวอร์ออนรีเซต							

บิต 7 : GIE / GIEH (Global Interrupt Enable Bit) บิตเอ็นเอเบิล (Enable) การอินเตอร์-รัพต์รวม และบิตเอ็นเอเบิลการอินเตอร์รัพต์รวมไบต์สูง

1 = เลือกให้มีการอินเตอร์รัพต์เกิดขึ้นทั้งหมด

0 = เลือกไม่ให้มีการอินเตอร์รัพต์เกิดขึ้นทั้งหมด

บิต 6 : PEIE / GIEL (Peripheral Interrupt Enable Bit) บิตเอ็นเอเบิลการอินเตอร์รัพต์จากอุปกรณ์ต่อพ่วง และบิตเอ็นเอเบิลการอินเตอร์รัพต์รวมไบต์ต่ำ

1 = เอ็นเอเบิลการอินเตอร์รัพต์แบบนี้

0 = ดิสเอเบิลการอินเตอร์รัพต์แบบนี้

บิต 5 : TMROIE (TMRO Overflow Interrupt Enable Bit) บิตเอ็นเอเบิลการอินเตอร์รัพต์จากการเกิดโอเวอร์โฟลว์ของไทเมอร์ 0

1 = เอ็นเอเบิลการอินเตอร์รัพต์แบบนี้

0 = ดิสเอเบิลการอินเตอร์รัพต์แบบนี้

บิต 4 : INTOIE (INT0 External Interrupt Enable Bit) บิตเอ็นเอเบิลการอินเตอร์รัพต์จากภายนอกที่ขา RB0 / INT

1 = เอ็นเอเบิลการอินเตอร์รัพต์แบบนี้

0 = ดิสเอเบิลการอินเตอร์รัพต์แบบนี้

บิต 3 : RBIE (Port B Change Interrupt Enable) บิตเอ็นเอเบิลการอินเตอร์รัพต์จากการเปลี่ยนแปลงระดับลอจิกที่พอร์ต B ขา RB4-RB7

บิต 2 : TOIF (TMRO Overflow Interrupt Flag Bit) บิตแสดงการโอเวอร์โฟลว์ของ TMR0

1 = ไทเมอร์ 0 โอเวอร์โฟลว์ ทำให้เกิดอินเตอร์รัพต์หากเอ็นเอเบิลไว้

0 = ไทเมอร์ 0 ไม่เกิดโอเวอร์โฟลว์

บิต 1 : INT0F (RBO / INT External Interrupt Flag Bit) บิตแสดงการอินเตอร์รัพต์จากภายนอกที่ขา RBO / INT

1 = มีสัญญาณอินเตอร์รัพต์จากภายนอกเกิดขึ้นที่ขา RBO / INT

0 = ไม่มีสัญญาณอินเตอร์รัพต์จากภายนอกเกิดขึ้นที่ขา RBO / INT

บิต 0 : RBIF (Port B Chang Interrupt Flag Bit) บิตแสดงการเปลี่ยนแปลงระดับลอจิกที่ขา RB4-RB7

1 = มีการเปลี่ยนแปลงเกิดขึ้นที่ขา RB4-RB7

0 = ไม่มีการเปลี่ยนแปลงเกิดขึ้นที่ขา RB4-RB7

รายระเอียดแหล่งของแหล่งกำเนิดการอินเตอร์รัพต์ของ PIC18F458

การอินเตอร์จากภายนอกที่ขา RB0/INT0, RB1/INT1 และ RB2/INT2

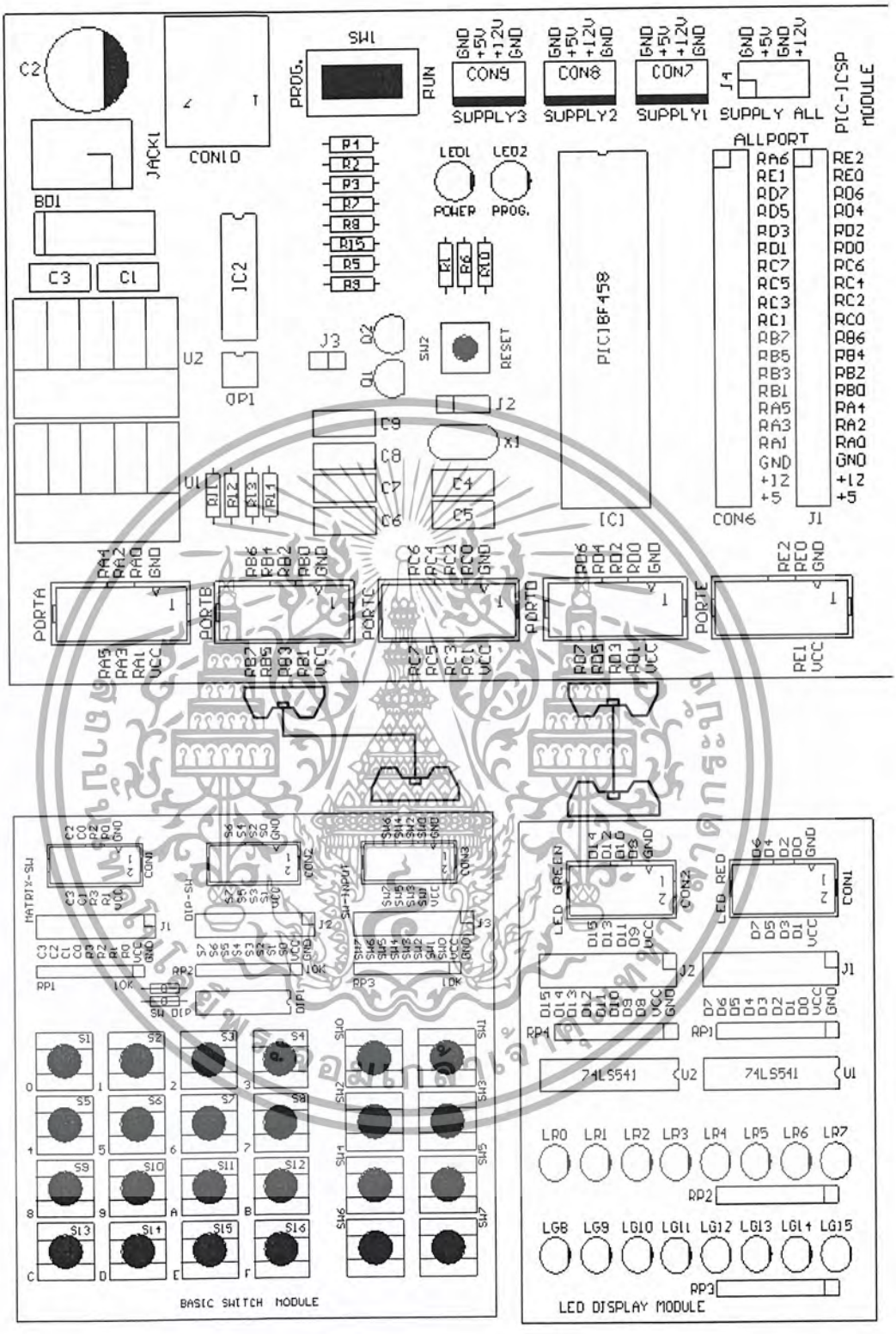
การอินเตอร์รัพต์แบบนี้จะเกิดขึ้นได้เมื่อมีการเปลี่ยนแปลงสัญญาณลอจิกที่ขา RB0-RB2 ซึ่งสามารถกำหนดรูปแบบของสัญญาณที่ต้องการได้ที่บิต INTEDGX ของรีจิสเตอร์ INTCON3 ถูกเซตจะเป็นการเลือกขอบขาของสัญญาณการอินเตอร์รัพต์จากแหล่งนี้โดยเมื่อเซตบิต INTEDGX คือการอินเตอรัพต์จะเกิดสัญญาณที่ขอบขาลงและถ้าเคลียร์จะเกิดสัญญาณการอินเตอร์รัพต์ที่ขอบขาขึ้นนั่นเองสำหรับการเกิดอินเตอร์รัพต์ทั้ง 3 ที่ขานี้สามารถเอ็นเอเบิลและดิสเอเบิลได้ที่บิต INTxIEโดยซึ่งปกติแล้วเมื่อเกิดอินเตอร์รัพต์ขึ้นที่ขาใดจะทำให้บิต INTxIF เซตคือแสดงว่ามีการเกิดอินเตอร์รัพต์ขึ้นที่ขาดังกล่าวโดยที่บิตนี้สามาถเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์เมื่อต้องการออกจากกระบวนอินเตอร์รัพต์แบบนี้และการอินเตอร์รัพต์แบบนี้สามารถทำให้เกิดเวกอัฟ (Wake Up) ได้การเกิดเวกอัฟคือการที่ซีพียูออกจากโหมดสลีปนั่นเองสำหรับในการเลือกลำดับความสำคัญของการเกิดอินเตอร์รัพต์แบบนี้สามารถกระทำได้ที่รีจิสเตอร์ INTCON3 ที่บิต INT1IP และ INT2IP ส่วนขาที่ไม่สามารถเลือกได้คือ INT0 เพียงขาเดียว

การอินเตอร์รัพต์เนื่องจากการเกิดโอเวอร์โฟลว์ของไทเมอร์ 0

การอินเตอร์รัพต์แบบนี้จะเกิดเนื่องมาจากการเกิดโอเวอร์โฟลว์ของไทเมอร์ 0 การเกิดโอเวอร์โฟลว์คือการที่ไทเมอร์ 0 ทำงานและมีการเปลี่ยนแปลงจาก FF เป็น 00 ในโหมด 8 บิตและ FFFF เป็น 0000 ในโหมด 16 บิต เนื่องจากไทเมอร์ 0 ของไมโครคอนโทรลเลอร์ PIC18F458 จะทำงานได้ในโหมด 8 บิตและ 16 บิตโหมด 8 คือรีจิสเตอร์ TMRO และบิตที่แสดงสถานะคือ TMROIF ซึ่งอยู่ในรีจิสเตอร์ INTCON ในส่วนของโหมด 16 คือรีจิสเตอร์ TMROL และ TMROH เป็นไทเมอร์ไบต์สูงและไบต์ต่ำและบิตที่แสดงสถานะคือ TMROIF สำหรับการเกิดอินเตอร์รัพต์ของไทเมอร์ 0 นี้จะต้องทำการเอ็นนาเบิลที่บิต TMROIE ก่อนโดยการเซตให้เป็น 1 ในรีจิสเตอร์ INTCON สำหรับในส่วนของการเลือกลำดับก่อนหลังการอินเตอร์รัพต์สามารถกระทำได้ที่บิต TMROIP ในรีจิสเตอร์ INTCON2

การอินเตอร์รัพต์จากการเปลี่ยนแปลงลอจิกที่ขา RB4-RB7 ของพอร์ต B

การอินเตอร์รัพต์แบบนี้จะเกิดเนื่องมาจากการเกิดเปลี่ยนแปลงระดับลอจิกของ RB4-RB7 ของพอร์ต B เกิดขึ้นการอินเตอร์รัพต์แบบนี้จะเกิดขึ้นเมื่อมีการเซตบิต RBIE (บิต 3 ของรีจิสเตอร์ INTCON) เมื่อมีการเอ็นเอเบิลการอินเตอร์รัพต์แบบนี้ ขาพอร์ต B ทั้ง 4 ขาคือ RB4-RB7 จะเริ่มตรวจสอบปรียบเทียบระดับลอจิกที่ขาสัญญาณว่าเกิดการเปลี่ยนแปลงหรือไม่ถ้ามีการเปลี่ยนแปลงขึ้นก็จะเซตบิต RBIF (บิต 0 ของรีจิสเตอร์ INTCON) ทำให้เกิดการอินเตอร์รัพต์ขึ้นและสำหรับการเลือกลำดับความสำคัญของการเกิดอินเตอร์รัพต์แบบนี้กระทำได้ที่บิต RBIP



รูปที่ จ.14 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับโมดูลแสดงผลแอลอีดีและโมดูลสวิตช์พื้นฐาน

```
;*********************************************************
; Filename :          lab6.asm
; Descripton :        test interrup
; MCU :               PIC18F458
;*********************************************************
                    list        p=18F458
          #include <p18F458.inc>
               DT1      EQU  0x20
               DT2      EQU  0x21
             COUNT  EQU  0x22
             W_TEMP EQU  0x23
             STATUS EQU  0x24
          PORTD_TEMP EQU  0x25
;*********************************************************
                    ORG 0x0000    ; RESET VECTOR
               GOTO      MAIN
;*********************************************************
                    ORG 0x0008    ; INTERRUP VECTOR
                    MOVWF     W_TEMP
                    SWAPF     STATUS,W
                    MOVWF     STATUS, W_TEMP
                    SWAPF     PORTD,W
                    MOVWF     PORTD _TEMP
                    MOVLW     .10
                    MOVWF     COUNT
INT_LOOP            CLRF      PORTD
                    CALL      DELAY
                    MOVLW     0Xff
                    MOVWF     PORTD
                    CALL      DELAY
                    DECFSZ    COUNT,F
                    GOTO      INT_LOOP
                    SWAPF     STATUS, W_TEMP
                    MOVWF     STATUS
                    SWAPF     PORTD _TEMP,W
                    MOVWF     PORTD
                    MOVF      W_TEMP,W
                    BCF       INTCON,RBIF
                    RETFIE
```

```
;***********************************************
            Main  Program
;***********************************************
                ORG   0x0030
MAIN            CLRF    STATUS
                CLRF    TRISD
                MOVLW   0XFF
                MOVWF   TRISB
                BCF     INTCON2,PBPU
                CLRF    PORTB
                BCF     INTCON,PBIF
                BSF     INTCON,GIE
                BSF     INTCON,RBIE
                MOVLW   0x01
                MOVWF   PORTD
LOOP            CALL    DELAY
                CALL    DELAY
                RRLCF   PORTD,F
                GOTO    LOOP
;******************* DELAY *****************************
DELAY           CLRF    DT1
DEL1            CLRF    DT2
DEL2            DECFSZ  DT2,F
                GOTO    DEL2
                DECFSZ  DT1,F
                GOTO    DEL1
                RETURN
                END
```

รูปที่ จ.15 โปรแกรมการทดลองการอินเตอร์รัพต์จากการเปลี่ยนแปลงลอจิกของพอร์ต B

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ได้ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.15 ทำการคอมไพล์ให้เป็น Hex File
2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์
3. ต่อวงจรตามรูปที่ จ.14
4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

.....

.....

.....

1. ทดลองกดสวิตช์ที่ต่ออยู่กับพอร์ต B ตำแหน่ง S0-S4 สังเกตผลที่เกิดขึ้นบันทึกผลในตาราง

ตารางที่ จ.8 ตารางการทดลองการอินเตอร์รัพต์จากพอร์ต B

ตำแหน่งสวิตช์ที่ที่ถูกกด	ผลที่เกิดขึ้นกับแอลอีดี
S0	
S1	
S2	
S3	

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายการทำงานของแหล่งกำเนิดการอินเตอร์รัพต์จากการเปลี่ยนแปลงลอจิกที่ขา RB4-RB7

2. จงอธิบายการอินเตอร์รัพต์จากแหล่งต่างๆ มาพอเข้าใจ

ใบงานที่ 7
การทดลองใช้งานไทเมอร์เคาน์เตอร์ภายใน PIC18F458

จุดประสงค์

1. เพื่อให้สามารถใช้งานไทเมอร์เคาน์เตอร์ใน PIC18F458ได้
2. เพื่อให้สามารถใช้งานวอตช์ด็อกไทเมอร์ร่วมกับ TMR0 ได้

อุปกรณ์ที่ใช้ในการทดลอง

1. คอมพิวเตอร์พีซีที่ติดตั้งโปรแกรม MPLAB และ Epic Win
2. ออสซิลโลสโคป 2 ช่องความถี่ไม่น้อยกว่า 20 MHz พร้อมสายวัด
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
5. โมดูลแสดงผลแอลอีดี
6. โมดูลพัลส์เจเนอเรเตอร์

ทฤษฎีเบื้องต้น

รายละเอียดเบื้องต้นของไทเมอร์เคาน์เตอร์

PIC18F458 มีไทเมอร์เคาน์เตอร์ 3 ตัว คือ ไทเมอร์ 0, ไทเมอร์ 1, ไทเมอร์ 2 ไทเมอร์ 0 ในการทดลองนี้จะขอกล่าวเฉพาะไทเมอร์ 0 เท่านั้น สามารถทำงานได้ในโหมด 8 และ 16 บิต ภายในไทเมอร์เคาน์เตอร์ประกอบด้วย รีจิสเตอร์ TMR0 ปรีสเกลเลอร์ขนาด 8 บิต และตัวนับหรือเคาน์เตอร์ โดย ไทเมอร์เคาน์เตอร์ใน PIC18F458 สามารถทำงานได้ทั้งจากสัญญาณนาฬิกาภายในและภายนอก ทั้งยังสามารถเลือกให้ทำงานที่ขอบขาของสัญญาณทั้งขอบขาขึ้นและขาลง โดยการกำหนดค่ารีจิสเตอร์ T0CON การนับภายในไทเมอร์เคาน์เตอร์ จะเป็นการนับขึ้นเท่านั้น เมื่อค่าของการนับภายในตัวนับของ ไทเมอร์เคาน์เตอร์เปลี่ยนจาก 0xFF เป็น 0x00 หรือเกิดโอเวอร์โฟลว์ จะเกิดการอินเตอร์รัพต์ขึ้น แต่ทั้งนี้ต้องขึ้นอยู่กับว่ามีการอีนาบิลการอินเตอร์รัพต์แบบนี้ที่บิต TOIE ในรีจิสเตอร์ T0CON หรือไม่ปรีสเกลเลอร์ภายในไทเมอร์เคาน์เตอร์มีขนาด 8 บิต

สามารถเลือกอัตราการหารความถี่ได้ถึง 8 ระดับ ตั้งแต่หาร 2 ไปจนถึงหาร 256 โดยการกำหนดค่าที่บิต PSA, T0PS2-T0PS0 ในรีจิสเตอร์ T0CON

รีจิสเตอร์ T0CON

รีจิสเตอร์ T0CON เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของไทเมอร์ 0 ซึ่งมีรายละเอียดของแต่ละบิตดังนี้

ตารางที่ จ.9 รายละเอียดบิตต่างๆ ของรีจิสเตอร์ T0CON

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
TMR0O	T08BIT	TOIE	TOCS	PSA	T0PS2	T0PS1	T0PS0
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
R : อ่านค่าได้, W : เขียนค่าได้, U : ไม่ใช้งานอ่านค่าได้เป็นศูนย์, -n : ค่าที่เกิดขึ้นหลังเกิดเพาเวอร์ออนรีเซต							

TMR0ON (Timer 0 On/Off Control Bit – บิต7) : บิตเอ็นนาเบิลการทำงานของไทเมอร์ 0

"0" – ปิดการทำงานของไทเมอร์ 0

"1" – เอ็นนาเบิลการทำงานของไทเมอร์ 0

T08BIT (Timer 0 8-Bit/16-Bit Control Bit – บิต6) : บิตเลือกโหมดการทำงานของไทเมอร์ 0

"0" – เลือกการทำงานของไทเมอร์ 0 เป็นโหมด 16 บิต ไทเมอร์/เคาน์เตอร์

"1" – เลือกการทำงานของไทเมอร์ 0 เป็นโหมด 8 บิต ไทเมอร์/เคาน์เตอร์

TOCS (Tmer 0 Clock Source Select Bit – บิต 5) : บิตเลือกแหล่งจ่ายสัญญาณนาฬิกาของไทเมอร์ 0

"0" – เลือกรับสัญญาณจากสัญญาณนาฬิกาจากภายในไมโครคอนโทรลเลอร์

"1" – เลือกรับสัญญาณจากภายนอกที่ขา RA4/TOCKI

T0SE (Timer0 Source Edge Select Bit – บิต4): บิตเลือกรูปแบบของสัญญาณลอจิกที่ขา RA4/TOCKI สำหรับไทเมอร์ 0

"0" – ไทเมอร์ 0 เพิ่มค่าขึ้นเมื่อลอจิกที่ขา TOCKI เปลี่ยนจาก "1" เป็น "0"

"1" –ไทเมอร์0 เพิ่มค่าขึ้นเมื่อลอจิกที่ขา TOCKI เปลี่ยนจาก "0" เป็น "1" PSA (Prescaler Assignment Bit – บิต 3) : บิตเลือกการทำงานของปรีสเกลเลอร์

"0" – เลือกให้ปรีสเกลเลอร์ทำงานร่วมกับไทเมอร์ 0

"1" – เลือกให้ปรีสเกลเลอร์ทำงานร่วมกับวอตซ์ด็อกไทเมอร์ เมื่อปรีสเกลทำงานกับวอตซ์ด็อกไทเมอร์จะเรียกว่า โพสต์สเกลเลอร์ (Postscaler)

T0PS2-T0PS0 (Timer0 Prescaler Select Bits – บิต 2,1 และ 0) : บิตเลือกอัตราส่วนของปรีสเกลเลอร์ ใช้กำหนดอัตราส่วนในการทำงานของปรีสเกลเลอร์เมื่อทำงานร่วมกับทั้งวอตซ์ด็อกไทเมอร์ ซึ่งจะมีอัตราส่วนที่แตกต่างกัน ดังมีรายละเอียดการกำหนดค่าดังนี้

ตารางที่ จ.10 อัตราส่วนของปรีสเกลเลอร์

T0PS2	T0PS1	T0PS0	อัตราส่วนของปรีสเกลเลอร์	
			เมื่อทำงานร่วมกับ TMRO	เมื่อทำงานร่วมกับ WDT
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

สำหรับสัญญาณนาฬิกาที่ใช้ในการกระตุ้นให้ไทเมอร์เคาน์เตอร์ทำงานมาจากแหล่งกำเนิด 2 แหล่ง คือ จากวงจรกำเนิดสัญญาณนาฬิกาภายในตัวไมโครคอนโทรลเลอร์เอง และวงจรกำเนิดสัญญาณนาฬิกาจากภายนอก บิตที่ใช้ในการเลือกแหล่งกำเนิดสัญญาณนาฬิกาก็คือ บิต TOCS ในรีจิสเตอร์T0CON บิตที่ใช้เลือกว่าต้องการให้สัญญาณผ่านการปรีสเกลเลอร์ หรือไม่ก็คือบิต PSA บิตที่ 3

การใช้ TMR0 สร้างโปรแกรมหน่วงเวลาเพื่อสร้างสัญญาณสี่เหลี่ยม

การกำเนิดสัญญาณสี่เหลี่ยมความถี่ 1 KHz สัญญาณลอจิกสูงจะปรากฏที่เอาต์พุตนานเพียงไรขึ้นอยู่กับการหน่วงเวลาให้เกิดสัญญาณลอจิกสูงค้างที่เอาต์พุตซึ่งในที่นี้ใช้ไทเมอร์

เคาน์เตอร์ทำหน้าที่นี้เริ่มต้นด้วยการเคลียร์ค่าวอตช์ด็อกไทเมอร์ จากนั้นเขียนข้อมูลลงในรีจิสเตอร์ T0CON เพื่อกำหนดให้ ปรีสเกลเลอร์ทำงานร่วมกับ TMR0 และเลือกที่จะใช้สัญญาณนาฬิกาจากแหล่งกำเนิดสัญญาณนาฬิกาภายในตัว PIC18F458 เองกำหนดให้ปรีสเกลเลอร์มีอัตราการลดทอนความถี่เท่ากับ 128 นั่นคือสัญญาณนาฬิกาภายใน จะถูกหารด้วย 128 ก่อนส่งเข้าสู่ตัวนับในไทเมอร์เคาน์เตอร์ภายใน PIC18F458 ดังนั้นสัญญาณเอาต์พุตที่ได้จะสามารถคำนวณค่าความถี่ได้ดังนี้

ความถี่ = 1/ (ค่าของปรีสเกลเลอร์ x ค่าของ TMR0 x คาบเวลาของสัญญาณนาฬิกาภายใน x 2)

ถ้าค่าปรีสเกลเลอร์ = 128, TMR0 = 4 และสัญญาณนาฬิกามีคาบเวลา 1 ไมโครวินาที ความถี่ที่ได้มีค่า 1,024 Hz

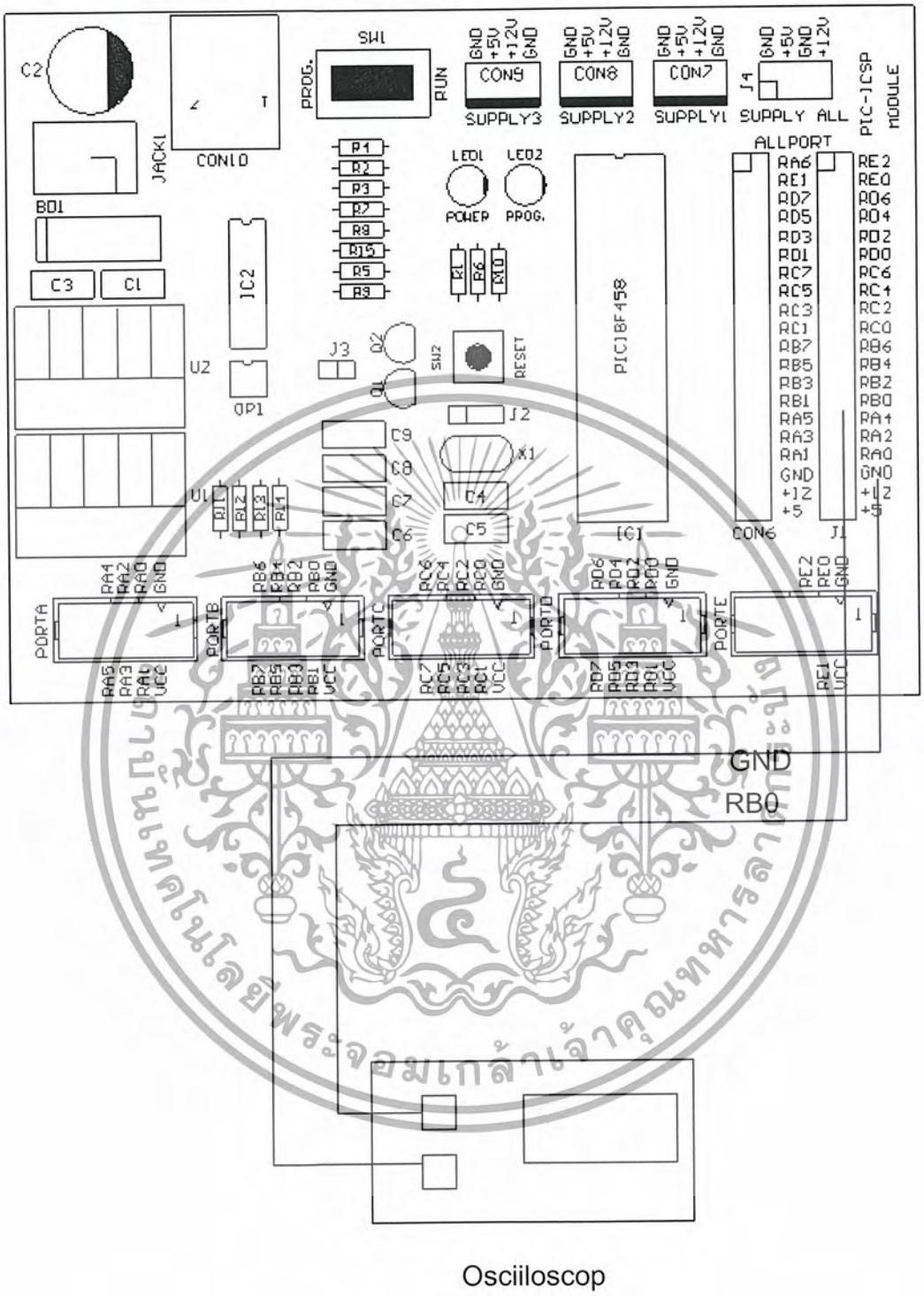
การกำหนดค่าของปรีสเกลเลอร์เป็น 128 สามารถทำได้โดยการกำหนดข้อมูลลงในรีจิสเตอร์ T0CON ตารางที่ จ.10 การทำงานของโปรแกรมเริ่มต้นด้วยการเคลียร์ค่าของ TMR0 จากนั้นตัวนับ จะเริ่มทำงาน เริ่มนับค่าเพิ่มขึ้นเรื่อยๆ จนกระทั่งบิต 2 ของข้อมูลใน TMR0 เกิดเป็น "1" ก็จะออกจากโปรแกรมย่อยของการหน่วงเวลาทันที

ไทเมอร์เคาน์เตอร์กับการเกิดอินเตอร์รัพต์

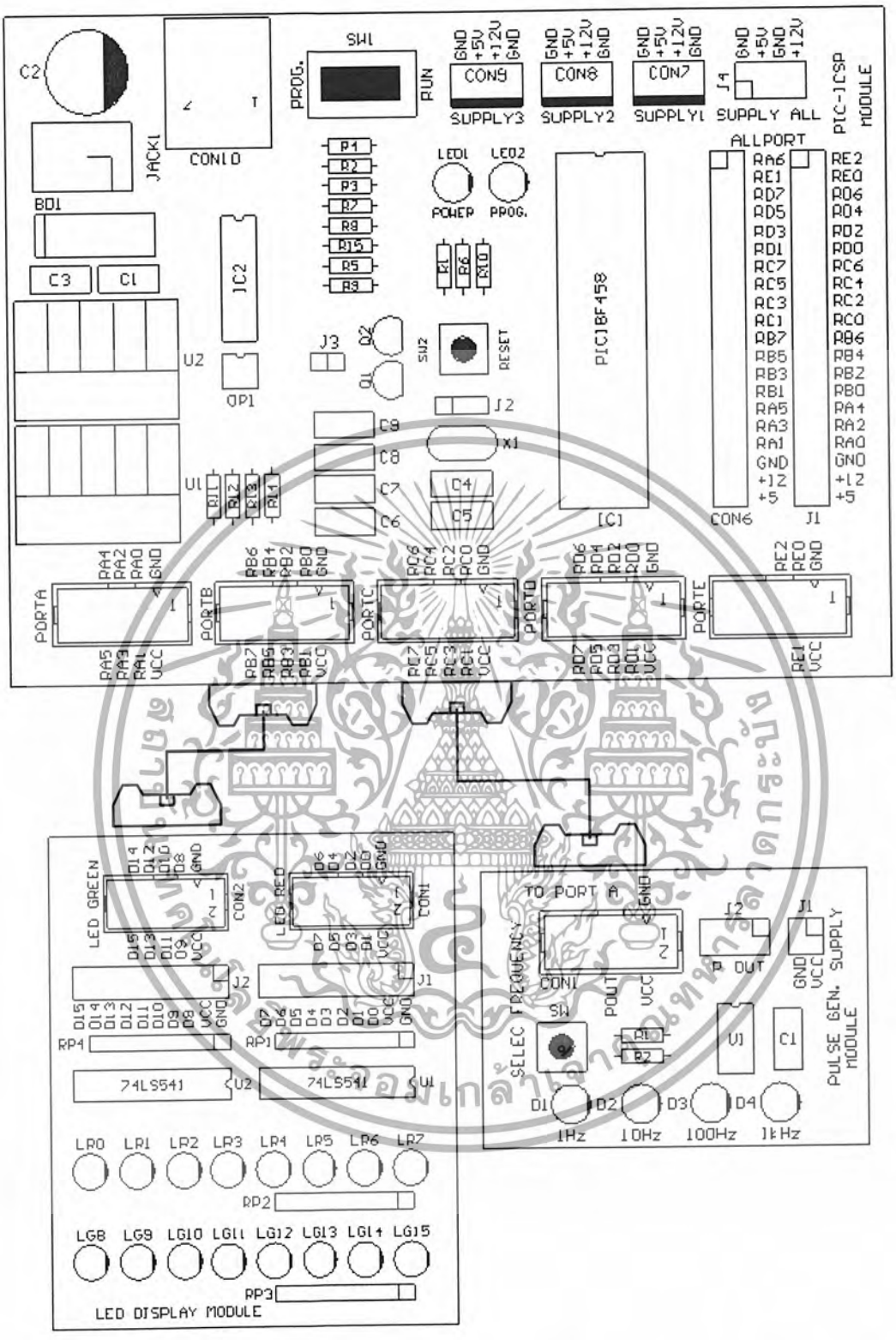
ในหัวข้อนี้จะกล่าวถึงการกำเนิดสัญญาณรูปสี่เหลี่ยมโดยใช้การอินเตอร์รัพต์ที่เกิดขึ้นในไทเมอร์เคาน์เตอร์ เมื่อ TMR0 เกิดโอเวอร์โฟลว์โดยการทำให้บิต RB0-RB2 ซึ่งเป็นขาที่ใช้รับสัญญาณการอินเตอร์รัพต์ โดยทำให้ขานี้มีค่าเป็น "0" และ "1" สลับกัน จะทำให้เกิดเป็นสัญญาณรูปสี่เหลี่ยมขึ้น

ไทเมอร์เคาน์เตอร์กับสัญญาณนาฬิกาภายนอก

การกำหนดให้ไทเมอร์เคาน์เตอร์นับสัญญาณนาฬิกาจากภายนอกนั้น เริ่มต้นด้วยการกำหนดให้บิต TOCS ของรีจิสเตอร์ T0CON เป็น "1" เพื่อเลือกการรับสัญญาณนาฬิกาจากภายนอก และเพื่อให้ การนับมีค่าถูกต้องจะต้องกำหนดขอบขาของสัญญาณ โดยสามารถกำหนดได้ที่บิต TOCS บิตที่ 4 ของรีจิสเตอร์ T0CON เป็นการกำหนดให้ไทเมอร์เคาน์เตอร์นับค่าของสัญญาณนาฬิกาจากภายนอก เท่ากับ 1 Hz เมื่อมีสัญญาณนาฬิกาเข้ามาครบ 8 ลูก LED ที่ต่ออยู่กับพอร์ต B จะเลื่อนไป 1 บิต



รูปที่ จ.16 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับออสซิลโลสโคป



รูปที่ จ.17 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP และโมดูลพัลส์เจเนอเตอร์

```
;********LAB 8-1 Timer 0 TEST Bit TMRO ***********
                LIST P=18f458
    #INCLUDE <P18f458.INC>
     config CP_OFF&PWT_ON&WDT_OFF&XT_OSC

                org         0x000
                goto        Start
;************* กำหนดพอร์ต**********************
Start           Clrwdt
                clrf        TRISB           ; PortB = Output
                movlw       b'11110110'     ; Prescaler = 128, 8  bit mode
                movwf       T0CON           ; Internal Clock
;**************เริ่มต้น*****************************
                clrf        PORTB
PLUSE           bsf         PORTB,0         ; Pulse High
                call        Delay
                bcf         PORTB,0         ; Pulse Low
                call        Delay
                goto        PLUSE
;************ หน่วงเวลา*************************
Delay           clrf        TMR0            ; Timer Start
Loop            btfss       TMR0,2          ; Timer =4 ?
                goto        Loop            ; No
                return                      ; Yes Back
                end
```

รูปที่ จ.18 โปรแกรมการทดสองวงจรกำเนิดสัญญาณพัลส์โดยการตรวจสอบบิตของ TMR0

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เขียนตามรูปที่ จ.18 เพื่อสร้างสัญญาณพัลส์โดยใช้ไทเมอร์เคาน์เตอร์ภายใน PIC18F458 โดยเลือกแหล่งกำเนิดสัญญาณนาฬิกาจากวงจรกำเนิดสัญญาณนาฬิกา ภายในของ PIC18F458

2. ต่อวงจรตามรูปที่ จ.16

3. แอสเซมเบอร์โปรแกรมที่ จ.18 แล้วโปรแกรมข้อมูลลงบน PIC18F458 ด้วยโปรแกรม Epic Win

4. รันโปรแกรมแล้วใช้ออสซิลโลสโคปวัดสัญญาณที่ขา RB0 บันทึกรูปสัญญาณที่วัดได้

ขนาดของรูปสัญญาณเอาต์พุต = ... Vp-p

ความถี่ = ... Hz

```
;********LAB 8-2 Timer 0 Interrup ***********
                LIST P=18f458
    #INCLUDE <P18f458.INC>
     config CP_OFF&PWT_ON&WDT_OFF&XT_OSC
                org         0x000
                goto        Start
                org         0x0008
                goto        INT_RB
; ***************** กำหนดพอร์ต*******************
Start           clrf        TRISB           ; PortB = Output
                clrwdt
                movlw       b'11010110'     ; Prescaler = 128
                movwf       OPTION_         ; Internal Clock
                clrf        PORTB
                bcf         INTCON,TOIF     ; Clear Flag
                bsf         INTCON,GIE      ; Global INT Enable
                bsf         INTCON,TOIE     ; TMR0 INT Enable
                movlw       0xfc            ; TMR0 Count = 4
                movwf       TMR0
                goto        $               ; Wait TMR0 Overflow
;*************โปรแกรมบริการการอินเตอร์รัพต์**************
INT_RB          bcf         INTCON,TOIF     ; Clear Flag
                movlw       0x01
                xorwf       PORTB,F         ; Toggle RB0
                movlw       0xfc
                movwf       TMR0            ; TMR0 Count Again
                retfie
                end
```

รูปที่ จ.19 โปรแกรมการทดลองการสร้างรูปสัญญาณสี่เหลี่ยมโดยใช้การอินเตอร์รัพต์ของไทเมอร์

5. จากผังการทำงานที่ออกแบบไว้เขียนโปรแกรมตามรูป จ.19 เพื่อสร้างสัญญาณสี่เหลี่ยมความถี่ 1 kHz โดยใช้การอินเตอร์รัพต์เนื่องจาก TMR0 เกิดโอเวอร์โฟลว์

6. ใช้วงจรตามรูปที่ จ.16 ในการทดลอง

7. ทำการคอมไพล์ แล้วเขียนลงบน PIC18F458 ด้วยโปรแกรม Epic Win

8. รันโปรแกรมแล้ว ใช้ออสซิลโลสโคปวัดสัญญาณที่ขา RB0 บันทึกรูปสัญญาณที่วัดได้

ขนาดของรูปสัญญาณเอาต์พุต = ... Vp-p

ความถี่ = ... Hz

10. ถ้าต้องการสร้างสัญญาณพัลส์ความถี่ 2 kHz โดยใช้การอินเตอร์รัพต์เนื่องจาก TMR0 เกิดโอเวอร์โฟลว์ ต้องกำหนดค่าของ

ปรีสเกลเลอร์ = ...

TMR0 = ...

11. เปลี่ยนค่าของอัตราส่วนปรีสเกลเลอร์เป็น 256 แล้วแก้ไขโปรแกรมเป็นจ.19 ทำการแอสเซมเบลอร์ แล้วเขียนลงบน PIC18F458 ทำการทดลองซ้ำในข้อที่ 8

ความถี่เอาต์พุต มีค่าเท่ากับ .....Hz

13. เขียนโปรแกรม จ.20 ทำการคอมไพล์ และเขียนลงบน PIC18F458 ด้วยโปรแกรม Epic Win

14. ต่อวงจรตามรูปที่ จ.17 โดยเลือกความถี่ของโมดูลพัลส์เจเนอเรเตอร์เท่ากับ 1 Hz

15. รันโปรแกรมเลื่อนสวิตช์ ไปที่ตำแหน่งรัน

16 นับจำนวนการกะพริบของ LED มอนิเตอร์ที่ต่ออยู่กับขา RA4/T0CKI แล้วสังเกตการทำงานของ LED มอนิเตอร์ที่ต่ออยู่กับพอร์ต B

LED ที่ RA4/TOCKI กระพริบ ..... ครั้ง แล้ว LED ที่พอร์ต B จึงเกิดการเลื่อน 1บิต

17. แก้ไขโปรแกรมที่ จ.20 เปลี่ยนค่าของปรีสเกลเลอร์เป็น 256

18. เปลี่ยนความถี่ของ โมดูลพัลส์เจเนอเรเตอร์ เป็น 1 kHz

19. แอสเซมเบลอร์ โปรแกรมที่ทำการแก้ไข แล้วเขียนลงบน PIC18F458 อีกครั้ง

20. รันโปรแกรมเลื่อนสวิตช์ ไปที่ตำแหน่งรัน

21. สังเกตการทำงานของ LED มอนิเตอร์ที่พอร์ต B

LED มอนิเตอร์ที่พอร์ต B เกิดการเลื่อนบิตในอัตรา ..... ครั้งต่อวินาที

22. จากการทดลองในข้อ 21 จงคำนวณหาค่าของจำนวนสัญญาณนาฬิกาจากภายนอกที่ให้ข้อมูลที่พอร์ต B เกิดการเลื่อนไป 1 บิต

จำนวนของสัญญาณนาฬิกาเท่ากับ ..... ลูก

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. ไทเมอร์เคาน์เตอร์ใน PIC18F458 มีกี่ตัวแต่ละตัวมีขนาดกี่บิต และมีส่วนประกอบภายในอะไรบ้าง จงอธิบาย

2. รีจิสเตอร์ที่เกี่ยวข้องในการใช้งานไทเมอร์เคาน์เตอร์ของ PIC18F458 มีอะไรบ้างและเกี่ยวข้องอย่างไรจงอธิบาย

3. จงอธิบายการทำงานของไทเมอร์เคาน์เตอร์ใน PIC18F458

4. จากการทดลองจงสรุปถึงการนำไทเมอร์เคาน์เตอร์ใน PIC18F458 ไปใช้ประโยชน์ว่านำไปใช้งานอย่างไรบ้าง และทำได้อย่างไร

4. จงอธิบายความหมายของปรีสเกลเลอร์และโพสต์สเกลเลอร์

ใบงานที่ 8

การใช้งาน PIC18F458 ขับแอลอีดีตัวเลข 7 ส่วน

จุดประสงค์

1. เพื่อให้สามารถอธิบายหลักการแสดงผลของแอลอีดีตัวเลข 7 ส่วนได้
2. เพื่อให้สามารถใช้ PIC18F458 นำข้อมูลออกไปแสดงผลที่แอลอีดีตัวเลข 7 ส่วนได้
3. เพื่อให้สามารถแก้ไขดัดแปลงโปรแกรมจากการทดลองได้
4. เพื่อให้สามารถสรุปและวิจารณ์ผลการทดลองเทียบกับทฤษฎีได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ EPIC win
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
5. โมดูลแสดงผลตัวเลข 7 ส่วน

ทฤษฎีเบื้องต้น

ความรู้เบื้องต้นเกี่ยวกับแอลอีดีตัวเลข 7 ส่วน

การแสดงผลที่เป็นแอลอีดีตัวเลข 7 ส่วน (LED 7 Segment) แต่ละส่วนหรือเซกเมนต์มีชื่อเรียกแตกต่างกันตามตำแหน่งที่ได้รับการจัดวาง คือ a, b, c, d, e, f และ g ดังรูปที่ ส่วน db เป็นแอลอีดีอีก 1 ใช้แสดงตัวเลขที่จุดทศนิยมแอลอีดีตัวเลข 7 ส่วนนี้มีทั้งแบบคาโถดร่วม (Common Cathode) และแบบอาโนดร่วม (Common Anode) การขับให้แอลอีดีตัวเลข 7 ส่วน แบบคาโถดร่วมสว่างจะต้องจ่ายไฟลบเข้าที่ขาร่วม แล้วจ่ายไฟบวกเข้าที่ขาอาโนด ซึ่งก็คือขาของแต่ละเซกเมนต์นั่นเองในขณะที่แอลอีดีตัวเลข 7 ส่วนแบบอาโนดร่วมจะต้องจ่ายไฟบวกเข้าที่ขาร่วมแล้วจ่ายไฟลบเข้าที่ขาคาโถดซึ่งเป็นขาของแต่ละเซกเมนต์

การขับแอลอีดีตัวเลข 7 ส่วนแบบหลักเดียว

การกำหนดให้แอลอีดีตัวเลข 7 ส่วนแสดงข้อมูลเป็นตัวเลขหรือสัญลักษณ์ใดๆ ก็ตาม ต้องมีการกำหนดรูปแบบการแสดงผลของเซกเมนต์ต่างๆด้วยข้อมูลแต่ละบิตของไมโครคอนโทรลเลอร์แล้วใช้วิธีการเปิดตารางข้อมูลของการแสดงผลตัวเลขฐานสิบหกของแอลอีดีตัวเลข 7 ส่วน ในตารางที่ จ.12 ซึ่งเป็นตารางข้อมูลของแสดงผลของแอลอีดีตัวเลข 7 ส่วนแบบคาโถดร่วมแต่ถ้าต้องการข้อมูลที่ใช้กับแอลอีดีตัวเลข 7 ส่วนแบบอาโนดร่วมก็จะต้องเป็นข้อมูลเลขตรงข้ามกับแบบคาโถด

ตารางที่ จ. 11 ข้อมูลของการแสดงผลตัวเลข 0-F ของแอลอีดีตัวเลข 7 ส่วนแบบคาโถดร่วม

ข้อมูลดิจิตอลเอาต์พุตสำหรับขับตัวเลข 7 ส่วน								ค่าเลขฐานสิบหกที่ใช้กับ PIC18F458	ค่าตัวเลขที่แสดงที่ 7 เซกเมนต์
D7	D6	D5	D4	D3	D2	D1	D0		
0	0	1	1	1	1	1	1	0x3F	0
0	0	0	0	0	1	1	0	0x06	1
0	1	0	1	1	0	1	1	0x5B	2
0	1	0	0	1	1	1	1	0x4F	3
0	1	1	0	0	1	1	0	0x66	4
0	1	1	0	1	1	0	1	0x6D	5
0	1	1	1	1	1	0	1	0x7D	6
0	0	0	0	0	1	1	1	0x07	7
0	1	1	1	1	1	1	1	0x7F	8
0	1	1	0	1	1	1	1	0x6F	9
0	1	1	1	0	1	1	1	0x77	A
0	1	1	1	1	1	0	0	0x7C	b
0	0	1	1	1	0	0	1	0x39	C
0	1	0	1	1	1	1	0	0x5E	D
0	1	1	1	1	0	0	1	0x79	E
0	1	1	1	0	0	0	1	0x71	F
1	1	1	1	1	1	1	1	0xFF	88

การขับแอลอีดีตัวเลข 7 ส่วน แบบมัลติเพล็กซ์

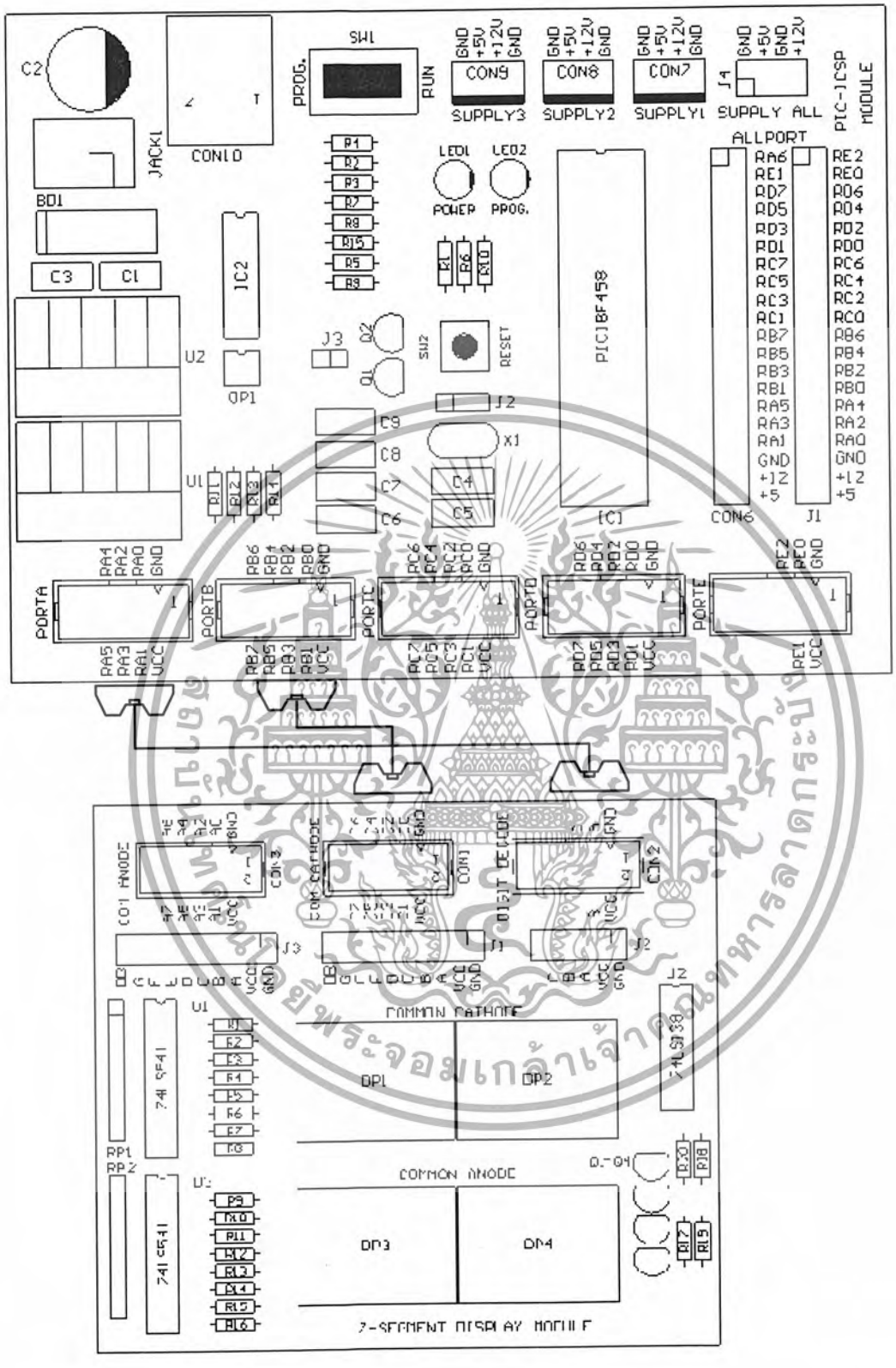
ถ้าเราต้องการให้ PIC18F458 ขับแอลอีดีตัวเลข 7 ส่วนได้มากกว่า 1 หลัก จะต้องใช้เทคนิคการแสดงผลที่เรียกว่า การแสดงผลแบบมัลติเพล็กซ์อันเป็นวิธีการขับให้แอลอีดีสว่างทีละหลักด้วยอัตราเร็วที่ตาของมนุษย์ไม่สามารถตรวจจับได้ทัน จึงเหมือนว่าแอลอีดีทุกหลักติดสว่างในเวลาเดียวกัน

ประโยชน์ของการแสดงผลแบบมัลติเพล็กซ์นี้มีหลายประการดังนี้

1. ช่วยลดพลังงานไฟฟ้าที่ใช้ ทำให้ขนาดของแหล่งจ่ายกำลังงานไฟฟ้าเล็กลง
2. ช่วยประหยัดพอร์ตในการขับแอลอีดีตัวเลข 7 ส่วนหลายๆ หลัก
3. ลดจำนวนตัวต้านทานที่ใช้ในการจำกัดกระแสของแอลอีดี

การขับแอลอีดีตัวเลข 7 ส่วนแบบมัลติเพล็กซ์จะทำการต่อขาของแต่ละเซกเมนต์ร่วมกันคือ เซกเมนต์ a ของทุกหลักจะต่อถึงกันไล่เรียงกันไปจนถึงเซกเมนต์ a,b การควบคุมให้หลักใดติดสว่างทำได้โดยการจ่ายไฟฟ้าที่ขาร่วมของตัวเลข 7 ส่วนหลักนั้นๆ เช่น ถ้าตัวเลข 7 ส่วนที่ใช้เป็นแบบคาโถดร่วม หากต้องการให้หลักที่ 3 ติดสว่างก็ให้ต่อขาร่วมของหลักที่ 3 ลงกราวด์หรือจ่ายไฟลบให้แล้วส่งข้อมูลที่ต้องการให้ติดเข้ามาทางเซกเมนต์ แอลอีดีตัวเลข 7 ส่วนหลักที่ 3 ก็จะติดสว่างตามข้อมูลที่ต้องการ

กระบวนการเริ่มต้นโดย PIC18F458 ส่งข้อมูลออกไปยังขาพอร์ตที่ต่ออยู่กับเซกเมนต์ a-g และ db (ในกรณีที่ต้องการใช้ขา db) ของแอลอีดีตัวเลข 7 ส่วนก่อน จากนั้นจึงส่งข้อมูล “0” ไปยังขาร่วมของแอลอีดีตัวเลข 7 ส่วนในหลักที่ต้องการให้แสดงผล เช่นถ้าต้องการแสดงตัวเลข 1234 ต้องส่งข้อมูลของเลข 1 ไปก่อนแล้วจึงส่ง “0” ไปยังขาร่วมของหลักที่ 4 จากนั้นจึงส่งข้อมูลเลข 2 แล้วส่ง “0” ไปยังขาร่วมของหลักที่ 3 ทยอยไล่ไปตามลำดับด้วยความเร็วสูง ในอัตราที่ตาของมนุษย์ไม่สามารถสังเกตเห็นความเปลี่ยนแปลงดังกล่าว ภาพที่เห็นจึงกลายเป็นตัวเลข 1234 ติดพร้อมกันทุกหลัก จะเห็นได้ว่าด้วยกระบวนการนี้แอลอีดีตัวเลข 7 ส่วนจะทำงานไม่พร้อมกัน การกินกระแสไฟฟ้าจึงมีค่าสูงสุดเท่ากับแอลอีดีทุกเซกเมนต์ในหนึ่งหลักติดสว่างพร้อมกันเท่านั้น



รูปที่ จ.21 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP และโมดูลแสดงตัวเลข 7 ส่วน

```
        List  p=18f458
              #include <p18F458.inc>
COUNT1        equ     0x20
COUNT2        equ     0x21
TEMP          equ     0x23
              ORG     0x0000
              CLRF    TRISA
              CLRF    TRISB
              CLRF    PORTB
CLRTEMP       CLRF    TEMP
LOOP          MOVF    TEMP, W
              CALL    TABLE1
              BCF     STATUS, Z
              ANDLW   0xFF
              BTFSS   STATUS, Z
              GOTO    CLRTEMP
              MOVWF   PORTB
              MOVLW   0x00
              MOVWF   PORTA
              CALL    DELAY
              INCF    TEMP, F
              GOTO    LOOP
TABLE2        ADDWF   PCL,F
              DT      3F,06,5B,4F,66,6D,7D,07,7F,6F,00
DELAY         MOVLW   0X20
              MOVWF   COUNT1
DELAY1        MOVLW   0XFF
              MOVWF   COUNT2
DELAY2        DECFSZ  COUNT1,1
              GOTO    DELAY1
              RETURN
              END
```

รูปที่ จ.22 โปรแกรมการทดลองแสดงผลตัวเลข 7 ส่วน

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ได้ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.22 ทำการคอมไพล์ให้เป็น Hex File

2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

3. ต่อวงจรตามรูปที่ จ.21

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

.....

.....

.....

5. ลองเปลี่ยนโปรแกรมจาก DT 3F, 06, 5B, 4F, 66, 6D, 7D, 07, 7F, 6F, 00 เป็น 6F, 7F, 07, 7D, 6D, 66, 4F, 5B, 06, 3F จากนั้นทำการคอมไพล์ และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้ง

6. จากการทดลองข้อ 5 เมื่อรันโปรแกรมบันสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลการทดลอง

.....

.....

.....

7. จากโปรแกรมการทดลองในรูปที่ จ.22 เปลี่ยนข้อมูลจาก MOVLW 0x00 มาเป็น MOVLW 0x01 ผลจากการสังเกตเป็นดังนี้

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายหลักการแสดงผลของแอลอีดีตัวเลข 7 ส่วนแบบคอมมอนคาโถดและอาโนด
2. จงอธิบายความแตกต่างระหว่าง แอลอีดีตัวเลข 7 ส่วนแบบคอมมอนคาโถดและอาโนด

ใบงานที่ 9

การทดลองใช้งาน PIC18F458 ร่วมกับสวิตช์เมตริกซ์ 4x4

จุดประสงค์

1. เพื่อให้สามารถเชื่อมต่อ PIC18F458 กับสวิตช์เมตริกซ์ (Switch Matrix) ขนาด 4x4 ได้
2. เพื่อให้สามารถเขียนโปรแกรมเพื่อรับค่าของสวิตช์ได้
3. เพื่อให้สามารถแก้ไขดัดแปลงโปรแกรมจากการทดลองได้
4. เพื่อให้สามารถสรุปและวิจารณ์ผลการทดลองเทียบกับทฤษฎีได้

อุปกรณ์ที่ใช้ในการทดลอง

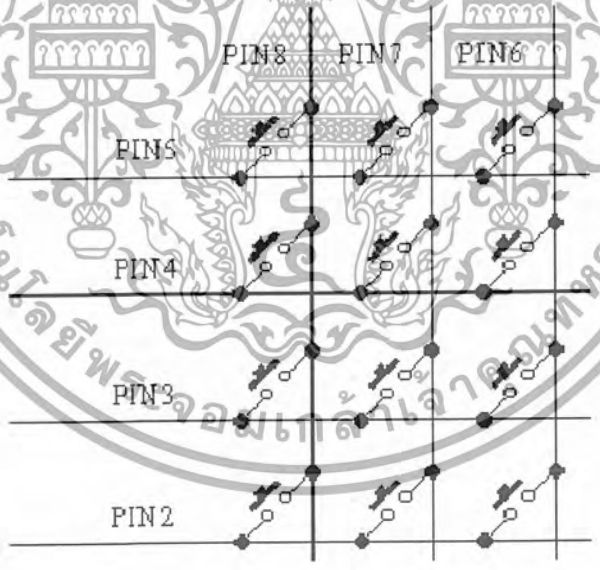
3. โมดูลหลัก PIC-ICSP
4. โมดูลสวิตช์พื้นฐาน
5. คอมพิวเตอร์พีซีที่ติดตั้งโปรแกรม MPLAB และ Epic Win
6. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์พีซี
7. สายเชื่อมต่อพอร์ตระหว่างโมดูล

ทฤษฎีเบื้องต้น

ไมโครคอนโทรลเลอร์สามารถรองรับและเชื่อมต่อใช้งานในการอ่านค่าและการกดสวิตช์ ได้วงจรของการกดสวิตช์มีการต่ออยู่ 2 แบบใหญ่ๆ คือ ต่อเข้ากับไฟเลี้ยงหรือต่อลงกราวด์โดยตรง เมื่อสวิตช์ตัวใดต่อวงจรสามารถอ่านค่าได้โดยตรง สามารถอ่านค่าได้ง่ายและรวดเร็วแต่มีข้อเสียคือ ถ้าจำนวนสวิตช์มีค่ามากๆ จำนวนของสายข้อมูลก็จะมีมากตาม ทำให้ระบบหรือวงจรมีขนาดใหญ่ และทำให้สิ้นเปลืองอุปกรณ์ด้วยวงจรของสวิตช์อีกลักษณะหนึ่งคือ การต่อวงจรแบบเมตริกซ์สวิตช์ จะถูกต่อกันในแนวแกนตั้งและแกนนอนจะเรียกแนวตั้งว่าหลักหรือคอลัมน์ (Column) ในขณะที่ แนวนอนจะเรียกว่า แถวหรือโรว์ (Row) ดังนั้นค่าของสวิตช์จะต้องประกอบด้วย ตำแหน่งในแนว หลักและแถว กระบวนการที่จะทำให้ได้มาซึ่งค่าของสวิตช์มีขั้นตอนที่ซับซ้อนพอสมควร

แต่วงจรของสวิตช์ แบบนี้มีข้อดีคือสามารถรองรับการเพิ่มของสวิตช์ได้อย่างสะดวก เพียงเพิ่มเติมจำนวนสวิตช์และแก้ไขซอฟต์แวร์อีกเล็กน้อยเท่านั้นทำให้วงจรสวิตช์แบบเมตริกซ์เป็นที่นิยมใช้กันมากในระบบควบคุมอัตโนมัติที่มีจำนวนสวิตช์มากกว่า 8 ตัว ในการใช้งานทั่วไปจะเรียกสวิตช์แบบเมตริกซ์นี้ว่าคีย์แพด (Keypad) สายสัญญาณในแนวคอลัมน์ของคีย์แพดจะต่อเข้ากับพอร์ตเอาต์พุตส่วนสายสัญญาณในแนวแถวจะต่อเข้ากับพอร์ตอินพุตไมโครคอนโทรลเลอร์จะส่งสัญญาณ “0” ออกมาทางพอร์ตเอาต์พุต ถ้าหากสวิตช์ถูกกดพอร์ตอินพุตจะอ่านค่า “0” เข้าสู่ไมโครคอนโทรลเลอร์

จากหลักการนี้ถ้าหากคีย์แพดมีจำนวนแถวและหลักมากกว่า 1 เส้นขั้นตอนการอ่านค่าสวิตช์ที่ถูกกดเริ่มต้นไมโครคอนโทรลเลอร์ส่งสัญญาณออกมาทางพอร์ตเอาต์พุตเข้าสู่แต่ละคอลัมน์ของคีย์แพดไล่เรียงกันไปทีละคอลัมน์แล้วอ่านค่าจากสายสัญญาณแต่ละหลักทางพอร์ตอินพุต ถ้าหากค่าของอินพุตที่อ่านเข้ามามีค่าไม่เท่ากับค่าของเอาต์พุตที่ส่งออกไปหรือยังเป็นค่าเดิม แสดงว่ายังไม่มีการกดสวิตช์ เมื่อใดที่ค่าของอินพุตอ่านเข้ามามีค่าเท่ากับค่าของเอาต์พุตที่ส่งออกไป แสดงว่ามีการกดสวิตช์เกิดขึ้น

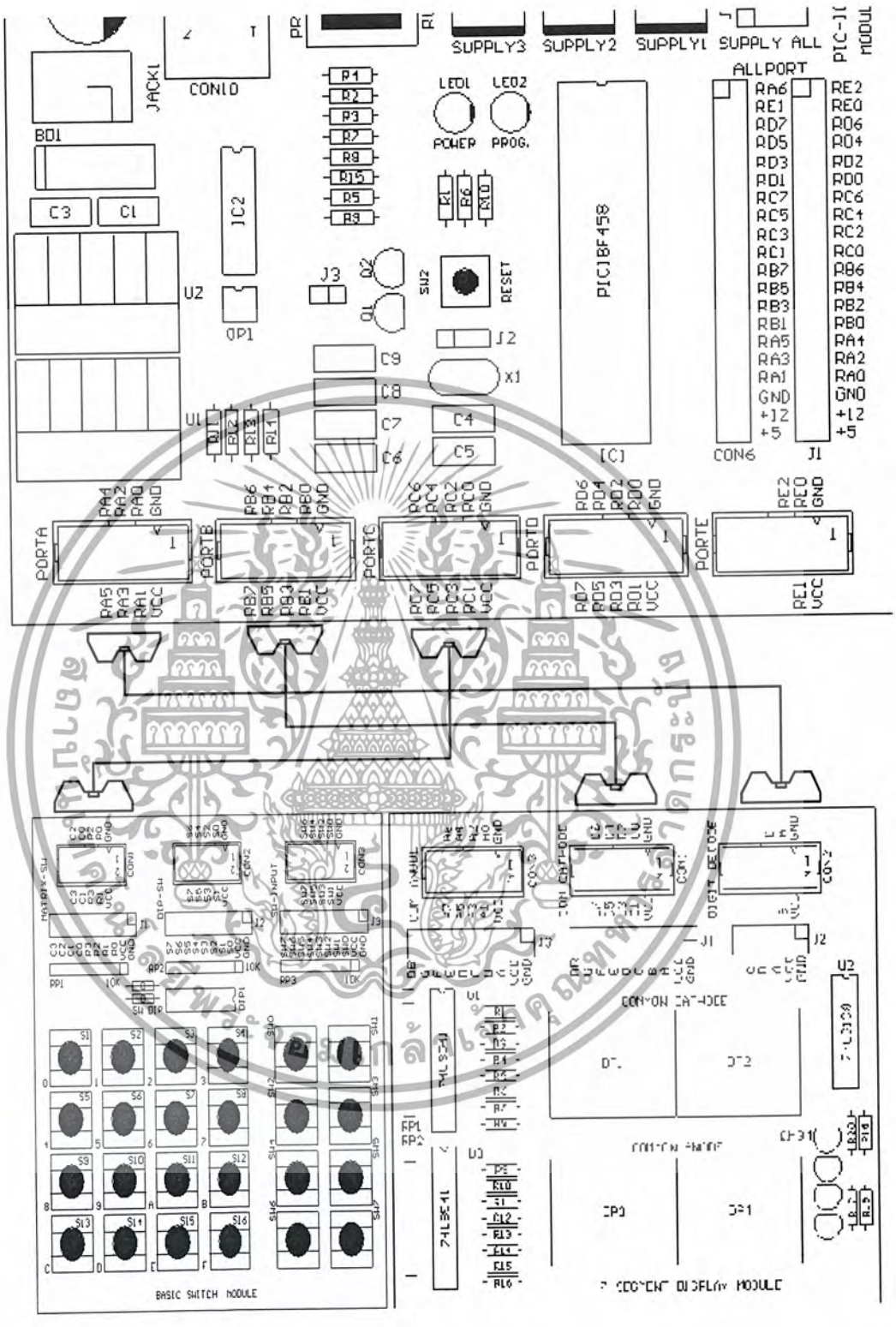


รูปที่ จ. 23 การต่อสวิตช์ เมตริกซ์ขนาด 4x4

ขั้นตอนการค้นหาตำแหน่งของสวิตช์ที่ถูกกด สายสัญญาณของแถวที่ต่อยู่กับสวิตช์ที่ถูกกดนั้นจะเป็นลอจิก "0" ไมโครคอนโทรลเลอร์จะนำค่าของเอาต์พุตที่ส่งออกไปและค่าของอินพุตที่อ่านเข้ามาทำการเปิดตารางเพื่อหาค่าของตำแหน่งสวิตช์ด้วยกระบวนการการเขียโปรแกรมการอ่านค่าสวิตช์แบบเมตริกส์จะต้องส่งค่า "0" ในแถวออกไปก่อนโดยในแถวแรกจะต้องกำหนดค่าตัวบวกเท่ากับ "0" จากนั้นอ่านค่าในแนวหลักกลับมา ถ้าไม่มีการกดคีย์ใดๆ ในแถวนั้นค่าที่อ่านได้จะเป็นค่าเดิมคือค่าลอจิก 1 อันเป็นผลมาจากค่าความต้านทานที่ต่อพูลอัปเอาไว้ จากนั้นก็ทำการอ่านค่าในแถวถัดไปพร้อมทั้งเพิ่มค่าตัวบวกเข้าไปด้วยจะทำเช่นนี้ไปจนครบทุกแถวแล้วกลับมาเริ่มต้นที่แถวแรกใหม่จนกว่าจะมีการกดคีย์เมื่อมีการกดีย์เกิดขึ้นไมโครคอนโทรลเลอร์จะนำค่าของคอลัมน์ที่มีการกดคีย์มีค่าอยู่ระหว่าง 4 กลับไปบวกกับค่าของแถวนั้น เช่น มีการกดคีย์ 5 เกิดขึ้น ดังนั้นถ้าสแกนคีย์ไปในแถวที่ 1 จะพบว่ามีการกดคีย์ในคอลัมน์ที่ 1 ค่าที่อ่านได้จะมีค่าเท่ากับ 4+1 = 5 ดังแสดงในตารางที่ จ.12 แสดงการค้นหาตำแหน่งของสวิตช์ที่ถูกกด

ตารางที่ จ.12 สรุปการค้นหาตำแหน่งของสวิตช์ที่ถูกกด

รหัสของคีย์	ตัวนับคอลัมน์	ตัวบวก	ข้อมูลของคีย์
0 = 1110	0	0	0
1 = 1110	1	0	1
2 = 1110	2	0	2
3 = 1110	3	0	3
4 = 1101	0	4	4
5 = 1101	1	4	5
6 = 1101	2	4	6
7 = 1101	3	4	7
8 = 1011	0	8	8
9 = 1011	1	8	9
A = 1011	2	8	A(10)
B = 1011	3	8	B(11)
C = 0111	0	12	C(12)
D = 0111	1	12	D(13)
E = 0111	2	12	E(14)
F = 0111	3	12	F(15)



รูปที่ จ. 23 การเชื่อมต่อโมดูลหลัก PIC-ICSP กับเมตริกซ์สวิตช์แสดงผลที่ LED ตัวเลข 7 ส่วน

```
;********** lab 9 *****************
        List  p = 18F458
        #Include <p18F458.inc>
__config _WDT_OFF & _PWRTE_ON & _LVP_OFF & _XT_OSC
COUNT       equ     0x21
COUNTD      equ     0x22
            org     0x000
            movlw   0x07
            movwf   ADCON1
            movlw   0x0f
            movwf   TRISA
            movlw   b'00000000'
            movwf   TRISC
            movlw   b'11111111'
            movwf   TRISD
            movlw   0xFF
            movwf   PORTB
Start       bcf     PORTB, 0
            call    TESTSW
            bsf     PORTB, 0
            bcf     PORTB, 1
            call    TESTSW
            bsf     PORTB, 1
            bcf     PORTB,2
            call    TESTSW
            bsf     PORTB, 2
            bcf     PORTB, 3
            call    TESTSW
            bsf     PORTB, 3
            goto    Start
TESTSW      movlw   0x00
            movwf   COUNT
            btfss   PORTB, 4
            call    Display
            incf    COUNT, F
            btfss   PORTB, 5
            goto    Display
            incf    COUNT, F
            btfss   PORTB, 6
            goto    Display
```

```
                incf    COUNT, F
                btfss   PORTB, 7
                goto    Display
                return
Display         clrf    PORTA
                movf    COUNT, W
                call    TABLE
                movwf   PORTC
                call    Delay
                movlw   0x01
                movwf   PORTA
                return
TABLE           addwf   PCL, F
                dt      3f, 06, 5b, 4f, 66, 6d, 7d, 07
                dt      7f, 6f, 77, 7c, 39, 5e, 79, 71, 00
Delay           movlw   0x02
                movwf   COUNTD
Delay0          decfsz  COUNTD, f
                Goto    Delay0
                Return
                end
```

รูปที่ จ.24 โปรแกรมการทดลองอ่านค่าคีย์แพดโดยใช้หลักการสแกนคีย์

ลำดับขั้นตอนการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB เขียนโปรแกรมดังรูปที่ จ.24 ทำการคอมไพล์
2. ต่อสายเชื่อมต่อระหว่างโมดูลตามรูปที่ จ.23
3. เขียนข้อมูลลงบน PIC18F458 ทำการรันโปรแกรมบน PIC18F458 เลื่อนสวิตช์ไปที่ RUN
4. เมื่อกดสวิตช์แต่ละตัว ผลที่ได้คือ

ตารางที่ จ.13 ผลการติดของไดโอดเปล่งแสงตัวเลข 7 ส่วนจากการกดสวิตช์

กดสวิตช์	แสดงผลที่ตัวเลข 7 ส่วน
0	.....
1	.....
2	.....
3	.....
4	.....
5	.....
6	.....
7	.....
8	.....
9	.....
A	.....
B	.....
C	.....
D	.....
E	.....
F	.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายหลักการทำงานของโปรแกรมดังรูปที่ จ.24
2. จงอธิบายหลักการทำงานของสวิตช์ เมตริกซ์ขนาด 4x4 มาพอเข้าใจ

ใบงานที่ 10
การทดลองใช้งาน PIC18F458 ขับสเตปปิ้งมอเตอร์

จุดประสงค์

1. เพื่อให้สามารถอธิบายการทำงานของสเตปปิ้งมอแตอร์ได้
2. เพื่อให้สามารถใช้งาน PIC18F458 ร่วมกับและอุปกรณ์ขับกระแสได้
3. เพื่อให้สามารถเขียนโปรแกรมควบคุมการหมุนของสเตปปิ้งมอเตอร์ได้
4. เพื่อให้สามารถสรุปและวิจารณ์ผลการทดลองเทียบกับทฤษฎีได้

อุปกรณ์ที่ใช้ในการทดลอง

1. คอมพิวเตอร์พีซีที่ติดตั้งโปรแกรม MPLAB และ EPIC win
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์พีซี
3. สายเชื่อมต่อพอร์ตระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
5. โมดูลการขับสเตปปิ้งมอเตอร์

ทฤษฎีเบื้องต้น

เมื่อต้องการขับอุปกรณ์ที่ต้องใช้กระแสและแรงดันสูงๆ เกินกว่าไมโครคอนโทรลเลอร์ PIC18F458 ขับได้โดยตรงดังนั้น ต้องมีการขับกระแสและแรงดันผ่านตัวอุปกรณ์ตัวหนึ่งที่เรียกว่า อุปกรณ์ไดรเวอร์ (Driver) ซึ่งอุปกรณ์ที่นิยมนำมาใช้คือไอซี ULN2003

สเตปปิ้งมอเตอร์แบบยูนิโพล่าร์ (Uni-polar Stepping Motor)

สเตปปิ้งมอเตอร์ที่นิยมใช้มากที่สุดและหาได้ง่ายคือ สเตปปิ้งมอเตอร์แบบยูนิโพล่าร์ มีลักษณะการพันขดลวดของมอเตอร์สเตปปิ้งมอเตอร์แบบนี้มีการพันขดลวด 2 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ แต่ละขดลวดแบ่งเป็น 2 เฟส รวมมอเตอร์ทั้งตัวจะมี 4 เฟสคือ เฟส 1, 2, 3 และ 4 มีการต่อสายออกมาจากขดลวดแต่ละขดเพื่อจ่ายไฟเลี้ยง ทำให้สเตปปิ้งมอเตอร์แบบนี้มีทั้งแบบ 5 สายและ 6 สาย ถ้าเป็นแบบ 6 สายจะเป็นการนำสายไฟเลี้ยงของขดลวดทั้งสองมาต่อรวมกันเป็นสายเดียว

การกระตุ้นและควบคุมการหมุนของสเตปปิ้งมอเตอร์

การกระตุ้นและการควบคุมการหมุนของสเตปปิ้งมอเตอร์ให้เคลื่อนที่ไปในแต่ละสเตป ทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ (Stater) ซึ่งต้องป้อนเป็นแบบซีเควน-เชียล (Sequential) ในรูปแบบที่ถูกต้องด้วย สามารถแบ่งได้เป็น 3 รูปแบบคือ แบบเวฟ (Wave), แบบ 2 เฟส (Two phase) และแบบครึ่งสเตป (Half Wave)

แบบเวฟ เป็นการกระตุ้นที่มีรูปแบบที่ง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งไล่เรียงถัดกันไป เช่น เริ่มต้นที่ขดที่ 1, 2, 3 และ 4 แล้ววนกลับมาที่ 1 วนไปเรื่อยๆ หรือเริ่มที่ขดที่ 4 แล้วย้อนกลับไปขดที่ 3, 2 และ 1 วนกลับไปเรื่อยๆ ซึ่งทำให้ทิศทางของการหมุนสวนทางกันในการกระตุ้นรูปแบบนี้จึงมีขดลวดเพียงขดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้นวงจรกระตุ้นแบบเวฟจึงมีราคาถูกและทำการกระตุ้นได้ง่าย ขั้นตอนการทำงานต่างๆ ดังตารางที่ จ.14

ตารางที่ จ.14 ลำดับการทำงานของสเตปปิ้งมอเตอร์ได้รับการกระตุ้นแบบเวฟ

สเตปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

แบบ 2 เฟส เป็นการกระตุ้นซึ่งคล้ายกับแบบเวฟแต่การกระตุ้นแบบนี้จะทำการกระตุ้นโดยจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขดที่อยู่ใกล้กันในเวลาเดียวกัน และเรียงถัดกันไปเช่นเดียวกับแบบเวฟ เช่น ขดลวดชุดแรกที่ถูกกระตุ้นจะเป็นขดที่ 1 และ 2 ตามด้วยการกระตุ้นขดที่ 2 และ 3 ต่อไปก็เป็นขดที่ 3 และ 4 ถัดไปก็เป็นขดที่ 4 และ 1 แล้วกลับมาที่ขดที่ 1 และ 2 วนไปเรื่อยๆ หรือเริ่มจากที่ขดที่ 1 และ 4 ตามด้วยขดที่ 4 และ 3 ต่อไปก็เป็นขดที่ 3 และ 2 ถัดไปก็เป็นขดที่ 2 และ 1 และวนกลับมาที่ 1 และ 4 วนไปเรื่อยๆ แต่การหมุนจะกลับทิศทางกัน การกระตุ้นแบบนี้สามารถเพิ่มแรงบิดได้มากกว่าแบบแรก โรเตอร์ (Roter) จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงด้วย 2 ขดลวดที่ถูกกระตุ้นพร้อมกัน และต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียในแบบนี้คือการกระตุ้นแบบนี้ต้องใช้กำลังไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่างๆ แสดงดังตารางที่ จ.15

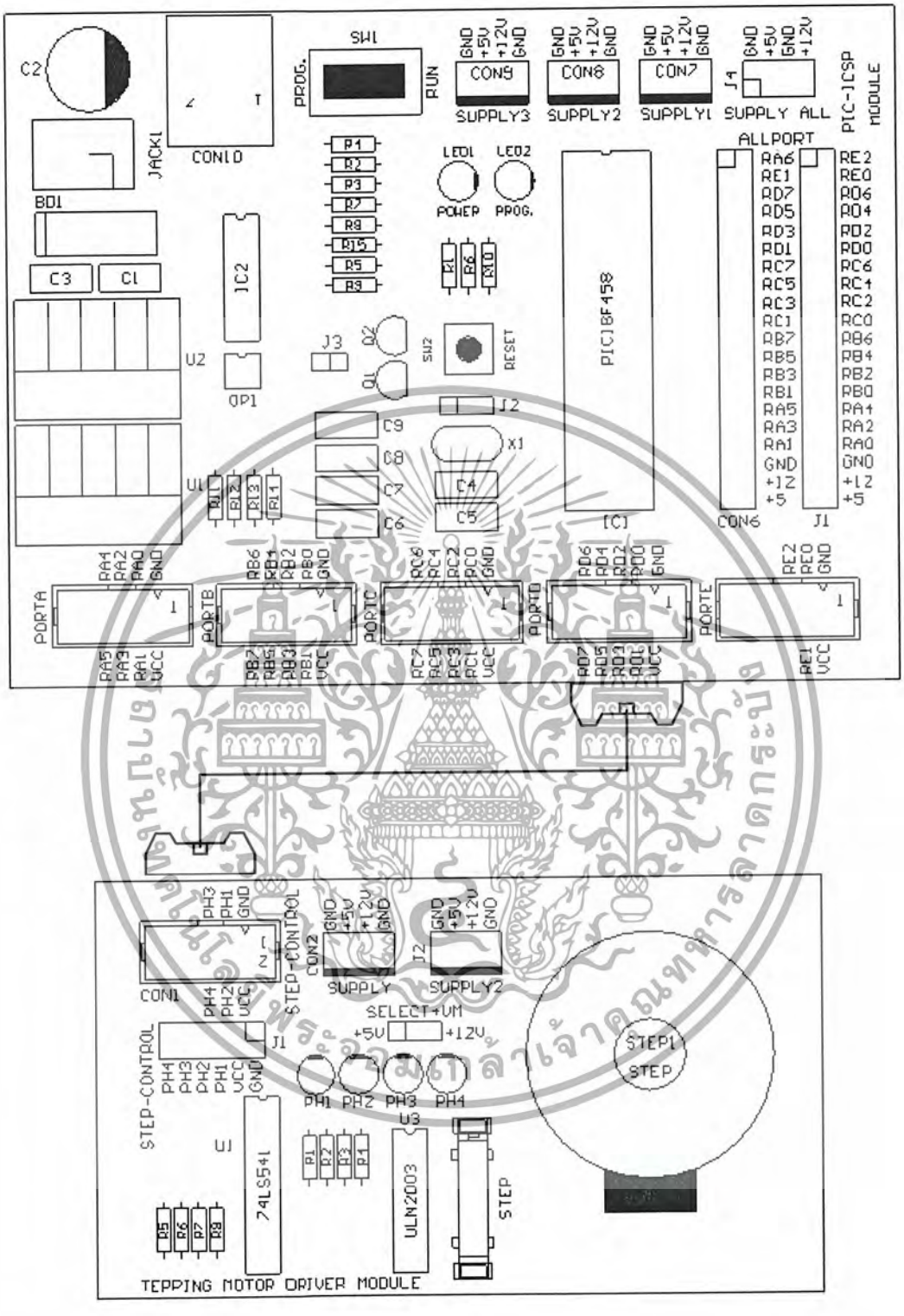
ตารางที่ จ.15 ลำดับการทำงานของสเตปปิ้งมอเตอร์ได้รับการกระตุ้นแบบสองเฟส

สเตปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

แบบครึ่งสเตป เป็นรูปที่ผสมระหว่างแบบเวฟและแบบ 2 เฟส เพื่อเพิ่มจำนวนของสเตปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นที่ขดลวดเรียงกันไปเป็นลำดับโดยเริ่มจากขดลวดที่ 1, 1 และ 2, 2, 2 และ 3, 3, 3 และ 4, 4, 4 และ 1 และวนกลับมายังขดที่ 1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีก เพราะช่วงสเตปมีระยะสั้นลง แต่ละสเตปเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น แต่ต้องระวังไว้อีกอย่างหนึ่งว่าเมื่อกระตุ้นให้ทำงานในรูปแบบนี้จะต้องทำการหมุนถึง 2 สเตป จึงจะได้เท่ากับระยะเท่ากับ 1 สเตปเต็มของการควบคุมใน 2 แบบแรก สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้ขนาดเท่ากับแบบ 2 เฟสเป็นอย่างน้อยจึงจะเพียงพอ ขั้นตอนการทำงานต่างๆ แสดงดังตารางที่ จ.16 ในการทดลองนี้เราจะกล่าวถึงเฉพาะสเตปปิ้งมอเตอร์แบบเวฟ และแบบ 2 เฟส ซึ่งแบบครึ่งสเตปนั้นก็จะคล้ายๆ กับ 2 แบบแรกรวมกัน ข้อมูลในการป้อนก็จะคล้ายๆ กัน

ตารางที่ จ.16 ลำดับการทำงานของสเตปปิ้งมอเตอร์ได้รับการกระตุ้นแบบครึ่งสเตป

สเตปที่	สเตปที่ 1	สเตปที่ 2	สเตปที่ 3	สเตปที่ 4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน	-	-
4	-	ทำงาน	ทำงาน	-
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	ทำงาน
8	ทำงาน	-	-	ทำงาน



รูปที่ จ.25 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับโมดูลขับสเตปปิ้งมอเตอร์

```
;*****************************************************************
; Filename :        lab10.asm
; Descripton :      Control Stepper Motor        [1 Phase]
; MCU :             PIC18F458
;*****************************************************************
        list p=18f458          ; list directive to define processor
        #include <p18f458.inc>    ; processor specific variable definitions
OFFSET  EQU     0x20
COUNT   EQU     0x21
COUNT1  EQU     0x22
COUNT2  EQU     0x23
        ORG     0x0000
        CLRF    TRISD          ; PORTD is output
ST      CLRF    OFFSET         ; OFFSET = 0
        MOVLW   .4
        MOVWF   COUNT                  ; COUNT = 4
START   MOVF    OFFSET,W       ; W <-- (OFFSET)
        CALL    TABLE          ; Read data form Table
        MOVWF   PORTD          ; Send to PORTD
        CALL    DELAY                  ; delay
        INCF    OFFSET,F
        INCF    OFFSET,F       ; Offset = Offset + 2
        DECFSZ  COUNT,F        ; COUNT = COUNT-1 and skip if COUNT =0
        GOTO    START          ;!= 0
        GOTO    ST
;******** Delay loop ********
DELAY   MOVLW   0x00
        MOVWF   COUNT1
DEL0    CLRF    COUNT2
DEL1    DECFSZ  COUNT2,F
        GOTO    DEL1
        DECFSZ  COUNT1,F
        GOTO    DEL0
        RETURN
;********* Tebel of data **********************
TABLE   ADDWF   PCL                    ; Move offset to PC lower
        DT      0x01,0x02,0x04,0x08    ; 0001 , 0010 , 0100 , 1000 <1 PHASE>
        END
```

รูปที่ จ. 26 โปรแกรมการทดลองขับสเตปิ้งมอเตอร์แบบเวฟ

```
;*****************************************************************
; Filename :          lab10-1.asm
; Descripton :        Control Stepper Motor        [2 Phase]
; MCU :               PIC18F458
;*****************************************************************
        list p=18f458          ; list directive to define processor
            #include <p18f458.inc>    ; processor specific variable definitions
  OFFSET    EQU   0x20
  COUNT     EQU   0x21
  COUNT1    EQU   0x22
  COUNT2    EQU   0x23
            ORG   0x0000
            CLRF      TRISD          ; PORTD is output
ST          CLRF      OFFSET         ; OFFSET = 0
            MOVLW     .4
            MOVWF     COUNT          ; COUNT = 4
START       MOVF      OFFSET,W       ; W <-- (OFFSET)
            CALL      TAB            ; Read data form Table
            MOVWF     PORTD          ; Send to PORTD
            CALL      DELAY          ; delay
            INCF      OFFSET,F
            INCF      OFFSET,F       ; Offset = Offset + 2
            DECFSZ    COUNT,F        ; COUNT = COUNT-1 and skip if COUNT =0
            GOTO      START          ; != 0
            GOTO      ST
;******** Delay loop ********
DELAY   MOVLW     0x00
        MOVWF     COUNT1
DEL0    CLRF      COUNT2
DEL1    DECFSZ    COUNT2,F
        GOTO      DEL1
        DECFSZ    COUNT1,F
        GOTO      DEL0
        RETURN
;********* Tebel of data **********************
TAB     ADDWF   PCL                     ; Move offset to PC lower
        DT      0x03,0x06,0x0C,0x09     ; 0011 , 0110 , 1100 , 1001
        END
```

รูปที่ จ.27 โปรแกรมการทดลองขับสเตปิ้งมอเตอร์แบบ 2 เฟส

ลำดับขั้นตอนการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.26 ทำการคอมไพล์ให้เป็น Hex File

2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

3. ต่อวงจรตามรูปที่ จ.25

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

.....

.....

.....

5. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.27 ทำการคอมไพล์ให้เป็น Hex File

6. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้งรันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายการทำงานของโปรแกรมรูปที่ จ.27

2. จากโปรแกรมรูปที่ จ.27 เป็นการกระตุ้นและควบคุมการหมุนสเตปปิ้งมอเตอร์แบบเวฟ เพราะเหตุใด

3. จงอธิบายความแตกต่างของการขับปิ้งมอเตอร์แบบเวฟกับแบบ 2 เฟส

ใบงานที่ 11
การทดลองใช้งาน PIC18F458 สร้างสัญญาณเสียง

วัตถุประสงค์

1. เพื่อให้สามารถอธิบายหลักการสร้างสัญญาเสียงได้
2. เพื่อให้สามารถประกอบวงจรสำหรับการทดลองได้
3. เพื่อให้สามารถแก้ไขโปรแกรมในการทดลองได้
4. เพื่อให้สามารถสรุปและวิจารณ์ผลการทดลองเทียบกับทฤษฎีได้

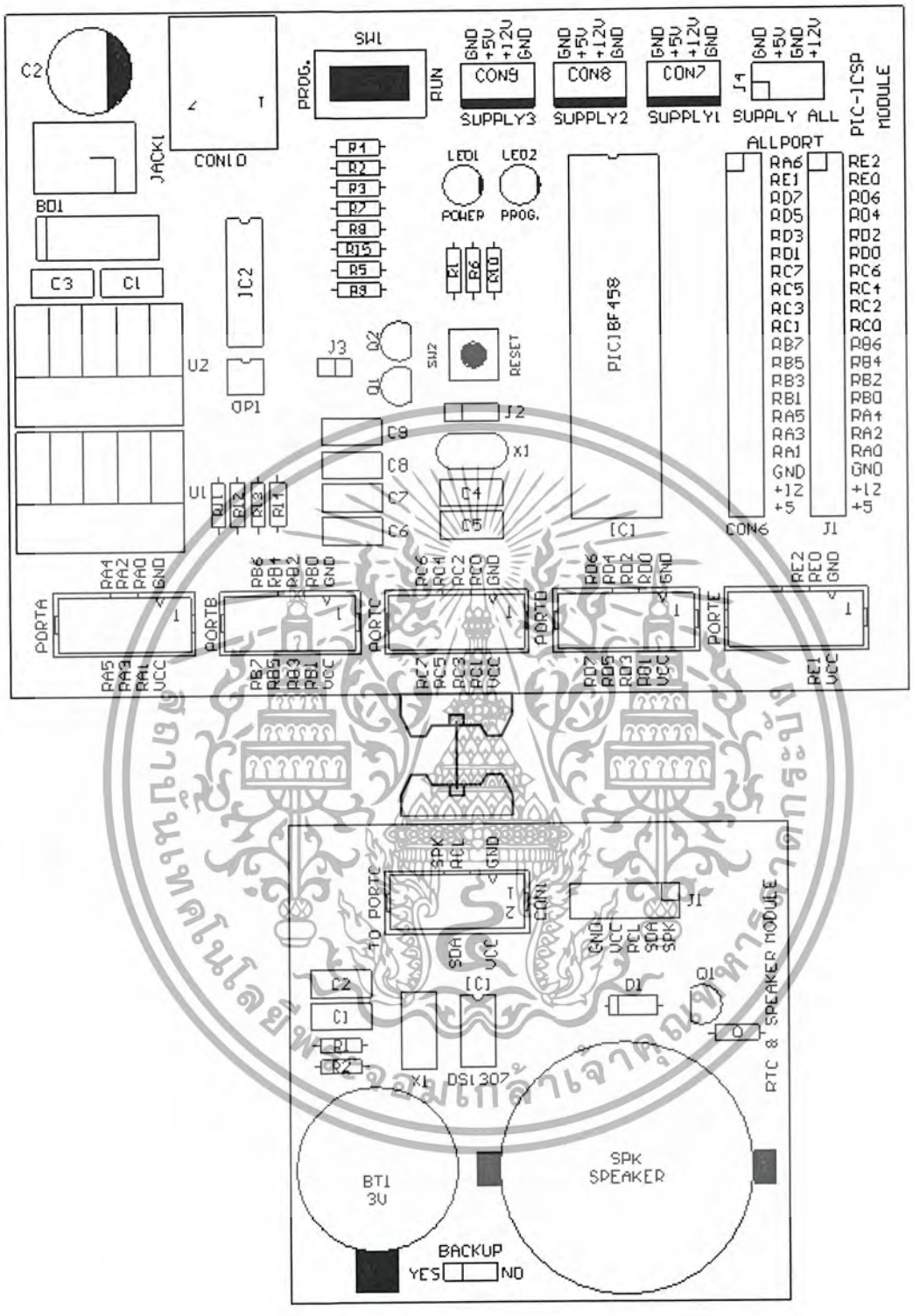
เครื่องมือและอุปกรณ์

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ EPICwin
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
5. โมดูล RTC/SPEAKER MODULE

ทฤษฎีเบื้องต้น

ลำโพง (Speaker)

เป็นอุปกรณ์ที่ทำหน้าที่กำเนิดสัญญาณเสียงโดยอาศัยหลักการป้อนความถี่ค่าต่างๆ ให้แก่ตัวมันทำให้ขดลวดที่อยู่ด้านในพองตัวสามารถกำเนิดสัญญาณเสียงออกมาได้โดยการเปลี่ยนสัญญาณทางไฟฟ้าเป็นสัญญาณเสียงในการใช้งาน PIC18F458 ใช้งานร่วมกันเพื่อสร้างสัญญาณเสียงนั้นจะใช้หลักการหน่วงเวลาค่าต่างๆ สามารถทำให้เกิดพัลส์ออกมาที่พอร์ตเอาต์พุตได้



รูปที่ จ.28 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP และโมดูล RTC/SPEAKER

```
;*************************************************

; Program    : Test  Speaker
; Filename   : lab11.asm
; MCU        : PIC 18F458

                                                    *

;*************************************************
            list p=18f458                ; list directive to define processor
           #include <p18f458.inc>        ; processor specific variable definitions
         #define   SPK    PORTC,5
        COUNT     EQU     0x21
        COUNT1    EQU     0x22
        COUNT2    EQU     0x23
        COUNT3    EQU     0x24
        ORG       0x0000
       CLRF      TRISC          ; PORTC is output
LOOP  CALL      PULSE          ; Generate Pulse
        BCF       SPK            ; Off speaker
        CALL      DELAY          ; delay
        GOTO      LOOP
;***** Generate Pulse *****
PULSE  CLRF    COUNT          ; COUNT = 0
PUL     BCF     SPK            ; Off Speaker
        CALL    DELAY          ; delay
        BSF     SPK            ; On Speaker
        CALL    DELAY          ; delay
          DECFSZ   COUNT,F
        GOTO    PUL
        RETURN

;******** Delay loop *********
DELAY   MOVLW   .8
          MOVWF   COUNT1
DEL1      CLRF      COUNT2
DEL2      DECFSZ    COUNT2
          GOTO      DEL2
          DECFSZ    COUNT1
          GOTO      DEL1
          RETURN
```

```
;****** Delay Loop 2 *******
DELAY2  MOVLW   .5
        MOVWF   COUNT1
DEL     CLRF    COUNT2
DEL_1   CLRF    COUNT3
DEL_2   DECFSZ  COUNT3
        GOTO    DEL_2
        DECFSZ  COUNT2
        GOTO    DEL_1
        DECFSZ  COUNT1
        GOTO    DEL
        RETURN
        END
```

รูปที่ จ. 29 โปรแกรมทดลองใช้งาน PIC 18F458 สร้างสัญญาณเสียง

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.29 ทำการคอมไพล์ให้เป็น Hex file
2. ต่อวงจรตามรูปที่ จ.28
3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์
4. รันโปรแกรมเลื่อนสวิตซ์มาที่ตำแหน่ง RUN
5. บันทึกผลการทดลอง

.....

.....

6. ทดลองเปลี่ยนโปรแกรมหน่วงเวลาเดิม .2 เปลี่ยนค่าเพิ่มขึ้นทีละ 1 ค่า ไปจนถึง .9 สังเกตผลการทดลองที่เกิดขึ้น บันทึกผลการทดลอง

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายหลักการสร้างสัญญาณเสียงของ PIC18F458

2. เมื่อต้องการเปลี่ยนระดับสัญญาณเสียงให้เป็นความถี่เสียงต่างๆ จะเปลี่ยนโปรแกรมที่ตำแหน่งไหน เพราะเหตุใด

3. จากการทดลองสามารถนำไปประยุกต์ใช้งานได้อย่างไรยกตัวอย่างการนำเอาไปประยุกต์ใช้งาน 1 ตัวอย่าง พร้อมอธิบาย

ใบงานที่ 12

การทดลองใช้งาน PIC18F458 ร่วมกับโมดูลแสดงผลแอลซีดี

วัตถุประสงค์

1. เพื่อให้สามารถต่อวงจรใช้งานแอลซีดีได้
2. เพื่อให้สามารถเขียนโปรแกรมเพื่อติดต่อใช้งานแอลซีดีได้
3. เพื่อให้สามารถแก้ไขโปรแกรมในการทดลองได้
4. เพื่อให้สามารถสรุปและวิจารณ์ผลการทดลองเทียบกับทฤษฎีได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ EPIC win
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
5. โมดูลแสดงผลแอลซีดี

ทฤษฎีเบื้องต้น

โมดูล LCD (Liquid Crystal Display)

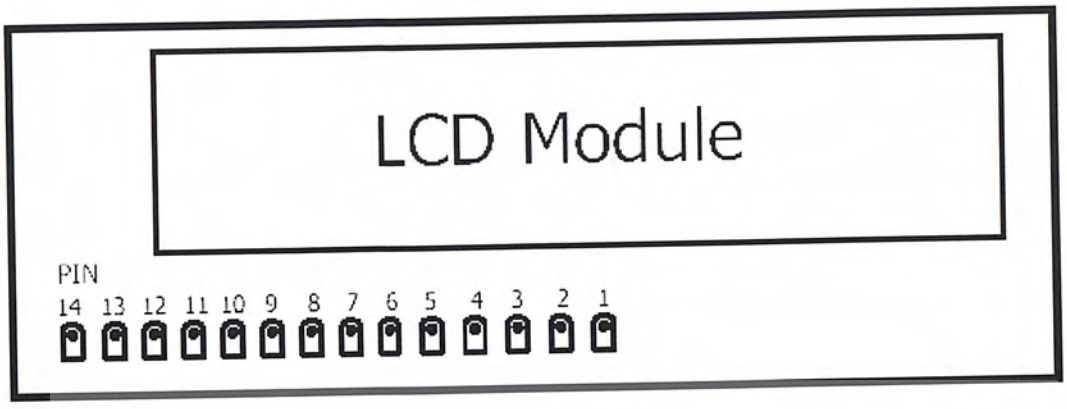
เป็นอุปกรณ์กำเนิดแสงชนิดหนึ่งที่นิยมใช้เป็นตัวแสดงผลข้อความหรือตัวเลขมากกว่านำไปใช้เป็นอุปกรณ์กำเหนิดแสงสว่าง เนื่องจากมันมีความเข้มของแสงต่ำมาก และกินกำลังไฟต่ำมากเช่นกัน จึงนิยมนำ LCD มาใช้ในเครื่องคิดเลขและนาฬิกาดิจิตอล LCD เป็นจอแสดงผลที่ได้รับความนิยมอย่างสูง ในปัจจุบัน LCD ถูกนำมาใช้งานแทนที่ 7-Segment เป็นจำนวนมาก เนื่องจาก LCD สามารถแสดงตัวอักษรและรายละเอียดมากกว่า 7-Segment ทำให้มีการนำไปใช้งานในเครื่องมือต่างๆ มากขึ้นเพราะการสื่อสารระหว่างผู้ใช้และเครื่องมือต่างๆ โดยผ่าน LCD นั้นมีความสะดวกซึ่งในที่นี้จะขอกล่าวถึงเฉพาะ LCD ที่ใช้ในการทดลอง ซึ่งเป็นแบบ 16 ตัวอักษร

LCD แบบตัวอักษร (Character) ขนาด 16 ตัวอักษร 2 บรรทัด

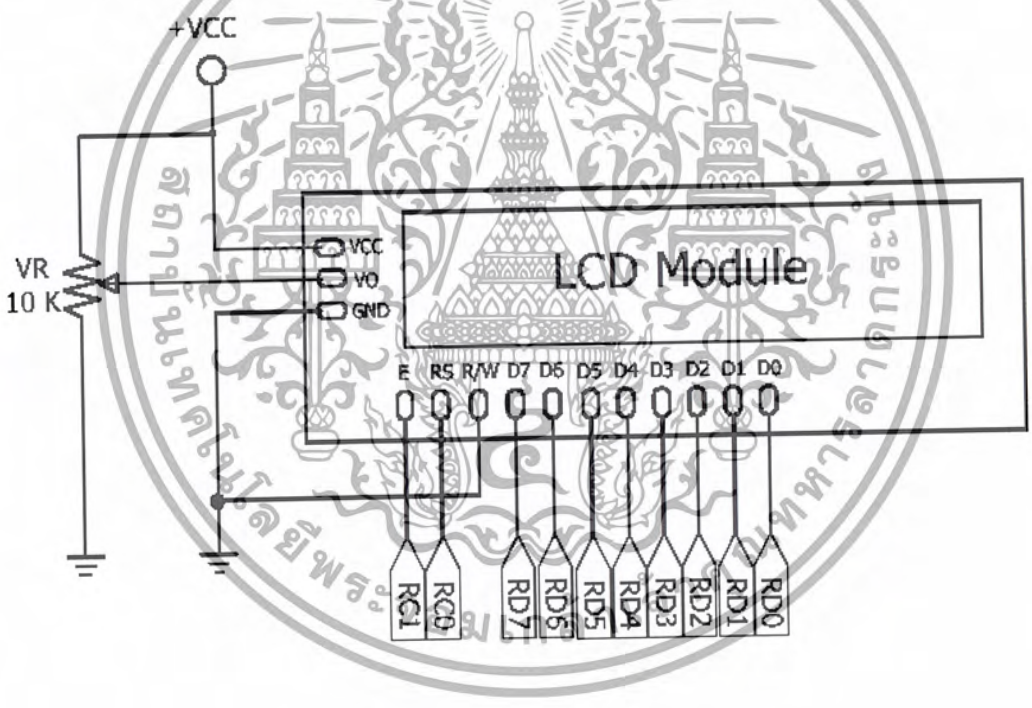
โมดูล LCD 16 ตัวอักษร 2 บรรทัด มีขาต่อใช้งาน 14 ขา ส่วนหน้าที่ของขาที่ใช้งานโมดูล LCD มีรายละเอียดดังตารางที่ จ.17

ตารางที่ จ.17 ชื่อและหน้าที่ของขาสัญญาณต่างๆ ของ LCD

ขาที่	ชื่อขา	หน้าที่การใช้งาน
1	GND	ต่อกับกราวด์ของวงจร
2	$+V_{DD}$	ต่อกับไฟเลี้ยง + 5 โวลต์
3	Vo	เป็นขาสำหรับป้อนแรงดันเพื่อปรับความสว่างของจอแสดงผล LCD
4	Rs	เป็นขาเลือกการติดต่อกับรีจิสเตอร์คำสั่งหรือรีจิสเตอร์ข้อมูล 0 : จะติดต่อกับรีจิสเตอร์คำสั่ง 1 : จะติดต่อกับรีจิสเตอร์ข้อมูลเพื่อนำข้อมูลไปแสดงผล
5	R/W	เป็นขาเลือกการอ่านหรือเขียนข้อมูลกับโมดูล LCD
6	E	เป็นขาสำหรับป้อนสัญญาณพัลส์เอ็นเอเบิลให้โมดูล LCD ทำงาน
7-14	D0-D7	เป็นขาข้อมูล 8 บิต โดยใช้ขา 7 คือ D0 ไล่ไปจนถึงขา 14 คือ D7



รูปที่ จ.30 การจัดขาของโมดูลแอลซีดีแบบอักขระ 16 ตัวอักษร 2 บรรทัด



รูปที่ จ. 31 ตัวอย่างการเชื่อมใช้งานโมดูลแสดงผลแอลซีดี 16 ตัวอักษร 2 บรรทัด

การติดต่อกับโมดูล LCD 16 ตัวอักษร 2 บรรทัด

มีอยู่ด้วยกัน 2 แบบ คือ แบบ 4 บิต และ แบบ 8 บิต โดยปกติจะใช้งานแบบ 8 บิตมากกว่า แต่หากมีข้อจำกัดเรื่องจำนวนของพอร์ต ควรเลือกการใช้งานแบบ 4 บิต ซึ่งจะมีขั้นตอนเพิ่มขึ้นเพียงเล็กน้อย แต่จะใช้สายสัญญาณเพียง 6 เส้น ในขณะที่ 8 บิต จะใช้ทั้งหมด 8 เส้น

การติดต่อแบบ 8 บิต

การเชื่อมต่อโมดูล LCD กับ CPU แบบ 8 บิต ขา D0-D7 ของโมดูล LCD เชื่อมต่อกับขา RD0-RD7, ขา RS ต่อกับ RC0 และ E ส่วนขา R/W ให้ต่อลงกราวด์ เพื่อให้แอลซีดีทำงานในลักษณะเขียนข้อมูลอย่างเดียว ในขณะที่ขา Vo ต่อกับตัวต้านทานปรับค่าได้ 10 โอห์ม เพื่อปรับความสว่างของจอแสดงผล

ลำดับขั้นตอนในการเขียนโปรแกรมเพื่อใช้งานโมดูล LCD

ในการเขียนโปรแกรมควบคุมให้จอแอลซีดีแสดงผลนั้น ในขั้นแรกจะต้องทำการกำหนดฟังก์ชั่นการทำงานต่างๆ ของแอลซีดีเสียก่อน หรือ เรียกว่าการอินิเชียลแอลซีดี (Initial LCD) ซึ่งก็คือการเขียนข้อมูลคำสั่งไปยังรีจิสเตอร์คำสั่งภายในโมดูล LCD เพื่อเตรียมความพร้อมให้แก่โมดูล LCD ซึ่งในการอินิเชียล LCD ก็คือ การกำหนดให้แอลซีดีมีการทำงานในรูปแบบต่างๆ เช่น กำหนดตำแหน่งของเคอร์เซอร์มาอยู่ที่จุดเริ่มต้นที่ตำแหน่งซ้ายมือสุด, เปิดจอแสดงผล, เปิด-ปิดเคอร์เซอร์, กำหนดรูปแบบการแสดงผลของตัวอักษร, กำหนดจำนวนบรรทัด และการกำหนดโหมดในการติดต่อ โดยในการส่งข้อมูลไปยัง LCD จะแบ่งข้อมูลออกเป็น 2 ชนิด คือ ข้อมูลคำสั่ง (Command) และข้อมูลในการแสดงผล (Data) ซึ่งการส่งข้อมูลทั้งสองชนิดจะมีลำดับขั้นตอนเหมือนกันแต่จะต่างกันตรง การกำหนดข้อมูลที่บิต RS เพื่อแยกชนิดของข้อมูลซึ่งลำดับขั้นตอนดังนี้

การเขียนข้อมูลคำสั่งไปยังจอแอลซีดี

1. ทำให้ขา RS เป็น “0” เพื่อแจ้งให้โมดูล LCD ทราบว่า ข้อมูลที่ขา Data เป็นข้อมูลคำสั่ง
2. ส่งข้อมูลคำสั่งที่ต้องการไปยังขา Data ทั้ง 8 เส้น
3. ส่งพัลส์ Enable ไปยังขา E

การเขียนข้อมูล (DATA) เพื่อแสดงข้อมูล

1. ทำให้ขา RS เป็น “1” เพื่อแจ้งให้ LCD ทราบว่า ข้อมูลที่ขา DATA เป็นข้อมูลที่จะแสดงผล
2. เขียนข้อมูลที่ต้องการไปยังขา Data ทั้ง 8 เส้น
3. ส่งพัลส์ Enable ไปยังขา E

การติดต่อแบบ 4 บิต

ในการต่อแบบ 4 บิตจะใช้ขาของข้อมูลที่ใช้มีเพียง 4 เส้น คือ D4-D7 ซึ่งต่อเข้ากับ RD4-RD7 ของ CPU สำหรับขา D0-D3 ของโมดูล LCD ให้ต่อลงกราวด์ ส่วนขา RS ต่อเข้ากับ RC0 และ E ต่อเข้ากับ RC1 จุดที่แตกต่างจากการติดต่อแบบ 8 บิตในการเขียนโปรแกรมคือ ต้องทำการส่งข้อมูล 2 ครั้ง คือส่ง 4 บิตบนของข้อมูลก่อน จากนั้นจึงส่งข้อมูล 4 บิตล่างตามไปสำหรับการอินิเชียลนั้นมีสิ่งที่ต้องทำก่อนเสมอ คือ ต้องส่งข้อมูล 03h (0011) ออกไปให้ LCD ที่ D7-D4 แล้วทำการส่งสัญญาณ Enable จำนวน 2 ครั้ง เพื่อจัดสถานะการทำงานของแอลซีดี จากนั้นส่งข้อมูลคำสั่ง 02h ออกไปที่ขา D7-D4 แล้ว Enable อีกเช่นกันเพื่อกำหนดให้แอลซีดีทำงานในโหมด 4 บิต เท่านี้แอลซีดีก็พร้อมที่จะทำงานในโหมด 4 บิตแล้ว ส่วนการกำหนดค่าอื่นๆ สามารถทำ ต่อจากนี้ได้เลยแต่ต้องส่งข้อมูลในแบบ 4 บิต

การเขียนข้อมูลคำสั่งไปยังจอแอลซีดี

1. ทำให้ขา RS เป็น "0" เพื่อแจ้งให้โมดูล LCD ทราบว่า ข้อมูลที่ขา DATA เป็นข้อมูลคำสั่ง
2. ส่งข้อมูลคำสั่ง 4 บิตบน (บิต 7 ⟶ บิต 4) ที่ต้องการ ไปยังขา Data ทั้ง 4 เส้น
3. ส่งพัลส์ Enable ไปยังขา E
4. ส่งข้อมูลคำสั่ง 4 บิตล่าง (บิต 3 ⟶ บิต 0) ที่ต้องการ ไปยังขา Data ทั้ง 4 เส้น
5. ส่งพัลส์ Enable ไปยังขา E

การเขียนข้อมูล (DATA) เพื่อแสดงผลข้อมูล

1. ทำให้ขา RS เป็น "1" เพื่อแจ้งให้โมดูล LCD ทราบว่า ข้อมูลที่ขา DATA เป็นข้อมูลในการแสดงผล
1. ส่งข้อมูล (Data) 4 บิตบน (บิต 7 ⟶ บิต 4) ที่ต้องการ ไปยังขา Data ทั้ง 4 เส้น
2. ส่งพัลส์ Enable ไปยังขา E
3. ส่งข้อมูล (Data) 4 บิตล่าง (บิต 3 ⟶ บิต 0) ที่ต้องการ ไปยังขา Data ทั้ง 4 เส้น
4. ส่งพัลส์ Enable ไปยังขา E

คำสั่งในการควบคุมการทำงานของ LCD

1. คำสั่งเคลียร์หน้าจอแสดงผล

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

ข้อมูลคำสั่งคือ 0x01 ซึ่งเมื่อส่งข้อมูลคำสั่งนี้ไปยัง LCD จะเป็นการเขียนข้อมูลที่เป็นช่องว่างเข้าไปยังหน่วยความจำ DDRAM ซึ่งเป็นพื้นที่ที่ใช้เก็บข้อมูลที่แสดงผลบน LCD จะทำให้จอแสดงผล LCD อยู่ในลักษณะจอแสดงผลว่างๆ หรือไม่แสดงผลใดๆ และจำทำให้เคอร์เซอร์กลับมาอยู่ ณ ตำแหน่งซ้ายสุดของจอแสดงผล DDRAM คือ หน่วยความจำที่เก็บข้อมูลที่จะแสดงผลบนจอแอลซีดี

2. คำสั่ง Return Home

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

ข้อมูลคำสั่งคือ 0x02 หรือ 0x03 ก็ได้แต่นิยมใช้ 0x02 เป็นคำสั่งควบคุมให้เคอร์เซอร์ไปแสดงผลในตำแหน่งซ้ายสุดของจอ LCD ซึ่งข้อมูลที่แสดงผลอยู่นั้นจะไม่เปลี่ยนแปลง

* X หมายถึง การกำหนดให้เป็นอะไรก็ได้ (Don't Coare)

4. คำสั่งกำหนดโหมดการป้อนข้อมูล (Entry Mode Set)

RS	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	1	I/ D	S

I/ D : เป็นบิตที่ใช้กำหนดการเพิ่มขึ้นหรือ ลดลงของแอดเดรสที่ใช้แสดงผล (DDRAM) หลังจากมีการเขียนข้อมูลเข้าไปแสดงผล

"0" = ลดแอดเดรสลงหนึ่ง

"1" = เพิ่มแอดเดรสขึ้นหนึ่ง

S : เป็นบิตใช้กำหนดรูปแบบการแสดงผล

"0" = เคอร์เซอร์เลื่อนไปทางขวามือเมื่อมีการเขียนตัวอักษรเข้ามาใหม่

"1" = เคอร์เซอร์อยู่ที่เดิม แต่ตัวอักษรเลื่อนไปทางซ้ายเมื่อมีการเขียนตัวอักษรเข้ามาใหม่

5. คำสั่งควบคุมลักษณะการแสดงผลของจอแอลซีดี

RS	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	1	D	C	B

เป็นคำสั่งในการกำหนดรูปแบบการทำงานของจอแอลซีดี ซึ่งมี 3 ส่วนดังนี้คือ

D : เป็นบิตที่ใช้กำหนดการปิดเปิดจอแสดงผลแอลซีดี

"0" = ปิดจอแสดงผล

"1" = เปิดจอแสดงผล

C : เป็นบิตที่ใช้ควบคุมการแสดงผลของเคอร์เซอร์

"0" = ไม่แสดงเคอร์เซอร์

"1" = แสดงเคอร์เซอร์

B : เป็นบิตที่ใช้กำหนดการแสดงตัวกระพริบของจอแสดงผลแอลซีดี

"0" = ไม่แสดงตัวกระพริบ

"1" = แสดงตัวกระพริบ

6. คำสั่งควบคุมการเลื่อนเคอร์เซอร์ และตัวอักษร

RS	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	S/C	R/L	X	X

เป็นคำสั่งควบคุมการเลื่อนตำแหน่งเคอร์เซอร์และตัวอักษร โดยมีรูปแบบการกำหนดค่าต่างๆ ดังนี้

S/C : เป็นบิตที่ใช้กำหนดลักษณะของการเลื่อนว่าเป็นการเลื่อนเคอร์เซอร์ หรือ เลื่อนตัวอักษร

"0" = เป็นการเลื่อนเคอร์เซอร์

"1" = เป็นการเลื่อนตัวอักษร

R/L : เป็นบิตที่ใช้กำหนดรูปแบบทิศทางการเลื่อนข้อมูล ไปทางขวา หรือ ซ้าย

"0" = เลื่อนซ้าย

"0" = เลื่อนขวา

* บิต D0 และ D1 กำหนดเป็นอะไรก็ได้

7. คำสั่งกำหนดฟังก์ชั่นการทำงานของแอลซีดี

RS	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	1	D L	N	F	X	X

เป็นคำสั่งกำหนดฟังก์ชั่นในการทำงานต่างๆ ของแอลซีดี ซึ่งมีการกำหนดค่าต่างๆ ดังนี้

DL : เป็นบิตที่ใช้กำหนดโหมดของการติดต่อกับแอลซีดี

"0" = กำหนดการทำงานเป็นโหมด 4 บิต

"1" = กำหนดการทำงานเป็นโหมด 8 บิต

N : เป็นบิตที่ใช้กำหนดจำนวนบรรทัดที่ต้องการแสดงผล

"0" = แสดงผลเป็น 1 บรรทัด

"1" = แสดงผลเป็น 2 บรรทัดขึ้นไป

F : ใช้กำหนดความละเอียดของตัวอักษรที่ใช้แสดงผล

"0" = แสดงผลแบบ 5x7 จุด

"1" = แสดงผลแบบ 5x10 จุด

ในกรณีการกำหนดจำนวนบรรทัด หากใช้แอลซีดีรุ่น 16 ตัวอักษร 1 บรรทัด ที่มีแอดเดรสไม่ต่อเนื่องกันจะต้องกำหนดให้บิต N เป็น "1" เสมือนกับการกำหนดให้แอลซีดีเป็นแบบ 2 บรรทัด เพื่อให้แอลซีดีมองเห็นแอดเดรสทั้งหมด

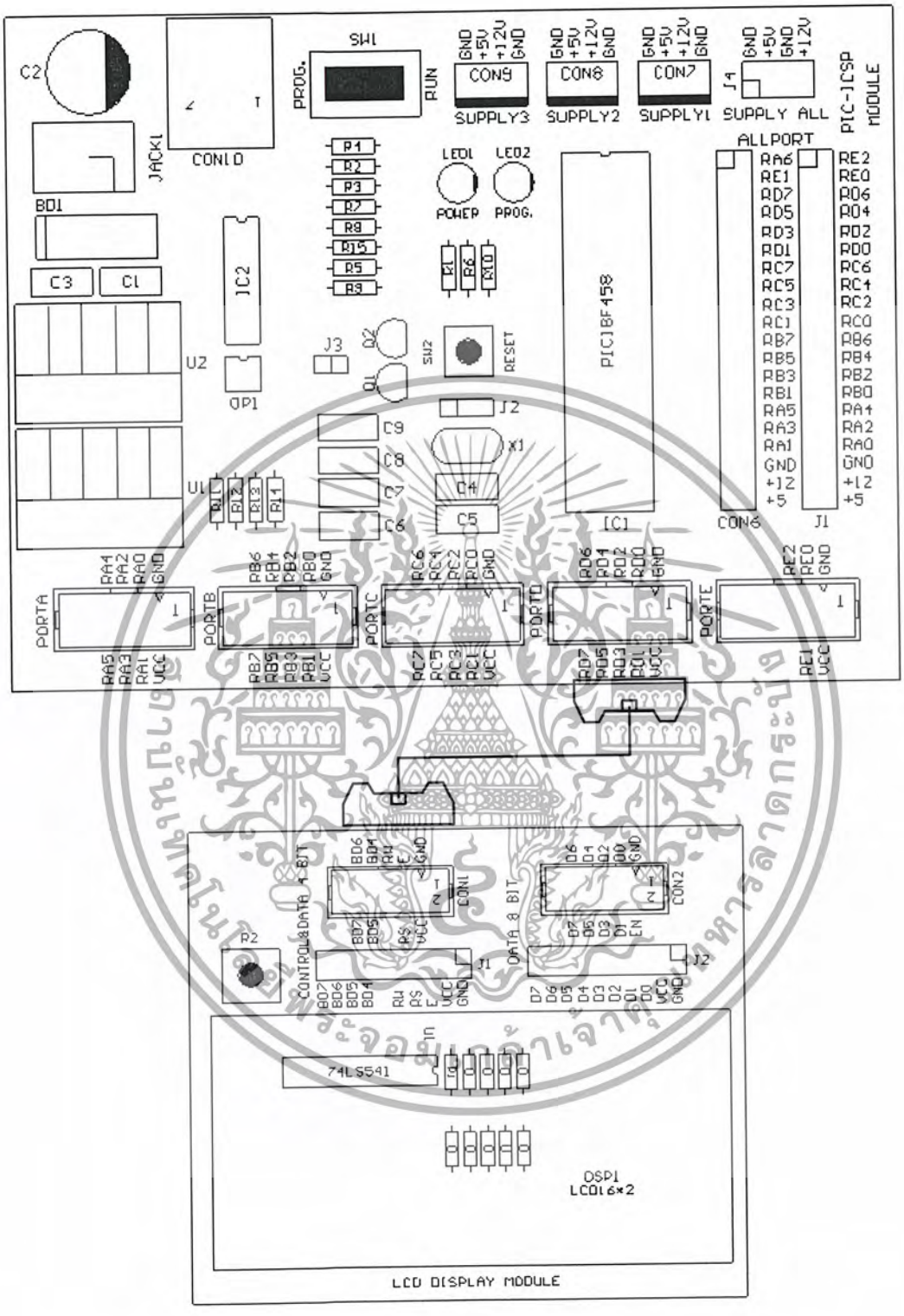
8. คำสั่งเลือกแอดเดรสของ CGRAM

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

เป็นคำสั่งกำหนดตำแหน่งแอดเดรสของ CGRAM ที่ต้องการติดต่อโดยจะต้องกำหนดให้บิต D7 เป็น "0" ส่วนใน 6 บิตที่เหลือ D5 - D0 จะเป็นค่าตำแหน่งแอดเดรสของ CGRAM โดยในการติดต่อกับ CGRAM จะต้องมีการกำหนดแอดเดรสตรงนี้เสียก่อนสามารถทำได้โดยกำหนดค่าที่ D7 เป็น "1" จากนั้นอีก 6 บิต

9. คำสั่งเลือกแอดเดรส DDRAM

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



รูปที่ จ. 32 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับโมดูลแสดงผลแอลซีดี

```
;*****************************************************
; Program     : Test LCD Display [4 Bit Data)
; Filename    : lab12.asm
; MCU         : PIC 18F458
; OSC         : 10 MHz [HS mode]
;*****************************************************
        list p=18f458           ; list directive to define processor
            #include <p18f458.inc>     ; processor specific variable definitions

  #DEFINE   RS      PORT0,0     ; กำหนดให้ขา RS เป็นขาส่งข้อมูล
  #DEFINE   E       PORTC,1     ; PORTC.1 is E pin
  COM       EQU     0x20
  DAT       EQU     0x21
  COUNT1    EQU     0x22
  COUNT2    EQU     0x23
  COUNT3    EQU     0x24

      ORG     0x0000
;************ initial ********************
      CLRF    TRISC           ; PORTC is output
      CLRF    TRISD           ; PORTD is output
      CALL    DELAY
      MOVLW   B'00110011'
      CALL    WR_INS          ; Send command to LCD
      MOVLW   B'00110010'
      CALL    WR_INS          ; Send command to LCD
      MOVLW   B'00101000'     ; LCD 4 mode, 2 line,5X7 dot
      CALL    WR_INS          ; Send command to LCD
      MOVLW   B'00001100'     ; On display off cursor
      CALL    WR_INS          ; send command to LCD
      MOVLW   B'00000110'     ; Entry mode
      CALL    WR_INS          ; Send command to LCD
      MOVLW   B'00000001'     ; Clear display (DDRAM)
      CALL    WR_INS          ; Send command to LCD
;********** Data to display LCD ********
      MOVLW   "K"             ; Ascii A
      CALL    WR_DATA         ; Send data to LCD
      MOVLW   "M"             ; Ascii B
      CALL    WR_DATA         ; Send data to LCD
      MOVLW   "I"             ; Ascii C
```

```
        CALL    WR_DATA         ; Send data to LCD
        MOVLW   "T"             ; Ascii D
        CALL    WR_DATA         ; Send data to LCD
        MOVLW   "L"             ; Ascii D
        CALL    WR_DATA         ; Send data to LCD
        GOTO    $               ; Stop
;********* Write command to LCD *********
WR_INS  BCF     RS              ; RS = "1"
        BSF     E               ; E = "1"
        MOVWF   COM             ; COM <--(W)
        ANDLW   0xF0            ; mask 4 bits MSB
        MOVWF   PORTD           ; Send 4 bit upper to LCD
        CALL    PULSE           ; Pulse enable LCD
        SWAPF   COM,W           ; Swap nibbles COM and save to W
        ANDLW   0xF0            ; mask 4 bits MSB
        MOVWF   PORTD           ; Send 4 bit lower to LCD
        CALL    PULSE           ; Pulse enable LCD
        RETURN
;*********** Write data to LCD ************
WR_DATA BSF     RS              ; RS = "1"
        BSF     E               ; E = 1
        MOVWF   DAT             ; Save to DAT
        ANDLW   0xF0            ; mask 4 Bit MSB
        MOVWF   PORTD           ; Send 4 bit upper to LCD
        CALL    PULSE           ; Pulse enable LCD
        SWAPF   DAT,W           ; Swap nibbles DAT and save to W
        ANDLW   0xF0            ; mask 4 Bit MSB
        MOVWF   PORTD           ; Send 4 bit lower to LCD
        CALL    PULSE           ; Pulse enable LCD
        RETURN
;********* Generate pulse *******
PULSE   BCF     E               ; E = "0"
        CALL    DELAY           ; delay
        BSF     E               ; E = "1"
        RETURN


****** Delay time *********
DELAY   MOVLW   .50
```

```
        MOVWF   COUNT1
DEL1    CLRF    COUNT2
DEL2    DECFSZ  COUNT2
        GOTO    DEL2
        DECFSZ  COUNT1
        GOTO    DEL1
        RETURN
        END
```

รูปที่ จ.33 โปรแกรมการทดลองใช้งาน PIC18F458 ขับแอลซีดีในโหมด 4 บิต

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.33 ทำการคอมไพล์ให้เป็น Hex File
2. ต่อวงจรตามรูปที่ จ.32
3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ด้วยโปรแกรม Epic Win
4. รันโปรแกรมเลื่อนสวิตช์ที่ตำแหน่ง RUN
5. สังเกตผลที่เกิดขึ้นกับจอแสดงผลแอลซีดี บันทึกผลการทดลอง

.....

.....

.....

.....

6. ทดลองเปลี่ยนข้อมูลจาก KMITL ทำการคอมไพล์ และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้ง บันทึกผลการทดลอง

.....

.....

.....

.....

7. เปลี่ยนโปรแกรมจาก MOVLW B'00001100 เป็น MOVLW B'00001110 ทำการคอมไพล์และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้ง บันทึกผลการทดลอง

.....

.....

.....

.....

8. เปลี่ยนโปรแกรมจาก MOVLW B'00001110 จากการทดลองข้อ 7 ทำการคอมไพล์และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้ง สังเกตผลที่เกิดขึ้น บันทึกผลการทดลอง

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงเขียนวงจรแสดงการเชื่อมต่อแอลซีดีแบบ 8 บิต และแบบ 4 บิต

2. จากการทดลองเมื่อต้องการให้แอลซีดีแสดงผลเป็น LCD 16x2 TEST และกระพริบ จะนั้นเขียนโปรแกรมอย่างไร หรือแทรกโปรแกรมที่ตำแหน่งไหนจงอธิบาย

3. ในการต่อแอลซีดีแบบ 8 บิต กับ 4 บิต แตกต่างกันอย่างไรและมีวิธีในการเขียนโปรแกรมส่งข้อมูลอย่างไรจงอธิบาย

ใบงานที่ 13

การสื่อสารข้อมูลกับคอมพิวเตอร์ผ่านพอร์ตอนุกรม RS-232

จุดประสงค์

1. เพื่อให้สามารถอธิบายหลักการสื่อสารข้อมูลกับคอมพิวเตอร์แบบอนุกรม RS-232 ได้
2. เพื่อให้สามารถใช้งานพอร์ตการสื่อสารแบบอนุกรมของไมโครคอนโทรลเลอร์ PIC18F458 ได้
3. เพื่อให้สามารถเชื่อมต่อไมโครคอนโทรลเลอร์ PIC18F458 เข้ากับคอมพิวเตอร์ผ่านพอร์ตอนุกรมมาตรฐาน RS-232C ได้

อุปกรณ์ที่ใช้ในการทดลอง

1. โมดูลหลัก PIC-ICSP
2. โมดูลแสดงผลแอลอีดี
3. โมดูลสื่อสารข้อมูลอนุกรม
4. คอมพิวเตอร์พีซีที่ติดตั้งโปรแกรม MPLAB, Epic Win และ Hyper Terminal
5. สายเชื่อมต่อระหว่างคอมพิวเตอร์พีซีและโมดูลสื่อสารข้อมูลอนุกรม
6. สายเชื่อมต่อพอร์ตระหว่างโมดูล

ทฤษฎีเบื้องต้น

RS-232 นั้นเป็นระบบการส่งข้อมูลในรูปแบบอนุกรมคือ ข้อมูลจะส่งไปได้ทีละบิตโดยจะมีอัตราการส่งข้อมูลเป็นบิตต่อวินาทีหรืออัตราการเปลี่ยนแปลงของสัญญาณใน 1 วินาทีเรียกว่า Baud Rate โดยจะมีการส่งบิตเริ่มต้น (Start Bit) มีระดับสัญญาณเป็น 0 และบิตข้อมูล (Data Bit) ซึ่งจะมีข้อมูล 7 บิตหรือ 8 บิต และอาจตามด้วยบิตพาริตี้ (Parity Bit) อาจจะเป็นแบบคู่ (Even) หรือแบบคี่ (Odd) เพื่อตรวจสอบความถูกต้องของข้อมูล บิตพาริตี้นั้นจะมีหรือไม่มีก็ได้และสุดท้ายจะต้องตามด้วยบิตหยุด (Stop bit) ซึ่งจะมีความกว้างของสัญญาณเป็น 1, 1.5, 2 บิตก็ได้ ซึ่งจะเห็นว่าในการส่งข้อมูลแบบนี้จึงจำเป็นต้องมีข้อตกลงกันระหว่างการรับและการส่งคือ

1. ความเร็วในการส่ง Baud Rate

2. จำนวนข้อมูล

3. มีบิตพาริตี้หรือ ถ้ามีจะเป็นแบบคู่หรือแบบคี่

4. จำนวนบิตหยุด 1, 1.5 หรือ 2 บิต

ในการส่งแบบ RS-232C นี้เราสามารถเลือก Baud Rate ได้หลาย Baud Rate เช่น 110, 200, 300, 1200, 2400, 4800, 9600 เช่นถ้าเราส่งข้อมูล 8 บิต โดย Baud Rate 2400 แล้ว 1 บิตต้องใช้เวลา 1/2400 = 416 ไมโครวินาที

ลักษณะของสัญญาณ RS-232C

เพื่อให้การส่งข้อมูลสามารถไปได้ไกลขึ้นจึงกำหนดให้การส่งสัญญาณมีระดับความแตกต่างเป็นบวก 15 โวลต์ หรืออาจจะเป็นเวลา 12 โวลต์ และลบ 12 โวลต์ ก็ได้

สัญญาณของขาต่างๆ ที่ใช้งาน

ขาที่ 2 Transmit Data

ขาที่ 3 Receive Data เป็นขาที่ได้รับสัญญาณข้อมูลจากเครื่องอื่น

ขาที่ 4 Request to Sent เป็นขาที่บอกเครื่องอื่นๆ ว่าตัวเองพร้อมที่จะส่งข้อมูล

ขาที่ 5 Clear to Send เป็นขาที่บอกต่อเครื่องอื่นๆ ว่าตัวเองพร้อมที่จะรับข้อมูล

ขาที่ 6 Data Set Ready เป็นขาที่บอกไมโครคอมพิวเตอร์ว่าโมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยแล้วและพร้อมที่จะส่งได้แล้ว

ขาที่ 7 Signal Ground เป็นขา GND ของสัญญาณ

ขาที่ 20 Data Terminal Ready เป็นขาบอกว่าโมเด็มไมโครคอมพิวเตอร์พร้อมที่จะติดต่อด้วยโดยทั่วไปเราสามารถต่อใช้เครื่องคอมพิวเตอร์ติดต่อกันทาง RS232C ได้ง่ายๆ โดยใช้สัญญาณ Transmit Data เพียง 2 เส้น

โมดูลสื่อสารข้อมูลอนุกรมใน PIC18F458

โมดูลสื่อสารข้อมูลอนุกรมใน PIC18F458 นั้นมีรูปแบบการสื่อสารแบบวิงโครนัสและอะซิงโครนัส ในลักษณะที่จะเชื่อมต่อกับ RS-232 นี้จะต้องกำหนดให้โมดูลสื่อสารข้อมูลอนุกรมนั้นทำงานในโหมดอะซิงโครนัสมีรีจิสเตอร์ที่เกี่ยวข้องดังต่อไปนี้

TXREG ใช้สำหรับเก็บค่าที่จะส่งออกไปยังพอร์ตอนุกรม

TXSTA ใช้กำหนดค่ามาตราฐานในการส่งข้อมูล

SPBRG ใช้สำหรับการสร้างบอดเรต

RCREG ใช้สำหรับเก็บข้อมูลที่อ่านได้จากภาครับของพอร์ตอนุกรม

RCSTA ใช้กำหนดค่ามาตราฐานในการรับข้อมูล

รีจิสเตอร์ TXSTA

มีรายละเอียดของการทำงานดังนี้

ตารางที่ จ.18 รายละเอียดของรีจิสเตอร์ TXSTA

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	RX9D
R/W –0	/W –0	R/W –0	R/W –0	-	R/W –0	R/W –1	R/W –0

CSRS (Clock Source Select Bit : บิต 7) บิตเลือกแหล่งกำเนิดสัญญาณนาฬิกา

ในโหมดอะซิงโครนัส : ไม่มีใช้งานในบิตนี้

ในโหมดอะซิงโครนัส :

"0" - ทำงานเป็นอุปกรณ์สเลฟ เลือกแหล่งกำเนิดสัญญาณนาฬิกาภายนอก

"1" - ทำงานเป็นอุปกรณ์มาสเตอร์ ใช้แหล่งกำเนิดสัญญาณนาฬิกาจากส่วนกำเนิดอัตราบอดหรืออัตราเร็วในการถ่ายทอดข้อมูลภายในไมโครคอนโทรลเลอร์

TX9 (9-Bit Transmit Enable Bit : บิต 6) บิตเลือกการส่งข้อมูลแบบ 9 บิต

"0" - เลือกการส่งข้อมูลแบบ 8 บิต

"1" - เลือกการส่งข้อมูลแบบ 9 บิต

TXEN (Transmit Enable Bit : บิต 5) บิตเลือกการทำงานของตัวส่ง

"0" - ดิสเอเบิลตัวส่งข้อมูล

"1" - เอ็นเอเบิลตัวส่งข้อมูล

SYNC (USART Mode Select Bit : บิต 4) บิตเลือกโหมดการทำงานของโมดูล USART

"0" - เลือกโหมดอะซิงโครนัส

"1" - เลือกโหมดซิงโครนัส

บิต 3 ไม่มีการใช้งาน อ่านค่าเป็น "0"

BRGH (High Baud Rate Seiect Bit : บิต 2) บิตเลือกโหมดของอัตราเร็วในการถ่ายทอดข้อมูล

"0" - เลือกอัตราเร็วในการถ่ายทอดข้อมูลต่ำ (อัตราบอดหรือบอดเรตต่ำ)

"1" - เลือกอัตราเร็วในการถ่ายทอดข้อมูลสูง (อัตราบอดหรือบอดเรตสูง)

ในโหมดซิงโครนัส : ไม่มีการใช้งานบิตนี้

TRMT (Trnasmit Shift Status Bit : บิต 1) บิตแสดงสถานะรีจิสเตอร์บัฟเฟอร์ของการส่ง

"0" - รีจิสเตอร์บัฟเฟอร์ (TSR) เต็มหรือไม่ว่าง

"1" - รีจิสเตอร์บัฟเฟอร์ (TSR) ว่าง

TX9D (9^{th} Bit transmit Data : บิต 0) บิตเก็บข้อมูลที่ 9 ของการส่งข้อมูลแบบ 9 บิต

บิตนี้ใช้สำหรับบรรจุข้อมูลในบิตที่ 9 ในกรณีที่เลือกการส่งข้อมูลแบบ 9 บิต โดยการเซต บิต TX9 และยังสามารถใช้เป็นบิตพาริตี้ของการสื่อสารข้อมูลอนุกรมได้

รีจิสเตอร์ RCSTA

รายละเอียดของการทำงานดังนี้

ตารางที่ จ.19 รายละเอียดของรีจิสเตอร์ RCSTA

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
R/W –0	/W –0	R/W –0	R/W –0	R/W –0	R/W –0	R/W –1	R/W –x

SPEN (Serial Port Enable Bit : บิต 7) บิตเลือกการทำงานของโมดูล USART

"0" - ดิสเอเบิล

"1" - เอ็นเอเบิลเพื่อใช้งานโมดูล USART ทำให้ขา RB7/RxD และ RB6/TxD ใช้งานกับโมดูล USART เพื่อการสื่อสารข้อมูลอนุกรม เมื่อบิต 7 และ 6 ของรีจิสเตอร์ TRISC ถูกเซตเป็น "1"

RX9 (9-Bit Receive Enable Bit : บิต 6) บิตเลือกการรับข้อมูลแบบ 9 บิต

"0" - เลือกการส่งข้อมูลแบบ 8 บิต

"1" - เลือกการส่งข้อมูลแบบ 9 บิต

SREN (Single Receive Enable Bit : บิต 5) บิตเลือกการรับข้อมูลครั้งเดียวจะเคลียร์หลังจากรับข้อมูลสมบูรณ์

ในโหมดอะซิงโครนัส และซิงโตรนัส-สเลฟ : ไม่ใช้งานบิตนี้

"0" - ดิสเอเบิลตัวการรับข้อมูลครั้งเดียว

"1" - เอ็นเอเบิลการรับข้อมูลครั้งเดียว

CREN (USART Mode Select Bit : บิต 4) บิตเลือกการรับข้อมูลต่อเนื่อง

"1" - ดิสเอเบิลการรับข้อมูลต่อเนื่อง

"0" - เอ็นเอเบิลการรับข้อมูลต่อเนื่อง

ADDEN (Address Detect Enable Bit : บิต 3) บิตเลือกการตรวจจับแอดเดรส

"0" - ดิสเอเบิลการตรวจจับแอดเดรส ข้อมูลจะถูกรับทั้งหมด และบิตที่ 9 ใช้เป็นบิตพาริตี้ได้

"1" - เอ็นเอเบิลการตรวจจับแอดเดรส ส่งผลให้เกิดการเอ็นเอเบิลอินเตอร์รัพต์ และมีการถ่ายทอดข้อมูลไปยังบัฟเฟอร์เมื่อบิต RSR ถูกเซต

ในโหมดซิงโครนัส : ไม่มีการใช้งานบิตนี้

FERR (Framing Error Bit : บิต 2) บิตแจ้งความผิดพลาดทางเฟรมข้อมูล

"0" - ไม่มีความผิดพลาดเกิดขึ้น

"1" - เกิดความผิดพลาดทางเฟรมข้อมูลขึ้น

OEER (Overrun error bit : บิต 1) บิตแจ้งความผิดพลาดเนื่องจากการชนกันของข้อมูล

"0" - ไม่มีความผิดพลาดเกิดขึ้น

"1" - เกิดความผิดพลาดขึ้น สามารถเคลียร์บิตได้ด้วยการเคลียร์บิต CREN

RX9D (9th Bit Of Data : บิต 0) บิตเก็บข้อมูลที่ 9 ของการส่งข้อมูลแบบ 9 บิต

บิตนี้ใช้สำหรับบรรจุข้อมูลในบิตที่ 9 ในกรณีที่เลือกการส่งข้อมูลแบบ 9 บิต โดยการเซตบิต RX9 และยังสามารถใช้เป็นบิตพาริตี้ของการสื่อสารข้อมูลอนุกรมได้ส่วนกำเนิดอัตราเร็วในการถ่ายทอดข้อมูลหรือบอดเรตเจเนอเรเตอร์

ในโมดูล USART ใช้ส่วนกำเนิดอัตราเร็วในการถ่ายทอดข้อมูลหรือบอดเรตเจเนเรเตอร์เพียงชุดเดียว สามารถรองรับทั้งการทำงานในโหมดอะซิงโครนัสและซิงโครนัส โดยมีความละเอียด 8 บิต รีจิสเตอร์ควบคุมการทำงานคือ SPBRG โดยจะควบคุมคาบเวลาของไทเมอร์อิสระขนาด 8 บิตเพื่อกำหนดอัตราเร็วในการถ่ายทอดข้อมูล เมื่อทำงานในโหมดอะซิงโครนัสต้องทำงานร่วมกับข้อมูลที่บิต BRGH (บิต 2 ของรีจิสเตอร์ TXSTA) เพื่อกำหนดย่านของอัตราเร็วแบบต่ำและสูง กำหนดให้ความถี่สัญญาณนาฬิกาหลัก (หรือความถี่ของคริสตอล) เท่ากับ 16MHz ต้องการบอดเรตเท่ากับ 9,600 บิตต่อวินาที เลือกย่านบอดเรตต่ำ ทำงานเป็นแบบอะซิงโครนัส สามารสรุปความสัมพันธ์ทางคณิตศาสตร์ได้ดังนี้

ค่าบอดเรต = FOSC/64 (X+1)

$9{,}600 = 16x10^{6}/64\ (X+1)$

X = 25.042 โดยค่าของ X นั้นคือค่าของรีจิสเตอร์ SPBRG แต่เนื่องจากข้อมูลในรีจิสเตอร์

SPBRG ต้องใช้เฉพาะจำนวนเต็มเท่านั้นจึงต้องกำหนดค่าของ SPBRG เป็น 25 แทนกลับเข้าไปในสูตรคำณวนค่าบอดเรตจะได้ $16x10^6$ /64 (25+1) = 9,615 บิตต่อวินาที

ขั้นตอนการเขียนโปรแกรมเพื่อส่งข้อมูลไปยังคอมพิวเตอร์

1. กำหนดค่าบอดเรตโดยป้อนค่าเข้าที่รีจิสเตอร์ SPBRG และเลือกโหมดของบอดเรตเป็นโหมดความเร็วต่ำหรือสูงโดยกำหนดที่บิต BRGH ของรีจิสเตอร์ TXSTA ซึ่งค่าที่กำหนดนี้จะอ้างอิงถึงความถี่คริสตอลที่ป้อนให้กับ CPU สามารถศึกษาได้จากดาต้าชีท

2. เอ็นเอเบิลการทำงานของพอร์ตอนุกรมโดยเซตบิต SPEN ในรีจิสเตอร์ RCSTA เป็น 1 และเลือกการทำงานเป็นแบบอะซิงโครนัสโดยเคลียร์บิต SYNC ในรีจิสเตอร์ TXSTA

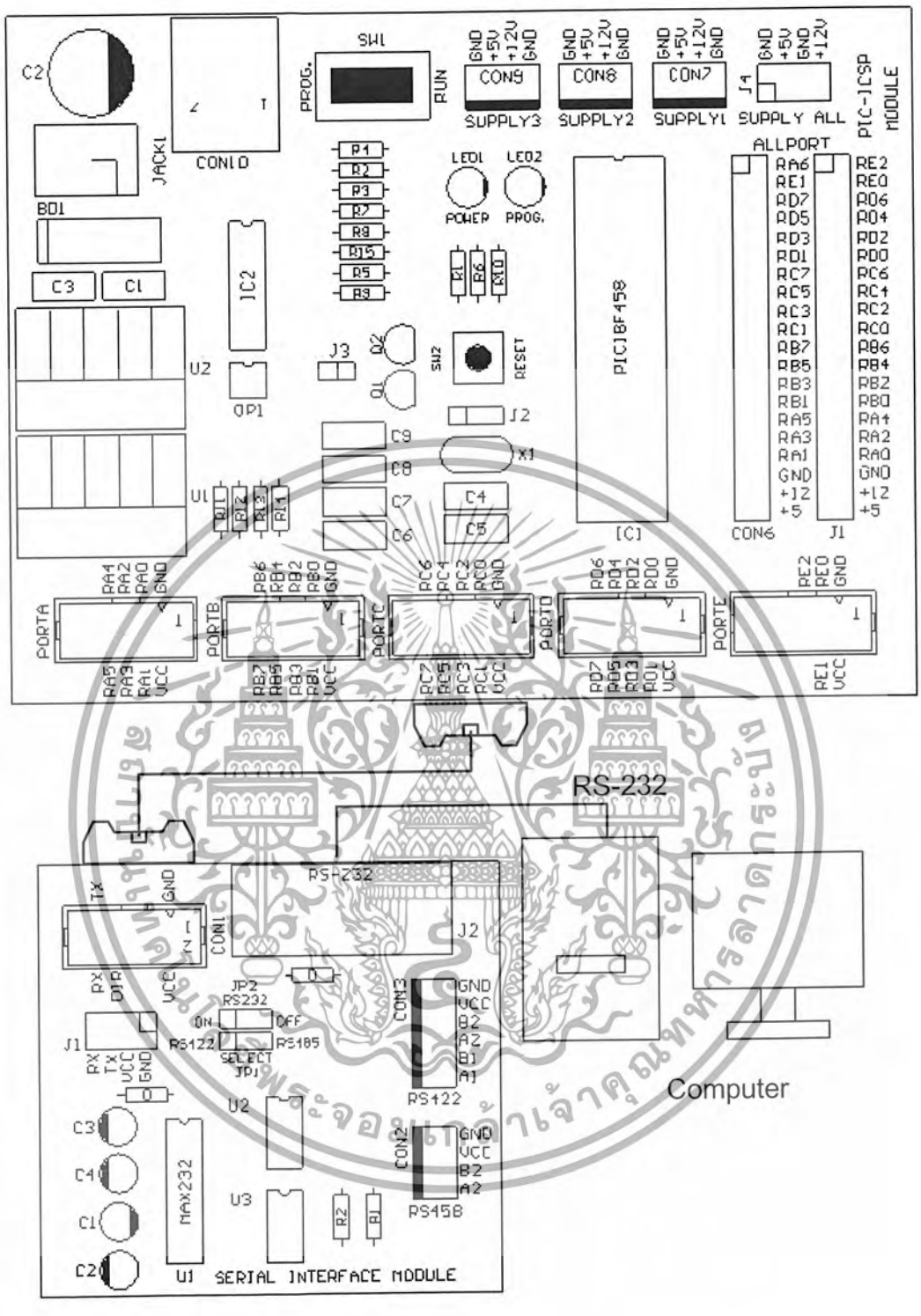
3. ถ้ามีการใช้งานการอินเตอร์รัพต์ต้องเอ็นเอเบิลการอินเตอร์รัพต์ด้วย

4. เอ็นเอเบิลการส่งข้อมูล โดยเซตบิต TXEN ในรีจิสเตอร์ TXSTA

5. นำข้อมูลที่จะส่งไปเก็บไว้ที่รีจิสเตอร์ TXREG

6. รอให้บัฟเฟอร์สำหรับส่งข้อมูลว่างแล้วค่อยส่งข้อมูลในไบต์ต่อๆไป

สำหรับรายระเอียดบิตต่างๆของรีจิสเตอร์ศึกษาได้จากภาคผนวกเรื่องรายระเอียดและคุณสมบัติของอุปกรณ์



รูปที่ จ.34 การเชื่อมต่อโมดูลสื่อสารข้อมูลอนุกรม กับคอมพิวเตอร์

ใบงานที่ 13
การสื่อสารข้อมูลกับคอมพิวเตอร์ผ่านพอร์ตอนุกรม RS-232

จุดประสงค์

1. เพื่อให้สามารถอธิบายหลักการสื่อสารข้อมูลกับคอมพิวเตอร์แบบอนุกรม RS-232 ได้
2. เพื่อให้สามารถใช้งานพอร์ตการสื่อสารแบบอนุกรมของไมโครคอนโทรลเลอร์ PIC18F458 ได้
3. เพื่อให้สามารถเชื่อมต่อไมโครคอนโทรลเลอร์ PIC18F458 เข้ากับคอมพิวเตอร์ผ่านพอร์ตอนุกรมมาตรฐาน RS-232C ได้

อุปกรณ์ที่ใช้ในการทดลอง

1. โมดูลหลัก PIC-ICSP
2. โมดูลแสดงผลแอลอีดี
3. โมดูลสื่อสารข้อมูลอนุกรม
4. คอมพิวเตอร์พีซีที่ติดตั้งโปรแกรม MPLAB, Epic Win และ Hyper Terminal
5. สายเชื่อมต่อระหว่างคอมพิวเตอร์พีซีและโมดูลสื่อสารข้อมูลอนุกรม
6. สายเชื่อมต่อพอร์ตระหว่างโมดูล

ทฤษฎีเบื้องต้น

RS-232 นั้นเป็นระบบการส่งข้อมูลในรูปแบบอนุกรมคือ ข้อมูลจะส่งไปได้ทีละบิตโดยจะมีอัตราการส่งข้อมูลเป็นบิตต่อวินาทีหรืออัตราการเปลี่ยนแปลงของสัญญาณใน 1 วินาทีเรียกว่า Baud Rate โดยจะมีการส่งบิตเริ่มต้น (Start Bit) มีระดับสัญญาณเป็น 0 และบิตข้อมูล (Data Bit) ซึ่งจะมีข้อมูล 7 บิตหรือ 8 บิต และอาจตามด้วยบิตพาริตี้ (Parity Bit) อาจจะเป็นแบบคู่ (Even) หรือแบบคี่ (Odd) เพื่อตรวจสอบความถูกต้องของข้อมูล บิตพาริตี้นั้นจะมีหรือไม่มีก็ได้และสุดท้ายจะต้องตามด้วยบิตหยุด (Stop bit) ซึ่งจะมีความกว้างของสัญญาณเป็น 1, 1.5, 2 บิตก็ได้ ซึ่งจะเห็นว่าในการส่งข้อมูลแบบนี้จึงจำเป็นต้องมีข้อตกลงกันระหว่างการรับและการส่งคือ

1. ความเร็วในการส่ง Baud Rate

2. จำนวนข้อมูล

3. มีบิตพาริตี้หรือ ถ้ามีจะเป็นแบบคู่หรือแบบคี่

4. จำนวนบิตหยุด 1, 1.5 หรือ 2 บิต

ในการส่งแบบ RS-232C นี้เราสามารถเลือก Baud Rate ได้หลาย Baud Rate เช่น 110, 200, 300, 1200, 2400, 4800, 9600 เช่นถ้าเราส่งข้อมูล 8 บิต โดย Baud Rate 2400 แล้ว 1 บิตต้องใช้เวลา 1/2400 = 416 ไมโครวินาที

ลักษณะของสัญญาณ RS-232C

เพื่อให้การส่งข้อมูลสามารถไปได้ไกลขึ้นจึงกำหนดให้การส่งสัญญาณมีระดับความแตกต่างเป็นบวก 15 โวลต์ หรืออาจจะเป็นเวลา 12 โวลต์ และลบ 12 โวลต์ ก็ได้

สัญญาณของขาต่างๆ ที่ใช้งาน

ขาที่ 2 Transmit Data

ขาที่ 3 Receive Data เป็นขาที่ได้รับสัญญาณข้อมูลจากเครื่องอื่น

ขาที่ 4 Request to Sent เป็นขาที่บอกเครื่องอื่นๆ ว่าตัวเองพร้อมที่จะส่งข้อมูล

ขาที่ 5 Clear to Send เป็นขาที่บอกต่อเครื่องอื่นๆ ว่าตัวเองพร้อมที่จะรับข้อมูล

ขาที่ 6 Data Set Ready เป็นขาที่บอกไมโครคอมพิวเตอร์ว่าโมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยแล้วและพร้อมที่จะส่งได้แล้ว

ขาที่ 7 Signal Ground เป็นขา GND ของสัญญาณ

ขาที่ 20 Data Terminal Ready เป็นขาบอกว่าโมเด็มไมโครคอมพิวเตอร์พร้อมที่จะติดต่อด้วยโดยทั่วไปเราสามารถต่อใช้เครื่องคอมพิวเตอร์ติดต่อกันทาง RS232C ได้ง่ายๆ โดยใช้สัญญาณ Transmit Data เพียง 2 เส้น

โมดูลสื่อสารข้อมูลอนุกรมใน PIC18F458

โมดูลสื่อสารข้อมูลอนุกรมใน PIC18F458 นั้นมีรูปแบบการสื่อสารแบบวิงโครนัสและอะซิงโครนัส ในลักษณะที่จะเชื่อมต่อกับ RS-232 นี้จะต้องกำหนดให้โมดูลสื่อสารข้อมูลอนุกรมนั้นทำงานในโหมดอะซิงโครนัสมีรีจิสเตอร์ที่เกี่ยวข้องดังต่อไปนี้

TXREG ใช้สำหรับเก็บค่าที่จะส่งออกไปยังพอร์ตอนุกรม

TXSTA ใช้กำหนดค่ามาตราฐานในการส่งข้อมูล

SPBRG ใช้สำหรับการสร้างบอดเรต

RCREGใช้สำหรับเก็บข้อมูลที่อ่านได้จากภาครับของพอร์ตอนุกรม

RCSTA ใช้กำหนดค่ามาตราฐานในการรับข้อมูล

รีจิสเตอร์ TXSTA

มีรายละเอียดของการทำงานดังนี้

ตารางที่ จ.18 รายละเอียดของรีจิสเตอร์ TXSTA

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	RX9D
R/W –0	/W –0	R/W –0	R/W –0	-	R/W –0	R/W –1	R/W –0

CSRS (Clock Source Select Bit : บิต 7) บิตเลือกแหล่งกำเนิดสัญญาณนาฬิกา

ในโหมดอะซิงโครนัส : ไม่มีใช้งานในบิตนี้

ในโหมดอะซิงโครนัส :

"0" - ทำงานเป็นอุปกรณ์สเลฟ เลือกแหล่งกำเนิดสัญญาณนาฬิกาภายนอก

"1" - ทำงานเป็นอุปกรณ์มาสเตอร์ ใช้แหล่งกำเนิดสัญญาณนาฬิกาจากส่วนกำเนิดอัตราบอดหรืออัตราเร็วในการถ่ายทอดข้อมูลภายในไมโครคอนโทรลเลอร์

TX9 (9-Bit Transmit Enable Bit : บิต 6) บิตเลือกการส่งข้อมูลแบบ 9 บิต

"0" - เลือกการส่งข้อมูลแบบ 8 บิต

"1" - เลือกการส่งข้อมูลแบบ 9 บิต

TXEN (Transmit Enable Bit : บิต 5) บิตเลือกการทำงานของตัวส่ง

"0" - ดิสเอเบิลตัวส่งข้อมูล

"1" - เอ็นเอเบิลตัวส่งข้อมูล

SYNC (USART Mode Select Bit : บิต 4) บิตเลือกโหมดการทำงานของโมดูล USART

"0" - เลือกโหมดอะซิงโครนัส

"1" - เลือกโหมดซิงโครนัส

บิต 3 ไม่มีการใช้งาน อ่านค่าเป็น "0"

BRGH (High Baud Rate Seiect Bit : บิต 2) บิตเลือกโหมดของอัตราเร็วในการถ่ายทอดข้อมูล

"0" - เลือกอัตราเร็วในการถ่ายทอดข้อมูลต่ำ (อัตราบอดหรือบอดเรตต่ำ)

"1" - เลือกอัตราเร็วในการถ่ายทอดข้อมูลสูง (อัตราบอดหรือบอดเรตสูง)

ในโหมดซิงโครนัส : ไม่มีการใช้งานบิตนี้

TRMT (Trnasmit Shift Status Bit : บิต 1) บิตแสดงสถานะรีจิสเตอร์บัฟเฟอร์ของการส่ง

"0" - รีจิสเตอร์บัฟเฟอร์ (TSR) เต็มหรือไม่ว่าง

"1" - รีจิสเตอร์บัฟเฟอร์ (TSR) ว่าง

TX9D (9^{th} Bit transmit Data : บิต 0) บิตเก็บข้อมูลที่ 9 ของการส่งข้อมูลแบบ 9 บิต

บิตนี้ใช้สำหรับบรรจุข้อมูลในบิตที่ 9 ในกรณีที่เลือกการส่งข้อมูลแบบ 9 บิต โดยการเซต บิต TX9 และยังสามารถใช้เป็นบิตพาริตี้ของการสื่อสารข้อมูลอนุกรมได้

รีจิสเตอร์ RCSTA

รายละเอียดของการทำงานดังนี้

ตารางที่ จ.19 รายละเอียดของรีจิสเตอร์ RCSTA

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
R/W –0	R/W –0	R/W –0	R/W –0	R/W –0	R/W –0	R/W –1	R/W –x

SPEN (Serial Port Enable Bit : บิต 7) บิตเลือกการทำงานของโมดูล USART

"0" - ดิสเอเบิล

"1" - เอ็นเอเบิลเพื่อใช้งานโมดูล USART ทำให้ขา RB7/RxD และ RB6/TxD ใช้งานกับโมดูล USART เพื่อการสื่อสารข้อมูลอนุกรม เมื่อบิต 7 และ 6 ของรีจิสเตอร์ TRISC ถูกเซตเป็น "1"

RX9 (9-Bit Receive Enable Bit : บิต 6) บิตเลือกการรับข้อมูลแบบ 9 บิต

"0" - เลือกการส่งข้อมูลแบบ 8 บิต

"1" - เลือกการส่งข้อมูลแบบ 9 บิต

SREN (Single Receive Enable Bit : บิต 5) บิตเลือกการรับข้อมูลครั้งเดียวจะเคลียร์หลังจากรับข้อมูลสมบูรณ์

ในโหมดอะซิงโครนัส และซิงโตรนัส-สเลฟ : ไม่ใช้งานบิตนี้

"0" - ดิสเอเบิลตัวการรับข้อมูลครั้งเดียว

"1" - เอ็นเอเบิลการรับข้อมูลครั้งเดียว

CREN (USART Mode Select Bit : บิต 4) บิตเลือกการรับข้อมูลต่อเนื่อง

"1" - ดิสเอเบิลการรับข้อมูลต่อเนื่อง

"0" - เอ็นเอเบิลการรับข้อมูลต่อเนื่อง

ADDEN (Address Detect Enable Bit : บิต 3) บิตเลือกการตรวจจับแอดเดรส

"0" - ดิสเอเบิลการตรวจจับแอดเดรส ข้อมูลจะถูกรับทั้งหมด และบิตที่ 9 ใช้เป็นบิตพาริตี้ได้

"1" - เอ็นเอเบิลการตรวจจับแอดเดรส ส่งผลให้เกิดการเอ็นเอเบิลอินเตอร์รัพต์ และมีการถ่ายทอดข้อมูลไปยังบัฟเฟอร์เมื่อบิต RSR ถูกเซต

ในโหมดซิงโครนัส : ไม่มีการใช้งานบิตนี้

FERR (Framing Error Bit : บิต 2) บิตแจ้งความผิดพลาดทางเฟรมข้อมูล

"0" - ไม่มีความผิดพลาดเกิดขึ้น

"1" - เกิดความผิดพลาดทางเฟรมข้อมูลขึ้น

OEER (Overrun error bit : บิต 1) บิตแจ้งความผิดพลาดเนื่องจากการชนกันของข้อมูล

"0" - ไม่มีความผิดพลาดเกิดขึ้น

"1" - เกิดความผิดพลาดขึ้น สามารถเคลียร์บิต ได้ด้วยการเคลียร์บิต CREN

RX9D (9th Bit Of Data : บิต 0) บิตเก็บข้อมูลที่ 9 ของการส่งข้อมูลแบบ 9 บิต

บิตนี้ใช้สำหรับบรรจุข้อมูลในบิตที่ 9 ในกรณีที่เลือกการส่งข้อมูลแบบ 9 บิต โดยการเซตบิต RX9 และยังสามารถใช้เป็นบิตพาริตี้ของการสื่อสารข้อมูลอนุกรมได้ส่วนกำเนิดอัตราเร็วในการถ่ายทอดข้อมูลหรือบอดเรตเจเนอเรเตอร์

ในโมดูล USART ใช้ส่วนกำเนิดอัตราเร็วในการถ่ายทอดข้อมูลหรือบอดเรตเจเนเรเตอร์เพียงชุดเดียว สามารถรองรับทั้งการทำงานในโหมดอะซิงโครนัสและซิงโครนัส โดยมีความละเอียด 8 บิต รีจิสเตอร์ควบคุมการทำงานคือ SPBRG โดยจะควบคุมคาบเวลาของไทเมอร์อิสระขนาด 8 บิตเพื่อกำหนดอัตราเร็วในการถ่ายทอดข้อมูล เมื่อทำงานในโหมดอะซิงโครนัสต้องทำงานร่วมกับข้อมูลที่บิต BRGH (บิต 2 ของรีจิสเตอร์ TXSTA) เพื่อกำหนดย่านของอัตราเร็วแบบต่ำและสูง กำหนดให้ความถี่สัญญาณนาฬิกาหลัก (หรือความถี่ของคริสตอล) เท่ากับ 16MHz ต้องการบอดเรตเท่ากับ 9,600 บิตต่อวินาที เลือกย่านบอดเรตต่ำ ทำงานเป็นแบบอะซิงโครนัส สามารสรุปความสัมพันธ์ทางคณิตศาสตร์ได้ดังนี้

ค่าบอดเรต = FOSC/64 (X+1)

$9,600 = 16x10^6/64 (X+1)$

X = 25.042 โดยค่าของ X นั้นคือค่าของรีจิสเตอร์ SPBRG แต่เนื่องจากข้อมูลในรีจิสเตอร์

SPBRG ต้องใช้เฉพาะจำนวนเต็มเท่านั้นจึงต้องกำหนดค่าของ SPBRG เป็น 25 แทนกลับเข้าไปในสูตรคำณวนค่าบอดเรตจะได้ $16x10^{6}$ /64 (25+1) = 9,615 บิตต่อวินาที

ขั้นตอนการเขียนโปรแกรมเพื่อส่งข้อมูลไปยังคอมพิวเตอร์

1. กำหนดค่าบอดเรตโดยป้อนค่าเข้าที่รีจิสเตอร์ SPBRG และเลือกโหมดของบอดเรตป็นโหมดความเร็วต่ำหรือสูงโดยกำหนดที่บิต BRGH ของรีจิสเตอร์ TXSTA ซึ่งค่าที่กำหนดนี้จะอ้างอิงถึงความถี่คริสตอลที่ป้อนให้กับ CPU สามารถศึกษาได้จากดาต้าชีท

2. เอ็นเอเบิลการทำงานของพอร์ตอนุกรมโดยเซตบิต SPEN ในรีจิสเตอร์ RCSTA เป็น 1 และเลือกการทำงานเป็นแบบอะซิงโครนัสโดยเคลียร์บิต SYNC ในรีจิสเตอร์ TXSTA

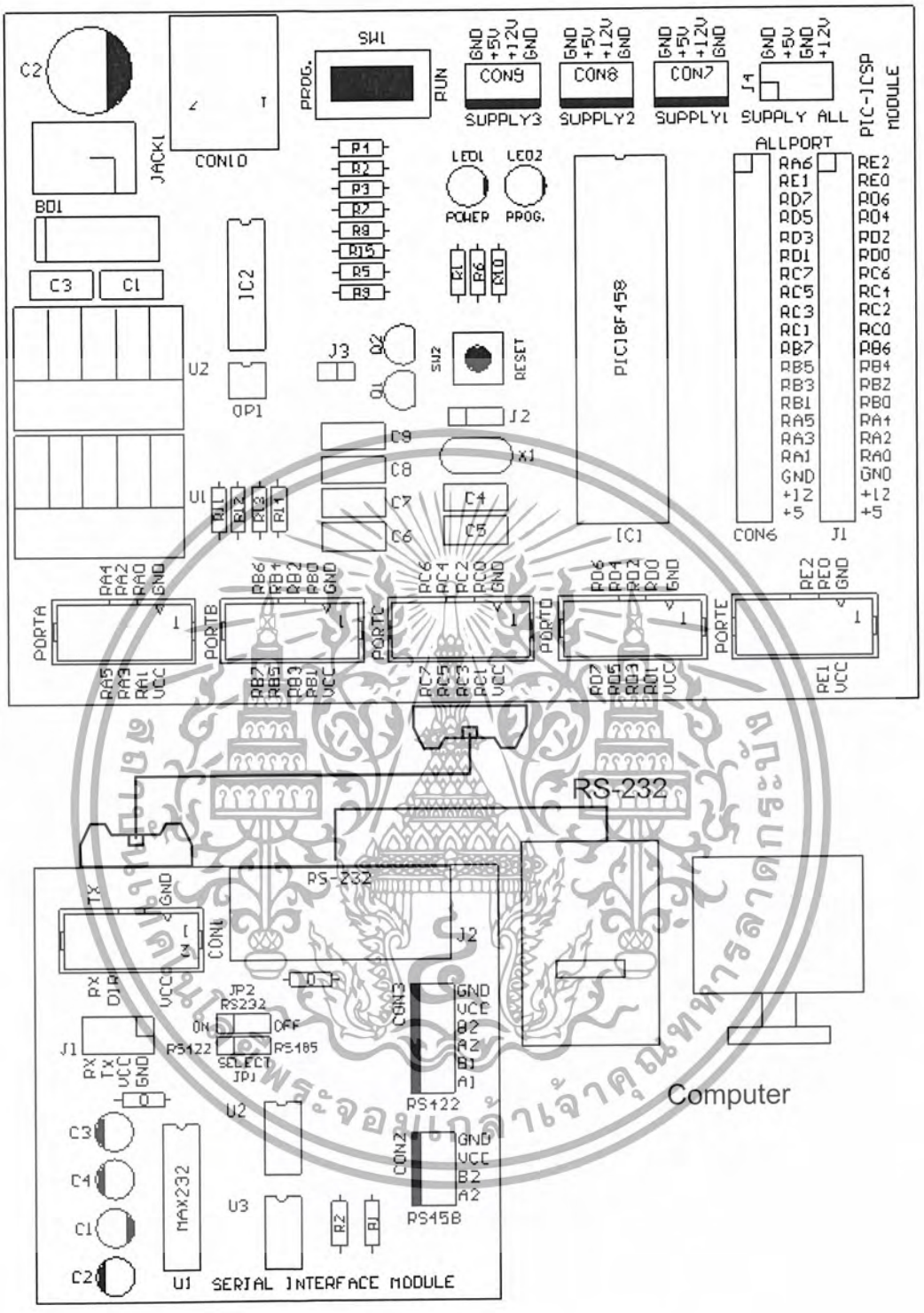
3. ถ้ามีการใช้งานการอินเตอร์รัพต์ต้องเอ็นเอเบิลการอินเตอร์รัพต์ด้วย

4. เอ็นเอเบิลการส่งข้อมูล โดยเซตบิต TXEN ในรีจิสเตอร์ TXSTA

5. นำข้อมูลที่จะส่งไปเก็บไว้ที่รีจิสเตอร์ TXREG

6. รอให้บัฟเฟอร์สำหรับส่งข้อมูลว่างแล้วค่อยส่งข้อมูลในไบต์ต่อๆไป

สำหรับรายระเอียดบิตต่างๆของรีจิสเตอร์ศึกษาได้จากภาคผนวกเรื่องรายระเอียดและคุณสมบัติของอุปกรณ์



รูปที่ จ.34 การเชื่อมต่อโมดูลสื่อสารข้อมูลอนุกรม กับคอมพิวเตอร์

```
        ; Filename :        lab13.asm
        ; Descripton :      Transmiit Data RS-232 to Computer
        ; MCU :             PIC18F458
        list    p=18F458                ; list directive to define processor
        #include <p18F458.inc>          ; processor specific variable definitions

COUNT   equ     0x20
        org     0x000
        goto    Start
Table   addwf   PCL,f                   ; Data table
        dt      "PIC18F458 EXPERIMENT SETS "
Start   call    Initial                 ; Set baud rate and enable data transmitting
        clrf    COUNT                   ;
Loop    call    Test_Empty              ; Check buffer empty
        movf    COUNT,w                 ; Use COUNT for convertion data table
        call    Table                   ; Get data from table
        movwf   TXREG                   ; Send to TXREG register
        incf    COUNT,f                 ; Increment counter
        movlw   .20                     ; Test counter = 20 times
        subwf   COUNT,w
        btfss   STATUS,Z
        goto    Loop                    ; If <> 20, loop again
        goto    $                       ; If = 20,  stop program here
Test_Empty      btfss   TXSTA,TRMT      ; If not empty, loop again
        goto    $-1
        return
Initial movlw   .25
        movwf   SPBRG                   ; Set 9600 buad rate
        bcf     TXSTA,SYNC              ; Set asynchonous Mode
        bsf     TXSTA,BRGH              ; Set high speed baudrate
        bsf     TXSTA,TXEN              ; Enable data transmittion
        bsf     RCSTA,SPEN              ; Enable serial port
        return
        end
```

รูปที่ จ.35 โปรแกรมการทดลองใช้งาน RS-232 สื่อสารกับคอมพิวเตอร์

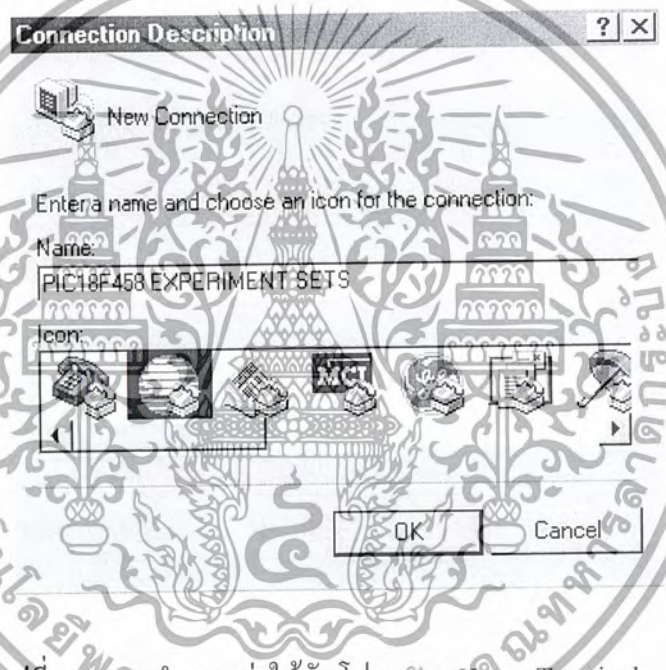
ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.35 ทำการคอมไพล์ให้เป็น Hex file

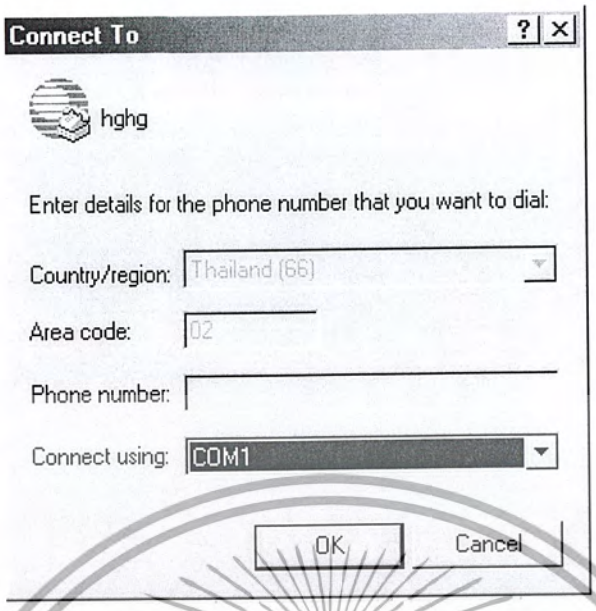
2. ต่อวงจรตามรูปที่ จ.34

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

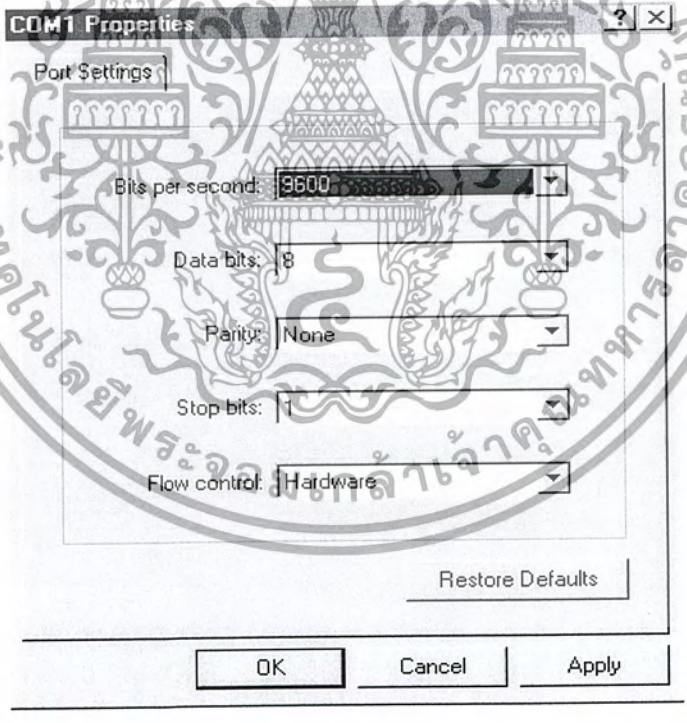
4. เปิดโปรแกรม Hyper Terminal ดังรูปที่ จ. 36 และ จ.37 ใส่ข้อมูลในการเชื่อมต่อเลือกไอคอนแล้วเลือกพอร์ตที่ใช้เชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ โดยส่วนใหญ่จะเป็น COM1 และ COM2



รูปที่ จ.36 การกำหนดค่าให้กับโปรแกรม Hyper Terminal



รูปที่ จ.37 การเลือกพอร์ตสำหรับต่อใช้งานพอร์ตอนุกรม



รูปที่ จ.38 การกำหนดค่าให้กับโปรแกรม Hyper Terminal เพื่อรับข้อมูล

1. กำหนดค่าให้กับโปรแกรม Hyper Terminal ดังรูปที่ จ.38 โดยเลือกบอดเรตเท่ากับ 9,600 จำนวนบิตข้อมูลเท่ากับ 8 บิต ค่าพาราตี้กำหนดเท่ากับ None จากนั้นกดปุ่ม OK

2. จากการทดลองข้อที่ 6 สังเกตผลที่เกิดขึ้นกับหน้าต่างโปรแกรม Hyper Terminal

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ถามท้ายการทดลอง

1. จงบอกรีจิสเตอร์ที่เกี่ยวข้องกับโมดูลสื่อสารข้อมูลอนุกรมพร้อมทั้งอธิบายหน้าที่และการทำงานมาพอเข้าใจ

2. ในการกำหนดค่าอัตราบอดเรตเพื่อสื่อสารข้อมูลนั้นมีวิธีการอย่างไรในการเขียนโปรแกรมจงอธิบาย

ใบงานที่ 14
การทดลองอ่านและเขียนหน่วยความจำโปรแกรมแบบแฟลชของ PIC18F458

จุดประสงค์

1. เพื่อให้สามารถอธิบายหลักการอ่านและเขียนข้อมูลลงบนหน่วยความจำโปรแกรมได้
2. เพื่อให้สามารถเขียนโปรแกรมอ่านและเขียนข้อมูลลงบนหน่วยความจำโปรแกรมได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ Epic Win
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
5. โมดูลแสดงผลแอลอีดี

ทฤษฎีเบื้องต้น

หน่วยความจำโปรแกรมแบบแฟลช (Flash Program Memory)

หน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ PIC 18F458 มีขนาด 32 KBytes ซึ่งเป็นส่วนของการเก็บซอร์สโค๊ดของโปรแกรมนั่นก็คือโปรแกรมที่เราเขียนขึ้นแล้วผ่านกระบวนการคอมไพล์ออกมาเป็น Hex file นั่นเองและนอกจากเราจะอ่านข้อมูลออกมาได้จากเครื่องโปรแกรมแล้วยังสามารถอ่านข้อมูลออกมาได้ในขณะที่ซียูกำลังทำงานอยู่ได้ด้วยซึ่งกระบวนการอ่านก็จะลักษณะที่คล้ายกับหน่วยความจำอีอีพรอม โดยจะมีรีจิสเตอร์ที่เกี่ยวข้อง 6 ตัวดังนี้

1. EECON1 รีจิเตอร์ควบคุมการเข้าถึงหน่วยความจำ
2. EECON2 รีจิสเตอร์จัดลำดับการเขียนข้อมูลในหน่วยความจำ
3. EEDATA รีจิสเตอร์บัฟเฟอร์ข้อมูลสำหรับการอ่านและเขียน
4. EEDATH รีจิสเตอร์บัฟเฟอร์ข้อมูลไบต์สูง
5. EEADR รีจิสเตอร์ที่เก็บตำแหน่งมีค่าตั้งแต่แอดเดรสตั้งแต่ 00h- FFh

6. EEADRH รีจิสเตอร์แอดเดรสไบต์สูง

หน่วยความจำโปรแกรมของ PIC18F458 มีความพิเศษที่แตกต่างจากไมโครคอนโทรเลอร์เบอร์อื่นๆ ตรงที่สามารถเข้าถึงและแก้ไขได้ตลอดเวลา ภายใต้แรงดันไฟเลี้ยงปกติ โดยทั่วไปแล้วหน่วยความจำโปรแกรมจะสามารถเข้าถึงเพื่ออ่านข้อมูลได้เพียงอย่างเดียวแล้วนำข้อมูลนั้นไปทำงานตามที่กำหนดต่อไป แต่สำหรับหน่วยความจำโปรแกรมของ PIC18F458 มีความยืดหยุ่นมากกว่านั้นเนื่องจาก PIC18F458 ออกแบบมาเพื่อให้สามารถแก้ไขข้อมูลในหน่วยความจำโปรแกรมได้ด้วยคุณสมบัติที่เรียกว่า ICD (In-Circuit Debugger) โดยหลักการคือ แบ่งหน่วยความจำโปรแกรมออกเป็น 2 ส่วน ส่วนแรกใช้เก็บโปรแกรมมอนิเตอร์หลักหรือ Boot Loader ในส่วนนี้จะสามารถ โปรแกรมใหม่ได้ด้วยเครื่องโปรแกรมภายนอกส่วนที่สองคือส่วนเก็บโปรแกรมการทำงานของผู้ใช้งานหรือ User Code ในส่วนที่สองนี้สามารถแก้ไขได้ตามต้องการ จะขออธิบายขั้นตอนโดยละเอียดดังนี้

1. เขียนโปรแกรม Boot Loader ขึ้น โดยในโปรแกรมนี้จะตรวจสอบก่อนว่า ต้องการให้ไมโครคอนโทรลเลอร์ทำงานในโหมดใด ระหว่างโหมดรันหรือโหมดโปรแกรม เพื่อตรวจสอบได้ว่าต้องการเข้าสู่โหมดโปรแกรมก็จะรอรับข้อมูลที่เป็นรหัสเลขฐานสิบหกจากเครื่องคอมพิวเตอร์ที่ส่งเข้ามาอาจส่งเข้ามาทาง RB6 และ RB7 หรือรับจากขา TxD กับ RxD ก็ได้ เมื่อข้อมูลเข้ามาแล้วซีพียูจะเข้าสู่โปรแกรมย่อยการเขียนข้อมูลลงในหน่วยความจำโปรแกรม ซึ่งจะมีพื้นที่อยู่อีกส่วนหนึ่งต่างหาก

2. หลังจากที่กระบวนการเขียนสิ้นสุดลง จะต้องมีการตรวจสอบ สามารถทำได้โดยเขียนโปรแกรมย่อยการอ่านข้อมูลจากหน่วยความจำโปรแกรมใช้งานเปรียบเทียบข้อมูลที่ส่งมาจากคอมพิวเตอร์ การเปรียบเทียบเป็นแอดเดรส นั่นคือทำได้ 2 ลักษณะคือ เปรียบเทียบครั้งเดียวหลังจากที่เขียนเสร็จ หรือเปรียบเทียบเป็นแอดเดรสนั่นคือเขียนข้อมูล 1 แอดเดรสทำการอ่านแล้วเปรียบเทียบเพื่อตรวจสอบทันทีหากถูกต้องโปรแกรมใช้งานที่เขียนขึ้นมาใหม่ก็จะสามารถรันได้ทันที เมื่อไมโครคอนโทรลเลอร์ถูกเลือกให้ทำงานในโหมดรัน

3. บรรจุโปรแกรม Boot Loader ลงในหน่วยความจำโปรแกรมของ PIC18F458 ด้วยเครื่องโปรแกรมในไมโครคอนโทรลเลอร์ PIC ภายนอก หลังจากโปรแกรมแล้ว พื้นที่ในส่วนของ Boot Loader ผู้ใช้งานจะไม่สามารถเข้าไปแก้ไขได้ทุกครั้งที่ไมโครคอนโทรลเลอร์อยู่ในโหมดโปรแกรมหรือเมื่อโปรแกรม Boot Loader ถูกเรียกให้ทำงาน

การเขียนข้อมูลลงในหน่วยความจำโปรแกรม

การเข้าถึงหน่วยความจำโปรแกรมจะเหมือนกับหน่วยความจำข้อมูลอีอีพรอมคือไม่สามารถเข้าถึงได้โดยตรง แต่ต้องกระทำผ่านรีจิสเตอร์ฟังก์ชั่นพิเศษทั้ง 6 ตัวตามที่กล่าวมาแล้ว

ข้างต้นและสำหรับการเขียนข้อมูลลงในหน่วยความจำโปรแกรมมีเงื่อนไขที่สำคัญต้องกระทำ 2 ประการคือ ต้องไม่มีการป้องกันข้อมูล และทำการเซตบิต WRT ใน Configuration Word ซึ่งทั้งสองเงื่อนไขนี้จะกระทำตั้งแต่ในขั้นตอนการบรรจุโปรแกรม Boot loader ลงในหน่วยความจำโปรแกรมตั้งแต่แรก

ขั้นตอนในการเขียนข้อมูลมีดังนี้

1. กำหนดแอดเดรสก่อน ตามด้วยกำหนดข้อมูลที่ต้องการเขียน
2. เซตบิต EEPGD เพื่อเข้าถึงหน่วยความจำโปรแกรม แล้วเอ็นเอเบิลการเขียนข้อมูลด้วยการเซตบิต WREN ดิสเอเบิลการตอบสนองอินเตอร์รัพต์
3. เขียนข้อมูล 0x55 และ 0xAA ลงในรีจิสเตอร์ EECON2 ตามขั้นตอนที่เหมือนกับการเขียนข้อมูลลงในหน่วยความจำอีอีพรอม
4. หลังจากที่เขียนข้อมูลลงในหน่วยความจำโปรแกรมแล้ว ซีพียูอยู่ในสภาวะหยุดทำงาน (Halt) ไม่ใช่การเข้าสู่โหมดสลีป เพื่อรอให้กลไกของการเขียนข้อมูลเสร็จสมบูรณ์ ซึ่งใช้เวลาประมาณ 2 ไซเกิลของสัญญาณนาฬิกา ดังนั้นซีพียูจะกลับมาทำงานต่อในคำสั่งที่ 3 หลังจากเซตบิต WR ใน EECON1 เพื่อเริ่มต้นการเขียนข้อมูล ดังนั้นในระหว่างรอ ต้องบรรจุคำสั่ง nop 2 คำสั่งติดกันเมื่อการเขียนข้อมูลเสร็จสิ้นลง ซีพียูกลับมาทำงานก็จะสามารถเฟตช์คำสั่งที่ 3 ที่อยู่ต่อจาก nop ได้ทันที ซึ่งจะเป็นการเอ็นเอเบิลการตอบสนองอินเตอร์รัพต์ และปิดท้ายด้วยการดิสเอเบิลการเขียนข้อมูล

การอ่านข้อมูลจากหน่วยความจำโปรแกรม

เริ่มด้วยการกำหนดแอดเดรสที่ต้องการอ่านลงในรีจิสเตอร์ EEADR และ EEADRH เหมือนกับการเขียนข้อมูล ตามด้วยการกำหนดข้อมูล เซตบิต EEPGD และ RD ในรีจิสเตอร์ EECON1 เมื่อเข้าถึงและเอ็นเนเบิลการอ่านข้อมูลจากหน่วยความจำโปรแกรม เช่นเดียวกับการเขียนข้อมูลลงในหน่วยความจำโปรแกรม ซีพียูจะใช้เวลาประมาณ 2 ไซเกิลของการกระทำคำสั่งในการอ่านข้อมูล ในระหว่างรอจึงต้องแทรกคำสั่ง nop เข้าไป 2 คำสั่ง ข้อมูลที่อ่านได้จะบรรจุอยู่ในรีจิสเตอร์ EEDATA และรักษาข้อมูลนี้ไว้จนกว่าจะเกิดการอ่านหรือเขียนใหม่ขึ้น

การป้องกันข้อมูลในหน่วยความจำข้อมูลอีอีพรอมและหน่วยความจำโปรแกรม

1. การป้องกันข้อมูลในหน่วยความจำข้อมูลอีอีพรอม

การป้องกันการอ่านข้อมูลออกจากหน่วยความจำอีอีพรอมนั้นสามารถกระทำได้จากเครื่องโปรแกรมภายนอกเท่านั้น โดยเลือกฟังก์ชั่น EEPROM Protection หรืออาจมีชื่ออื่นที่แตกต่างออกไปและการป้องกันนี้จะป้องกันเฉพาะการอ่านโดยใช้เครื่องโปรแกรมภายนอกเท่านั้นโดยในระหว่างการทำงานปกติของไมโครคอนโทรลเลอร์ ซีพียูจะเข้าถึงหน่วยความจำข้อมูลอีอีพรอมได้

ตามปกติหรืออาจกล่าวได้ว่าเป็นการป้องกันการคัดลอกโดยใช้เครื่องอ่านหรือเครื่องโปรแกรมภายนอกนั่นเอง

2. การป้องกันข้อมูลในหน่วยความจำโปรแกรม

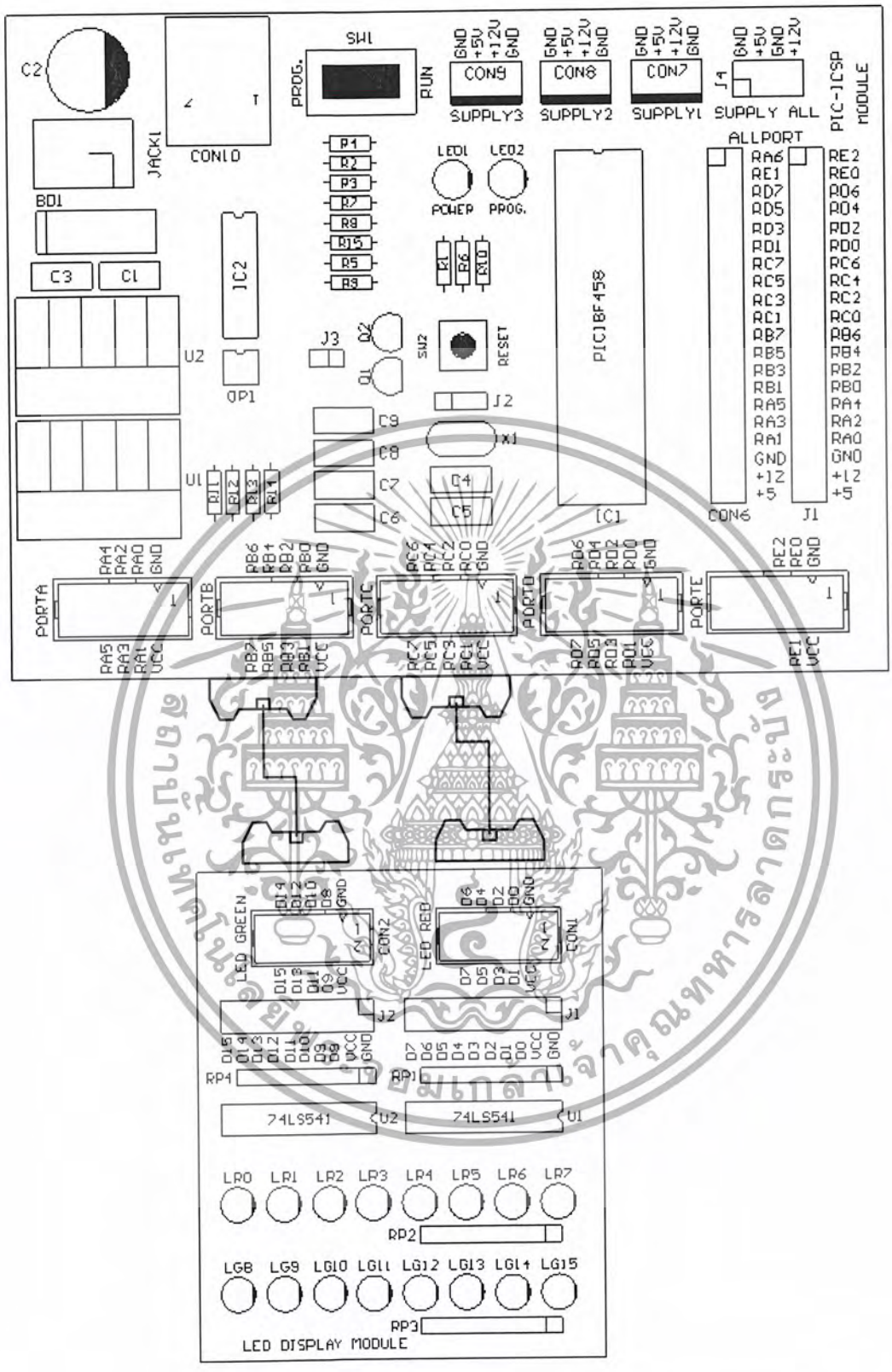
ในไมโครคอนโทรลเลอร์ PIC18F458 สามารถเลือกระดับการป้องกันการอ่านและเขียนข้อมูลในหน่วยความจำโปรแกรมอยู่หลายระดับ โดยการกำหนดสถานะของบิต CP0, CP1 และ WRT ใน Configuration Word แต่อย่างไรก็ตามการป้องกันที่เกิดขึ้นทั้งหมดมีจุดประสงค์หลักอยู่ 2 ประการคือ ป้องกันการอ่านข้อมูลจากเครื่องอ่านหรือเครื่องโปรแกรมภายนอกเพื่อจุดประสงค์ในการคัดลอกโปรแกรม และป้องกันการเขียนข้อมูลทับลงในหน่วยความจำโปรแกรมอันจะส่งผลให้การทำงานของไมโครคอนโทรลเลอร์ผิดพลาด

ขั้นตอนการอ่านเขียนหน่วยความจำโปรแกรม

1. กำหนดแอดเดรสไบต์สูงให้กับ EEADRH และไบต์ต่ำ EEDATA
2. กำหนดข้อมูลไบต์สูงให้กับ EEDATA และ ไบต์ต่ำให้กับ EEDATA
3. เซตบิต EEPGD ใน EECON1
6. เปิดการอินเตอร์รัพต์
7. เขียน 0xAA ไปยัง EECON2
8. เซตบิต WR ใน EECON1 เพื่อเริ่มกระบวนการเขียน
9. หน่วงเวลา 2 ไซเคิล
10. รอให้การเขียนข้อมูลเสร็จสมบูรณ์
11. เคลียร์อินเตอร์รัพต์แฟล็กบิต EEIF ในรีจิสเตอร์ PIR2
12. ดิสเอเบิลการเขียนข้อมูล

ขั้นตอนการอ่านเขียนหน่วยความจำโปรแกรม

1. กำหนดแอดเดรสให้กับ EEADR
2. เคลียร์บิต EEPGD ใน EECON1
3. เซตบิตRD ใน EECON1
4. อ่านข้อมูลจาก EECON2



รูปที่ จ.39 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับโมดูลแสดงผลแอลอีดี

```
;*********************************************************
; Filename :        lab13.asm
; Descripton :      Demonstrate flash memory writing
; Project           Pic18F458 Experiment Sets
        LIST    P=18F458
      #INCLUDE <P18F458.INC>
      START     ORG     0x000
;***************กำหนดพอร์ต***************************
        BCF     TRISB,0         ; RB0 = output
        BCF     PORTB,0         ; Turn-off LED
        CALL    WRITE
        GOTO    $-1
;***************กำหนดการเขียนข้อมูล*****************************
WRITE   MOVLW   0x1F            ; High address of EEPROM
        MOVWF   EEADRH
        MOVLW   F8              ; Low address of EEPROM
        MOVWF   EEADR
        MOVLW   0x04            ; Data high byte
        MOVWF   EEDATH
        MOVLW   0x40            ; Data low byte
        MOVWF   EEDATA
        BSF     EECON1,EEPGD    ; Program EEPROM section
        BSF     EECON1,WREN     ; Enable writing
        MOVLW   0x55            ; Writing sequence
        MOVWF   EECON2
        MOVLW   0xAA
        MOVWF   EECON2
;***************เริ่มการเขียนข้อมูล*************************
        BSF     EECON1,WR       ; Writing start here
        NOP                     ; Delay 2 cycles
        NOP
        BTFSC   EECON1,WR       ; Skip if completed
        GOTO    $-1             ; Wait
        BCF     EECON1,WREN     ; Disable writing
        BSF     PORTB,0         ; Turn-on LED
        RETURN
        END
```

รูปที่ จ.40 โปรแกรมการทดลองเขียนข้อมูลลงบนหน่วยความจำโปรแกรม

```
;*********************************************************
; Filename :        lab14-2.asm
; Descripton :      Demonstrate flash program memory reading
; Project           Pic18F458 Experiment Sets
;*********************************************************
        LIST    P=18F458
        #INCLUDE <P18F458.INC>
      START     ORG     0x00
;*************** กำหนดพอร์ต ********************
        CLRF    TRISB           ; PORTB = output
        CLRF    TRISC           ; PORTC = output;
        CLRF    PORTB
        CLRF    PORTC
READ
        CLRF    EEADRH          ; Address = 0x0050
        MOVLW   0x50
        MOVWF   EEADR
LOOP
        BSF     EECON1,EEPGD    ; Program memory section
        BSF     EECON1,RD       ; Read enable
        NOP                     ; Delay 2 cycles
        NOP
        MOVFW   EEDATA
        MOVWF   PORTB
        MOVFW   EEDATH          ; Read data
        MOVWF   PORTC           ; Show data
        GOTO    $               ; Stop

        ORG     0x0050
        FILL    0x1678,1

        END
```

รูปที่ จ.41 โปรแกรมการทดลองอ่านข้อมูลจากหน่วยความจำโปรแกรม

ลำดับขั้นการทดลอง

1. จากผังโปรแกรมที่ได้ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.40 ทำการคอมไพล์ให้เป็น Hex File

2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

3. ต่อวงจรตามรูปที่ จ.39

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่เกิดขึ้นกับโมดูลแสดงผลแอลอีดี

.....

.....

.....

5. เปิดโปรแกรม Epic Win ไปที่เมนู View เลือก Data สังเกตผลจากโปรแกรมบันทึกผลการทดลอง

ค่าของข้อมูลในแอดเดรส 0x1FE8 = ...

6. ทดลองเปลี่ยนโปรแกรมเพื่อกำหนดแอดเดรสให้กับ EEADRH และ EEADRL แล้วทำตามการทดลองในข้อ 1- 5 ใหม่อีกครั้ง

7. เขียนโปรแกรมตามรูปที่ จ.39 ทำการคอมไพล์ให้เป็น Hex File

8. ต่อวงจรตามรูปที่ จ.41

9. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่เกิดขึ้นกับโมดูลแสดงผลแอลอีดีที่ต่ออยู่กับ PORTC และ PORTB

สังเกตผลที่แอลอีดีที่ต่ออยู่กับพอร์ต C แปลงเป็นเลขฐานสิบหก = ...

สังเกตผลที่แอลอีดีที่ต่ออยู่กับพอร์ต D แปลงเป็นเลขฐานสิบหก = ...

10. ทดลองเปลี่ยนค่าข้อมูลและแอดเดรสที่คำสั่ง FILL แล้วทำการทดลองใหม่รันโปรแกรมสังเกตผลที่กับโมดูลแสดงผลแอลอีดีบันทึกผลการทดลอง

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงอธิบายหลักการเขียนและอ่านข้อมูลจากหน่วยความจำโปรแกรมแบบแฟลชของไมโครคอนโทรลเลอร์ PIC18F458

2. จงอธิบายความแตกต่างของการอ่านและเขียนหน่วยความจำอีอีพรอมและหน่วยความจำโปรแกรมว่ามีวิธีการหรือแตกต่างกันอย่างไร

ใบงานที่ 16
การทดลองใช้งานโหมดสลีปของ PIC18F458

จุดประสงค์

1. เพื่อให้เข้าใจถึงการทำงานในโหมดประหยัดพลังงานหรือโหมดสลีปของ PIC18F458 ได้
2. เพื่อให้ใช้งานไมโครคอนโทลเลอร์ในวงจรที่ต้องการลดการจ่ายกระแสไฟฟ้าในขณะที่ไม่ทำงานได้
3. เพื่อให้สามารถประยุกต์แก้ไขโปรแกรมในการทดลองได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ Epic Win
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลหลัก PIC-ICSP
5. โมดูลแสดงผลแอลอีดี
6. โมดูลสวิตช์พื้นฐาน

ทฤษฎีเบื้องต้น

ในไมโครคอนโทรลเลอร์ PIC18F458 เมื่อไม่มีการใช้งานในช่วงระยะเวลาหนึ่งสามารถกำหนดให้ไมโครคอนโทรลเลอร์ทำงานในโหมดที่กินกำลังงานต่ำได้ซึ่งจะช่วยให้สามารถประหยัดพลังงานที่ป้อนให้กับตัวไมโครคอนโทรลเลอร์ได้หลายเท่าทีเดียว

สำหรับคำสั่งที่ทำให้ไมโครคอนโทรลเลอร์ทำงานในโหมดประหยัดพลังงานหรือโหมดสลีป (Sleep Mode) นั้นคือ คำสั่ง Sleep เมื่อ PIC18F458 เอ็กซิคิวต์คำสั่ง Sleep วอตช์ด็อกไทเมอร์จะเครียร์บิต PD (ในรีจิสเตอร์ RCON บิต 2) จะเคลียร์วงจรกำเนิดสัญญาณนาฬิกาจะหยุดทำงาน พอร์ตเอาต์พุตที่ถูกสั่งให้ทำงานก่อนหน้าคำสั่ง Sleep จะยังคงทำงานต่อไป ไม่ว่าจะถูกขับด้วยลอจิก “1”, “0” หรือเป็นอิมพิแดนซ์สูงก็ตาม

การทำงานในโหมด Sleep นี้ขา MCLR จะต้องได้รับลอจิก “1” อย่างต่อเนื่องตลอดเวลาที่ยังอยู่ในโหมดสลีปนี้ เนื่องจากเมื่อออกจากโหมดสลีป PIC18F458 ควรจะทำงานต่อไปแต่ถ้าหากขา MCLR ได้รับลอจิก “0” PIC18F458 จะเกิดการรีเซ็ต ทำให้ต้องไปเริ่มต้นทำงานที่ตอนต้นของโปรแกรมเสมอไม่สามารถทำงานต่อเนื่องต่อไปได้

รีจิสเตอร์ RCON

รีจิสเตอร์ RCON เป็นรีจิสเตอร์ที่บรรจุบิตสถานะต่างๆ เกี่ยวกับการเกิดรีเซตแบบต่างๆ โดย รีจิสเตอร์ RCON สามารถเซตและเคลียร์ได้ด้วยกระบวนการทางซอฟต์แวร์

ตารางที่ จ.20 รายระเอียดบิตต่างๆ ของรีจิสเตอร์ RCON

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
IPEN	-	-	RI	TO	PD	POR	BOR
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
R : อ่านค่าได้, W : เขียนค่าได้, U : ไม่ใช้งานอ่านค่าได้เป็นศูนย์, -n : ค่าที่เกิดขึ้นหลังเกิดเพาเวอร์ออนรีเซต							

บิต 7 : IPEN (Interrupt Priority Enable Bit) บิตเอ็นเอเบิล (Enable) การเลือกลับดับการเกิดอินเตอร์รัพต์

1 = เอ็นเอเบิลการเลือกลำดับการเกิดอินเตอร์รัพต์

0 = ไม่มีการเลือกลำดับการเกิดอินเตอร์รัพต์

บิต 6-5 : ไม่ใช้อ่านค่าได้เป็นศูนย์

บิต 4 : RI (Reset Instruction Flag Bit) บิตแสดงสถานะการรีเซตจากคำสั่งทางซอฟต์แวร์

1 = มีการเกิดรีเซต

0 = ไม่มีการเกิดรีเซต

บิต 3 :TO (Watchdog Time Out Flag Bit) บิตแสดงสถานะเกิดไทม์เอาต์ของวอตซ์ด็อกไทเมอร์

1 = หลังจากเกิดเพาเวอร์อัปไทเมอร์ เมื่อใช้คำสั่ง CLRWDT และ SLEEP

0 = เมื่อเกิดไทม์เอาต์ของวอตซ์ด็อกไทเมอร์

บิต 2 : PD (Power Down Detection Flag Bit) บิตแสดงสถานะการเกิด Power Down

1 = หลังจากเกิด เพาเวอร์อัปไทเมอร์ หรือ มีการใช้คำสั่ง CLRWDT

0 = เมื่อกระทำคำสั่ง SLEEP

บิต 1 : POR (Power On Reset Status Bit) บิตแสดงสถานะของการเกิดเพาเวอร์ออนรีเซต

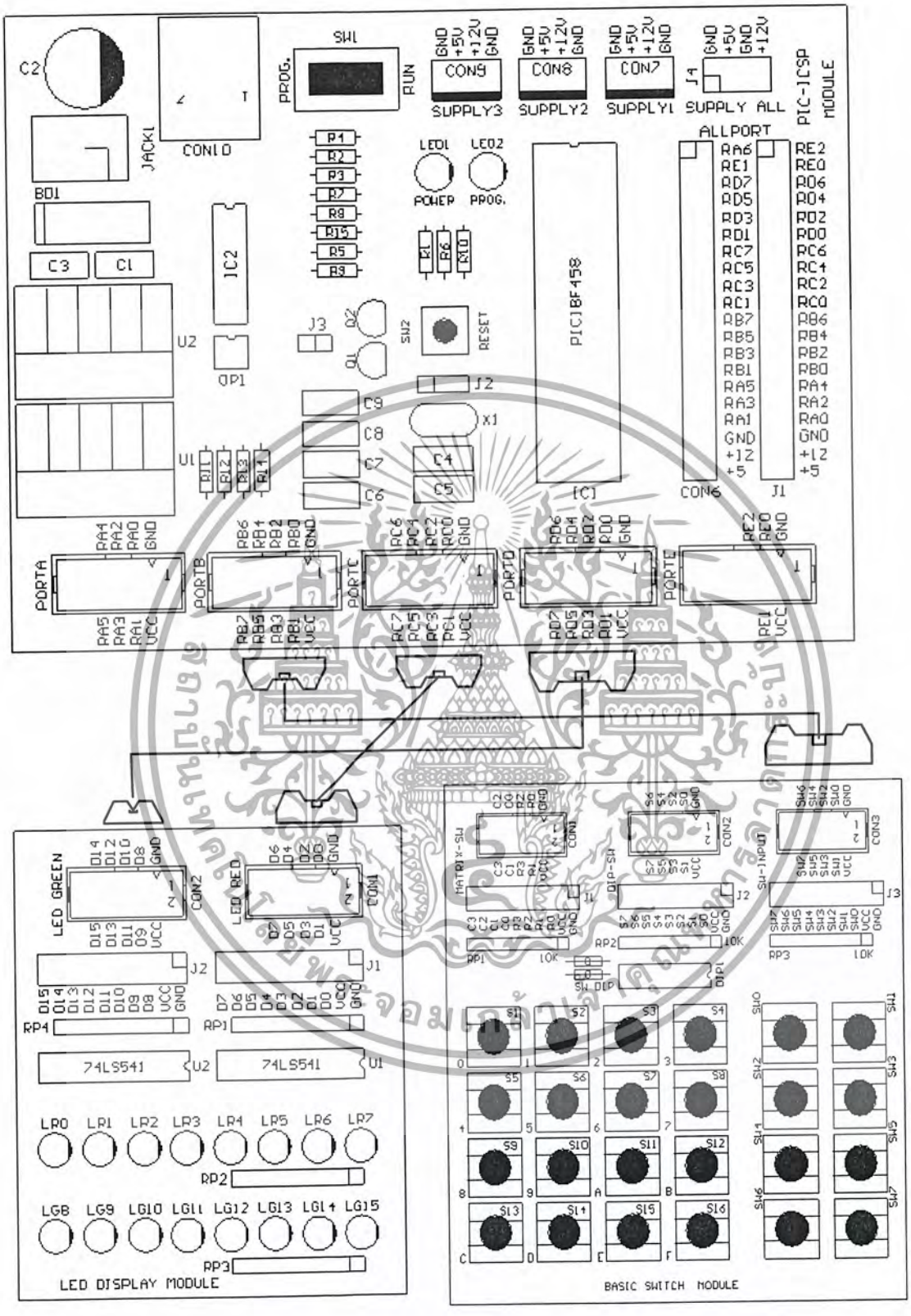
1 = ไม่มีการเกิดเพาเวอร์ออนรีเซต

0 = เกิดเพาเวอร์ออนรีเซต (ควรจะเซตด้วยซอฟต์แวร์)

บิต 0 : BOR (Brown Out Reset Status Bit) บิตแสดงสถานะการเกิดบราวเอาต์รีเซต

1 = ไม่มีการเกิดบราวเอาต์รีเซต

0 = มีการเกิดบราวเอาต์รีเซต (ควรจะเซตด้วยซอฟต์แวร์)



รูปที่ จ.44 การเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับโมดูลสวิตช์พื้นฐานแสดงผลที่โมดูลแสดงผลแอลอีดี

```
;*************************************************************
; Filename :          lab16.asm
; Descripton :        Test  Sleep Mode
; MCU :               PIC18F458
;*************************************************************
list        p=18f458
  #include <p18f458.inc>
ROUND   EQU     0x20
COUNT1  EQU     0x21     ; RAM
COUNT2  EQU     0x22     ; RAM
COUNT3  EQU     0x23     ; RAM
        ORG     0x0000
        MOVLW   B'00000000'
        MOVWF   TRISD          ;Portd  All output
        MOVLW   B'00000000'
        MOVWF   TRISC          ;Portc All output
        MOVLW   B'00000001'
        MOVWF   TRISB          ; RB0 is input
        CLRF    PORTC
        CLRF    PORTD
START   MOVLW   0X0A
        MOVWF   COUNT3
        MOVLW   0XF0
        MOVWF   PORTD
        CALL    DELAY
BLINK   MOVLW   0XF0
        MOVWF   COUNT3
        XORWF   PORTD,F
        CALL    DELAY
        DECFSZ  COUNT3,F
        GOTO    BLINK
        CALL    DISPLAY
        BSF     INTCON,INTE
        SLEEP
        NOP
        CALL    DISPLAY
        MOVLW   C0
        MOVWF   PORTB
LOOP    CLRWDT
        GOTO    LOOP
```

```
DELAY    MOVLW    0XAA
         MOVWF    COUNT1
DELAY2   CLRWDT
         MOVLW    0XFF
         MOVWF    COUNT2
DELAY1   DECFSZ   COUNT2,F
         GOTO     DELAY1
         DECFSZ   COUNT1,F
         GOTO     DELAY2
         BCF      STATUS,C
         RETURN
;**********DISPLAY STATUS  T0,PD ON LED**********
DISPLAY  BTFSS    RCON,PD
         GOTO     CLR_PD
         BSF      PORTC,0
         GOTO     CLR_PD
CHK_TO   BTFSS    RCON,TO
         GOTO     CLR_TO
         BSF      PORTC,1
         RETURN
CLR_PD   BCF      PORTC,0
         GOTO     CHK_TO
CLR_TO   BCF      PORTC,1
         RETURN
         END
```

รูปที่ จ.45 โปรแกรมทดลองใช้งานพอร์ตของ PIC18F458

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.45 ทำการคอมไพล์ให้เป็น Hex File
2. ต่อวงจรตามรูปที่ จ.44
3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นกับแอลอีดีที่ต่ออยู่กับ PORTD บันทึกผลที่ได้จากการทดลอง

.....

.....

.....

5. ปลดแหล่งจ่ายไฟออกจากวงจรจากนั้นป้อนแหล่งจ่ายไฟเข้าอีกครั้ง (เพื่อให้เกิดเพาเวอร์ออนรีเซต) สังเกตการทำงานของ LED0 และ LED1 หลังจากที่ LED12 – LED15 หยุดกระพริบ

LED0 ติดหรือดับ.....

LED1 ติดหรือดับ.....

6. หลังจากนั้นกดสวิตช์ S0 สังเกตการเปลี่ยนแปลงที่เกิดขึ้นที่ LED0 – LED1

LED0 ติดหรือดับ.....

LED1 ติดหรือดับ.....

7. กดสวิตช์ S0 มีผลอย่างไรต่อการทำงานของวงจร

.....

.....

.....

8. ทำการโปรแกรม PIC18F458 ใหม่อีกครั้ง โดยครั้งนี้กำหนดให้วอตช์ดอกไทเมอร์ ON เพื่อสังเกตการเปลี่ยนแปลงที่บิต TO

9. ทำการรันโปรแกรม โดยการเลื่อนไปที่สวิตช์ RUN

10. สังเกตการทำงานของ LED0 และ LED1 แล้วบันทึกผลที่เกิดขึ้น

.....

.....

.....

11. จากการทดลอง PIC18F458 เข้าสู่โหมดสลีปเมื่อใด และเวกอัปได้อย่างไร

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงให้คำจำกัดความของโหมดสลีป
2. การเวกอัป คืออะไร เกี่ยวข้องอย่างไรกับการทำงานในโหมดสลีปของ PIC18F458
3. จงอธิบายความสัมพันธ์ระหว่างบิต TO และ PD กับการทำงานในโหมดสลีปของ PIC18F458
4. รีจีสเตอร์ที่เกี่ยวข้องกับการทำงานในโหมดสลีปมีกี่ตัว อะไรบ้าง และเกี่ยวข้องอย่างไร
5. จงยกตัวอย่างประโยชน์ของการทำงานในโหมดสลีปของ PIC18f458 มาให้เห็นอย่างชัดเจนอย่างน้อย 1 ตัวอย่าง

ใบงานที่17
การทดลองใช้งานโมดูลเปรียบเทียบแรงดันแอนะลอกของ PIC18F458

จุดประสงค์

1. เพื่อให้สามารถอธิบายหลักการเปรียบเทียบแรงดันแอนะลอกของ PIC18F458 ได้
2. เพื่อให้สามารถเขียนโปรแกรมใช้งานเปรียบเทียบแรงดันแอนะลอกของ PIC18F458 ได้
3. เพื่อให้สามารถแก้ไขโปรแกรมในการทดลองได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องคอมพิวเตอร์ที่ติดตั้งโปรแกรม MPLAB และ Epic Win
2. สายเชื่อมต่อระหว่างโมดูลหลัก PIC-ICSP กับคอมพิวเตอร์
3. สายเชื่อมต่อระหว่างโมดูล
4. โมดูลแสดงผลแอลอีดี
5. ตัวต้านทานปรับค่าได้ 10 กิโลโอห์ม

ทฤษฎีเบื้องต้น

โมดูลเปรียบเทียบแรงดันอนาลอก PIC18F458

ใน PIC18F458 มีส่วนจัดการสัญญาณแอนะลอกเพิ่มมาอีก 2 ส่วนคือ โมดูลเปรียบเทียบแรงดันอะนาลอก (Analog Comparator Module) และโมดูลสร้างแรงดันอ้างอิง (Voltage Reference Module)

โมดูลเปรียบเทียบแรงดันอะนาลอก

ประกอบด้วยวงจรเปรียบเทียบแรงดัน 2 ชุด โดยอินพุตของวงจรเปรียบเทียบนั้นจะใช้ร่วมกับขาพอร์ต RA0-RA3 โดยสามารถใช้แรงดันจากโมดูลกำเนิดแรงดันอ้างอิงมาต่อเข้ากับอินพุตของวงจรเปรียบเทียบได้ ส่วนเอาต์พุตของโมดูลนี้คือขา RA4 และ RA5

โมดูลเปรียบเทียบแรงดันแอนะลอกนี้สามารถเลือกโหมดการทำงานได้ 8 แบบ อินพุตของโมดูลเปรียบเทียบแรงดันแอนะลอกสามารถรับแรงดันแอนะลอกโดยตรงได้ตั้งแต่ VSS (คือกราวด์) ถึง VDD (คือไฟเลี้ยง) เมื่อแรงดันที่อินพุต Vin+ มากกว่า Vin- เอาต์พุตของวงจรเปรียบเทียบ

จะเป็นลอจิกสูง ในทางตรงข้ามหากแรงดันที่อินพุต Vin+ น้อยกว่า Vin- เอาต์พุตของวงจรเปรียบเทียบจะเป็นลอจิกต่ำ ในกรณีที่ทำงานในโหมดไม่กลับลอจิก

สำหรับเวลาในการทำงานของโมดูลเปรียบเทียบแอนะลอกใน PIC18F458 อยู่ระหว่าง 150 นาโนวินาที ถึง 10 ไมโครวินาที ขึ้นอยู่กับการเลือกแหล่งจ่ายอ้างอิง หากใช้แรงดันอ้างอิงจากภายนอก วงจรเปรียบเทียบจะใช้เวลา 150-600 นาโนวินาที แต่ถ้าหากใช้แรงดันอ้างอิงจากภายนอกจะใช้เวลาสูงสุด10 ไมโครวินาที

CMCON รีจีสเตอร์ควบคุมการทำงานของโมดูลเปรียบเทียบแรงดันแอนะลอก

ใช้กำหนดและควบคุมการทำงานของโมดูลเปรียบเทียบแรงดันแอนะลอกรายละเอียดในแต่ละบิตของรีจีสเตอร์ CMCON มีดังนี้

ตารางที่ จ.21 บิตต่างของรีจีสเตอร์ CMCON

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x

C2OUT (Comparator 2 Output : บิต7) บิตแสดงผลการเปรียบเทียบของวงจรเปรียบเทียบแรงดันชุดที่ 2 โดยการแสดงผลนั้นจะสัมพันธ์กับการกำหนดสถานะที่บิต C2INV โดย

ถ้าบิต C2INV เป็น “0”

C2OUT เป็น “0” เมื่อแรงดันที่อินพุต C2 VIN+ น้อยกว่า C2 VIN-

C2OUT เป็น “1” เมื่อแรงดันที่อินพุต C2 VIN+ มากกว่า C2 VIN-

ถ้าบิต C2INV เป็น “1”

C2OUT เป็น “0” เมื่อแรงดันที่อินพุต C2 VIN+ มากกว่า C2 VIN-

C2OUT เป็น “1” เมื่อแรงดันที่อินพุต C2 VIN+ น้อยกว่า C2 VIN-

C1OUT (Comparator 1 Output : บิต6) บิตแสดงผลการเปรียบเทียบของวงจรเปรียบเทียบแรงดันชุดที่ 1 โดยการแสดงผลนั้นจะสัมพันธ์กับการกำหนดสถานะที่บิต C1INV โดย

ถ้าบิต C1INV เป็น “0”

C1OUT เป็น “0” เมื่อแรงดันที่อินพุต C1 VIN+ น้อยกว่า C1 VIN-

C1OUT เป็น “1” เมื่อแรงดันที่อินพุต C1 VIN+ มากกว่า C1 VIN-

ถ้าบิต C1INV เป็น “1”

C1OUT เป็น “0” เมื่อแรงดันที่อินพุต C1 VIN+ มากกว่า C1 VIN-

C1OUT เป็น “1” เมื่อแรงดันที่อินพุต C1 VIN+ น้อยกว่า C1 VIN-

C2INV (Comparator 2 Output Inversion : บิต5) บิตกลับสภาวะเอาต์พุตของวงจรเปรียบเทียบชุดที่ 2 ใช้สำหรับกำหนดการแสดงผลเอาต์พุตของวงจรเปรียบเทียบแรงดันชุดที่ 2 ว่าจะให้แสดงผลกลับตรงข้ามหรือไม่

“0” - เลือกให้การแสดงผลทางเอาต์พุตไม่กลับ

“1” - เลือกให้การแสดงผลทางเอาต์พุตมีการกลับสถานะ

C1INV (Comparator 1 Output Inversion : บิต4) บิตกลับสภาวะเอาต์พุตของวงจรเปรียบเทียบชุดที่ 1 ใช้สำหรับกำหนดการแสดงผลเอาต์พุตของวงจรเปรียบเทียบแรงดันชุดที่ 1 ว่าจะให้แสดงผลกลับตรงข้ามหรือไม่

“0” - เลือกให้การแสดงผลทางเอาต์พุตไม่กลับ

“1” - เลือกให้การแสดงผลทางเอาต์พุตมีการกลับสถานะ

CIS (Comparator Input Switch : บิต3) บิตเลือกขาพอร์ตอินพุตให้แก่โมดูลเปรียบเทียบแรงดัน เมื่อเลือกโหมดการทำงานเป็นวงจรเปรียบเทียบแรงดัน 2 ชุด ที่มีแรงดันอ้างอิงร่วมกันและส่งค่าออกทางขาพอร์ต หรือโหมด “110” ดังมีรายละเอียดต่อไปนี้

“0” - กำหนดให้ขาอินพุต C1 VIN- ต่อกับขาพอร์ต RA0/AN0

กำหนดให้ขาอินพุต C2 VIN- ต่อกับขาพอร์ต RA1/AN1

“1” - กำหนดให้ขาอินพุต C1 VIN- ต่อกับขาพอร์ต RA3/AN3

กำหนดให้ขาอินพุต C2 VIN- ต่อกับขาพอร์ต RA2/AN2

CM2-CM0 (Comparator Mode : บิต2-0) บิตเลือกโหมดทำงานของโมดูลเปรียบเทียบแรงดัน

“000” - อยู่ในสภาวะรีเซต ไม่มีการทำงาน

“001” – ทำงานเป็นโมดูลเปรียบเทียบแรงดันอิสระ 1 ชุด โดยใช้โมดูลเปรียบเทียบชุดที่ 1 และมีการส่งค่าออกเอาต์พุต

“010” – ทำงานเป็นโมดูลเปรียบเทียบแรงดัน 2 ชุด ที่ทำงานแยกอิสระจากกัน

“011” – ทำงานเป็นโมดูลเปรียบเทียบแรงดัน 2 ชุด ที่ทำงานอิสระจากกันและมีการส่งค่าออกเอาต์พุต

“100” – ทำงานเป็นโมดูลเปรียบเทียบแรงดัน 2 ชุด ที่ใช้ขาอินพุตรับแรงดันอ้างอิงจากภายนอกร่วมกัน

"101" – ทำงานเป็นโมดูลเปรียบเทียบแรงดัน 2 ชุด ที่ใช้ขาอินพุตรับแรงดันอ้างอิงจากภายนอกร่วมกันและมีการส่งค่าออกเอาต์พุต

"110" – ทำงานเป็นโมดูลเปรียบเทียบแรงดัน 2 ชุด ที่มีแรงดันอ้างอิงร่วมกันจากโมดูลสร้างแรงดันอ้างอิงภายใน PIC18F458 และมีการส่งค่าออกทางขาพอร์ต โดยสามารถเลือกขาพอร์ตอินพุตได้ขึ้นอยู่กับการกำหนดที่บิต CIS

"111" - วงจรเปรียบเทียบแรงดันไม่ทำงาน

โหมดการทำงานของแรงดันเปรียบเทียบแรงดันอะนาลอก

โหมดรีเซต (CM2-CM0 : 000)

โมดูลเปรียบเทียบแรงดันจะไม่มีการทำงานใดๆ ในโหมดนี้เมื่อไมโครคอนโทรลเลอร์รีเซต โมดูลเปรียบเทียบแรงดันจะรีเซตด้วยเช่นกัน เอาต์พุตอ่านค่าได้ "0"

โหมดวงจรเปรียบเทียบอิสระ 1 ชุด (CM2-CM0 : 001)

เป็นการเลือกใช้โมดูลเปรียบเทียบแรงดันเพียงชุดเดียวเพื่อให้มีพอร์ตเหลือสามารถไปใช้งานอื่นได้โมดูลที่ใช้ คือ โมดูลเปรียบเทียบแรงดันชุดที่ 1 ซึ่งเปรียบเทียบแรงดันระหว่างอินพุต RA0/AN0 กับ RA3/AN3 ผลการเปรียบเทียบจะปรากฏที่บิต C1OUT และขาพอร์ต RA4/C1OUT

โหมดวงจรเปรียบเทียบอิสระ 2 ชุดแบบอิสระ (CM2-CM0 : 010)

ในโหมดนี้โมดูลเปรียบเทียบแรงดันทั้ง 2 ชุดจะทำงานแยกจากกันอย่างอิสระ โดยชุดที่ 1 เปรียบเทียบแรงดันระหว่างอินพุต RA0/AN0 กับ RA3/AN3 แล้วแจ้งผลการเปรียบเทียบที่บิต CIOUT ส่วนชุดที่ 2 จะเปรียบเทียบแรงดันระหว่างอินพุต RA1/AN1 กับ RA2/AN2 แจ้งผลการทำงานผ่านทางบิต C2OUT

โหมดวงจรเปรียบเทียบ 2 ชุด แบบอิสระและส่งค่าออกเอาต์พุต (CM2-CM0 : 011)

จะคล้ายกับโหมด 010 ที่เพิ่มขึ้นมาคือ สามารถส่งผลการเปรียบเทียบออกไปทางเอาต์พุต RA4/C1OUT สำหรับโมดูลเปรียบเทียบแรงดันชุดที่ 1 และขา RA5/C2OUT สำหรับโมดูลชุดที่ 2

โหมดวงจรเปรียบเทียบ 2 ชุดใช้แรงดันอ้างอิงภายนอกร่วมกัน (CM2-CM0 : 100)

ในโหมดนี้กำหนดให้โมดูลเปรียบเทียบแรงดันอะนาลอกทำงาน 2 ชุด โดยมีเอาต์พุตแยกกัน แต่ใช้แรงดันอ้างอิงภายนอกร่วมกันโดยป้อนแรงดันอ้างอิงเข้าที่อินพุต RA3/AN3 ซึ่งต่อกันขา Vin+ ของโมดูลเปรียบเทียบแรงดันภายใน PIC18F458 ทั้ง 2 ชุด ส่วนแรงดันที่นำมาเปรียบเทียบสำหรับโมดูลเปรียบเทียบแรงดันชุดที่ 1 ให้ป้อนเข้าที่อินพุต RA0/AN0 ในขณะที่โมดูลชุดที่ 2 ให้ป้อนเข้าที่อินพุต RA1/AN1 ส่วนอินพุต RA2/AN2 จะถูกกำหนดให้ทำงานเป็นพอร์ตดิจิตอล

โหมดวงจรเปรียบเทียบ 2 ชุดใช้แรงดันอ้างอิงภายนอกร่วมกันและส่งค่าออกเอาต์พุต (CM2-CM0 : 101)

จะคล้ายในโหมด 100 ที่เพิ่มเติมขึ้นมาคือ สามารถส่งผลการเปรียบเทียบออกไปทางเอาต์พุต RA4/C1OUT สำหรับโมดูลเปรียบเทียบแรงดันชุดที่ 1 และขา RA5/C2OUT สำหรับโมดูลชุดที่ 2

โหมดวงจรเปรียบเทียบ 2 ชุดใช้แรงดันอ้างอิงภายในร่วมกันและส่งค่าออกเอาต์พุต (CM2-CM0 : 110)

ในโหมดนี้มีการทำงานคล้ายกับโหมด 101 แต่แรงดันอ้างอิงที่ใช้จากโมดูลแรงดันอ้างอิงภายในไมโครคอนโทรลเลอร์ PIC18F458 และผู้ใช้งานสามารถเลือกขาอินพุตของโมดูลเปรียบเทียบแรงดันแต่ละชุดได้ โดยโมดูลเปรียบเทียบแรงดันชุดที่ 1 จะเปรียบเทียบแรงดันระหว่างอินพุต RA0/AN0 หรือ RA3/AN3 กับแรงดันอ้างอิงภายใน ขึ้นอยู่กับการกำหนดบิต CIS หากบิต CIS เป็น "0" จะเปรียบเทียบแรงดันอินพุต RA0/AN0 กับแรงดันอ้างอิงภายใน และถ้าบิต CIS เป็น "1" จะเลือกแรงดันอินพุต RA3/AN3 มาเปรียบเทียบแทน

โหมดหยุดการทำงาน (CM2-CM0 : 111)

ในโหมดนี้วงจรเปรียบเทียบทั้งหมดจะหยุดทำงานหรือปิด (off) นั่นหมายความว่าขาพอร์ต A ทั้งหมดจะถูกกำหนดให้ทำงานเป็นขาพอร์ตดิจิตอลแทนที่บิต C1OUT และ C2OUT อ่านค่าเป็น "0"

ส่วนเอาต์พุตของโมดูลเปรียบเทียบแรงดัน

ผลการทำงานของโมดูลเปรียบเทียบแรงดันใน PIC18F458 แสดงออก 2 ทางคือ ผ่านทางรีจีสเตอร์ CMCON ที่บิต C1OUT และ C2OUT อีกทางหนึ่งคือ ส่งค่าออกทางขาพอร์ต RA4 และ RA5 นอกจากนั้นที่ขาพอร์ต RA4 และ RA5 ยังมีหน้าที่อื่นทับซ้อนอยู่ด้วย แต่จะไม่มีการทำงานพร้อมกันในแต่ละหน้าที่ เมื่อเลือกให้โมดูลเปรียบเทียบแรงดันทำงานในโหมดที่ส่งค่าออกทางขาพอร์ตการกำหนดทิศทางของขาพอร์ตที่รีจีสเตอร์ TRISA ยังจำเป็นต้องกระทำอยู่ โดยหากต้องการให้มีสัญญาณออกมาที่ขาพอร์ต RA4 และ RA5 เป็นเอาต์พุตเสมอ

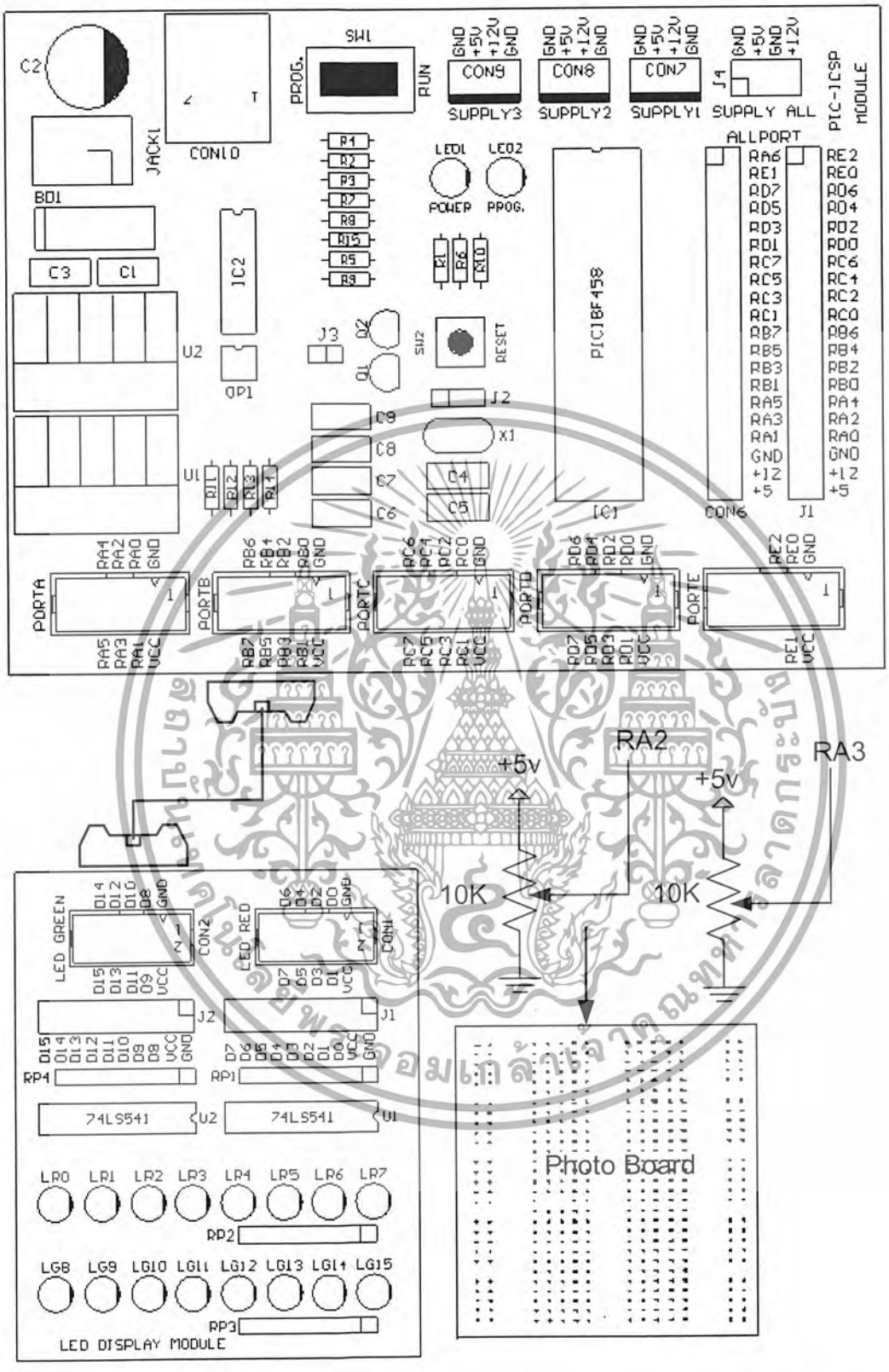
การอินเตอร์รัพต์ในโมดูลเปรียบเทียบแรงดันอะนาลอก

การอินเตอร์รัพต์เนื่องจากโมดูลเปรียบเทียบแรงดันจะเกิดขึ้นเมื่อมีการเปลี่ยนแปลงค่าเอาต์พุตของโมดูลเปรียบเทียบแรงดันทั้งสองชุด ดังนั้นในการเขียนโปรแกรมเพื่อใช้งานโมดูลเปรียบเทียบแรงดันอะนาลอกจำเป็นอย่างยิ่งที่จะต้องดูแลสถานะที่บิต 6 และ 7 ของรีจิสเตอร์ CMCON เนื่องจากทั้งข้อมูลของทั้งสองบิตนั้นจะเป็นตัวกำหนดเงื่อนไขในการเปลี่ยนแปลง เช่น หากข้อมูลเดิมเป็น "0" เมื่อข้อมูลเปลี่ยนเป็น "1" การอินเตอร์รัพต์จะเกิดขึ้นหากมีการเอ็นเอเบิลไว้

เมื่อเกิดการเปลี่ยนแปลงขึ้นที่เอาต์พุตของวงจรเปรียบเทียบ จะเกิดการเซตบิต CMIF (บิต6 ในรีจิสเตอร์ PIR2) ถ้าหากมีการเอ็นเอเบิลการอินเตอร์รัปต์แบบนี้ไว้ ก็จะเกิดการอินเตอร์-รัพต์ขึ้นและซีพียูในไมโครคอนโทรลเลอร์ก็ต้องกระโดดไปทำงานบริการอินเตอร์รัพต์ต่อไป

การทำงานของโมดูลเปรียบเทียบแรงดันเมื่ออยู่ในโหมดสลีป

เมื่อ PIC18F458 ทำงานในโหมดสลีป โมดูลเปรียบเทียบแรงดันยังคงทำงานอยู่ต่อไป หากเอ็นเอเบิลไว้ เมื่อเกิดการเปลี่ยนแปลงค่าเอาต์พุตขึ้น และเอ็นเอเบิลการอินเตอร์รัพต์ไว้ ซีพียูก็จะออกจากโหมดสลีปเพื่อตอบสนองการอินเตอร์รัพต์ที่เกิดขึ้น อย่างไรก็ตามเมื่อ PIC18F458 ออกจากโหมดสลีป โดยที่โมดูลเปรียบเทียบแรงดันยังทำงานอยู่ จะเกิดกระแสไฟฟ้าสูญเสียจำนวนมากขึ้น ดังนั้นถ้าไม่มีการใช้งานโมดูลเปรียบเทียบแรงดันในโหมดสลีป ควรปิดการทำงานก่อนเพื่อลดพลังงานสูญเสียที่เกิดขึ้นเมื่อไมโครคอนโทรลเลอร์กลับมาทำงานในโหมดปกติอย่างไรก็ตามข้อมูลใน CMCON จะไม่ได้รับผลกระทบใดๆ เมื่อโมดูลเปรียบเทียบแรงดันหยุดทำงานในโหมดสลีป



รูปที่ จ.46 การเชื่อมต่อสายระหว่างโมดูลหลัก PIC-ICSP กับโมดูลแสดงผลแอลอีดี

```
;*************************************************************
        ; Filename :            lab17.asm
        ; Descripton :          Analog Comparator Operation in PIC18F458
        ; MCU :                 PIC18F458
;*************************************************************
                list        p=18F458
        #include <p18F458.INC>
        COUNT   EQU         0x20
                ORG         0x0000
                MOVLW   0x01
                MOVWF   TRISD                   ; RD0 = digital port
                MOVLW   B'00000010'
                MOVWF   ADCON1
NON_INVERT      MOVLW   B'00000011'             ; C2OUT = non-invert logic ,mode 3
 ;INVERT        MOVLW   B'00100011'             ; C2OUT = invert logic, mode 3
                MOVWF   CMCON
OPERATE         CALL    DELAY                   ; Delay ~10 microsecond
                GOTO    OPERATE
        DELAY   MOVLW   .17
                MOVWF   COUNT
                DECFSZ  COUNT,F
                GOTO    $-1
                RETURN
                END
```

รูปที่ จ.47 โปรแกรมการทดลองใช้งานโมดูลเปรียบเทียบแรงดันแอนะล็อกของ PIC18F458

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.47 เสร็จแล้วทำการคอมไพล์ให้เป็น Hex File
2. ต่อสายเชื่อมต่อระหว่างโมดูลตามรูปที่ จ.46
3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

4. ทำการปรับค่าความต้านทานที่ต่ออยู่กับ RA1 ให้เท่ากับ 2 โวลต์ โดยสังเกตการเปลี่ยนแปลงที่เกิดขึ้นวลต์มิเตอร์ต่อวัดแรงดันจากนั้นปรับค่าแรงดันที่ต้านทานที่ต่ออยู่กับ RA1 ให้เท่ากับ 0 โวลต์ สังเกตผลที่เกิดขึ้นกับแอลอีดี

5. จากข้อ 4 ผลการทดลองเป็นดังนี้

.....

.....

.....

6. ทำการปรับค่าตัวต้านทานที่ต่ออยู่กับขา RA2 ขึ้นช้าๆ พร้อมทั้งใช้โวลต์มิเตอร์ต่อวัดแรงดันจนกระทั่งมีค่า 2 โวลต์ สังเกตผลที่เกิดขึ้นกับแอลอีดี

7. จากข้อ 6 ผลการทดลองเป็นดังนี้

.....

.....

.....

8. ทดลองปรับค่าความต้านทานที่ต่ออยู่กับ RA2/AN2 ให้มีค่ามากกว่า 2 โวลต์ สังเกตผลที่เกิดขึ้นกับโมดูลแอลอีดี

9. จากข้อ 8 ผลการทดลองเป็นดังนี้

.....

.....

.....

10. ทดลองนำเครื่องหมายหน้า INVERT ออกทำการคอมไพล์และโปรแกรมข้อมูลอีกครั้งและทำตามขั้นตอนในข้อ 4-9 อีกครั้ง สังเกตผลที่เกิดขึ้นกับโมดูลแอลอีดี

11. จากข้อ 10 ผลการทดลองเป็นดังนี้

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

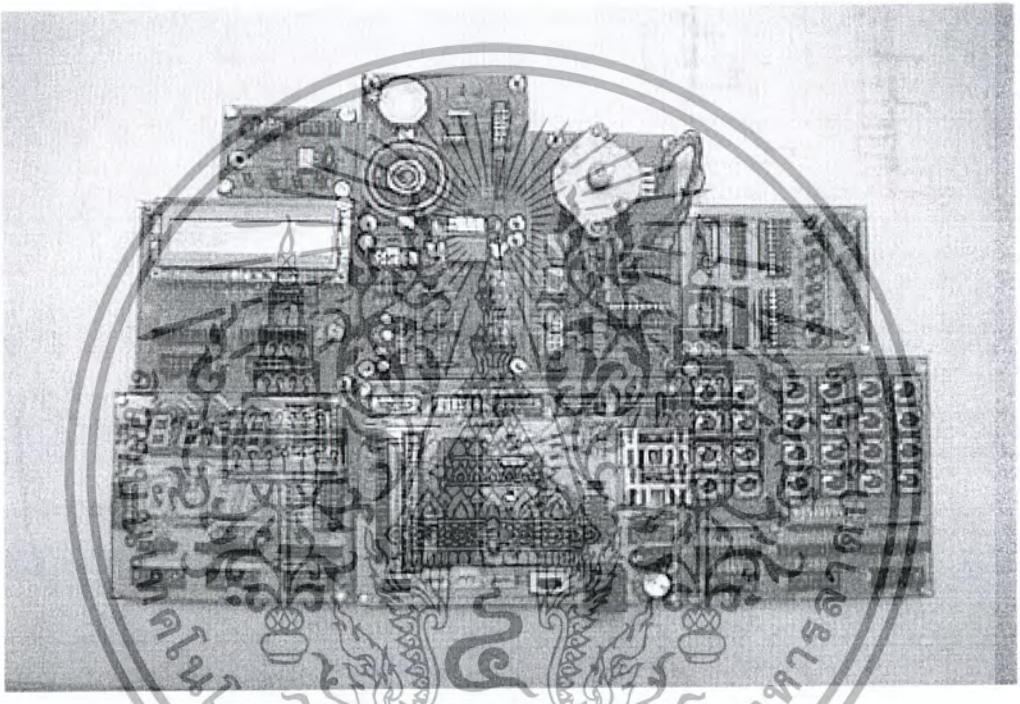
.....

คำถามท้ายการทดลอง

1. จงบอกริสเตอร์ที่เกี่ยวข้องกับโมดูลเปรียบเทียบแรงดันแอนะลอกของ PIC18F458 พร้อมอธิบายหน้าที่และการทำงาน
2. จงอธิบายโปรแกรมในรูปที่ จ.47

ภาคผนวก ฉ
คู่มือการใช้งาน

คู่มือการใช้งาน
ชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458

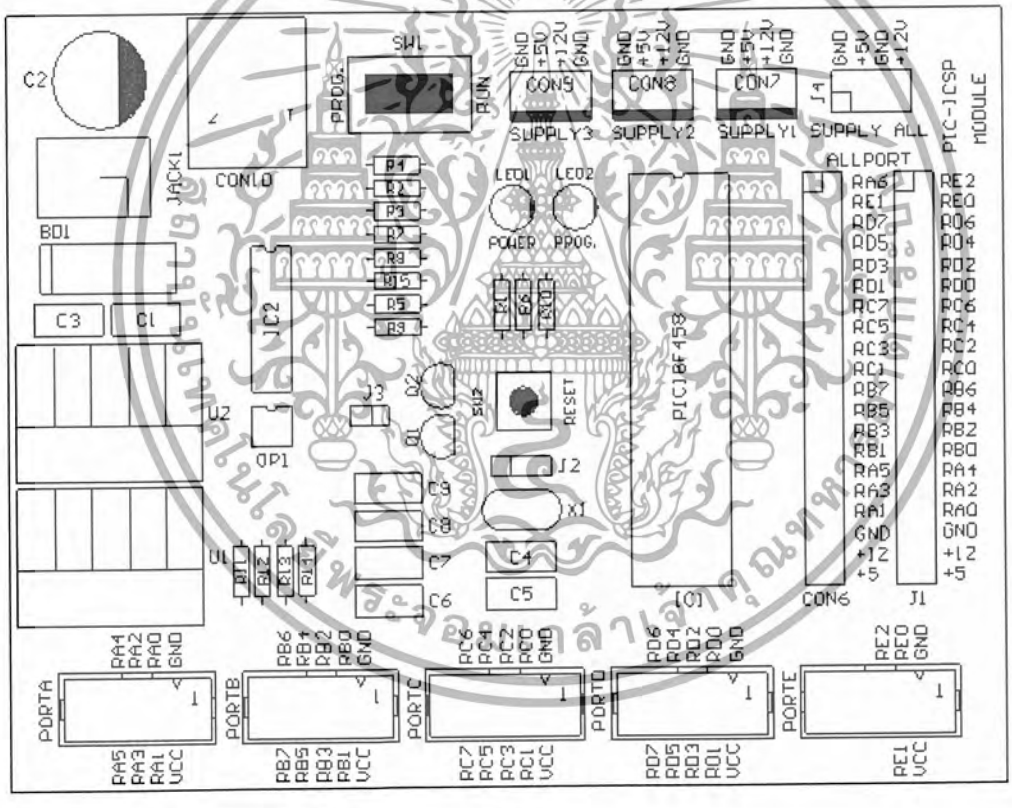


ภาควิชาครุศาสตร์วิศวกรรม
คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2546

1.1 คู่มือการใช้งานโมดูลชุดทดลองไมโครคอนโทรลเลอร์ PIC18F458

ชุดทดลองไมครคอนโทรลเลอร์ PIC18F458 ได้ออกแบบมาเพื่อให้ใช้งานสะดวกมากขึ้น โดยได้แบ่งแยกออกเป็นโมดูลเมื่อต้องการใช้งานโมดูลที่เกี่ยวข้องกับการทดลองก็สามารถใช้งานได้ โดยมีคอนเนคเตอร์เชื่อมต่อระหว่างโมดูลต่างๆ โดยมีให้เลือก 2 แบบ คือ แบบเชื่อมต่อเข้ากับพอร์ตทั้งชุดโดยจะเป็นคอนเนคเตอร์ตัวผู้ โดยจะมีสัญลักษณ์ที่ใช้แทนคอนเนคเตอร์แบบนี้ คือ CON ส่วนอีกแบบคือแบบที่เลือกบิตในการเชื่อมต่อได้ จะเป็นคอนเนคเตอร์ตัวเมียมีช่องสำหรับเชื่อมต่อสายเพื่อจั๊มไปใช้งานยังบิตต่างๆ โดยจะมีสัญลักษณ์ที่ใช้แทนคอนเนคเตอร์แบบนี้คือตัวอักษรตัว J นำหน้า

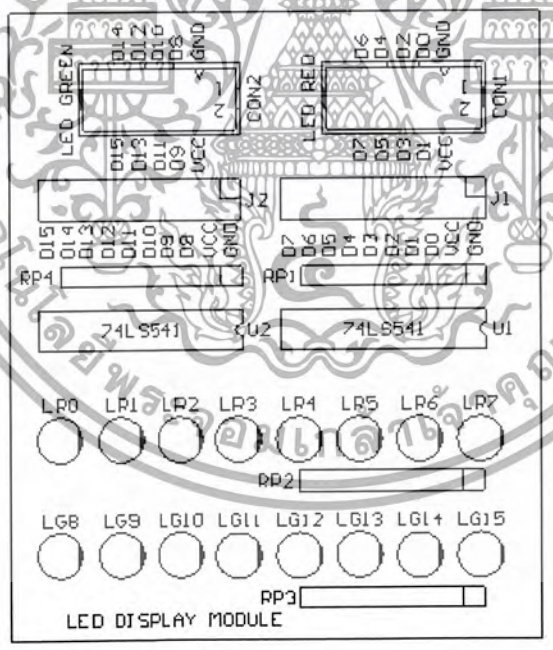
1.1.1 โมดูลหลัก PIC-ICSP (PIC- In-Circuit Serial Programming Module)



รูปที่ ฉ.1 จุดเชื่อมต่อใช้งานของโมดูลหลัก PIC-ICSP

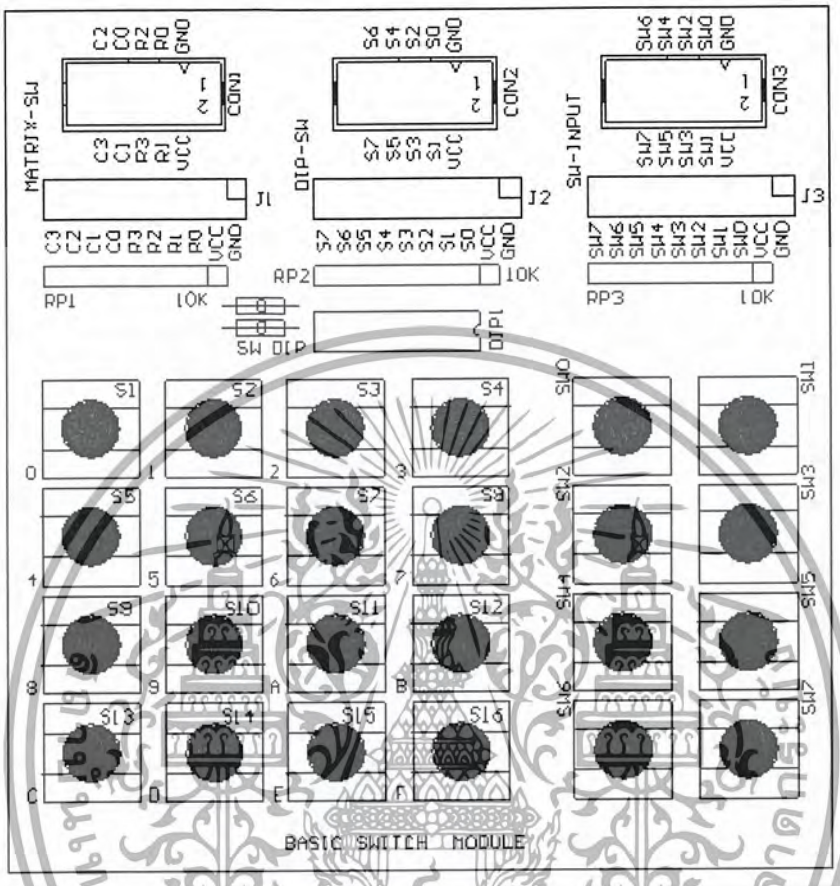
PORTA – PORTE	เป็นคอนเนคเตอร์โดยใช้เชื่อม ต่อพอร์ต A ไล่เรียงไปถึงพอร์ต E ตามลำดับ
JACK 1	ใช้เชื่อมต่อกับแหล่งจ่ายไฟเลี้ยงให้แก่โมดูลหลัก PIC-ICSP
J2	ใช้จั๊มเพื่อสลับเลือกใช้งานสัญญาณนาฬิกาแบบ XT และ RC
CON 6 (ALL PORT)	เป็นคอนเนคเตอร์ที่รวมพอร์ตเลือกใช้งานพอร์ตทั้งหมดของ PIC18F458 มีทั้งตัวผู้และตัวเมีย
SUPPLY 1-SUPPLY 3	เป็นคอนเนคเตอร์ต่อแหล่งจ่ายไฟไปใช้งานยังโมดูลต่างๆ
J4 (SUPPLY ALL)	เป็นคอนเนคเตอร์ต่อแหล่งจ่ายไฟไปใช้งานยังโมดูลต่างๆ
CON 10	เป็น RJ-45 ในการเชื่อมต่อระหว่างพอร์ตขนานของคอมพิวเตอร์
SW 1	เป็นสวิตช์ในการเลือกว่าจะใช้งานในโหมดโปรแกรมหรือใช้งานในโหมดรันโปรแกรม
SW 2	เป็นสวิตช์รีเซตไมโครคอนโทรลเลอร์

1.1.2 โมดูลแสดงผลแอลอีดี (LED Display Module)



รูปที่ ฉ.2 จุดเชื่อมต่อใช้งานของโมดูลแสดงผลแอลอีดี

1.4.3 โมดูลสวิตช์พื้นฐาน (Basic Switch)

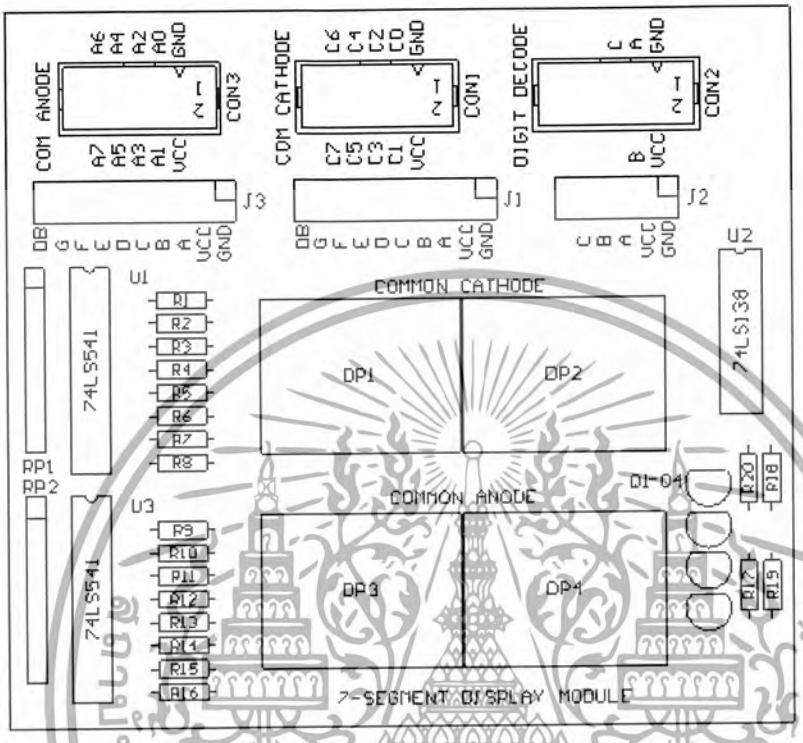


รูปที่ ฉ.3 จุดเชื่อมต่อใช้งานของโมดูลสวิตช์พื้นฐาน

CON 1	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานเมตริกซ์สวิตช์
J1	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานเมตริกซ์สวิตช์แบบเลือกบิตเชื่อมต่อได้
CON 2	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานดิพสวิตช์
J2	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานดิพสวิตช์แบบเลือกบิตเชื่อมต่อได้
CON 3	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานสวิตช์อินพุต
J3	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานสวิตช์อินพุต
CON 1	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานแอลอีดีแสดงผล 8 บิต สีแดง
CON 2	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานแอลอีดีแสดงผล 8 บิต สีเขียว
J1	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานแอลอีดีแสดงผล 8 บิต สีแดง

J2 พอร์ตใช้งานแอลอีดีแสดงผล 8 บิต สีเขียว

1.1.3 โมดูลแสดงผลแอลอีดีตัวเลขเจ็ดส่วน (Seven Segment Display Module)



รูปที่ ฉ.4 จุดเชื่อมต่อใช้งานของโมดูลแสดงผลแอลอีดีตัวเลขเจ็ดส่วน

CON 1 เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานข้อมูลแสดงผล 8 บิต ของคอมมอนคาโถด

CON 2 เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานในการเลือกคอมมอนโดยใช้ 3 บิตล่าง

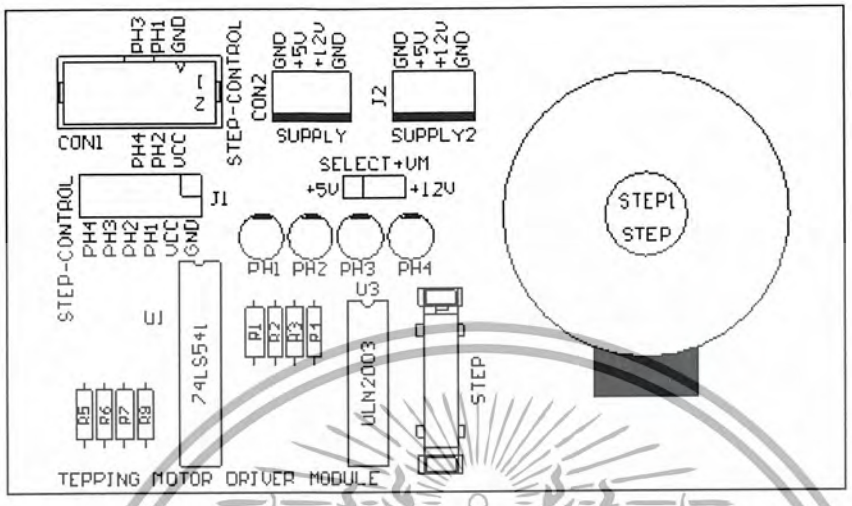
CON 3 เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานข้อมูลแสดงผล 8 บิต ของคอมมอนคาโถด

J1 เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานข้อมูลแสดงผล 8 บิต ของคอมมอนอาโนด

J2 เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานข้อมูลแสดงผล 8 บิต ของคอมมอนคาโถด

J3 เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตต่อใช้งานในการเลือกคอมมอน

1.1.4 โมดูลขับสเตปปิ้งมอเตอร์ (Stepping Motor Driver Module)



รูปที่ ฉ.5 จุดเชื่อมต่อใช้งานของโมดูลขับสเตปปิ้งมอเตอร์

CON 1 เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตต่อใช้งานในการควบคุมสเตปปิ้งมอเตอร์

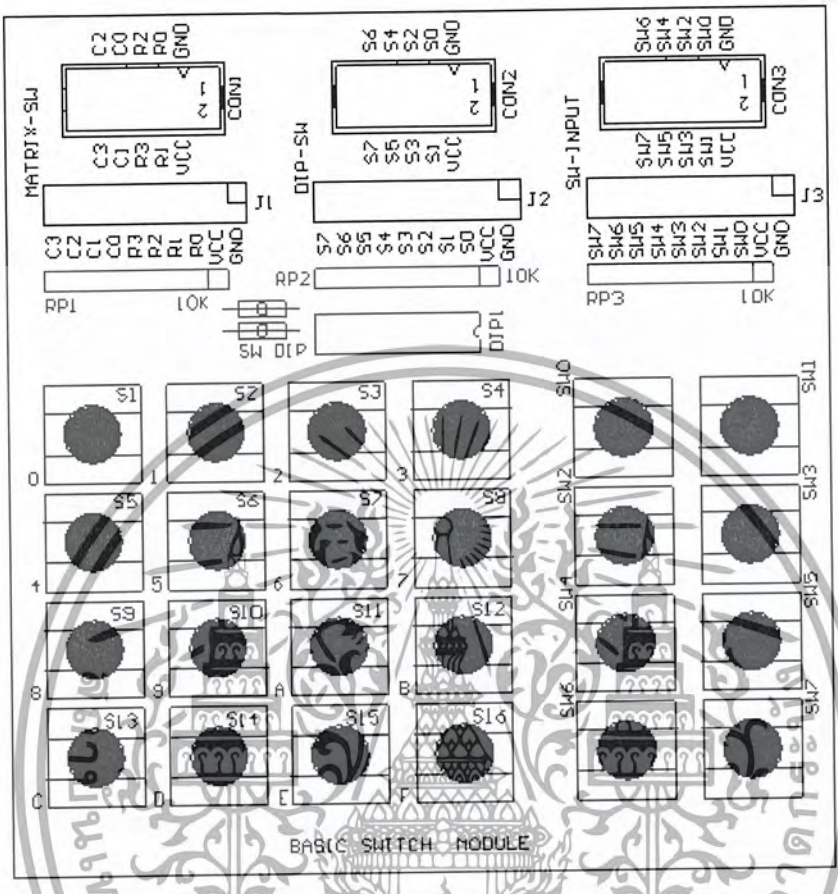
J1 เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตต่อใช้งานในการควบคุมสเตปปิ้งมอเตอร์แบบเลือกบิตเชื่อมต่อได้

SELECT +VM จัมเปอร์เลือกระดับแรงดัน VM 12 โวลต์ และ 5 โวลต์

CON 2 เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตสำหรับรับแหล่งจ่ายไฟ (VM) ให้แก่สเตปปิ้งมอเตอร์

J2 เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตสำหรับรับแหล่งจ่ายไฟ (VM) ให้แก่สเตปปิ้งมอเตอร์

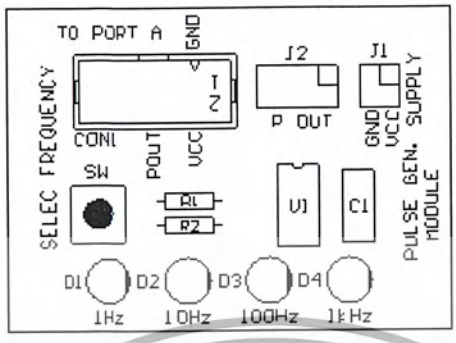
1.4.5 โมดูลสวิตช์พื้นฐาน (Basic Switch)



รูปที่ ฉ.6 จุดเชื่อมต่อใช้งานของโมดูลสวิตช์พื้นฐาน

CON 1	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานเมตริกซ์สวิตช์
J1	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานเมตริกซ์สวิตช์แบบเลือกบิตเชื่อมต่อได้
CON 2	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานดิพสวิตช์
J2	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานดิพสวิตช์แบบเลือกบิตเชื่อมต่อได้
CON 3	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานสวิตช์อินพุต
J3	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานสวิตช์อินพุต

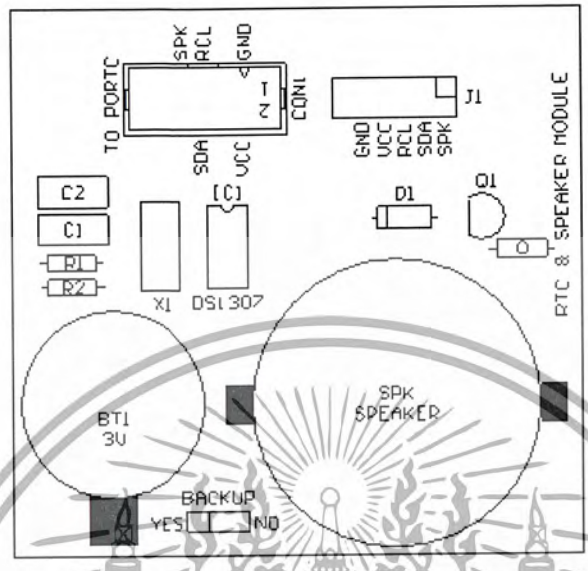
1.4.6 โมดูลพัลส์เจเนอเรเตอร์ (Pulse Generator Module)



รูปที่ ฉ.7 จุดเชื่อมต่อใช้งานของโมดูลโมดูลพัลส์เจเนอเรเตอร์

CON 1	เป็นคอนเนคเตอร์ใช้งานร่วมกับพอร์ต C ขา RA4 เพื่อทดลองส่งพัลส์ให้กับไมโครคอนโทรลเลอร์
J1	เป็นคอนเนคเตอร์พอร์ตเชื่อมต่อเพื่อจ่ายไฟเลี้ยง +5V ให้แก่โมดูล
J2	เป็นคอนเนคเตอร์เชื่อมต่อพอร์ตใช้งานร่วมกับพอร์ต C ขา RA4 เพื่อทดลองส่งพัลส์ให้กับไมโครคอนโทรลเลอร์
SW	ใช้กดเพื่อเลือกความถี่โดยที่ได้ 4 ค่า คือ 1 Hz 10 Hz 100Hz และ 1 KHz

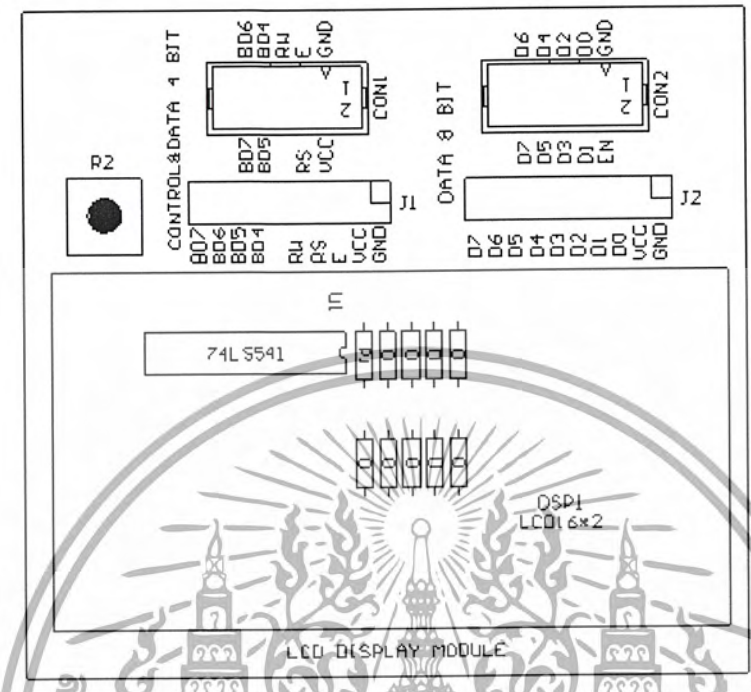
1.4.7 โมดูลเชื่อมต่ออุปกรณ์ RTC / SPEAKER



รูปที่ ฉ.8 จุดเชื่อมต่อใช้งานของโมดูลโมดูลพัลส์เจนอเรเตอร์

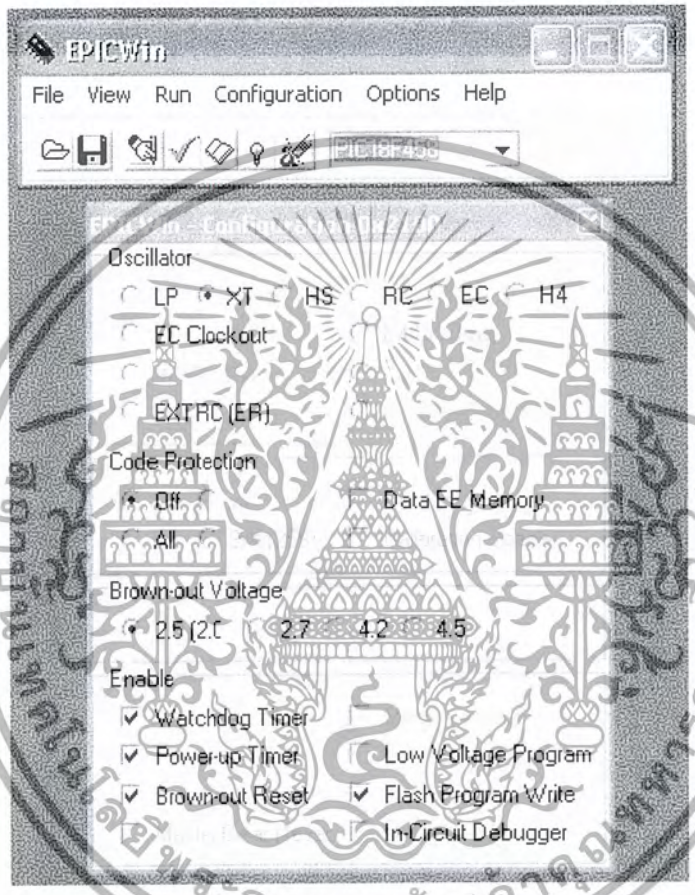
CON 1	พอร์ตเชื่อมต่อกับพอร์ต C ของไมโครคอนโทรลเลอร์
J1	พอร์ตเชื่อมต่อกับพอร์ต C ของไมโครคอนโทรลเลอร์
BACKUP	จั๊มเปอร์เซตค่าให้แก่ RTC เพื่อให้รีเซตเริ่มค่าเวลาใหม่

1.4.7 โมดูลแสดงผลแอลซีดี (LCD Display Module)



รูปที่ ฉ.9 จุดเชื่อมต่อใช้งานของโมดูลแสดงผลแอลซีดี

คู่มือการใช้งาน
โปรแกรม Epic Win



ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

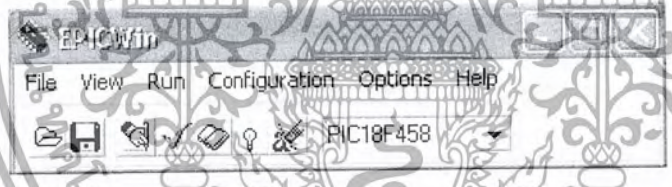
1.2 คู่มือการใช้งานโปรแกรม EPIC Win

1.2.1. ปุ่มควบคุมการทำงานหลัก

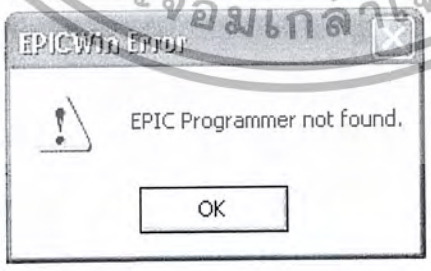
เมื่อทำการเรียกโปรแกรม EPIC for Windows ขึ้นมาทำงาน จะปรากฏหน้าต่างหลักดังรูปที่ ฉ.10 จะเห็นว่ามีปุ่มควบคุมการทำงานอยู่ 7 ปุ่มและมีช่องแสดงเบอร์ไมโครคอนโทรลเลอร์ที่ต้องการโปรแกรมอยู่ในตำแหน่งขวาสุด สำหรับรายละเอียดของปุ่มควบคุมการทำงานทั้งหมดมีดังนี้

1. ปุ่ม  OPEN ใช้สำหรับเปิดไฟล์นามสกุล .HEX ที่ต้องการโปรแกรมลงในไมโครคอนโทรลเลอร์ เมื่อทำการเปิดไฟล์แล้ว ที่ไตเติ้ลบาร์จะปรากฏชื่อไฟล์พร้อมตำแหน่งที่อยู่ ถ้าหากมีการเลือกให้แสดงค่าพารามิเตอร์ของการโปรแกรม โปรแกรมก็จะเรียกพารามิเตอร์เหล่านั้นมาแสดงด้วย โดยการเลือกนี้สามารถกระทำได้ที่เมนู Option เลือก Update Configuration

2. ปุ่ม  SAVE ใช้สำหรับบันทึกซอร์สโค้ดซึ่งเป็นไฟล์นามสกุล .HEX ลงในฮาร์ดดิสก์หรือฟลอปปี้ดิสก์ โดยในการบันทึกนี้จะเก็บทั้งซอร์สโค้ด ID และค่าพารามิเตอร์ต่างๆ ทั้งหมด



รูปที่ ฉ.10 หน้าต่างหลักของโปรแกรม EPIC for Windows



รูปที่ ฉ.11 ไดอะล็อกบล็อกแจ้งว่าไม่สามารถติดต่อกับเครื่องโปรแกรมได้

3. ปุ่ม  PORGRAM ใช้สำหรับโปรแกรมข้อมูลทั้งหมดของไฟล์นามสกุล .HEX ลงในหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลที่เป็นหน่วยความจำแบบ อีอีพรอมของไมโครคอนโทรลเลอร์ที่เลือกไว้ ในการโปรแกรมนี้ผู้ใช้สามารถเลือกได้ว่าต้องการให้ตรวจสอบข้อมูลว่างภายในไมโครคอนโทรลเลอร์ก่อนหรือไม่ รวมถึงเลือกให้โปรแกรมแสดงข้อความการโปรแกรมเสร็จสิ้นหรือไม่ก็ได้ โดยทั้งสองทางเลือกหลังนี้สามารถเข้าไปตั้งได้ที่เมนู Option เลือก Skip Blank Check และ Disable Completion Message ตามลำดับ

4. ปุ่ม  VERIFY ใช้สำหรับการตรวจสอบโปรแกรม โดยทำการเปรียบเทียบข้อมูลในบัฟเฟอร์กับข้อมูลที่อ่านได้จากไมโครคอนโทรลเลอร์หลังจากที่โปรแกรมข้อมูลแล้วหากถูกต้องแสดงว่าการโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์เสร็จสมบูรณ์

ถ้าหากการตรวจสอบไม่ผ่านจะปรากฏไดอะล็อกบ็อกซ์ แจ้งข้อความผิดพลาดให้ทราบ อย่างไรก็ตาม หากมีการป้องกันข้อมูล (Code protect) จะไม่สามารถทำการตรวจสอบด้วยวิธีการนี้ได้

5. ปุ่ม  READ ใช้สำหรับอ่านข้อมูลจากหน่วยความจำของไมโครคอนโทรลเลอร์ รวมทั้งพารามิเตอร์อื่นๆ ด้วยในกรณีที่มีการเลือก Update Configuration ในเมนู Option ไว้

6. ปุ่ม  BLANK CHECK ใช้ตรวจสอบข้อมูลภายในไมโครคอนโทรลเลอร์ โดยตรวจสอบว่าไมโครคอนโทรลเลอร์นี้มีข้อมูลอยู่หรือไม่เท่านั้น ไม่มีการตรวจสอว่ามีที่ว่างเหลือเท่าใดหรือค่าพารามิเตอร์ต่างๆ เป็นอย่างไร

7. ปุ่ม  ERASE ใช้สำหรับลบข้อมูลภายในหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ ที่เลือกฟังก์ชั่นนี้ใช้ได้กับไมโครคอนโทรลเลอร์ที่มีหน่วยความจำแบบแฟลชหรืออีอีพรอมเท่านั้น อาทิ เบอร์ PIC18F458

8. ช่องสำหรับเลือกเบอร์ไมโครคอนโทรลเลอร์ 

เมื่อต้องการเลือกเบอร์ของไมโครคอนโทรลเลอร์ทำได้โดย ใช้เมาส์คลิกที่ลูกศร ก็จะปรากฏเบอร์ของไมโครคอนโทรลเลอร์ให้เลือกมากมาย จากนั้นทำการกดลูกศรขึ้นลงที่สกอลล์บาร์ จนพบเบอร์ที่ต้องการจึงทำการคลิกที่เบอร์นั้น ในช่องแสดงเบอร์ก็จะปรากฏเบอร์ของไมโครคอนโทรลเลอร์ที่ต้องการ

1.2.2 เมนูการใช้งานของโปรแกรม

1. เมนู File

ในเมนูนี้ยังมีเมนูย่อยที่ต้องใช้งานอีก 5 เมนูคือ

Open : ใช้สำหรับเปิดไฟล์นามสกุล .HEX ที่ต้องการโปรแกรม ทำงานเหมือนกับปุ่ม OPEN

Save : ใช้สำหรับบันทึกแฟ้มข้อมูลในชื่อเดิม มีการทำงานเหมือนกับปุ่ม SAVE

Save As : ใช้สำหรับบันทึกแฟ้มข้อมูลในชื่ออื่น

EPIC Port : ใช้ในการเลือกพอร์ตขนานที่ติดต่อกับโปรแกรม EPIC for Windows ซึ่งสามารถเลือกได้ 3 ตำแหน่ง คือ LPT1, LPT2 และ LPT3 ปกติโปรแกรมจะทำการตรวจสอบเองเมื่อเปิดการใช้งานครั้งแรก หากตรวจสอบไม่พบเครื่องโปรแกรมจะแสดงข้อความเตือน ดังรูปที่ ฉ. 12

2. เมนู View

ใช้สำหรับดูค่าพารามิเตอร์ทั้งหมดของไมโครคอนโทรลเลอร์ PIC ซึ่งแบ่งออกได้อีก 5 เมนูย่อยโดยเมื่อเลือกแต่ละเมนูแล้ว จะปรากฏหน้าต่างของพารามิเตอร์นั้นๆ ขึ้นมา โดยหน้าต่างของพารามิเตอร์พารามิเตอร์ที่สามารถแสดงได้มี 5 ตัวดังนี้

Configuration แสดงรายละเอียดทางด้านฮาร์ดแวร์ที่ต้องทำการเลือกของไมโครคอนโทรเลอร์ PIC ซึ่งแตกต่างกันไปในแต่ละเบอร์ โดยผู้ใช้งานสามารถที่จะเลือกหรือกำหนดค่าพารามิเตอร์ต่างๆ ได้ที่เมนูนี้เลย หรือจะทำการเลือกที่เมนู Configuration ก็ได้ สำหรับรายละเอียดของการกำหนดค่าพารามิเตอร์จะกล่าวถึงต่อไปในหัวข้อ การกำหนดค่า Configuration

Code ใช้แสดงรายละเอียดของซอร์สโค้ดที่เป็นเลขฐานสิบหกหรือแอสกี้ก็ได้

Data ใช้แสดงรายละเอียดของหน่วยความจำข้อมูลอีอีพรอมภายในไมโครคอนโทรลเลอร์ ถ้าเบอร์ใดไม่มีหน่วยความจำอีอีพรอม หน้าต่างนี้จะว่าง ไม่มีข้อมูลแสดง

ID ใช้แสดงค่า Identify เฉพาะของไมโครคอนโทรลเลอร์ตัวนั้นๆ ซึ่งผู้ใช้งานสามารถกำหนดได้ในขั้นตอนการโปรแกรม

Serial Number ใช้แสดงค่า Serial Number เฉพาะของไมโครคอนโทรลเลอร์ตัวนั้น ซึ่งผู้ใช้งานสามารถกำหนดได้ในขั้นตอนการโปรแกรม

Count ใช้แสดงจำนวนของไมโครคอนโทรลเลอร์ที่นำมาโปรแกรมว่าทำการโปรแกรมทั้งหมดกี่ตัว มีจำนวนของไมโครคอนโทรลเลอร์ที่โปรแกรมไม่สำเร็จเท่าใด นั่นคือ ใช้สำหรับกรณีที่ทำการโปรแกรมไฟล์เดียวกันลงในไมโครคอนโทรเลอร์จำนวนมาก เพื่อแจ้งให้ทราบว่าได้ทำการ

โปรแกรมไปแล้วเท่าใด มีจำนวนที่ไม่ผ่านเท่าใดนอกจากนั้นออปชั่นของการการแสดงหน้าต่างของโปรแกรมอีก 2 ตัว

Close all Windows ใช้สำหรับปิดหน้าต่างพารามิเตอร์ทั้งหมด

Stay On Top ใช้สำหรับกำหนดให้หน้าต่างหลักของโปรแกรม EPIC for Windows อยู่ในตำแหน่งบนสุดทุกครั้ง ไม่ว่าจะมีการเปิดโปรแกรมประยุกต์อื่นใดในวินโดวส์ก็ตาม หน้าต่างของโปรแกรม EPIC for Windows จะอยู่บนสุดเสมอ ทั้งนี้เพื่อความสะดวกในการพัฒนา เมื่อผู้ใช้งานทำการเขียนโปรแกรมด้วย MPLAB เมื่อทำการเขียนและแอสแซมเบลอร์เรียบร้อย ต้องการโปรแกรมก็จะพบหน้าต่างของ EPIC for Windows อยู่บนสุดก่อนเสมอ จึงสามารถใช้งานได้ทันทีโดยไม่ต้องค้นหาให้เสียเวลา

1.2.3 การกำหนดค่า Configuration

ในการโปรแกรมไมโครคอนโทรลเลอร์ PIC สิ่งหนึ่งที่ต้องกระทำเสมอก่อนการเขียนโปรแกรมคือ การกำหนดค่า Configuration สำหรับโปรแกรม EPIC for Windows นี้ผู้งานใช้งานสามารถกำหนดค่า Configuration ได้อย่างครบถ้วน โดยเลือกที่เมนู View/Configuration จะปรากฏหน้าต่างตามรูปที่ ฉ.13 ไมโครคอนโทรลเลอร์เบอร์ใดที่ไม่ความสามารถใด พารามิเตอร์ที่ใช้แสดงความสามารถนั้นๆ ก็ไม่ปรากฏขึ้นมาให้สามารถเลือกได้ สำหรับรายละเอียดทั้งหมดของ Configuration สามารถอธิบายได้ดังนี้

1. Oscillator ใช้สำหรับเลือกชนิดของวงจรกำเนิดสัญญาณนาฬิกาหรือวงจรออสซิลเลเตอร์ของไมโครคอนโทรลเลอร์ PIC มีให้เลือก 8 แบบ คือ

LP : คริสตอลแบบกำลังงานต่ำ

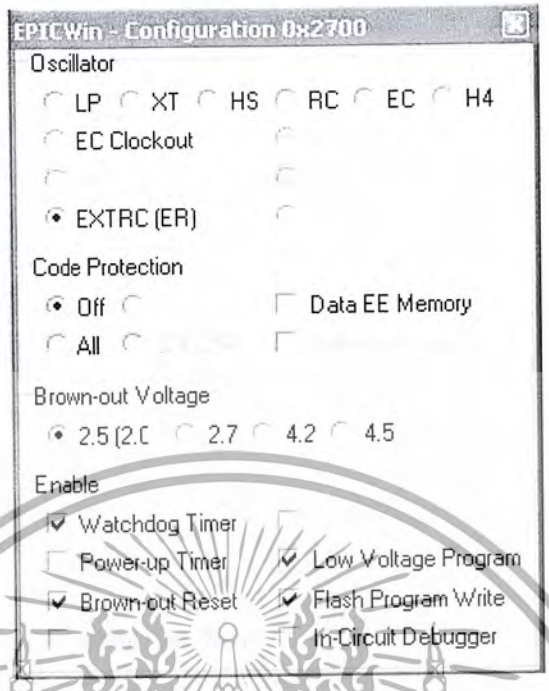
XT : คริสตอลทั่วไป ความถี่ไม่เกิน 4 MHz

HS : คริสตอลความถี่สูงตั้งแต่ 4 MHz ขึ้นไป

RC : ใช้วงจร RC ต่อภายนอก

INTRC (IN) : ใช้วงจร RC ภายในสำหรับไมโครคอนโทรลเลอร์อนุกรม PIC 12Cxxx

EXTRC : ใช้วงจร RC ภายนอกสำหรับไมโครคอนโทรลเลอร์อนุกรม PIC12Cxxx



รูปที่ ฉ.12 หน้าต่าง View /Configuration สำหรับกำหนดค่าพารามิเตอร์ต่างๆ ของไมโครคอนโทรลเลอร์ PIC ที่ทำการโปรแกรม

INTRC Clock Out : ใช้วงจร RC ภายในสำหรับไมโครคอนโทรลเลอร์อนุกรม PIC 12Cxxx และ PIC16C50x โดยที่ขา OSC2 จะมีความถี่ 1 MHz ส่งออกมาด้วย แต่อย่างไรก็ตาม ผู้ใช้งานสามารถเลือกได้ว่า ต้องการให้ขา OSC2 เป็นขาสำหรับส่งสัญญาณนาฬิกาหรือเป็นขาพอร์ตก็ได้

EXTRC Clock Out : ใช้วงจร RC ภายนอกสำหรับไมโครคอนโทรเลอร์อนุกรม PIC12Cxxx และ PIC16C50x โดยที่ขา OSC2 จะมีความถี่ 1 MHz ส่งออกมาด้วย แต่อย่างไรก็ตาม ผู้ใช้งานสามารถเลือกได้ว่า ต้องการให้ขา OSC2 เป็นขาสำหรับส่งสัญญาณนาฬิกาหรือเป็นขาพอร์ตก็ได้

2. Code Protection ใช้เลือกโหมดของการป้องกันการอ่านข้อมูลออกจากไมโครคอนโทรลเลอร์มีให้เลือก 6 โหมด คือ ไม่ป้องกัน (Off), ป้องกันทั้งหมด (All), ป้องกันครึ่งหนึ่งด้านบน (Upper 1/2), ป้องกัน 3/4 ด้านบน (Upper 3/4), ป้องกันเฉพาะหน่วยความจำข้อมูลอีอีพรอม (Data EE Memory) และป้องกันเฉพาะพื้นที่ปรับแต่งของส่วนประมวลผลสัญญาณแอนะลอก (Calibration Space) สำหรับไมโครคอนโทรลเลอร์เบอร์ PIC14000

3. Brow Oout Voltage เป็นการกำหนดระดับแรงดันไฟเลี้ยงที่ทำให้ไมโครคอนโทรลเลอร์ PIC เกิดการรีเซตโดยอัตโนมัติ เลือกได้ 4 ระดับคือ 2.5, 2.7, 4.2 และ 4.5V ทั้งนี้ต้องขึ้นอยู่กับไมโครคอนโทรลเลอร์ในแต่ละเบอร์ บางเบอร์ไม่มีความสามารถนี้ บางเบอร์ไม่มีความสามารถนี้ บางเบอร์มีความสามารถนี้แต่ไม่สามารถเลือกค่าแรงดันได้

4. Enableใช้เลือกให้ความสามารถพิเศษอื่นๆ ของไมโครคอนโทรลเลอร์ PIC ทำงานมี 8 อย่าง คือ

Watchdog Timer : วอตช์ด็อกไทเมอร์

Power Timer : เพาเวอร์อัปไทเมอร์

Brown-out Reset : เอ็นนาเบิลให้วงจรบราวเอาต์ซึ่งหน้าที่รีเซตไมโครคอนโทรลเลอร์ PIC เมื่อแรงดันไฟเลี้ยงมีค่าตามที่กำหนดไว้ทำงาน หมายความว่า หากต้องการให้วงจรบราวเอาต์ทำงานต้องเอ็นนาเบิลความสามารถนี้ แล้วเลือกระดับแรงดันที่ต้องการควบคู่กัน แต่ในบางเบอร์มีความสามารถนี้แต่ไม่สามารถเลือกระดับแรงดันได้

Master Clear Reset : ใช้เอ็นนาเบิลการรีเซตจากภายนอกที่ขา MCLR หากไม่มีเอ็นนาเบิลความสามารถนี้ ขา MCLR จะสามารถใช้งานเป็นขาพอร์ตอินพุตได้ ความสามารถนี้จะมีในไมโครคอนโทรลเลอร์รุ่นเล็กอนุกรม PIC12C5xx

Memory Parity Error : ใช้อีนาเบิลการตรวจสอบความผิดพลาดทางพาริตี้ในหน่วยความจำ ความสามารถนี้มีในไมโครคอนโทรลเลอร์ PIC อนุกรม PIC16C64x

Low Voltage Program : ใช้เอ็นนาเบิลการโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ PIC ด้วยแรงดันต่ำ +5 V แทนที่จะใช้ +12 V ตามปกติพบในไมโครคอนโทรลเลอร์อนุกรมใหม่ PIC16F87x โดยเมื่อเอ็นนาเบิลความสามารถนี้แล้วที่ขา RB3 จะเป็นอินพุตรับแรงดัน +5 V สำหรับการโปรแกรมจะไม่ใช้ขา MCLR / Vpp

Flash Program Write : หากเลือกให้ความสามารถนี้ทำงาน จะสามรถเขียนข้อมูลลงในหน่วยความจำโปรแกรมแบบแฟลชภายในผ่านการควบคุมโดยรีจิสเตอร์ EECON ได้ หากไม่เลือกการเขียนข้อมูลลงในหน่วยความจำโปรแกรมแบบแฟลชต้องกกระทำผ่านขา RB6 และ RB7 อันเป็นกระบวนการตามปกติ

In-Circuit Debugger : เป็นความสามารถที่ยอมให้เข้าไปแก้ไขหน่วยความจำโปรแกรมในตำแหน่ง 0x0100 (100H) เป็นต้นไป โดยผ่านทางขา RB6 และ RB7 ทำให้ขา RB6 และ RB7 ไม่สามารถใช้งานเป็นขาพอร์ตได้

3. เมนู Run

เป็นเมนูสำหรับจัดการเกี่ยวกับการอ่าน, อ่าน, เขียน และตรวจสอบข้อมูลในหน่วยความจำของไมโครคอนโทรลเลอร์ PIC มีเมนูย่อย 5 เมนูคือ

Program (Ctrl+P) ใช้สำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ มีการทำงานเหมือนกับปุ่ม PROGRAM

Verify (Crtl+V) ใช้ตรวจสอบการเขียนข้อมูลลงในไมโครคอนโทรลเลอร์ว่าถูกต้องหรือไม่ มีการทำงนเหมือนกับปุ่ม VERIFY

Read (Ctrl+R) ใช้สำหรับอ่านข้อมูลจากไมโครคอนโทรลเลอร์ PIC มีการทำงานเหมือนกับปุ่ม READ

Blank check (Ctrl+B) ใช้สำหรับอ่านข้อมูลว่างภายในหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ PIC มีการทำงานเหมือนกับปุ่ม BLANK CHECK

Erase (Ctrl+E) ใช้ลบข้อมูลภายในหน่วยความจำของไมโครคอนโทรลเลอร์ PIC มีการทำงานเหมือนกับปุ่ม ERASE

4. เมนู Configuration

ใช้สำหรับกำหนดค่าพารามิเตอร์ของไมโครคอนโทรลเลอร์ที่ต้องการโปรแกรมซึ่งการกำหนดค่าต่างๆ เลือกได้เหมือนกับในเมนู View แต่กต่างกันตรงที่วิธีการเข้าถึงโดยในเมนู Configuration นี้จะใช้การเข้าถึงแบบเมนูเป็นชั้นๆ แต่ที่เมนู View จะเป็นในลักษณะหน้าต่างเมนู Option โดยในเมนูนี้แบ่งออกเป็น 2 ส่วนคือ ส่วนกำหนดรูปแบบการตรวจสอบ (verify) ในลักษณะต่างๆ หลังการโปรกรม ดังในรูปที่ ฉ.14 โดยมีรายละเอียดต่อไปนี้

Program/Verify Code เลือกเพื่อกำหนดให้ตรวจสอบข้อมูลที่โปรแกรมลงในไมโครคอนโทรลเลอร์ล่าสุดกับข้อมูลในบัฟเฟอร์

Program/Verify Configuration เลือกเพื่อกำหนดให้ตรวจสอบการกำหนดค่าพารามิเตอร์ของข้อมูลที่โปรแกรมลงในไมโคคอนโทรลเลอร์ล่าสุด

Program/Verify Data เลือกเพื่อกำหนดให้ตรวจสอบข้อมูลในหน่วยความจำข้อมูลในไมโครคอนโทรลเลอร์หลังจากได้รับการโปรแกรมล่าสุด

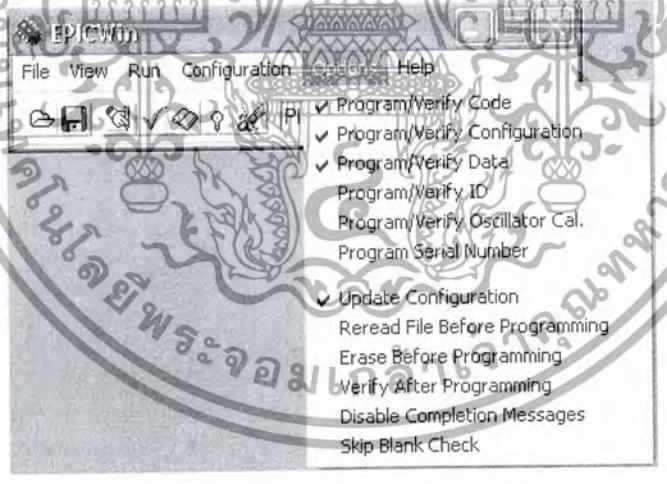
Program/Verify ID เลือกเพื่อกำหนดให้ตรวจสอบค่า ID ของไมโครคอนโทรลเลอร์หลังจากได้รับการโปรแกรมล่าสุด

Program/Verify Oscillator Cal เลือกเพื่อกำหนดให้ตรวจสอบข้อมูลของการปรับแต่งความถี่ออสซิลเลเตอร์ของไมโครคอนโทรลเลอร์หลังจากได้รับการโปรแกรม

Program/Verify Serial Number เลือกเพื่อกำหนดให้ตรวจสอบค่าเลขหมายประจำ (Serial Number) ของไมโครคอนโทรลเลอร์หลังจากได้รับการโปรแกรมล่าสุดโดยปกติจะเลือกทั้ง 5 รูปแบบแรก ดังตัวอย่างในรูปที่ 4 ส่วนในรูปแบบสุดท้ายสำหรับผู้ใช้งานทั่วไป มักจะไม่มีการใช้งาน ส่วนที่สองเป็น การเลือกใช้ความสามารถพิเศษอื่นของโปรแกรม EPIC for Windows เพื่ออำนวยความสะดวกให้แก่ผู้ใช้งาน ประกอบด้วย

Update Configuration เป็นการเลือกให้ปรับปรุงค่า Configuration ของไมโครคอนโทรลเลอร์ตามที่มีการกำหนดในไฟล์นามสกุล .HEX ที่ใช้ในการโปรแกรมทุกครั้งที่มีการเปลี่ยนหรือเปิดไฟล์ใหม่

Verify File Checksum เป็นการเลือกให้มีการตรวจสอบค่า Check Sum ของไฟล์ที่ใช้โปรแกรมกับข้อมูลภายในไมโครคอนโทรลเลอร์ที่โปรแกรมแล้ว



รูปที่ ฉ.13 รายละเอียดของหน้าต่าง Option ซึ่งในการกำหนดรูปแบบการตรวจสอบโปรแกรม

รวมถึงความสมารถพิเศษอื่นๆ ของโปรแกรม EPIC for Windows ด้วย

Read File Before Programming เป็นการเลือกให้มีการอ่านไฟล์ก่อนการโปรแกรม

Disable Completion Message เป็นการเลือกไม่แสดงข้อความแจ้งการเสร็จสิ้นของแต่ละกระบวนการ สำหรับผู้ใช้งานใหม่ที่ยังไม่เชี่ยวชาญ ไม่แนะนำให้เลือกฟังก์ชั่นนี้ เพราะจะไม่มีทราบเลยว่า กระบวนการต่างๆ ที่สั่งให้โปรแกรมทำนั้นเสร็จสมบูรณ์หรือไม่

Skip Blank Check เป็นการเลือกให้ข้ามขั้นตอนการตรวจสอบข้อมูลว่างก่อนการโปรแกรม

5. เมนู Help

เป็นเมนูที่บรรจุคำแนะนำในการใช้งานโปรแกรม

คู่มือการใช้งาน
โปรแกรม MPLAB

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

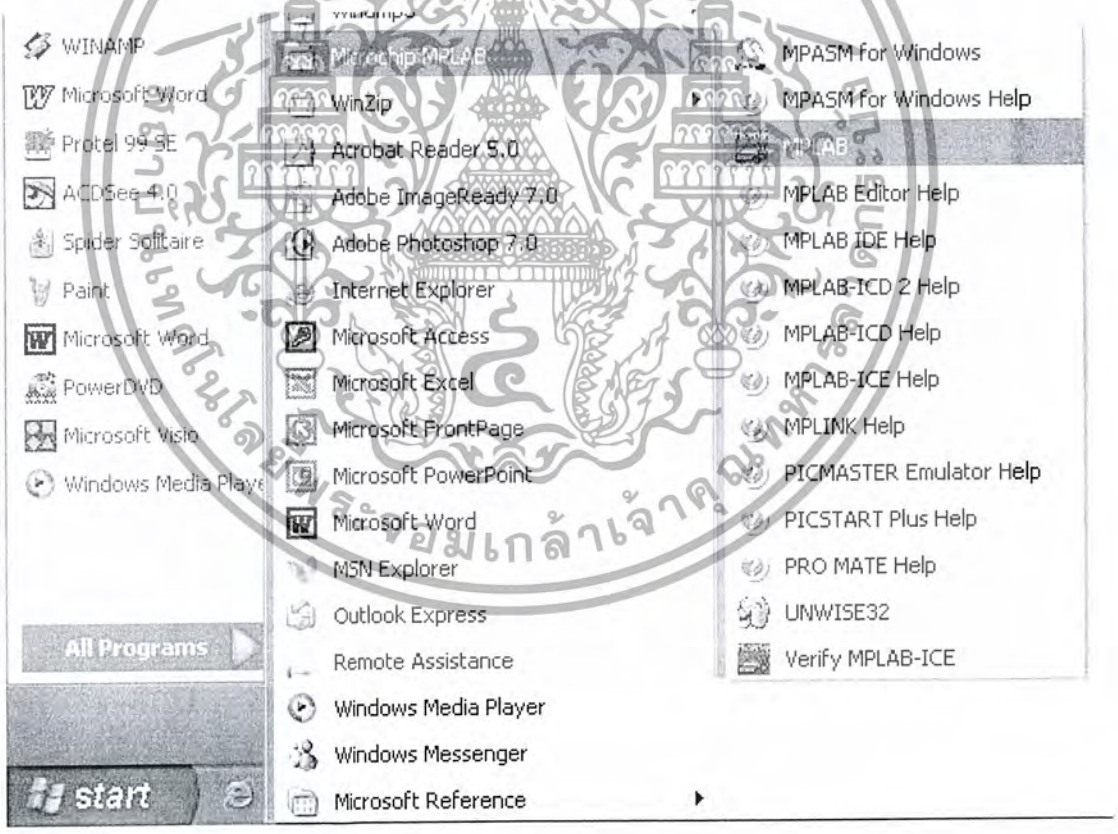
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

1.3 การใช้งานโปรแกรม MPLAB

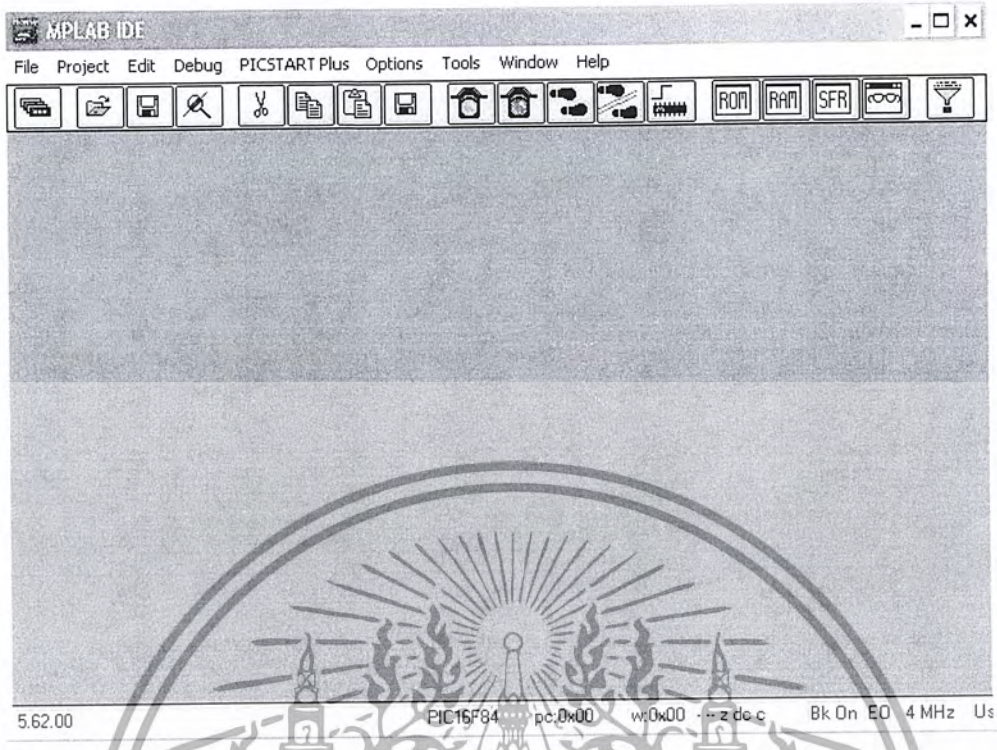
ในการพัฒนาโปรแกรม สิ่งจำเป็นและขาดไม่ได้ก็คือ เครื่องมือในการพัฒนาโปรแกรมต่างๆ ไม่ว่าจะเป็นเครื่องมือที่เราใช้ในการเขียนโปรแกรม (Text File) และ ตัวแปลภาษา (Assembler) ต่างๆ ซึ่งในที่นี้จะขอแนะนำโปรแกรมที่สามารถใช้ได้ทั้ง สำหรับเขียนโปรแกรม และแอสเซมเบลอร์ ได้ในตัว อีกทั้งยังมีฟังก์ชั่นการทำงานที่เอื้ออำนวยความสะดวกในการพัฒนาโปรแกรมอื่นๆ อีกมากมาย แต่ในที่นี้จะยกตัวอย่าง เพียงเบื้องต้นเท่านั้น โดยในการใช้โปรแกรม MPLAB สำหรับการสร้างโปรเจ็คงาน จะมีวิธีและ ขั้นตอนต่างๆ ดังต่อไปนี้

1. เมื่อทำการติดตั้งโปรแกรม MPLAB เรียบร้อยแล้วซึ่งโปรแกรมสามารถดาวน์โหลดได้ฟรี ในเว็บไซต์ของทางไมโครชิฟ (www.microchip.com) จากนั้นทำการเปิดโปรแกรม ดังรูปที่ ฉ.14



รูปที่ ฉ.14 แสดงการเข้าสู่โปรแกรม MPLAB

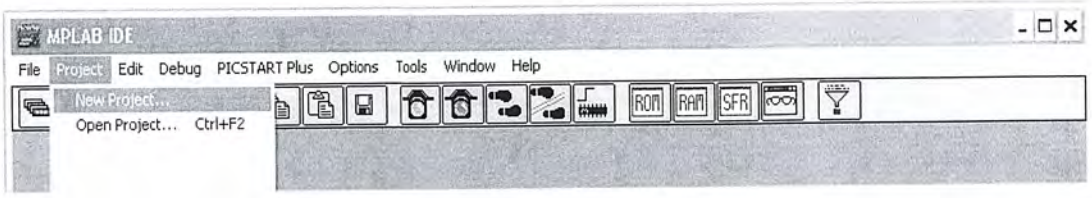
เมื่อเปิดโปรแกรมดังกล่าวแล้วจะปรากฏหน้าต่าง ของโปรแกรม MPLAB ดังรูปที่ ฉ.15



รูปที่ ฉ.15 แสดงหน้าต่างของโปรแกรม MPLAB

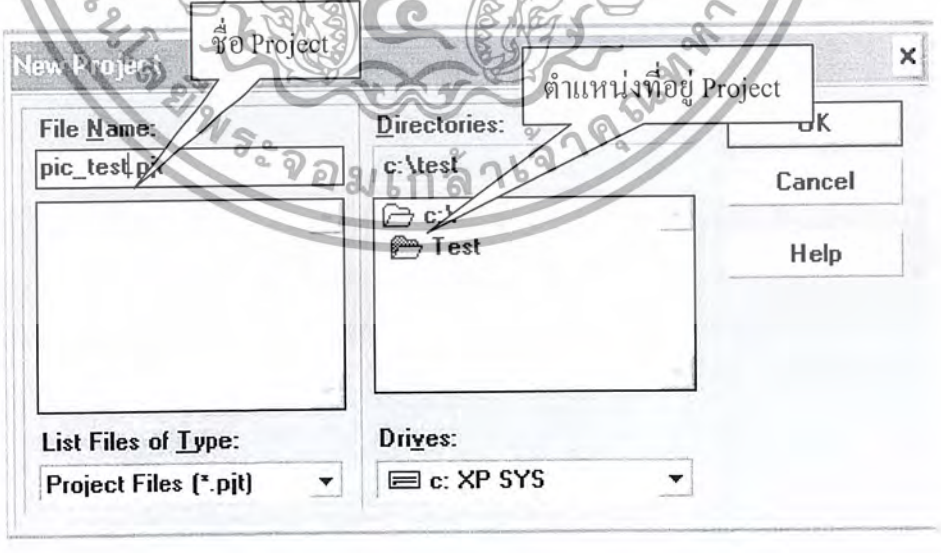
จากรูปที่ ฉ.15 จะประกอบไปด้วยเมนูคำสั่งต่างๆ ที่ใช้ในการพัฒนาโปรแกรม โดยจะกล่าวถึงเฉพาะเมนูคำสั่งที่จำเป็นต้องใช้งานเท่านั้น

ทำการสร้างโปรเจ็ค โดยเปิดที่เมนู Project New Project ดังรูปที่ ฉ.16



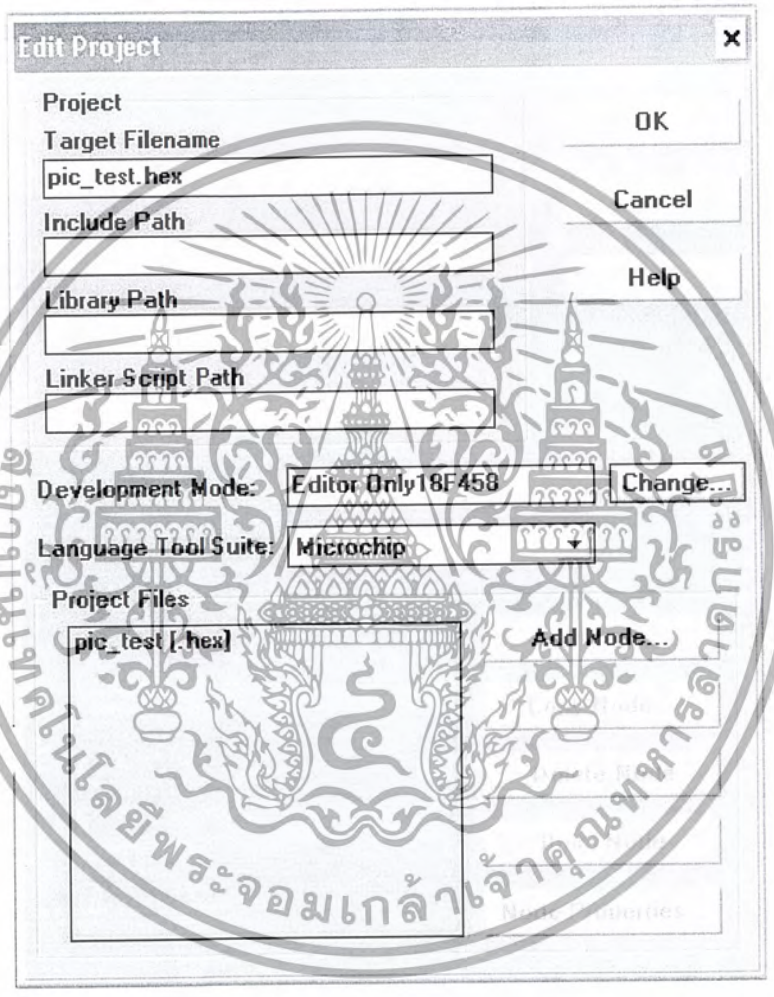
รูปที่ ฉ.16 แสดงการเปิดเมนูเพื่อสร้างโปรเจ็คใหม่

จะปรากฏหน้าต่างดังรูปที่ ฉ.17 โดยจะมีช่องให้ใส่ชื่อ Project ให้เราพิมพ์ชื่อโปรเจ็คที่เราต้องการลงในช่อง File Name ซึ่งจะมีนามสกุลเป็น .pjt เราสามรถเลือก Directories สำหรับเก็บไฟล์โปรเจ็คได้ในช่อง Directories ซึ่งแนะนำการสร้างโฟลเดอร์สำหรับเก็บไฟล์โปรเจ็คนี้โดยเฉพาะ เพื่อไม่ให้สับสนกับไฟล์ต่างๆ เนื่องจาก เมื่อเราสร้างโปรแกรม หรือทำการคอมไพล์โปรแกรมจะเกิดไฟล์ต่างๆ ขึ้นมา หากเราสร้างโปรเจ็คใน Directories ที่มีไฟล์อื่นๆ อยู่ก่อนแล้วอาจเกิดการสับสนได้ ดังนั้นพื้นที่ในการเก็บโปรแกรมครั้งแรกนั้น ควรเป็นพื้นที่ว่างเพื่อใช้งานใน Project นี้โดยเฉพาะดังในตัวอย่าง จะสร้าง Project ชื่อ Pic_Test.Pjt อยู่ใน Directories C:\ Test (เป็นไดเร็ก-ทรอรี่ที่สร้างเพื่อเก็บไฟล์โปรเจ็คโดยเฉพาะ) เมื่อเรียบร้อยแล้วให้คลิกที่ปุ่ม OK เพื่อเข้าสู่ขั้นตอนต่อไป



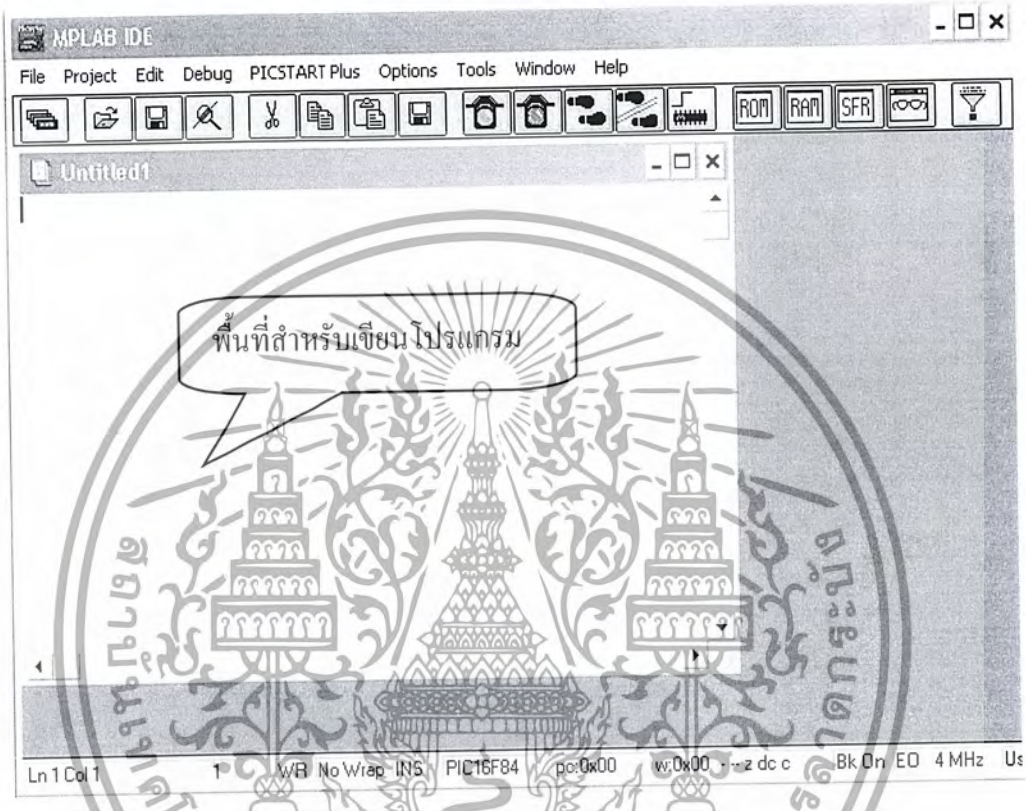
รูปที่ ฉ.17 การสร้างโปรเจ็คใหม่

เมื่อตั้งชื่อโปรเจ็คเรียบร้อยแล้ว จะเกิดหน้าต่างโปรแกรม Edit Project ดังแสดงในรูปที่ ฉ.18 เป็นส่วนของการกำหนดค่าต่างๆ ในการตั้งค่าในส่วนนี้สามารถกำหนดในภายหลังจากการเขียนโปรแกรมเสร็จแล้วก็ได้ โดยแนะนำให้ปิดหน้าต่างนี้ไปก่อน โดยการคลิกที่ปุ่ม Cancel เนื่องจากเรายังไม่มีไฟล์ที่เป็น ASM File ดังนั้นจึงควรข้ามขั้นตอนนี้ไปก่อน



รูปที่ ฉ.18 หน้าต่าง Edit Project

3. เมื่อทำการปิดหน้าต่าง Edit Project ก็จะเข้าสู่โปรแกรม Dialog ของโปรแกรม MPLAB ปกติดังรูปที่ ฉ.19 ซึ่งจะเป็นพื้นที่เปล่า ๆ ดังนั้นให้เราทำการสร้าง ซอร์สโปรแกรม ภาษาแอสเซมบลี โดยเลือกที่เมน File เลือก New (Ctrl-N) จะปรากฏหน้าต่าง ดังรูปที่ ฉ.19



รูปที่ ฉ.19 แสดงการสร้างไฟล์สำหรับเขียนโปรแกรม

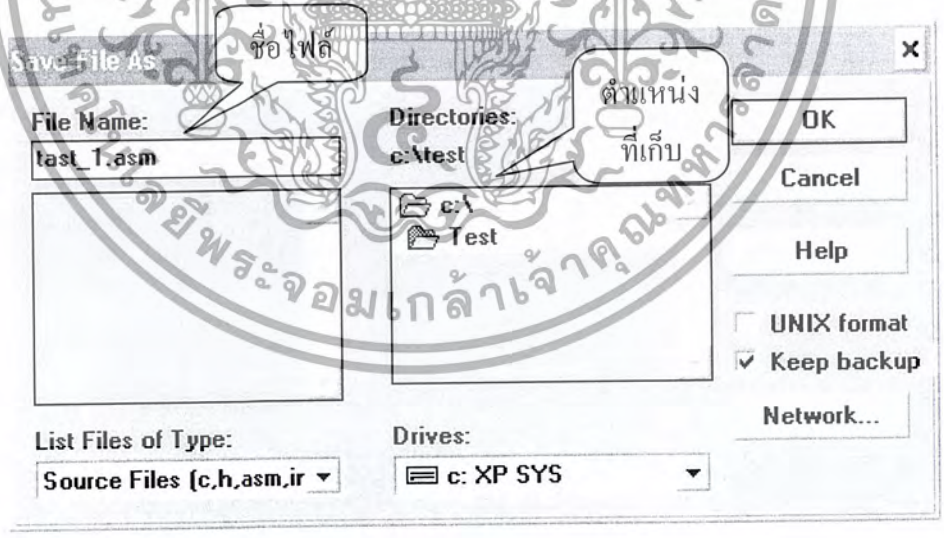
ซึ่งจะเป็นพื้นที่ว่างๆ สำหรับให้เราพิมพ์ Source Code โปรแกรม โดยในที่นี้จะยกตัวอย่างโปรแกรมง่ายๆ เพื่อทดสอบการใช้งานโปรแกรม MPLAB ให้พิมพ์โปรแกรมตามตัวอย่างโปรแกรมด้านล่าง

```
   #include <p18f458.inc>      ; processor specific variable definitions
#define    DIR  PORTC,5        ; control Direction
count1     EQU  0x70
count2     EQU  0x71
temp       EQU  0x72
           ORG    0x0000
           goto     init
           ORG     0x0004
           movlw   0x0f
           movwf    PORTD
           goto     $
           retfie
init       bcf      PORTC,5
           bcf      TRISC,5          ; Direction control
           movlw    0x40             ; BAUD rate 9600
           movwf    SPBRG            ;
           bsf        TXSTA,BRGH     ; HI SPEED
           clrf     TRISD            ; PortD output
           lfsr     0,TXSTA
           bsf      INDF0,5
           clrf     RCSTA
           bsf      RCSTA,SPEN       ; Serial port enabled
           bsf      RCSTA,CREN       ; continuous receive
           bcf      DIR              ; control direction to receive RX
wait1      btfss    PIR1,RCIF        ; Check RCIF  bit in PIR1 register
           goto     wait1            ; RCREG emty or RCIF = 0
           movf     RCREG,w          ; RCREG full or RCIF = 1
           movwf    temp
           bsf      DIR              ; Dir =  TX
           movf     temp,w
           movwf    TXREG
wait4      lfsr     0,TXSTA           ;
           btfss    INDF0,1           ; TSR = empty ?
           goto     wait4             ; NO  (TRMT = 0) ,TSR full
                                      ; Yes (TRMT = 1) ,TSR empty
           call     delay
           call     delay
           bcf      DIR               ; Dir = RX
           goto     wait1
```

```
        movwf   count1
del2    movlw   0xff
        movwf   count2
del1    decfsz  count2
        goto    del1
        decfsz  count1
        goto    del2
        return
        END
```

รูปที่ ฉ.20 โปรแกรมตัวอย่าง

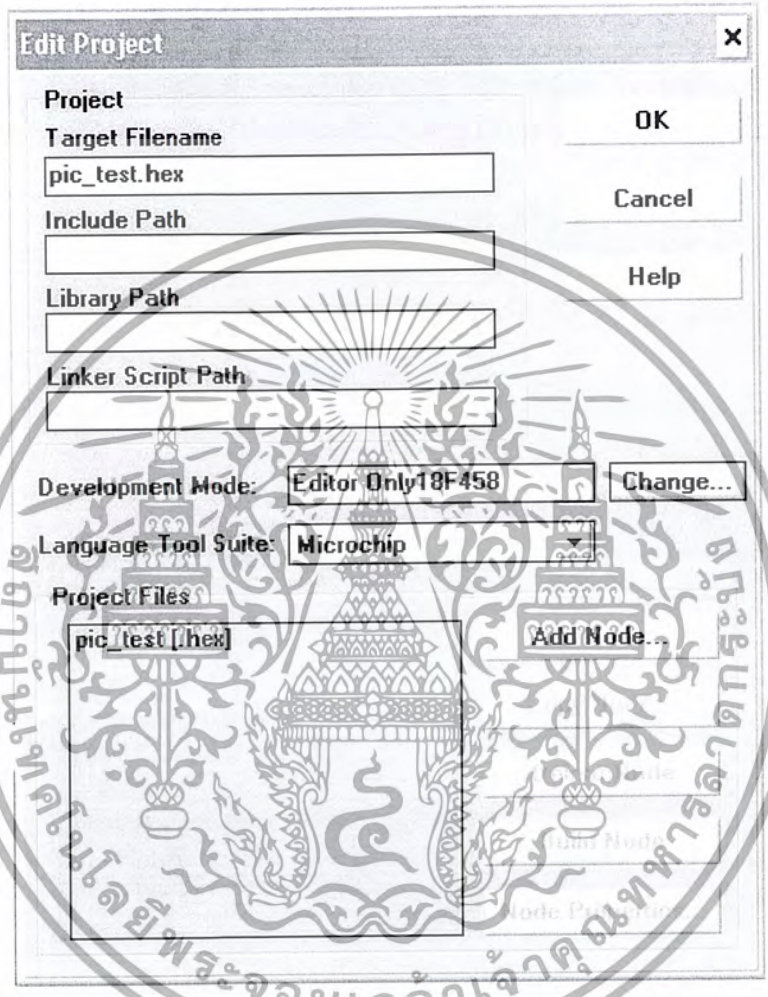
4. เมื่อเราพิมพ์ ซอร์สโค้ดดังกล่าวเสร็จเรียบร้อยแล้วให้ทำการ Save โปรแกรมโดยคลิกเลือกที่เมนู File ⟶ Save Ctrl + S จะปรากฏหน้าต่างโปรแกรม Save File As ให้ทำการตั้งชื่อไฟล์โดยจะมีนามสกุลเป็น .ASM การบันทึกนี้จะต้องบันทึกลงใน Directories เดียวกันกับไฟล์โปรเจ็ค (.PJT) ที่เราสร้างในที่นี้จะใช้ชื่อ Test_1 .asm และ เก็บในไดเร็คทรอรี่ Test ดังแสดงในรูปที่ ฉ.21



รูปที่ ฉ.21 การบันทึกไฟล์โปรแกรม

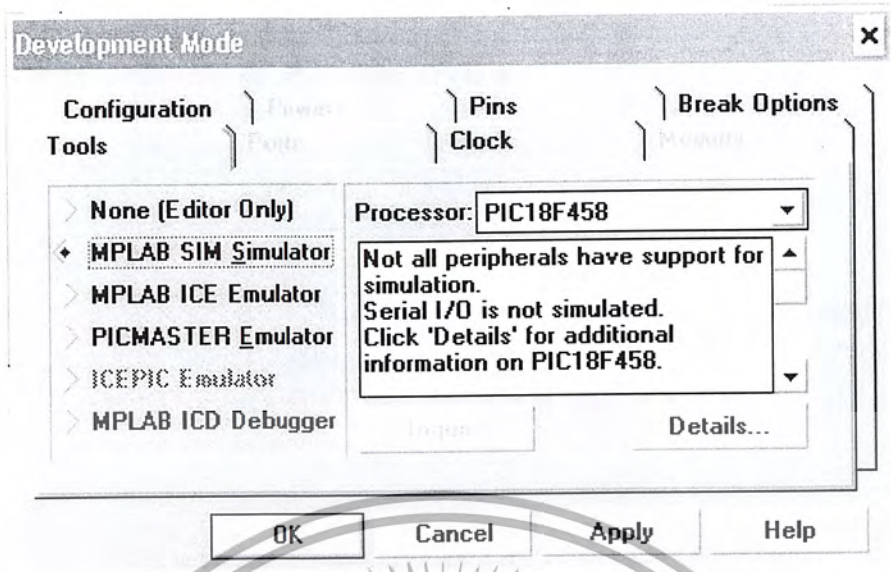
5. กำหนดค่าต่างๆ ในการคอมไพล์ ซึ่งหลังจากทำการบันทึก เราจะได้ ASM File ต่อไปจะเป็นการเปลี่ยนจากไฟล์ที่เป็น ASM File ไปเป็น HEX file ซึ่งเป็นไฟล์ที่จะต้องดาวน์โหลดลง

CPU กระบวนการนี้เรียกว่า การ Build หรือการคอมไพล์ ซึ่งเราจะต้องทำการกำหนดค่าต่างๆ โดยเข้าไปที่ Project เลือก Edit Project (Ctrl+F3) จากนั้นจะเกิดหน้าต่าง Edit Project เหมือนกับที่เราเข้ามาในตอนแรก ดังรูปที่ ฉ.22



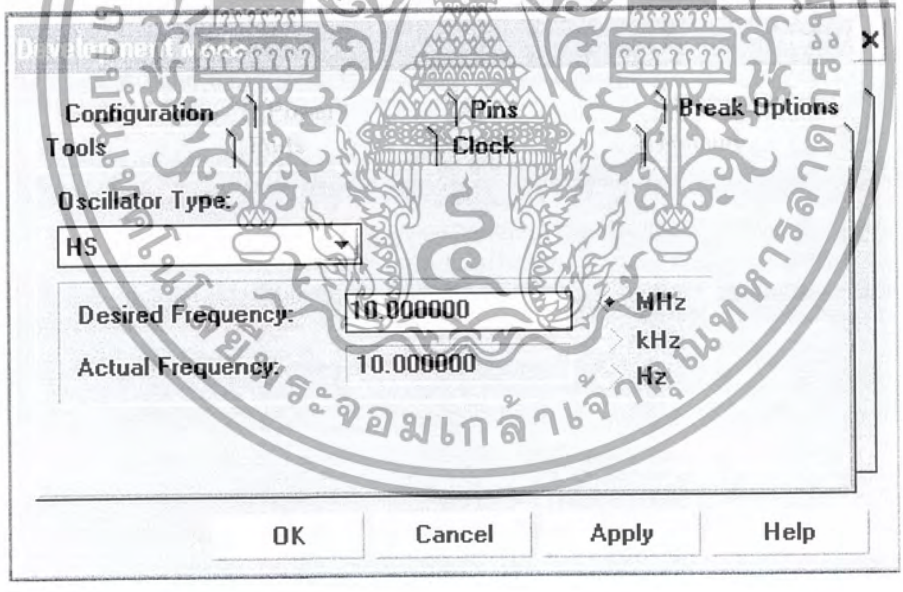
รูปที่ ฉ.22 หน้าต่างของ Edit Project

ให้เราทำการกำหนดค่าต่างๆ โดยการเลือกคลิกที่ปุ่ม Change... จะปรากฏหน้าต่าง Development Mode ในแถบของ Tools ให้คลิกเลือก MPLAB SIM Simulator ส่วนในช่อง Processor ให้เลือกเบอร์ของ CPU ที่เรานำมาใช้ในตัวอย่างนี้จะเป็นเบอร์ PIC16F458 ดังรูปที่ ฉ.23



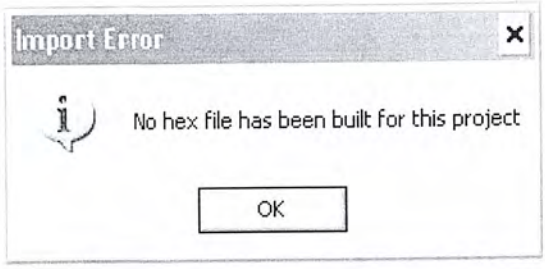
รูปที่ ฉ.23 หน้าต่าง Development Mode

ในแถบ Clock เป็นการกำหนดค่าของ Oscillator โดยในที่นี้ตะกำหนดเป็น Hs Mode ที่ความถี่ 10 MHz ดังแสดงในรูปที่ ฉ.24



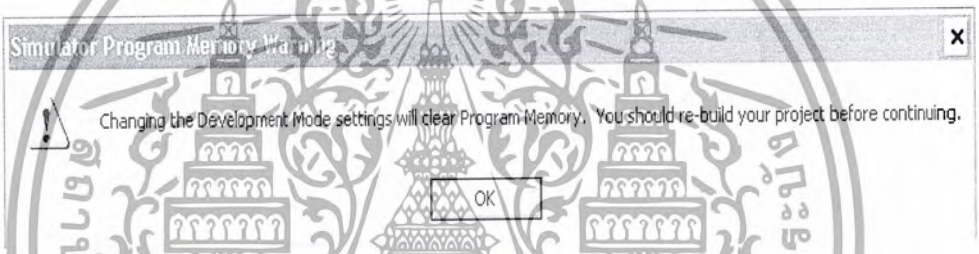
รูปที่ ฉ.24 การกำหนด Oscillator

หลังจากนั้นให้คลิกเลือกที่ปุ่ม OK จะเกิด Dialog ต่างๆ ขึ้นมาดังนี้



รูปที่ ฉ.25 หน้าต่างของ Import Error

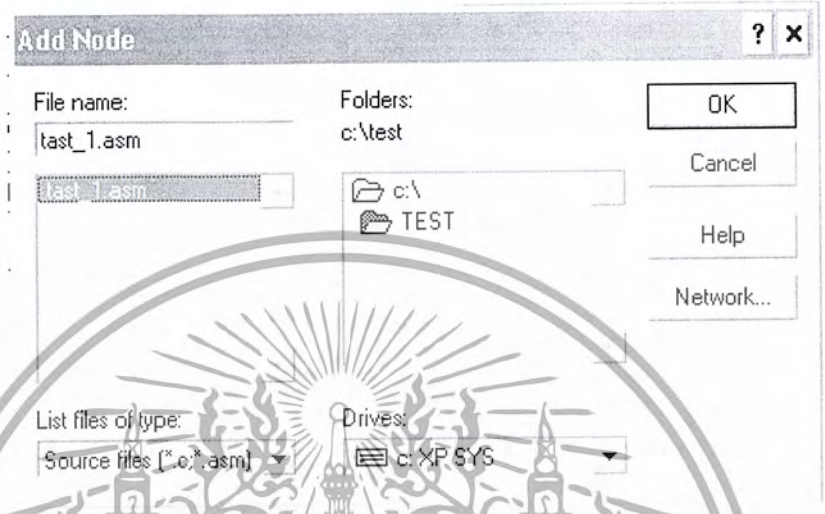
เกิดขึ้นเนื่องจากโปรเจ็คที่เราสร้างยังไม่มี HEX File ซึ่งจะข้ามไปก่อนโดยการคลิก OK มื่อเกิด Dialog Project Error นี้ให้คลิกเลือก OK ดังรูปที่ ฉ.26



รูปที่ ฉ.26 หน้าต่าง Simulator Project Memory Warning

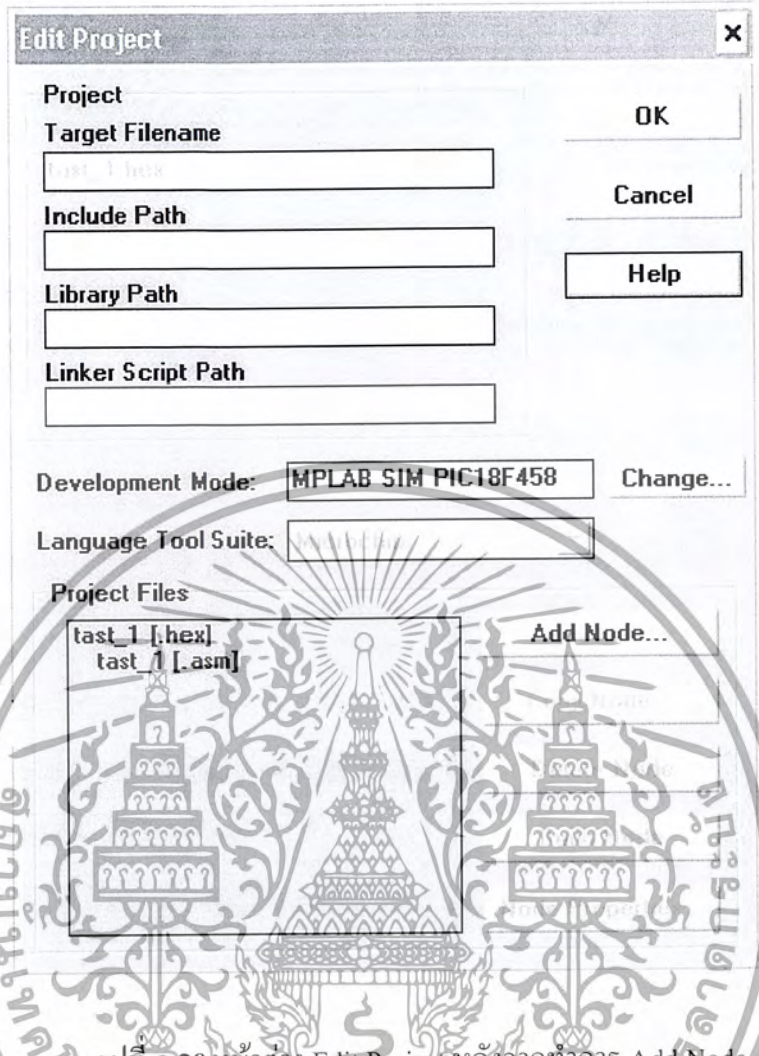
ในรูปที่ ฉ.26 นี้จะเป็นการเตือนให้เราทำการ Build โปรเจ็คใหม่เนื่องจากเรายังไม่ได้ทำการ Build โปรเจ็คดังนั้นให้คลิกเลือก OK หลังจากนั้นก็จะกลับเข้าสู่หน้าต่าง Edit Project เหมือนกับในรูปที่ ฉ.25

คลิกเลือกเพื่อกำหนดไฟล์ที่เราต้องการ Build หรือ คอมไพล์ให้เป็น HEX File โดย หลังจากคลิกที่ ปุ่ม Add Node แล้วจะเกิดหน้าต่าง Add Node เกิดขึ้นดังรูปที่ ฉ.27



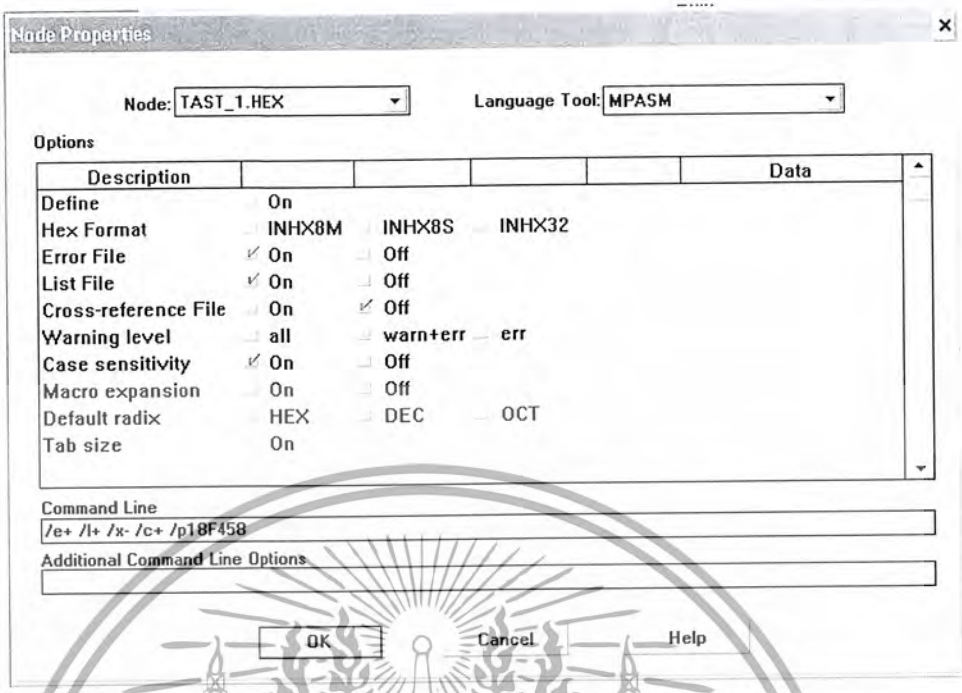
รูปที่ ฉ.27 หน้าต่าง Add Node

ทำการเลือกไฟล์ที่ต้องการสร้าง (Build) โดยไฟล์นี้จะเป็น ASM File ให้เราเลือกไฟล์ที่ต้องการ โดยจะแสดงอยู่ในช่องด้านซ้าย ในที่นี้คือ test_1 .asm เสร็จแล้วคลิกเลือก OK หลังจากนั้นในช่องของ Project Files ในหน้าต่าง Edit Project ก็จะเกิดไฟล์ test_1.asm ดังรูปที่ ฉ.28



รูปที่ ฉ.28 หน้าต่าง Edit Project หลังจากทำการ Add Node

ต่อไปให้คลิกเลือกที่ Node Properties ซึ่งหากปุ่มนี้ยังเป็นสีจางอยู่จะไม่สามารถคลิกได้ ให้เอาเมาส์ไปคลิกที่ test_1 [.hex] จากนั้นเมื่อคลิกที่ Node Properties ได้แล้วจะเกิดหน้าต่าง Node Properties เป็นการกำหนดค่า Options ต่างๆ ของ MPASM ซึ่งเป็นตัวที่ทำหน้าที่คอมไพล์โปรแกรม เราสามารถกำหนดชนิดของไฟล์ที่เราต้องการหลังจากการคอมไพล์หรือการ Build ได้ เช่นในตัวอย่างรูปที่ ฉ.29 ไฟล์ที่ได้หลังจากการคอมไพล์จะเป็น Hex File, Error File, List File เป็นต้นให้เราทำการกำหนดค่าต่างๆ ให้เหมือนดังรูปที่ ฉ.29 จากนั้นเลือก OK

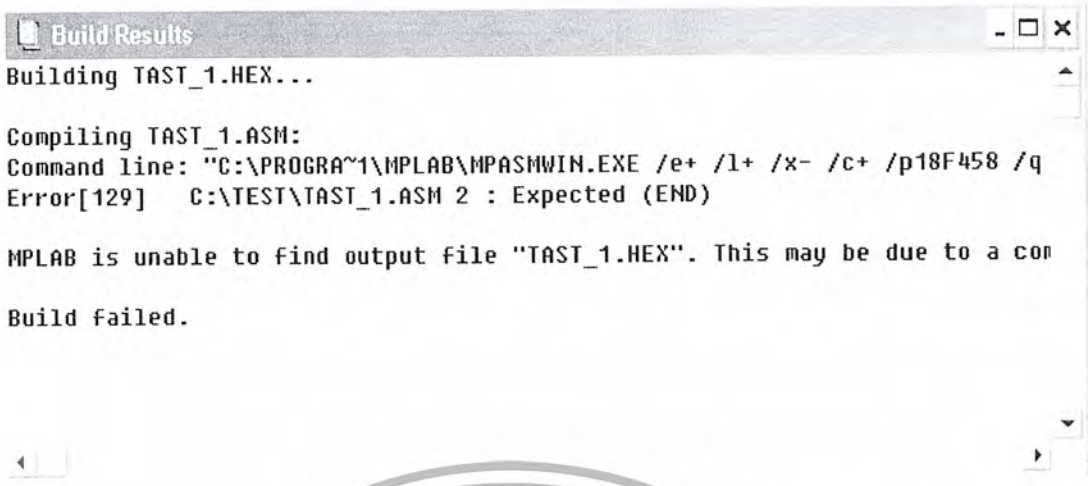


รูปที่ ฉ.29 หน้าต่างของ Node Properties

โดยจะกลับเข้าสู่หน้าต่างของ Edit Project อีกครั้ง ซึ่งถึงขั้นนี้เราได้ทำการกำหนด และตั้งค่าต่างๆ เรียบร้อยแล้วต่อไปเป็นการ Build หรือ การคอมไพล์โปรแกรมเพื่อให้ได้ไฟล์เอาต์พุต (Hex File) ที่ต้องการ

6. ทำการ Build หรือ คอมไพล์โปรแกรม โดยในการ Build นี้จะทำได้หลายวิธี คือ การคลิกที่ปุ่ม Build Node ในหน้าต่างของ Edit Project, ในเมนู Project เลือก Build All หรือ Project เลือก Build Node และ ในการคลิกที่ปุ่ม  ขึ้นอยู่กับความสะดวกในการใช้งาน

เมื่อทำการ Build หรือ คอมไพล์โปรแกรมหากไม่มีข้องผิดพลาดใดๆ เราก็จะได้ Hex File และไฟล์อื่นๆ ที่เราได้กำหนดไว้ออกมา แต่ไฟล์ที่จะนำไปโหลดลง CPU จะมีไฟล์เดียว คือ ไฟล์นามสกุล .Hex แต่หากเกิด Error ให้เราตรวจสอบข้องผิดพลาดต่างๆ ในการเขียนโปรแกรมเช่น เรื่องของ ไวยากรณ์ ซึ่งจะมี Dialog แจ้งให้ทราบว่าเกิด Error จากอะไร ดังรูปที่ ฉ.30



รูปที่ ฉ.30 ตัวอย่างการเกิด Error จากการคอมไพล์

รูปที่ ฉ.31 ตัวอย่างผลลัพธ์จากการคอมไพล์ในกรณีที่ไม่เกิดข้อผิดพลาด (Error)

โดยจากรูปที่ ฉ.31 เป็นผลลัพธ์จากการคอมไพล์โปรแกรมในกรณีที่ไม่มีข้อผิดพลาดใดๆ แต่ก็จะมี Message เตือนในเรื่องตำแหน่งของรีจิสเตอร์ในแบงก์ต่างๆ แม้ว่าในโปรแกรมเราจะใช้งานรีจิสเตอร์ตรงตำแหน่งแบงก์ต่างๆ ถูกต้องก็ตาม คำเตือนนี้ก็จะยังปรากฏอยู่ ทั้งนี้เพื่อเตือนให้เราระวังในเรื่องดังกล่าวนั้นเอง

คู่มือ
เฉลยใบงานการทดลอง

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

เฉลยใบงานที่ 1

การทดลองใช้งานพอร์ตของ PIC18F458

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.1 ทำการคอมไพล์ให้เป็น Hex File

2. ไปที่เมนู OPTION เลือก Development Mode กำหนดโหมดเป็น MPLAB-SIM Simulator

3. เลือก Process เป็นรุ่น PIC18F458 จากนั้นกดปุ่ม Apply แล้วกด OK

4. ไปที่เมนู Project เลือก New Project ตั้งชื่อ Project Name แล้วกด OK

5. ไปที่เมนู Edit Project ให้ Remove Project File ออกให้หมดแล้วกด OK

6. ไปที่เมนู File เลือก New Souse เพื่อเขียนโปรแกรมดังรูปที่ จ.3 ลงไป

7. จากนั้น Save File เป็นนามสกุล . asm จากนั้นทำการแอสเซมเบลอร์

8. ไปที่เมนู Project เลือก Edit Porject แล้วกำหนดชื่อไฟล์เป็น ชื่อที่เราบันทึกไว้ใน ข้อ 6 เพื่อ Add ไปยัง Project File

9. ไปที่เมนู Project เลือก Make Project เพื่อแอสเซมเบลอร์โปรแกรม

10. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

11. ต่อวงจรตามรูปที่ จ.2

12. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

<u>จะเห็นแอลอีดีที่ต่ออยู่กับพอร์ต A และ พอร์ต B ติดทุกดวง</u>

13. เปลี่ยนโปรแกรมจาก MOVLW 0xFF ในบรรทัดที่ ถัดจาก START เป็น MOVLW 0x01 ทำการคอมไพล์และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ใหม่อีกครั้ง บันทึกผลการทดลอง

<u>แอลอีดีต่ออยู่กับพอร์ตA ที่บิต RA0 ติดเพียงดวงเดียว</u>

14. ทดลองนำเอาเครื่องหมายที่อยู่หน้า MOVLW '00000000' ออกทำการคอมไพล์ใหม่อีกครั้งแล้วโปรแกรมลงในตัวไมโครคอนโทรลเลอร์บันทึกผลการทดลอง

<u>แอลอีดีทุกดวงดับ</u>

คำถามท้ายการทดลอง

1. จงอธิบายการทำงานของพอร์ต A และ E มาพอเข้าใจ

พอร์ต A และพอร์ต E เป็นทั้งพอร์ตอินพุตและเอาต์พุตแต่จะมีการใช้งานต่างจากพอร์ตอื่นๆ ตรงที่ใช้ในการรับสัญญาณแอนะลอกด้วยและนอกจากนี้พอร์ต E ยังมีหน้าที่พิเศษอีกอย่างหนึ่งคือใช้เป็นขาสัญญาณในการอ่านและเขียนข้อมูลในโหมด PSP

2. จากโปรแกรมที่ จ.3 เมื่อต้องการเขียนโปรแกรมให้พอร์ต RA0 ดับส่วน RA2 และ RA3 ติดจะแทรกโปรแกรมที่ตำแหน่งไหนจงอธิบาย

เปลี่ยนโปรแกรมจากบรรทัดใน บรรทัดจาก START จาก MOVLW 0xFF เป็น MOVLW 0x03

3. เมื่อต้องการใช้งานพอร์ต A และ E ให้ทำงานในโหมด PSP นั้นจะเขียนโปรแกรมอย่างไร

เขียนค่าข้อมูลไปยังรีจิสเตอร์ ADCON0 เป็น MOVLW B'00000000'

4. ถ้าต้องการกำหนดพอร์ต D เป็นอินพุตทั้งหมดจะเขียนโปรแกรมเพื่อกำหนดพอร์ตอย่างไร

```
        MOVLW   0x00
        MOVWF   TRISD   ; PORTD=00000000
        CLRF    PORTD
START   ……      ……..
```

เฉลยใบงานที่ 2

การทดลองใช้งานสัญญาณนาฬิกาของ PIC18F458

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.6 ทำการคอมไพล์ให้เป็น Hex File

2. ต่อสายเชื่อมต่อระหว่างโมดูลตามรูปที่ จ.5

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ เลือกโหมดสัญญาณนาฬิกาจากโปรแกรม Epic Win ให้เป็น RC

4. เซตจั๊มเปอร์ที่ตำแหน่ง XT ให้เป็น RC ในโมดูลหลัก PIC-ICSP

5. รันโปรแกรมสังเกตผลที่เกิดขึ้นกับแอลอีดีทั้ง 8 ดวง บันทึกผลการทดลอง

แอลอีดีทั้ง 8 ดวงติดและวิ่งเรียงกันจากไปซ้ายขวา

6.ใช้โปรแกรมในรูปที่ จ.6 คอมไพล์อีกครั้ง แต่ในการโปรแกรมเลือกโหมดสัญญาณนาฬิกาเป็น HS โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ อีกครั้ง

7. เซตจั๊มเปอร์ที่ตำแหน่ง XT สังเกตผลที่เกิดขึ้นกับแอลอีดีทั้ง 8 ดวง บันทึกผลการทดลอง

แอลอีดีทั้ง 8 ดวงติดและวิ่งเรียงกันจากซ้ายไปขวาแต่จะวิ่งเร็วขึ้นกว่าเดิม

คำถามท้ายการทดลอง

1. จงอธิบายวิธีการป้อนสัญญาณนาฬิกาแบบ RC และ XT ว่ามีวิธีการอย่างไร

การป้อนสัญญาณนาฬิกาแบบ RC นั้นจะต่อขา OSC1 เข้ากับวงจร RC เพื่อกำเนิดความถี่ โดยการต่อลักษณะนี้ขา OSC2 จะมีค่าความถี่ออกมาเท่ากับความถี่ที่ขา OSC1/4 ส่วนการต่อแบบ XT นั้นจะใช้ขา OSC1 และ OSC2 ต่อใช้งานร่วมกับคริสตรอลจะต่างจากแบบ RC คือขา OSC2 ไม่ปล่อยลอยนั่นเอง

2. จงบอกความแตกต่างของการต่อสัญญาณนาฬิกาแบบ RC และ XT

ความแตกต่างของการต่อสัญญาณนาฬิกาแบบ RC และ XT คือ การต่อแบบ RC จะลอยขา OSC2 ไว้ส่วนแบบ XT จะถูกใช้งานทั้ง ขา OSC1 และ OSC2

3. จงบอกข้อดีและข้อเสียของการป้อนสัญญาณนาฬิกาแต่ละแบบมาพอเข้าใจ

ในการป้อนสัญญาณนาฬิกาแต่ละแบบจะมีข้อดีและข้อเสียที่แตกต่างกันในแบบ RC นั้น ข้อดีคือหาซื้อง่าย ประหยัด ส่วนข้อเสียคือ เกิดความผิดพลาดสูง ไม่เที่ยงตรง ส่วนการป้อนสัญญาณนาฬิกาในแบบ XT, HS, นั้นข้อดีคือมีความเที่ยงตรงสูง สูญเสียความร้อนน้อยกว่าแบบ RC ข้อเสียคือไม่สามารถนำขา OSC 2 ไปใช้งานปกติได้

เฉลยใบงานที่ 3
การทดลองใช้งานหน่วยความจำข้อมูลอีอีพรอมภายใน
PIC18F458

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ได้ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.8 ทำการคอมไพล์ให้เป็น Hex File

2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

3. ต่อวงจรตามรูปที่ จ.7

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

ผลการทดลองเมื่อรันโปรแกรมจะเห็นแอลอีดีที่ต่ออยู่กับ RB0 ติดแสดงให้ทราบว่าขณะนี้การเขียนข้อมูลเสร็จสิ้นแล้วและแอลอีดีที่ต่ออยู่กับพอร์ต D ติดสว่างเป็น 0x03 ในเลขฐานสิบหกเป็นการอ่านข้อมูลที่นำไปเก็บไว้จากหน่วยความจำข้อมูลตำแหน่ง 0x20 ออกมาแสดงผลยังแอลอีดี

5. ลองเปลี่ยนโปรแกรมจาก MOVLW 0xFF เป็น MOVLW 0x03 จากนั้นทำการคอมไพล์และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้ง

6. จากการทดลองข้อ 5 เมื่อรันโปรแกรมบันสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลการทดลอง

ผลการทดลองเมื่อรันโปรแกรมแอลอีดีติดเป็นข้อมูล 0xFF ในเลขฐานสิบหก

คำถามท้ายการทดลอง

1. รีจิสเตอร์ที่เกี่ยวข้องกับการอ่านและเขียนหน่วยความจำข้อมูลอีอีพรอมมีกี่ตัวแต่ละตัวมีหน้าที่อะไรบ้างจงอธิบายและการทำงานอย่างไรจงอธิบาย

มี 4 ตัว ดังนี้

1. EECON1 รีจิเตอร์ควบคุมการเข้าถึงหน่วยความจำ
2. EECON2 รีจิสเตอร์จัดลำดับการเขียนข้อมูลในหน่วยความจำ
3. EEDATA รีจิสเตอร์บัฟเฟอร์ข้อมูลสำหรับการอ่านและเขียน
4. EEADR รีจิสเตอร์ที่เก็บตำแหน่งมีค่าตั้งแต่แอดเดรสตั้งแต่ 00h- FFh

2 . เมื่อต้องการเขียนโปรแกรมให้เขียนข้อมูล 0x07 ไปยังหน่วยความจำตำแหน่ง 0x70 จากนั้นทำการอ่านออกมาแสดงผลยังแอลอีดีจะแทรกโปรแกรมตรงไหนหรือเขียนโปรแกรมอย่างไรจงอธิบาย

จากโปรแกรมในรูปที่ จ.8 ต้องทำการแทรกโปรแกรมในบรรทัดที่ 10 ดังนี้

```
;**********WRITE**************
WRITE   MOVLW       0x70                ; Assign address
        MOVWF       EEADR
        MOVLW       0X07                ; Assign data
        MOVWF       EEDATA
        BCF         EECON1,EEPGD        ; Data EEPROM section
        BSF         EECON1,WREN         ; Write enable
```

รูปที่ จ.32_ โปรแกรมให้เขียนข้อมูล 0x07 ไปยังหน่วยความจำตำแหน่ง 0x70

เฉลยใบงานที่ 4

การทดลองใช้งานโมดูลสร้างแรงดันอ้างอิงของ PIC18F458

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.11 เสร็จแล้วทำการคอมไพล์ให้เป็น Hex file

2 ต่อสายเชื่อมต่อระหว่างโมดูลตามรูปที่ จ.10

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

4. รันโปรแกรม ทดลองกด SW0-SW3 ที่โมดูลแอลอีดีสังเกตผลที่เกิดขึ้นกัลแอลอีดีทั้ง 4 ดวงที่ต่ออยู่กับ RD4-RD7

5. เลือกแรงดันเอาต์พุตจากสวิตซ์ SW0-SW2 โดยทำการกดเลือกย่านความถี่ เมื่อกดสวิตซ์ตามย่านความถี่ต่างๆ สังเกตผลที่เกิดขึ้นบันทึกผล

6. วัดค่าแรงดันที่ขา RA2/AN2 เทียบกับแรงดันเอาต์พุตที่เลือกบันทึกผลในตารางที่ จ.2

ตารางที่ จ.1 ตารางการทดลอง

สวิตซ์	แรงดันอ้างอิงของโมดูลสร้างแรงดันอ้างอิง	
	ค่าที่คำนวณได้	ค่าที่วัดได้
S1	1.25	1.21
S2	2.08	2.03
S3	2.71	2.66
S4	3.13	3.08

คำถามท้ายการทดลอง

1. จงบอกรีจิสเตอร์ที่เกี่ยวข้องกับโมดูลสร้างแรงดันอ้างอิงภายใน PIC18F458 มีอะไรบ้าง พร้อมอธิบายหลักการทำงานมาพอเข้าใจ

รีจิสเตอร์ที่เกี่ยวข้อง คือ CVRCON ในการใช้งานสร้างค่าแรงดันอ้างอิงนั้นจะกระทำได้โดยผ่านบิต CVR3-CVR0

2. จงบอกการนำไปใช้งานของโมดูลสร้างแรงดันอ้างอิงของภายใน PIC18F458

สามารถนำโมดูลสร้างแรงดันอ้างอิงไปใช้ประโยชน์ได้ เช่น เป็นแรงดันจ่ายให้แก่วงจรอื่นๆ ได้

3. เมื่อต้องการให้แรงดันอ้างอิงที่ขา RA2 มีค่าเป็นศูนย์จะเขียนโปรแกรมอย่างไร

เขียนโปรแกรมกำหนดให้บิต CVROE ให้เป็น 0 หรือเคลียร์บิต CVROE

เฉลยใบงานที่ 5
การทดลองใช้งานโมดูลการแปลงสัญญาณแอนะลอกเป็นดิจิตอล ภายใน PIC18F458

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.13 เสร็จแล้วทำการคอมไพล์ให้เป็น Hex File

2. ต่อสายเชื่อมต่อระหว่างโมดูลตามรูปที่ จ.12

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

4. ทำการปรับค่าความต้านทานที่ต่ออยู่กับ RA5 สังเกตการเปลี่ยนแปลงที่เกิดขึ้น

5. จากข้อ 4 ผลการทดลองเป็นดังนี้

จะเห็นว่าเมื่อทำการปรับค่าตัวต้านทานที่ต่ออยู่กับ RA5 แล้วผลที่เกิดขึ้นคือ แอลอีดีที่ต่ออยู่กับพอร์ตนั้นมีการเปลี่ยนแปลงตามการปรับ

6. ทำการเปลี่ยนโปรแกรมจาก movlw 10100001 เป็น 10000001 ทำการคอมไพล์และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ใหม่อีกครั้ง

7. รันโปรแกรม ทดลองปรับค่าความต้านทานที่ต่ออยู่กับ RA5/AN5 ผลที่ได้เป็นดังนี้

ไม่มีการเปลี่ยนแปลงเกิดขึ้นเนื่องจากตอนนี้ได้เปลี่ยนตำแหน่งการรับสัญญาณแอนะลอก

8. ทดลองปรับค่าความต้านทานที่ต่ออยู่กับ RA2/AN2 สังเกตผลที่เกิดขึ้นกับโมดูลแอลอีดี บันทึกผลการทดลอง

มีการเปลี่ยนแปลงแอลอีดีที่ต่ออยู่กับพอร์ตนั้นมีการเปลี่ยนแปลงตามการปรับ

คำถามท้ายการทดลอง

1. จงอธิบายหลักการในการแปลงสัญญาณแอนะลอกเป็นดิจิตอลของไมโครคอนโทรลเลอร์ PIC18F458 มาพอเข้าใจ

ในการแปลงสัญญาณแอนะลอกเป็นดิจิตอลของไมโครคอนโทรลเลอร์ PIC18F458 นั้นจะหลักการคือการกำหนดให้พอร์ต A และพอร์ต E ของ PIC18F458 นั้นเป็นอินพุตรับค่าแรงดันแอนะลอกที่รีจิสเตอร์ ADCON1 และทำการเลือกช่องหรือขาที่รับที่ ADCON0

2. จงบอกรีจิสเตอร์ที่เกี่ยวข้องกับการแปลงสัญญาณแอนะลอกเป็นดิจิตอลคอนโทรลเลอร์ PIC18F458 พร้อมอธิบายหน้าที่ในการทำงานมาพอเข้าใจ

รีจิสเตอร์ที่เกี่ยวข้อง คือ ADCON0 และ ADCON1

ADCON0 ขนาด 8 บิต เป็นรีจิสเตอร์หลักที่ใช้ในการควบคุมการทำงานของโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอลเป็นรีจิสเตอร์ควบคุมการทำงานของโมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอลที่ต้องทำงานร่วมกับ ADCON0 มีขนาด 8 บิต โดยรีจิสเตอร์ตัวนี้ใช้กำหนดการทำงานของขาพอร์ตที่เกี่ยวข้องกับ โมดูลแปลงสัญญาณแอนะลอกเป็นดิจิตอล และเลือกใช้รูปแบบของข้อมูลผลลัพธ์ที่ได้จากการแปลงสัญญาณ

3. เมื่อต้องการให้ขา RA3 รับสัญญาณแอนะลอกอินพุตจะเขียนโปรแกรมอย่างไรหรือแทรกโปรแกรมที่ตำแหน่งไหน

กระทำได้โดยการเปลี่ยนโปรแกรมในบรรทัดที่ 7 จาก MOVLW 10000001 เป็น MOVLW 10000010

เฉลยใบงานที่ 6

การทดลองการอินเตอร์รัพต์ของ PIC18F458

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ได้ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.15 ทำการคอมไพล์ให้เป็น Hex File

2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

3. ต่อวงจรตามรูปที่ จ.14

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

แอลอีดีดับ

5. ทดลองกดสวิตซ์ที่ต่ออยู่กับพอร์ต B ตำแหน่ง S0-S4 สังเกตผลที่เกิดขึ้นบันทึกผลในตาราง

ตารางที่ จ.2 ผลการติดของแอลอีดีเมื่อกดสวิตช์ S0-S2

ตำแหน่งสวิตซ์ที่ถูกกด	ผลที่เกิดขึ้นกับแอลอีดี
S0	ติด
S1	ติด
S2	ติด
S3	ติด

คำถามท้ายการทดลอง

1. จงอธิบายการทำงานของแหล่งกำเนิดการอินเตอร์รัพต์จากการเปลี่ยนแปลงลอจิกที่ขา RB4-RB7

การอินเตอร์รัพต์จากเปลี่ยนแปลงลอจิกที่ขา RB4-RB7 เป็นการตรวจสอบลอกจิกที่ขาพอร์ตดังกล่าวโดยการใช้งานจะต้องทำการเปิดการอินเตอร์รัพต์รวมก่อนที่บิต GIE จากนั้นเลือก

เปิดการทำงานของการอินเตอร์รัพต์ขา RB4-RB7 ที่บิต RBIE ในรีจิสเตอร์ INTCON โดยการทำงานนั้นเมื่อเกิดการเปลี่ยนแปลงลอจิกที่ขาดังกล่าวเกิดขึ้นนั้นจะโปรแกรมก็จะกระโดดไปแอดเดรสบริการการอินเตอร์รัพต์ที่ 0008h

2. จงอธิบายการทำงานของโปรแกรมที่ จ.15

จากโปรแกรมในรูปที่ จ.15 เป็นโปรแกรมทดสอบการอินเตอร์รัพต์ที่เกิดขึ้นจากพอร์ต B ที่ขา RB4-RB7 โดยใช้สวิตช์ในการทดสอบเมื่อมีการกดสวิตช์ที่ขาดังกล่าวจะทำให้เกิดการอินเตอร์รัพต์ขึ้น และแอลอีดีที่ต่ออยู่นั้นแสดงผลของการกดสวิตช์ที่ตำแหน่งต่างๆ ที่มีการอินเตอร์รัพต์

เฉลยใบงานที่ 7
การใช้งานไทเมอร์เคาน์เตอร์ภายใน PIC18F458

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เขียนตามรูปที่ จ.18 เพื่อสร้างสัญญาณพัลส์โดยใช้ไทเมอร์เคาน์เตอร์ภายใน PIC18F458 โดยเลือกแหล่งกำเนิดสัญญาณนาฬิกาจากวงจรกำเนิดสัญญาณนาฬิกาภายในของ PIC18F458

2. ต่อวงจรตามรูปที่ จ.16

3. แอสเซมเบอร์โปรแกรมที่ จ.18 แล้วโปรแกรมข้อมูลลงบน PIC18F458 ด้วยโปรแกรม Epic Win

4. รันโปรแกรมแล้ว ใช้ออสซิลโลสโคปวัดสัญญาณที่ขา RB0 บันทึกรูปสัญญาณที่วัดได้

5. จากผังการทำงานที่ออกแบบไว้เขียนโปรแกรมตามรูป จ.19 เพื่อสร้างสัญญาณสี่เหลี่ยมความถี่ 1 kHz โดยใช้การอินเตอร์รัพต์เนื่องจาก TMR0 เกิดโอเวอร์โฟลว์

6. ใช้วงจรตามรูปที่ จ.16 ในการทดลอง

7. ทำการคอมไพล์ แล้วเขียนลงบน PIC18F458 ด้วยโปรแกรม Epic Win

8. รันโปรแกรมแล้ว ใช้ออสซิลโลสโคปวัดสัญญาณที่ขา RB0 บันทึกรูปสัญญาณที่วัดได้

10. ถ้าต้องการสร้างสัญญาณพัลส์ความถี่ 2 kHz โดยใช้การอินเตอร์รัพต์เนื่องจาก TMR0 เกิด โอเวอร์โฟลว์ ต้องกำหนดค่าของ

ปรีสเกลเลอร์ = 256

TMR0 = 250

11. เปลี่ยนค่าของอัตราส่วนปรีสเกลเลอร์เป็น 256 แล้วแก้ไขโปรแกรมเป็นจ.19 ทำการแอสเซมเบลอร์ แล้วเขียนลงบน PIC18F458 ทำการทดลองซ้ำในข้อที่ 8

12. จากผังการทำงานที่ออกแบบไว้กำหนดให้ไทเมอร์เคาน์เตอร์รับสัญญาณนาฬิกาจากภายนอกผ่านทางขา RA4/T0CKI เมื่อ TMR0 เกิดโอเวอร์โฟลว์จะทำการเลื่อนบิตของพอร์ต B

13. เขียนโปรแกรม จ.19 ทำการคอมไพล์ และเขียนลงบน PIC18F458 ด้วยโปรแกรม Epic Win

14. ต่อวงจรตามรูปที่ จ.17 โดยเลือกความถี่ของโมดูลพัลส์เจเนอร์เรเตอร์เท่ากับ 1 Hz

15. รันโปรแกรมเลื่อนสวิตช์ ไปที่ตำแหน่งรัน

16 นับจำนวนการกะพริบของ LED มอนิเตอร์ที่ต่ออยู่กับขา RA4/T0CKI แล้วสังเกตการทำงานของ LED มอนิเตอร์ที่ต่ออยู่กับพอร์ต B

LED ที่ RA4/TOCKI กระพริบ__4__ ครั้ง แล้ว LED ที่พอร์ต B จึงเกิดการเลื่อน 1 บิต

17. แก้ไขโปรแกรมที่ จ.19 เปลี่ยนค่าของปรีสเกลเลอร์เป็น 256

18. เปลี่ยนความถี่ของโมดูลพัลส์เจเนอเรเตอร์ เป็น 1 kHz

19. แอสเซมเบลอร์โปรแกรมที่ทำการแก้ไข แล้วเขียนลงบน PIC18F458 อีกครั้ง

20. รันโปรแกรมเลื่อนสวิตช์ ไปที่ตำแหน่งรัน

21. สังเกตการทำงานของ LED มอนิเตอร์ที่พอร์ต B

LED มอนิเตอร์ที่พอร์ต B เกิดการเลื่อนบิตในอัตรา__1__ครั้งต่อวินาที

22. จากการทดลองในข้อ 21 จงคำนวณหาค่าของจำนวนสัญญาณนาฬิกาจากภายนอกที่ให้ข้อมูลที่พอร์ต B เกิดการเลื่อนไป 1 บิต

จำนวนของสัญญาณนาฬิกาเท่ากับ__2__ลูก

คำถามท้ายการทดลอง

1. ไทเมอร์เคาน์เตอร์ใน PIC18F458 มีกี่ตัว แต่ละตัวมีขนาดกี่บิต และมีส่วนประกอบภายในอะไรบ้าง จงอธิบาย

ไมเมอร์ภายในไมโครคอนโทรลเลอร์ PIC18F458 มีทั้งหมด 3 ตัวคือ ไทเมอร์ 0, ไทเมอร์ 1, ไทเมอร์ 3 โดยไทเมอร์ในไทเมอร์ 0 ทำงานได้ทั้งในโหมด 8 บิต และ 16 บิต มีปรีสเกลเลอร์ขนาด 8 บิต และตัวนับหรือเคาน์เตอร์ โดยไทเมอร์เคาน์เตอร์ ทุกตัว ส่วนไทเมอร์ 2 มีรีสเตอร์คาบเวลาทำหน้าที่เปรียบเทียบค่ากับไทเมอร์ 2

2. รีจิสเตอร์ที่เกี่ยวข้องในการใช้งานไทเมอร์เคาน์เตอร์ของ PIC18F458 มีอะไรบ้างและเกี่ยวข้องอย่างไรจงอธิบาย

3. จงอธิบายการทำงานของไทเมอร์เคาน์เตอร์ใน PIC18F458

ในไมเมอร์ของ PIC18F458 นั้นสามารถที่จะทำงานได้ทั้งโหมดไมเมอร์และเคาน์เตอร์ ไทไมอร์ คือ ตัวตั้งเคาน์เตอร์ คือ ตัวนับ

4. จากการทดลองจงสรุปถึงการนำไทเมอร์เคาน์เตอร์ใน PIC18F458 ไปใช้ประโยชน์ว่านำไปใช้งานอย่างไรบ้าง และทำได้อย่างไร

สร้างเครื่องหน่วงเวลาเพื่อสร้างสัญญาณรูปสี่เหลี่ยมทำได้โดยใช้ Timer ในการหน่วงเวลาในการสร้างลอจิก Low กับ High ซึ่งจะทำให้เกิด Pulse ขึ้นตามต้องการ

5. จงอธิบายความหมายของปรีสเกลเลอร์และโพสต์สเกลเลอร์

ปรีสเกลเลอร์ จะเรียก เมื่อทำงานร่วมกับไทเมอร์ เคาน์เตอร์ โพสต์สเกลเลอร์ จะเรียกเมื่อทำงานร่วมกับ Watch Dog Timer

เฉลยใบงานที่ 8

การใช้งาน PIC18F458 ขับแอลอีดีตัวเลข 7 ส่วน

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ได้ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.21 ทำการคอมไพล์ให้เป็น Hex File

2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

3. ต่อวงจรตามรูปที่ จ.21

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้

จากการทดลองผลที่เกิดขึ้นคือ โมดูลแสดงผลแอลอีดีตัวเลข 7 ส่วนแบบคาโถดหลักที่ 1 ติดเป็น 0-9

5. ลองเปลี่ยนโปรแกรม จาก DT 3F, 06, 5B, 4F, 66, 6D, 7D, 07, 7F, 6F, 00 เป็น 6F, 7F, 07, 7D , 6D ,66, 4F, 5B, 06, 3F จากนั้นทำการคอมไพล์ และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้ง

6. จากการทดลองข้อ 5 เมื่อรันโปรแกรมบันสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลการทดลอง

จากการทดลองผลที่เกิดขึ้นคือ โมดูลแสดงผลแอลอีดีตัวเลข 7 ส่วนแบบคาโถดหลักที่ 2 ติดเป็น 0-9

7. จากโปรแกรมการทดลองในรูปที่ จ.21 เปลี่ยนข้อมูลจาก MOVL 0x00 มาเป็น MOVLW 0x01 ผลจากการสังเกตเป็นดังนี้

จากการทดลองผลที่เกิดขึ้นคือ โมดูลแสดงผลแอลอีดีตัวเลข 7 ส่วนแบบอาโถดหลักที่ 1 ติดเป็น 0-9

คำถามท้ายการทดลอง

1. จงอธิบายหลักการแสดงผลของแอลอีดีตัวและเลข 7 ส่วนแบบคอมมอนคาโถดอาโนด

หลักการ การขับให้แอลอีดีตัวเลข 7 ส่วน แบบคาโถดร่วมสว่างจะต้องจ่ายไฟลบเข้าที่ขาร่วม แล้วจ่ายไฟบวกเข้าที่ขาอาโนด

เฉลยใบงานที่ 9

การทดลองใช้งาน PIC18F458 ร่วมกับสวิตช์เมตริกซ์ 4x4

ลำดับขั้นตอนการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB เขียนโปรแกรมดังรูปที่ จ.24 ทำการคอมไพล์
2. ต่อสายเชื่อมต่อระหว่างโมดูลตามรูปที่ จ.23
3. เขียนข้อมูลลงบน PIC18F458 ทำการรันโปรแกรมบน PIC18F458
4. เมื่อกดสวิตช์แต่ละตัว ผลที่ได้คือ

ตารางที่ ฉ.3 ผลการติดของไดโอดเปล่งแสงตัวเลข 7 ส่วนจากการกดสวิตช์

กดสวิตช์	แสดงผลที่ตัวเลข 7 ส่วน
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	A
B	B
C	C
D	D
E	E
F	F

คำถามท้ายการทดลอง

1. จงอธิบายหลักการทำงานของโปรแกรมดังรูปที่ จ.24

ในโปรแกรมที่ จ.24 นั้นเป็นโปรแกรมรับค่าจากสวิตซ์เมตริกซ์ซึ่งหลักการคือการค้นหาตำแหน่งสวิตซ์ที่ถูกกดในขณะนั้นจากนั้นนำค่าที่ได้ไปเปิดตารางข้อมูลแสดงผลที่แอลอีดีตัวเลข 7 ส่วนเป็นเลขของตำแหน่งสวิตซ์ที่ถูกกด

2. จงอธิบายหลักการทำงานของสวิตซ์เมตริกส์ขนาด 4x4 มาพอเข้าใจ

การทำงานของเมตริกซ์สวิตซ์นั้นจะเป็นการสแกนแถวที่ละแถวแล้วตรวจสอบคีย์ในแถวนั้น ว่าถูกกดหรือไม่โดยการตรวจสอบคอลัมน์ถ้าในแนคอลัมน์ถูกกดก็จะนำไปแสดงผลที่แอลอีดีตัวเลข 7 ส่วน

เฉลยใบงานที่ 10
การทดลองใช้งาน PIC18F458 ขับสเตปปิ้งมอเตอร์

ลำดับขั้นตอนการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.26 ทำการคอมไพล์ให้เป็น Hex File

2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

3. ต่อวงจรตามรูปที่ จ.25

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

สังเกตการหมุนของสเตปจะเป็นทีละสเตปโดยการสังเกตจากแอลอีดีแสดงผลทั้ง 4 เฟส

5. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.27 ทำการคอมไพล์ให้เป็น Hex File

6.โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้งรันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

สังเกตการหมุนของสเตปจะมีความละเอียดของการหมุนมากยิ่งขึ้นโดยการสังเกตจากแอลอีดีแสดงผลทั้ง 4 เฟส

คำถามท้ายการทดลอง

1. จงอธิบายการทำงานของโปรแกรมรูปที่ จ.27

ในรูปที่ จ.27 เป็นโปรแกรมการขับสเตปปิ้งมอเตอร์แบบที่ง่ายที่สุดคือการจ่ายกระแสให้แก่สเตปปิ้งมอเตอร์ทีละเฟสคือใน 1 สเตปจะทำงานเพียง 1 เฟสนั่นเอง

2. จากโปรแกรมรูปที่ จ.27 เป็นการกระตุ้นและควบคุมการหมุนสเตปปิ้งมอเตอร์แบบเวฟเพราะเหตุใด

เนื่องจากเป็นการป้อนกระแสให้สเตปปิ้งมอเตอร์หมุนที่ละเฟสใน 1 สเตป

3. จงอธิบายความแตกต่างของการขับปิ้งมอเตอร์แบบเวฟกับแบบ 2 เฟส

ความแตกต่างของทั้ง 2 แบบ คือการรูปแบบการป้อนค่าให้แก่สเตปปิ้งมอเตอร์ทำงาน โดยแบบเวฟนั้นจะเป็นการป้อนให้ทำงานทีละเฟส ใน 1สเตป ส่วนแบบเวฟจะทำงาน 2 เฟส พร้อมกันใน 1 สเตป

3. จงอธิบายความแตกต่างระหว่าง แอลอีดีตัวเลข 7 ส่วนแบบคอมมอนคาโถดและอาโนด

การแสดงผลที่เป็นแอลอีดีตัวเลข 7 ส่วน (LED 7 Segment) แต่ละส่วนหรือเซกเมนต์มีชื่อเรียกแตกต่างกันตามตำแหน่งที่ได้รับการจัดวาง คือ a, b, c, d, e, f และ g ส่วน db เป็น แอลอีดีอีก 1 ใช้แสดงตัวเลขที่จุดทศนิยมแอลอีดีตัวเลข 7 ส่วนนี้มีทั้งแบบคาโถดร่วม (Common Cathode) และแบบอาโนดร่วม (Common Anode) การขับให้แอลอีดีตัวเลข 7 ส่วน แบบคาโถดร่วมสว่างจะต้องจ่ายไฟลบเข้าที่ขาร่วม แล้วจ่ายไฟบวกเข้าที่ขาอาโนด ซึ่งก็คือขาของแต่ละเซกเมนต์ นั่นเอง ในขณะที่แอลอีดีตัวเลข 7 ส่วนแบบอาโนดร่วมจะต้องจ่ายไฟบวกเข้าที่ขาร่วมแล้วจ่ายไฟลบเข้าที่ขาคาโถดซึ่งเป็นขาของแต่ละเซกเมนต์

เฉลยใบงานที่ 11

การทดลองใช้งาน PIC18F458 สร้างสัญญาณเสียง

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.29 ทำการคอมไพล์ให้เป็น Hex file

2. ต่อวงจรตามรูปที่ จ.28

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

4. รันโปรแกรมเลื่อนสวิตช์มาที่ตำแหน่ง RUN

5. บันทึกผลการทดลอง

ผลการทดลองเมื่อรันโปรแกรมจะได้ยินเสียงดังของลำโพงเป็นดังแล้วหยุดเป็นจังหวะ

6. ทดลองเปลี่ยน โปรแกรมหน่วงเวลาเดิม .2 เปลี่ยนค่าเพิ่มขึ้นทีละ 1 ค่า ไปจนถึง .9 สังเกตผลการทดลองที่เกิดขึ้น บันทึกผลการทดลอง

ผลการทดลองเมื่อรันโปรแกรมจะต่างจากการทดลองในข้อ 5 ตรงที่เสียงจะช้าลงเรื่อยๆ เมื่อเปลี่ยนค่าตัวแปลให้กับโปรแกรมหน่วงเวลา

คำถามท้ายการทดลอง

1. จงอธิบายหลักการสร้างสัญญาณเสียงของ PIC18F458

การสร้างสัญญาณเสียงของ PIC18F458 นั้นจะใช้หลักการหน่วงเวลาเพื่อสร้างพัลส์ให้แก่ลำโพงแล้วลำโพงก็จะเปล่งเสียงตามค่าความถี่ที่ป้อนให้นั่นเอง

2. เมื่อต้องการเปลี่ยนระดับสัญญาณเสียงให้เป็นความถี่เสียงต่างๆ จะเปลี่ยนโปรแกรมที่ตำแหน่งไหน เพราะเหตุใด

ทำการเปลี่ยนค่าตัวแปลหน่วงเวลาเป็นค่าต่างๆ เพราะการหน่วงเวลาช้าหรือเร็วจะมีผลต่อค่าความถี่ที่ป้อนให้กับลำโพง

3. จากการทดลองสามารถนำไปประยุกต์ใช้งานได้อย่างไร ยกตัวอย่างการนำเอาไปประยุกต์ใช้งาน 1 ตัวอย่าง พร้อมอธิบาย

ตัวอย่างการนำไปประยุกต์ใช้งานคือสามารถสร้างเป็นเครื่องดนตรีได้เช่นเมโรดี้ต่างๆ และประยุกต์ใช้งานในงานรักษาความปลอดภัยได้

เฉลยใบงานที่ 12

การทดลองใช้งาน PIC18F458 ร่วมกับโมดูลแสดงผลแอลซีดี

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม MPLAB ทำการเชื่อมโปรแกรมตามรูปที่ จ.33 ทำการคอมไพล์ให้เป็น Hex.File

2. ต่อวงจรตามรูปที่ จ.32

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ด้วยโปรแกรม Epic Win

4. รันโปรแกรมเลื่อนสวิตช์ที่ตำแหน่ง RUN

5. สังเกตผลที่เกิดขึ้นกับจอแสดงผลแอลซีดี บันทึกผลการทดลอง

ทันทีที่รันโปรแกรมจะปรากฏข้อความแสดงผลที่แอลซีดีเป็น KMITL และไม่แสดงเคอร์เซอร์

6. ทดลองเปลี่ยนข้อมูลจาก KMITL ทำการคอมไพล์ และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้ง บันทึกผลการทดลอง

จะปรากฏข้อความดังกล่าวที่ได้ทำการเปลี่ยนแปลงแต่ไม่แสดงเคอร์เซอร์

7. เปลี่ยนโปรแกรมจาก MOVLW B'00001100 เป็น MOVLW B'00001110 ทำการคอมไพล์และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้ง บันทึกผลการทดลอง

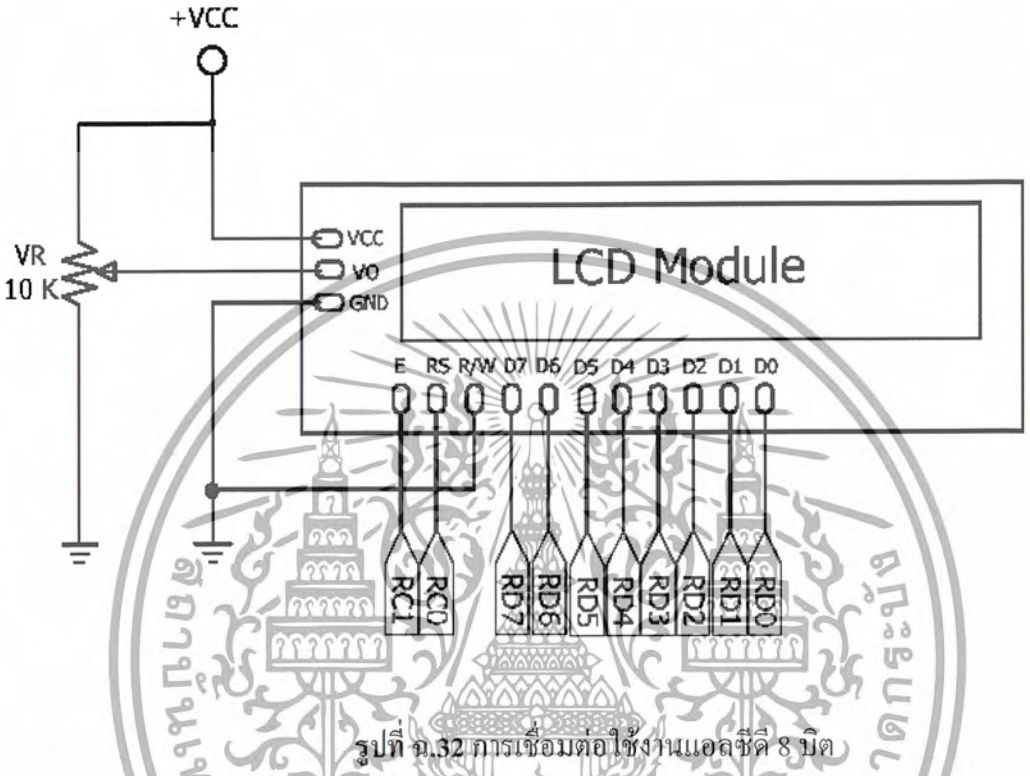
ทันทีที่รันโปรแกรมจะปรากฏข้อความและข้อความและแสดงเคอร์เซอร์

8. เปลี่ยนโปรแกรมจาก MOVLW B'00001111 จากการทดลองข้อ 7 ทำการคอมไพล์และโปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์อีกครั้ง สังเกตผลที่เกิดขึ้น บันทึกผลการทดลอง

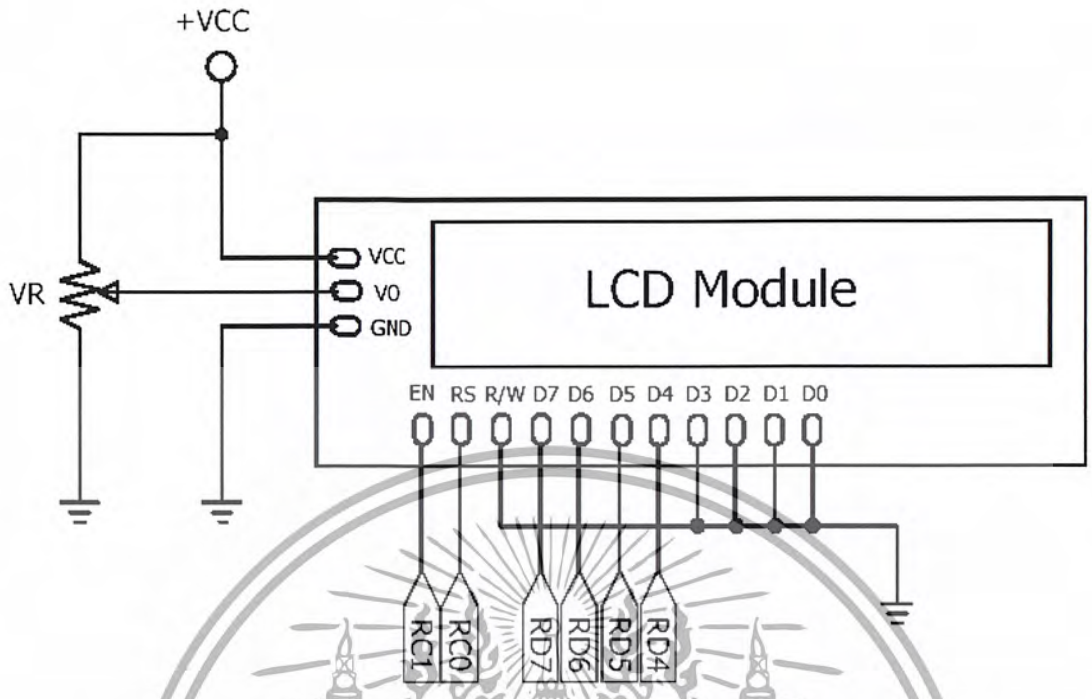
ทันทีที่รันโปรแกรมจะปรากฏข้อความและข้อความจะกระพริบติดดับเป็นจังหวะ

คำถามท้ายการทดลอง

1. จงเขียนวงจรแสดงการเชื่อมต่อแอลซีดีแบบ 8 บิต และแบบ 4 บิต



รูปที่ ฉ.32 การเชื่อมต่อใช้งานแอลซีดี 8 บิต



รูปที่ ฉ.32 การเชื่อมต่อใช้งานแอลซีดี 4 บิต

2. จากการทดลองเมื่อต้องการให้แอลซีดีแสดงผลเป็น LCD TEST และกระพริบจะนั้นเขียนโปรแกรมอย่างไร หรือแทรกโปรแกรมที่ตำแหน่งไหนจงอธิบาย

เปลี่ยนโปรแกรมจาก LCD TEST เป็นข้อมูลที่ต้องการและเปลี่ยนโปรแกรมจาก MOVLW '00001100' เป็น MOVLW '00001111'

3. ในการต่อแอลซีดีแบบ 8 บิต กับ 4 บิต แตกต่างกันอย่างไรและมีวิธีในการเขียนโปรแกรมส่งข้อมูลอย่างไรจงอธิบาย

แตกต่างกันตรงที่การต่อแบบ 4 บิตนั้นจะใช้สายสัญญาณข้อมูล (Data) เพียง 4 เส้นส่วนการต่อแบบ 8 บิต จะใช้สายสัญญาณ 8 เส้น และในเรื่องของการเขียนโปรแกรมนั้นการต่อแบบ 4 บิตจะต้องทำการส่งข้อมูลให้กับแอลซีดีถึง 2 ครั้งทำให้การเขียนโปรแกรมยาวขึ้น

เฉลยใบงานที่ 13
การสื่อสารข้อมูลกับคอมพิวเตอร์ผ่านพอร์ตอนุกรม RS-232

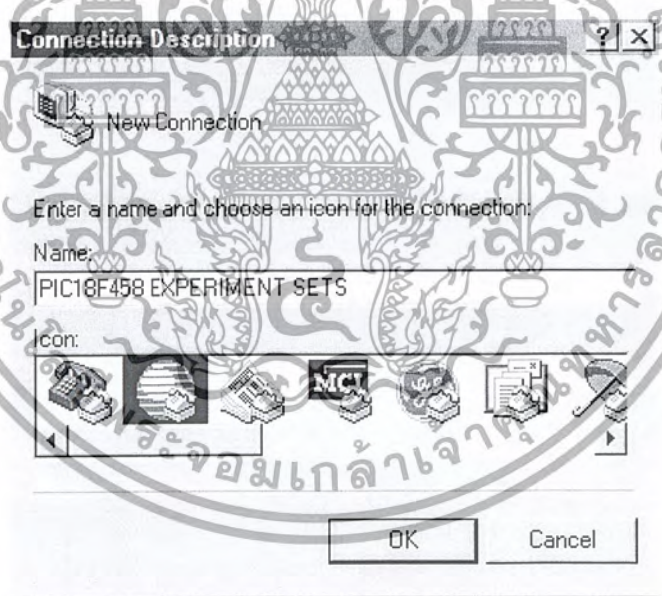
ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.31 ทำการคอมไพล์ให้เป็น Hex file

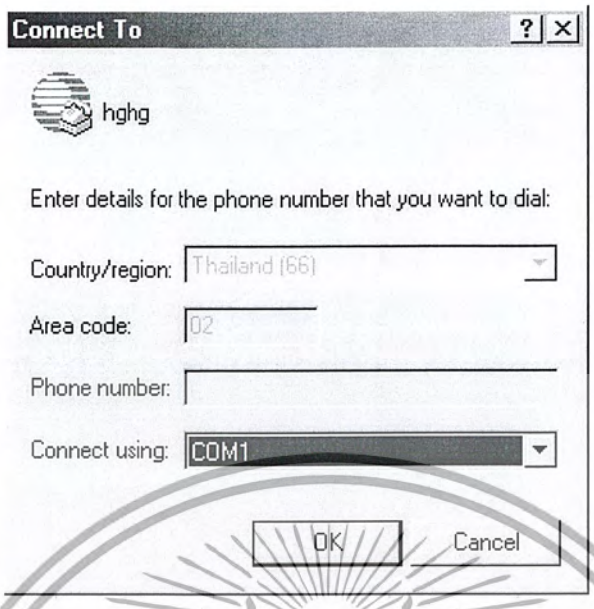
2. ต่อวงจรตามรูปที่ จ.30

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

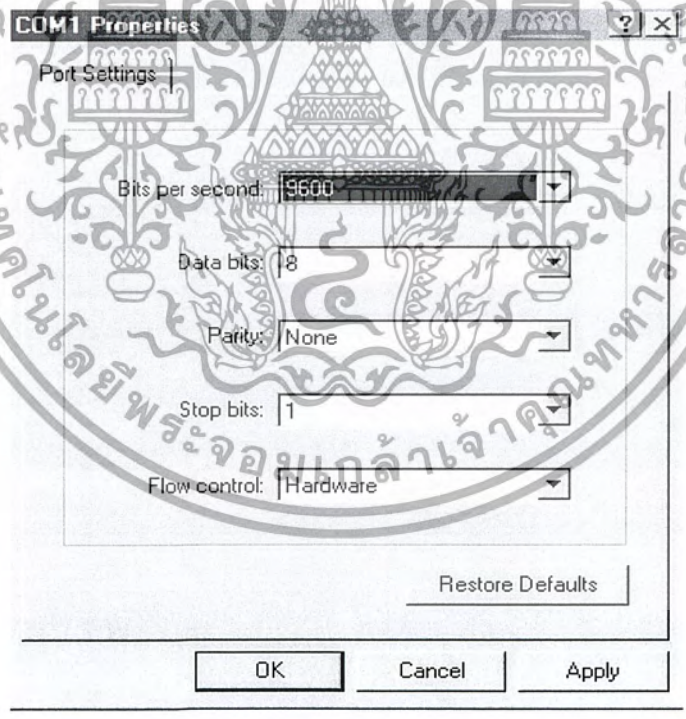
4. เปิดโปรแกรม Hyper Terminal ดังรูปที่ ฉ. 33 และ ฉ.34 ใส่ข้อมูลในการเชื่อมต่อเลือกไอคอนแล้วเลือกพอร์ตที่ใช้เชื่อมต่อระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ โดยส่วนใหญ่จะเป็น COM1 และ COM2



รูปที่ ฉ.33 การกำหนดค่าให้กับโปรแกรม Hyper Terminal



รูปที่ ฉ.34 การเลือกพอร์ตสำหรับต่อใช้งานพอร์ตอนุกรม



รูปที่ ฉ.35 การกำหนดค่าให้กับโปรแกรม Hyper Terminal เพื่อรับข้อมูล

5. กำหนดค่าให้กับโปรแกรม Hyper Terminal ดังรูปที่ ฉ.35 โดยเลือกบอดเรตเท่ากับ 9,600 จำนวนบิตข้อมูลเท่ากับ 8 บิต ค่าพาราตี้กำหนดเท่ากับ None จากนั้นกดปุ่ม OK

6. จากการทดลองข้อที่ 6 สังเกตผลที่เกิดขึ้นกับหน้าต่างโปรแกรม Hyper Terminal

เมื่อรันโปรแกรมจะปรากฏข้อความที่หน้าต่าง โปรแกรม Hyper Terminal เป็น PIC18F458 EXPERIMENT SETS

คำถามท้ายการทดลอง

1. จงบอกรีจิสเตอร์ที่เกี่ยวข้องกับการกับการโมดูลสื่อสารข้อมูลอนุกรมพร้อมทั้งอธิบาย

TXREG ใช้สำหรับเก็บค่าที่จะส่งออกไปยังพอร์ตอนุกรม

TXSTA ใช้กำหนดค่ามาตราฐานในการส่งข้อมูล

SPBRG ใช้สำหรับการสร้างบอดเรต

RCREG ใช้สำหรับเก็บข้อมูลที่อ่านได้จากภาครับของพอร์ตอนุกรม

RCSTA ใช้กำหนดค่ามาตราฐานในการรับข้อมูล

2. ในการกำหนดค่าอัตราบอดเรตเพื่อสื่อสารข้อมูลนั้นมีวิธีการอย่างไรในการเขียนโปรแกรมจงอธิบาย

กำหนดค่าบอดเรตโดยป้อนค่าเข้าที่รีจิสเตอร์ SPBRG และเลือกโหมดของบอดเรตเป็นโหมดความเร็วต่ำหรือสูงโดยกำหนดที่บิต BRGH ของรีจิสเตอร์ TXSTA สูตรในการหาค่าคือ

ค่าบอดเรต = FOSC/64 (X+1)

เฉลยใบงานที่ 14

การทดลองอ่านและเขียนหน่วยความจำโปรแกรมแบบแฟลชของ PIC18F458

ลำดับขั้นการทดลอง

1. จากผังโปรแกรมที่ได้ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.36 ทำการคอมไพล์ให้เป็น Hex File

2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

3. ต่อวงจรตามรูปที่ จ.35

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่เกิดขึ้นกับโมดูลแสดงผลแอลอีดีแอลอีดีที่ต่ออยู่กับ PORTB บิต RB0 แจ้งให้ทราบว่าเขียนข้อมูลเสร็จแล้ว

5. เปิดโปรแกรม Epic Win ไปที่เมนู View เลือก Data สังเกตผลจากโปรแกรมบันทึกผลการทดลอง

ค่าของข้อมูลในแอดเดรส 0x1FF8 = 0440

6. ทดลองเปลี่ยนโปรแกรมเพื่อกำหนดแอดเดรสให้กับ EEADRH และ EEADRL แล้วทำตามการทดลองในข้อ 1- 5 ใหม่อีกครั้ง

7. เขียนโปรแกรมตามรูปที่ จ.37 ทำการคอมไพล์ให้เป็น Hex File

8. ต่อวงจรตามรูปที่ จ.35

9. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่เกิดขึ้นกับโมดูลแสดงผลแอลอีดีที่ต่ออยู่กับ PORTC และ PORTB

สังเกตผลที่แอลอีดีที่ต่ออยู่กับพอร์ต C แปลงเป็นเลขฐานสิบหก = 01 0110

สังเกตผลที่แอลอีดีที่ต่ออยู่กับพอร์ต D แปลงเป็นเลขฐานสิบหก = 0111 1000

10. ทดลองเปลี่ยนค่าข้อมูลและแอดเดรสที่คำสั่ง FILL แล้วทำการทดลองใหม่รันโปรแกรมสังเกตผลที่กับโมดูลแสดงผลแอลอีดีบันทึกผลการทดลอง

ถ้าไม่มีการผิดพลาดจะปรากฏข้อมูลที่นำไปเก็บไว้ในแอดเดรสดังกล่าว

คำถามท้ายการทดลอง

1. จงอธิบายหลักการเขียนและอ่านข้อมูลจากหน่วยความจำโปรแกรมแบบแฟลชของไมโครคอนโทรลเลอร์ PIC18F458

การอ่านและเขียนข้อมูลของ PIC18F458 นั้นมีวิธีการคล้ายกับหน่วยความจำข้อมูลอีอีพรอมโดยการเขียนข้อมูลนั้นต้องกำหนดค่าแอดเดรสก่อนส่วนที่จะอ่านก็เช่นกัน

2. จงอธิบายความแตกต่างของหน่วยความจำอีอีพรอมและหน่วยความจำโปรแกรม

ความแตกต่างคือหน่วยความจำโปรแกรมนั้นเป็นที่รองรับหรือเก็บค่าของโปรแกรมที่เราเขียนขึ้นส่วนหน่วยความจำอีอีพรอมนั้นเป็นหน่วยความจำพิเศษที่เพิ่มเติมเข้ามาภายในไมโครคอนโทรลเลอร์ PIC18F458 เป็นหน่วยความจำเอนกประสงค์ใช้งานทั่งไป

เฉลยใบงานที่ 15
การทดลองมินิเมโลดี้

ลำดับขั้นการทดลอง

1. จากผังการทำงานที่ได้ออกแบบไว้เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.39 ทำการคอมไพล์ให้เป็น Hex File

2. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

3. ต่อวงจรตามรูปที่ จ.38

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นบันทึกผลที่ได้จากการทดลอง

จะได้ยินเสียงลำโพงดังที่อย่างต่อเนื่อง

5. ทดลองกดสวิตซ์ที่ต่ออยู่กับพอร์ต B ตำแหน่ง S0-S2 สังเกตผลที่เกิดขึ้นบันทึกผลการทดลอง

เมื่อกดสวิตช์ S0-S2 จะทำให้ได้ยินเสียงที่ระดับต่างๆ กัน

คำถามท้ายการทดลอง

1. เมื่อต้องการเปลี่ยนการรับค่าจากสวิตช์จากที่เชื่อมต่อกับพอร์ต A ที่บิต RA0, RA1, และ RA2 แก้โปรแกรมที่ตำแหน่งไหนบ้างจงอธิบาย

กำหนดให้พอร์ต A ทำงานในโหมดอินพุตดิจิตอลก่อน โดยการเขียนค่า 0x07 หรือ 0x06 ไปยัง ADCON1 จากนั้นกำหนดพอร์ต A เป็น MOVLW 0x0F ให้ RA3 : RA0 เป็น อินพุตรับค่าจากสวิตช์และเปลี่ยนค่า ในบรรทัดที่ต่อจากคำสั่ง BTFSS จากที่เป็นพอร์ต B จาก RB0 : RB2 มาเป็น RA3 : RA0

2. เมื่อต้องการเปลี่ยนระดับเสียงให้ต่างไปจากนี้จะแก้โปรแกรมที่ใด

เปลี่ยนค่าให้กับตัวแปรหน่วงเวลาเป็นค่าต่างๆ

เฉลยใบงานที่ 16

การทดลองใช้งานโหมดสลีปของ PIC18F458

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.45 ทำการคอมไพล์ให้เป็น Hex File

2. ต่อวงจรตามรูปที่ จ.44

3 . โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์

4. รันโปรแกรมสังเกตผลการทดลองที่เกิดขึ้นกับแอลอีดีที่ต่ออยู่กับ PORTD บันทึกผลที่ได้จากการทดลอง

5. ปลดแหล่งจ่ายไฟออกจากวงจรจากนั้นป้อนแหล่งจ่ายไฟเข้าอีกครั้ง (เพื่อให้เกิดเพาเวอร์ออนรีเซต) สังเกตการทำงานของ LED0 และ LED1 หลังจากที่ LED12 – LED15 หยุดกระพริบ

LED0 ติดหรือดับ ติด

LED1 ติดหรือดับ ดับ

6. หลังจากนั้นกดสวิตช์ S0 สังเกตการเปลี่ยนแปลงที่เกิดขึ้นที่ LED0 – LED1

LED0 ติดหรือดับ ติด

LED1 ติดหรือดับ ดับ

7. กดสวิตช์ S0 มีผลอย่างไรต่อการทำงานของวงจร

ทำให้เกิดการอินเตอร์รัพต์ซึ่งจะมีผลทำให้เกิดการ Wake Up

8. ทำการโปรแกรม PIC18F458 ใหม่อีกครั้ง โดยครั้งนี้กำหนดให้วอตช์ดอกไทเมอร์ ON เพื่อสังเกตการเปลี่ยนแปลงที่บิต TO

9. ทำการรันโปรแกรม โดยการเลื่อนไปที่สวิตช์ RUN

10. สังเกตการทำงานของ LED0 และ LED1 แล้วบันทึกผลที่เกิดขึ้นกับ LED12- LED15

12-15 กระพริบ แล้วติดตลอดหลังจากนั้นจะติดเฉพาะ 12 และ 15

11. จากการทดลอง PIC18F458 เข้าสู่โหมดสลีปเมื่อใด และเวกอัปได้อย่างไร

เข้าสู่โหมดสลีปเมื่อกระทำคำสั่ง sleep และเวกอัปเมื่อ วอตช์ด็อกไทเมอร์เกิดไทม์เอาต์

คำถามท้ายการทดลอง

1. จงให้คำจำกัดความของโหมดสลีป

สำหรับคำสั่งที่ทำให้ไมโครคอนโทรลเลอร์ทำงานในโหมดประหยัดพลังงานหรือโหมดสลีป (Sleep Mode) นั้นคือ คำสั่ง sleep โดยวงจรกำเนิดสัญญาณนาฬิกาจะหยุดทำงาน พอร์ตเอาต์พุตที่ถูกสั่งให้ทำงานก่อนหน้าคำสั่ง sleep จะยังคงทำงานต่อไป ไม่ว่าจะถูกขับด้วยลอจิก "1", "0" หรือเป็นอิมพิแดนซ์สูงก็ตาม

2. การเวกอัปคืออะไร เกี่ยวข้องอย่างไรกับการทำงานในโหมดสลีปของ PIC18F458

การที่ตื่นขึ้นมาจากการประหยัดพลังงานซึ่งสามารถทำได้ 3 วิธี

- การเกิด Reset
- การ Timeout ของ Watch Dog
- การเกิดอินเตอร์รัพต์

3. จงอธิบายความสัมพันธ์ระหว่างบิต TO และ PD กับการทำงานในโหมดสลีปของ PIC18F458

ถ้าบิต PD เซตหมายความว่า PIC18F458 เวกอัปเนื่องจากเกิดเพาเวอร์ออนรีเซต ถ้าบิต TO เป็น "0" แสดงว่า PIC18F458 เกิดการเวกอัปเนื่องจากวอชด็อกไทเมอร์เกิดไทม์เอาต์ กรณีสุดท้าย ถ้าบิตPD เป็น "0" และบิต TO เป็น "1" หมายความว่า PIC18F458 เกิดการเวกอัปเนื่องจากการอินเตอร์รัพต์

4. รีจีสเตอร์ที่เกี่ยวข้องกับการทำงานในโหมดสลีปมีกี่ตัว อะไรบ้าง และเกี่ยวข้องอย่างไร

-PD ใน RCON ใช้ในการตรวจว่าเกิดการ WAKEUP จาก POWER ON RESET หรือไม่

-TO ใน RCON ใช้ตรวจร่วมกับ PD ว่าเกิดการ WAKE UP จาก INTERRUPT หรือ WATCHDOG TIMER

-INTE ใน INTCON ใช้ในการเปิดหรือปิดการใช้งาน INTERRUPT เพื่อใช้ในการ WAKEUP จาก SLEEP MODE

5. จงยกตัวอย่างประโยชน์ของการทำงานในโหมดสลีปของ PIC18F458 มาให้เห็นอย่างชัดเจนอย่างน้อย 1 ตัวอย่าง

ทำให้สูญเสียพลังงานที่ใช้งานน้อยมาก

เฉลยใบงานที่17
การทดลองใช้งานโมดูลเปรียบเทียบแรงดันแอนะลอกของ PIC18F458

จุดประสงค์

1. เพื่อให้สามารถอธิบายหลักการเปรียบเทียบแรงดันแอนะลอกของ PIC18F458 ได้
2. เพื่อให้สามารถเขียนโปรแกรมใช้งานเปรียบเทียบแรงดันแอนะลอกของ PIC18F458 ได้
3. เพื่อให้สามารถแก้ไขโปรแกรมในการทดลองได้

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม MPLAB ทำการเขียนโปรแกรมตามรูปที่ จ.47 เสร็จแล้วทำการคอมไพล์ให้เป็น Hex File

2. ต่อสายเชื่อมต่อระหว่างโมดูลตามรูปที่ จ.46

3. โปรแกรมข้อมูลลงบนตัวไมโครคอนโทรลเลอร์ทำการปรับค่าความต้านทานที่ต่ออยู่กับ RA1 ให้เท่ากับ 2 โวลต์ โดย สังเกตการเปลี่ยนแปลงที่เกิด ขึ้นวลต์มิเตอร์ต่อวัดแรงดันจากนั้นปรับค่าแรงดันที่ต้านทานที่ต่ออยู่กับ RA1 ให้เท่ากับ 0 โวลต์ สังเกตผลที่เกิดขึ้นกับแอลอีดี

5. จากข้อ 4 ผลการทดลองเป็นดังนี้

เมื่อรันโปรแกรมผลคือแอลอีดีจะไม่ติดสว่างเนื่องจากได้กำหนดให้โมดูลเปรียบเทียบแรงดันทำงานในลักษณะไม่กลับลอจิก คือ เมื่อแรงดันที่ขา RA2 น้อยกว่าขา RA1 ก็จะทำให้เอาต์พุตเป็น 0

ทำการปรับค่าตัวต้านทานที่ต่ออยู่กับขา RA2 ขึ้นช้าๆ พร้อมทั้งใช้โวลต์มิเตอร์ต่อวัด แรงดันจนกระทั่งมีค่า 2 โวลต์ สังเกตผลที่เกิดขั้นกับแอลอีดี

7. จากข้อ 6 ผลการทดลองเป็นดังนี้

ทันทีที่รันโปรแกรมจะเห็นได้ว่าแอลอีดีติดแต่จะไม่สว่างมากนักเนื่องจากว่าโมดูลเปรียบเทียบแรงดันแอนะลอกยังไม่ให้ค่าลอจิกที่ถูกต้อง

8. ทดลองปรับค่าความต้านทานที่ต่ออยู่กับ RA2 /AN2 ให้มีค่ามากกว่า 2 โวลต์สังเกตผลที่เกิดขึ้นกับโมดูลแอลอีดี

9. จากข้อ 8 ผลการทดลองเป็นดังนี้

แอลอีดีติดสว่างเต็มที่เนื่องจากว่าโมดูลเปรียบเทียบแรงดันแอนะลอกให้ค่าลอจิกที่ถูกต้องแล้ว

10. ทดลองนำเครื่องหมายหน้า INVERT ออกทำการคอมไพล์และโปรแกรมข้อมูลอีกครั้งและทำตามขั้นตอนในข้อ 4-9 อีกครั้ง สังเกตผลที่เกิดขึ้นกับโมดูลแอลอีดี

11. จากข้อ 10 ผลการทดลองเป็นดังนี้

แอลอีดีติดสว่างเมื่อแรงดันที่ขา RA2 น้อยกว่า RA1 คือเป็นลักษณะกลับลอจิกแล้วนั่นเอง

คำถามท้ายการทดลอง

1. จงบอกริสเตอร์ที่เกี่ยวข้องกับโมดูลเปรียบเทียบแรงดันแอนะลอกของ PIC18F458 พร้อมอธิบายหน้าที่และการทำงาน

CMCON รีจีสเตอร์หน้าที่คือควบคุมการทำงานของโมดูลเปรียบเทียบแรงดันแอนะลอกใช้กำหนดและควบคุมการทำงานของโมดูลเปรียบเทียบแรงดันแอนะลอก

2. เมื่อต้องการรีเซต โมดูลเปรียบเทียบแรงดันแอนะลอกนั้นจะมีวิธีการเขียนโปรแกรมอย่างไรหรือแก้โปรแกรมที่ตำแหน่งไหน

เขียนค่าน้อมูลไปยังรีจิสเตอร์ CMCON บิต CM2:CM0 เป็น 000

ภาคผนวก ช

รายละเอียดและคุณสมบัติของอุปกรณ์

MICROCHIP

PIC18FXX8
Data Sheet

28/40-Pin High Performance, Enhanced FLASH Microcontrollers with CAN

MICROCHIP

PIC18FXX8

28/40-Pin High Performance, Enhanced FLASH Microcontrollers with CAN

High Performance RISC CPU:

- Linear program memory addressing up to 2 Mbytes
- Linear data memory addressing to 4 Kbytes
- Up to 10 MIPs operation
- DC - 40 MHz clock input
- 4 MHz - 10 MHz osc./clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 8 x 8 Single Cycle Hardware Multiplier

Peripheral Features:

- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time-base for PWM)
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option - Timer1/Timer3
- Capture/Compare/PWM (CCP) modules CCP pins can be configured as:
  - Capture input: 16-bit, max resolution 6.25 ns
  - Compare: 16-bit, max resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1- to 10-bit. Max. PWM freq. @: 8-bit resolution = 156 kHz, 10-bit resolution = 39 kHz
- Enhanced CCP module which has all the features of the standard CCP module, but also has the following features for advanced motor control:
  - 1, 2, or 4 PWM outputs
  - Selectable PWM polarity
  - Programmable PWM dead-time
- Master Synchronous Serial Port (MSSP) with two modes of operation:
  - 3-wire SPI™ (Supports all 4 SPI modes)
  - I^2C™ Master and Slave mode
- Addressable USART module:
  - Supports Interrupt on Address bit

Advanced Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter module (A/D) with:
  - Conversion available during SLEEP
  - Up to 8 channels available
- Analog Comparator Module:
  - Programmable input and output multiplexing
- Comparator Voltage Reference Module
- Programmable Low Voltage Detection (LVD) module
  - Supports interrupt on low voltage detection
- Programmable Brown-out Reset (BOR)

CAN bus Module Features:

- Complies with ISO CAN Conformance Test
- Message bit rates up to 1 Mbps
- Conforms to CAN 2.0B ACTIVE Spec with:
  - 29-bit Identifier Fields
  - 8-byte message length
  - 3 Transmit Message Buffers with prioritization
  - 2 Receive Message Buffers
  - 6 full 29-bit Acceptance Filters
  - Prioritization of Acceptance Filters
  - Multiple Receive Buffers for High Priority Messages to prevent loss due to overflow
  - Advanced Error Management Features

Special Microcontroller Features:

- Power-on Reset (POR), Power-up Timer (PWRT), and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options, including:
  - 4X Phase Lock Loop (of primary oscillator)
  - Secondary Oscillator (32 kHz) clock input
- In-Circuit Serial Programming™ (ICSP™) via two pins

FLASH Technology:

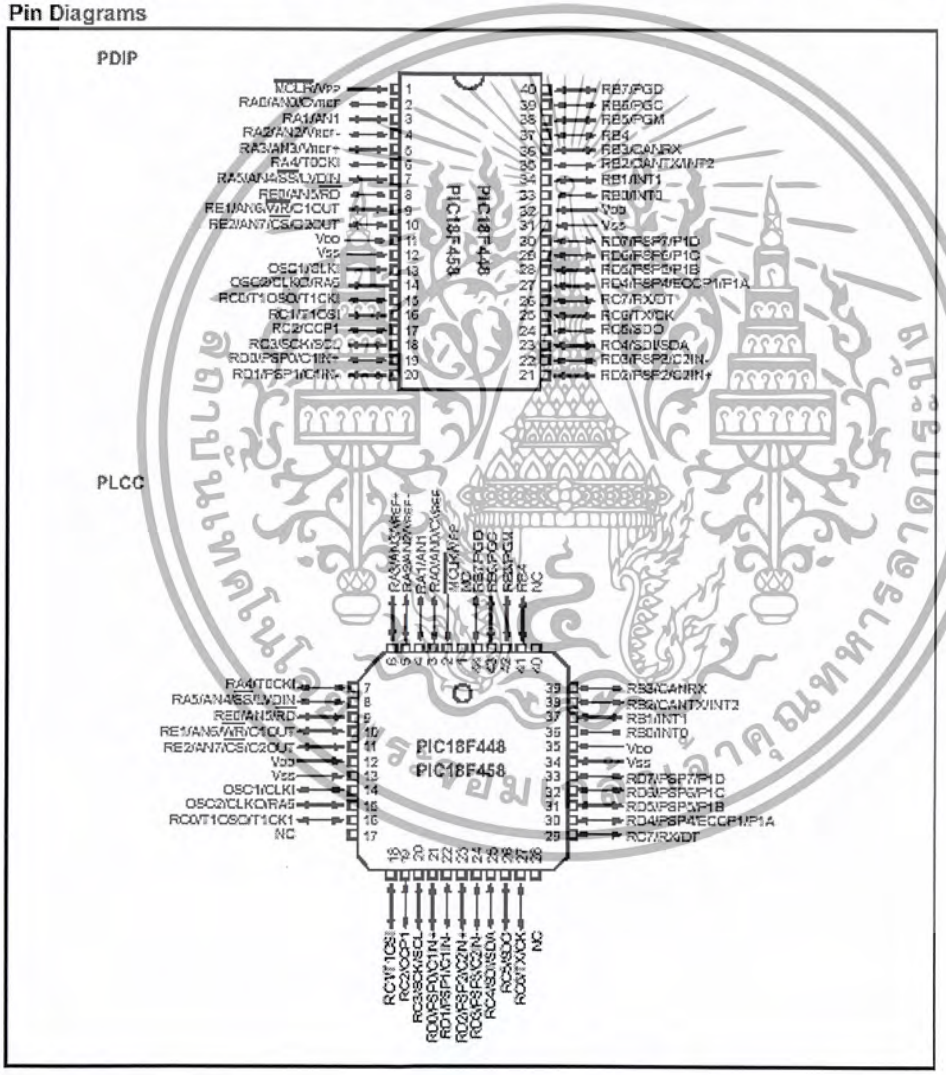
- Low power, high speed Enhanced FLASH technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges

PIC18FXX8

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	Comparators	CCP/ ECCP (PWM)	MSSP		USART	Timers 8/16-bit
	FLASH (bytes)	# Single Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I^2C		
PIC18F248	16K	8192	768	256	22	5	—	1/0	Y	Y	Y	1/3
PIC18F258	32K	16384	1536	256	22	5	—	1/0	Y	Y	Y	1/3
PIC18F448	16K	8192	768	256	33	8	2	1/1	Y	Y	Y	1/3
PIC18F458	32K	16384	1536	256	33	8	2	1/1	Y	Y	Y	1/3

Pin Diagrams

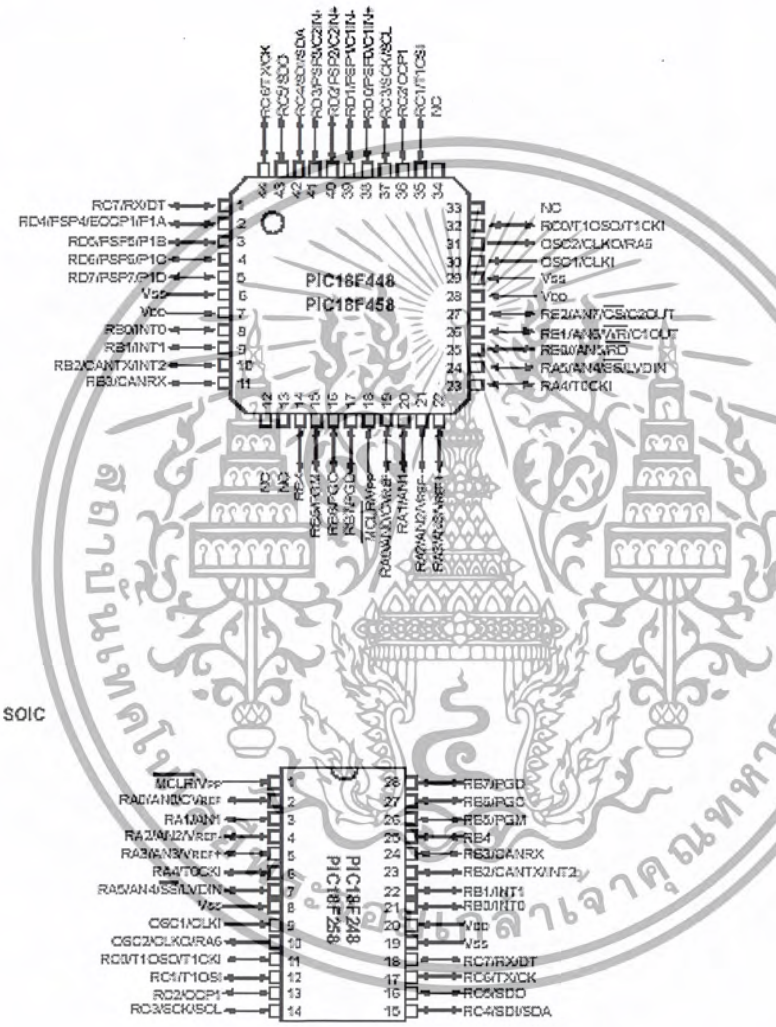
PDIP



PLCC

Pin Diagrams (Continued)

TQFP



SPDIP, SOIC

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F248
- PIC18F258
- PIC18F448
- PIC18F458

These devices are available in 28-pin, 40-pin and 44-pin packages. They are differentiated from each other in four ways:

1. PIC18FX58 devices have twice the FLASH program memory and data RAM of PIC18FX48 devices (32 Kbytes and 1536 bytes vs. 16 Kbytes and 768 bytes, respectively).
2. PIC18F2X8 devices implement 5 A/D channels, as opposed to 8 for PIC18F4X8 devices.
3. PIC18F2X8 devices implement 3 I/O ports, while PIC18F4X8 devices implement 5.
4. Only PIC18F4X8 devices implement the Enhanced CCP module, analog comparators and the Parallel Slave Port.

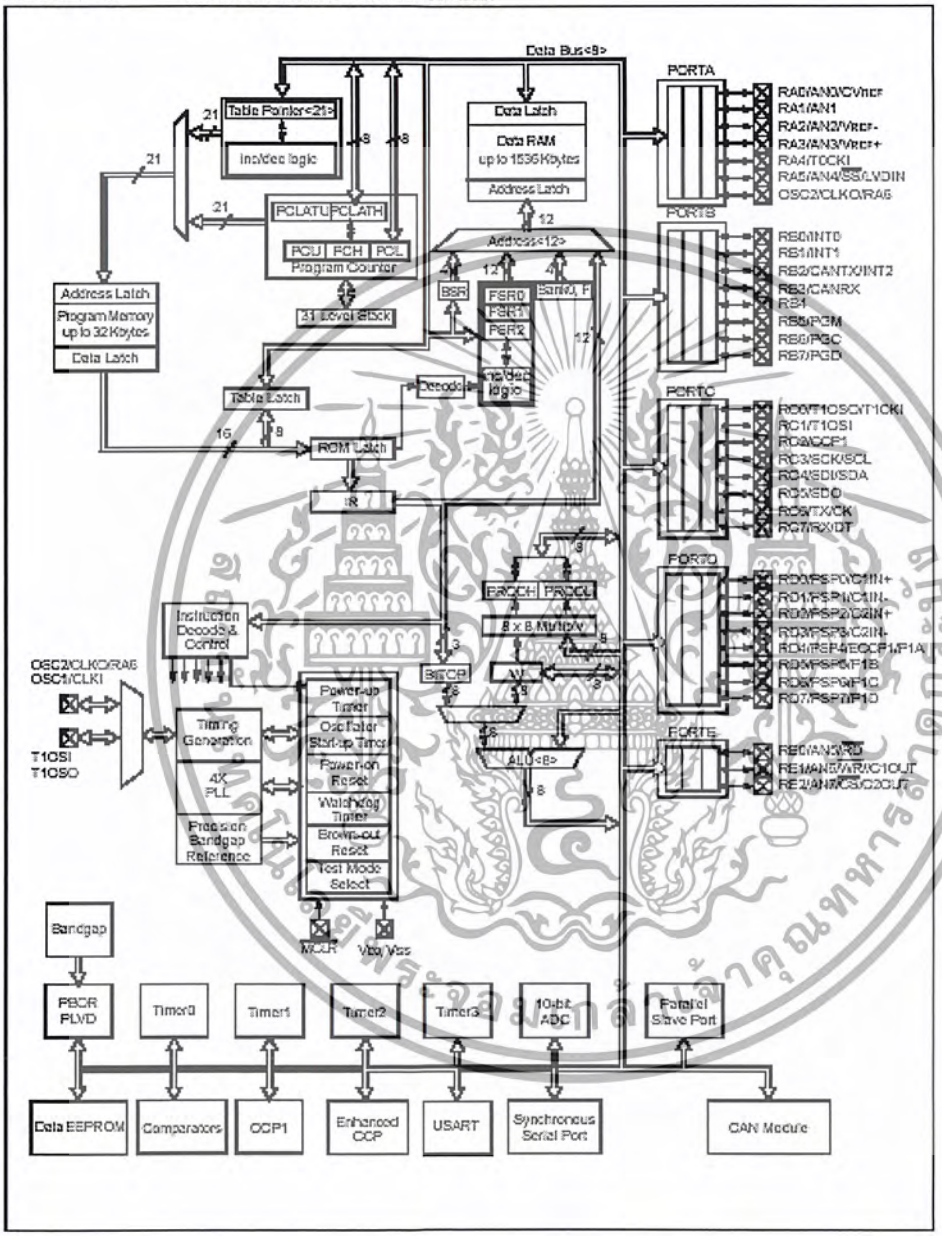
All other features for devices in the PIC18FXX8 family, including the serial communications modules, are identical. These are summarized in Table 1-1.

Block diagrams of the PIC18F2X8 and PIC18F4X8 devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2.

TABLE 1-1: PIC18FXX8 DEVICE FEATURES

Features		PIC18F248	PIC18F258	PIC18F448	PIC18F458
Operating Frequency		DC - 40 MHz	DC - 40 MHz	DC - 40 MHz	DC - 40 MHz
Internal Program Memory	Bytes	16K	32K	16K	32K
	# of Single Word Instructions	8192	16384	8192	16384
Data Memory (Bytes)		768	1536	768	1536
Data EEPROM Memory (Bytes)		256	256	256	256
Interrupt Sources		17	17	21	21
I/O Ports		Ports A, B, C	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers		4	4	4	4
Capture/Compare/PWM Modules		1	1	1	1
Enhanced Capture/Compare/PWM Modules		—	—	1	1
Serial Communications		MSSP, CAN, Addressable USART	MSSP, CAN, Addressable USART	MSSP, CAN, Addressable USART	MSSP, CAN, Addressable USART
Parallel Communications (PSP)		No	No	Yes	Yes
10-bit Analog-to-Digital Converter		5 input channels	5 input channels	8 input channels	8 input channels
Analog Comparators		No	No	2	2
Analog Comparators V_{REF} Output		N/A	N/A	Yes	Yes
RESETS (and Delays)		POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low Voltage Detect		Yes	Yes	Yes	Yes
Programmable Brown-out Reset		Yes	Yes	Yes	Yes
CAN Module		Yes	Yes	Yes	Yes
In-Circuit Serial Programming™ (ICSP™)		Yes	Yes	Yes	Yes
Instruction Set		75 Instructions	75 Instructions	75 Instructions	75 Instructions
Packages		28-pin SPDIP 28-pin SOIC	28-pin SPDIP 28-pin SOIC	40-pin PDIP 44-pin PLCC 44-pin TQFP	40-pin PDIP 44-pin PLCC 44-pin TQFP

FIGURE 1-2: PIC18F448/458 BLOCK DIAGRAM



PIC18FXX8

TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18F248/258	PIC18F448/458					
	SPDIP, SOIC	PDIP	TQFP	PLCC			
MCLR/VPP	1	1	18	2			Master Clear (input) or programming voltage (output).
MCLR					I	ST	Master Clear (Reset) input. This pin is an active low RESET to the device.
VPP					P	—	Programming voltage input.
NC	—	—	12, 13, 33, 34	1, 17, 28, 40	—	—	These pins should be left unconnected.
OSC1/CLKI	9	13	30	14			Oscillator crystal or external clock input.
OSC1					I	CMOS/ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode. Otherwise CMOS.
CLKI					I	CMOS	External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKO/RA6	10	14	31	15			Oscillator crystal or clock output.
OSC2					O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO					O	—	In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6					I/O	TTL	General purpose I/O pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open Drain (no P diode to VDD)

TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18F248/258	PIC18F448/458					
	SPDIP, SOIC	PDIP	TQFP	PLCC			
							PORTA is a bi-directional I/O port.
RA0/AN0/CVREF	2	2	19	3			
RA0					I/O	TTL	Digital I/O.
AN0					I	Analog	Analog input 0.
CVREF					O	Analog	Comparator voltage reference output.
RA1/AN1	3	3	20	4			
RA1					I/O	TTL	Digital I/O.
AN1					I	Analog	Analog input 1.
RA2/AN2/VREF-	4	4	21	5			
RA2					I/O	TTL	Digital I/O.
AN2					I	Analog	Analog input 2.
VREF-					I	Analog	A/D reference voltage (Low) input.
RA3/AN3/VREF+	5	5	22	6			
RA3					I/O	TTL	Digital I/O.
AN3					I	Analog	Analog input 3.
VREF+					I	Analog	A/D reference voltage (High) input.
RA4/T0CKI	6	6	23	7			
RA4					I/O	TTL/OD	Digital I/O - open drain when configured as output.
T0CKI					I	ST	Timer0 external clock input.
RA5/AN4/$\overline{SS}$/LVDIN	7	7	24	8			
RA5					I/O	TTL	Digital I/O.
AN4					I	Analog	Analog input 4.
$\overline{SS}$					I	ST	SPI slave select input.
LVDIN					I	Analog	Low voltage detect input.
RA6							See the OSC2/CLKO/RA6 pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open Drain (no P diode to VDD)

TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18F248/258	PIC18F448/458					
	SPDIP, SOIC	PDIP	TQFP	PLCC			
							PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT0	21	33	8	36			
RB0					I/O	TTL	Digital I/O.
INT0					I	ST	External interrupt 0.
RB1/INT1	22	34	9	37			
RB1					I/O	TTL	Digital I/O.
INT1					I	ST	External interrupt 1.
RB2/CANTX/INT2	23	35	10	38			
RB2					I/O	TTL	Digital I/O.
CANTX					O	TTL	Transmit signal for CAN bus.
INT2					I	ST	External interrupt 2.
RB3/CANRX	24	36	11	39			
RB3					I/O	TTL	Digital I/O.
CANRX					I	TTL	Receive signal for CAN bus.
RB4	25	37	14	41	I/O	TTL	Digital I/O. Interrupt-on-change pin.
RB5/PGM	26	38	15	42			
RB5					I/O	TTL	Digital I/O. Interrupt-on-change pin.
PGM					I	ST	Low voltage ICSP programming enable.
RB6/PGC	27	39	16	43			
RB6					I/O	TTL	Digital I/O. In-Circuit Debugger pin. Interrupt-on-change pin.
PGC					I	ST	ICSP programming clock.
RB7/PGD	28	40	17	44			
RB7					I/O	TTL	Digital I/O. In-Circuit Debugger pin. Interrupt-on-change pin.
PGD					I/O	ST	ICSP programming data.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open Drain (no P diode to VDD)

TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18F248/258	PIC18F448/458					
	SPDIP, SOIC	PDIP	TQFP	PLCC			
							PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	11	15	32	16			
RC0					I/O	ST	Digital I/O.
T1OSO					O	—	Timer1 oscillator output.
T1CKI					I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI	12	16	35	18			
RC1					I/O	ST	Digital I/O.
T1OSI					I	CMOS	Timer1 oscillator input.
RC2/CCP1	13	17	36	19			
RC2					I/O	ST	Digital I/O.
CCP1					I/O	ST	Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	14	18	37	20			
RC3					I/O	ST	Digital I/O.
SCK					I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL					I/O	ST	Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	15	23	42	25			
RC4					I/O	ST	Digital I/O.
SDI					I	ST	SPI data in.
SDA					I/O	ST	I^2C data I/O.
RC5/SDO	16	24	43	26			
RC5					I/O	ST	Digital I/O.
SDO					O	—	SPI data out.
RC6/TX/CK	17	25	44	27			
RC6					I/O	ST	Digital I/O.
TX					O	—	USART asynchronous transmit.
CK					I/O	ST	USART synchronous clock (see RX/DT).
RC7/RX/DT	18	26	1	29			
RC7					I/O	ST	Digital I/O.
RX					I	ST	USART asynchronous receive.
DT					I/O	ST	USART synchronous data (see TX/CK).

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open Drain (no P diode to VDD)

PIC18FXX8

TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18F248/258	PIC18F448/458					
	SPDIP, SOIC	PDIP	TQFP	PLCC			
							PORTD is a bi-directional I/O port. These pins have TTL input buffers when external memory is enabled.
RD0/PSP0/C1IN+	—	19	38	21			
RD0					I/O	ST	Digital I/O.
PSP0					I/O	TTL	Parallel slave port data.
C1IN+					I	Analog	Comparator 1 input.
RD1/PSP1/C1IN-	—	20	39	22			
RD1					I/O	ST	Digital I/O.
PSP1					I/O	TTL	Parallel slave port data.
C1IN-					I	Analog	Comparator 1 input.
RD2/PSP2/C2IN+	—	21	40	23			
RD2					I/O	ST	Digital I/O.
PSP2					I/O	TTL	Parallel slave port data.
C2IN+					I	Analog	Comparator 2 input.
RD3/PSP3/C2IN-	—	22	41	24			
RD3					I/O	ST	Digital I/O.
PSP3					I/O	TTL	Parallel slave port data.
C2IN-					I	Analog	Comparator 2 input.
RD4/PSP4/ECCP1/ P1A	—	27	2	30			
RD4					I/O	ST	Digital I/O.
PSP4					I/O	TTL	Parallel slave port data.
ECCP1					I/O	ST	ECCP1 capture/compare.
P1A					O	—	ECCP1 PWM output A.
RD5/PSP5/P1B	—	28	3	31			
RD5					I/O	ST	Digital I/O.
PSP5					I/O	TTL	Parallel slave port data.
P1B					O	—	ECCP1 PWM output B.
RD6/PSP6/P1C	—	29	4	32			
RD6					I/O	ST	Digital I/O.
PSP6					I/O	TTL	Parallel slave port data.
P1C					O	—	ECCP1 PWM output C.
RD7/PSP7/P1D	—	30	5	33			
RD7					I/O	ST	Digital I/O.
PSP7					I/O	TTL	Parallel slave port data.
P1D					O	—	ECCP1 PWM output D.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open Drain (no P diode to VDD)

TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18F248/258	PIC18F448/458					
	SPDIP, SOIC	PDIP	TQFP	PLCC			
							PORTE is a bi-directional I/O port.
RE0/AN5/$\overline{RD}$	—	8	25	9			
RE0					I/O	ST	Digital I/O.
AN5					I	Analog	Analog input 5.
$\overline{RD}$					I	TTL	Read control for parallel slave port (see $\overline{WR}$ and $\overline{CS}$ pins).
RE1/AN6/$\overline{WR}$/C1OUT	—	9	26	10			
RE1					I/O	ST	Digital I/O.
AN6					I	Analog	Analog input 6.
$\overline{WR}$					I	TTL	Write control for parallel slave port (see $\overline{CS}$ and $\overline{RD}$ pins).
C1OUT					O	Analog	Comparator 1 output.
RE2/AN7/$\overline{CS}$/C2OUT	—	10	27	11			
RE2					I/O	ST	Digital I/O.
AN7					I	Analog	Analog input 7.
$\overline{CS}$					I	TTL	Chip select control for parallel slave port (see $\overline{RD}$ and $\overline{WR}$ pins).
C2OUT					O	Analog	Comparator 2 output.
VSS	19, 8	12, 31	6, 29	13, 34	—	—	Ground reference for logic and I/O pins.
VDD	20	11, 32	7, 28	12, 35	—	—	Positive supply for logic and I/O pins.

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open Drain (no P diode to VDD)

2.0 OSCILLATOR CONFIGURATIONS

2.1 Oscillator Types

The PIC18FXX8 can be operated in one of eight Oscillator modes, programmable by three configuration bits (FOSC2, FOSC1, and FOSC0).

1. LP Low Power Crystal
2. XT Crystal/Resonator
3. HS High Speed Crystal/Resonator
4. HS4 High Speed Crystal/Resonator with PLL enabled
5. RC External Resistor/Capacitor
6. RCIO External Resistor/Capacitor with I/O pin enabled
7. EC External Clock
8. ECIO External Clock with I/O pin enabled

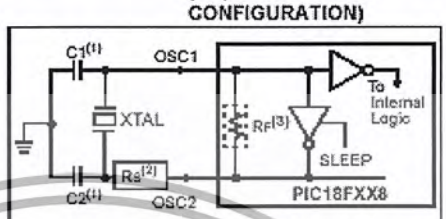
2.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HS4 (PLL) Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2-1 shows the pin connections. An external clock source may also be connected to the OSC1 pin, as shown in Figure 2-3 and Figure 2-4.

The PIC18FXX8 oscillator design requires the use of a parallel cut crystal.

Note:	Use of a series cut crystal may give a frequency out of the crystal manufacturer's specifications.

FIGURE 2-1: CRYSTAL/CERAMIC RESONATOR OPERATION (HS, XT OR LP OSC CONFIGURATION)



Note 1: See Table 2-1 and Table 2-2 for recommended values of C1 and C2.
2: A series resistor (RS) may be required for AT strip cut crystals.
3: RF varies with the crystal chosen.

TABLE 2-1: CERAMIC RESONATORS

Ranges Tested:			
Mode	Freq	OSC1	OSC2
XT	455 kHz	68 - 100 pF	68 - 100 pF
	2.0 MHz	15 - 68 pF	15 - 68 pF
	4.0 MHz	15 - 68 pF	15 - 68 pF
HS	8.0 MHz	10 - 68 pF	10 - 68 pF
	16.0 MHz	10 - 22 pF	10 - 22 pF
These values are for design guidance only. See notes following Table 2-2.			

Resonators Used:		
455 kHz	Panasonic EFO-A455K04B	± 0.3%
2.0 MHz	Murata Erie CSA2.00MG	± 0.5%
4.0 MHz	Murata Erie CSA4.00MG	± 0.5%
8.0 MHz	Murata Erie CSA8.00MT	± 0.5%
16.0 MHz	Murata Erie CSA16.00MX	± 0.5%
All resonators used did not have built-in capacitors.		

TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Osc Type	Crystal Freq	Cap. Range C1	Cap. Range C2
LP	32.0 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1.0 MHz	15 pF	15 pF
	4.0 MHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0 MHz	15-33 pF	15-33 pF
	20.0 MHz	15-33 pF	15-33 pF
	25.0 MHz	15-33 pF	15-33 pF

These values are for design guidance only. See notes on this page.

Crystals Used		
32.0 kHz	Epson C-001R32.768K-A	± 20 PPM
200 kHz	STD XTL 200.000KHz	± 20 PPM
1.0 MHz	ECS ECS-10-13-1	± 50 PPM
4.0 MHz	ECS ECS-40-20-1	± 50 PPM
8.0 MHz	EPSON CA-301 8.000M-C	± 30 PPM
20.0 MHz	EPSON CA-301 20.000M-C	± 30 PPM

Note 1: Recommended values of C1 and C2 are identical to the ranges tested (Table 2-1).

2: Higher capacitance increases the stability of the oscillator, but also increases the start-up time.

3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

4: Rs may be required in HS mode, as well as XT mode, to avoid overdriving crystals with low drive level specification.

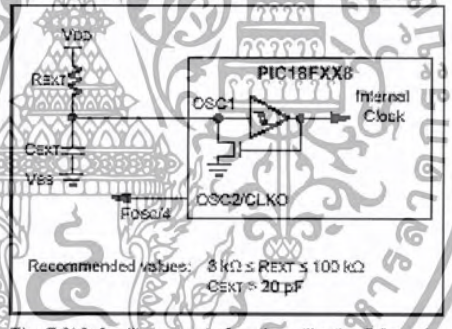
2.3 RC Oscillator

For timing insensitive applications, the "RC" and "RCIO" device options offer additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low CEXT values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 2-2 shows how the RC combination is connected.

In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic.

Note: If the oscillator frequency divided by 4 signal is not required in the application, it is recommended to use RCIO mode to save current.

FIGURE 2-2: RC OSCILLATOR MODE



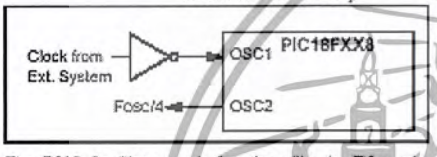
The RCIO Oscillator mode functions like the RC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

2.4 External Clock Input

The EC and ECIO Oscillator modes require an external clock source to be connected to the OSC1 pin. The feedback device between OSC1 and OSC2 is turned off in these modes to save current. There is no oscillator start-up time required after a Power-on Reset or after a recovery from SLEEP mode.

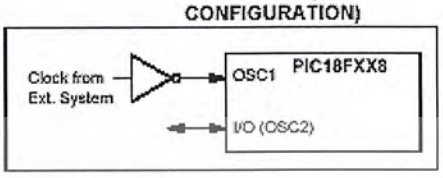
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-3 shows the pin connections for the EC Oscillator mode.

FIGURE 2-3: EXTERNAL CLOCK INPUT OPERATION (EC OSC CONFIGURATION)



The ECIO Oscillator mode functions like the EC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. Figure 2-4 shows the pin connections for the ECIO Oscillator mode.

FIGURE 2-4: EXTERNAL CLOCK INPUT OPERATION (ECIO CONFIGURATION)



2.5 HS4 (PLL)

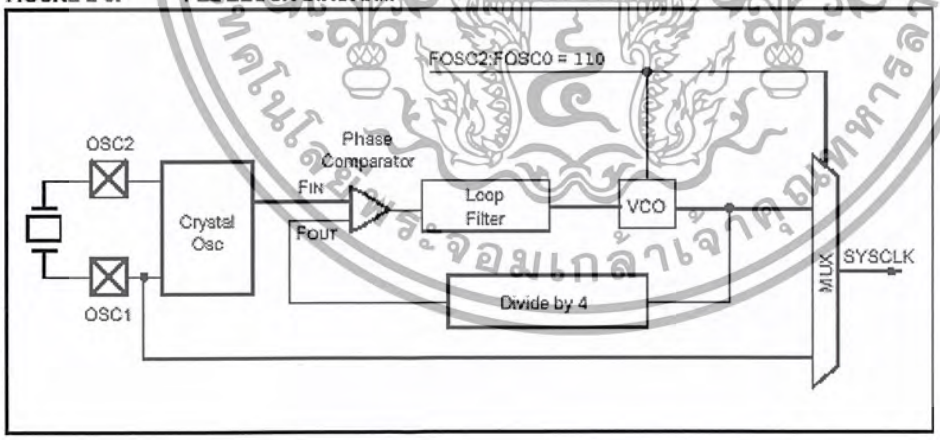
A Phase Locked Loop circuit is provided as a programmable option for users that want to multiply the frequency of the incoming crystal oscillator signal by 4. For an input clock frequency of 10 MHz, the internal clock frequency will be multiplied to 40 MHz. This is useful for customers who are concerned with EMI due to high frequency crystals.

The PLL can only be enabled when the oscillator configuration bits are programmed for HS mode. If they are programmed for any other mode, the PLL is not enabled and the system clock will come directly from OSC1.

The PLL is one of the modes of the FOSC2:FOSC0 configuration bits. The Oscillator mode is specified during device programming.

A PLL lock timer is used to ensure that the PLL has locked before device execution starts. The PLL lock timer has a time-out referred to as T_{PLL}.

FIGURE 2-5: PLL BLOCK DIAGRAM



2.6 Oscillator Switching Feature

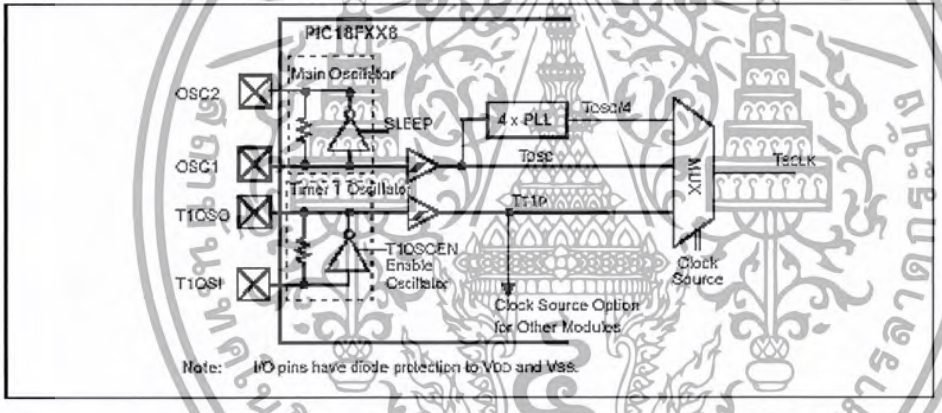
The PIC18FXX8 devices include a feature that allows the system clock source to be switched from the main oscillator to an alternate low frequency clock source. For the PIC18FXX8 devices, this alternate clock source is the Timer1 oscillator. If a low frequency crystal (32 kHz, for example) has been attached to the Timer1 oscillator pins and the Timer1 oscillator has been enabled, the device can switch to a Low Power Execution mode. Figure 2-6 shows a block diagram of the system clock sources. The clock switching feature is enabled by programming the Oscillator Switching Enable (OSCSEN) bit in Configuration register, CONFIG1H, to a '0'. Clock switching is disabled in an erased device. See Section 12.2 for further details of the Timer1 oscillator, and Section 24.1 for Configuration Register details.

2.6.1 SYSTEM CLOCK SWITCH BIT

The system clock source switching is performed under software control. The system clock switch bit, SCS (OSCCON register), controls the clock switching. When the SCS bit is '0', the system clock source comes from the main oscillator selected by the FOSC2:FOSC0 configuration bits. When the SCS bit is set, the system clock source comes from the Timer1 oscillator. The SCS bit is cleared on all forms of RESET.

Note:	The Timer1 oscillator must be enabled to switch the system clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 control register (T1CON). If the Timer1 oscillator is not enabled, any write to the SCS bit will be ignored (SCS bit forced cleared) and the main oscillator continues to be the system clock source.

FIGURE 2-6: DEVICE CLOCK SOURCES



REGISTER 2-1: OSCCON REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-1
—	—	—	—	—	—	—	SCS
bit 7							bit 0

bit 7-1 **Unimplemented:** Read as '0'

bit 0 **SCS:** System Clock Switch bit

When OSCSEN configuration bit = 0 and T1OSCEN bit is set:
1 = Switch to Timer1 oscillator/clock pin
0 = Use primary oscillator/clock input pin

When OSCSEN is clear or T1OSCEN is clear:
Bit is forced clear

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.6.2 OSCILLATOR TRANSITIONS

The PIC18FXX8 devices contain circuitry to prevent "glitches" when switching between oscillator sources. Essentially, the circuitry waits for eight rising edges of the clock source that the processor is switching to. This ensures that the new clock source is stable and that its pulse width will not be less than the shortest pulse width of the two clock sources.

Figure 2-7 shows a timing diagram indicating the transition from the main oscillator to the Timer1 oscillator. The Timer1 oscillator is assumed to be running all the time. After the SCS bit is set, the processor is frozen at the next occurring Q1 cycle. After eight synchronization cycles are counted from the Timer1 oscillator, operation resumes. No additional delays are required after the synchronization cycles.

The sequence of events that takes place when switching from the Timer1 oscillator to the main oscillator will depend on the mode of the main oscillator. In addition to eight clock cycles of the main oscillator, additional delays may take place.

If the main oscillator is configured for an external crystal (HS, XT, LP), the transition will take place after an oscillator start-up time (TOST) has occurred. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for HS, XT, and LP modes is shown in Figure 2-8.

FIGURE 2-7: TIMING DIAGRAM FOR TRANSITION FROM OSC1 TO TIMER1 OSCILLATOR

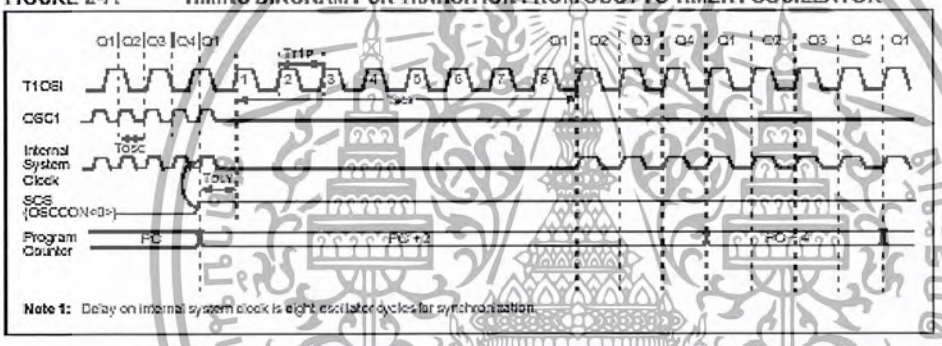
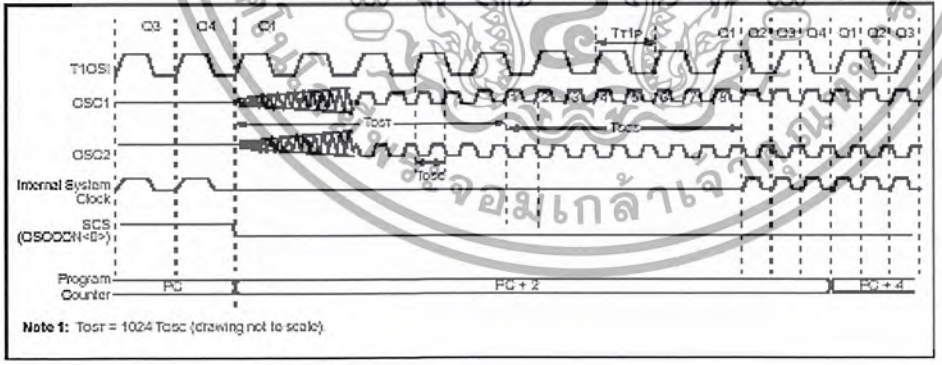


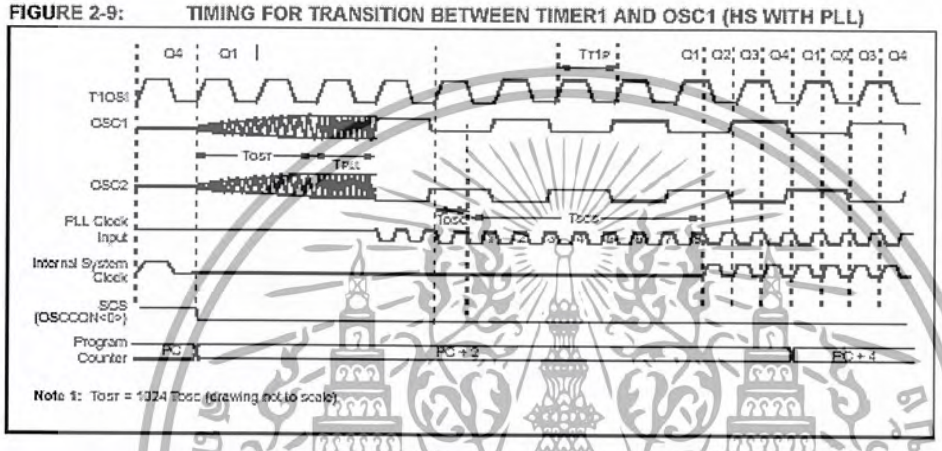
FIGURE 2-8: TIMING DIAGRAM FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS, XT, LP)



If the main oscillator is configured for HS4 (PLL) mode, an oscillator start-up time (TOST) plus an additional PLL time-out (TPLL) will occur. The PLL time-out is typically 2 ms and allows the PLL to lock to the main oscillator frequency. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for HS4 mode is shown in Figure 2-9.

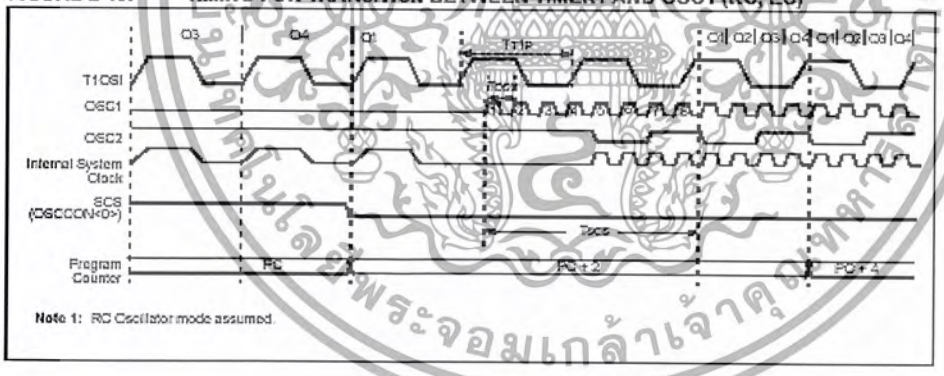
If the main oscillator is configured in the RC, RCIO, EC or ECIO modes, there is no oscillator start-up time-out. Operation will resume after eight cycles of the main oscillator have been counted. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for RC, RCIO, EC and ECIO modes is shown in Figure 2-10.

FIGURE 2-9: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS WITH PLL)



Note 1: TOST = 1024 TOSC (drawing not to scale)

FIGURE 2-10: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (RC, EC)



Note 1: RC Oscillator mode assumed.

2.7 Effects of SLEEP Mode on the On-Chip Oscillator

When the device executes a SLEEP instruction, the on-chip clocks and oscillator are turned off and the device is held at the beginning of an instruction cycle (Q1 state). With the oscillator off, the OSC1 and OSC2 signals will stop oscillating. Since all the transistor switching currents have been removed, SLEEP mode achieves the lowest current consumption of the device (only leakage currents). Enabling any on-chip feature that will operate during SLEEP will increase the current consumed during SLEEP. The user can wake from SLEEP through external RESET, Watchdog Timer Reset, or through an interrupt.

2.8 Power-up Delays

Power-up delays are controlled by two timers, so that no external RESET circuitry is required for most applications. The delays ensure that the device is kept in RESET until the device power supply and clock are stable. For additional information on RESET operation, see Section 3.0.

The first timer is the Power-up Timer (PWRT), which optionally provides a fixed delay of TPWRT (parameter #D033) on power-up only (POR and BOR). The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable.

With the PLL enabled (HS4 Oscillator mode), the time-out sequence following a Power-on Reset is different from other Oscillator modes. The time-out sequence is as follows: the PWRT time-out is invoked after a POR time delay has expired, then the Oscillator Start-up Timer (OST) is invoked. However, this is still not a sufficient amount of time to allow the PLL to lock at high frequencies. The PWRT timer is used to provide an additional fixed 2 ms (nominal) to allow the PLL ample time to lock to the incoming clock frequency.

TABLE 2-3: OSC1 AND OSC2 PIN STATES IN SLEEP MODE

OSC Mode	OSC1 Pin	OSC2 Pin
RC	Floating, external resistor should pull high	At logic low
RCIO	Floating, external resistor should pull high	Configured as PORTA, bit 6
ECIO	Floating	Configured as PORTA, bit 6
EC	Floating	At logic low
LP, XT, and HS	Feedback inverter disabled, at quiescent voltage level	Feedback inverter disabled, at quiescent voltage level

Note: See Table 3-1 in Section 3.0, for time-outs due to SLEEP and MCLR Reset.

3.0 RESET

The PIC18FXX8 differentiates between various kinds of RESET:

a) Power-on Reset (POR)
b) $\overline{MCLR}$ Reset during normal operation
c) $\overline{MCLR}$ Reset during SLEEP
d) Watchdog Timer (WDT) Reset during normal operation
e) Programmable Brown-out Reset (PBOR)
f) RESET Instruction
g) Stack Full Reset
h) Stack Underflow Reset

Most registers are unaffected by a RESET. Their status is unknown on POR and unchanged by all other RESETS. The other registers are forced to a "RESET" state on Power-on Reset, $\overline{MCLR}$, WDT Reset, Brown-out Reset, $\overline{MCLR}$ Reset during SLEEP and by the RESET instruction.

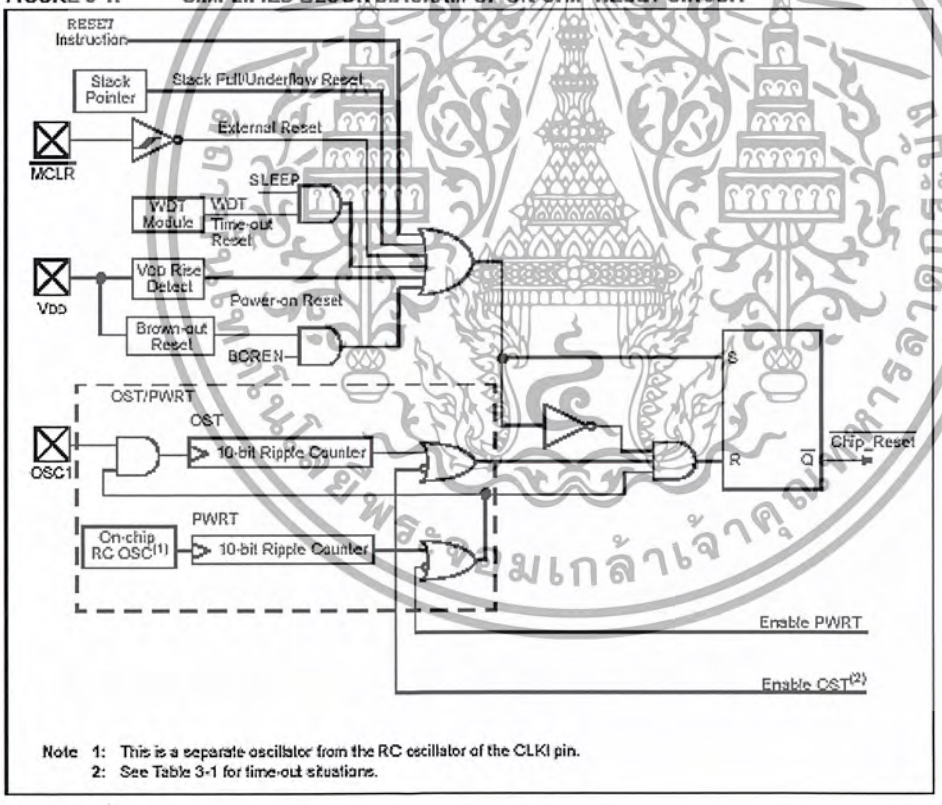
Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register, $\overline{RI}$, $\overline{TO}$, $\overline{PD}$, $\overline{POR}$ and $\overline{BOR}$ are set or cleared differently in different RESET situations, as indicated in Table 3-2. These bits are used in software to determine the nature of the RESET. See Table 3-3 for a full description of the RESET states of all registers.

A simplified block diagram of the on-chip RESET circuit is shown in Figure 3-1.

The Enhanced MCU devices have a $\overline{MCLR}$ noise filter in the $\overline{MCLR}$ Reset path. The filter will detect and ignore small pulses.

A WDT Reset does not drive $\overline{MCLR}$ pin low.

FIGURE 3-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



Note 1: This is a separate oscillator from the RC oscillator of the CLKI pin.
2: See Table 3-1 for time-out situations.

3.1 Power-on Reset (POR)

A Power-on Reset pulse is generated on-chip when a VDD rise is detected. To take advantage of the POR circuitry, connect the MCLR pin directly (or through a resistor) to VDD. This eliminates external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for VDD is specified (refer to parameter D004). For a slow rise time, see Figure 3-2.

When the device starts normal operation (exits the RESET condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in RESET until the operating conditions are met. Brown-out Reset may be used to meet the voltage start-up condition.

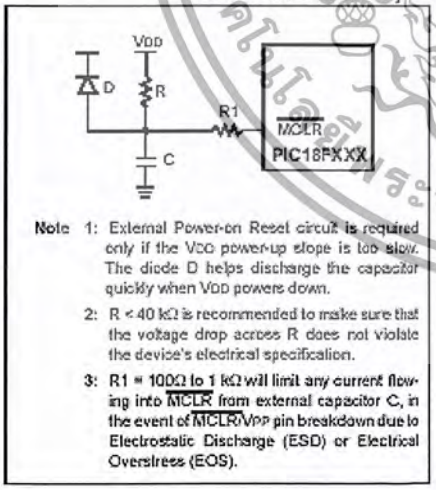
3.2 MCLR

PIC18FXX8 devices have a noise filter in the MCLR Reset path. The filter will detect and ignore small pulses.

It should be noted that a WDT Reset does not drive MCLR pin low.

The behavior of the ESD protection on the MCLR pin differs from previous devices of this family. Voltages applied to the pin that exceed its specification can result in both RESETS and current draws outside of device specification during the RESET event. For this reason, Microchip recommends that the MCLR pin no longer be tied directly to VDD. The use of an RC network, as shown in Figure 3-2, is suggested.

FIGURE 3-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)



Note 1: External Power-on Reset circuit is required only if the VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.

3: R1 = 100Ω to 1 kΩ will limit any current flowing into MCLR from external capacitor C, in the event of MCLR/VPP pin breakdown due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

3.3 Power-up Timer (PWRT)

The Power-up Timer provides a fixed nominal time-out (parameter #33), only on power-up from the POR. The Power-up Timer operates on an internal RC oscillator. The chip is kept in RESET as long as the PWRT is active. The PWRT's time delay allows VDD to rise to an acceptable level. A configuration bit (PWRTEN in CONFIG2L register) is provided to enable/disable the PWRT.

The power-up time delay will vary from chip to chip due to VDD, temperature and process variation. See DC parameter #33 for details.

3.4 Oscillator Start-up Timer (OST)

The Oscillator Start-up Timer (OST) provides 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over (parameter #32). This additional delay ensures that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP, HS and HS4 modes and only on Power-on Reset or wake-up from SLEEP.

3.5 PLL Lock Time-out

With the PLL enabled, the time-out sequence following a Power-on Reset is different from other oscillator modes. A portion of the Power-up Timer is used to provide a fixed time-out that is sufficient for the PLL to lock to the main oscillator frequency. This PLL lock time-out (TPLL) is typically 2 ms and follows the oscillator start-up time-out (OST).

3.6 Brown-out Reset (BOR)

A configuration bit, BOREN, can disable (if clear/programmed), or enable (if set), the Brown-out Reset circuitry. If VDD falls below parameter D005 for greater than parameter #35, the brown-out situation resets the chip. A RESET may not occur if VDD falls below parameter D005 for less than parameter #35. The chip will remain in Brown-out Reset until VDD rises above BVDD. The Power-up Timer will then be invoked and will keep the chip in RESET an additional time delay (parameter #33). If VDD drops below BVDD while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be initialized. Once VDD rises above BVDD, the Power-up Timer will execute the additional time delay.

3.7 Time-out Sequence

On power-up, the time-out sequence is as follows: First, PWRT time-out is invoked after the POR time delay has expired, then OST is activated. The total time-out will vary based on oscillator configuration and the status of the PWRT. For example, in RC mode with the PWRT disabled, there will be no time-out at all. Figure 3-3, Figure 3-4, Figure 3-5, Figure 3-6 and Figure 3-7 depict time-out sequences on power-up.

Since the time-outs occur from the POR pulse, if $\overline{MCLR}$ is kept low long enough, the time-outs will expire. Bringing $\overline{MCLR}$ high will begin execution immediately (Figure 3-5). This is useful for testing purposes or to synchronize more than one PIC18FXX8 device operating in parallel.

Table 3-2 shows the RESET conditions for some Special Function Registers, while Table 3-3 shows the RESET conditions for all registers.

TABLE 3-1: TIME-OUT IN VARIOUS SITUATIONS

Oscillator Configuration	Power-up[2]		Brown-out[2]	Wake-up from SLEEP or Oscillator Switch
	$\overline{PWRTEN}$ = 0	$\overline{PWRTEN}$ = 1		
HS with PLL enabled[1]	72 ms + 1024 Tosc + 2 ms	1024 Tosc + 2 ms	72 ms + 1024 Tosc + 2 ms	1024 Tosc + 2 ms
HS, XT, LP	72 ms + 1024 Tosc	1024 Tosc	72 ms + 1024 Tosc	1024 Tosc
EC	72 ms	—	72 ms	—
External RC	72 ms	—	72 ms	—

Note 1: 2 ms = Nominal time required for the 4X PLL to lock.
2: 72 ms is the nominal power-up timer delay.

REGISTER 3-1: RCON REGISTER BITS AND POSITIONS

R/W-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-1
IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

TABLE 3-2: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER

Condition	Program Counter	RCON Register	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	STKFUL	STKUNF
Power-on Reset	0000h	0--1 1100	1	1	1	0	0	u	u
$\overline{MCLR}$ Reset during normal operation	0000h	0--u uuuu	u	u	u	u	u	u	u
Software Reset during normal operation	0000h	0--0 uuuu	0	u	u	u	u	u	u
Stack Full Reset during normal operation	0000h	0--u uu11	u	u	u	1	1	u	1
Stack Underflow Reset during normal operation	0000h	0--u uu11	u	u	u	1	1	1	u
$\overline{MCLR}$ Reset during SLEEP	0000h	0--u 10uu	u	1	0	u	u	u	u
WDT Reset	0000h	0--u 01uu	u	0	1	u	u	u	u
WDT Wake-up	PC + 2	u--u 00uu	u	0	0	u	u	u	u
Brown-out Reset	0000h	0--1 11u0	1	1	1	u	0	u	u
Interrupt Wake-up from SLEEP	PC + 2[1]	u--u 00uu	u	1	0	u	u	u	u

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0'
Note 1: When the wake-up is due to an interrupt and the GIEH or GIEL bits are set, the PC is loaded with the interrupt vector (000008h or 000018h).

FIGURE 3-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD)

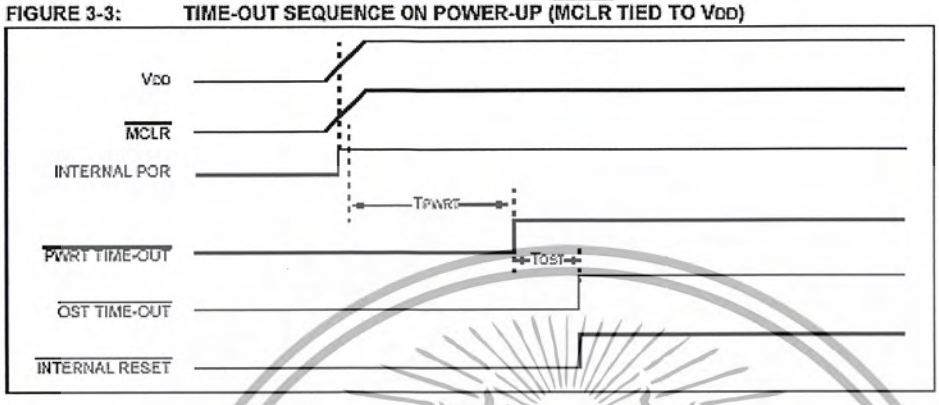


FIGURE 3-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1

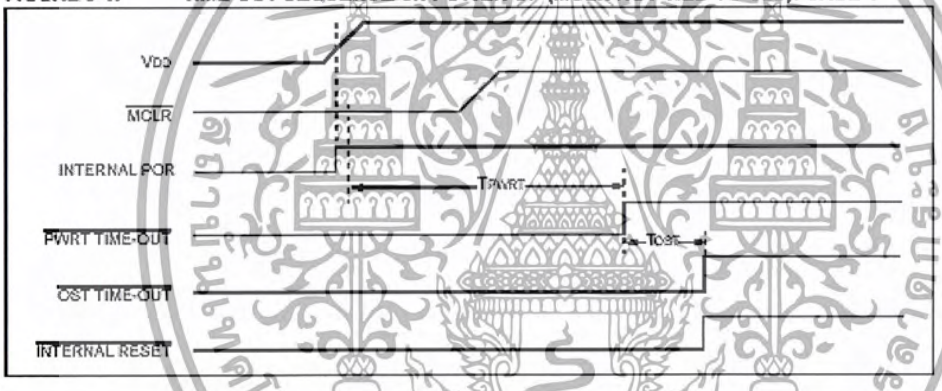


FIGURE 3-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2

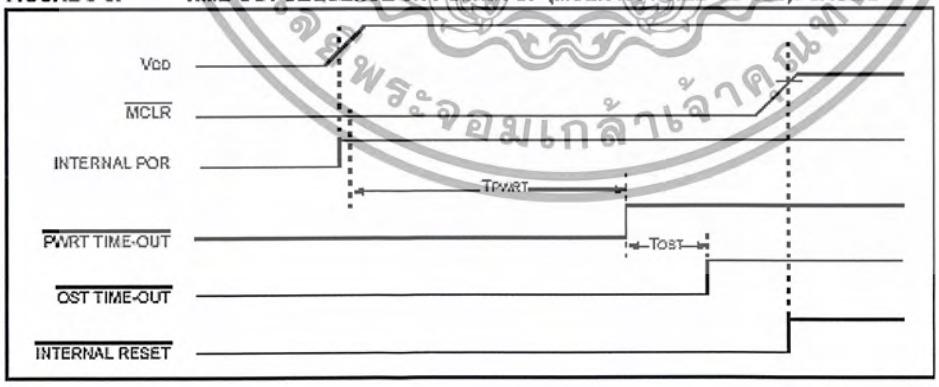


FIGURE 3-6: SLOW RISE TIME ($\overline{MCLR}$ TIED TO V_{DD})

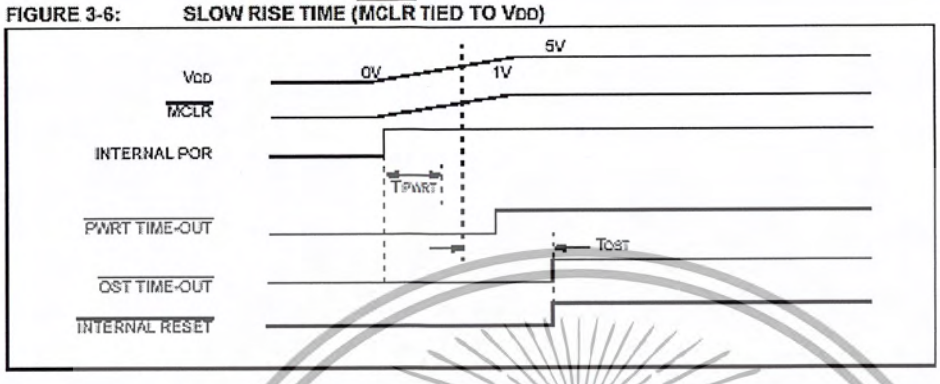
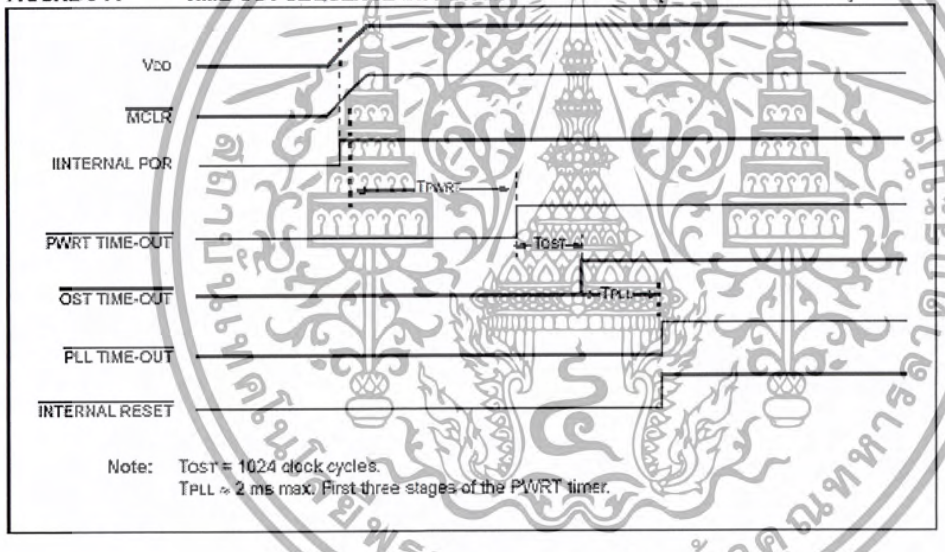


FIGURE 3-7: TIME-OUT SEQUENCE ON POR W/ PLL ENABLED ($\overline{MCLR}$ TIED TO V_{DD})



Note: TOST = 1024 clock cycles.
TPLL ≈ 2 ms max. First three stages of the PWRT timer.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Reset WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
TOSU	PIC18F2X8	PIC18F4X8	---0 0000	---0 0000	---0 uuuu[3]
TOSH	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu[3]
TOSL	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu[3]
STKPTR	PIC18F2X8	PIC18F4X8	00-0 0000	uu-0 0000	uu-u uuuu[3]
PCLATU	PIC18F2X8	PIC18F4X8	---0 0000	---0 0000	---u uuuu
PCLATH	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
PCL	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	PC + 2[2]
TBLPTRU	PIC18F2X8	PIC18F4X8	--00 0000	--00 0000	--uu uuuu
TBLPTRH	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
TABLAT	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
PRODH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	PIC18F2X8	PIC18F4X8	0000 000x	0000 000u	uuuu uuuu[1]
INTCON2	PIC18F2X8	PIC18F4X8	111- -1-1	111- -1-1	uuuu -u-u[1]
INTCON3	PIC18F2X8	PIC18F4X8	11-- 0-00	11-- 0-00	uu-u u-uu[1]
INDF0	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
POSTINC0	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
POSTDEC0	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
PREINC0	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
PLUSW0	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
FSR0H	PIC18F2X8	PIC18F4X8	---- 0000	---- 0000	---- uuuu
FSR0L	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
POSTINC1	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
POSTDEC1	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
PREINC1	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
PLUSW1	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.

6: Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Reset WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
FSR1H	PIC18F2X8	PIC18F4X8	---- 0000	---- 0000	---- uuuu
FSR1L	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	PIC18F2X8	PIC18F4X8	---- 0000	---- 0000	---- uuuu
INDF2	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
POSTINC2	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
POSTDEC2	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
PREINC2	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
PLUSW2	PIC18F2X8	PIC18F4X8	N/A	N/A	N/A
FSR2H	PIC18F2X8	PIC18F4X8	---- 0000	---- 0000	---- uuuu
FSR2L	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	PIC18F2X8	PIC18F4X8	---x xxxx	---u uuuu	---u uuuu
TMR0H	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR0L	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	PIC18F2X8	PIC18F4X8	1111 1111	1111 1111	uuuu uuuu
OSCCON	PIC18F2X8	PIC18F4X8	-110 ---0	-110 ---0	-uuu ---u
LVDCON	PIC18F2X8	PIC18F4X8	--00 0101	--00 0101	--uu uuuu
WDTCON	PIC18F2X8	PIC18F4X8	---- ---0	---- ---0	---- ---u
RCON[4]	PIC18F2X8	PIC18F4X8	0--1 11q0	0--1 qquu	u--u qquu
TMR1H	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	PIC18F2X8	PIC18F4X8	0-00 0000	u-uu uuuu	u-uu uuuu
TMR2	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
PR2	PIC18F2X8	PIC18F4X8	1111 1111	1111 1111	1111 1111
T2CON	PIC18F2X8	PIC18F4X8	-000 0000	-000 0000	-uuu uuuu
SSPBUF	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
SSPCON1	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
SSPCON2	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.

6: Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Reset WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
ADRESH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	PIC18F2X8	PIC18F4X8	0000 00-0	0000 00-0	uuuu uu-u
ADCON1	PIC18F2X8	PIC18F4X8	00-- 0000	00-- 0000	uu-- uuuu
CCPR1H	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	PIC18F2X8	PIC18F4X8	--00 0000	--00 0000	--uu uuuu
ECCPR1H	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
ECCPR1L	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
ECCP1CON	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	0000 0000
ECCP1DEL	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	0000 0000
ECCPAS	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	0000 0000
CVRCON	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
CMCON	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
TMR3H	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	PIC18F2X8	PIC18F4X8	0000 0000	uuuu uuuu	uuuu uuuu
SPBRG	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RCREG	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXREG	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXSTA	PIC18F2X8	PIC18F4X8	0000 -01x	0000 -01u	uuuu -uuu
RCSTA	PIC18F2X8	PIC18F4X8	0000 000x	0000 000u	uuuu uuuu
EEADR	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEDATA	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
EECON2	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
EECON1	PIC18F2X8	PIC18F4X8	xx-0 x000	uu-0 u000	uu-0 u000
IPR3	PIC18F2X8	PIC18F4X8	1111 1111	1111 1111	uuuu uuuu
PIR3	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
PIE3	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.

6: Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Reset WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
IPR2	PIC18F2X8	PIC18F4X8	-1-1 1111	-1-1 1111	-u-u uuuu
PIR2	PIC18F2X8	PIC18F4X8	-0-0 0000	-0-0 0000	-u-u uuuu(1)
PIE2	PIC18F2X8	PIC18F4X8	-0-0 0000	-0-0 0000	-u-u uuuu
IPR1	PIC18F2X8	PIC18F4X8	1111 1111	1111 1111	uuuu uuuu
PIR1	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu(1)
PIE1	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
TRISE	PIC18F2X8	PIC18F4X8	0000 -111	0000 -111	uuuu -uuu
TRISD	PIC18F2X8	PIC18F4X8	1111 1111	1111 1111	uuuu uuuu
TRISC	PIC18F2X8	PIC18F4X8	1111 1111	1111 1111	uuuu uuuu
TRISB	PIC18F2X8	PIC18F4X8	1111 1111	1111 1111	uuuu uuuu
TRISA(5)	PIC18F2X8	PIC18F4X8	-111 1111(5)	-111 1111(5)	-uuu uuuu(5)
LATE	PIC18F2X8	PIC18F4X8	---- -xxx	---- -uuu	---- -uuu
LATD	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA(5)	PIC18F2X8	PIC18F4X8	-xxx xxxx(5)	-uuu uuuu(5)	-uuu uuuu(6)
PORTE	PIC18F2X8	PIC18F4X8	---- -xxx	---- -000	---- -uuu
PORTD	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA(5)	PIC18F2X8	PIC18F4X8	-x0x 0000(5)	-u0u 0000(5)	-uuu uuuu(6)
TXERRCNT	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
RXERRCNT	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
COMSTAT	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
CIOCON	PIC18F2X8	PIC18F4X8	1000 1000	1000 1000	uuuu ----
BRGCON3	PIC18F2X8	PIC18F4X8	-0-- -000	-0-- -000	-u-- -uuu
BRGCON2	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
BRGCON1	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
CANCON	PIC18F2X8	PIC18F4X8	xxxx xxx-	uuuu uuu-	uuuu uuu-
CANSTAT(5)	PIC18F2X8	PIC18F4X8	xxx- xxx-	uuu- uuu-	uuu- uuu-

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.

6: Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Reset WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
RXB0D7	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D6	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D5	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D4	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D3	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D2	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D1	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D0	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0DLC	PIC18F2X8	PIC18F4X8	0xxx xxxx	0uuu uuuu	uuuu uuuu
RXB0EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0SIDL	PIC18F2X8	PIC18F4X8	xxxx x-xx	uuuu u-uu	uuuu u-uu
RXB0SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0CON	PIC18F2X8	PIC18F4X8	0x0- 0000	0u0- 0000	uuu- uuuu
RXB1D7	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D6	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D5	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D4	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D3	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D2	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D1	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D0	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1DLC	PIC18F2X8	PIC18F4X8	0xxx xxxx	0uuu uuuu	uuuu uuuu
RXB1EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1SIDL	PIC18F2X8	PIC18F4X8	xxxx xOxx	uuuu uuuu	uuuu uuuu
RXB1SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1CON	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
TXB0D7	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D6	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D5	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D4	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D3	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D2	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D1	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D0	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).
2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
4: See Table 3-2 for RESET value for specific condition.
5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.
6: Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Reset WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
TXB0DLC	PIC18F2X8	PIC18F4X8	0x00 xxxx	0u00 uuuu	uuuu uuuu
TXB0EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0SIDL	PIC18F2X8	PIC18F4X8	xxx0 x0xx	uuu0 u0uu	uuuu uuuu
TXB0SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0CON	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
TXB1D7	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D6	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D5	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D4	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D3	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D2	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D1	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D0	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1DLC	PIC18F2X8	PIC18F4X8	0x00 xxxx	0u00 uuuu	uuuu uuuu
TXB1EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1SIDL	PIC18F2X8	PIC18F4X8	xxx0 x0xx	uuu0 u0uu	uuuu uuuu
TXB1SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1CON	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu
TXB2D7	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2D6	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2D5	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2D4	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2D3	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2D2	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2D1	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2D0	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2DLC	PIC18F2X8	PIC18F4X8	0x00 xxxx	0u00 uuuu	uuuu uuuu
TXB2EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2SIDL	PIC18F2X8	PIC18F4X8	xxx0 x0xx	uuu0 u0uu	uuuu uuuu
TXB2SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2CON	PIC18F2X8	PIC18F4X8	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.

6: Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Reset WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
RXM1EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM1EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM1SIDL	PIC18F2X8	PIC18F4X8	xxx- --xx	uuu- --uu	uuu- --uu
RXM1SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0SIDL	PIC18F2X8	PIC18F4X8	xxx- --xx	uuu- --uu	uuu- --uu
RXM0SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF5SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF4SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF3SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF2SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF1SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0EIDL	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0EIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0SIDL	PIC18F2X8	PIC18F4X8	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF0SIDH	PIC18F2X8	PIC18F4X8	xxxx xxxx	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.

6: Values for CANSTAT also apply to its other instances (CANSTATRO1 through CANSTATRO4).

4.0 MEMORY ORGANIZATION

There are three memory blocks in Enhanced MCU devices. These memory blocks are:

- Enhanced FLASH Program Memory
- Data Memory
- EEPROM Data Memory

Data and program memory use separate busses, which allows concurrent access of these blocks. Additional detailed information on Data EEPROM and FLASH program memory is provided in Section 5.0 and Section 6.0, respectively.

4.1 Program Memory Organization

The PIC18F258/458 devices have a 21-bit program counter that is capable of addressing a 2-Mbyte program memory space.

The RESET vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F248/448

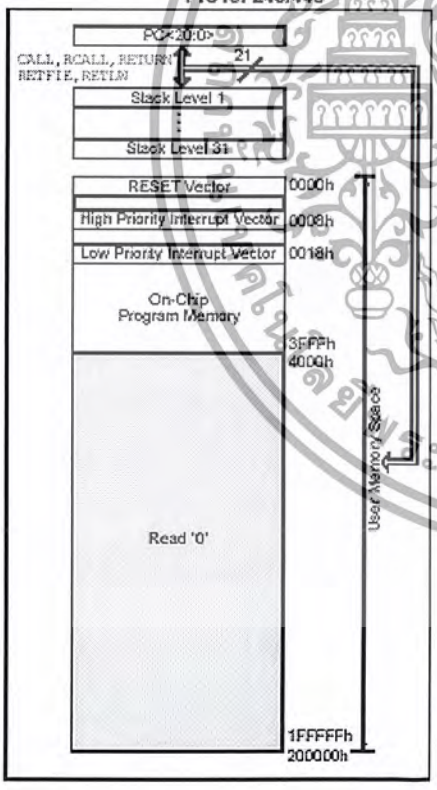


FIGURE 4-2: PROGRAM MEMORY MAP AND STACK FOR PIC18F258/458

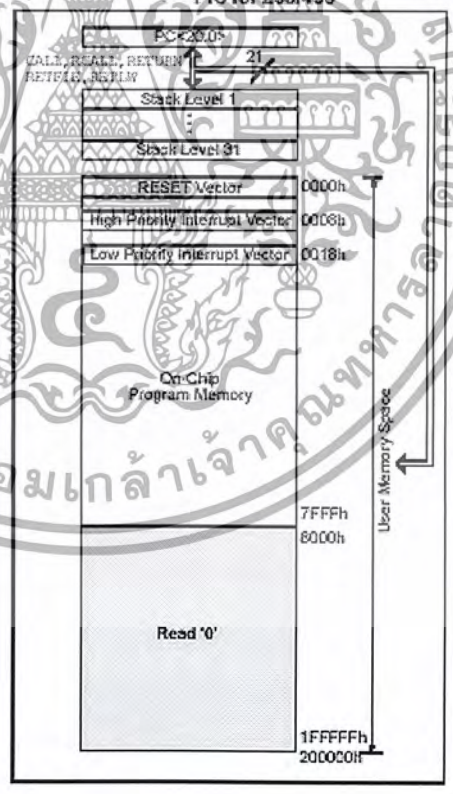


Figure 4-1 shows the diagram for program memory map and stack for the PIC18F248 and PIC18F448. Figure 4-2 shows the diagram for the program memory map and stack for the PIC18F258 and PIC18F458.

4.1.1 INTERNAL PROGRAM MEMORY OPERATION

The PIC18F258 and the PIC18F458 have 32 Kbytes of internal Enhanced FLASH program memory. This means that the PIC18F258 and the PIC18F458 can store up to 16K of single word instructions. The PIC18F248 and PIC18F448 have 16 Kbytes of Enhanced FLASH program memory. This translates into 8192 single word instructions, which can be stored in the program memory. Accessing a location between the physically implemented memory and the 2-Mbyte address will cause a read of all '0's (a NOP instruction).

4.2 Return Address Stack

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC (Program Counter) is pushed onto the stack when a PUSH, CALL or RCALL instruction is executed, or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the return instructions.

The stack operates as a 31-word by 21-bit stack memory and a 5-bit stack pointer, with the stack pointer initialized to 00000b after all RESETS. There is no RAM associated with stack pointer 00000b. This is only a RESET value. During a CALL type instruction, causing a push onto the stack, the stack pointer is first incremented and the RAM location pointed to by the stack pointer is written with the contents of the PC. During a RETURN type instruction, causing a pop from the stack, the contents of the RAM location indicated by the STKPTR is transferred to the PC and then the stack pointer is decremented.

The stack space is not part of either program or data space. The stack pointer is readable and writable, and the data on the top of the stack is readable and writable through SFR registers. Status bits indicate if the stack pointer is at or beyond the 31 levels provided.

4.2.1 TOP-OF-STACK ACCESS

The top of the stack is readable and writable. Three register locations, TOSU, TOSH and TOSL allow access to the contents of the stack location indicated by the STKPTR register. This allows users to implement a software stack, if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU, TOSH and TOSL registers. These values can be placed on a user defined software stack. At return time, the software can replace the TOSU, TOSH and TOSL and do a return.

The user should disable the global interrupt enable bits during this time to prevent inadvertent stack operations.

4.2.2 RETURN STACK POINTER (STKPTR)

The STKPTR register contains the stack pointer value, the STKFUL (stack full) status bit, and the STKUNF (stack underflow) status bits. Register 4-1 shows the STKPTR register. The value of the stack pointer can be 0 through 31. The stack pointer increments when values are pushed onto the stack and decrements when values are popped off the stack. At RESET, the stack pointer value will be '0'. The user may read and write the stack pointer value. This feature can be used by a Real-Time Operating System for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The STKFUL bit can only be cleared in software or by a POR.

The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) configuration bit. Refer to Section 21.0 for a description of the device configuration bits. If STVREN is set (default), the 31st push will push the (PC + 2) value onto the stack, set the STKFUL bit, and reset the device. The STKFUL bit will remain set and the stack pointer will be set to '0'.

If STVREN is cleared, the STKFUL bit will be set on the 31st push and the stack pointer will increment to 31. The 32nd push will overwrite the 31st push (and so on), while STKPTR remains at 31.

When the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC and sets the STKUNF bit, while the stack pointer remains at '0'. The STKUNF bit will remain set until cleared in software or a POR occurs.

Note:	Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the RESET vector, where the stack conditions can be verified and appropriate actions can be taken.

REGISTER 4-1: STKPTR: STACK POINTER REGISTER

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKFUL	STKUNF	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

bit 7 **STKFUL**: Stack Full Flag bit
1 = Stack became full or overflowed
0 = Stack has not become full or overflowed

bit 6 **STKUNF**: Stack Underflow Flag bit
1 = Stack underflow occurred
0 = Stack underflow did not occur

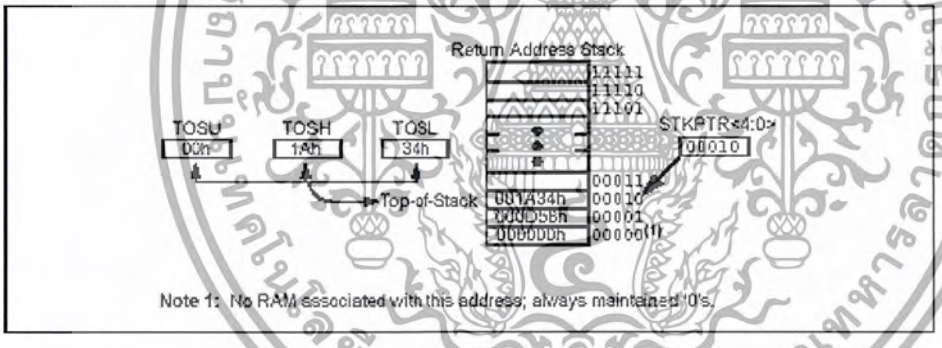
bit 5 **Unimplemented**: Read as '0'

bit 4-0 **SP4:SP0**: Stack Pointer Location bits

Note: Bit 7 and bit 6 need to be cleared following a stack underflow or a stack overflow.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	C = Clearable bit

FIGURE 4-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



Note 1: No RAM associated with this address; always maintained '0's.

4.2.3 PUSH AND POP INSTRUCTIONS

Since the Top-of-Stack (TOS) is readable and writable, the ability to push values onto the stack and pull values off the stack without disturbing normal program execution is a desirable option. To push the current PC value onto the stack, a PUSH instruction can be executed. This will increment the stack pointer and load the current PC value onto the stack. TOSU, TOSH and TOSL can then be modified to place a return address on the stack.

The POP instruction discards the current TOS by decrementing the stack pointer. The previous value pushed onto the stack then becomes the TOS value.

4.2.4 STACK FULL/UNDERFLOW RESETS

These RESETS are enabled by programming the STVREN configuration bit. When the STVREN bit is disabled, a full or underflow condition will set the appropriate STKFUL or STKUNF bit, but not cause a device RESET. When the STVREN bit is enabled, a full or underflow condition will set the appropriate STKFUL or STKUNF bit and then cause a device RESET. The STKFUL or STKUNF bits are only cleared by the user software or a POR.

4.3 Fast Register Stack

A "fast return" option is available for interrupts and calls. A fast register stack is provided for the STATUS, WREG and BSR registers and is only one layer in depth. The stack is not readable or writable and is loaded with the current value of the corresponding register when the processor vectors for an interrupt. The values in the fast register stack are then loaded back into the working registers if the `fast return` instruction is used to return from the interrupt.

A low or high priority interrupt source will push values into the stack registers. If both low and high priority interrupts are enabled, the stack registers cannot be used reliably for low priority interrupts. If a high priority interrupt occurs while servicing a low priority interrupt, the stack register values stored by the low priority interrupt will be overwritten.

If high priority interrupts are not disabled during low priority interrupts, users must save the key registers in software during a low priority interrupt.

If no interrupts are used, the fast register stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the fast register stack for a subroutine call, a `fast call` instruction must be executed.

Example 4-1 shows a source code example that uses the fast register stack.

EXAMPLE 4-1: FAST REGISTER STACK CODE EXAMPLE

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
          •
          •
SUB1      •
          •
          •
     RETURN FAST   ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

4.4 PCL, PCLATH and PCLATU

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21-bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<15:8> bits and is not directly readable or writable. Updates to the PCH register may be performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits and is not directly readable or writable. Updates to the PCU register may be performed through the PCLATU register.

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the LSb of PCL is fixed to a value of '0'. The PC increments by 2 to address sequential instructions in the program memory.

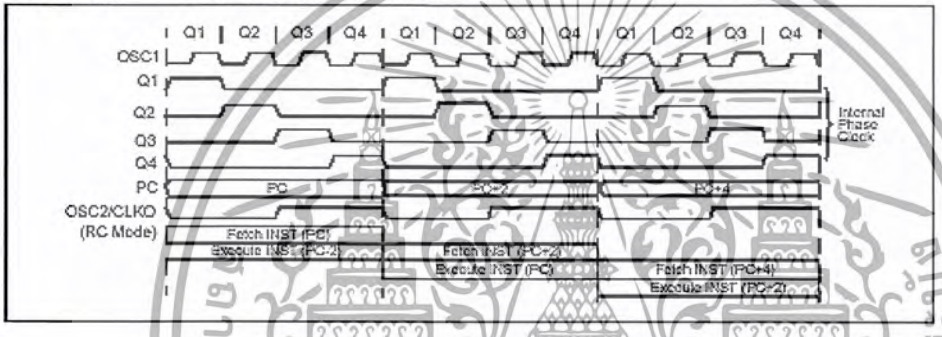
The `CALL`, `RCALL`, `GOTO` and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

The contents of PCLATH and PCLATU will be transferred to the program counter by an operation that writes PCL. Similarly, the upper two bytes of the program counter will be transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see Section 4.8.1).

4.5 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 4-4.

FIGURE 4-4: CLOCK/INSTRUCTION CYCLE



4.6 Instruction Flow/Pipelining

An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle, while decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO), two cycles are required to complete the instruction (Example 4-2).

A fetch cycle begins with the program counter (PC) incrementing in Q1.

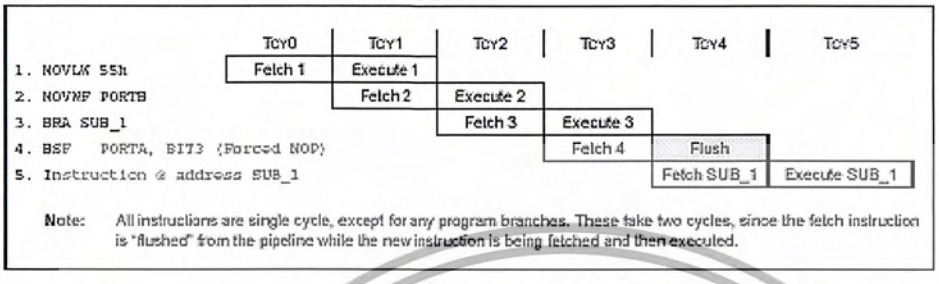
In the execution cycle, the fetched instruction is latched into the "Instruction Register" (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

4.7 Instructions in Program Memory

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSB = 0). Figure 4-3 shows an example of how instruction words are stored in the program memory. To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read '0' (see Section 4.4).

The CALL and GOTO instructions have an absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Example 4-3 shows how the instruction "GOTO 000006h" is encoded in the program memory. Program branch instructions that encode a relative address offset operate in the same manner. The offset value stored in a branch instruction represents the number of single word instructions by which the PC will be offset. Section 25.0 provides further details of the instruction set.

EXAMPLE 4-2: INSTRUCTION PIPELINE FLOW



Note: All instructions are single cycle, except for any program branches. These take two cycles, since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

EXAMPLE 4-3: INSTRUCTIONS IN PROGRAM MEMORY

Instruction	Opcode	Memory	Address
—			000007h
MOVLW 055h	0E55h	55h	000008h
		0Eh	000009h
GOTO 000006h	0EF03h, 0F000h	03h	00000Ah
		0EFh	00000Bh
		00h	00000Ch
		0F0h	00000Dh
MOVFF 123h, 456h	0C123h, 0F456h	23h	00000Eh
		0C1h	00000Fh
		56h	000010h
		0F4h	000011h
—			000012h

4.7.1 TWO-WORD INSTRUCTIONS

The PIC18FXX8 devices have 4 two-word instructions: MOVFF, CALL, GOTO and LFSR. The 4 Most Significant bits of the second word are set to '1's, and indicate a special NOP instruction. The lower 12 bits of the second word contain the data to be used by the instruction. If the first word of the instruction is executed, the data in the second word is accessed. If the second word of the instruction is executed by itself (first word was skipped), it will execute as a NOP. This action is necessary when the two-word instruction is preceded by a conditional instruction that changes the PC. A program example that demonstrates this concept is shown in Example 4-4. Refer to Section 25.0 for further details of the instruction set.

4.8 Lookup Tables

Lookup tables are implemented two ways. These are:

- Computed GOTO
- Table Reads

4.8.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL).

A lookup table can be formed with an ADDWF PCL instruction and a group of RETLW 0xnn instructions. WREG is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW 0xnn instructions that returns the value 0xnn to the calling function.

The offset value (value in WREG) specifies the number of bytes that the program counter should advance.

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

Note 1: The LSb of PCL is fixed to a value of '0'. Hence, computed GOTO to an odd address is not possible.

2: The ADDWF PCL instruction does not update PCLATH/PCLATU. A read operation on PCL must be performed to update PCLATH and PCLATU.

4.8.2 TABLE READS/TABLE WRITES

A better method of storing data in program memory allows 2 bytes of data to be stored in each instruction location.

Lookup table data may be stored as 2 bytes per program word by using table reads and writes. The table pointer (TBLPTR) specifies the byte address and the table latch (TABLAT) contains the data that is read from, or written to, program memory. Data is transferred to/from program memory, one byte at a time.

A description of the Table Read/Table Write operation is shown in Section 6.1.

EXAMPLE 4-4: TWO-WORD INSTRUCTIONS

CASE 1:

Object Code	Source Code		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; No, execute 2-word instruction
1111 0100 0101 0110			; 2nd operand holds address of REG2
0010 0100 0000 0000	ADDWF	REG3	; continue code

CASE 2:

Object Code	Source Code		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; Yes
1111 0100 0101 0110			; 2nd operand becomes NOP
0010 0100 0000 0000	ADDWF	REG3	; continue code

4.9 Data Memory Organization

The data memory is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. Figure 4-6 shows the data memory organization for the PIC18FXX8 devices.

The data memory map is divided into as many as 16 banks that contain 256 bytes each. The lower 4 bits of the Bank Select Register (BSR<3:0>) select which bank will be accessed. The upper 4 bits for the BSR are not implemented.

The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPR's are used for data storage and scratch pad operations in the user's application. The SFR's start at the last location of Bank 15 (FFFh) and grow downwards. GPRs start at the first location of Bank 0 and grow upwards. Any read of an unimplemented location will read as '0's.

The entire data memory may be accessed directly, or indirectly. Direct addressing may require the use of the BSR register. Indirect addressing requires the use of the File Select Register (FSR). Each FSR holds a 12-bit address value that can be used to access any location in the Data Memory map without banking.

The instruction set and architecture allow operations across all banks. This may be accomplished by indirect addressing or by the use of the MOVFF instruction. The MOVFF instruction is a two-word/two-cycle instruction, that moves a value from one register to another.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access RAM. Section 4.10 provides a detailed description of the Access RAM.

4.9.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly, or indirectly. Indirect addressing operates through the File Select Registers (FSR). The operation of indirect addressing is shown in Section 4.12.

Enhanced MCU devices may have banked memory in the GPR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other RESETS.

Data RAM is available for use as GPR registers by all instructions. Bank 15 (F00h to FFFh) contains SFRs. All other banks of data memory contain GPR registers, starting with Bank 0.

4.9.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and Peripheral Modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 4-1.

The SFRs can be classified into two sets: those associated with the "core" function and those related to the peripheral functions. Those registers related to the "core" are described in this section, while those related to the operation of the peripheral features are described in the section of that peripheral feature.

The SFRs are typically distributed among the peripherals whose functions they control.

The unused SFR locations will be unimplemented and read as '0's. See Table 4-1 for addresses for the SFRs.

FIGURE 4-5: DATA MEMORY MAP FOR PIC18F248/448

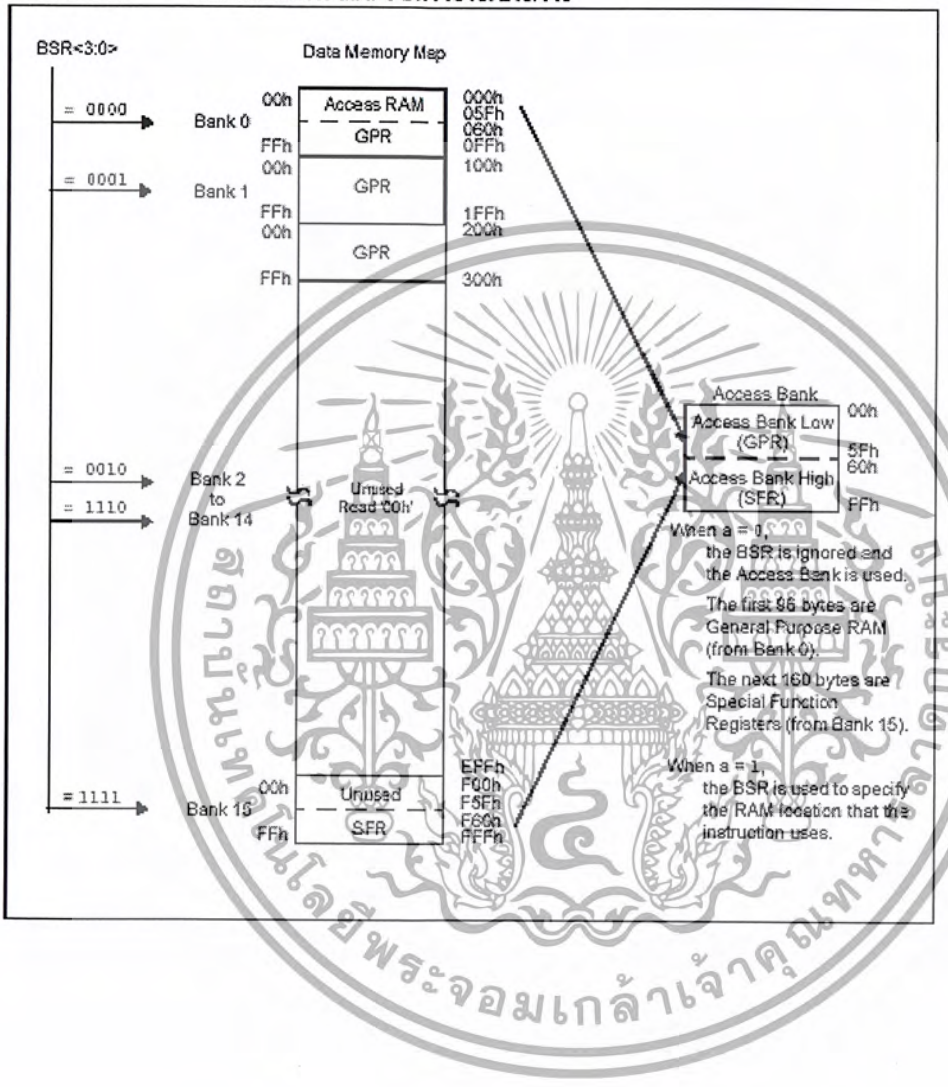


FIGURE 4-6: DATA MEMORY MAP FOR PIC18F258/458

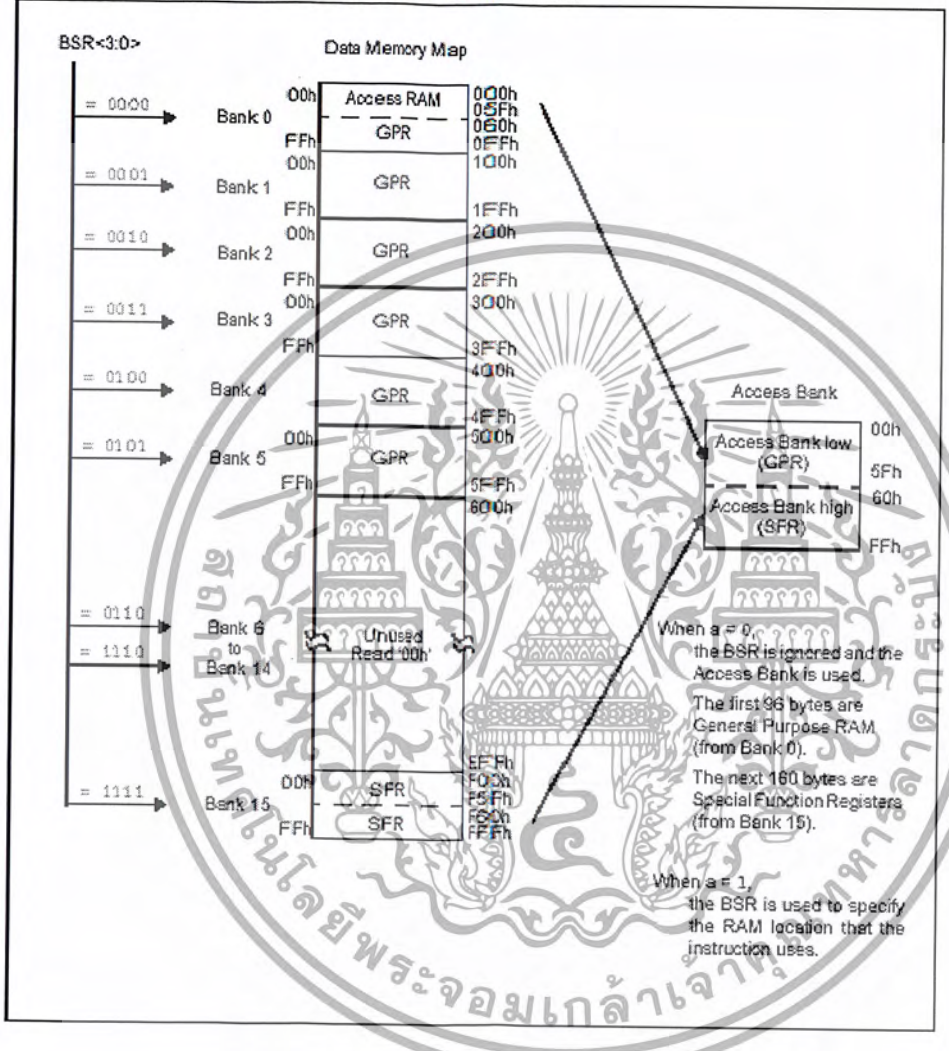


TABLE 4-1: SPECIAL FUNCTION REGISTER MAP

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(2)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(2)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(2)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(2)	FBCh	ECCPR1H(5)	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2(2)	FBBh	ECCPR1L(5)	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	ECCP1CON(5)	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	ECCP1DEL(5)	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCPAS(5)	F96h	TRISE(5)
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON(5)	F95h	TRISD(5)
FF4h	PRODH	FD4h	—	FB4h	CMCON(5)	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0(2)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0(2)	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0(2)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE(5)
FECh	PREINC0(2)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD(5)
FEBh	PLUSW0(2)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—
FE7h	INDF1(2)	FC7h	SSPSTAT	FA7h	EECON2	F87h	—
FE6h	POSTINC1(2)	FC6h	SSPCON1	FA6h	EECON1	F86h	—
FE5h	POSTDEC1(2)	FC5h	SSPCON2	FA5h	IPR3	F85h	—
FE4h	PREINC1(2)	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE(5)
FE3h	PLUSW1(2)	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD(5)
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

Note 1: Unimplemented registers are read as '0'.

2: This is not a physical register.

3: Contents of register are dependent on WIN2:WIN0 bits in CANCON register.

4: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the CANSTAT register, due to the Microchip Header file requirement.

5: These registers are not implemented on the PIC18F248 and PIC18F258.

TABLE 4-1: SPECIAL FUNCTION REGISTER MAP (CONTINUED)

Address	Name	Address	Name	Address	Name	Address	Name
F7Fh	—	F5Fh	—	F3Fh	—	F1Fh	RXM1EIDL
F7Eh	—	F5Eh	CANSTATRO1(4)	F3Eh	CANSTATRO3(4)	F1Eh	RXM1EIDH
F7Dh	—	F5Dh	RXB1D7	F3Dh	TXB1D7	F1Dh	RXM1SIDL
F7Ch	—	F5Ch	RXB1D6	F3Ch	TXB1D6	F1Ch	RXM1SIDH
F7Bh	—	F5Bh	RXB1D5	F3Bh	TXB1D5	F1Bh	RXM0EIDL
F7Ah	—	F5Ah	RXB1D4	F3Ah	TXB1D4	F1Ah	RXM0EIDH
F79h	—	F59h	RXB1D3	F39h	TXB1D3	F19h	RXM0SIDL
F78h	—	F58h	RXB1D2	F38h	TXB1D2	F18h	RXM0SIDH
F77h	—	F57h	RXB1D1	F37h	TXB1D1	F17h	RXF5EIDL
F76h	TXERRCNT	F56h	RXB1D0	F36h	TXB1D0	F16h	RXF5EIDH
F75h	RXERRCNT	F55h	RXB1DLC	F35h	TXB1DLC	F15h	RXF5SIDL
F74h	COMSTAT	F54h	RXB1EIDL	F34h	TXB1EIDL	F14h	RXF5SIDH
F73h	CIOCON	F53h	RXB1EIDH	F33h	TXB1EIDH	F13h	RXF4EIDL
F72h	BRGCON3	F52h	RXB1SIDL	F32h	TXB1SIDL	F12h	RXF4EIDH
F71h	BRGCON2	F51h	RXB1SIDH	F31h	TXB1SIDH	F11h	RXF4SIDL
F70h	BRGCON1	F50h	RXB1CON	F30h	TXB1CON	F10h	RXF4SIDH
F6Fh	CANCON	F4Fh	—	F2Fh	—	F0Fh	RXF3EIDL
F6Eh	CANSTAT	F4Eh	CANSTATRO2(4)	F2Eh	CANSTATRO4(4)	F0Eh	RXF3EIDH
F6Dh	RXB0D7(3)	F4Dh	TXB0D7	F2Dh	TXB2D7	F0Dh	RXF3SIDL
F6Ch	RXB0D6(3)	F4Ch	TXB0D6	F2Ch	TXB2D6	F0Ch	RXF3SIDH
F6Bh	RXB0D5(3)	F4Bh	TXB0D5	F2Bh	TXB2D5	F0Bh	RXF2EIDL
F6Ah	RXB0D4(3)	F4Ah	TXB0D4	F2Ah	TXB2D4	F0Ah	RXF2EIDH
F69h	RXB0D3(3)	F49h	TXB0D3	F29h	TXB2D3	F09h	RXF2SIDL
F68h	RXB0D2(3)	F48h	TXB0D2	F28h	TXB2D2	F08h	RXF2SIDH
F67h	RXB0D1(3)	F47h	TXB0D1	F27h	TXB2D1	F07h	RXF1EIDL
F66h	RXB0D0(3)	F46h	TXB0D0	F26h	TXB2D0	F06h	RXF1EIDH
F65h	RXB0DLC(3)	F45h	TXB0DLC	F25h	TXB2DLC	F05h	RXF1SIDL
F64h	RXB0EIDL(3)	F44h	TXB0EIDL	F24h	TXB2EIDL	F04h	RXF1SIDH
F63h	RXB0EIDH(3)	F43h	TXB0EIDH	F23h	TXB2EIDH	F03h	RXF0EIDL
F62h	RXB0SIDL(3)	F42h	TXB0SIDL	F22h	TXB2SIDL	F02h	RXF0EIDH
F61h	RXB0SIDH(3)	F41h	TXB0SIDH	F21h	TXB2SIDH	F01h	RXF0SIDL
F60h	RXB0CON(3)	F40h	TXB0CON	F20h	TXB2CON	F00h	RXF0SIDH

Note: Shaded registers are available in Bank 15, while the rest are in Access Bank low.

Note 1: Unimplemented registers are read as '0'.

2: This is not a physical register.

3: Contents of register are dependent on WIN2:WIN0 bits in CANCON register.

4: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the CANSTAT register, due to the Microchip Header file requirement.

5: These registers are not implemented on the PIC18F248 and PIC18F258.

TABLE 4-2: REGISTER FILE SUMMARY

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on Page:
TOSU	—	—	—	Top-of-Stack Upper Byte (TOS<20:16>)					---0 0000	30, 38
TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	30, 38
TOSL	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	30, 38
STKPTR	STKFUL	STKUNF	—	Return Stack Pointer					00-0 0000	30, 39
PCLATU	—	—	bit21(2)	Holding Register for PC<20:16>					---0 0000	30, 40
PCLATH	Holding Register for PC<15:8>								0000 0000	30, 40
PCL	PC Low Byte (PC<7:0>)								0000 0000	30, 40
TBLPTRU	—	—	bit21(2)	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					--00 0000	30, 68
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	30, 68
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	30, 68
TABLAT	Program Memory Table Latch								0000 0000	30, 68
PRODH	Product Register High Byte								xxxx xxxx	30, 75
PRODL	Product Register Low Byte								xxxx xxxx	30, 75
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	30, 79
INTCON2	RBPU	INTEDG0	INTEDG1	—	—	TMR0IP	—	RBIP	111- -1-1	30, 80
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-1 0-00	30, 81
INDF0	Uses contents of FSR0 to address data memory - value of FSR0 not changed (not a physical register)								n/a	30, 55
POSTINC0	Uses contents of FSR0 to address data memory - value of FSR0 post-incremented (not a physical register)								n/a	30, 55
POSTDEC0	Uses contents of FSR0 to address data memory - value of FSR0 post-incremented (not a physical register)								n/a	30, 55
PREINC0	Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register)								n/a	30, 55
PLUSW0	Uses contents of FSR0 to address data memory - value of FSR0 offset by W (not a physical register)								n/a	30, 55
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High				---- xxxx	30, 55
FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx	30, 55
WREG	Working Register								xxxx xxxx	30, 55
INDF1	Uses contents of FSR1 to address data memory - value of FSR1 not changed (not a physical register)								n/a	30, 55
POSTINC1	Uses contents of FSR1 to address data memory - value of FSR1 post-incremented (not a physical register)								n/a	30, 55
POSTDEC1	Uses contents of FSR1 to address data memory - value of FSR1 post-incremented (not a physical register)								n/a	30, 55
PREINC1	Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented (not a physical register)								n/a	30, 55
PLUSW1	Uses contents of FSR1 to address data memory - value of FSR1 offset by W (not a physical register)								n/a	30, 55
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High				---- xxxx	31, 55
FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	31, 55
BSR	—	—	—	—	Bank Select Register				---- 0000	31, 54
INDF2	Uses contents of FSR2 to address data memory - value of FSR2 not changed (not a physical register)								n/a	31, 55
POSTINC2	Uses contents of FSR2 to address data memory - value of FSR2 post-incremented (not a physical register)								n/a	31, 55
POSTDEC2	Uses contents of FSR2 to address data memory - value of FSR2 post-incremented (not a physical register)								n/a	31, 55
PREINC2	Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented (not a physical register)								n/a	31, 55
PLUSW2	Uses contents of FSR2 to address data memory - value of FSR2 offset by W (not a physical register)								n/a	31, 55
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High				---- xxxx	31, 55
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	31, 55
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	31, 57
TMR0H	Timer0 Register High Byte								0000 0000	31, 111
TMR0L	Timer0 Register Low Byte								xxxx xxxx	31, 111
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	31, 109
OSCCON	—	—	—	—	—	—	—	SCS	---- ---0	31, 20
LVDCON	—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0	--00 0101	31, 261
WDTCON	—	—	—	—	—	—	—	SWDTEN	---- ---0	31, 272
RCON	IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	0--1 11qq	31, 58, 91

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition

Note 1: These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other Oscillator modes.

TABLE 4-2: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on Page:
TMR1H	Timer1 Register High Byte								xxxx xxxx	31, 115
TMR1L	Timer1 Register Low Byte								xxxx xxxx	31, 115
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	31, 113
TMR2	Timer2 Register								0000 0000	31, 118
PR2	Timer2 Period Register								1111 1111	31, 118
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	31, 117
SSPBUF	SSP Receive Buffer/Transmit Register								xxxx xxxx	31, 146
SSPADD	SSP Address Register in I²C Slave mode. SSP Baud Rate Reload Register in I²C Master mode.								0000 0000	31, 152
SSPSTAT	SMP	CKE	$\overline{\text{D/A}}$	P	S	R/$\overline{\text{W}}$	UA	BF	0000 0000	31, 144, 153
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	31, 145, 145
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	31, 155
ADRESH	A/D Result Register High Byte								xxxx xxxx	32, 243
ADRESL	A/D Result Register Low Byte								xxxx xxxx	32, 243
ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON	0000 00-0	32, 241
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	32, 242
CCPR1H	Capture/Compare/PWM Register1 High Byte								xxxx xxxx	32, 124
CCPR1L	Capture/Compare/PWM Register1 Low Byte								xxxx xxxx	32, 124
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	32, 123
ECCPR1H[1]	Enhanced Capture/Compare/PWM Register1 High Byte								xxxx xxxx	32, 133
ECCPR1L[1]	Enhanced Capture/Compare/PWM Register1 Low Byte								xxxx xxxx	32, 133
ECCP1CON[1]	EPWM1M1	EPWM1M0	EDC1B1	EDC1B0	ECCP1M3	ECCP1M2	ECCP1M1	ECCP1M0	0000 0000	32, 131
ECCP1DEL[1]	EPDC7	EPDC6	EPDC5	EPDC4	EPDC3	EPDC2	EPDC1	EPDC0	0000 0000	32, 140
ECCPAS[1]	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	0000 0000	32, 142
CVRCON[1]	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	32, 255
CMCON[1]	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	32, 249
TMR3H	Timer3 Register High Byte								xxxx xxxx	32, 121
TMR3L	Timer3 Register Low Byte								xxxx xxxx	32, 121
T3CON	RD16	T3ECCP1	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	32, 119
SPBRG	USART1 Baud Rate Generator								0000 0000	32, 185
RCREG	USART1 Receive Register								0000 0000	32, 191
TXREG	USART1 Transmit Register								0000 0000	32, 189
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	32, 183
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	32, 184
EEADR	EEPROM Address Register								xxxx xxxx	32, 59
EEDATA	EEPROM Data Register								xxxx xxxx	32, 59
EECON2	EEPROM Control Register2 (not a physical register)								xxxx xxxx	32, 59
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	$\overline{\text{WR}}$	$\overline{\text{RD}}$	xx-0 x000	32, 60, 67
IPR3	IRXIP	WAKIP	ERRIP	TXB2IP	TXB1IP	TXB0IP	RXB1IP	RXB0IP	1111 1111	32, 90
PIR3	IRXIF	WAKIF	ERRIF	TXB2IF	TXB1IF	TXB0IF	RXB1IF	RXB0IF	0000 0000	32, 84
PIE3	IRXIE	WAKIE	ERRIE	TXB2IE	TXB1IE	TXB0IE	RXB1IE	RXB0IE	0000 0000	32, 87
IPR2	—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	ECCP1IP[3]	-1-1 1111	33, 89
PIR2	—	CMIF	—	EEIF	BCLIF	LVDIF	TMR3IF	ECCP1IF[3]	-0-0 0000	33, 83
PIE2	—	CMIE	—	EEIE	BCLIE	LVDIE	TMR3IE	ECCP1IE[3]	-0-0 0000	33, 86

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition

Note 1: These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.
2: Bit 21 of the TBLPTRU allows access to the device configuration bits.
3: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other Oscillator modes.

TABLE 4-2: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on Page:
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	33, 88
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	33, 82
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	33, 85
TRISE[1]	IBF	OBF	IBOV	PSPMODE	—	Data Direction bits for PORTE[1]			0000 -111	33, 105
TRISD[1]	Data Direction Control Register for PORTD[1]								1111 1111	33, 102
TRISC	Data Direction Control Register for PORTC								1111 1111	33, 100
TRISB	Data Direction Control Register for PORTB								1111 1111	33, 96
TRISA[3]	—	Data Direction Control Register for PORTA							--11 1111	33, 93
LATE[1]	—	—	—	—	—	Read PORTE Data Latch, Write PORTE Data Latch[1]			---- -xxx	33, 104
LATD[1]	Read PORTD Data Latch, Write PORTD Data Latch[1]								xxxx xxxx	33, 102
LATC	Read PORTC Data Latch, Write PORTC Data Latch								xxxx xxxx	33, 100
LATB	Read PORTB Data Latch, Write PORTB Data Latch								xxxx xxxx	33, 96
LATA[3]	—	Read PORTA Data Latch, Write PORTA Data Latch							-xxx xxxx	33, 93
PORTE[1]	—	—	—	—	—	Read PORTE pins, Write PORTE Data Latch[1]			---- -000	33, 104
PORTD[1]	Read PORTD pins, Write PORTD Data Latch[1]								xxxx xxxx	33, 102
PORTC	Read PORTC pins, Write PORTC Data Latch								xxxx xxxx	33, 100
PORTB	Read PORTB pins, Write PORTB Data Latch								xxxx xxxx	33, 96
PORTA[3]	—	Read PORTA pins, Write PORTA Data Latch							-xxx 0000	33, 93
TXERRCNT	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	0000 0000	33, 209
RXERRCNT	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	0000 0000	33, 214
COMSTAT	RXB0OVFL	RXB1OVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN	0000 0000	33, 205
CIOCON	—	—	ENDRH	CANCAP	—	—	—	—	--20 ----	33, 221
BRGCON3	—	WAKFIL	—	—	—	SEG2PH2	SEG2PH1	SEG2PH0	-0-- -000	33, 220
BRGCON2	SEG2PHTS	SAM	SEG1PH2	SEG1PH1	SEG1PH0	PRSEG2	PRSEG1	PRSEG0	0000 0000	33, 219
BRGCON1	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	0000 0000	33, 218
CANCON	REQOP2	REQOP1	REQOP0	ABAT	WIN2	WIN1	WIN0	—	xxxx xxx-	33, 201
CANSTAT	OPMODE2	OPMODE1	OPMODE0	—	ICODE2	ICODE1	ICODE0	—	xxx- xxx-	33, 202
RXBCD7	RXB0D77	RXB0D76	RXB0D75	RXB0D74	RXB0D73	RXB0D72	RXB0D71	RXB0D70	xxxx xxxx	34, 214
RXBCD6	RXB0D67	RXB0D66	RXB0D65	RXB0D64	RXB0D63	RXB0D62	RXB0D61	RXB0D60	xxxx xxxx	34, 214
RXBCD5	RXB0D57	RXB0D56	RXB0D55	RXB0D54	RXB0D53	RXB0D52	RXB0D51	RXB0D50	xxxx xxxx	34, 214
RXBCD4	RXB0D47	RXB0D46	RXB0D45	RXB0D44	RXB0D43	RXB0D42	RXB0D41	RXB0D40	xxxx xxxx	34, 214
RXBCD3	RXB0D37	RXB0D36	RXB0D35	RXB0D34	RXB0D33	RXB0D32	RXB0D31	RXB0D30	xxxx xxxx	34, 214
RXBCD2	RXB0D27	RXB0D26	RXB0D25	RXB0D24	RXB0D23	RXB0D22	RXB0D21	RXB0D20	xxxx xxxx	34, 214
RXBCD1	RXB0D17	RXB0D16	RXB0D15	RXB0D14	RXB0D13	RXB0D12	RXB0D11	RXB0D10	xxxx xxxx	34, 214
RXBCD0	RXB0D07	RXB0D06	RXB0D05	RXB0D04	RXB0D03	RXB0D02	RXB0D01	RXB0D00	xxxx xxxx	34, 214
RXBCDLC	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	34, 213
RXBCEIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	34, 213
RXBCEIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	34, 212
RXBCSIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	xxx- x-xx	34, 212
RXBCSIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	34, 212
RXBCCON	RXFUL	RXM1	RXM0	—	RXRTRRO	RXBCCEN	JTOFF	FILHIT0	000- 0000	34, 210

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition

Note 1: These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other Oscillator modes.

TABLE 4-2: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on Page:
CANSTATRO1	OPMODE2	OPMODE1	OPMODE0	—	ICODE2	ICODE1	ICODE0	—	xxx- xxx-	33, 202
RXB1D7	RXB1D77	RXB1D76	RXB1D75	RXB1D74	RXB1D73	RXB1D72	RXB1D71	RXB1D70	xxxx xxxx	34, 214
RXB1D6	RXB1D67	RXB1D66	RXB1D65	RXB1D64	RXB1D63	RXB1D62	RXB1D61	RXB1D60	xxxx xxxx	34, 214
RXB1D5	RXB1D57	RXB1D56	RXB1D55	RXB1D54	RXB1D53	RXB1D52	RXB1D51	RXB1D50	xxxx xxxx	34, 214
RXB1D4	RXB1D47	RXB1D46	RXB1D45	RXB1D44	RXB1D43	RXB1D42	RXB1D41	RXB1D40	xxxx xxxx	34, 214
RXB1D3	RXB1D37	RXB1D36	RXB1D35	RXB1D34	RXB1D33	RXB1D32	RXB1D31	RXB1D30	xxxx xxxx	34, 214
RXB1D2	RXB1D27	RXB1D26	RXB1D25	RXB1D24	RXB1D23	RXB1D22	RXB1D21	RXB1D20	xxxx xxxx	34, 214
RXB1D1	RXB1D17	RXB1D16	RXB1D15	RXB1D14	RXB1D13	RXB1D12	RXB1D11	RXB1D10	xxxx xxxx	34, 214
RXB1D0	RXB1D07	RXB1D06	RXB1D05	RXB1D04	RXB1D03	RXB1D02	RXB1D01	RXB1D00	xxxx xxxx	34, 214
RXB1DLC	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	34, 213
RXB1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	34, 213
RXB1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	34, 212
RXB1SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	xxxx x-xx	34, 212
RXB1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	34, 212
RXB1CON	RXFUL	RXM1	RXM0	—	RXRTRRO	FILHIT2	FILHIT1	FILHIT0	000- 0000	34, 211
CANSTATRO2	OPMODE2	OPMODE1	OPMODE0	—	ICODE2	ICODE1	ICODE0	—	xxx- xxx-	33, 202
TXB0D7	TXB0D77	TXB0D76	TXB0D75	TXB0D74	TXB0D73	TXB0D72	TXB0D71	TXB0D70	xxxx xxxx	34, 208
TXB0D6	TXB0D67	TXB0D66	TXB0D65	TXB0D64	TXB0D63	TXB0D62	TXB0D61	TXB0D60	xxxx xxxx	34, 208
TXB0D5	TXB0D57	TXB0D56	TXB0D55	TXB0D54	TXB0D53	TXB0D52	TXB0D51	TXB0D50	xxxx xxxx	34, 208
TXB0D4	TXB0D47	TXB0D46	TXB0D45	TXB0D44	TXB0D43	TXB0D42	TXB0D41	TXB0D40	xxxx xxxx	34, 208
TXB0D3	TXB0D37	TXB0D36	TXB0D35	TXB0D34	TXB0D33	TXB0D32	TXB0D31	TXB0D30	xxxx xxxx	34, 208
TXB0D2	TXB0D27	TXB0D26	TXB0D25	TXB0D24	TXB0D23	TXB0D22	TXB0D21	TXB0D20	xxxx xxxx	34, 208
TXB0D1	TXB0D17	TXB0D16	TXB0D15	TXB0D14	TXB0D13	TXB0D12	TXB0D11	TXB0D10	xxxx xxxx	34, 208
TXB0D0	TXB0D07	TXB0D06	TXB0D05	TXB0D04	TXB0D03	TXB0D02	TXB0D01	TXB0D00	xxxx xxxx	34, 208
TXB0DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	-x-- xxxx	35, 209
TXB0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	35, 208
TXB0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	35, 207
TXB0SIDL	SID2	SID1	SID0	—	EXIDE	—	EID17	EID16	xxx- x-xx	35, 207
TXB0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	35, 207
TXB0CON	—	TXABT	TXLARB	TXERR	TXREQ	—	TXPR1	TXPR0	-000 0-00	35, 206
CANSTATRO3	OPMODE2	OPMODE1	OPMODE0	—	ICODE2	ICODE1	ICODE0	—	xxx- xxx-	33, 202
TXB1D7	TXB1D77	TXB1D76	TXB1D75	TXB1D74	TXB1D73	TXB1D72	TXB1D71	TXB1D70	xxxx xxxx	35, 208
TXB1D6	TXB1D67	TXB1D66	TXB1D65	TXB1D64	TXB1D63	TXB1D62	TXB1D61	TXB1D60	xxxx xxxx	35, 208
TXB1D5	TXB1D57	TXB1D56	TXB1D55	TXB1D54	TXB1D53	TXB1D52	TXB1D51	TXB1D50	xxxx xxxx	35, 208
TXB1D4	TXB1D47	TXB1D46	TXB1D45	TXB1D44	TXB1D43	TXB1D42	TXB1D41	TXB1D40	xxxx xxxx	35, 208
TXB1D3	TXB1D37	TXB1D36	TXB1D35	TXB1D34	TXB1D33	TXB1D32	TXB1D31	TXB1D30	xxxx xxxx	35, 208
TXB1D2	TXB1D27	TXB1D26	TXB1D25	TXB1D24	TXB1D23	TXB1D22	TXB1D21	TXB1D20	xxxx xxxx	35, 208
TXB1D1	TXB1D17	TXB1D16	TXB1D15	TXB1D14	TXB1D13	TXB1D12	TXB1D11	TXB1D10	xxxx xxxx	35, 208
TXB1D0	TXB1D07	TXB1D06	TXB1D05	TXB1D04	TXB1D03	TXB1D02	TXB1D01	TXB1D00	xxxx xxxx	35, 208
TXB1DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	-x-- xxxx	35, 209
TXB1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	35, 208
TXB1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	35, 207
TXB1SIDL	SID2	SID1	SID0	—	EXIDE	—	EID17	EID16	xxx- x-xx	35, 207
TXB1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	35, 207
TXB1CON	—	TXABT	TXLARB	TXERR	TXREQ	—	TXPR1	TXPR0	0000 0000	35, 206

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition

Note 1: These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other Oscillator modes.

TABLE 4-2: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on Page:
CANSTATRO4	OPMODE2	OPMODE1	OPMODE0	—	ICODE2	ICODE1	ICODE0	—	xxx- xxx-	33, 202
TXB2D7	TXB2D77	TXB2D76	TXB2D75	TXB2D74	TXB2D73	TXB2D72	TXB2D71	TXB2D70	xxxx xxxx	35, 208
TXB2D6	TXB2D67	TXB2D66	TXB2D65	TXB2D64	TXB2D63	TXB2D62	TXB2D61	TXB2D60	xxxx xxxx	35, 208
TXB2D5	TXB2D57	TXB2D56	TXB2D55	TXB2D54	TXB2D53	TXB2D52	TXB2D51	TXB2D50	xxxx xxxx	35, 208
TXB2D4	TXB2D47	TXB2D46	TXB2D45	TXB2D44	TXB2D43	TXB2D42	TXB2D41	TXB2D40	xxxx xxxx	35, 208
TXB2D3	TXB2D37	TXB2D36	TXB2D35	TXB2D34	TXB2D33	TXB2D32	TXB2D31	TXB2D30	xxxx xxxx	35, 208
TXB2D2	TXB2D27	TXB2D26	TXB2D25	TXB2D24	TXB2D23	TXB2D22	TXB2D21	TXB2D20	xxxx xxxx	35, 208
TXB2D1	TXB2D17	TXB2D16	TXB2D15	TXB2D14	TXB2D13	TXB2D12	TXB2D11	TXB2D10	xxxx xxxx	35, 208
TXB2D0	TXB2D07	TXB2D06	TXB2D05	TXB2D04	TXB2D03	TXB2D02	TXB2D01	TXB2D00	xxxx xxxx	35, 208
TXB2DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	-x-- xxxx	35, 209
TXB2EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	35, 208
TXB2EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	35, 207
TXB2SIDL	SID2	SID1	SID0	—	EXIDE	—	EID17	EID16	xxx- x-xx	35, 207
TXB2SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	35, 207
TXB2CON	—	TXABT	TXLARB	TXERR	TXREQ	—	TXPR1	TXPR0	-000 0-00	35, 206
RXM1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	36, 217
RXM1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	36, 217
RXM1SIDL	SID2	SID1	SID0	—	—	—	EID17	EID16	xxx- --xx	36, 217
RXM1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	36, 216
RXM0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	36, 217
RXM0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	36, 217
RXM0SIDL	SID2	SID1	SID0	—	—	—	EID17	EID16	xxx- --xx	36, 217
RXM0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	36, 216
RXF5EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	36, 216
RXF5EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	36, 216
RXF5SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	36, 215
RXF5SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	36, 215
RXF4EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	36, 216
RXF4EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	36, 216
RXF4SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	36, 215
RXF4SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	36, 215
RXF3EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	36, 216
RXF3EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	36, 216
RXF3SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	36, 215
RXF3SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	36, 215
RXF2EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	36, 216
RXF2EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	36, 216
RXF2SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	36, 215
RXF2SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	36, 215
RXF1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	36, 216
RXF1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	36, 216
RXF1SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	36, 215
RXF1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	36, 215
RXF0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	36, 216
RXF0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	36, 216
RXF0SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xxx- x-xx	36, 215
RXF0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	36, 215

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition

Note 1: These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other Oscillator modes.

4.10 Access Bank

The Access Bank is an architectural enhancement that is very useful for C compiler code optimization. The techniques used by the C compiler are also useful for programs written in assembly.

This data memory region can be used for:

- Intermediate computational values
- Local variables of subroutines
- Faster context saving/switching of variables
- Common variables
- Faster evaluation/control of SFRs (no banking)

The Access Bank is comprised of the upper 160 bytes in Bank 15 (SFRs) and the lower 96 bytes in Bank 0. These two sections will be referred to as Access Bank High and Access Bank Low, respectively. Figure 4-6 indicates the Access Bank areas.

A bit in the instruction word specifies if the operation is to occur in the bank specified by the BSR register, or in the Access Bank.

When forced in the Access Bank (a = 0), the last address in Access Bank Low is followed by the first address in Access Bank High. Access Bank High maps most of the Special Function Registers so that these registers can be accessed without any software overhead.

4.11 Bank Select Register (BSR)

The need for a large general purpose memory space dictates a RAM banking scheme. The data memory is partitioned into sixteen banks. When using direct addressing, the BSR should be configured for the desired bank.

BSR<3:0> holds the upper 4 bits of the 12-bit RAM address. The BSR<7:4> bits will always read '0's, and writes will have no effect.

A MOVLB instruction has been provided in the instruction set to assist in selecting banks.

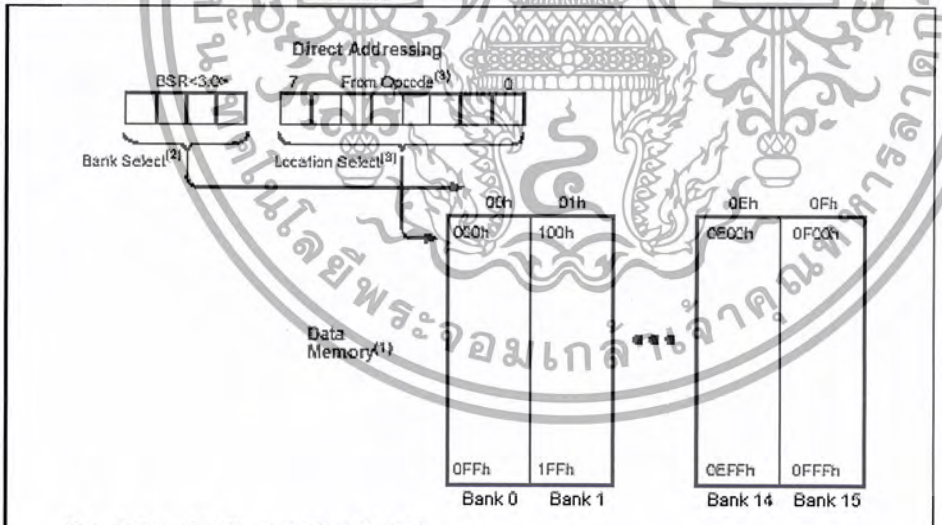
If the currently selected bank is not implemented, any read will return all '0's and all writes are ignored. The STATUS register bits will be set/cleared as appropriate for the instruction performed.

Each Bank extends up to FFh (256 bytes). All data memory is implemented as static RAM.

A MOVFF instruction ignores the BSR, since the 12-bit addresses are embedded into the instruction word.

Section 4.12 provides a description of indirect addressing, which allows linear addressing of the entire RAM space.

FIGURE 4-7: DIRECT ADDRESSING



Note 1: For register file map detail, see Table 4-1.

2: The access bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.

3: The MOVFF instruction embeds the entire 12-bit address in the instruction.

4.12 Indirect Addressing, INDF and FSR Registers

Indirect addressing is a mode of addressing data memory, where the data memory address in the instruction is not fixed. A SFR register is used as a pointer to the data memory location that is to be read or written. Since this pointer is in RAM, the contents can be modified by the program. This can be useful for data tables in the data memory and for software stacks. Figure 4-8 shows the operation of indirect addressing. This shows the moving of the value to the data memory address specified by the value of the FSR register.

Indirect addressing is possible by using one of the INDF registers. Any instruction using the INDF register actually accesses the register indicated by the File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = 0), will read 00h. Writing to the INDF register indirectly, results in a no operation. The FSR register contains a 12-bit address, which is shown in Figure 4-8.

The INDFn ($0 \le n \le 2$) register is not a physical register. Addressing INDFn actually addresses the register whose address is contained in the FSRn register (FSRn is a pointer). This is indirect addressing.

Example 4-5 shows a simple use of indirect addressing to clear the RAM in Bank 1 (locations 100h-1FFh) in a minimum number of instructions.

EXAMPLE 4-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
        LFSR    FSR0, 100h ;
NEXT    CLRF    POSTINC0   ; Clear INDF
                           ; register
                           ; & inc pointer
        BTFSS   FSR0H, 1   ; All done
                           ; w/ Bank1?
        BRA     NEXT       ; NO, clear next
CONTINUE
                           ;
    :                      ; YES, continue
```

There are three indirect addressing registers. To address the entire data memory space (4096 bytes), these registers are 12-bits wide. To store the 12 bits of addressing information, two 8-bit registers are required. These indirect addressing registers are:

1. FSR0: composed of FSR0H:FSR0L
2. FSR1: composed of FSR1H:FSR1L
3. FSR2: composed of FSR2H:FSR2L

In addition, there are registers INDF0, INDF1 and INDF2, which are not physically implemented. Reading or writing to these registers activates indirect addressing, with the value in the corresponding FSR register being the address of the data.

If an instruction writes a value to INDF0, the value will be written to the address indicated by FSR0H:FSR0L. A read from INDF1 reads the data from the address indicated by FSR1H:FSR1L. INDFn can be used in code anywhere an operand can be used.

If INDF0, INDF1 or INDF2 are read indirectly via an FSR, all '0's are read (zero bit is set). Similarly, if INDF0, INDF1 or INDF2 are written to indirectly, the operation will be equivalent to a NOP instruction and the STATUS bits are not affected.

4.12.1 INDIRECT ADDRESSING OPERATION

Each FSR register has an INDF register associated with it, plus four additional register addresses. Performing an operation on one of these five registers determines how the FSR will be modified during indirect addressing.

- When data access is done to one of the five INDFn locations, the address selected will configure the FSRn register to:
  - Do nothing to FSRn after an indirect access (no change) - INDFn
  - Auto-decrement FSRn after an indirect access (post-decrement) - POSTDECn
  - Auto-increment FSRn after an indirect access (post-increment) - POSTINCn
  - Auto-increment FSRn before an indirect access (pre-increment) - PREINCn
  - Use the value in the WREG register as an offset to FSRn. Do not modify the value of the WREG or the FSRn register after an indirect access (no change) - PLUSWn

When using the auto-increment or auto-decrement features, the effect on the FSR is not reflected in the STATUS register. For example, if the indirect address causes the FSR to equal '0', the Z bit will not be set.

Incrementing or decrementing an FSR affects all 12 bits. That is, when FSRnL overflows from an increment, FSRnH will be incremented automatically.

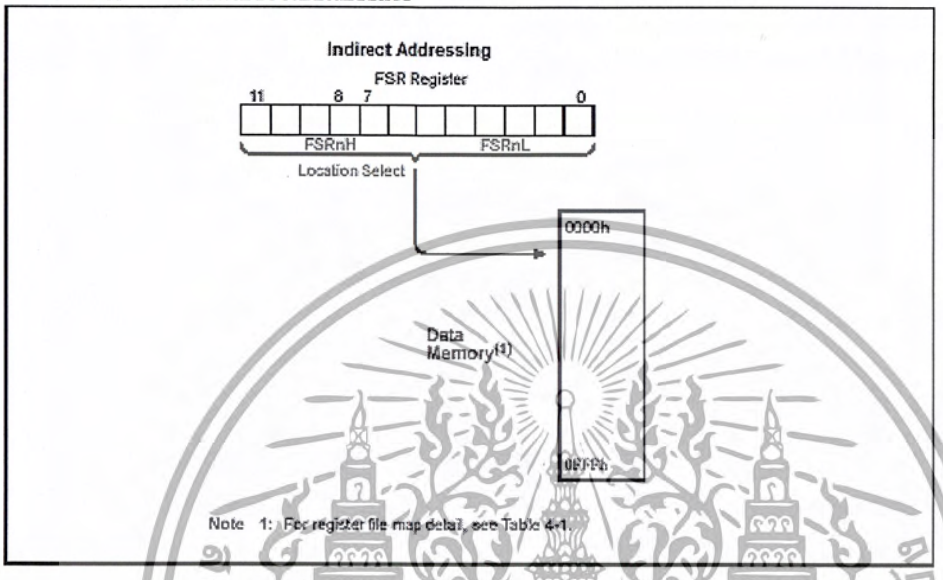
Adding these features allows the FSRn to be used as a software stack pointer, in addition to its uses for table operations in data memory.

Each FSR has an address associated with it that performs an indexed indirect access. When a data access to this INDFn location (PLUSWn) occurs, the FSRn is configured to add the 2's complement value in the WREG register and the value in FSR to form the address before an indirect access. The FSR value is not changed.

If an FSR register contains a value that indicates one of the INDFn, an indirect read will read 00h (zero bit is set), while an indirect write will be equivalent to a NOP (STATUS bits are not affected).

If an indirect addressing operation is done where the target address is an FSRnH or FSRnL register, the write operation will dominate over the pre- or post-increment/decrement functions.

FIGURE 4-8: INDIRECT ADDRESSING



4.13 STATUS Register

The STATUS register, shown in Register 4-2, contains the arithmetic status of the ALU. The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV, or N bits, then the write to these five bits is disabled. These bits are set or cleared according to the device logic. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u uluu` (where `u` = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF`, `MOVFF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C, DC, OV, or N bits from the STATUS register. For other instructions which do not affect the status bits, see Table 26-2.

Note:	The C and DC bits operate as a borrow and digit borrow bit respectively, in subtraction.

REGISTER 4-2: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC	C
bit 7							bit 0

bit 7-5 Unimplemented: Read as '0'

bit 4 N: Negative bit

This bit is used for signed arithmetic (2's complement). It indicates whether the result of the ALU operation was negative (ALU MSb = 1).

1 = Result was negative
0 = Result was positive

bit 3 OV: Overflow bit

This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit 7) to change state.

1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 2 Z: Zero bit

1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 DC: Digit carry/borrow bit

For `ADDWF`, `ADDLW`, `SUBLW`, and `SUBWF` instructions

1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

Note: For borrow, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (`RRCF`, `RRNCF`, `RLCF`, and `RLNCF`) instructions, this bit is loaded with either the bit 4 or bit 3 of the source register.

bit 0 C: Carry/borrow bit

For `ADDWF`, `ADDLW`, `SUBLW`, and `SUBWF` instructions

1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For borrow, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.14 RCON Register

The Reset Control (RCON) register contains flag bits that allow differentiation between the sources of a device RESET. These flags include the $\overline{TO}$, $\overline{PD}$, $\overline{POR}$, $\overline{BOR}$ and $\overline{RI}$ bits. This register is readable and writable.

Note 1: If the BOREN configuration bit is set, $\overline{BOR}$ is '1' on Power-on Reset. If the BOREN configuration bit is clear, $\overline{BOR}$ is unknown on Power-on Reset. The $\overline{BOR}$ status bit is a "don't care" and is not necessarily predictable if the brown-out circuit is disabled (the BOREN configuration bit is clear). $\overline{BOR}$ must then be set by the user and checked on subsequent RESETS to see if it is clear, indicating a brown-out has occurred.

2: It is recommended that the $\overline{POR}$ bit be set after a Power-on Reset has been detected, so that subsequent Power-on Resets may be detected.

REGISTER 4-3: RCON REGISTER

R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (16CXXX Compatibility mode)

bit 6-5 **Unimplemented: Read as '0'**

bit 4 $\overline{RI}$: RESET Instruction Flag bit
1 = The RESET instruction was not executed
0 = The RESET instruction was executed causing a device RESET (must be set in software after a Brown-out Reset occurs)

bit 3 $\overline{TO}$: Watchdog Time-out Flag bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 2 $\overline{PD}$: Power-down Detection Flag bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 1 $\overline{POR}$: Power-on Reset Status bit
1 = A Power-on Reset has not occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 $\overline{BOR}$: Brown-out Reset Status bit
1 = A Brown-out Reset has not occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

ประวัติผู้แต่ง

ชื่อ-สกุล นายอภิวัธน์ ศรีบรรเทา

วัน เดือน ปีเกิด 8 กุมภาพันธ์ พ.ศ. 2524

ภูมลำเนา 110 ม.2 ต.สำราญ กิ่ง อ.สามชัย จ.กาฬสินธุ์ 46180

โทรศัพท์ 0-4381-8171

ประวัติการศึกษา

ประถมศึกษา โรงเรียนบ้านหนองกุงใหญ่ จังหวัดกาฬสินธุ์

มัธยมศึกษาตอนต้น โรงเรียนสามชัย จังหวัดกาฬสินธุ์

ประกาศนียบัตรวิชาชีพ วิทยาลัยเทคนิคกาฬสินธุ์

ประกาศนียบัตรวิชาชีพชั้นสูง วิทยาลัยเทคนิคกาฬสินธุ์

ปริญญาตรี สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

ผลงานที่เคยได้รับ รางวัลผ่านเกณฑ์มาตรฐานการแข่งขันทักษะวิชาชีพระดับภาค

ตะวันออกเฉียงเหนือ ปี 2544

คติพจน์ ทำดีได้ดี ทำวันนี้ให้ดีที่สุด

ประวัติผู้แต่ง

ชื่อ-สกุล นายถาวร แสงใส

วัน เดือน ปีเกิด 2 พฤษภาคม พ.ศ. 2524

ภูมลำเนา 19 ม.17 ต.หนองปลาไหล กิ่ง อ.เมือง จ.สระบุรี 18000

โทรศัพท์ 0-9779-3323

ประวัติการศึกษา

ประถมศึกษา โรงเรียนมัธยมเขาจำปา

มัธยมศึกษาตอนต้น โรงเรียนมัธยมเขาจำปา

ประกาศนียบัตรวิชาชีพ วิทยาลัยเทคนิคสระบุรี

ประกาศนียบัตรวิชาชีพชั้นสูง วิทยาลัยเทคนิคสระบุรี

ปริญญาตรี สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

ผลงานที่เคยได้รับ -

คติพจน์ -

ประวัติผู้แต่ง

ชื่อ-สกุล นายคมสันต์ แวงวรรณ

วัน เดือน ปีเกิด 23 มีนาคม พ.ศ. 2524

ภูมลำเนา 52 ม.17 ต.โพธิ์ทอง อ.โพนทอง จ.ร้อยเอ็ด 45110

โทรศัพท์ 0-9608-0161

ประวัติการศึกษา

ประถมศึกษา โรงเรียนบ้านโพนทอง

มัธยมศึกษาตอนต้น โรงเรียนโพนทองพัฒนาวิทยา

ประกาศนียบัตรวิชาชีพ วิทยาลัยการอาชีพโพนทอง

ประกาศนียบัตรวิชาชีพชั้นสูง วิทยาลัยเทคนิคยโสธร

ปริญญาตรี สาขาวิชาอิเล็กทรอนิคส์และคอมพิวเตอร์

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

ผลงานที่เคยได้รับ รางวัลชนะเลิศการแข่งขันทักษะวิชาชีพ ปี 2540

คติพจน์ ตนเป็นที่พึ่งแห่งตน