ชุดทดลองการเขียนโปรแกรมควบคุมไมโครคอนโทรเลอร์ด้วยภาษา C

PROGRAMMING THE MICROCONTROLLER BASED WITH C

นางสาวจามิกา บูรภาสถิตย์

นายณัฐวุฒิ หงษ์ศิริ

โครงงานพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

PROGRAMMING THE MICROCONTROLLER BASED WITH C

MS. JAMIKA BURAPASATIT

MR. NATTAWUT HONGSIRI

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIRMENT FOR DEGREE OF BACHELOR OF SCIENCE

IN COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

หัวข้อปัญหาพิเศษ	ชุดทดลองการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ด้วยภาษา C PROGRAMMING THE MICROCONTROLLER BASED WITH C
ชื่อนักศึกษา	นางสาวจามิกา บูรภาสถิตย์ 50050107 นายณัฐวุฒิ หงษ์ศิริ 50050131
ปริญญา	วิทยาศาสตรบัณฑิต
ภาควิชา	วิทยาการคอมพิวเตอร์
สาขาวิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2553
อาจารย์ที่ปรึกษา	รศ. ธีรวัฒน์ ประกอบผล

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2553

คณะกรรมการสอบ	ลายมือชื่อ
อ.ศังกรศรัณย์ ล่องชูผล ประธานกรรมการ	
ดร.อนันตพร ศรีสวัสดิ์ กรรมการ	
รศ.ธีรวัฒน์ ประกอบผล กรรมการและอาจารย์ที่ปรึกษา	

หัวข้อปัญหาพิเศษ	ชุดทดลองการเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ด้วยภาษา C
ชื่อนักศึกษา	นางสาวจามิกา บูรภาสถิตย์ 50050107
	นายณัฐวุฒิ หงษ์ศิริ 50050131
ปริญญา	วิทยาศาสตรบัณฑิต
ภาควิชา	วิทยาการคอมพิวเตอร์
สาขาวิชา	วิทยาการคอมพิวเตอร์
ปีการศึกษา	2553
อาจารย์ที่ปรึกษา	รศ. ธีรวัฒน์ ประกอบผล

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอการเขียนโปรแกรมควบคุมไมโครคอนโทรเลอร์ด้วยภาษา C โดยแสดงผลผ่านอุปกรณ์ต่างๆ เช่น หลอด LED LED 7 ส่วน ระบบฐานเวลาจริง และตัววัดอุณหภูมิ ซึ่งแผงวงจรไมโครคอนโทรลเลอร์หรือชุดทดลองที่ได้สร้างขึ้นมานี้ เรียกว่า OBEC-BOX V.1.0 ซึ่งทางคณะผู้จัดทำได้รับทุนส่วนหนึ่งจากทางสำนักงานคณะกรรมการศึกษาขั้นพื้นฐาน (สพฐ.) ในการสร้างชุดทดลองในการที่จะแสดงผลผ่านอุปกรณ์ต่างๆที่กล่าวมาข้างต้นได้นั้น ต้องประกอบด้วย วงจรที่ควบคุมการแสดงผลแต่ละส่วน ชิปที่ใช้ควบคุมการแสดงผลแต่ละตัว เช่น MAX7219 DS18S20 และ DS1307 เพื่อลดความซับซ้อนของโปรแกรม จึงมีการสร้างฟังก์ชันต่างๆ ในการเรียกใช้งานให้ง่ายมากขึ้น

คำสำคัญ : ไมโครคอนโทรลเลอร์, หลอด LED, LED 7 ส่วน, ระบบฐานเวลาจริง, ตัววัดอุณหภูมิ

Title	PROGRAMMING THE MICROCONTROLLER BASED WITH C
Students	Ms.Jamika Burapasatit 50050107
	Mr.Nattawut Hongsiri 50050131
Degree	Bechelor of Science
Department	Computer Science, Faculty of Science
Programme	Computer Science
Academic Year	2010
Advisor	Assoc.Prof. Teerawat Prakobphon

ABSTRACT

This thesis presents the microcontroller programming based with C language by display on device such as Light-Emitting Diode (LED), 7-Segment LED, Real-time clock and Temperature sensor. The microcontroller circuit or experimental kit that was created is called OBEX-BOX V.1.0 which the team received a grant from The Office of The Basic Education Commission (OBEC). In an experimental kit in the order to display on various device according to the mention above it. Must be include a circuit that controls the display segment, the chip used to control each display such as MAX7219, DS1307 to reduce the complexity of program. Therefore, it must have a function for easier to call when running a program.

Keywords : microcontroller, LED(Light Emitting Diode), LED 7 segment, real time clock, temperature sensor

กิตติกรรมประกาศ

ในการจัดทำคู่มือการทำปัญหาพิเศษนี้สำเร็จลุล่วงได้ด้วยดี เนื่องจากได้รับความช่วยเหลือและสนับสนุนจากอาจารย์หลายท่าน รศ.ธีรวัฒน์ ประกอบผล กรรมการและอาจารย์ที่ปรึกษา ซึ่งเป็นผู้เสียสละเวลาให้คำแนะนำและชี้แนะแนวทางต่างๆในการแก้ปัญหาจนโครงงานปัญหาพิเศษนี้สำเร็จลุล่วงไปได้ด้วยดี อาจารย์ศังกรณ์ศรัณย์ ล่องชูผล และ ดร.อนันตพร ศรีสวัสดิ์ ประธานกรรมการ และกรรมการ ซึ่งเป็นผู้ให้คำแนะนำและชี้จุดบกพร่องที่ควรแก้ไข ทางคณะผู้จัดทำจึงขอกราบขอบพระคุณเป็นอย่างยิ่งในความกรุณาของทุกท่านไว้ ณ ที่นี้

สุดท้ายนี้ขอขอบพระคุณคณาอาจารย์ในภาควิชา วิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ ซึ่งได้ให้ความรู้ทางวิชาการ จนกระทั่งผู้จัดทำพอมีความสามารถที่จะดำเนินปัญหาพิเศษสำเร็จลุล่วงได้เช่นนี้ ขอบพระคุณทุกท่านจากใจจริง

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ(ต่อ)

สารบัญภาพ(ต่อ)

สารบัญภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

วิชาวิทยาศาสตร์เป็นวิชาที่ทำให้ผู้เรียนเข้าใจกฎเกณฑ์ต่างๆทางธรรมชาติ สามารถอธิบายสิ่งต่างๆที่อยู่รอบตัวเราได้ ความรู้ที่ได้รับจากวิชานี้ทำให้นำไปสู่การพัฒนาสิ่งต่างๆรอบตัวเรารวมไปถึงเทคโนโลยีสมัยใหม่มากมาย ในปัจจุบันพบว่า นักเรียนสนใจเรียนวิชาด้านนี้น้อยลง ทำให้เทคโนโลยีหลายๆอย่างที่ใช้กันอยู่ในปัจจุบันเป็นเทคโนโลยีที่ซื้อมาจากต่างประเทศ ซึ่งมีเทคโนโลยีหลายๆอย่างที่สามารถพัฒนาได้เองในเมืองไทย โดยการพัฒนาดังกล่าวจะต้องทำอย่างเป็นระบบ ต้องปลูกฝังให้เยาวชนรักที่จะเป็นผู้สร้างวิทยาการด้านวิทยาศาสตร์และเทคโนโลยี โดยการสร้างเยาวชนของชาติให้มีความรู้ความสามารถทางวิทยาศาสตร์เทคโนโลยีเป็นสิ่งที่ต้องเริ่มตั้งแต่เยาว์วัย โดยอาศัยการศึกษาเป็นรากฐาน เพื่อนำไปสู่การมีเทคโนโลยีเป็นของตนเอง

ในประเทศไทยได้มีกิจกรรมที่กระตุ้นให้นักเรียนอยากเรียนรู้วิทยาศาสตร์มากมาย บางกิจกรรมจะกระตุ้นให้นักเรียนมีจินตนาการต่อออกไปจากความรู้เดิมที่มีอยู่ เช่น สิ่งประดิษฐ์ต่างๆ การทำโครงงานด้านหุ่นยนต์ การทำโครงงานวิทยาศาสตร์ ซึ่งโครงงานเหล่านี้นอกจากจะช่วยพัฒนาความคิดให้กับผู้เรียนแล้ว ยังสามารถนำไปใช้สำหรับเสริมความรู้ด้านต่างๆได้อีกด้วย ถ้าหากมีการทำสื่อประกอบโครงงาน หรือมีการอธิบายหลักการต่างๆอย่างถูกวิธี

โครงงานด้านวิทยาศาสตร์ที่ได้รับความนิยมในขณะนี้คือ โครงงานที่ใช้ระบบคอมพิวเตอร์มาควบคุม เยาวชนและครูผู้ดูแลที่จะทำโครงงานลักษณะนี้ได้จะต้องมีความรู้ทางวิทยาศาสตร์ และการเขียนโปรแกรมคอมพิวเตอร์เป็นอย่างดี บางคนมีความรู้ทางวิทยาศาสตร์ดี แต่ไม่มีความสามารถทางด้านการเขียนโปรแกรมก็มีมาก ถ้าหากพัฒนาสิ่งประดิษฐ์ที่ช่วยเสริมในเรื่องของการเขียนโปรแกรมก็จะทำให้ผู้ศึกษาสนใจและสนุกกับการเขียนโปรแกรมมากขึ้น สำหรับนักเรียนที่ถนัดทางการเขียนโปรแกรมก็มีจำนวนมากที่ไม่ทราบว่าการเขียนโปรแกรมคอมพิวเตอร์สามารถนำไปประยุกต์ใช้กับอุปกรณ์ต่างๆได้ง่าย ถ้าหากนักเขียนโปรแกรมมีความรู้ทางด้านวิทยาศาสตร์ดีจะสามารถพัฒนาโปรแกรมให้ทำงานร่วมกับอุปกรณ์ทางวิทยาศาสตร์ได้ ทำให้เกิดโครงงานและสิ่งประดิษฐ์ต่างๆขึ้นอีกมากมาย ดังนั้นควรมีการสนับสนุนให้นักเรียนที่เขียนโปรแกรมคอมพิวเตอร์เป็นอยู่แล้วได้รับความรู้ในการนำโปรแกรมมาประยุกต์ใช้กับอุปกรณ์ทางวิทยาศาสตร์ ได้ฝึกเขียนโปรแกรมกับอุปกรณ์ที่สามารถเชื่อมโยงกับงานวิทยาศาสตร์ได้ ก็จะทำให้เด็กเข้าใจวิทยาศาสตร์มากขึ้น แต่ชุดทดลองต่างๆยังมีราคาแพงอยู่ และส่วนใหญ่ก็ไม่ได้มีการอธิบายหลักการทางวิทยาศาสตร์ให้เหมาะสมกับเยาวชนไทย ดังนั้นคณะผู้วิจัยจึงมีความตั้งใจที่จะพัฒนาชุดทดลองลักษณะนี้ขึ้น

สำหรับผู้ที่เริ่มสนใจการเขียนโปรแกรมคอมพิวเตอร์ บางครั้งการเขียนโปรแกรมให้ประมวลผลโดยเครื่องคอมพิวเตอร์แล้วแสดงผลลัพธ์ออกทางจอภาพอย่างเดียวอาจทำให้ผู้เรียนรู้สึกเบื่อ ไม่สนุก ถ้าหากทำให้โปรแกรมที่เขียนขึ้นสามารถแสดงผลทางอุปกรณ์อื่นๆ เช่น หลอดไฟ การหมุนของมอเตอร์ ก็จะทำให้ผู้เรียนสนุกมากขึ้น เช่น ออกแบบโปรแกรมให้แสดงผลเป็นไฟวิ่งรูปแบบต่างๆ ออกแบบโปรแกรมให้สามารถควบคุมการหมุนของมอเตอร์ได้ ทำให้สามารถพัฒนาต่อเป็นโครงการหุ่นยนต์ลักษณะต่างๆได้

สำหรับผู้ที่สนใจด้านการทำโครงงานคอมพิวเตอร์ หรือสนใจทางด้านเทคโนโลยีหุ่นยนต์ ในโครงงานที่ทำขึ้นจะต้องมีส่วนของสมองกล หรือไมโครคอนโทรลเลอร์เป็นตัวควบคุม และปัจจุบันไมโครคอนโทรลเลอร์ที่ผลิตออกมาสามารถใช้งานง่าย มีพอร์ตสำหรับเชื่อมต่อกับอุปกรณ์ภายนอกได้หลายลักษณะ แต่การออกแบบระบบไมโครคอนโทรลเลอร์นั้นจะต้องมีความรู้ทางด้านวิศวกรรม มีความรู้ทางดิจิตอลอิเล็กทรอนิกส์ ทำให้นักเรียนส่วนใหญ่ไม่สามารถออกแบบระบบควบคุมขึ้นมาเองได้ การทำโครงงานต่างๆที่ผ่านมาโรงเรียนต่างๆจึงต้องซื้อแผงวงจรไมโครคอนโทรลเลอร์ที่มีในท้องตลาดมาใช้ ซึ่งมีทั้งของในประเทศไทยและของที่ผลิตจากต่างประเทศ และโครงงานในลักษณะนี้กำลังได้รับความนิยมมากขึ้น คณะผู้วิจัยจึงมีความคิดที่จะผลิตแผงวงจรไมโครคอนโทรลเลอร์ขึ้นมาเอง เพื่อเป็นต้นแบบให้กับโรงเรียนต่างๆสามารถนำไปใช้หรือผลิตขึ้นมาเองได้

ไมโครคอนโทรลเลอร์ในปัจจุบันได้รวมพอร์ตต่างๆไว้ภายในมากมาย มีวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลที่สามารถนำไปใช้งานได้ง่าย แผงวงจรไมโครคอนโทรลเลอร์ที่คณะผู้วิจัยจะสร้างขึ้นนั้นจะออกแบบให้สามารถเขียนโปรแกรมร่วมกับอุปกรณ์ทางวิทยาศาสตร์ได้ โดยแผงวงจรนี้จะออกแบบขึ้นมาให้มีลักษณะเฉพาะเพื่อนำไปใช้เป็นเทคโนโลยีพื้นฐานในการทำโครงงานด้านหุ่นยนต์ และโครงงานด้านวิทยาศาสตร์ต่อไป

1.2 วัตถุประสงค์

1) ศึกษาการใช้งานไมโครคอนโทรลเลอร์กับระบบสมองกลฝังตัว (Embedded System)
2) ศึกษาและทำความเข้าใจการทำงานของไมโครคอนโทรลเลอร์
3) รู้จักวิธีการเขียนโปรแกรมควบคุมฮาร์ดแวร์

1.3 ข้อจำกัดและขอบเขต

ออกแบบชุดทดลองต้นแบบให้เขียนโปรแกรมควบคุมด้วยภาษา C ได้

1.4 ขั้นตอนการดำเนินการ

การทำงานสามารถแบ่งเป็นช่วงหลักๆได้ดังนี้

1) ศึกษาข้อมูล
   - ศึกษาโครงสร้างของไมโครคอนโทรลเลอร์และอุปกรณ์ประกอบ
   - ศึกษาวิธีการเขียนโปรแกรมภาษา C สำหรับควบคุมไมโครคอนโทรลเลอร์
   - ศึกษาการสร้างวงจรอิเล็กทรอนิกส์กับไมโครคอนโทรลเลอร์
2) กำหนดขอบเขตของโครงงาน
   - กำหนดโครงสร้างและการทำงานของชุดทดลอง
   - ออกแบบชุดทดลอง
3) สร้างชุดทดลอง
4) เขียนโปรแกรม
   - โปรแกรมควบคุมไมโครคอนโทรลเลอร์
   - ทดสอบการทำงาน
5) ทดสอบระบบโดยรวม

1.5 ประโยชน์ที่คาดว่าจะได้รับ

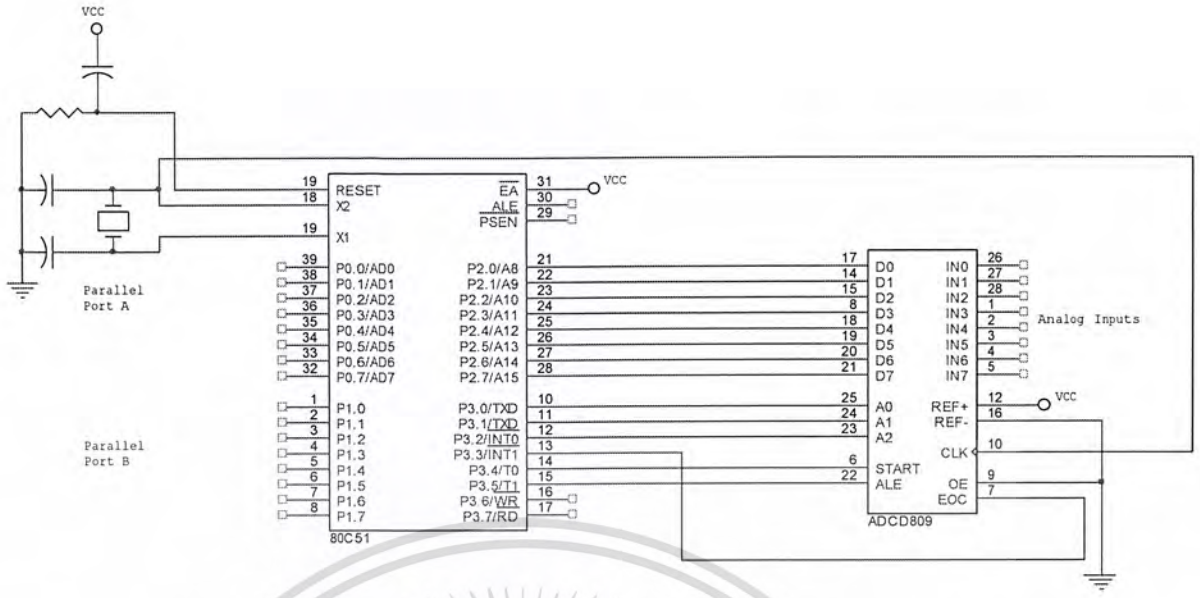
1) มีความรู้ในการออกแบบชุดทดลองที่เขียนโปรแกรมควบคุมด้วยภาษา C ได้
2) สามารถนำไมโครคอนโทรลเลอร์ไปใช้กับระบบสมองกลฝังตัว (Embedded System) อื่นๆได้
3) มีความรู้ความเข้าใจในการออกแบบการทำงานที่สอดคล้องกัน ระหว่างฮาร์ดแวร์และซอฟต์แวร์
4) สามารถเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์กับงานอื่นๆได้

บทที่ 2
ไมโครคอนโทรลเลอร์และการเขียนโปรแกรม

อุปกรณ์อิเล็กทรอนิกส์ในปัจจุบันจะถูกควบคุมด้วยระบบคอมพิวเตอร์เล็ก ๆ หรือที่เรียกว่าไมโครโพรเซสเซอร์เกือบทั้งสิ้น ในปัจจุบันจะเรียกชื่อเป็นศัพท์เทคนิคว่า Embedded System หรือระบบสมองกลฝังตัว ระบบควบคุมด้วยคอมพิวเตอร์จะต้องมีหน่วยประมวลผลกลางที่เรียกว่าไมโครโพรเซสเซอร์เป็นหัวใจหลักในการทำงาน โดยบริษัทอินเทลได้สร้างไมโครโพรเซสเซอร์เบอร์ 4004 ซึ่งประมวลผลแบบ 8 บิต ออกมาเป็นรุ่นแรก ต่อมาได้ออกรุ่นที่ประมวลผลแบบ 8 บิต ตามมาได้แก่เบอร์ 8008, 8080 และ 8085 ทำให้การประมวลผลทำได้รวดเร็วขึ้น ส่วนบริษัทโมโตโลราได้ออกเบอร์ 6800 และบริษัทไซลอกได้ออกเบอร์ Z80 ซึ่งจะประมวลผลแบบ 8 บิต เช่นกัน ไมโครโพรเซสเซอร์รุ่นต่อ ๆ มา จะมีประสิทธิภาพในการทำงานมากขึ้น และได้มีรุ่นใหม่ ๆ ออกตามมามากขึ้น

เมื่อนำไมโครโพรเซสเซอร์แบบ 4 บิต มาใช้ในงานควบคุมจะทำให้ระบบควบคุมทำงานได้ดีขึ้น ฉลาดมากขึ้น ปัจจุบันไมโครโพรเซสเซอร์แบบ 4 บิต นี้ถูกนำมาใช้ในเตาไมโครเวฟ โทรทัศน์ และของเด็กเล่น เป็นต้น สำหรับระบบควบคุมที่ต้องการประสิทธิภาพมากขึ้นจะใช้ไมโครโพรเซสเซอร์แบบ 8 บิต เป็นตัวประมวลผล แต่ราคาของระบบก็จะแพงขึ้นตามไปด้วย

การนำไมโครโพรเซสเซอร์มาใช้งานจะต้องมีการเขียนโปรแกรมควบคุมการทำงานและมีหน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลที่ได้จากการประมวลผล และต้องมีพอร์ต I/O สำหรับให้ระบบติดต่อกับอุปกรณ์ภายนอก ตัวอย่างเช่น ถ้านำไมโครโพรเซสเซอร์เบอร์ 8085 มาออกแบบเป็นระบบให้ทำงานงานหนึ่งจะต้องนำไอซีต่าง ๆ มาต่อเพิ่มเข้าไป เช่น ใช้หน่วยความจำสองตัวคือเบอร์ 2764 เป็นหน่วยความจำ ROM สำหรับเก็บโปรแกรม และเบอร์ 6264 เป็นหน่วยความจำ RAM สำหรับเก็บข้อมูลขณะที่ไมโครโพรเซสเซอร์ประมวลผล ส่วน 8255 เป็นพอร์ตขนานสำหรับต่อกับอุปกรณ์ภายนอกจำนวน 3 พอร์ต และยังมีไอซีเบอร์ ADC0809 สำหรับแปลงสัญญาณแอนะล็อกจากภายนอกให้เป็นสัญญาณดิจิทัลให้ไมโครโพรเซสเซอร์ประมวลผล แต่ถ้าหากใช้ไมโครคอนโทรลเลอร์จะมีหน่วยความจำและพอร์ตอยู่ภายในตัวชิป ที่เรียกว่า single-chip microcontroller ทำให้การใช้งานทำได้เพียงแต่ต่ออุปกรณ์ที่จำเป็นภายนอกเท่านั้น บางครั้งไมโครคอนโทรลเลอร์จะถูกเรียกว่าคอมพิวเตอร์ชิปเดี่ยว (single-chip microcontroller) ดังตัวอย่างวงจรในรูปที่ 2.1 เป็นการออกแบบระบบไมโครคอนโทรลเลอร์ให้ทำงานคล้ายกับระบบดังรูปที่ 1.1 ซึ่งจะเห็นว่าใช้อุปกรณ์น้อยมาก



รูปที่ 2.1 การนำไมโครคอนโทรลเลอร์มาต่อกับตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล

ไมโครคอนโทรลเลอร์หรือคอมพิวเตอร์แบบชิปเดี่ยวเป็นอุปกรณ์ที่นิยมใช้ในระบบควบคุมต่าง ๆ ภายในประกอบด้วยไมโครโปรเซสเซอร์ หน่วยความจำโปรแกรม หน่วยความจำข้อมูล และอุปกรณ์อินพุตเอาต์พุต นอกจากนี้ยังมีส่วนประกอบอื่น ๆ ให้ใช้ด้วย เช่น ตัวนับตัวจับเวลา ตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล ในปัจจุบันไมโครคอนโทรลเลอร์ผลิตออกมาหลายรุ่นหลายตระกูล สามารถเลือกมาใช้ได้ตามต้องการ สำหรับไมโครคอนโทรลเลอร์ตระกูลพื้นฐานที่นิยมใช้ในการศึกษาด้านระบบสมองกลฝังตัวคือตระกูล MCS-51 ไมโครคอนโทรลเลอร์ตระกูลนี้จะมีพอร์ตขนาด 8 บิตอยู่ภายใน มีชื่อว่า P0, P1, P2 และ P3 สามารถนำมาโปรแกรมให้เป็นพอร์ตอินพุตและเอาต์พุตได้

2.1 ไมโครคอนโทรลเลอร์ตระกูล MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 หรือ 8051 เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต มีผลิตออกมาหลายบริษัทแต่ละรุ่นมีประสิทธิภาพที่แตกต่างกัน เบอร์ที่เป็นพื้นฐานคือเบอร์ 8051 มี ขา 40 ขา เบอร์ 89C51 เป็นไอซีที่มีโครงสร้างเป็น CMOS ที่ประหยัดพลังงาน เบอร์ 8751 เป็นเบอร์ที่มี EPROM สำหรับเก็บโปรแกรมอยู่ภายในแต่การลบข้อมูลเก่าออกจะต้องแสงอัลตร้าไวโอเลตในการลบ สำหรับรุ่น 8052 จะมีหน่วยความจำโปรแกรมขนาดใหญ่ขึ้น มีตัวนับตัวจับเวลาให้ใช้งานมากกว่าเบอร์อื่น ๆ ส่วนเบอร์ AT89C1051 และ AT89C2051 ของบริษัท Atmel เป็นตระกูล MCS-51 ที่มีขา 20 ขา เหมาะสำหรับงานเล็ก ๆ ต้องการใช้พอร์ตไม่มาก และใช้พลังงานไฟฟ้าน้อยกว่ารุ่น

ที่มีขา 40 ขา สำหรับปัจจุบันนิยมใช้รุ่นที่มีหน่วยความจำแบบ Flash สำหรับเก็บโปรแกรมอยู่ภายใน

การเขียนโปรแกรมให้กับไมโครคอนโทรลเลอร์นั้นยุคแรก ๆ จะต้องเขียนด้วยภาษาแอสเซมบลี จากนั้นแปลโปรแกรมที่ได้ให้เป็นรหัสภาษาเครื่องแล้วจึงโหลดลงหน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์ ต่อมาได้มีตัวแปลภาษาซีสำหรับไมโครคอนโทรลเลอร์เบอร์ต่าง ๆ ทำให้ผู้พัฒนาสามารถเขียนโปรแกรมด้วยภาษาซีได้ จากนั้นจึงแปลให้เป็นรหัสภาษาเครื่องแล้วจึงโหลดลงสู่หน่วยความจำ เมื่อเขียนด้วยภาษาซีได้ทำให้การพัฒนางานด้านสมองกลฝังตัวทำได้ง่ายขึ้น แต่ในยุคแรก ๆ การโปรแกรมลงสู่หน่วยความจำทำได้ยาก เนื่องจากต้องใช้เครื่องมือเฉพาะในการโปรแกรม แต่ปัจจุบันสามารถเขียนโปรแกรมและโหลดลงสู่ตัวชิปได้ง่ายขึ้น เนื่องจากชิปรุ่นใหม่ ๆ จะมีโปรแกรมมอนิเตอร์สำหรับสื่อสารข้อมูลแบบอนุกรมกับคอมพิวเตอร์อยู่ในตัวชิป ไมโครคอนโทรลเลอร์ของบริษัท Megawin เป็นไมโครคอนโทรลเลอร์ตระกูล MCS-51 อีกตัวหนึ่งที่นำมาใช้งานได้ง่าย

ในบทนี้จะกล่าวถึงโครงสร้างและการทำงานพื้นฐานของไมโครคอนโทรลเลอร์ โดยจะเน้นไปที่ชิปเบอร์ MPC82G516A ของบริษัท Megawin ที่นำมาใช้ในการสร้างชุดทดลองในโครงงานนี้ ซึ่งเป็นชิปที่มีราคาไม่แพงและสามารถเขียนโปรแกรมเข้าไปในตัวชิปได้โดยตรง โดยจะเริ่มอธิบายถึง ภาพรวมของ MPC82G516A ส่วนประกอบหลักๆ ซึ่งมีโครงสร้างภายในคล้ายกับ MCS-51 มาตรฐาน แต่จะมีคุณสมบัติที่เพิ่มเติมขึ้นมา เช่น มีหน่วยความจำภายในแบบแฟลช การโปรแกรมแบบ ISP(In-System Programming) และสุดท้ายจะอธิบายถึงการทำงานพื้นฐานของ MPC82G516A

2.2 ภาพรวมของ MPC82G516A

MPC82G516A เป็นไมโครคอนโทรลเลอร์ชิปเดี่ยว สามารถประมวลผลคำสั่งได้ใน 1-7 รอบสัญญาณนาฬิกา และมีชุดคำสั่งที่ทำงานเหมือนกับ 8051 เพราะฉะนั้นจึงมีลักษณะต่างๆที่เป็นมาตรฐานเหมือน 8051 แต่ชิป MPC82G516A สามารถทำงานได้ที่ความเร็วต่ำได้ ใช้แรงดันไฟฟ้าต่ำว่าได้ และด้วยเหตุนี้พลังงานที่ถูกใช้ก็จะลดน้อยลงไปด้วย

MPC82G516A มีหน่วยความจำภายในแบบแฟลช(Flash)ขนาด 64 กิโลไบต์ สำหรับเก็บโปรแกรมและข้อมูล โดยโปรแกรมได้แบบ In-system Programming(ISP) ซึ่งบนชิปจะมีโปรแกรม Boot Loader บรรจุอยู่มาตั้งแต่การผลิตมาจากโรงงาน และแบบ In-Circuit Programming(ICP) สามารถที่จะทำงานได้ทั้งแบบขนาน (parallel) และอนุกรม(serial) และยังมีความสามารถในการโปรแกรมลงบนไมโครคอนโทรลเลอร์แบบ In-Application Programming(IAP) ISP และ ICP อนุญาตให้ผู้ใช้สามารถเขียนหรือลบโปรแกรมได้โดยตรง โดยไม่ต้องถอดชิพออกจากแผงวงจร

สำหรับ IAP หมายถึง ความสามารถในการโปรแกรมลงในหน่วยความจำภายในแบบแฟลช ขณะที่โปรแกรมกำลังทำงานอยู่

ส่วนประกอบหลักของ MPC82G516A ได้แก่ หน่วยความจำ scratch-pad RAM ขนาด 256 ไบต์ พอร์ต I/O ขนาด 8 บิต 4 พอร์ต ตัวจับเวลา/ตัวนับ 3 ตัว พอร์ตอนุกรมซึ่งสามารถรับส่งข้อมูลแบบฟลูดูเพล็กซ์ วงจรอินเทอร์รัปต์ นอกจากนี้ยังมีหน่วยความจำแรมเพิ่มเติม(XRAM) ขนาด 1024 ไบต์ พอร์ต I/O แบบพิเศษ(P4) ตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล(ADC) ขนาด 10 บิต โปรแกรมวงจรพิเศษ(PCA) การสื่อสารแบบอนุกรม(SPI) การเชื่อมต่อผ่านพอร์ตอนุกรมกับคอมพิวเตอร์(UART) keypad interrupt วอตช์ดอกไทเมอร์(Watchdog Timer) และอื่นๆอีก นอกจากนี้ MPC82G516A ยังกลายเป็นไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพ เหมาะสำหรับแอพพลิเคชั่นที่หลากหลาย

MPC82G516A มีโหมดประหยัดพลังงาน 2 โหมด และการกำหนดระบบสัญญาณนาฬิกาเพื่อลดการใช้พลังงาน ใน Idle mode CPU จะไม่ทำงาน ในขณะที่การทำงานของอุปกรณ์ต่อพ่วงอื่นๆและการขัดจังหวะยังคงทำงานอยู่ ใน Power-Down mode ค่าของ RAM และ SFRs' จะถูกเก็บเอาไว้ และจะหยุดการทำงานทั้งหมด ใน Power-Down mode สิ่งสำคัญที่สุด คือสามารถรับสัญญาณอินเทอร์รัปต์จากภายนอกได้ ทำให้ผู้ใช้สามารถลดการใช้พลังงานได้มากขึ้น

2.3 ส่วนประกอบของ MPC82G516A

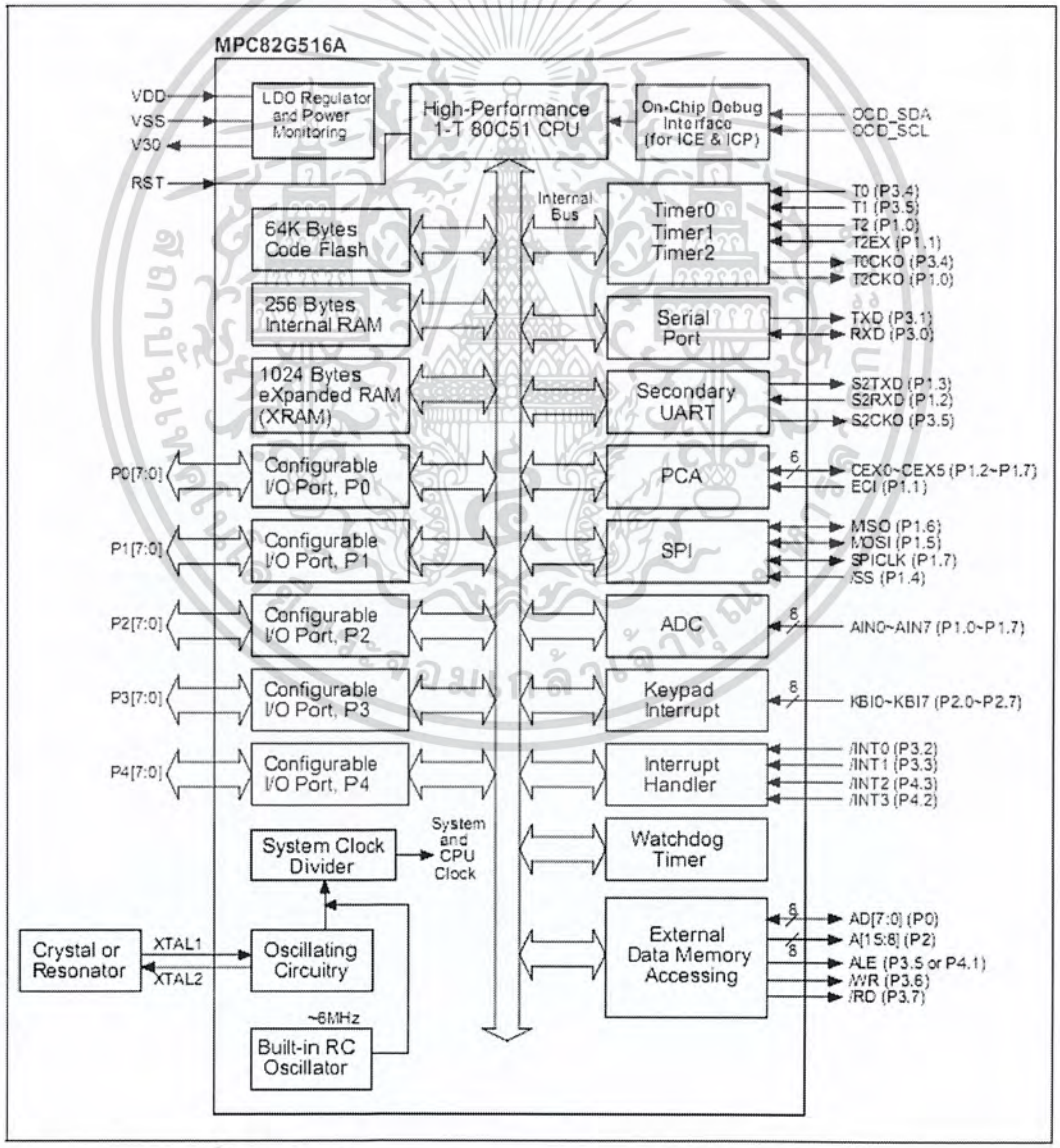
- โครงสร้างโดยทั่วไปที่สอดคล้องกับไมโครคอนโทรลเลอร์เบอร์ 8051 มีดังนี้
  - สามารถทำงานกับชุดคำสั่งของ 8051 ได้
  - มีหน่วยความจำภายใน(Internal Scratch-pad RAM) ขนาด 256 ไบต์
  - พื้นที่หน่วยความจำภายนอก 64 กิโลไบต์
  - พอร์ต I/O ขนาด 8 บิตรับส่งข้อมูลได้สองทิศทาง 4 พอร์ต
  - ตัวจับเวลา/ตัวนับ(Timer/Counters) ขนาด 16 บิต จำนวน 3 ตัว
  - มีพอร์ตอนุกรมที่สามารถรับส่งข้อมูลแบบฟลูดูเพล็กซ์
  - สามารถอินเทอร์รัปต์ได้ 14 แหล่ง
  - ทำงานในโหมดประหยัดพลังงาน : Idle Mode & Power-Down Mode
- หน่วยความจำภายในแบบแฟลช(Flash) 64 กิโลไบต์ สำหรับเก็บโปรแกรม
- หน่วยความจำแรมเพิ่มเติม(XRAM) บนชิปขนาด 1024 ไบต์
- พอร์ต I/O ที่สามารถใช้เป็น bit-address ติดต่อกับหน่วยความจำภายนอก(P4)

- การใช้งานพอร์ต I/O สามารถใช้ได้หลายแบบ
  - พอร์ตพิเศษ 2 ทิศทาง(Quasi-bidirectional Output)
  - พอร์ตที่รับกระแสไฟฟ้าจากภายนอกเข้าไปในตัวได้(Open-drain Output)
  - พอร์ตอินพุตสำหรับเท่านั้น(Input-only)
  - พอร์ตสำหรับขับกระแสสูงๆ(Push-pull Output)
- เพิ่มสัญญาณการขัดจังหวะภายนอกอีกสองแหล่งคือ /INT2 และ /INT3
- เพิ่มความสามารถในการนับถอยหลังใน Timer2
- โปรแกรมวงจรพิเศษ(Programmable Counter Array : PCA)
  - Capture Mode
  - Software Timer Mode 16 bit
  - High Speed Output Mode
  - PWM (Pulse Width Modulator) Mode
- มีพอร์ตสื่อสารแบบ UART แบบที่สอง
- มีการเชื่อมต่อแบบ SPI Interface (Master/Slave Mode)
- ตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล(ADC) 10 บิต สามารถมัลติเพล็กซ์ได้ 8 ช่องสัญญาณ
- รับอินเทอร์รัปต์จากคีย์บอร์ดได้ 8 อินพุต
- สามารถรับสัญญาณอินเทอร์รัปต์จากภายนอก ขณะที่พลังงานลดต่ำลง
- สามารถโปรแกรมการสร้างสัญญาณนาฬิกาเพื่อส่งออกทาง output ได้
- มีวอตช์ดอกไทมเมอร์(Watchdog Timer)
- รีจิสเตอร์แบบ 2 ทาง ใช้อ้างตำแหน่งหน่วยความจำ(Dual DPTR)
- การติดต่อกับหน่วยความจำภายนอกจะใช้คำสั่ง 'MOVX'
- สามารถกำหนดระบบสัญญาณนาฬิกาเพื่อลดการใช้พลังงาน
- ใช้การโปรแกรมแบบ ISP (In-System Programming) & ICP (In-Circuit Programming) ไปอัพเดตโปรแกรมในหน่วยความจำ
- สามารถโปรแกรมแบบ IAP (In-Application Programming) ได้
- สามารถอ่านเขียนแฟลชได้ 20,000 ครั้ง

- สามารถใช้ความถี่การประมวลผลได้สูงถึง 24 เมกะเฮิร์ต
- ใช้แหล่งจ่ายไฟ 2.4V ~ 3.6V (สำหรับระบบ 3.3V) หรือ 2.7V ~ 5.5V (สำหรับระบบ 5V)
- ช่วงของอุณหภูมิ : -40 ถึง +85 องศาเซลเซียล
- แพ็ตเก็ต : PDIP40, PLCC44, PQFP44, LQFP48 และ SSOP28

2.4 การทำงานของ MPC82G516A

ในรูปที่ 2.2 เป็นโครงสร้างภายในแสดงบล็อกไดอะแกรมแสดงการทำงานของ MPC82G516A โดยจะสรุปส่วนสำคัญของอุปกรณ์ ซึ่งผู้ใช้สามารถค้นหาการทำงานร่วมกับอุปกรณ์ต่อพ่วงจากไดอะแกรมได้ง่าย

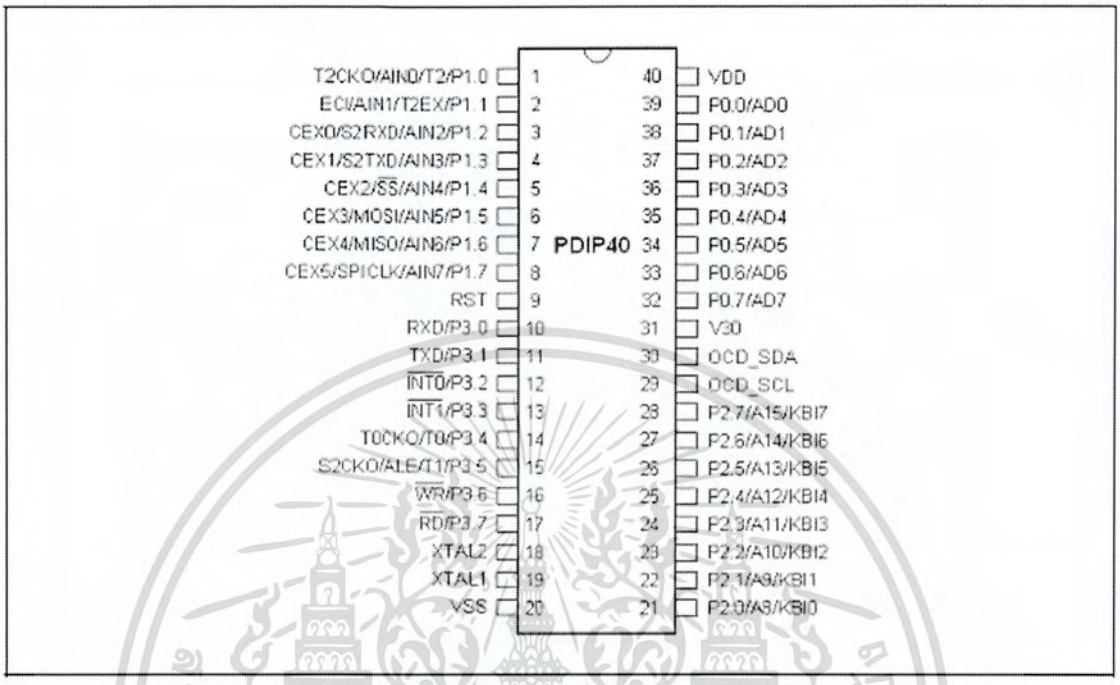


รูปที่ 2.2 บล็อกไดอะแกรมแสดงการทำงานของ MPC82G516A

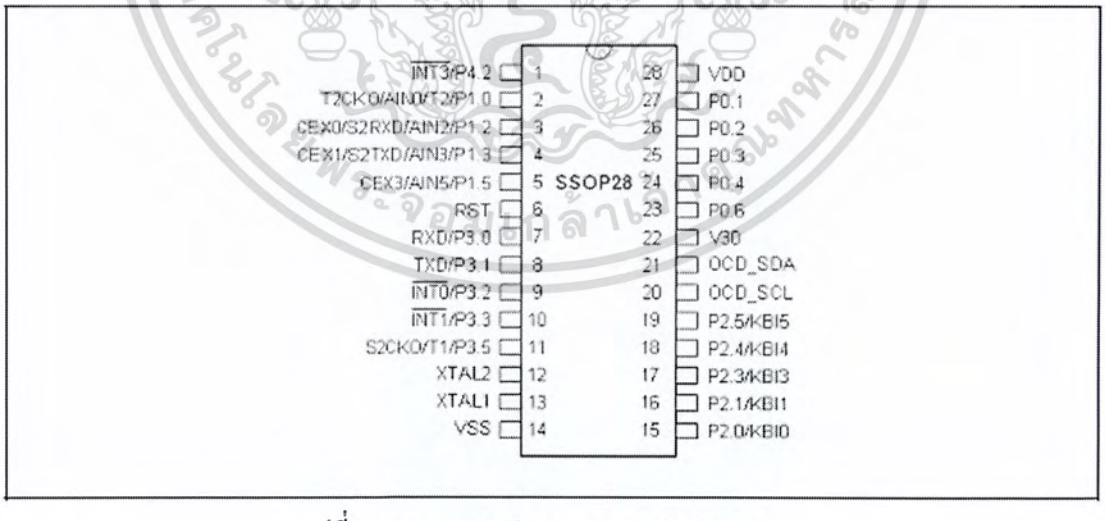
2.5 รูปแบบตัวถังและขา

ไมโครคอนโทรลเลอร์เบอร์ 82g516 มีตัวถังและการจัดขาหลายรูปแบบ ดังนี้

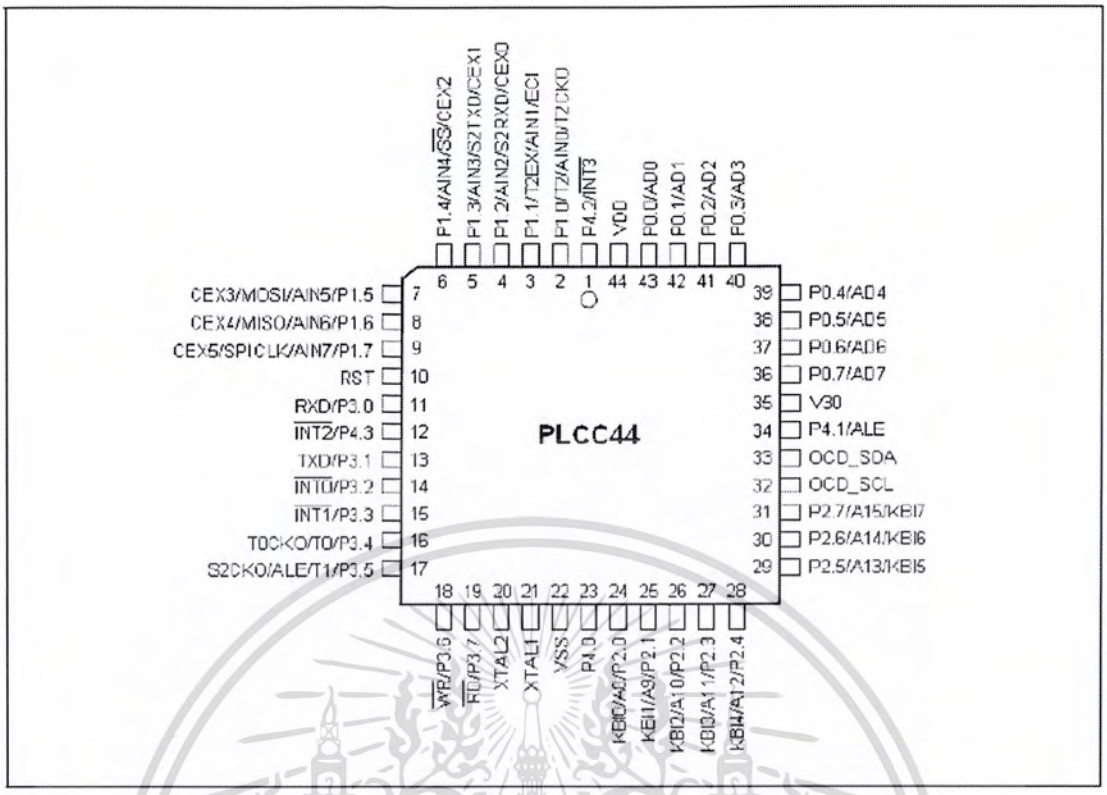
2.5.1 การทำงานของขาต่างๆ



รูปที่ 2.3 แสดงการทำงานของ PDIP 40 ขา



รูปที่ 2.4 แสดงการทำงานของ SSOP 28 ขา



รูปที่ 2.5 แสดงการทำงานของ PLC 44 ขา

2.5.2 รายละเอียดของขาต่างๆ

หน้าที่การทำงานของขาต่าง ๆ สรุปได้ดังตารางที่ 2.1

ตารางที่ 2.1 รายละเอียดของขาต่างๆ

MNEMONIC	40-Pin PDIP	28-Pin SSOP	44-Pin PLC	I/O Type	รายละเอียด
P0.0 (Alt. Fun.) AD0	39	-	37	I/O I/O	-ใช้เป็นขาพอร์ต P0 bit-0 -แปลงสัญญาณอนาลอกเป็นดิจิตอล AD0
P0.1 (Alt. Fun.) AD1	38	27	36	I/O I/O	-ใช้เป็นขาพอร์ต P0 bit-1 -แปลงสัญญาณอนาลอกเป็นดิจิตอล AD1
P0.2 (Alt. Fun) AD2	37	26	35	I/O I/O	-ใช้เป็นขาพอร์ต P0 bit-2 -แปลงสัญญาณอนาลอกเป็นดิจิตอล AD2

MNEMONIC	40-Pin PDIP	28-Pin SSOP	44-Pin PLC	I/O Type	รายละเอียด
P0.3 (Alt. Fun) AD3	36	25	34	I/O I/O	-ใช้เป็นขาพอร์ต P0 bit-3 -แปลงสัญญาณอนาลอกเป็นดิจิตอล AD3
P0.4 (Alt. Fun) AD4	35	24	33	I/O I/O	-ใช้เป็นขาพอร์ต P0 bit-4 -แปลงสัญญาณอนาลอกเป็นดิจิตอล AD4
P0.5 (Alt. Fun) AD5	34	-	32	I/O I/O	-ใช้เป็นขาพอร์ต P0 bit-5 -แปลงสัญญาณอนาลอกเป็นดิจิตอล AD5
P0.6 (Alt. Fun) AD6	33	23	31	I/O I/O	-ใช้เป็นขาพอร์ต P0 bit-6 -แปลงสัญญาณอนาลอกเป็นดิจิตอล AD6
P0.7 (Alt. Fun) AD7	32	-	30	I/O I/O	-ใช้เป็นขาพอร์ต P0 bit-7 -แปลงสัญญาณอนาลอกเป็นดิจิตอล AD7
P1.0 (Alt. Fun) T2 (Alt. Fun) AIN0 (Alt. Fun) T2CKO	1	2	40	I/O I I O	-ใช้เป็นขาพอร์ต P1 bit-0 -ตัวจับเวลา ตัวนับตัวที่ 2 T2 -แปลงสัญญาณอนาลอกเป็นดิจิตอล -เป็นสัญญาณนาฬิกาโปรแกรมได้
P1.1 (Alt. Fun) T2EX (Alt. Fun) AIN1 (Alt. Fun) ECI	2	-	41	I/O I I I	-ใช้เป็นขาพอร์ต P1 bit-1 -ตัวจับเวลา ตัวนับตัวที่ 2 -แปลงสัญญาณอนาลอกเป็นดิจิตอล
P1.2 (Alt. Fun) AIN2 (Alt. Fun) S2RXD (Alt. Fun) CEX0	3	3	42	I/O I I I/O	-ใช้เป็นขาพอร์ต P1 bit-2 -แปลงสัญญาณอนาลอกเป็นดิจิตอล -รับข้อมูลทางพอร์ตอนุกรม

MNEMONIC	40-Pin PDIP	28-Pin SSOP	44-Pin PLC	I/O Type	รายละเอียด
P1.3 (Alt. Fun) AIN3 (Alt. Fun) S2TXD (Alt. Fun) CEX1	4	4	43	I/O I O I/O	-ใช้เป็นขาพอร์ต P1 bit-3 -แปลงสัญญาณอนาลอกเป็นดิจิตอล -ส่งข้อมูลทางพอร์ตอนุกรม
P1.4 (Alt. Fun) AIN4 (Alt. Fun) /SS (Alt. Fun) CEX2	5	-	44	I/O I I I/O	-ใช้เป็นขาพอร์ต P1 bit-4 -แปลงสัญญาณอนาลอกเป็นดิจิตอล -รับ-ส่งข้อมูล
P1.5 (Alt. Fun) AIN5 (Alt. Fun) MOSI (Alt. Fun) CEX3	6	5	1	I/O I I/O I/O	-ใช้เป็นขาพอร์ต P1 bit-5 -แปลงสัญญาณอนาลอกเป็นดิจิตอล
P1.6 (Alt. Fun) AIN6 (Alt. Fun) MISO (Alt. Fun) CEX4	7	-	2	I/O I I/O I/O	-ใช้เป็นขาพอร์ต P1 bit-6 -แปลงสัญญาณอนาลอกเป็นดิจิตอล
P1.7 (Alt. Fun) AIN7 (Alt. Fun) SPICLK (Alt. Fun) CEX5	8	-	3	I/O I I/O I/O	-ใช้เป็นขาพอร์ต P1 bit-7 -แปลงสัญญาณอนาลอกเป็นดิจิตอล
P2.0 (Alt. Fun) A8 (Alt. Fun) KBI0	21	15	18	I/O O I	-ใช้เป็นขาพอร์ต P2 bit-0 -เชื่อมต่อกับ address -รับข้อมูลเข้าจาก keypad 0
P2.1 (Alt. Fun) A9 (Alt. Fun) KBI1	22	16	19	I/O O I	-ใช้เป็นขาพอร์ต P2 bit-1 -เชื่อมต่อกับ address -รับข้อมูลเข้าจาก keypad 1

MNEMONIC	40-Pin PDIP	28-Pin SSOP	44-Pin PLC	I/O Type	รายละเอียด
P2.2	23	-	20	I/O	-ใช้เป็นขาพอร์ต P2 bit-2
(Alt. Fun) A10				O	-เชื่อมต่อกับ address
(Alt. Fun) KBI2				I	-รับข้อมูลเข้าจาก keypad 2
P2.3	24	17	21	I/O	-ใช้เป็นขาพอร์ต P2 bit-3
(Alt. Fun) A11				O	-เชื่อมต่อกับ address
(Alt. Fun) KBI3				I	-รับข้อมูลเข้าจาก keypad 3
P2.4	25	18	22	I/O	-ใช้เป็นขาพอร์ต P2 bit-4
(Alt. Fun) A12				O	-เชื่อมต่อกับ address
(Alt. Fun) KBI4				I	-รับข้อมูลเข้าจาก keypad 4
P2.5	26	19	23	I/O	-ใช้เป็นขาพอร์ต P2 bit-5
(Alt. Fun) A13				O	-เชื่อมต่อกับ address
(Alt. Fun) KBI5				I	-รับข้อมูลเข้าจาก keypad 5
P2.6	27	-	24	I/O	-ใช้เป็นขาพอร์ต P2 bit-6
(Alt. Fun) A14				O	-เชื่อมต่อกับ address
(Alt. Fun) KBI6				I	-รับข้อมูลเข้าจาก keypad 6
P2.7	28	-	25	I/O	-ใช้เป็นขาพอร์ต P2 bit-7
(Alt. Fun) A15				O	-เชื่อมต่อกับ address
(Alt. Fun) KBI7				I	-รับข้อมูลเข้าจาก keypad 7
P3.0	10	7	5	I/O	-ใช้เป็นขาพอร์ต P3 bit-0
(Alt. Fun) RXD				I/O	-รับข้อมูลทางพอร์ตอนุกรม
P3.1	11	8	7	I/O	-ใช้เป็นขาพอร์ต P3 bit-1
(Alt. Fun) TXD				O	-ส่งข้อมูลทางพอร์ตอนุกรม
P3.2	12	9	8	I/O	-ใช้เป็นขาพอร์ต P3 bit-2
(Alt. Fun) /INT0				I	- interrupt ภายนอกหมายเลข 0
P3.3	13	10	9	I/O	-ใช้เป็นขาพอร์ต P3 bit-3
(Alt. Fun) /INT1				I	- interrupt ภายนอกหมายเลข 1
P3.4	14	-	10	I/O	-ใช้เป็นขาพอร์ต P3 bit-4
(Alt. Fun) T0				I	-ตัวจับเวลา ตัวนับตัวที่ 0
(Alt. Fun) T0CKO				O	-เป็นสัญญาณนาฬิกา

MNEMONIC	40-Pin PDIP	28-Pin SSOP	44-Pin PLC	I/O Type	รายละเอียด
P3.5 (Alt. Fun) T1 (Alt. Fun) ALE (Alt. Fun) S2CKO	15	11	11	I/O I O O	-ใช้เป็นขาพอร์ต P3 bit-5 -ตัวจับเวลา ตัวนับตัวที่ 1 -เป็นสัญญาณที่ติดต่อกับหน่วยความจำภายนอก - เป็นสัญญาณนาฬิกา
P3.6 (Alt. Fun) /WR	16	-	12	I/O O	-สัญญาณเขียนข้อมูลหน่วยความจำภายนอก
P3.7 (Alt. Fun) /RD	17	-	13	I/O O	-สัญญาณอ่านข้อมูลหน่วยความจำภายนอก
P4.0	-	-	17	I/O	-ใช้เป็นขาพอร์ต P4 bit-0
P4.1 (Alt. Fun) ALE	-	-	28	I/O O	-ใช้เป็นขาพอร์ต P4 bit-1 -เป็นสัญญาณที่ติดต่อกับหน่วยความจำภายนอก
P4.2 (Alt. Fun) /INT3	-	1	39	I/O I	-ใช้เป็นขาพอร์ต P4 bit-2 - interrupt ภายนอกหมายเลข 3
P4.3 (Alt. Fun) /INT2	-	-	6	I/O I	-ใช้เป็นขาพอร์ต P4 bit-3 - interrupt ภายนอกหมายเลข 2
P4.4	-	-	-	I/O	-ใช้เป็นขาพอร์ต P4 bit-4
P4.5	-	-	-	I/O	-ใช้เป็นขาพอร์ต P4 bit-5
P4.6	-	-	-	I/O	-ใช้เป็นขาพอร์ต P4 bit-6
P.4.7	-	-	-	I/O	-ใช้เป็นขาพอร์ต P4 bit-7
OCD_SDA	30	21	27	I/O	-เป็นขารับส่งข้อมูลแบบอนุกรม
OCD_SCL	29	20	26	I	-เป็นขาอินพุตสัญญาณนาฬิกา
XTAL1	19	13	15	I	-ใช้ต่อกับคริสตอล สำหรับกำหนดสัญญาณนาฬิกาให้ไมโครคอนโทรลเลอร์
XTAL2	18	12	14	O	-ใช้ต่อกับคริสตอล สำหรับกำหนดสัญญาณนาฬิกาให้ไมโครคอนโทรลเลอร์

MNEMONIC	40-Pin PDIP	28-Pin SSOP	44-Pin PLC	I/O Type	รายละเอียด
RST	9	6	4	I	-สำหรับรีเซตระบบ
V30	31	22	29	O	-เป็นสัญญาณเอาท์พุตจากวงจรกำเนิด LDO ภายใน
VDD	40	28	38	I	-แหล่งจ่ายไฟ
VSS	20	14	16	I	-ขาต่อกราวด์

2.6 การจัดการหน่วยความจำ

ไมโครคอนโทรลเลอร์ MPC82G516A มีการจัดแบ่งพื้นที่หน่วยความจำออกเป็น 2 ส่วน คือ หน่วยความจำโปรแกรม (Program Memory) และหน่วยความจำข้อมูล (Data Memory) โดยทั้งสองส่วนนี้จะมีแอดเดรสแยกออกจากกัน โดยหน่วยความจำข้อมูลจะใช้บิตแอดเดรสในการอ้างขนาด 8 บิต

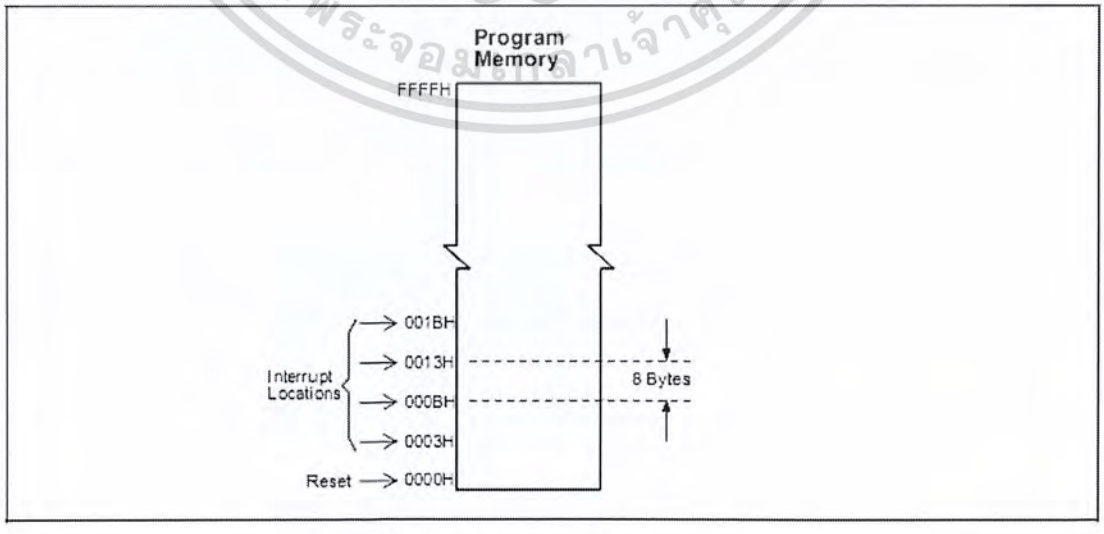
หน่วยความจำโปรแกรม(ROM) เป็นหน่วยความจำแบบอ่านได้อย่างเดียว เขียนไม่ได้ ใน MPC82G516A สามารถที่จะเก็บโปรแกรมได้มากถึง 64 กิโลไบต์ และโปรแกรมทั้งหมดจะถูกเก็บลงบนหน่วยความจำภายในแบบแฟลช โดยจะไม่มีความสามารถในการอ้างถึงหน่วยความจำภายนอก เพราะว่าชิปตัวนี้ไม่มีขา External Access Enable (/EA) และ Program Store Enable (/PSEN)

ใน MPC82G516A มีหน่วยความจำภายในเป็นแรมสแครตช์แพด(Internal scratch-pad RAM) ขนาด 256 ไบต์ โดยพื้นที่ของหน่วยความจำข้อมูลมีการแบ่งแยกแอดเดรสออกจากพื้นที่ของหน่วยความจำโปรแกรม หน่วยความจำข้อมูลภายนอกจะมีพื้นที่สำหรับเก็บข้อมูลมากถึง 64 กิโลไบต์ เมื่อต้องการติดต่อกับหน่วยความจำภายนอกส่วนนี้ CPU จะสร้างแอดเดรส 16 บิต ผ่านรีจิสเตอร์ DPTR และใช้สัญญาณการอ่านและเขียน(/RD และ /WR) เป็นตัวควบคุม ซึ่งเป็นสิ่งจำเป็นในระหว่างการเข้าถึงหน่วยความจำข้อมูลภายนอก ใน MPC82G516A มีการรวมกันของหน่วยความจำแรมภายนอกจำนวน 1024 ไบต์ กับหน่วยความจำบนชิป กลายเป็นแรมส่วนขยายหรือ expanded RAM (เรียกว่า XRAM)

2.6.1 หน่วยความจำโปรแกรม (Program Memory)

หน่วยความจำโปรแกรม เป็นหน่วยความจำที่มีหน้าที่เก็บโปรแกรมที่จะส่งไปให้ CPU ทำการประมวลผล โดยมีการจัดผังพื้นที่หน่วยความจำ ดังรูปที่ 2.6 หลังจากการรีเซต CPU จะเริ่มประมวลผลจากตำแหน่ง 0000H ซึ่งเป็นตำแหน่งเริ่มต้นของโปรแกรม เมื่อ CPU ถูกอินเทอร์รัปต์ CPU จะไปทำโปรแกรมตอบสนองการอินเทอร์รัปต์ โดยตำแหน่งที่เก็บโปรแกรมตอบสนองการอินเทอร์รัปต์ เรียกว่า อินเทอร์รัปต์เวกเตอร์(interrupt vectors) ซึ่งเป็นตำแหน่งที่มีค่าแอดเดรสที่แน่นอนในหน่วยความจำโปรแกรม การอินเทอร์รัปต์ทำให้ CPU กระโดดไปยังตำแหน่งที่เป็นตำแหน่งเริ่มต้นในการประมวลผล ตัวอย่างจากผังหน่วยความจำกำหนดให้ตำแหน่ง 0003H เป็นตำแหน่งของโปรแกรมตอบสนองการอินเทอร์รัปต์จากภายนอกหมายเลข 0 ดังนั้นถ้ามีการอินเทอร์รัปต์จากภายนอกเข้ามาทางสัญญาณหมายเลข 0 เกิดขึ้น การตอบสนองการอินเทอร์รัปต์จะเริ่มต้นขึ้นที่ตำแหน่ง 0003H แต่ถ้าการอินเทอร์รัปต์ไม่เกิดขึ้นตำแหน่งนั้นๆก็จะยังใช้ประโยชน์ได้ตามความต้องการของหน่วยความจำโปรแกรม

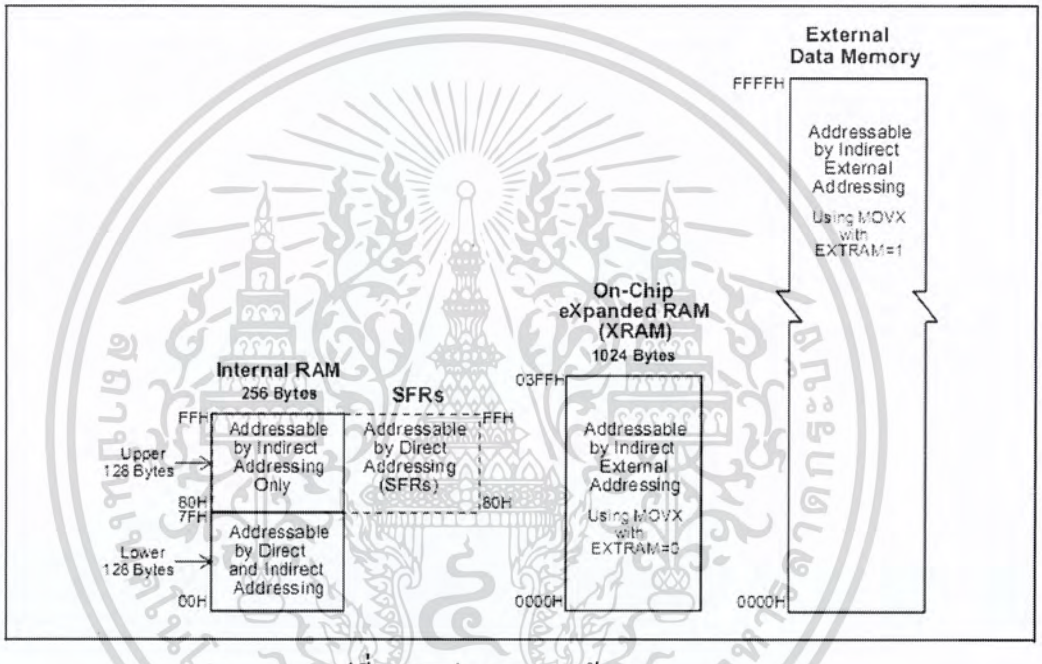
ตำแหน่งของอินเทอร์รัปต์เวกเตอร์แต่ละตัวจะมีระยะห่างกัน 8 ไบต์ เช่นตำแหน่ง 0003H ใช้สำหรับการอินเทอร์รัปต์จากภายนอกหมายเลข 0 ตำแหน่ง 000BH ใช้สำหรับไทเมอร์ 0 ตำแหน่ง 0013H สำหรับการอินเทอร์รัปต์จากภายนอกหมายเลข 1 ตำแหน่ง 001BH สำหรับไทเมอร์ 1 และอื่นๆอีกมากมาย ถ้าโปรแกรมตอบสนองการอินเทอร์รัปต์ที่เกิดขึ้นมีขนาดสั้นๆ(ที่เป็นปัญหาที่เกิดขึ้นบ่อย) โปรแกรมจะสามารถจัดการทั้งหมดได้ภายในระยะ 8 ไบต์ แต่ถ้าการอินเทอร์รัปต์เกิดขึ้นต้องทำโปรแกรมนาน(โปรแกรมตอบสนองการอินเทอร์รัปต์มีความยาวเกิน 8 ไบต์) สามารถใช้คำสั่งกระโดดไปยังตำแหน่งอินเทอร์รัปต์เวกเตอร์และให้ไปทำโปรแกรมการตอบสนองการอินเทอร์รัปต์ที่อยู่ตำแหน่งอื่นๆได้



รูปที่ 2.6 หน่วยความจำโปรแกรม

2.6.2 หน่วยความจำข้อมูล (Data Memory)

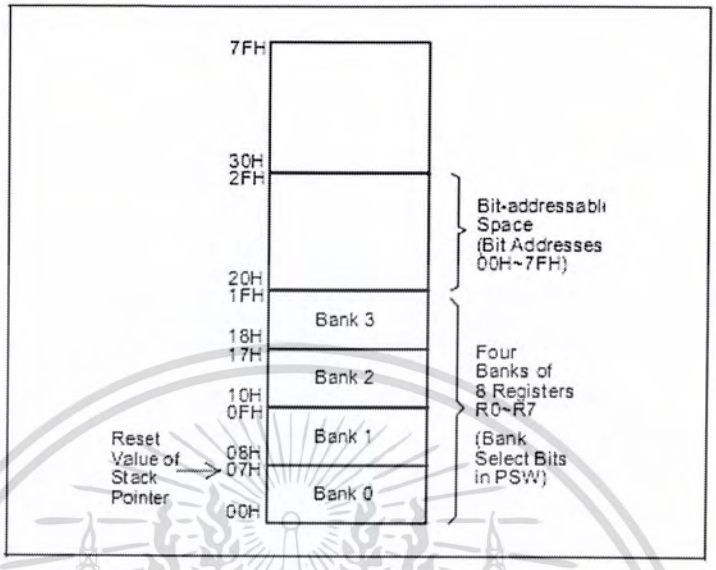
พื้นที่ของหน่วยความจำข้อมูลภายในและภายนอกแสดงได้ดังผังหน่วยความจำในรูปที่ 2.7 โดย หน่วยความจำข้อมูลภายในของ MPC82G516A มีพื้นที่ 256 ไบต์ สามารถแบ่งออกเป็น 2 ส่วน คือ 128 ไบต์แรกที่มีแอดเดรสอยู่ในช่วง 00H-7FH ส่วน 128 ไบต์หลังตั้งแต่แอดเดรส 80H เป็นต้นไปสามารถเข้าถึงหน่วยความจำได้โดยการอ้างแอดเดรสโดยอ้อม(Indirect Addressing) หน่วยความจำ 128 ไบต์หลังนี้แบ่งออกเป็นสองส่วนคือ ส่วนของรีจิสเตอร์ฟังก์ชันพิเศษ(SFR) และหน่วยความจำแรมภายในที่อยู่ตำแหน่ง 128 ไบต์หลังซึ่งจะมีแอดเดรสที่เหมือนกัน โดยจะอยู่ในช่วงแอดเดรส 80H - FPH



รูปที่ 2.7 หน่วยความจำข้อมูล

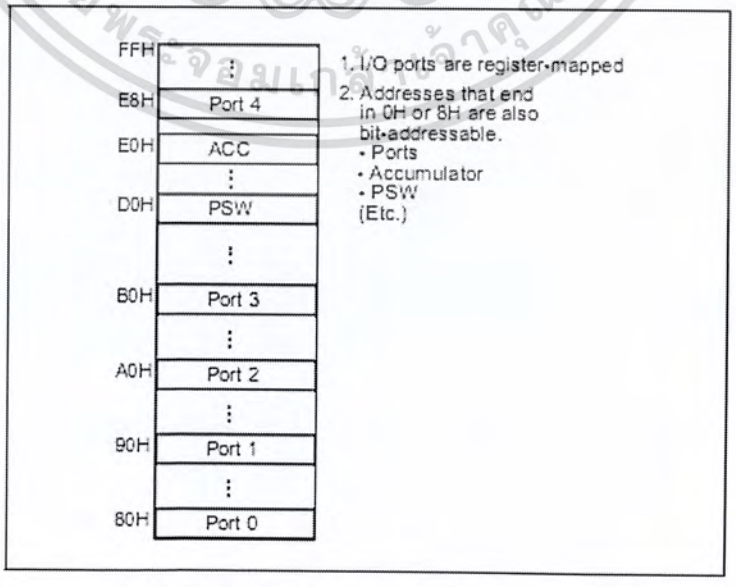
หน่วยความจำแรมภายในที่อยู่ในตำแหน่ง 128 ไบต์แรก จะถูกแบ่งออกเป็น 3 ส่วน ดังรูปที่ 2.8 โดยใน 32 ไบต์แรก ถูกแบ่งออกเป็น 4 แบงก์ แต่ละแบงก์จะถูกอ้างเป็นรีจิสเตอร์ 8 ตัว คือ R0 - R7 การเรียกใช้งานรีจิสเตอร์ในแบงก์ใดจะต้องกำหนดค่าลงไป 2 บิตในโปรแกรมสเตตัสเวิร์ด (Program Status Word : PSW) เนื่องจากขนาดของคำสั่งที่ใช้อ้างตำแหน่งในรีจิสเตอร์โดยตรงจะสั้นกว่าขนาดของคำสั่งที่ใช้อ้างตำแหน่งของหน่วยความจำโดยตรง ส่วน 16 ไบต์ต่อมา จะเป็นส่วนของหน่วยความจำที่เข้าถึงข้อมูลระดับบิตได้ (Bit-addressable) เป็นหน่วยความจำที่อ่านเขียนข้อมูลแบบบิต และอ้างตำแหน่งเป็นตำแหน่งบิตได้ และในพื้นที่ 128 บิตสุดท้ายที่มีแอดเดรสอยู่ในช่วง 00H - 7FH นี้ สามารถที่จะอ้างตำแหน่งของหน่วยความจำได้โดยตรง

128 ไบต์แรกสามารถอ้างตำแหน่งของหน่วยความจำโดยตรง(Direct Addressing)หรืออ้างตำแหน่งโดยทางอ้อม(Indirect Addressing)ก็ได้ ในขณะที่ 128ไบต์หลัง จะอ้างตำแหน่งโดยทางอ้อมได้เท่านั้น



รูปที่ 2.8 แสดง 128 ไบต์แรกของหน่วยความจำแรมภายใน

สำหรับการจัดพื้นที่ของรีจิสเตอร์ฟังก์ชันพิเศษ(SFR) เป็นดังรูปที่ 2.9 ประกอบไปด้วย รีจีสเตอร์ของพอร์ต ตัวจับเวลา ตัวควบคุมภายนอกและอื่นๆอีก โดยรีจิสเตอร์ฟังก์ชันพิเศษต่างๆนี้จะเริ่มที่หน่วยความจำตั้งแต่ 80H – FFH และสามารถอ้างตำแหน่งของหน่วยความจำโดยตรง(Direct Addressing)เท่านั้น ในรีจิสเตอร์ฟังก์ชันพิเศษสามารถเข้าถึงข้อมูลได้ทั้งระดับไบต์(Byte-addressable) และระดับบิต(Bit-addressable) ซึ่งจะมีอยู่ 16 ไบต์ที่สามารถเข้าถึงข้อมูลในระดับบิตได้



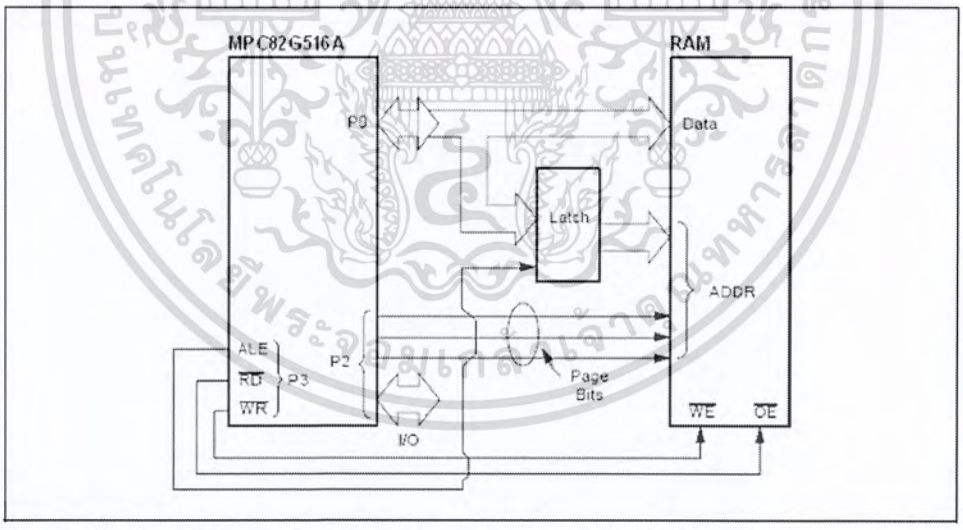
รูปที่ 2.9 ส่วนของรีจิสเตอร์ฟังก์ชันพิเศษ(SFR)

การอ้างถึงหน่วยความจำข้อมูลภายนอก(External Data Memory) EXTRAM bit จะเป็น 1 การติดต่อกับหน่วยความจำที่เก็บข้อมูล 16 บิต (ใช้คำสั่ง 'MOVX @DPTR') หรือการติดต่อกับหน่วยความจำที่เก็บข้อมูล 8 บิต (ใช้คำสั่ง 'MOVX @Ri') ดังนี้

<u>การอ้างตำแหน่งแบบ 8 บิต</u>

การติดต่อกับหน่วยความจำข้อมูลภายนอก 8 บิต จะใช้บ่อยในการรวมกันของสายอินพุตเอาท์พุตหนึ่งหรือหลายเส้นไปยังแรม ในการติดต่อกับหน่วยความจำที่เก็บข้อมูล 8 บิต ในส่วนรีจิสเตอร์ฟังก์ชันพิเศษ ของพอร์ต 2 จะถูกใช้อ้างไปที่ขาพอร์ต 2 ตลอดทั้งหน่วยความจำภายนอกทำให้เข้าถึงได้ง่ายขึ้น รูปที่ 2.10 แสดงตัวอย่างของการอ้างถึงหน่วยความจำแรมภายนอกแบบ 8 บิต เนื่องจากพอร์ต 0 มีการทำงาน 2 หน้าที่ ในการติดต่อกับหน่วยความจำจะใช้วิธีมัลติเพล็กซ์ (Multiplex) ระหว่างแอดเดรสกับข้อมูล ขาแอดเดรสจะถูกส่งออกไปทางพอร์ต 0 และพอร์ต 2 ทั้ง 3 แถว CPU จะสร้างสัญญาณอ่านข้อมูล(/RD หรือ P3.7) และสัญญาณเขียนข้อมูล(/WR หรือ P3.6) ไปยังหน่วยความจำแรมภายนอก ผู้ใช้อาจจะใช้ I/O แถวอื่นๆแทนที่พอร์ต 2 ไปยังแรมก็ได้

เนื่องจากตำแหน่งของหน่วยความจำภายนอกมีได้ถึง 64K รีจิสเตอร์ที่ใช้เก็บค่าตำแหน่งของหน่วยความจำภายนอกจะใช้รีจิสเตอร์ 8 บิต 2 ตัวคือ R0 และ R1 ในการติดต่อกับหน่วยความจำภายนอกจะใช้คำสั่ง MOVX

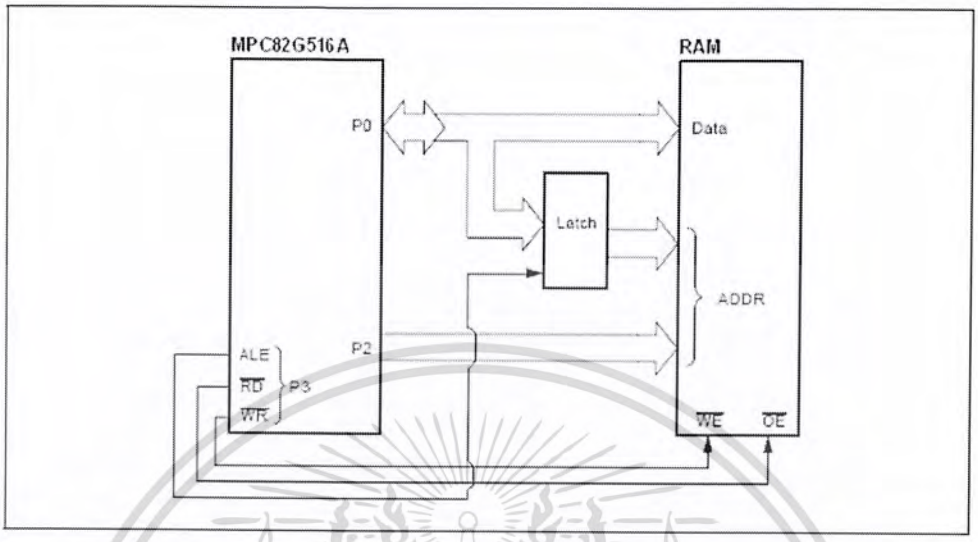


รูปที่ 2.10 การอ้างถึงหน่วยความจำแรมภายนอกแบบ 8 บิต (โดยใช้ 'MOVX @Ri' และ Page Bits)

<u>การอ้างตำแหน่งแบบ 16 บิต</u>

การติดต่อกับหน่วยความจำข้อมูลภายนอก 16 บิต จะใช้บ่อยในการอ้างถึงหน่วยความจำข้อมูลภายนอกที่มากถึง 64 กิโลไบต์ โดยสัญญาณที่ใช้เป็นดังรูปที่ 2.11 ซึ่งแสดงการเข้าถึงหน่วยความจำแรมภายนอกแบบ 16 บิต จะใช้รีจิสเตอร์เก็บค่าตำแหน่งของหน่วยความจำภายนอก 16 บิต คือ รีจีสเตอร์ DPTR โดย DPH จะส่งตำแหน่งที่ 83H ที่เก็บเป็น 8 บิตสูงไปทางพอร์ต 2 และ

DPL จะส่งตำแหน่งที่ 82H ที่เก็บเป็นส่งค่า 8 บิตต่ำไปทางพอร์ต 0 ในช่วงดังกล่าวนี้จะมีการอ่าน (/RD) และเขียนข้อมูล(/WR)



รูปที่ 2.11 การเข้าถึงหน่วยความจำแรมภายนอกแบบ 16 บิต (โดยใช้ 'MOVX @DPTR')

2.7 รีจิสเตอร์ฟังก์ชันพิเศษ (Special Function Registers : SFRs)

2.7.1 SFR Memory Map

แผนผังของพื้นที่หน่วยความจำภายในสำหรับรีจิสเตอร์ฟังก์ชันพิเศษ เรียกว่า "SFR Memory Map" แสดงในตารางที่ 2.2 ในแผนผังหน่วยความจำรีจิสเตอร์ฟังก์ชันพิเศษนี้ จะไม่ครอบครองแอดเดรสทั้งหมด แอดเดรสที่ไม่ถูกครอบครองก็จะไม่ถูกนำไปใช้ การอ่านเพื่อจะเข้าถึงแอดเดรสเหล่านี้ โดยทั่วไปจะคืนค่าข้อมูลแบบสุ่มกลับมา และการเขียนเพื่อจะเข้าถึงแอดเดรสเหล่านี้ อาจจะมีผลทำให้ฮาร์ดแวร์มีการทำงานผิดไปจากเดิม ผู้ใช้ซอฟต์แวร์จะไม่สามารถเข้าถึงแอดเดรสที่ยังไม่ถูกครอบครองนี้ได้

การอ้างถึงพอร์ตหรือหน่วยต่าง ๆ ในไมโครคอนโทรลเลอร์นอกจากจะอ้างถึงชื่อรีจีสเตอร์แล้วยังสามารถอ้างถึงแอดเดรสได้อีกด้วย ตัวอย่างเช่น ถ้าหากต้องการเขียนโปรแกรมส่งออกไปทางพอร์ต P1 นอกจากจะกำหนดค่าให้กับ P1 แล้วยังสามารถทำได้โดยการเขียนข้อมูลไปยังแอดเดรส 90H เนื่องจากพอร์ต P1 ถูกกำหนดไว้หน่วยความจำตำแหน่ง 90H นั้นเอง ทำนองเดียวกันถ้าหากต้องการอ่านข้อมูลที่เข้ามาทางพอร์ต P1 สามารถทำได้โดยการอ่านข้อมูลจากตำแหน่ง 90H ได้เช่นกัน ดังนั้นอาจกล่าวได้ว่ารีจีสเตอร์ฟังก์ชันพิเศษนี้เป็นส่วนของหน่วยความจำภายในที่ใช้ติดต่อกับส่วนต่าง ๆ ในไมโครคอนโทรลเลอร์นั้นเอง นอกจากนี้การใช้งานส่วนต่าง ๆ ภายในไมโครคอนโทรลเลอร์ยังต้องโปรแกรมลักษณะการใช้งานลงไปด้วย ซึ่งต้องโปรแกรมลงในรีจีสเตอร์ควบคุมต่าง ๆ ด้วย รีจีสเตอร์เหล่านี้อยู่ในส่วนของรีจีสเตอร์ฟังก์ชันพิเศษเช่นกัน

	8 BYTES								
F8H	-	CH	CCAP0H	CCAP1H	CCAP2H	CCAP3H	CCAP4H	CCAP5H	FFH
F0H	B	-	PCAPWM0	PCAPWM1	PCAPWM2	PCAPWM3	PCAPWM4	PCAPWM5	F7H
E8H	P4	CL	CCAP0L	CCAP1L	CCAP2L	CCAP3L	CCAP4L	CCAP5L	EFH
E0H	ACC	WDTCR	IFD	IFADRH	IFADRL	IFMT	SCMD	ISPCR	E7H
D8H	CCON	CMOD	CCAPM0	CCAPM1	CCAPM2	CCAPM3	CCAPM4	CCAPM5	DFH
D0H	PSW	-	-	-	-	KBPATN	KBCON	KBMASK	D7H
C8H	T2CON	T2MOD	RCAP2L	RCAP2H	TL2	TH2	-	-	CFH
C0H	XICON	-	-	-	-	ADCTL	ADCH	PCON2	C7H
B8H	IP	SADEN	S2BRT	-	-	-	ADCL	-	BFH
B0H	P3	P3M0	P3M1	P4M0	P4M1			IPH	B7H
A8H	IE	SADDR	S2CON	-	-	AUXIE	AUXIP	AUXIPH	AFH
A0H	P2		AUXR1	-	-	-	AUXR2	-	A7H
98H	SCON	SBUF	S2BUF	-	-	-	-	-	9FH
90H	P1	P1M0	P1M1	P0M0	P0M1	P2M0	P2M1	EVRCR	97H
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	STRETCH	8FH
80H	P0	SP	DPL	DPH	SPSTAT	SPCTL	SPDAT	PCON	87H

↑ Bit-Addressable SFRs

Note that new added SFRs are marked by the blue bold.

ตารางที่ 2.2 แผนผังหน่วยความจำภายในสำหรับรีจิสเตอร์ฟังก์ชันพิเศษ

จากตารางจะพบว่าพอร์ต P2 ถูกกำหนดไว้ในตำแหน่ง A0H รีจีสเตอร์ไทเมอร์ 2 ถูกกำหนดไว้ในตำแหน่ง CCH และ CDH

2.8 โครงสร้างและการทำงานของพอร์ต I/O

การที่ไมโครคอนโทรลเลอร์จะติดต่อกับอุปกรณ์ภายนอกได้นั้นจะต้องติดต่อผ่านทางพอร์ต ใน MPC82G516A มีพอร์ต I/O จำนวน 5 พอร์ต : พอร์ต 0 พอร์ต 1 พอร์ต 2 พอร์ต 3 และพอร์ต 4 ซึ่งพอร์ตทั้งหมดเป็นแบบ 8 บิต จำนวนของขาอินพุต เอาต์พุตที่ใช้จะขึ้นอยู่กับประเภทของแพ็กเกตที่เลือกใช้ ตามตารางที่ 2.3

Package Type	I/O Pins	Number of I/O Pins
40-pin DIP	P0, P1, P2, P3	32
28-pin SSOP	P0.1~P0.4, P0.6, P1.0, P1.2, P1.3, P1.5, P2.0, P2.1, P2.3, P2.4, P2.5 P3.0~P3.3, P3.5 P4.2	20
44-pin PLCC	P0, P1, P2, P3, P4.0~P4.3	36
44-pin PQFP	P0, P1, P2, P3, P4.0~P4.3	36
48-pin LQFP	P0, P1, P2, P3, P4	40

ตารางที่ 2.3 หมายเลยของขา I/O ที่มีอยู่

2.8.1 การตั้งค่าพอร์ต

ขาของพอร์ต I/O ทั้งหมดบน MPC82G516A สามารถเลือกใช้งานได้หลายลักษณะ โดยที่การตั้งค่าอาจจะกระทำโดยผู้ใช้หรือถูกตั้งค่าโดยโปรแกรมให้เป็นหนึ่งในสี่ประเภท ซึ่งเป็นพื้นฐานแบบบิตต่อบิต (bit-by-bit) ที่แสดงในตารางที่ 2.3 คือ โปรแกรมเป็นพอร์ตพิเศษแบบ 2 ทาง(quasi-bidirectional) โปรแกรมให้ใช้ขับกระแสสูงๆ(push-pull output) โปรแกรมให้เป็นพอร์ตที่รับกระแสไฟฟ้าจากภายนอกเข้าไปในตัวชิป (open-drain output) และใช้เป็นพอร์ตอินพุทเท่านั้น (input-only) ในการตั้งค่าการใช้งานสำหรับแต่ละพอร์ตนั้น จะกำหนดโดยการตั้งค่าบิตเพื่อเลือกโหมดการใช้งานดังตารางที่ 2.4

PxM0.y	PxM1.y	Port Mode
0	0	Quasi-bidirectional
0	1	Push-Pull Output
1	0	Input-Only (High Impedance Input)
1	1	Open-Drain Output

ตารางที่ 2.4 การตั้งค่าพอร์ต

การใช้งานพื้นฐานจะโปรแกรมให้พอร์ต 0 ถึง พอร์ต 3 เป็นพอร์ตที่ติดต่อได้สองทิศทาง สามารถเป็นได้ทั้งอินพุตและเอาต์พุต เนื่องจากโครงสร้างภายในมีวงจรแลตช์ (latch) และวงจรขับบัฟเฟอร์อินพุต ในวงจรของบิตพอร์ตจะมีวงจรมัลติเพล็กซ์สำหรับกำหนดการทำงานของพอร์ตว่าจะให้เป็นอินพุตหรือเอาต์พุต หรือใช้เป็นขาพิเศษอื่น ๆ เช่น ขารับสัญญาณอนาลอก ขารับสัญญาณอินเทอร์รัปต์ หรือขารับสัญญาณนาฬิกาภายนอกเมื่อถูกใช้เป็นตัวนับ

สำหรับการกำหนดใช้พอร์ตลักษณะต่าง ๆ ต้องกำหนดในรีจีสเตอร์ P0M0 ดังตัวอย่างต่อไปนี้ ขณะที่ x=0~4 (หมายเลขพอร์ต) และ y=0~7 (ขาพอร์ต) รีจิสเตอร์ PxM0 และ PxM1n ตามรายการด้านล่าง

P0M0 (Address=93H, Port 0 Mode Register 0, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P0M0.7	P0M0.6	P0M0.5	P0M0.4	P0M0.3	P0M0.2	P0M0.1	P0M0.0

P0M1 (Address=94H, Port 0 Mode Register 1, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P0M1.7	P0M1.6	P0M1.5	P0M1.4	P0M1.3	P0M1.2	P0M1.1	P0M1.0

P1M0 (Address=91H, Port 1 Mode Register 0, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P1M0.7	P1M0.6	P1M0.5	P1M0.4	P1M0.3	P1M0.2	P1M0.1	P1M0.0

P1M1 (Address=92H, Port 1 Mode Register 1, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P1M1.7	P1M1.6	P1M1.5	P1M1.4	P1M1.3	P1M1.2	P1M1.1	P1M1.0

P2M0 (Address=95H, Port 2 Mode Register 0, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P2M0.7	P2M0.6	P2M0.5	P2M0.4	P2M0.3	P2M0.2	P2M0.1	P2M0.0

P2M1 (Address=96H, Port 2 Mode Register 1, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P2M1.7	P2M1.6	P2M1.5	P2M1.4	P2M1.3	P2M1.2	P2M1.1	P2M1.0

P3M0 (Address=B1H, Port 3 Mode Register 0, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P3M0.7	P3M0.6	P3M0.5	P3M0.4	P3M0.3	P3M0.2	P3M0.1	P3M0.0

P3M1 (Address=B2H, Port 3 Mode Register 1, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P3M1.7	P3M1.6	P3M1.5	P3M1.4	P3M1.3	P3M1.2	P3M1.1	P3M1.0

P4M0 (Address=B3H, Port 4 Mode Register 0, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P4M0.7	P4M0.6	P4M0.5	P4M0.4	P4M0.3	P4M0.2	P4M0.1	P4M0.0

P4M1 (Address=B4H, Port 4 Mode Register 1, Reset Value=0000,0000B)

7	6	5	4	3	2	1	0
P4M1.7	P4M1.6	P4M1.5	P4M1.4	P4M1.3	P4M1.2	P4M1.1	P4M1.0

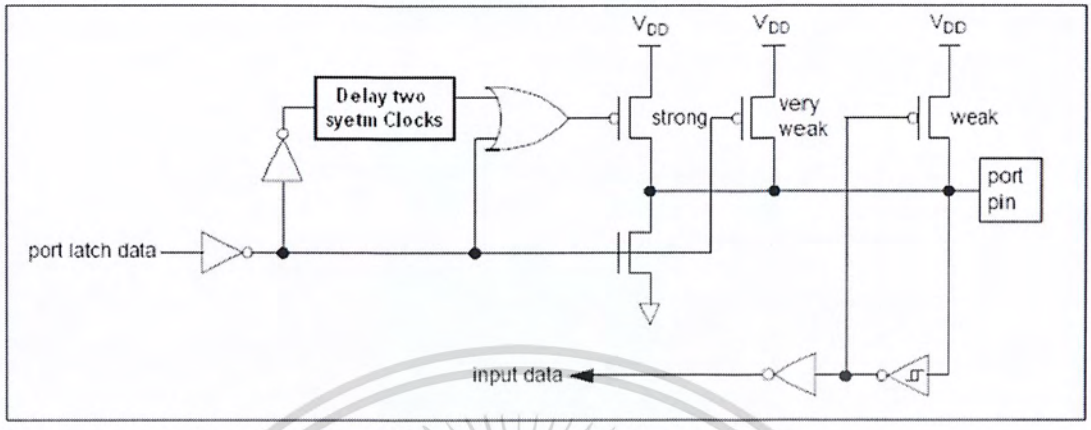
2.8.2 การกำหนดการทำงานเป็นพอร์ตอินพุต

ในการใช้งานเป็นพอร์ตอินพุตจะต้องเขียนข้อมูลให้พอร์ตที่ต้องการใช้งานเป็นอินพุตเสียก่อน โดยกำหนดให้มีสภาวะเป็นลอจิก “1” ทุกบิต เพื่อหยุดการทำงานของตัวเฟตที่อยู่ภายในโครงสร้างของพอร์ต เมื่อตัวเฟตหยุดทำงานจะทำให้สัญญาณของพอร์ตถูกเชื่อมต่อกับวงจรพูลอัปภายในโดยตรง ส่งผลให้ขาพอร์ตมีสภาวะลอจิก “1” สามารถรอรับลอจิก “0” จากอุปกรณ์ภายนอกได้ ดังนั้นเมื่อต้องการให้พอร์ตทำงานลักษณะอินพุตโดยรับสัญญาณไฟฟ้าเข้ามาที่ขาพอร์ต ควรกำหนดการทำงานของพอร์ตให้รับสัญญาณไฟฟ้าสภาวะลอจิก “0” ที่ใช้รับค่าจากอินพุต โดยอย่างเช่นการเขียนโปรแกรมภาษาแอสเซมบลีด้วยคำสั่ง MOV A , P1 หมายความว่าให้นำค่าจากขาพอร์ต P1 ไปเก็บในรีจิสเตอร์ A หรือรีจิสเตอร์แอกคิวมูเลเตอร์ (Accumulator)

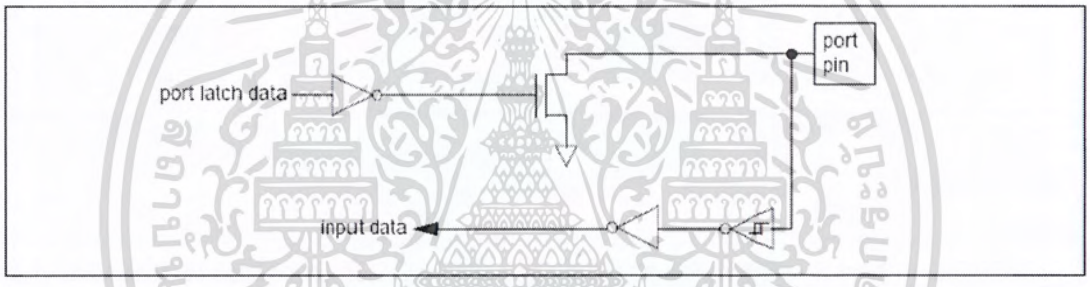
2.8.3 การกำหนดการทำงานเป็นพอร์ตเอาต์พุต

ในการใช้งานโดยทั่วไปพอร์ตของไมโครคอนโทรลเลอร์ถูกโปรแกรมเป็นพอร์ตเอาต์พุตมาตั้งแต่เริ่มต้นการทำงานแล้ว ถ้าหากต้องการให้พอร์ตส่งค่าลอจิก “0” ก็ทำเพียงส่งลอจิก “0” ไปให้วงจรแลตช์ส่งสัญญาณไปขับตัวเฟตภายในโครงสร้างของพอร์ตให้ทำงาน จากนั้นที่ขอพอร์ตที่กำหนดจะปรากฏภาวะลอจิก “0” ขึ้นทันที และเมื่อต้องการให้ขาพอร์ตส่งลอจิก “1” ก็ทำเพียงเขียน

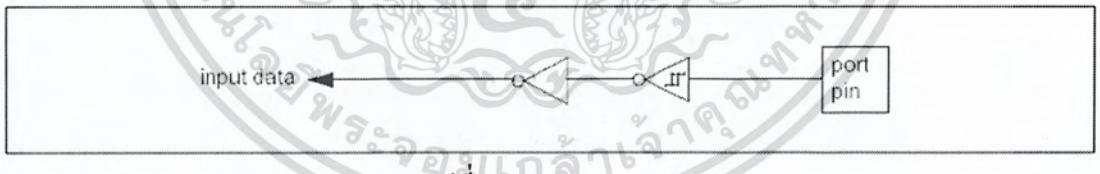
ข้อมูลลอจิก “1” ให้กับวงจรแลตช์ ซึ่งจะส่งผลให้ตัวเฟตหยุดทำงาน ทำให้ขอพอร์ตที่เชื่อมต่ออยู่กับวงจรพูลอัปภายในส่งลอจิก “1” มายังบิตพอร์ตที่กำหนดทันที



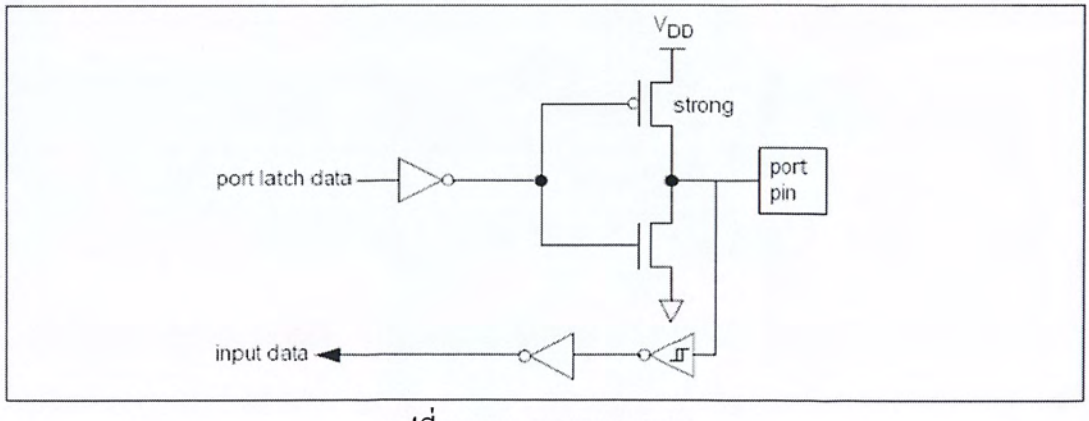
รูปที่ 2.12 Quasi-Bidirectional I/O



รูปที่ 2.13 Open-Drain Output



รูปที่ 2.14 Input Only



รูปที่ 2.15 Push-Pull Output

2.9 ภาษาซีสำหรับไมโครคอนโทรลเลอร์

โปรแกรม C51 เป็นซีคอมไพเลอร์ที่ออกแบบมาสำหรับการเขียนโปรแกรมภาษาซี กับไมโครคอนโทรลเลอร์เบอร์ MCS-51 ของ Franklin and keil ซึ่งสามารถดาวน์โหลดมาทดลองใช้ได้ที่ www.keil.com โปรแกรมตัวนี้มีทั้งรุ่นที่ทำงานบน DOS และทำงานบน WINDOW โปรแกรมนี้มีข้อดีหลายอย่าง เช่น ได้รหัสภาษาเครื่องของ MCS-51 ที่มีขนาดเล็ก สามารถใช้กับ MCS-51 ได้หลายเบอร์ มีฟังก์ชันเกี่ยวกับการอินเทอร์รัปต์ ฟังก์ชันเกี่ยวกับสตริง รวมทั้งฟังก์ชันทางคณิตศาสตร์ต่าง ๆ ให้ใช้ได้ โปรแกรม C51 นี้มีรายละเอียดต่าง ๆ มากมาย ผู้สนใจสามารถศึกษาได้จากคู่มือโดยตรง สำหรับในเอกสารเล่มนี้จะเน้นการเขียนโปรแกรมควบคุมและการออกแบบวงจรเป็นหลัก

โดยทั่วไปนักพัฒนาไมโครคอนโทรลเลอร์ที่เขียนด้วยภาษาแอสเซมบลีจะทำการเขียนโปรแกรมและแปลให้เป็นออปเจกต์ไฟล์ (นามสกุล .OBJ) หรือเฮ็กไฟล์ (นามสกุล .HEX) จากนั้นจะนำไปโปรแกรมลงบนชิปไมโครคอนโทรลเลอร์ แต่การเขียนโปรแกรมด้วยภาษาซีจะมีขั้นตอนต่าง ๆ มากขึ้นดังนี้

1. เขียนชุดคำสั่งบนโปรแกรม Editor ให้มีนามสกุลเป็น .C
2. กำหนดพรีโพรเซสเซอร์ (preprocessor) สำหรับกำหนดการคอมไพล์โปรแกรม ซึ่งอาจกำหนดไว้ใน source code หรือกำหนดตอนคอมไพล์ก็ได้
3. คอมไพล์โปรแกรม ในที่นี้จะใช้โปรแกรม C51 ในการคอมไพล์ และจะได้ไฟล์ออปเจกต์ ที่มีนามสกุลเป็น .OBJ ออกมา
4. เชื่อมต่อชุดคำสั่งเสริม (link) โดยใช้โปรแกรม L51 ซึ่งจะทำการรวมชุดคำสั่งต่าง ๆ ที่อ้างอิงถึงกัน และโปรแกรมจะสร้างออปเจกต์ไฟล์ที่สมบูรณ์ออกมา
5. แปลงให้เป็นไฟล์ hexadecimal file ด้วยโปรแกรม OHS51 ซึ่งจะได้ไฟล์นามสกุล .HEX ออกมา

ถ้าหากใช้ซีคอมไพเลอร์ตัวอื่น ๆ ขั้นตอนเหล่านี้อาจแตกต่างกันไป

2.9.1 คำสั่งพื้นฐานและการประกาศตัวแปร

สำหรับผู้ที่เคยศึกษาการเขียนโปรแกรมภาษาซีบนคอมพิวเตอร์พีซีมาแล้วจะพบว่าโปรแกรมแรกที่ตำราหลายเล่มแนะนำให้เขียนจะเป็นดังนี้

```
#include<stdio.h>

main()

{
      printf("Hello Program C\n");
}
```

เมื่อรันโปรแกรมคอมพิวเตอร์จะแสดงคำว่า Hello Program C ออกทางจอภาพ จากโปรแกรมจะพบว่าบรรทัดแรกจะเป็นการเรียกฟังก์ชันที่ใช้ติดต่อกับไอโอมาตรฐานออกมา จากนั้นก็เรียกใช้ printf ที่เก็บอยู่ใน stdio.h นั้น สำหรับการเขียนโปรแกรมกับไมโครคอนโทรลเลอร์การทำงานต่าง ๆ จะเป็นการกระทำกับพอร์ต เรจิสเตอร์ และหน่วยความจำต่าง ๆ ดังนั้นผลลัพธ์จากการทำงานของโปรแกรมจะเกี่ยวข้องกับการย้ายข้อมูลระหว่างเรจิสเตอร์กับหน่วยความจำต่าง ๆ เป็นส่วนใหญ่

ถ้าหากเราต้องการเขียนโปรแกรมให้กับไมโครคอนโทรลเลอร์ส่งค่าตั้งแต่ 0 ถึง 255 ออกไปทางพอร์ต P1 เราสามารถเขียนโปรแกรมด้วยภาษาแอสเซมบลีได้ดังนี้

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

แต่ถ้าหากเขียนโปรแกรมเป็นภาษาซีจะได้เป็น

```
#include<req52.h>

main()
{
      unsigned char i;

      for(i=0;i<=255;i++)

            P1 = i;
}
```

ในบรรทัดแรกจะคล้ายกับบรรทัดแรกของการเขียนโปรแกรมบนเครื่องคอมพิวเตอร์ PC โดยจะบอกให้คอมไพเลอร์รู้จักตัวแปรและรีจิสเตอร์ต่าง ๆ ของ MCS-51 โดยในไฟล์ req52.h จะบรรจุชื่อรีจิสเตอร์และหน่วยความจำต่าง ๆ ของ 8052 เอาไว้

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

การเขียนโปรแกรมด้วยภาษาซีนี้สามารถเขียนไดเร็กทีฟควบคุม (control directive) ได้ ซึ่งจะเป็นการบอกว่าจะให้ทำการใด ๆ ขณะคอมไพล์โปรแกรมโดยอาจเขียนตอนคอมไพล์ หรือเขียนภายในโปรแกรมก็ได้ โดยการกำหนดในส่วนของ preprocess ซึ่งจะใช้คำว่า #pragma นำหน้า ไดเร็กทีฟควบคุมนั้นมีอยู่หลายคำสั่งสำหรับรายละเอียดต่าง ๆ จะกล่าวในภายหลัง สำหรับคำสั่งที่ใช้กันมากที่สุด คือ คำสั่ง code ซึ่งจะบอกว่าให้สร้างภาษาแอสเซมบลีขณะที่ทำการแปลโปรแกรมด้วย พิจารณาตัวอย่างโปรแกรมต่อไปนี้

```
#pragma code
#include<req51.h>
unsigned char i;
main()
{
      a = 5;
      b = 8;
      c = a=b;
      P1 = c;
}
```

จากโปรแกรมจะเห็นว่าในบรรทัดแรกจะบอกว่าขณะแปลโปรแกรมทำการสร้างไฟล์ภาษาแอสเซมบลีด้วยโดยระบุว่า #pragma code

2.9.2 โครงสร้างของภาษาซี

ด้วยภาษาซีเป็นภาษาที่สามารถเขียนโปรแกรมเป็นแบบ โครงสร้างได้ โดยโปรแกรมจะแบ่งการทำงานต่าง ๆ ออกเป็นกลุ่ม ๆ หรือฟังก์ชัน โดยฟังก์ชันเหล่านั้นสามารถเรียกขึ้นมาใช้ใหม่ได้ ในการเขียนโปรแกรมจะต้องระบุไว้ว่าในโปรแกรมนั้นมีฟังก์ชันใดให้ใช้บ้าง แต่ทุกโปรแกรมจะต้องมีฟังก์ชันหลักที่ชื่อว่า main() เสมอ พิจารณาตัวอย่าง โครงสร้างของโปรแกรมต่อไปนี้

```
#include<req52.h>                  /*Preprocessor*/

void func1(void);                  /*Pro to type*/

int func2(int x);

void main()                        /*ฟังก์ชันหลัก*/
{
      int a;                       /*ประกาศตัวแปร*/
      P1 = 0x0FF;
      func1();                     /*เรียกใช้ฟังก์ชัน*/
      a = func2(4);                /*เรียกใช้ฟังก์ชันที่มีการส่งค่า*/
      P1 = a;
}

void func1(void)                   /*ฟังก์ชันที่ไม่มีการคืนค่า*/
{
      .....
      .....
}

int func2(int x)                   /*ฟังก์ชันที่มีการคืนค่า*/
{
      return(x*2);
}
```

จากโปรแกรมจะพบว่า ส่วนประกาศโพรโตไทป์จะบอกว่าโปรแกรมนี้มีฟังก์ชันชื่อ func1 ให้ใช้งานและฟังก์ชันนี้จะทำงานเป็นโปรแกรมย่อยเพราะมี void นำหน้า และมีฟังก์ชันชื่อ func2(int x) ซึ่งจะรับค่าเข้าไปผ่านทางตัวแปร x และคืนค่าออกมาเป็นจำนวนเต็ม

2.9.3 ตัวแปรและค่าคงที่

การใช้งานตัวแปรและค่าคงที่ต่าง ๆ จะต้องมีการประกาศชื่อตัวแปรขึ้นมาเสียก่อนเมื่อมีการคอมไพล์โปรแกรมตัวคอมไพเลอร์จะเตรียมพื้นที่ในหน่วยความจำแรมเอาไว้สำหรับเก็บตัวแปรและค่าคงที่เหล่านั้น ในการประกาศตัวแปรสามารถทำได้ดังนี้

ประเภทของข้อมูล ชื่อตัวแปร [, ... ,];

การประกาศตัวแปรจะต้องเริ่มด้วยชื่อประเภทของข้อมูล และตามด้วยชื่อตัวแปร โดยจะประกาศครั้งละกี่ตัวก็ได้ ส่วนชื่อของตัวแปรนั้นจะซ้ำกันไม่ได้และต้องไม่ซ้ำกับชื่อของคำสงวน (keywords) ของคอมไพเลอร์ตัวนั้น ๆ สำหรับประเภทของข้อมูลในการประกาศตัวแปรเป็นดังนี้

ประเภทของข้อมูล	ขนาด (บิต)	ค่าที่เก็บได้
Bit	1	0 ถึง 1
Char	8	-128 ถึง 127
unsigned char	8	0 ถึง 255
Int	16	-32768 ถึง 32767
unsigned int	16	0 ถึง 65535
Long	32	-2147483648 ถึง +2147483647
unsigned long	32	0 ถึง 4294967295
Float	32	-1.17549e -38 ถึง +3.402823e +38

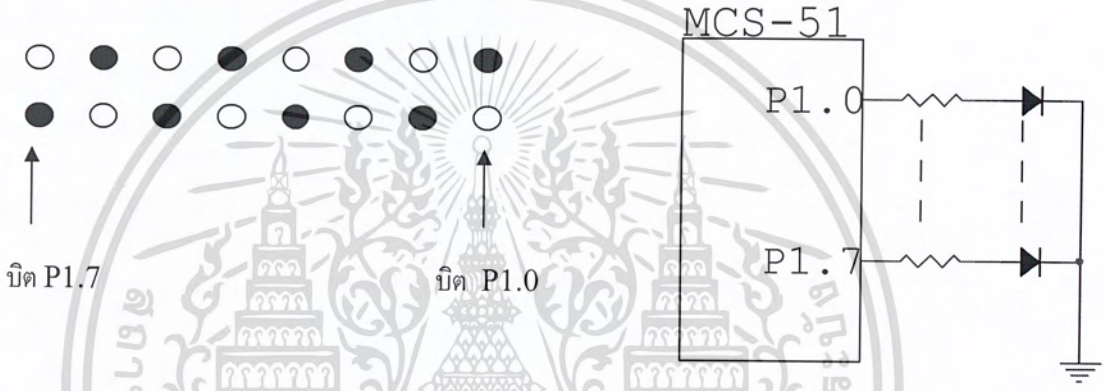
สำหรับคำสงวนเป็นคำที่คอมไพเลอร์รู้จักและจะถูกใช้งานเฉพาะ เราไม่สามารถนำมาตั้งเป็นชื่อตัวแปรและฟังก์ชันได้ คำสงวนของโปรแกรม C51 เป็นดังต่อไปนี้

_at_	idata	sfr
alien	interrupt	sfr16
bdata	large	small
bit	pdata	_task_
code	_priority_	using
compact	reentrant	xdata
data	sbit	

ตัวอย่างโปรแกรมต่อไปนี้จะเป็นการส่งข้อมูลตั้งแต่ 0 ถึง 10 ออกทางพอร์ต P1

```
#pragma code
#include<req51.h>
unsigned char i;
main()
{
      for(i=0;i<=10;i++)
      P1 = i;
}
```

ตัวอย่าง ถ้าหากพอร์ต P1 ของ MCS-51 ต่ออยู่กับหลอด LED จำนวน 8 หลอด ดังรูป จงเขียนโปรแกรมให้ LED สว่างและดับสลับกันดังรูปต่อไปนี้



วิธีทำ การทำให้หลอด LED สว่างดวงเว้นดวงทำได้โดยส่งค่า 55H ออกไปทางพอร์ต P1 จากนั้นหน่วงเวลาและส่งค่า 0AAH ออกไปทางพอร์ต P1 และหน่วงเวลาจากนั้นทำโปรแกรมซ้ำ จะเขียนโปรแกรมได้ดังต่อไปนี้

```
#pragma code                          1
#include<req51.h>
void delay(int count);                2
void main()
{
      while(1)                        3
      {
            P1 = 0x55;
            delay(100);
            P1 = 0x0AA;
            delay(100);
      }
}
void delay(int count)                 4

{
      int i,j;                        5
      for(i=0;i<count;i++)
            for(j=0;j<500;j++);
}
```

จากโปรแกรมจะเห็นว่ามีการสร้างฟังก์ชันชื่อ delay ขึ้นมาใช้งาน ทำให้โปรแกรมหลักสามารถเรียกขึ้นมาทำงานได้โดยละเอียดของโปรแกรมเป็นดังนี้

บรรทัดแรกตามหมายเลข 1 จะเป็นการกำหนดไดเร็กทีฟควบคุมเพื่อบอกว่าให้แปลเป็นรหัสภาษาแอสเซมบลีด้วย โดยบรรทัดนี้จะไม่ต้องมีก็ได้ ถ้าไม่มีจะไม่มีการสร้างไฟล์ภาษาแอสเซมบลีขึ้นมา สำหรับหมายเลข 2 จะเป็นการประกาศโพรโตไทป์ว่ามีฟังก์ชันชื่อ delay อยู่เพื่อให้โปรแกรมหลัก main สามารถเรียกฟังก์ชันขึ้นมาใช้ได้ ถ้าหากไม่ประกาศเมื่อมีการคอมไพล์โปรแกรมจะมี warning และ error เกิดขึ้นซึ่งจะบอกว่าโปรแกรมหลักไม่รู้จักฟังก์ชันชื่อ delay

ในส่วนของโปรแกรมหลักในหมายเลข 3 จะเป็นฟังก์ชัน while ของภาษาซี ซึ่งจะให้โปรแกรมทำฟังก์ชันที่อยู่ในเครื่องหมายปีกกาซ้ำ ซึ่งจะทำให้หลอด LED ที่ต่ออยู่ที่พอร์ต P1 เกิดการติดดับสลับกัน สำหรับโปรแกรมย่อยตามหมายเลข 4 จะมีการส่งผ่านค่าเข้าไปทางตัวแปร count ตัวแปรนี้บางครั้งเรียกว่าฟอร์มอลพารามิเตอร์ (formal parameters) ถ้าหากค่านี้มีค่าน้อยโปรแกรมจะหน่วงเวลาน้อย ถ้าหากมีค่ามากโปรแกรมจะหน่วงเวลามาก ในฟังก์ชัน delay จะมีการประกาศตัวแปรขึ้นมาสองตัวคือ i และ j (หมายเลข 5) ตัวแปรนี้เรียกว่าตัวแปรโลเคิล เพราะว่าประกาศขึ้นมาใช้ภายในฟังก์ชันเท่านั้น

ในการเขียนโปรแกรมภาษาซีในลักษณะมอดูล ถ้าหากนำฟังก์ชันต่าง ๆ ขึ้นมาเขียนก่อนฟังก์ชัน main() ก็สามารถทำได้ดังแสดงในรูปต่อไป การเขียนแบบนี้จะไม่ต้องประกาศโพรโตไทป์ก็ได้ เพราะว่าฟังก์ชันหลัก (main) สามารถมองเห็นฟังก์ชันที่อยู่ก่อนหน้าได้

```
#pragma code

#include<req51.h>

void delay(int count);
{
      int i,j;

      for(i=0;i<count;i++)

            for(j=0;j<500;j++);
}

void main()
{

      while(1)
      {
            P1 = 0x55;
            delay(100);
            P1 = 0x0AA;
            delay(100);
      }
}
```

2.9.4 ตัวดำเนินการในภาษาซี

ตัวดำเนินการจะเป็นตัวที่ใช้กระทำกับตัวแปร ค่าคงที่ต่าง ๆ ให้รวมเป็นค่าเดียวกัน โดยอาจกระทำทางคณิตศาสตร์หรือการกระทำทางลอจิกก็ได้ ในการเขียนโปรแกรมด้วยภาษาซีนั้น ตัวดำเนินการจะแบ่งออกเป็นสองกลุ่มใหญ่ ๆ คือตัวดำเนินการที่กระทำกับตัวถูกกระทำตัวเดียว (single operand operators) และตัวดำเนินการที่กระทำกับตัวถูกกระทำสองตัว (two operands operators)

- ตัวดำเนินการที่กระทำกับตัวถูกกระทำตัวเดียว

ตัวดำเนินการประเภทนี้จะกระทำกับตัวถูกกระทำเพียงตัวเดียว ประกอบด้วยตัวดำเนินการต่าง ๆ ดังนี้

-	ลบ (negate)
~	กลับค่าลอจิกของบิตข้อมูล (bit wise complement)
!	กลับค่าทางลอจิก (logical complement)
++	เพิ่มค่าขึ้นหนึ่งค่า (increment)
--	ลดค่าลงหนึ่งค่า (decrement)
*	ตัวดำเนินการทางพอยน์เตอร์
&	ตำแหน่งหน่วยความจำของตัวแปร

- ตัวดำเนินการที่กระทำกับตัวถูกกระทำสองตัว

ตัวดำเนินการประเภทนี้จะกระทำกับตัวถูกกระทำสองตัว ถ้าหากมีตัวถูกกระทำหลาย ๆ ตัว สามารถนำมาเขียนรวมกันเป็นประโยคได้ ซึ่งประกอบไปด้วยตัวดำเนินการที่ใช้กำหนดค่า ตัวดำเนินการทดสอบค่าซึ่งจะให้ผลลัพธ์เป็นค่าทางลอจิก (จริง,เท็จ) ตัวดำเนินการทางคณิตศาสตร์ และตัวดำเนินการทางลอจิก ดังต่อไปนี้

=	กำหนดค่าในประโยค (assignment)
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หารแบบบวก (modulo)
&&	การแอนด์ (logical AND)
||	การออร์ (logical OR)

&	การแอนด์แบบบิตต่อบิต (bit wise AND)
|	การออร์แบบบิตต่อบิต (bit wise OR)
^	การเอ็กคลูซีฟออร์แบบบิตต่อบิต (bit wise exclusive OR)
<<	เลื่อนบิตไปทางซ้าย
>>	เลื่อนบิตไปทางขวา
==	ทดสอบว่าเท่ากันหรือไม่
!=	ทดสอบว่าไม่เท่ากันหรือไม่
>	ทดสอบว่ามากกว่าหรือไม่
<	ทดสอบว่าน้อยกว่าหรือไม่
>=	ทดสอบว่ามากกว่าหรือเท่ากับหรือไม่
<=	ทดสอบว่าน้อยกว่าหรือเท่ากับหรือไม่

นอกจากนี้ตัวดำเนินการบางประเภทสามารถนำมารวมกันเป็น compound operators ได้ ตัวอย่างเช่น ถ้ามีประโยค y = y*2; อาจเขียนตัวดำเนินการรวมได้เป็น y *= 2; รูปแบบของตัวดำเนินการที่รวมกันได้เป็นดังนี้

+=	บวกและให้เท่ากับ
-=	ลบและให้เท่ากับ
*=	คูณและให้เท่ากับ
/=	หารและให้เท่ากับ
%=	หารแบบบวกและให้เท่ากับ
&=	ทำการแอนด์และให้เท่ากับ
|=	ทำการออร์และให้เท่ากับ
^=	ทำการเอ็กคลูซีฟออร์และให้เท่ากับ
<<=	เลื่อนบิตไปทางซ้ายและให้เท่ากับ
>>=	เลื่อนบิตไปทางขวาและให้เท่ากับ

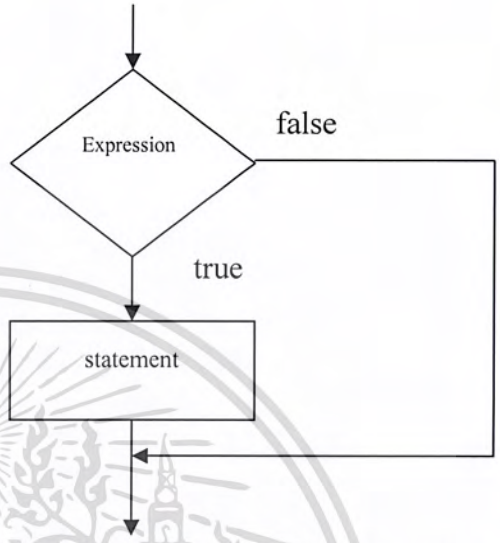
2.9.5 คำสั่งควบคุมในภาษาซี

การทำงานของโปรแกรมนั้นจะทำคำสั่งแต่ละคำสั่งเรียงลำดับกันไป และเราสามารถให้โปรแกรมตัดสินใจในการเลือกทำได้ หรือให้ทำงานใดงานหนึ่งซ้ำ ๆ ตามเงื่อนไขที่กำหนดได้โดยใช้คำสั่งควบคุม ในภาษาซีนั้นจะมีประโยคคำสั่งควบคุมที่ใช้ในการเลือกทำและทำงานซ้ำ ๆ ดังนี้

- **คำสั่ง IF / ELSE**

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

```
if(expression)
        statement;
```

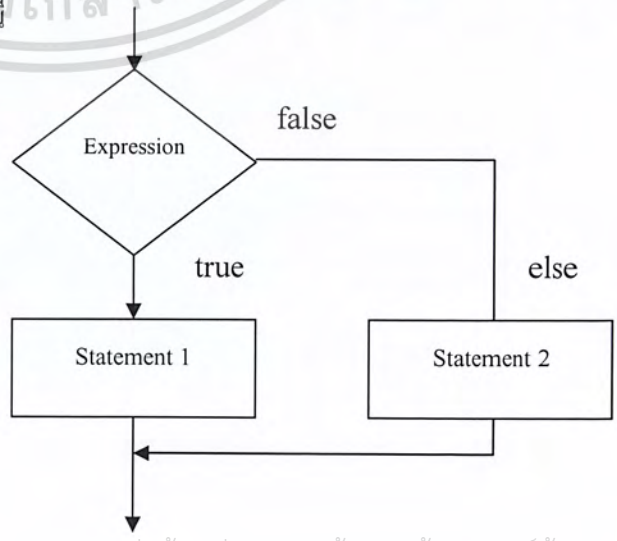


ประโยคนี้ใช้ในการทดสอบว่าจะทำสเตตเมนต์ที่ตามมาหรือไม่ ถ้าค่าใน expression เป็นจริงหรือมีค่าไม่เป็นศูนย์จะทำสเตตเมนต์ที่ตามมา ถ้าเป็นเท็จหรือมีค่าเป็นศูนย์จะไม่ทำ และสเตตเมนต์ที่จะทำงานนั้นอาจเป็นประโยคคำสั่งประโยคเดียว หรือเป็นสเตตเมนต์ซ้อนก็ได้ (ต้องมีปีกกาคลุม) ตัวอย่างเช่น ถ้าค่าของพอร์ต P1 มีค่าไม่เป็น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

```
if(P1 != 0)
        c = 0;
```

ถ้าหากเป็นการทำงานเลือกทำแบบมีสองทางเลือก และต้องการทำงานเพียงอย่างใดอย่างหนึ่งจะใช้ ประโยค if-else ซึ่งมีรูปแบบดังนี้

```
if(expression)
        statement1;
else statement2;
```



ตัวอย่างเช่น ถ้าจะทดสอบว่าค่าของพอร์ต P1 ถ้ามีค่าไม่เป็นศูนย์ให้ตัวแปร c มีค่าเป็น 20 ถ้ามีค่าเป็นศูนย์ให้ตัวแปร c มีค่าเป็นศูนย์ จะเขียนได้ดังนี้

```
if(P1 != 0)        c = 20;
      else         c = 0;
```

- **ประโยค switch**

การเลือกทำที่มีทางเลือกหลาย ๆ ทางเลือกนั้นเราสามารถนำประโยค if-else มาซ้อนกันได้ แต่จะทำให้มองดูเข้าใจได้ยาก ในภาษาซีจะมีประโยค switch ที่ใช้ในการเลือกทำอย่างใดอย่างหนึ่งจากหลาย ๆ ทางเลือกโดยมีรูปแบบของประโยคดังนี้

```
switch(k)
{
      case 1:     statement 1;
                  break;

      case 2:     statement 2;
                  break;

      case 3:     statement 3;
                  break;

          :       :
          :       :

     default:     statement n;
}
```

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

2.9.6 คำสั่งการทำซ้ำ

คำสั่งให้โปรแกรมทำงานซ้ำถือว่าเป็นประโยคคำสั่งควบคุมอย่างหนึ่ง การทำซ้ำหรือที่เรียกว่าการทำลูปนั้นจะมีประโยคคำสั่งอยู่สามประเภทคือ for, while และ do – while ซึ่งแต่ละแบบจะต่างกันตรงเงื่อนไขของการทำซ้ำ

- **คำสั่งทำซ้ำแบบ for**

ประโยคคำสั่งนี้จะใช้ในกรณีที่มีจำนวนรอบของการทำซ้ำที่แน่นอน โดยมีรูปแบบดังนี้

```
for(initialization; condition; increment)
      statement;
```

โดยที่ initialization เป็นค่ากำหนดเริ่มต้นให้กับตัวแปรของการทำลูป condition เป็นเงื่อนไขที่ใช้ทดสอบการทำซ้ำครั้งต่อไป ซึ่งจะเป็นการกระทำลอจิก increment เป็นการเพิ่มค่าให้ตัวแปรในการทำซ้ำแต่ละครั้ง สำหรับ statement จะเป็นสเตตเมนต์ของคำสั่งที่จะทำซ้ำ ซึ่งอาจเป็นสเตตเมนต์รวมก็ได้ ตัวอย่างเช่น

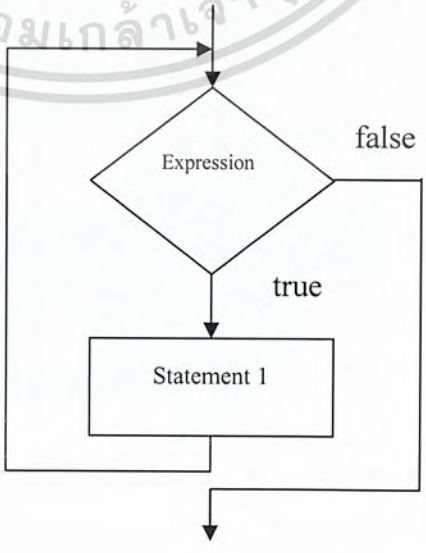
```
unsigned char x;
for(x=0;x<=255;x++)
      P1 = x;
```

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

- **คำสั่งทำซ้ำแบบ while**

การทำซ้ำแบบนี้จะตรวจสอบเงื่อนไขก่อนการทำซ้ำ ถ้าเงื่อนไขเป็นจริงจะทำสเตตเมนต์ที่กำหนด และทดสอบเงื่อนไขใหม่ ถ้าเงื่อนไขเป็นเท็จจะออกจากการทำซ้ำทันที โดยมีรูปแบบดังนี้

```
while(expression)
      statement1;
```



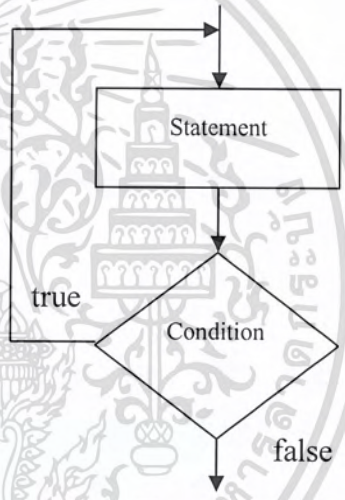
ในส่วนของ expression นั้นสามารถตรวจสอบค่าคงที่ได้ด้วย ถ้าค่าไม่เท่ากับศูนย์จะทำซ้ำ ถ้าค่าเท่ากับศูนย์จะไม่ทำซ้ำ ตัวอย่างเช่น ถ้าหากต้องการให้ค่าลอจิกทางพอร์ต P1 ของ MCS-51 ทุกบิตมีค่าลอจิกกลับไปมาจะเขียนคำสั่งได้ดังนี้

```
while(1)
{
      P1 = 0x55;
      P1 = 0x0AA;

}
```

- คำสั่งทำซ้ำแบบ do – while

การทำซ้ำประเภทนี้จะตรวจสอบเงื่อนไขภายหลังการทำสเตตเมนต์แต่ละครั้ง ถ้าหากมีเงื่อนไขเป็นเท็จจะออกจากการทำซ้ำทันทีโดยมีรูปแบบดังนี้

```
do{
      statement;
  }
while(condition);
```



ตัวอย่าง จงเขียนฟังก์ชันหน่วงเวลาโดยใช้คำสั่งการทำซ้ำให้กับ MCS-51

วิธีทำ การหน่วงเวลาจะทำได้โดยให้มีไมโครคอนโทรลเลอร์เสียเวลาไปเปล่า ๆ แต่ต้องไม่ทำให้ระบบทำงานผิดพลาด ซึ่งอาจทำได้โดยเขียนโปรแกรมให้มันวนลูปซ้ำ ๆ ดังนี้

```
void delay(unsigned int x)
{
      unsigned char j;

      while(x--)
      {

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

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

2.9.7 อาร์เรย์ พอยน์เตอร์ และสตรักเจอร์

ในการเขียนโปรแกรมด้วยภาษาซีถ้าหากต้องการใช้งานตัวแปรหลายตัวเราสามารถประกาศชื่อตัวแปรออกมาหลายตัวได้ เช่น x1, x2, x3, …, xn แต่ถ้าหากตัวแปรทุกตัวใช้เก็บข้อมูลประเภทเดียวกัน เราสามารถประกาศเป็นตัวแปรแบบอาร์เรย์ (array) ได้ การประกาศตัวแปรแบบอาร์เรย์สามารถทำได้ดังรูปแบบต่อไปนี้

ประเภทของข้อมูล ชื่ออาร์เรย์[ขนาดอาร์เรย์]

ถ้าเป็นอาร์เรย์แบบสองมิติสามารถประกาศได้ดังนี้

ประเภทของข้อมูล ชื่ออาร์เรย์[ขนาดอาร์เรย์][ขนาดอาร์เรย์]

ตัวอย่างเช่น `char x[8];`

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

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

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

ในระบบไมโครคอนโทรลเลอร์ เมื่อมีการประกาศตัวแปรอาร์เรย์ ระบบจะจองหน่วยความจำสำหรับเก็บตัวแปรอาร์เรย์นั้น ตัวแปรประเภทนี้สามารถนำไปประยุกต์ใช้ในการเขียนโปรแกรมแบบเปิดตารางได้

ในการอ้างถึงข้อมูลแต่ละเซลในอาร์เรย์สามารถใช้ตัวแปรพอยน์เตอร์ชี้ไปที่ตำแหน่งของอาร์เรย์ได้โดยตรง โดยพอยน์เตอร์จะเป็นตัวแปรที่ใช้เก็บแอดเดรสหรือตำแหน่งหน่วยความจำ ตัวดำเนินการที่ใช้กับพอยน์เตอร์คือ & และ * ตัวอย่างเช่น ถ้ามีการประกาศตัวแปรเป็น

```
char j;
```

จะเป็นการประกาศหน่วยความจำชื่อ j สำหรับเก็บตัวอักขระ เราสามารถอ้างแอดเดรสของตัวแปร j ได้ดังนี้

```
&j;
```

ถ้าหากมีการประกาศตัวแปรพอยน์เตอร์สำหรับเก็บแอดเดรสของตัวแปร j จะทำได้ดังนี้

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

เป็นการประกาศตัวแปรพอยน์เตอร์ชื่อ dptr และให้ชี้ไปที่แอดเดรสของตัวแปร j ถ้าหากมีการประกาศตัวแปรเป็น

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

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

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

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

```
x = *ip;
```

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

ในระบบไมโครคอนโทรลเลอร์บางครั้งจะต้องมีการออกแบบหน่วยความจำ ROM และ RAM ต่ออยู่ภายนอกตัวไมโครคอนโทรลเลอร์ และอุปกรณ์ต่าง ๆ ที่ต่ออยู่กับ

ไมโครคอนโทรลเลอร์จะมีตำแหน่งหรือแอดเดรสที่แน่นอน เราสามารถใช้พอยน์เตอร์ชี้ไปที่ตำแหน่งหน่วยความจำภายนอกได้ ตัวอย่างเช่น

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

เป็นการประกาศตัวแปรพอยน์เตอร์ชื่อ abs_ptr ให้ชี้ไปที่ตำแหน่ง 8000H

จากที่ผ่านมาจะพบว่าตัวแปรประเภทอาร์เรย์นั้นเราอาจมองว่าเป็นกลุ่มของข้อมูลได้ โดยข้อมูลในกลุ่มนั้นจะเป็นข้อมูลประเภทเดียวกัน ถ้าหากต้องการประกาศตัวแปรเป็นกลุ่มของข้อมูลที่ข้อมูลในกลุ่มนั้นเป็นชนิดต่างกัน จะต้องประกาศตัวแปรเป็นแบบโครงสร้าง หรือสตรักเจอร์ (structure) ซึ่งมีรูปแบบดังนี้

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

โดยกลุ่มของข้อมูลที่ประกาศขึ้นนั้นจะอยู่ในเครื่องหมายปีกกา และเราสามารถอ้างไปที่ข้อมูลตัวใดๆ ได้ โดยใช้เครื่องหมายจุด (.) ตัวอย่างเช่น

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

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

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

ถ้าหากมีการประกาศตัวแปรดังต่อไปนี้จะทำให้ผลลัพธ์มีค่าเหมือนกับตัวแปรแบบโครงสร้างในตัวอย่างที่ผ่านมา

```
#define uchar unsigned char
#define uint unsigned          ←——  ประกาศตัวแปรใหม่
struct stateform{
                    unsigned long s;
                    uint t;    ←——  เรียกใช้ในโครงสร้าง
                    uchar done;
                    };
```

จะเห็นว่าจะใช้ #define ประกาศตัวแปรประเภทข้อมูลใหม่ขึ้นมา และถูกเรียกใช้ในตัวแปรโครงสร้าง ถ้าหากต้องการให้ตัวแปรโครงสร้างนี้มีชื่อว่า state จะทำได้ดังนี้

```
struct stateform  state;
       ↑           ↑
ข้อมูลแบบโครงสร้าง   ชื่อตัวแปร
```

ถ้าหากต้องการประกาศตัวแปรแบบโครงสร้างหลายตัว สามารถทำได้ในรูปของอาร์เรย์ของโครงสร้าง (array of structures) ดังตัวอย่างต่อไปนี้

```
#define uchar unsigned char
#define uint unsigned
struct stateform{
                    unsigned long s;
                    uint t;
                    uchar done;
                    };
struct stateform state[20];
```

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

2.10 การเขียนโปรแกรมจัดการหน่วยความจำ

การจัดหน่วยของจำของไมโครคอนโทรลเลอร์ MCS-51 จะใช้การจัดหน่วยความจำแบบ Harvard โดยแยกหน่วยความจำออกเป็นส่วนของหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล สำหรับหน่วยความจำข้อมูลนั้นยังถูกแบ่งออกเป็นหน่วยความจำภายในตัว MCS-51 เอง และ

หน่วยความจำภายนอก ในการขยายพอร์ตเพิ่มเพื่อใช้กับตัว MCS-51 ตำแหน่งของพอร์ตจะเป็นตำแหน่งของหน่วยความจำภายนอกนี้ด้วย

สำหรับหน่วยความจำภายในของ MCS-51 ยังถูกแบ่งออกเป็นหน่วยความจำ 128 ไบต์แรก (ตำแหน่ง 00H – 7FH) ซึ่งสามารถเข้าถึงข้อมูลได้แบบ direct และ indirect ส่วนตำแหน่ง 80H ถึง FFH จะเป็นเรจิสเตอร์ฟังก์ชันพิเศษ หรือ SFR (Special Function Register) ซึ่งจะใช้การเข้าถึงข้อมูลแบบ direct

ในการใช้ภาษาซีกับ MCS-51 จะต้องมีการจัดวางหน่วยความจำให้ตัวแปรต่าง ๆ ที่ถูกประกาศขึ้นนั้นสามารถอยู่ในหน่วยความจำที่เหมาะสมได้ โดยการจัดหน่วยความจำของตัวคอมไพเลอร์มีรูปแบบดังนี้

TINY เป็นการจัดหน่วยความจำให้ตัวคอมไพเลอร์มองว่าไม่มีหน่วยความจำแรมภายนอก ซึ่งจะใช้กับ MCS-51 แบบที่เป็นชิงเกิลลิป เช่น เบอร์ 89C8051

SMALL หน่วยความจำแบบนี้คอมไพเลอร์จะมองว่าแรมและรอมอยู่ซ้อนกันภายในพื้นที่ 64 กิโลไบต์ ตัวแปรต่าง ๆ ที่ถูกประกาศใช้งาน หรือประกาศเป็นแบบโลเคิลจะถูกกำหนดอยู่ในหน่วยความจำข้อมูลภายใน (internal data memory) ขนาดของหน่วยความจำสแตกจะถูกกำหนดตามที่ใช้งานจริง รูปแบบหน่วยความจำโหมดนี้จะเป็นค่าที่โปรแกรมคอมไพเลอร์ได้กำหนดไว้แล้ว

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

LARGE หน่วยความจำแบบนี้จะมีขนาดใหญ่ ตัวแปรทั้งหมดจะถูกกำหนดอยู่ในหน่วยความจำข้อมูลภายนอก

การเขียนโปรแกรมด้วยภาษาซีให้กับไมโครคอนโทรลเลอร์ด้วย C51 นี้สามารถประกาศตัวแปรต่างๆ แบบเจาะจงได้ โดยใช้แมโครที่ถูกกำหนดไว้ในไฟล์ absacc.h ดังต่อไปนี้

CODE	แทนพื้นที่หน่วยความจำโปรแกรมภายนอก
DATA	แทนพื้นที่หน่วยความจำข้อมูลภายใน 128 ไบต์แรก (ตำแหน่ง 00H ถึง 7FH)
IDATA	แทนเนื้อที่หน่วยความจำข้อมูลภายใน 256 ไบต์
BDATA	แทนตำแหน่งของหน่วยความจำระดับบิต 128 บิต (20H ถึง 2FH) ของหน่วยความจำภายใน

XDATA	แทนตำแหน่งหน่วยความจำภายนอก
PDATA	แทนตำแหน่งหน่วยความจำภายนอก 256 ไบต์แรก

ในการกำหนดชื่อตัวแปรต่าง ๆ เพื่อแทนตำแหน่งหน่วยความจำนั้นมีรูปแบบดังนี้

ชนิดของตัวแปร ชื่อแมโคร ตัวแปร

ตัวอย่างเช่น

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

การกำหนดบรรทัดแรกจะทำให้คำว่า COMPUTER ถูกเก็บอยู่ในหน่วยความจำโปรแกรมภายนอก บรรทัดที่สองจะทำให้อาร์เรย์ AB ถูกประกาศเอาไว้ในหน่วยความจำข้อมูลภายนอก บรรทัดที่สามจะทำให้ตัวแปร x,y และ z ถูกประกาศเอาไว้ในหน่วยความจำข้อมูลภายใน

ในการเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์ติดต่อกับพอร์ตจะต้องทำการโอนย้ายข้อมูลกับพอร์ตตำแหน่งแอดเดรสนั้น ๆ ซึ่งการอ้างแอดเดรสของพอร์ตสามารถใช้แมโครได้เช่นกัน ทำให้สามารถแทนพอร์ตด้วยตัวแปรต่าง ๆ ได้ โดยแมโครที่ใช้จะเก็บอยู่ในไฟล์ absacc.h โดยมีชื่อดังต่อไปนี้

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

ในการกำหนดตัวแปรแทนตำแหน่งต่าง ๆ จะต้องใช้ร่วมกับไดเร็กทีฟ #define ดังตัวอย่างต่อไปนี้

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

จะเป็นการแทนตัวแปร port8 ด้วยหน่วยความจำภายนอกตำแหน่ง 8000H ถ้าหากต้องการส่งค่า FF ไปยังหน่วยความจำตำแหน่ง 8000H สามารถทำได้ดังนี้

```
port8 = 0x0FF;
```

ถ้าหากต้องการอ่านข้อมูลจากตำแหน่ง 8000H มาเก็บในตัวแปร x สามารถทำได้ดังนี้

```
char x;
x = port8;
```

2.10.1 การเข้าถึงข้อมูลระดับบิตและเรจิสเตอร์พิเศษ

ในโปรแกรม C51 ถ้ามีการประกาศตัวแปรและกำหนดให้เป็นหน่วยความจำแบบ BDATA จะเป็นการใช้งานในแอดเดรสหน่วยความจำที่สามารถเข้าถึงข้อมูลในระดับบิตได้ นอกจากนี้เราสามารถใช้คำสั่ง sbit ของ C51 กำหนดตัวแปรแบบบิตขึ้นมาได้อีกด้วย ตัวอย่างเช่น

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

บรรทัดแรกจะเป็นการประกาศตัวแปรชื่อ ibase สำหรับเก็บเลขจำนวนเต็มลงในหน่วยความจำ พื้นที่ที่สามารถเข้าถึงข้อมูลในระดับบิตได้ บรรทัดต่อมาประกาศตัวแปรอาร์เรย์จำนวน 4 ไบต์ลงในหน่วยความจำระดับบิต ในบรรทัดที่สามจะใช้คำสั่ง sbit กำหนดตัวแปรขึ้นมาใหม่ แทนบิตต่ำสุด โดยใช้สัญลักษณ์ carat (‘^’) ซึ่งจะทำให้ตัวแปรชื่อ mybit0 แทนบิตที่ 0 ของหน่วยความจำชื่อ ibase บรรทัดที่สี่จะให้ตัวแปรชื่อ mybit15 แทนบิตสูงสุดของตัวแปร ibase และบรรทัดสุดท้ายจะให้ตัวแปรชื่อ Ary07 แทนบิตที่ 7 ของ bary[0] สำหรับขนาดของบิตของตัวแปรจะขึ้นกับประเภทของข้อมูลที่กำหนด เช่น ถ้าเป็นข้อมูลแบบ char และ unsigned char ข้อมูลบิตจะอยู่ในช่วง 0 ถึง 7 บิต ถ้าเป็นแบบ int, unsigned int,short และ unsigned short จะอยู่ในช่วง 0 ถึง 15 บิต

ถ้าต้องการกำหนดตัวแปรชื่อ OUT แทนบิตสูงสุดของพอร์ต P1 ของ MCS-51 สามารถทำได้ดังนี้

```
sbit OUT = P1^7;
```

พื้นที่หน่วยความจำภายในของ MCS-51 ส่วนหนึ่งจะเป็นพื้นที่ของเรจิสเตอร์ฟังก์ชันพิเศษ (SFR) เช่น เรจิสเตอร์ควบคุมไทเมอร์ เรจิสเตอร์ตัวนับ พอร์ตอนุกรม พอร์ตอินพุตเอาต์พุตต่าง ๆ ซึ่งอยู่ในหน่วยความจำภายในตั้งแต่ตำแหน่ง 0x80 ถึง 0xFF ถ้าหากเราต้องการกำหนดชื่อใหม่แทนสัญลักษณ์เหล่านั้น สามารถใช้คำสั่ง sfr ในการประกาศได้ โดยใช้เครื่องหมายเท่ากับ (=) ระบุค่าแอดเดรสให้กับตัวแปรนั้น เช่น

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

แต่ถ้าหากต้องการกำหนดชื่อแทนเรจิสเตอร์ฟังก์ชันพิเศษที่มีขนาด 16 บิต จะต้องใช้คำสั่ง sfr16 และระบุแอดเดรสไบต์ต่ำให้กับตัวแปรนั้น ตัวอย่างเช่น

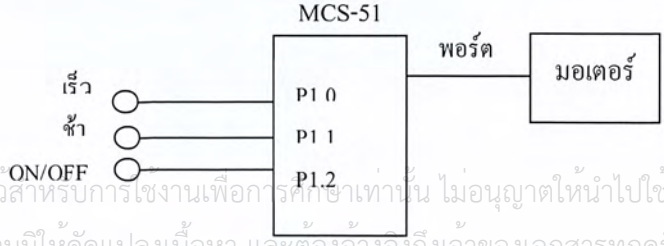
```
sfr16 T2 = 0xcc;                /* Timer 2 : T2L 0CCH , T2H 0CDH */
```

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

สำหรับในการเขียนโปรแกรมด้วย C51 นั้นจะเห็นว่าเรจิสเตอร์บางตัว ตำแหน่งบิตบางตำแหน่ง เราสามารถอ้างชื่อขึ้นมาได้เลย เนื่องจากในไฟล์ res51.h ที่ได้ include ขึ้นมาเป็นไฟล์ที่กำหนดเรจิสเตอร์ต่าง ๆ เอาไว้แล้ว เช่น P0, P1, P2 และ P3 สำหรับตำแหน่งบิตของเรจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตได้ เช่น TF1, TR1, TF0, TR0, IE1, IT1, IE0 และ IT0 ซึ่งเป็นบิตที่อยู่ในเรจิสเตอร์ TCON ก็กำหนดไว้แล้วเช่นกัน โดยในไฟล์ reg51.h ส่วนหนึ่งได้กำหนดไว้ดังนี้

```
/* BYTE register */
sfr P0 = 0x80;
sfr P1 = 0x90;
sfr P2 = 0xA0;
sfr P3 = 0xB0;
/* BIT register */
/* TCON */
sbit TF1 = 0x8F;
sbit TR1 = 0x8E;
sbit TF0 = 0x8D;
sbit TR0 = 0x8C;
```

ตัวอย่าง ถ้าหากมีระบบควบคุมความเร็วของมอเตอร์ระบบหนึ่ง โดยมีสวิตช์ควบคุมอยู่สามตัวคือ สวิตช์กำหนดให้มีความเร็วสูงต่ออยู่กับบิต P1.0 สวิตช์กำหนดให้มีความเร็วต่ำต่ออยู่กับบิต P1.1 และ สวิตช์ปิดเปิดระบบต่ออยู่กับบิต P1.2 ดังรูป



ถ้าหากมีการเขียนโปรแกรมย่อยควบคุมความเร็วของมอเตอร์ชื่อ speed1() สำหรับให้มีความเร็วสูง และ speed2() สำหรับควบคุมให้มีความเร็วต่ำ จะสามารถเขียนโปรแกรมควบคุมได้ดังตัวอย่างต่อไปนี้

```
#include<req51.h>

#define ON 1

sbit MOTOR = P1^2;

sbit SPEED1 = P1^0;

sbit SPEED2 = P1^1;

void speed1();

void speed2();
main()
{
     do
     {
          if SPEED1 == ON
               speed1();
          if SPEED2 == ON
               speed2();
          MOTOR == ~ON;
     }
}
```

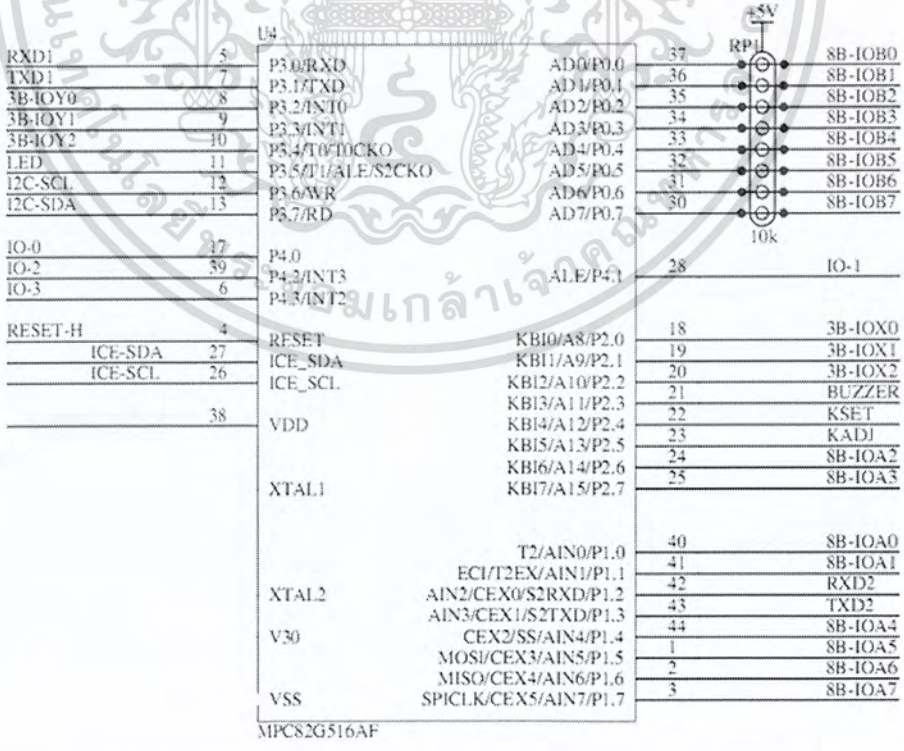
จากโปรแกรมจะเห็นว่าจะใช้ sbit มากำหนดบิตขึ้นมาหนึ่งบิต ถ้าหากสวิตช์ตัวแรกซึ่งต่ออยู่กับ SPEED1 ถูกกดจะทำให้ระบบทำโปรแกรมย่อย speed1() ซึ่งจะทำให้มอเตอร์หมุนด้วยความเร็วสูง ถ้าสวิตช์ตัวที่สองถูกกดจะทำให้ระบบทำโปรแกรมย่อย speed2() ซึ่งจะทำให้มอเตอร์หมุนด้วยความเร็วต่ำ

บทที่ 3
หลักการออกแบบ

สำหรับการออกแบบโครงงานนี้มี 3 ขั้นตอนใหญ่ ๆ ที่สำคัญคือ การออกแบบวงจรชุดทดลอง การออกแบบโปรแกรม และการออกแบบการทดลองสำหรับฝึกเขียนโปรแกรม ในส่วนของการออกแบบชุดทดลองนั้นผู้พัฒนาจะต้องศึกษาไมโครคอนโทรลเลอร์ตัวที่สนใจ และอุปกรณ์ประกอบต่าง ๆ ที่จะนำมาสร้างเป็นบอร์ต เช่น ชิปสัญญาณนาฬิกา ชิปสำหรับสร้างเป็นพอร์ตตัวเซนเซอร์ที่เลือกใช้ และอุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ

3.1 การจัดการขาของ MPC82G516

ในการออกแบบสร้างชุดทดลองหรือบอร์ตควบคุมขึ้นมานั้นขั้นตอนแรกจะต้องศึกษาการจัดขาของไอซีไมโครคอนโทรลเลอร์ตัวที่สนใจนำมาสร้างชุดทดลองก่อน เพื่อที่จะได้ศึกษาอุปกรณ์ประกอบต่อไป ไมโครคอนโทรลเลอร์ MPC82G516A มีทั้งหมด 40 ขา โดยจะประกอบด้วยขาพอร์ต ขาแอดเดรส ขาข้อมูล ขาอินเทอร์รัปต์ และขาควบคุมต่างๆ มีการกำหนดดังนี้



รูปที่ 3.1 แสดงขาต่างๆของไมโครคอนโทรลเลอร์ MPC82G516A

ขาพอร์ต

- พอร์ต P0 มี bit ที่ 0 – 7 ทำงานได้ 2 หน้าที่ คือ ใช้เป็นขาพอร์ต และแปลงสัญญาณอนาลอกเป็นดิจิตอล
- พอร์ต P1 มี bit ที่ 0 – 7 นอกจากจะใช้เป็นขาพอร์ตแล้ว เป็นสัญญาณนาฬิกา เป็นตัวนับ ตัวจับเวลา และแปลงสัญญาณอนาลอกเป็นดิจิตอล
- พอร์ต P2 มี bit ที่ 0 – 7 นอกจากจะใช้เป็นขาพอร์ตแล้วยังใช้เชื่อมต่อแอดเดรสอีกด้วย
- พอร์ต P3 มี bit ที่ 0 – 7 นอกจากจะใช้เป็นขาพอร์ตแล้ว ยังมีหน้าที่เป็นสัญญาณควบคุม รับส่งข้อมูล และใช้สื่อสารกับอุปกรณ์ภายนอกได้
- พอร์ต P4 มี bit ที่ 0 – 7 นอกจากจะใช้เป็นขาพอร์ตแล้ว ยังใช้รับสัญญาณอินเทอร์รัปต์ได้อีกด้วย

V30

ขาสัญญาณเอาท์พุตจากวงจรกำเนิด LDO ภายใน

VDD

ขาที่ใช้ต่อกับแหล่งจ่ายไฟ

VSS

ขาที่ใช้ต่อกราวด์

ALE (Address Latch Enable)

เป็นขาเอาต์พุตที่แอกทีฟ logic 1 โดย CPU จะส่งสัญญาณนี้ออกมาเอง เนื่องจากพอร์ต P0 สามารถทำงานได้ 2 หน้าที่ คือ เป็นขาแอดเดรส และขาข้อมูล ตัว MPC82G516A จะใช้สัญญาณของ ALE มัลติเพล็กซ์พอร์ต P0 ว่าขณะนั้นจะใช้งานเป็นขาแอดเดรส หรือขาของข้อมูล

RST

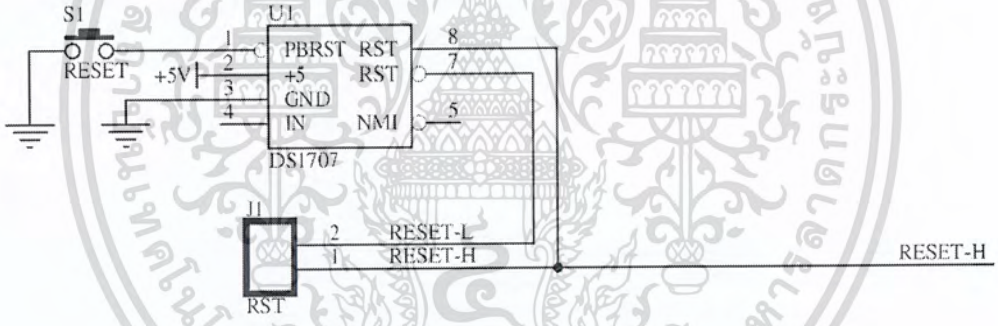
เป็นขารีเซต(reset) เมื่อต้องการรีเซต ตัว MPC82G516A จะต้องทำให้ขานี้เปลี่ยนจาก logic 0 เป็น logic 1 จะทำให้ CPU เริ่มต้นทำโปรแกรมที่ตำแหน่ง 0000 การรีเซต MPC82G516A จะมี 2 แบบ คือ การรีเซตเมื่อเริ่มต้นจ่ายไฟให้กับวงจร กับ การรีเซตโดยกดปุ่มรีเซต

XTAL1 และ XTAL2

แม้ว่า ตัว MPC82G516A จะมีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิป แต่ก็ต้องต่อกับคริสตอล(crystal) ภายนอกเพื่อควบคุมสัญญาณนาฬิกา โดยต่อเข้าทางขา XTAL1 และ XTAL2 แต่ถ้าต้องการสัญญาณนาฬิกาจากภายนอก ก็ทำได้โดยต่อเข้าทางขา XTAL1

3.2 วงจรรีเซตไมโครคอนโทรลเลอร์

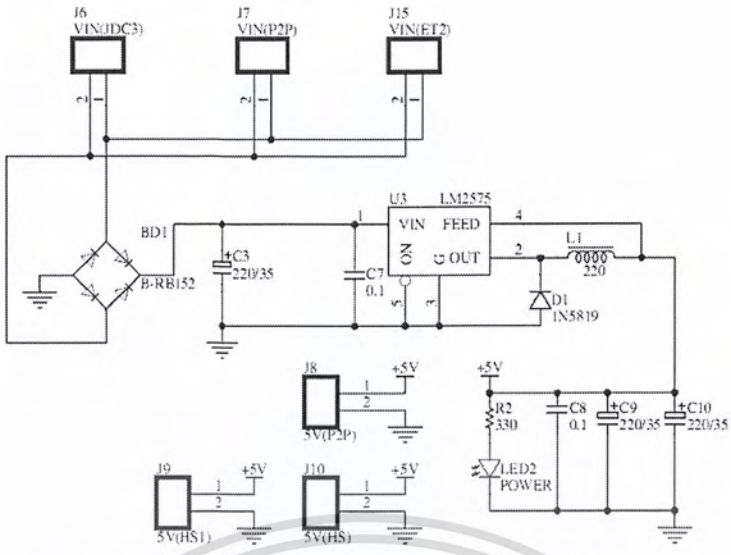
การรีเซต (reset) เป็นกระบวนการที่ทำให้ไมโครคอนโทรลเลอร์เริ่มต้นทำงานใหม่ โดยการรีเซตนี้จะช่วยแก้ไขปัญหาการทำงานผิดพลาดของโปรแกรมได้ เมื่อไมโครคอนโทรลเลอร์ถูกรีเซต ตัวมันจะกระโดดไปทำโปรแกรมที่เก็บไว้ในตำแหน่ง 0000H ดังนั้น ตำแหน่งเริ่มต้นของโปรแกรมจะต้องเก็บเอาไว้ที่แอดเดรส 0000H ของหน่วยความจำโปรแกรมภายใน สำหรับวงจรรีเซตนั้นออกแบบได้หลายวิธี อาจใช้สวิตซ์ตัวเดียวมาต่อ หรือใช้วงจร RC ในการรีเซต สำหรับโครงงานนี้จะเลือกใช้ไอซี DS1707 มาสร้างเป็นวงจรรีเซต เนื่องจากเป็นวงจรที่มีความเที่ยงตรงสูง โดยวงจรที่ออกแบบขึ้นเป็นดังรูปที่ 3.2



รูปที่ 3.2 แสดงวงจรรีเซตไมโครคอนโทรลเลอร์

3.3 วงจรจ่ายไฟให้กับบอร์ดไมโครคอนโทรลเลอร์

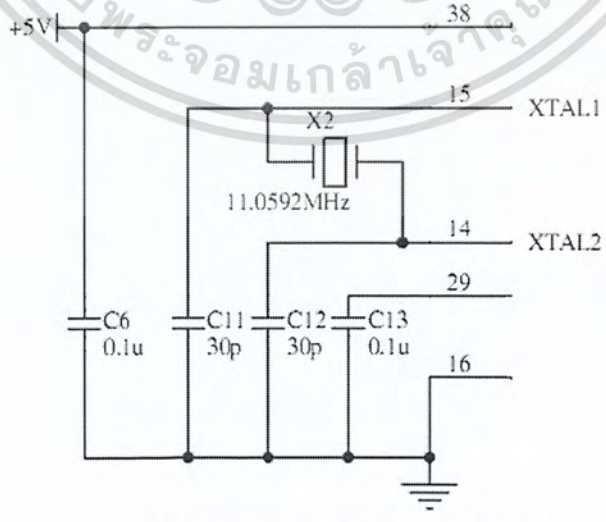
ภาคจ่ายไฟสามารถรับกระแสไฟฟ้าได้ทั้ง AC และ DC ตั้งแต่ช่วงความต่างศักย์ 7-24 V ผ่านวงจรไดโอด เปลี่ยนไฟฟ้ากระแสสลับเป็นกระแสตรงผ่าน IC เปลี่ยนแรงดันไฟฟ้าตั้งแต่ 7V ขึ้นไป เป็นไฟฟ้ากระแสตรงขนาด 5V สำหรับใช้ในวงจร โดยมีตัวเก็บประจุใช้ในการกรองกระแสให้ระบบ ดังรูป



รูปที่ 3.3 แสดงวงจรจ่ายไฟให้กับบอร์ดไมโครคอนโทรลเลอร์

3.4 วงจรสัญญาณนาฬิกาที่ป้อนให้ไมโครคอนโทรลเลอร์ทำงาน

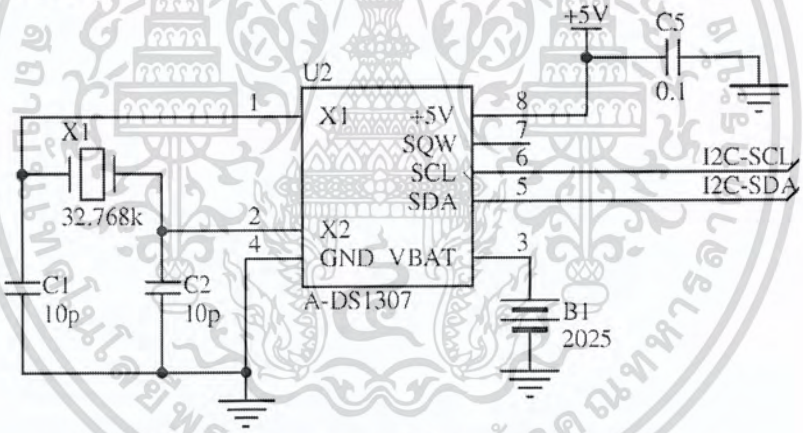
ภายในตัวไมโครคอนโทรลเลอร์ จะมีวงจรออสซิลเลเตอร์(Oscillator) สำหรับสร้างสัญญาณนาฬิกาอยู่ภายใน การกำหนดจังหวะเวลาให้ไมโครคอนโทรลเลอร์ก็ทำเพียงต่อกับคริสตอล(Crystal)ที่อยู่ภายนอกให้กับมันเท่านั้น โดยปกติจะใช้คริสตอลความถี่ 11.0592 MHz (ซึ่งเป็นความถี่ที่เหมาะสม) และตัวประจุเพื่อลดสัญญาณรบกวน โดยต่อเข้าทางขา XTAL1 และ XTAL2 สำหรับวงจรที่ออกแบบขึ้นเป็นดังรูปที่ 3.5 ค่ าความถี่ที่เลือกใช้นี้ทำให้บอร์ตไมโครคอนโทรลเลอร์ที่สร้างขึ้นสามารถสื่อสารข้อมูลแบบอนุกรมกับอุปกรณ์อื่น ๆ ได้อย่างเม่นยำ



รูปที่ 3.4 แสดงวงจรสัญญาณนาฬิกาที่ป้อนให้กับไมโครคอนโทรลเลอร์

3.5 วงจรฐานเวลาจริง (Real Time Clock)

วงจรฐานเวลาจริง หรือ Real Time Clock (RTC) เป็นวงจรนาฬิกาที่สามารถโปรแกรมเวลาได้ และทำงานได้ตลอดเวลา ทำให้ระบบไมโครคอนโทรลเลอร์สามารถอ่านค่าเวลาจริงออกมาได้ วงจรที่ออกแบบขึ้นเลือกใช้ชิปไอซีเบอร์ DS1307 ซึ่งชิปนี้จะมีวงจรกำเนิดสัญญาณนาฬิกาของมันเอง และทำงานอิสระจากระบบไมโครคอนโทรลเลอร์ มีแบตเตอรี่สำรองขนาดเล็กเพื่อเป็นแหล่งจ่ายไฟขณะที่ระบบหลักหยุดทำงาน และที่สำคัญชิปเหล่านี้มีความเที่ยงตรงสูงมาก เป็นไอซีฐานเวลาที่สามารถเชื่อมต่อกับไมโครคอนโทรลเลอร์แบบอนุกรมได้โดยตรง รับพลังงานจากแหล่งจ่ายไฟหรือแบตเตอรี่ กินพลังงานต่ำมากและจะรันอยู่ตลอดเวลา ไม่ว่าเครื่องจะปิดหรือเปิดอยู่ มีปฏิทินเวลาแบบที่ให้ข้อมูลแบบ BCD สามารถใช้ข้อมูลเกี่ยวกับเวลา เช่น วินาที นาที ชั่วโมง (ทั้งแบบ 24 ชั่วโมง/ 12 ชั่วโมง พร้อมระบุค่า AM/PM) และบอก วัน เดือน ปี ได้ โดยจะมีการปรับวันที่โดยอัตโนมัติ ในแต่ละเดือนจะแสดงวันได้สูงสุดไม่เกิน 31 วันและจะปรับวันต่างๆให้อย่างถูกต้องเมื่อครบปี การใช้งานกับไมโครคอนโทรลเลอร์จะใช้การส่งข้อมูลและแอดเดรสของค่าต่างๆ แบบอนุกรม โดยมีขาหนึ่งเป็นขาสัญญาณ อีกขาหนึ่งเป็นขากำหนดสัญญาณนาฬิกา วงจรที่ออกแบบขึ้นเป็นดังรูป

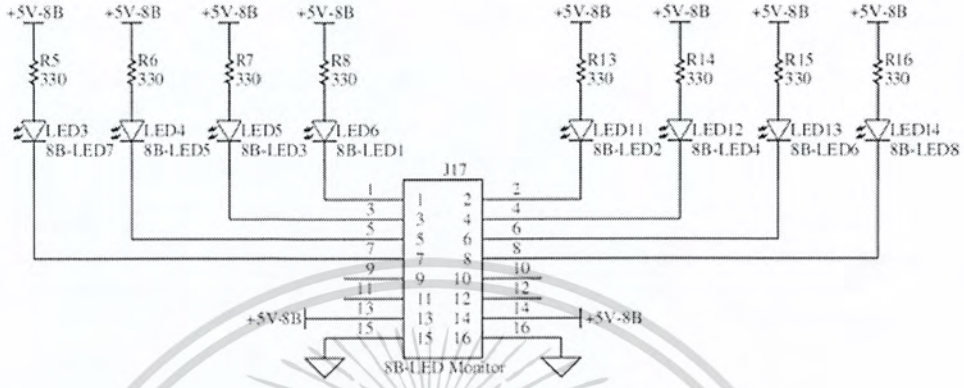


รูปที่ 3.5 แสดงวงจรเวลาจริง(Real-Time Clock)

3.6 วงจรขับหลอด LED

ขาแต่ละพอร์ตของไมโครคอนโทรลเลอร์นั้นสามารถจ่ายกระแสไฟฟ้าหรือเรียกว่ากระแสซอร์ซ (Source Current) ได้สูงสุด 10 mA ค่ากระแสนี้สามารถนำไปสื่อสารกับอุปกรณ์ดิจิตอลต่างๆ แต่ถ้าหากทุกบิตจ่ายกระแสพร้อมกันจะทำให้มีกระแสไฟฟ้าไหลผ่านไอซีเป็นจำนวนมาก ในการใช้งานทั่วไปแล้วมักจะนำบัฟเฟอร์มาช่วยขับกระแสให้กับอุปกรณ์ภายนอกด้วย นอกจากนี้บิตพอร์ตบางบิตสามารถรับกระแสไฟฟ้าเข้าไปภายในได้หรือที่เรียกว่ากระแสซิงค์ (Sink Current) สำหรับวงจรเชื่อมต่อพอร์ตกับหลอด LED จะออกแบบให้ LED สว่างด้วยกระแสที่ไหลเข้าไมโครคอนโทรลเลอร์ โดยออกแบบวงจรดังรูปที่ 3.7 ซึ่งแสดงการต่อพอร์ตกับหลอด LED การ

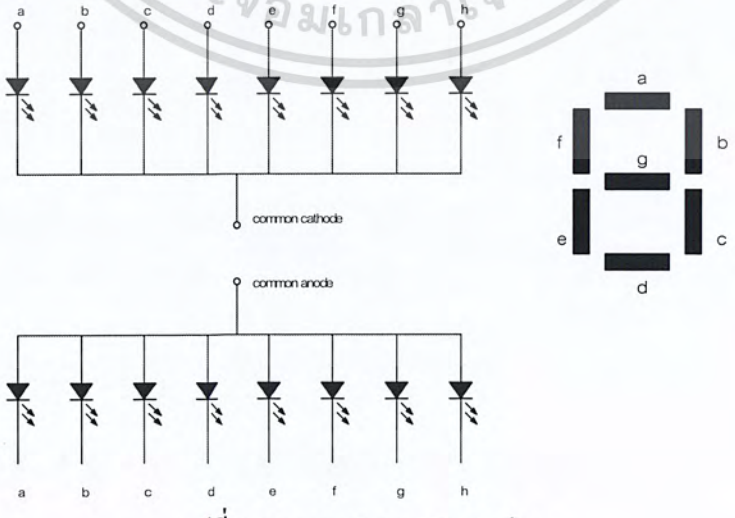
แสดงผลของไมโครคอนโทรลเลอร์ สามารถทำได้โดยการนำหลอด LED มาต่อ หลอด LED นี้เป็นอุปกรณ์ที่ใช้แสดงผลลอจิก 0 หรือลอจิก 1 โดยที่ LED จะสว่างเมื่อมีเอาต์พุตเป็นลอจิก 0 จะทำให้มีกระแสไฟฟ้าไหลมาจากแรงดัน 5 โวลต์ ผ่านตัวต้านทาน 330 โอห์มที่ใช้สำหรับควบคุมกระแสไฟฟ้าก่อนไหลผ่านไปเข้าไปยังบิตพอร์ตของไมโครคอนโทรลเลอร์



รูปที่ 3.6 แสดงวงจรขับหลอด LED

3.7 วงจรหลอด LED 7 ส่วน

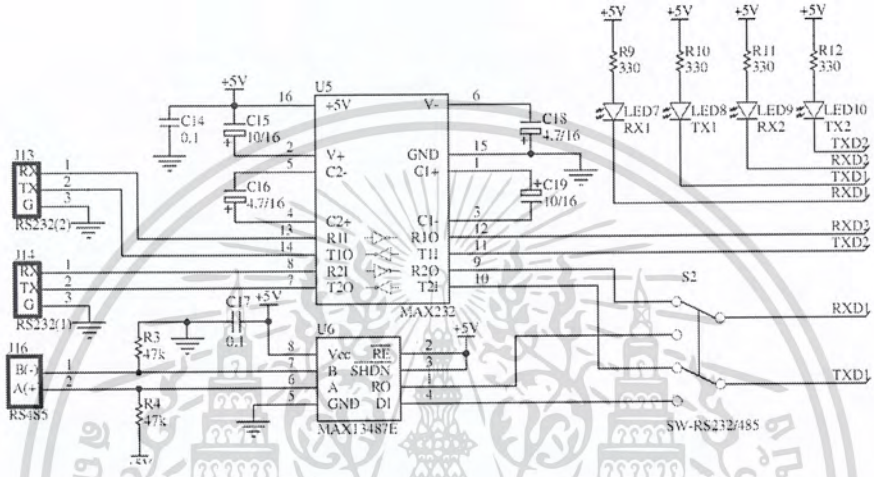
หลอด LED 7 ส่วน (LED 7-segment) จะเกิดจากการนำหลอด LED จำนวน 7 หลอดมาต่อเรียงกันเพื่อให้แสดงผลเป็นตัวเลขได้ ตัว LED แต่ละหลอดหรือแต่ละเซกเมนต์จะมีชื่อประจำคือ a, b, c, d, e, f และ g ถ้าหลอด LED ทั้ง 7 หลอดต่อกันโดยนำขาแอโนดมาต่อร่วมกันจะเรียกว่าแบบแอโนดร่วม (common anode seven-segment) ถ้าเป็นแบบแคโทดต่อร่วมกันจะเรียกว่าแบบแคโทดร่วม (common cathode seven-segment) ดังแสดงในรูปที่ 3.8 ถ้าหากต้องการให้หลอด LED สว่างเป็นเลขใดก็ทำโดยให้หลอดแต่ละหลอดสว่างประกอบกันให้เป็นเลขนั้น ถ้าหากต้องการให้หลอด LED ทุกหลอดสว่างพร้อมกันหมดจะแสดงเป็นเลข 8



รูปที่ 3.7 วงจรหลอด LED 7 ส่วน

3.8 วงจรสื่อสารกับคอมพิวเตอร์ผ่านพอร์ตอนุกรม RS232 และRS485

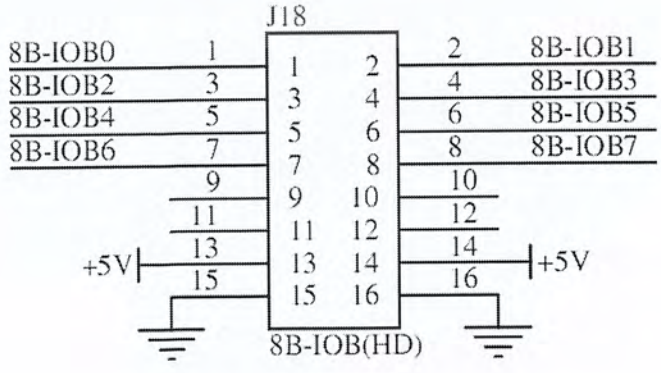
โดยทั่วไปแล้วถ้าหากต้องการให้ไมโครคอนโทรลเลอร์ติดต่อกับคอมพิวเตอร์ PC ตามมาตรฐาน RS232 จะต้องออกแบบวงจรอิเล็กทรอนิกส์เพิ่มเติม แต่ในปัจจุบันจะใช้ไอซี MAX232 ทำหน้าที่ปรับระดับแรงดันระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ ให้สามารถสื่อสารกันได้อย่างเข้าใจ ส่วนพอร์ตสื่อสาร RS485 จะใช้ไอซี MAX13487E ทำหน้าที่ในการปรับระดับแรงดันระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์เช่นกัน ในการรับส่งข้อมูลแบบอนุกรมของ MPC82G516A จะรับส่งออกมาทางขา RXD และ TXD



รูปที่ 3.8 แสดงวงจรติดต่อกับคอมพิวเตอร์ผ่านพอร์ตอนุกรม RS232 และ RS485

3.9 คอนเน็กเตอร์เพิ่มเติม

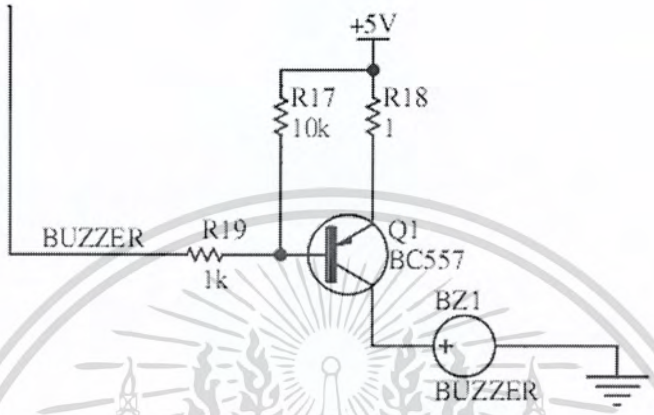
ชุดทดลองนี้ยังต่อบิตพอร์ตเพิ่มเติมสำหรับต่ออุปกรณ์ภายนอกอื่น ๆ ที่อาจมีได้ในอนาคต และใช้เป็นตัวขับสัญญาณให้กับอุปกรณ์ภายนอก อย่างเช่น หลอด LED มอเตอร์ เป็นต้น โดยออกแบบขั้วต่อไว้ดังรูปที่ 3.8



รูปที่ 3.9 แสดงคอนเน็กเตอร์เพิ่มเติม

3.10 วงจรขับลำโพงให้ทำงาน

ถ้าหากระบบไมโครคอนโทรลเลอร์ที่สร้างขึ้นต้องการให้มีเสียงออกมาด้วย ก็สามารถนำลำโพงขนาดเล็ก(Buzzer) มาต่อที่บิตพอร์ต บิตใดบิตหนึ่ง Buzzer ก็จะส่งเสียงออกมาตามคำสั่งการควบคุมของโปรแกรมการที่กระแสไหลผ่านลำโพงกลับไปกลับมา จะทำให้ลำโพงนั้นสั่นออกมาเป็นเสียง



รูปที่ 3.10 แสดงวงจรขับลำโพง

3.11 ไทเมอร์ (Timer)

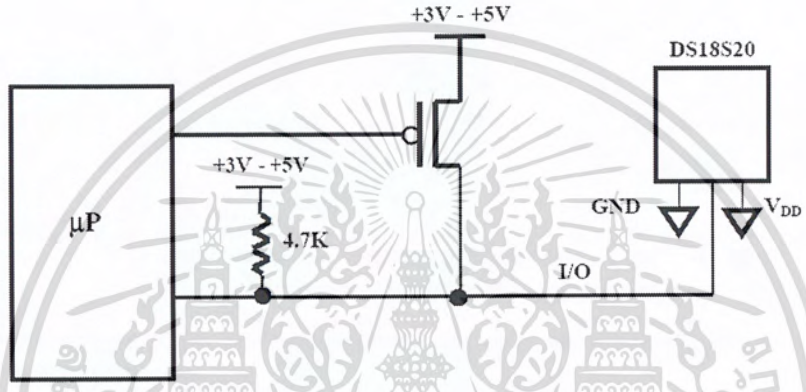
ใน MCS-51 ทุกตัวจะมีไทเมอร์เคาน์เตอร์อยู่ภายใน บางเบอร์จะมีสามตัว แต่ส่วนใหญ่แล้วจะมีสองตัวคือไทเมอร์ 0 (Timer 0) และไทเมอร์ 1 (Timer 1) โดยเป็นเรจิสเตอร์ขนาด 16 บิต (Timer 0 แบ่งเป็น TH0 และ TL0 ส่วน Timer1 แบ่งเป็น TH1 และ TL1) และจะสามารถโปรแกรมให้เป็นตัวนับหรือตัวจับเวลาก็ได้ การใช้งานเป็นไทเมอร์จะเป็นการตั้งเวลาด้วยการโปรแกรม เมื่อเวลาถึงค่าที่กำหนดไทเมอร์จะแสดงผลออกมาทางบิตแฟลก (TF)

3.12 รีเลย์ (Relay)

รีเลย์เป็นอุปกรณ์อิเล็กทรอนิกส์ซึ่งทำหน้าคล้ายสวิตช์ แต่ใช้หลักการหน้าสัมผัส แต่รีเลย์นั้นจะถูกควบคุมด้วยกระแสไฟฟ้า การทำงานของรีเลย์คือ เมื่อมีกระแสไฟฟ้าไหลผ่านขดลวดจะทำให้ขดลวดเกิดสนามแม่เหล็กไปดึงแผ่นหน้าสัมผัสให้ดึงลงมาแตะหน้าสัมผัสอีกอันทำให้มีกระแสไหลผ่านหน้าสัมผัสไปได้ เรียกว่า วงจรปิด

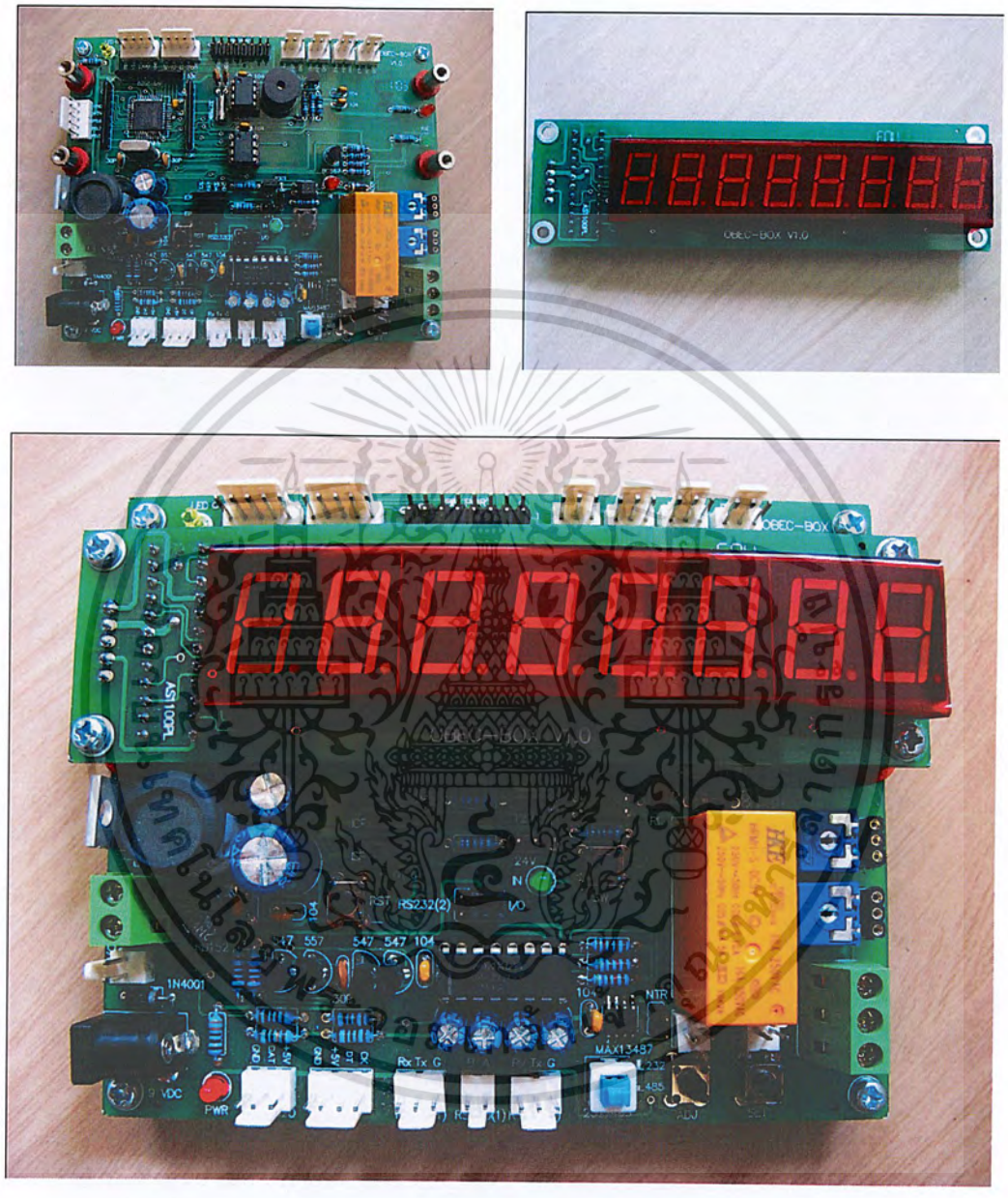
3.13 วงจรต่อกับตัววัดอุณหภูมิ

ชิปอุณหภูมิที่ใช้คือ DS18S20 เป็นตัววัดอุณหภูมิแบบดิจิตอล ที่มีความเที่ยงตรง และแม่นยำในการวัดสูง การสื่อสารและควบคุม DS18S20 นั้นสามารถทำได้โดยใช้บัสข้อมูลแบบ 1-wire ของ Dallas Semiconductor ซึ่งใช้สายสัญญาณเพียงแค่เส้นเดียวเท่านั้น นอกจากนี้ DS18S20 ยังสามารถทำงานในโหมดพาราสิต (Parasite Power Mode) ซึ่งเป็นการทำงานโดยไม่ใช้ไฟเลี้ยง แต่ใช้พลังงานจากสายสัญญาณ 1-wire ซึ่งมีประโยชน์มากสำหรับการวัดอุณหภูมิระยะไกล หรือในการใช้งานในที่ ๆ มีเนื้อที่จำกัด



รูปที่ 3.11 แสดงวงจรต่อกับตัววัดอุณหภูมิ

แผงวงจรไมโครคอนโทรลเลอร์หรือชุดทดลองที่ออกแบบ เป็นดังรูปที่ถ่ายมานี้



รูปที่ 3.12 แสดงชุดทดลอง OBEC-BOX V1.0

บทที่ 4
การใช้งานโปรแกรม

4.1 แหล่งที่มา

แผงวงจรไมโครคอนโทรลเลอร์หรือชุดทดลองที่คณะผู้วิจัยสร้างขึ้นมานี้ มีชื่อว่า OBEC BOX V.1 ย่อมาจาก Office of The Basic Education Commission (OBEC) หรือสำนักงานคณะกรรมการศึกษาขั้นพื้นฐาน(สพฐ.) เนื่องจากทางคณะผู้จัดทำได้รับทุนส่วนหนึ่งจากทางสำนักงานคณะกรรมการศึกษาขั้นพื้นฐาน เพื่อสร้างชุดทดลองและนำชุดทดลองที่สร้างขึ้นไปอบรมให้กับโรงเรียนที่เป็นศูนย์หุ่นยนต์ต่างๆ ซึ่งแผงวงจรไมโครคอนโทรลเลอร์นี้ถูกออกแบบให้สามารถเขียนโปรแกรมร่วมกับอุปกรณ์ทางวิทยาศาสตร์ได้ โดยแผงวงจรนี้ถูกออกแบบขึ้นมาให้มีลักษณะเฉพาะเพื่อนำไปใช้เป็นเทคโนโลยีพื้นฐานในการทำโครงงานด้านหุ่นยนต์ และโครงงานด้านวิทยาศาสตร์ต่อไป ซึ่งทางคณะผู้จัดทำได้สร้างฟังก์ชันต่างๆ โปรแกรมลงไปในไมโครคอนโทรลเลอร์ในรูปของโปรแกรมย่อยหรือ include file เพื่อให้สะดวกในการเรียกใช้งาน ทำให้น้องๆนักเรียนและผู้ที่สนใจดูโปรแกรมได้ง่ายและมีความเข้าใจมากขึ้น ลดความสับสนวุ่นวายกับโปรแกรมที่ยืดยาวและ ซับซ้อน ซึ่งมีผลทำให้เกิดความรู้สึกอยากลองเขียนโปรแกรมและเล่นกับชุดทดลองอีกด้วย ส่วนสิ่งที่จะได้คือนักเรียนและผู้ที่สนใจจะได้ฝึกการเขียนโปรแกรม และมีกระบวนการคิดอย่างเป็นระบบ โดยฟังก์ชันที่คณะผู้จัดทำสร้างขึ้นมามีดังนี้

4.1.1 การเขียนโปรแกรมหลอด LED ไฟวิ่ง 8 ดวง

การเขียนโปรแกรมควบคุมหลอด LED ที่ต่ออยู่กับพอร์ตให้สว่างหรือดับ ทำได้โดยควบคุมบิตที่ใช้งานให้มีลอจิกเป็น 0 หรือ 1

โดยพอร์ตที่ใช้เป็น output สำหรับหลอด LED มีดังนี้

P10 P11 P12 P13 P14 P15 P16 P17

4.1.2 การเขียนโปรแกรมแสดงผลตัวเลขทาง LED 7 ส่วน

สำหรับในชุดทดลองนี้ได้พัฒนาโปรแกรมย่อยสำหรับควบคุมการแสดงผลทางแผง LED ขนาด 8 หลักไว้ให้ใช้งานแล้ว โดยได้สร้างฟังก์ชันต่าง ๆ สำหรับควบคุมการแสดงผลให้ใช้งานได้ง่าย โดยรวมไว้ในไฟล์ชื่อ “SHOWNUM.h” ดังนั้นการเขียนโปรแกรมแสดงผลตัวเลขทางหลอด LED 7 เซกเมนต์เราจะต้อง include ไฟล์ “SHOWNUM.h” ขึ้นมาก่อน ซึ่งในไฟล์ประกอบไปด้วยฟังก์ชันที่สำคัญ 3 ฟังก์ชันดังนี้

- dis_num_digit(x,y);

  ฟังก์ชันนี้จะแสดงผลตัวเลขทางหลอด LED 7 เซกเมนต์ โดยที่พารามิเตอร์ x คือ หลัก(1-8)ที่ต้องการแสดงผล และพารามิเตอร์ y คือ ค่า(0-9)ที่ต้องการแสดงผล

- dis_num(x);

  ฟังก์ชัน dis_num() จะแสดงผลตัวเลข LED 7 เซกเมนต์ โดยที่ค่า x คือค่าที่ต้องการจะให้แสดงผลโดยค่าที่จะแสดงนั้นจะแสดงได้ตั้งแต่ 1- 8 หลัก และแต่ละหลักจะเป็นตัวเลข 0 – 9

- disclear();

  เป็นฟังก์ชันสำหรับเคลียร์ค่าต่างๆที่อยู่ใน DISBUF ให้มีค่าเป็น 0

4.1.3 การเขียนโปรแกรมระบบเวลาจริง

การเขียนโปรแกรมแสดงเวลาจริง เราจะต้อง include ไฟล์ “RTC.h” ซึ่งไฟล์ “RTC.h” จะประกอบไปด้วยฟังก์ชัน 2 ฟังก์ชันดังนี้

- setrtc(d,mo,y,h,mi,se)

  ฟังก์ชันนี้จะเป็นการตั้งค่าวันเวลา ในการตั้งค่านั้นเราจะใส่เลขฐาน16 ให้กับพารามิเตอร์แต่ละตัวโดยพารามิเตอร์แต่ละตัวมีค่า ดังนี้

d	คือ	วันที่ ตัวอย่างเช่น 0x10 คือวันที่ 10
mo	คือ	เดือน ตัวอย่างเช่น 0x02 คือ เดือนที่2(เดือนกุมภาพันธ์)
y	คือ	ปี เช่น 0x10 คือ ปี 2010
h	คือ	ชั่วโมง เช่น 0x11 คือ 11 นาฬิกา
mi	คือ	นาที เช่น 0x55 คือ 55 นาที
se	คือ	วินาที เช่น 0x59 คือ 59 วินาที

หลังจากการตั้งค่านี้ไฟล์ rtc.h จะมีฟังก์ชันที่ทำการเขียนข้อมูลลงบนชิป DS1307 ทำให้เวลาเดินตลอดเวลาเนื่องจากใช้พลังงานจากแบตเตอร์รี่ และเราสามารถเรียกวันเวลาปัจจุบันดูได้ตามที่ต้องการ

ตัวอย่างการใช้งานฟังก์ชัน setrtc();

setrtc(0x12,0x02,0x10,0x12,0x26,0x50); //คือ วันที่ 12 เดือน 2 ปี 2010
เวลา 12:26:50

- dis_rtc()

ฟังก์ชันนี้เป็นการแสดงผลนาฬิกาเวลาปัจจุบัน เราสามารถเรียกใช้ฟังก์ชันนี้ได้เลย โดยการแสดงค่าของเวลานั้นจะทำการอ่านข้อมูลจากชิป DS 1307 แล้วแสดงผลออกทาง LED 7 เซกเมนต์ ตัวอย่างการใช้งานฟังก์ชัน dis_rtc()

dis_rtc(); //เรียกใช้ฟังก์ชันเพื่อแสดงเวลาขณะนั้นทาง หลอด LED 7เซกเมนต์

- rt_rtc(m)

คำสั่ง rt_rtc(m) เป็นคำสั่งคืนค่าของเวลามาเป็นเลขฐานสิบหก ซึ่งการขอคืนค่าจะต้องประกาศตัวแปร unsigned char sec = 0,min = 1,hour = 2,day = 4,month = 5,year = 6; พารามิเตอร์ที่ใช้ในการขอคืนค่าแต่ละตัวจะให้ค่าที่คืนกลับมาต่างกัน ดังนี้

rt_rtc(sec)	คืนค่าของวินาที
rt_rtc(min)	คืนค่าของนาที
rt_rtc(hour)	คืนค่าของชั่วโมง
rt_rtc(day)	คืนค่าของวัน
rt_rtc(month)	คืนค่าของเดือน
rt_rtc(year)	คืนค่าของปี

ส่วนมากเราจะคืนค่าของเวลามาใช้ในการเขียนโปรแกรมกำหนดการทำงานของชุดทดลอง เช่น กำหนด รีเลย์ เปิด-ปิดไฟ ตั้งเป็นนาฬิกาปลุก ตัวอย่างเช่น

```
if (rt_rtc(min) = = 0x00)//min เป็นการคืนค่านาทีดังนั้นเมื่อนาทีเป็น 00 จะมีเสียง
  beep();
```

4.1.4 การเขียนโปรแกรมวัดอุณหภูมิ

การเขียนโปรแกรมที่เกี่ยวข้องกับตัววัดอุณหภูมิจะต้องทำการ include ไฟล์ “TEMP.h”ซึ่งไฟล์“TEMP.h” จะประกอบด้วยฟังก์ชันดังนี้

- dis_temp()

  ฟังก์ชันนี้เป็นฟังก์ชันแสดงอุณหภูมิออกมาทางหลอด LED 7 เซกเมนต์ ซึ่งอุณหภูมิที่ได้จะมีหน่วยเป็นองศาเซลเซียสและมีความละเอียดเพียง .5 เช่น 20, 20.5

- rt_temp();

  คำสั่ง rt_temp();เป็นคำสั่งคืนค่าของอุณหภูมิมาเป็นข้อมูลประเภท char ซึ่งค่าของอุณหภูมิที่ได้ที่ได้มักจะนำมาใช้ในการเขียนโปรแกรมกำหนดการทำงานของชุดทดลอง เช่น เมื่ออุณหภูมิเกิน 25 องศาเซลเซียสให้มีเสียงเตือน

4.1.5 ฟังก์ชันหน่วงเวลา

การเรียกใช้ฟังก์ชันหน่วงเวลา เราจะต้อง include ไฟล์ dmsec.h ซึ่งไฟล์จะมีฟังก์ชันให้ใช้งาน คือ

- dmsec(milisec)

  ฟังก์ชัน dmsec(milisec) เป็นฟังก์ชันใช้สำหรับหน่วงเวลา โดยพารามิเตอร์ที่ใส่จะเป็นเลขฐานสิบดังนั้นหากใส่เลข 1000 จะมีการหน่วงเวลา 1 วินาที

4.2 ผลการทดลอง

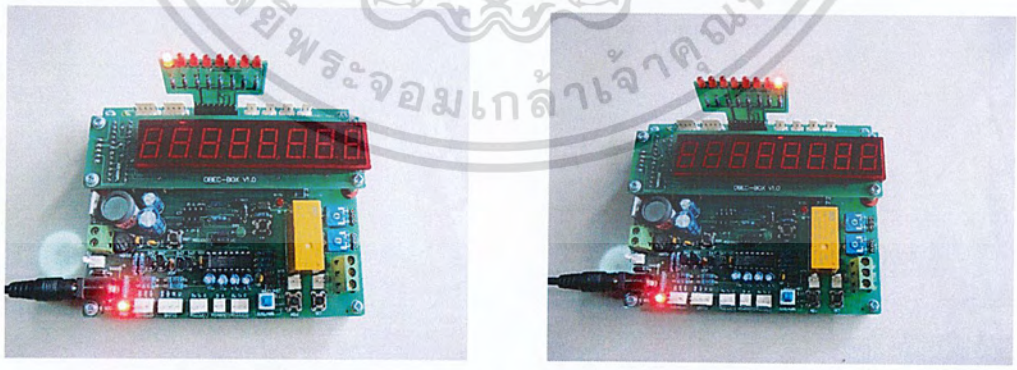
ในการเขียนโปรแกรมที่ใช้กับชุดทดลองนี้จะสามารถเรียกใช้ฟังก์ชันต่างๆที่กล่าวมาแล้วข้างต้นได้ ซึ่งสะดวกและง่ายสำหรับการใช้งานมาก แต่ต้องทำการ include ไฟล์นั้นเข้ามาก่อน

ตัวอย่างโปรแกรมหลอดไฟ LED วิ่งทีละดวง

```
#include <reg52.h>
#include <dmsec.h>

sbit    P10    = P1^0;  // 8B Port
```

```
sbit    P11    = P1^1;
sbit    P12     = P1^2;
sbit    P13     = P1^3;
sbit    P14    = P1^4;
sbit    P15    = P1^5;
sbit    P16    = P1^6;
sbit    P17    = P1^7;

unsigned char bdata    BITM;
sbit              B0 = BITM^0;
sbit              B1 = BITM^1;
sbit              B2 = BITM^2;
sbit              B3 = BITM^3;
sbit              B4 = BITM^4;
sbit              B5 = BITM^5;
sbit              B6 = BITM^6;
sbit              B7 = BITM^7;

void test8bsub(){
            P10 = B0;
            P11 = B1;
            P12 = B2;
            P13 = B3;
            P14 = B4;
            P15 = B5;
            P16 = B6;
            P17 = B7;
}

void dis_8led (void){
            unsigned char i,k;
             for (i = 1;i<=3;i++){
```

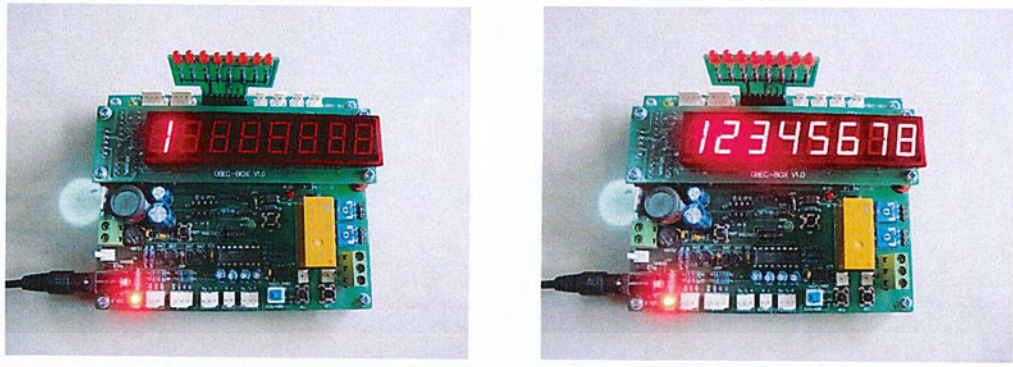
```
                BITM = 0xfe;
                for(k=1;k<=8;k++){
                    test8bsub();
                    dmsec(200);
            BITM = (BITM <<1) |0x01;
        }
    }
            BITM = 0xff;
            test8bsub();
}

void start (void) {              // start process
   dmsec (250);
}

void main (void) {
   start();
   dis_8led();
}
```



รูปที่ 4.1 แสดง LED ติดทีละหลอดจากหลอดที่ 1 ไปถึงหลอดที่ 8

ตัวอย่างโปรแกรมแสดงผลตัวเลข 1-8 ตั้งแต่หลักแรกจนถึงหลักสุดท้าย

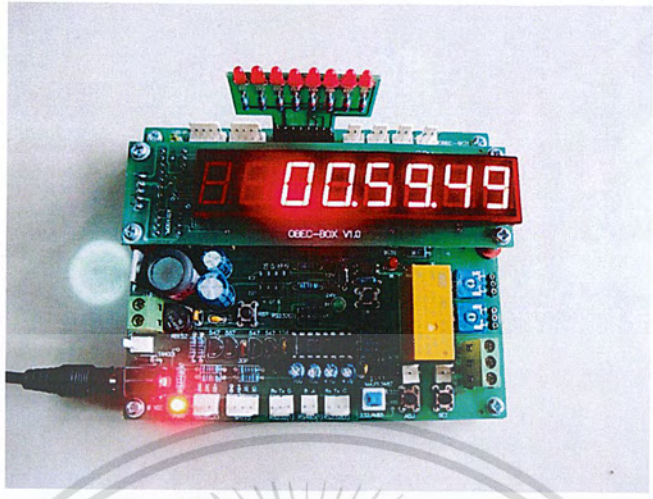
```
#include <reg52.h>
#include <shownum.h>                    //include "shownum.h" เข้ามา
#include <dmsec.h>
sbit    BUZZER   =   P2^3;   // buzzer

void beep (void) {BUZZER = 0; dmsec (250); BUZZER = 1;}    // ส่งเสียง beep สั้น ๆ ออกทาง
ลำโพง

void test_disdigit(void){
        unsigned char i;
        for(i=1;i<=8;i++){
                dis_num_digit(i,i);
                dmsec(1000);
        }
}
void start (void) {             // start process
       dmsec (250);
         beep ();
}
void main (void) {
     start();
    disclear();
     test_disdigit();
}
```



รูปที่ 4.2 แสดงตัวเลขทีละหลักตั้งแต่หลักที่ 1 ถึงหลักที่ 8 แสดงตามค่าของหลัก

ตัวอย่างโปรแกรมการตั้งค่าของระบบเวลาจริงแล้วแสดงผล

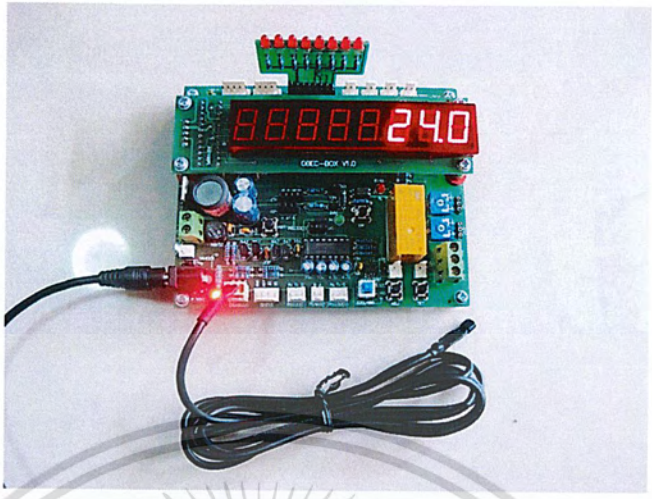
```
#include <reg52.h>
#include <intrins.h>
#include <shownum.h>                    // display number
#include <rtc.h>                        // real time clock
#include <dmsec.h>

void start (void) {             // start process
   dmsec (250);
   disclear();
}

void main (void) {
   start();
            setrtc(0x02,0x02,0x10,0x00,0x58,0x59);
            while(1)
            dis_rtc();

}
```



รูปที่ 4.3 แสดงผลการตั้งเวลาและเวลาจะเดินไปเรื่อยๆ

ตัวอย่างโปรแกรมแสดงผลตัววัดอุณหภูมิ

```
#include <reg52.h>
#include <shownum.h>   // display number
#include <temp.h>        // temp  sensor
#include <dmsec.h>



void start (void) {                  // start process
   dmsec (250);
   disclear();
}

void main (void) {
   start();
  while(1)
   dis_temp();
}
```



รูปที่ 4.4 แสดงค่าของอุณหภูมิที่ตัวตรวจวัดได้

จะเห็นได้ว่าการเขียนโปรแกรมมีความยาวที่สั้นลงและง่ายต่อการทำความเข้าใจ เนื่องจากโปรแกรมย่อยที่เกี่ยวกับการแสดงผลได้ถูกซ่อนไว้ใน include ไฟล์ “SHOWNUM.h” เรียบร้อยแล้ว จึงทำให้ส่วนโปรแกรมหลักๆมีขนาดที่สั้น ทำให้ผู้ที่ศึกษาเข้าใจโปรแกรมได้ง่าย

บทที่ 5
สรุปผลการดำเนินงาน และข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

การดำเนินงานนี้มีจุดมุ่งหมายเพื่อที่จะพัฒนาให้เด็กนักเรียนและผู้ที่สนใจได้ฝึกการเขียนโปรแกรม โดยใช้แผงวงจรไมโครคอนโทรลเลอร์หรือชุดทดลองที่สร้างขึ้นนี้โดยให้แสดงผลออกมาทางอุปกรณ์แสดงผลต่างๆ และฝึกทักษะกระบวนการคิดอย่างมีระบบมากขึ้น

เนื่องจากการเขียนโปรแกรมและให้ผลลัพธ์แสดงออกมาทางอุปกรณ์ต่างๆทางวิทยาศาสตร์เหล่านี้ จะทำให้มีความน่าสนใจ เกิดความสนุกสนานมากกว่าการแสดงผลลัพธ์ออกทางหน้าจอคอมพิวเตอร์ธรรมดา ซึ่งการดำเนินงานในลักษณะนี้ จะทำให้เด็กพยายามที่จะคิดและเขียนโปรแกรมเพื่อที่จะให้ได้ผลลัพธ์ออกมาตามที่ตนต้องการ โดยอุปกรณ์ในการแสดงผลที่ใช้มีด้วยกันหลายอย่าง เช่น หลอด LED LED 7-segment ระบบฐานเวลาจริง และตัววัดอุณหภูมิ เป็นต้น

แต่ในความจริงแล้วการที่จะเขียนโปรแกรมให้ไมโครคอนโทรลเลอร์แสดงผลออกมาตามที่ต้องการได้นั้น จะต้องมีทักษะการเขียนโปรแกรมพอสมควร และเนื่องจากลักษณะโปรแกรมที่ได้จะมีความซับซ้อนและความยาวที่ค่อนข้างมาก ซึ่งอาจจะทำให้ผู้เรียนเกิดความเบื่อหน่ายและสับสน ซึ่งถือว่าเป็นอุปสรรคสำคัญในการเรียนรู้การเขียนโปรแกรมเบื้องต้น ทั้งนี้ทางคณะผู้จัดทำได้เล็งเห็นถึงความสำคัญของปัญหาจึงได้มีการรวบรวมการทำงานที่สำคัญเป็น include ไฟล์ไว้อย่างเป็นหมวดหมู่และทำฟังก์ชันต่างๆขึ้นมาให้สามารถเรียกใช้ได้ง่ายมากขึ้น

โดยที่ทางคณะผู้จัดทำได้นำชุดทดลองไปทดลองใช้จริงกับโรงเรียนที่เป็นศูนย์หุ่นยนต์ของภูมิภาคนั้นๆมาแล้วด้วยกัน 2 ศูนย์ คือ ที่โรงเรียนกำแพงเพชรพิทยาคม และโรงเรียนเมืองตลุงพิทยาสรรพ์

โรงเรียนกำแพงเพชรพิทยาคม นักเรียนที่เข้าร่วมการอบรมครั้งนี้ส่วนใหญ่จะเป็นนักเรียนระดับมัธยมศึกษาชั้นปีที่ 4 ซึ่งส่วนใหญ่เคยได้เรียนการเขียนโปรแกรมเบื้องต้นมาก่อนแล้ว จึงสามารถที่จะเรียนรู้ได้ค่อนข้างเร็ว และมักจะทำการดัดแปลงโปรแกรมให้มีการแสดงผลที่หลากหลายและซับซ้อนมากขึ้น

ส่วนโรงเรียนเมืองตลุงพิทยาสรรพ์ นักเรียนที่เข้าร่วมการอบรมครั้งนี้ส่วนใหญ่จะเป็นนักเรียนระดับมัธยมศึกษาชั้นปีที่ 2 และ 4 ซึ่งน้องๆนักเรียนส่วนใหญ่ยังไม่เคยได้เรียนการเขียนโปรแกรมมาก่อน แต่จะเห็นได้ว่าน้องๆนักเรียนทุกคนมีความสนใจ มีความตั้งใจในการอบรมมาก และดูเหมือนว่าน้องๆจะตื่นเต้นและสนุกสนานกับการเขียนโปรแกรมสั่งการให้ตัวไมโครคอนโทรเลอร์แสดงผลออกมาได้ รวมถึงเมื่อเกิดข้อสงสัยใดๆน้องๆก็จะซักถามอยู่เสมอ

5.2 ข้อเสนอแนะ

การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ด้วยภาษา C สำหรับชุดทดลองนี้ สามารถที่จะเขียนโปรแกรม สร้างฟังก์ชันให้แสดงผลผ่านอุปกรณ์ต่างๆได้อีกมากมาย ซึ่งสามารถที่จะพัฒนาต่อไปอีกได้ ไม่ว่าจะเป็นการเขียนโปรแกรมควบคุมมอเตอร์ รวมถึงตัวเซ็นเซอร์ต่างๆ ซึ่งแผงวงจรไมโครคอนโทรลเลอร์หรือชุดทดลองรวมถึงฟังก์ชันต่างๆที่สร้างขึ้นมานี้ จึงน่าจะเป็นประโยชน์แก่ผู้ที่สนใจในการทำโครงงานหุ่นยนต์หรือโครงงานวิทยาศาสตร์ต่อไป

เอกสารอ้างอิง

[1] ผศ.ธีรวัฒน์ ประกอบผล. 2545. การพัฒนาไมโครคอนโทรลเลอร์ด้วยภาษาซี. กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี(ไทย-ญี่ปุ่น).

[2] ผศ.ธีรวัฒน์ ประกอบผล. 2537. การประยุกต์ใช้งานไมโครคอนโทรลเลอร์. กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี(ไทย-ญี่ปุ่น).

[3] ผศ.ธีรวัฒน์ ประกอบผล. 2553. คู่มือการเขียนโปรแกรมภาษา C. กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี(ไทย-ญี่ปุ่น).

[4] ผศ.ธีรวัฒน์ ประกอบผล. 2546. ภาษาแอสเซมบลี MCS-51. กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี(ไทย-ญี่ปุ่น).

[5] Padmanabhan T.R. 2007. Introduction to Microcontrollers and their Applications. U.K. : Alpha Science International Ltd. Oxford.

[6] Richard A. Cox. 1995. Technician's Guide to Programmable Controllers. 3rd Edition. USA. : Delmar Publisher.

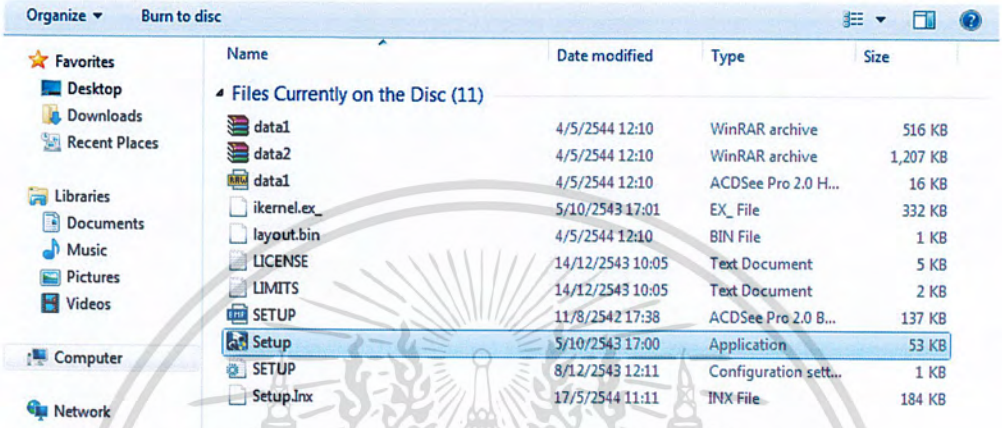
[7] Martin Bates. 2000. PIC Microcontrollers an Introduction to Microelectronics. Burlington : Linacre House. Jordan Hill.

ภาคผนวก ก.

การติดตั้งโปรแกรม Keil ISP-516 และ ขั้นตอนการใช้งานโปรแกรม

ก.1 การติดตั้งโปรแกรม Keil

เปิดแผ่นโปรแกรมขึ้นมาแล้วเข้าไฟล์ “Setup”

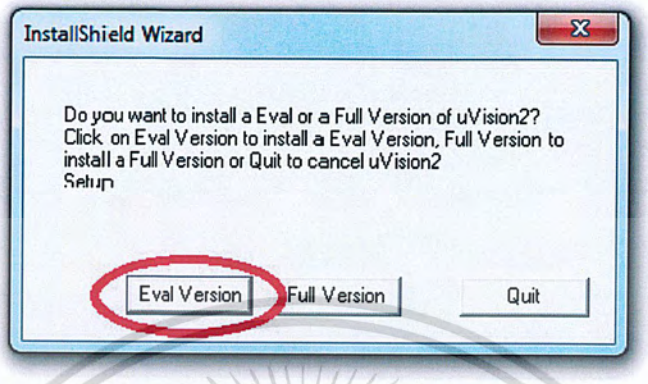


รูปที่ ก.1.1 หน้าจอแสดงการเลือกไฟล์ setup

หลังจากเข้าไฟล์ โปรแกรมจะโหลดขึ้นมาดังรูป

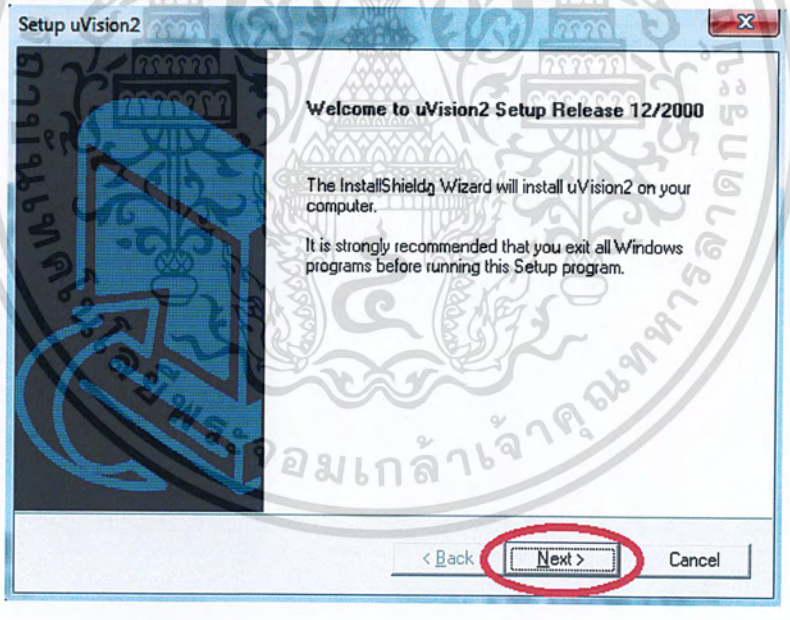
รูปที่ ก.1.2 หน้าจอแสดงการโหลดโปรแกรม

เลือกลงโปรแกรมแบบ “Eval Version”



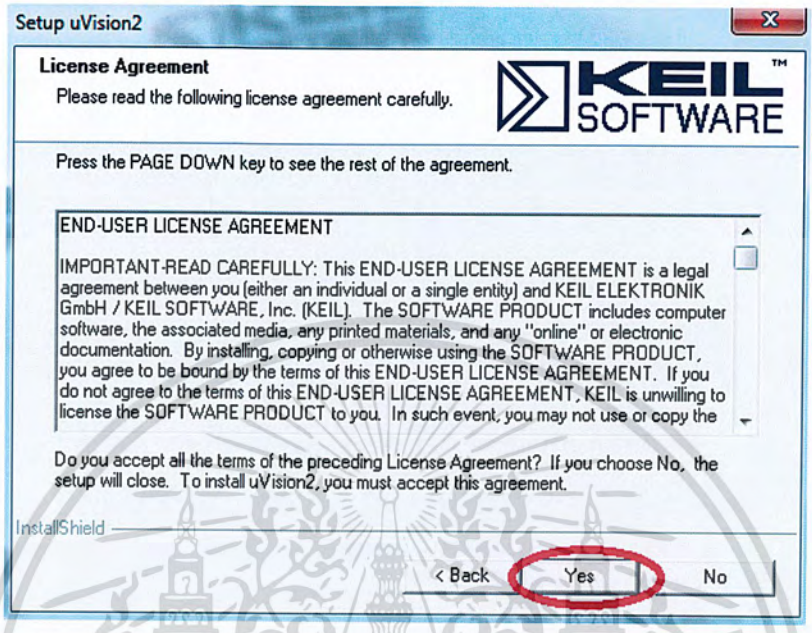
รูปที่ ก.1.3 หน้าจอแสดงการเลือกลงโปรแกรม

หลังจากเลือกแบบEval Version แล้วกด “Next”

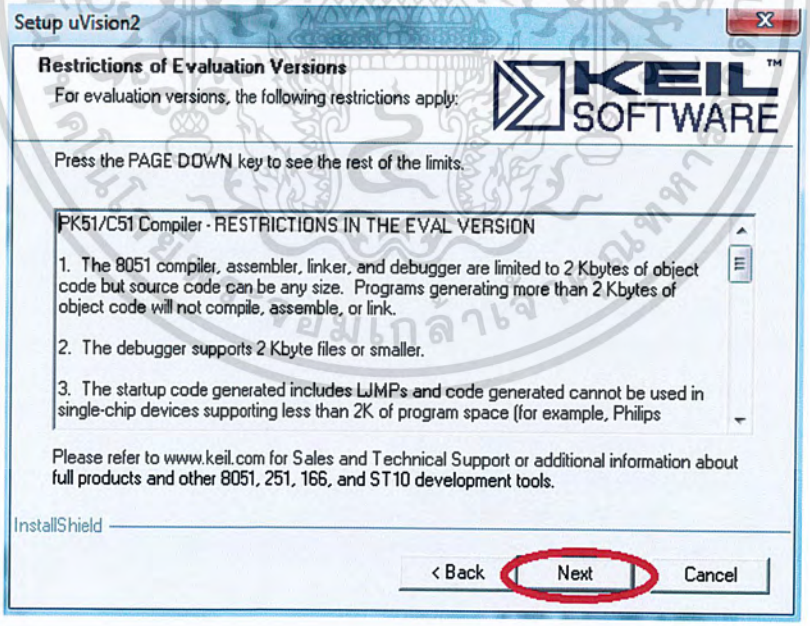


รูปที่ ก.1.4 หน้าจอแสดงการเริ่มต้นการติดตั้งโปรแกรม

ทำการยอมรับข้อตกลงของโปรแกรมโดยการกด “Yes”

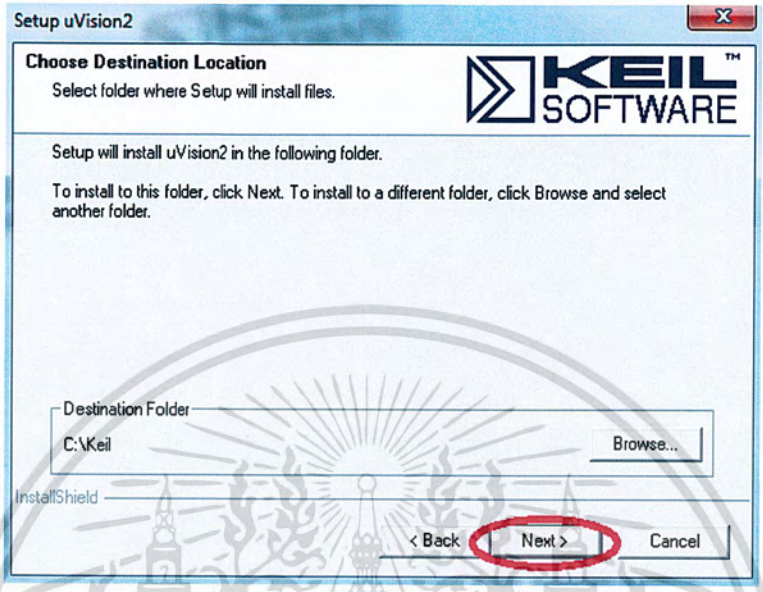


รูปที่ ก.1.5 หน้าจอแสดงข้อตกลงเกี่ยวกับโปรแกรม



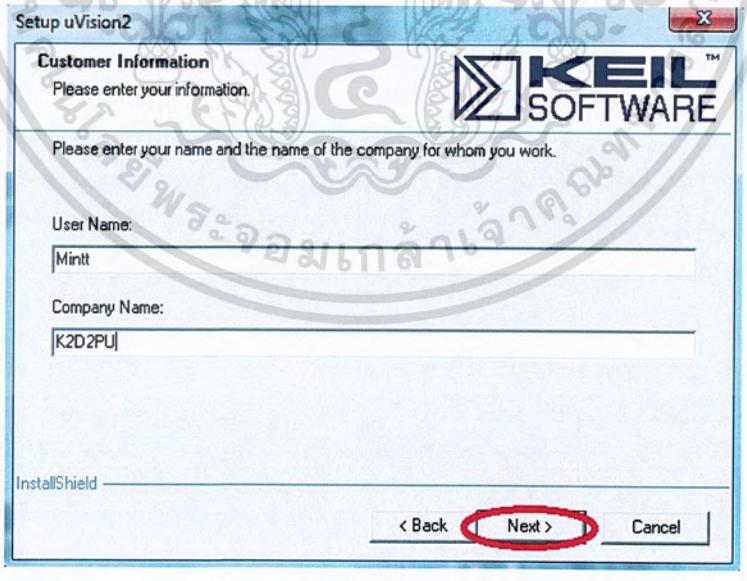
รูปที่ ก.1.6 หน้าจอแสดงข้อจำกัดเกี่ยวกับโปรแกรม

เลือกตำแหน่งของหน่วยความจำที่จะลงโปรแกรมแล้วกด"Next"



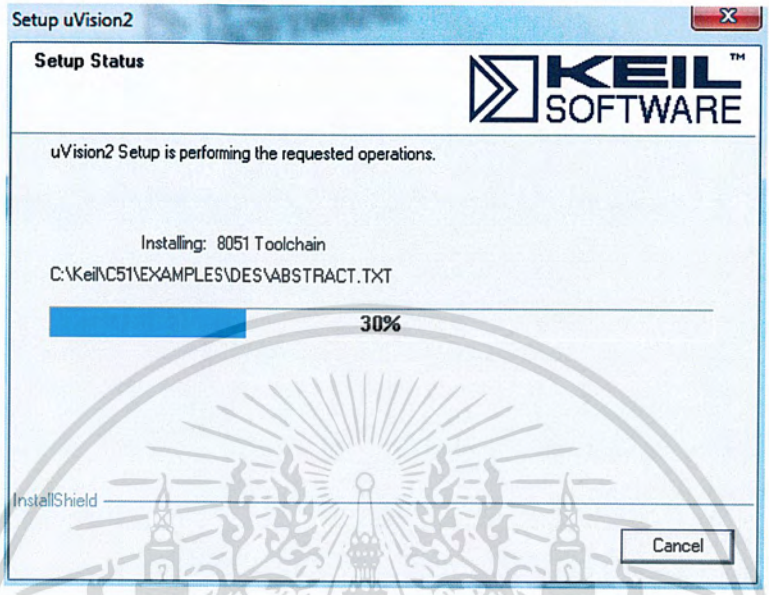
รูปที่ ก.1.7 หน้าจอแสดงการเลือกโฟลเดอร์สำหรับการติดตั้งโปรแกรม

ตั้งชื่อของผู้ใช้ ใส่รหัส แล้วกด "Next"



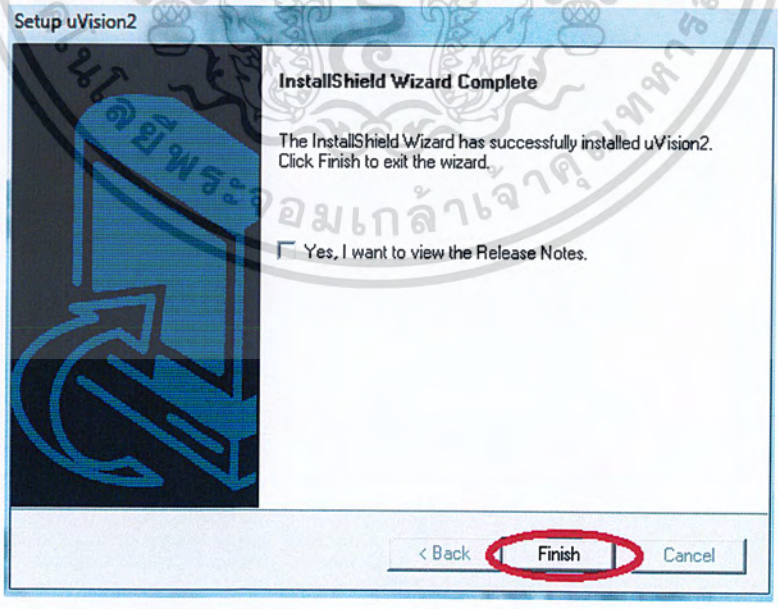
รูปที่ ก.1.8 หน้าจอแสดงการตั้งชื่อของผู้ใช้

โปรแกรมจะทำการติดตั้งดังรูป



รูปที่ ก.1.9 หน้าจอแสดงการติดตั้งโปรแกรม

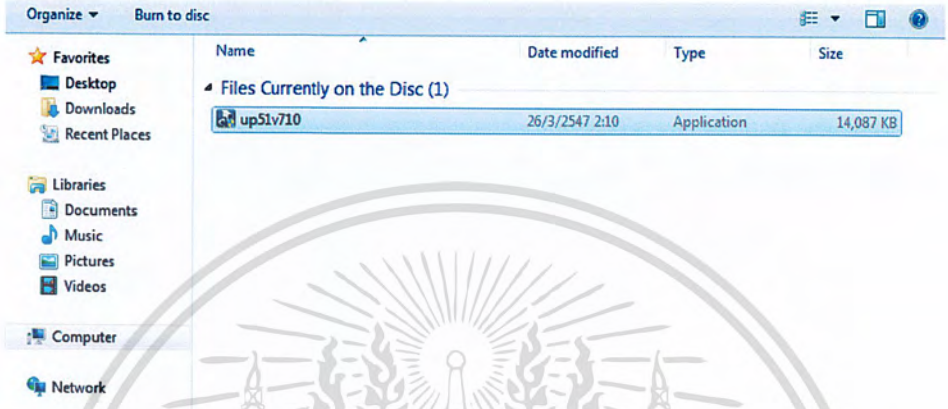
หลังจากโปรแกรมติดตั้งเสร็จแล้ว กด "Finish" เป็นอันเสร็จสิ้นการติดตั้งโปรแกรม



รูปที่ ก.1.10 หน้าจอแสดงการติดตั้งโปรแกรมเสร็จสิ้น

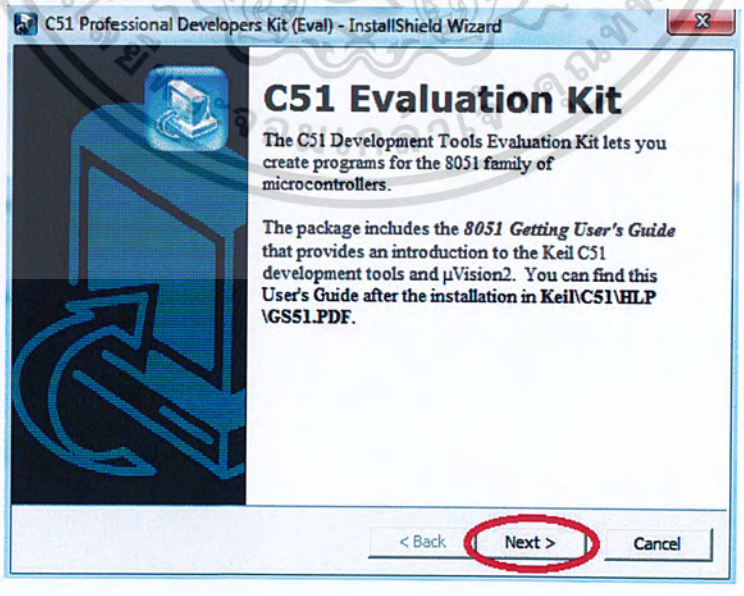
ก.2 การติดตั้งโปรแกรม Keil update

เปิดแผ่นโปรแกรมขึ้นมาแล้วเข้าไปยังโฟลเดอร์ Keil PK51 7.10 update
ดับเบิลคลิก เพื่อทำการติดตั้ง



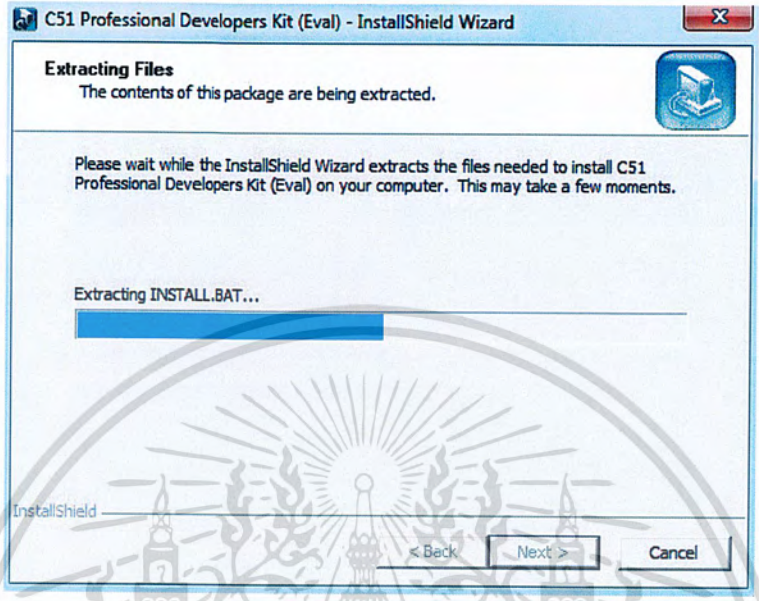
รูปที่ ก.2.1 หน้าจอแสดงไฟล์ setup

หลังจากนั้นกด"Next"เพื่อติดตั้งโปรแกรม



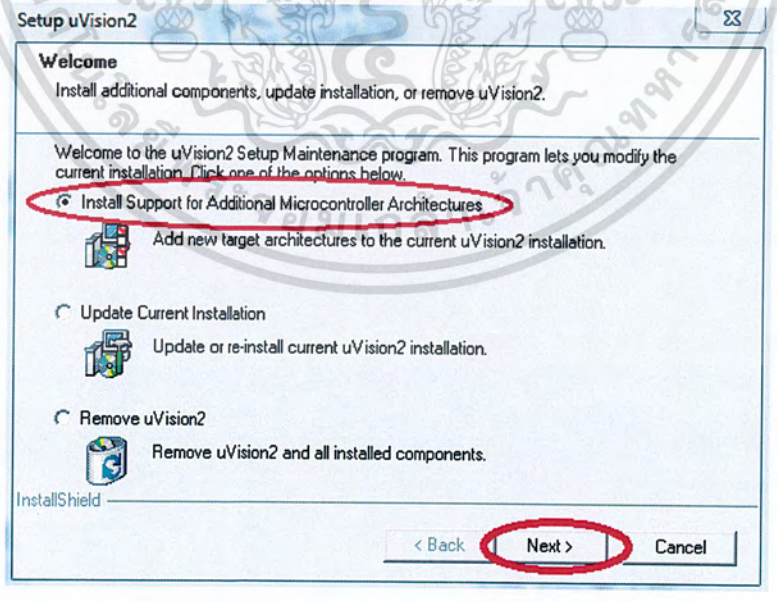
รูปที่ ก.2.2 หน้าจอแสดงรายละเอียดโปรแกรม

โปรแกรมจะทำการดึงไฟล์ออกมาเพื่อติดตั้ง



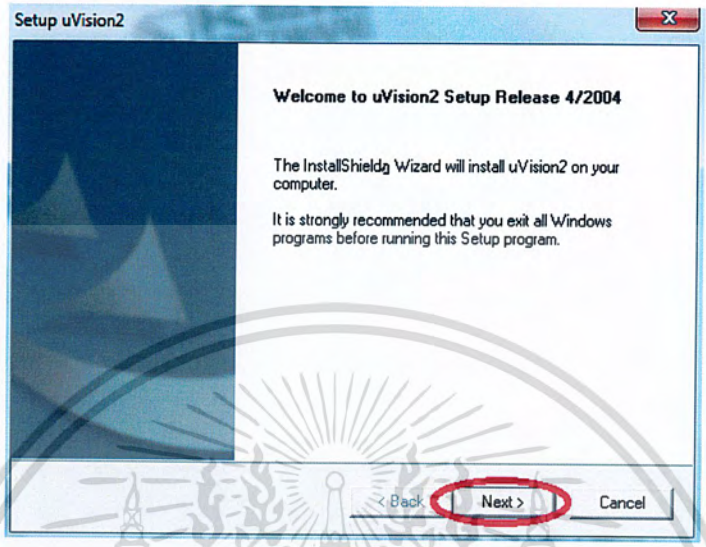
รูปที่ ก.2.3 หน้าจอแสดงการดึงไฟล์

เลือก Install แล้วกด “Next”



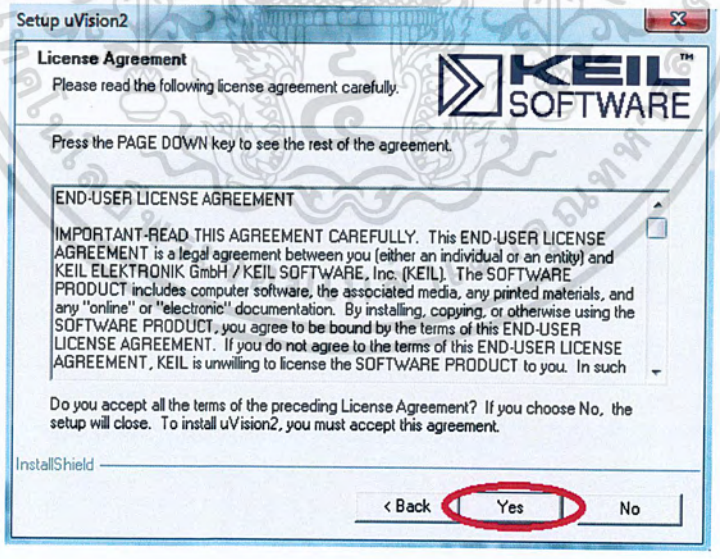
รูปที่ ก.2.4 หน้าจอแสดงการเลือกการติดตั้งโปรแกรม

จะมีข้อความขึ้นมาแนะนำโปรแกรมหลังจากนั้นกด “Next”



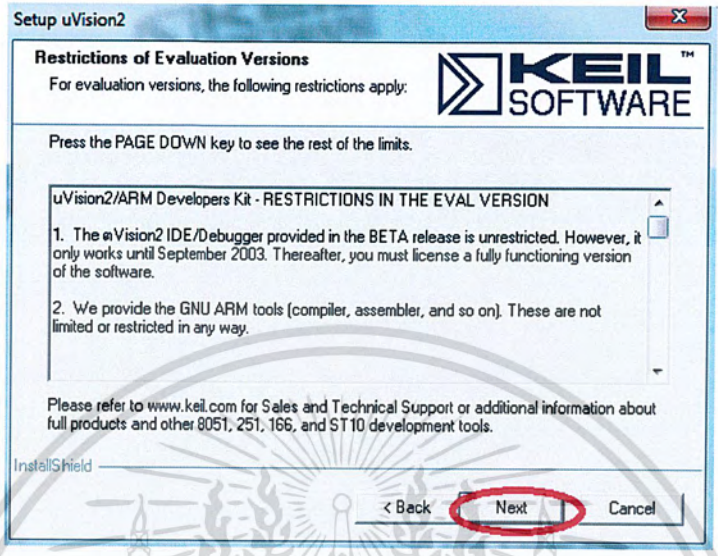
รูปที่ ก.2.5 หน้าจอแสดงการเริ่มต้นการติดตั้งโปรแกรม

ทำการยอมรับข้อตกลงของโปรแกรมโดยการกด “Yes”



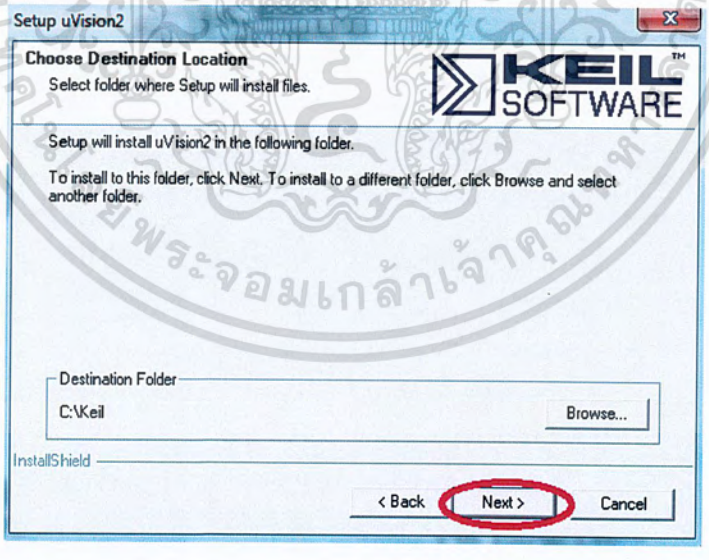
รูปที่ ก.2.6 หน้าจอแสดงข้อตกลงเกี่ยวกับโปรแกรม

กด "Next" เพื่อทำการติดตั้งโปรแกรม



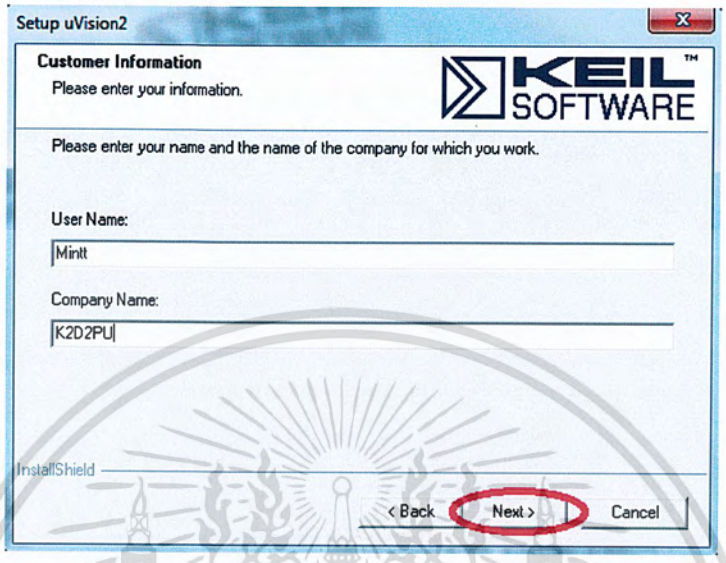
รูปที่ ก.2.7 หน้าจอแสดงข้อจำกัดเกี่ยวกับโปรแกรม

เลือกตำแหน่งของหน่วยความจำที่จะลงโปรแกรมแล้วกด"Next"



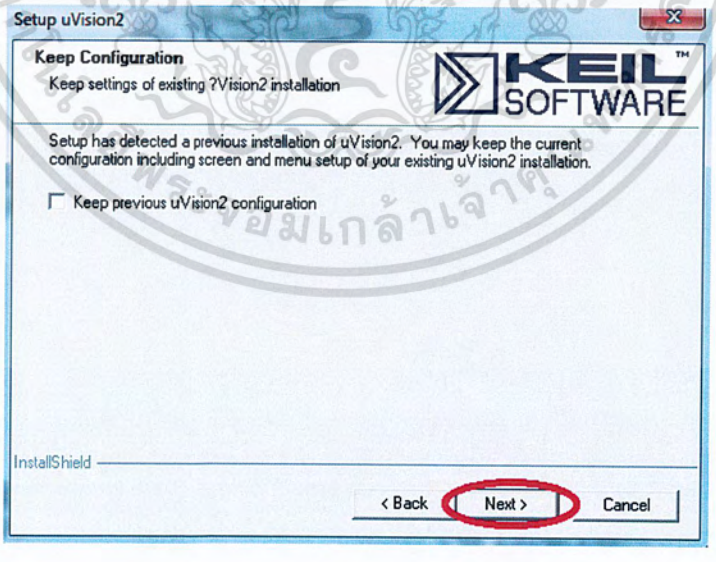
รูปที่ ก.2.8 หน้าจอแสดงการเลือกโฟลเดอร์สำหรับการติดตั้งโปรแกรม

ตั้งชื่อของผู้ใช้ ใส่รหัส แล้วกด “Next”



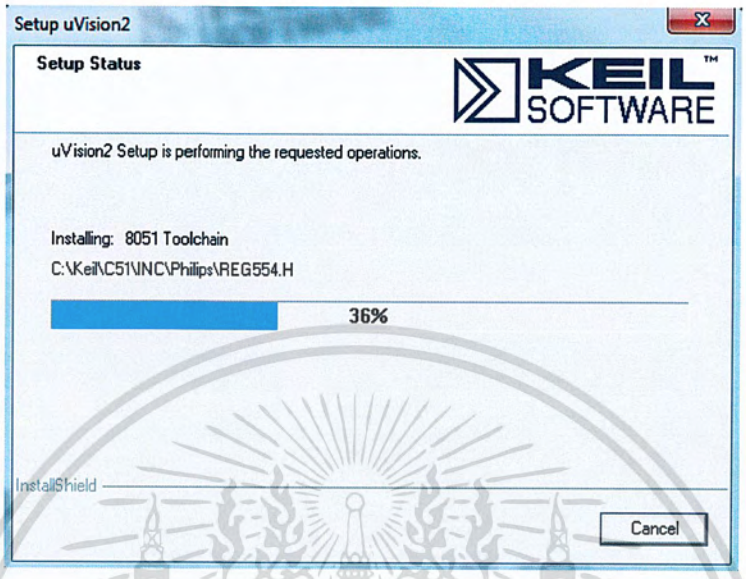
รูปที่ ก.2.9 หน้าจอแสดงการตั้งชื่อของผู้ใช้

โปรแกรมจะถามว่าจะเก็บการตั้งค่าที่เคยตั้งมาก่อนหรือไม่ ถ้าจะเก็บของ version เดิมไว้ให้เลือกเครื่องหมายถูกแล้วกด “Next”



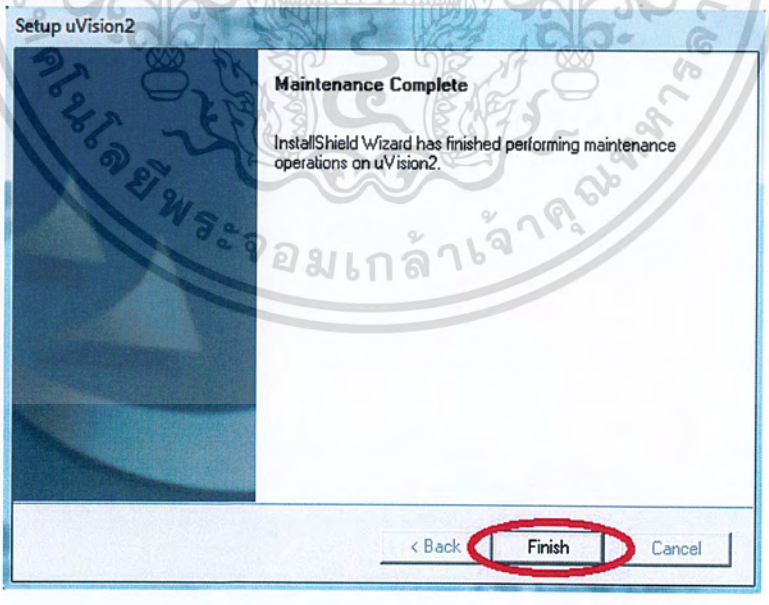
รูปที่ ก.2.10 หน้าจอแสดงการเลือกเก็บการตั้งค่าของโปรแกรมเวอร์ชันเดิม

หลังจากนั้นโปรแกรมจะทำการติดตั้งดังรูป



รูปที่ ก.2.11 หน้าจอแสดงการติดตั้งโปรแกรม

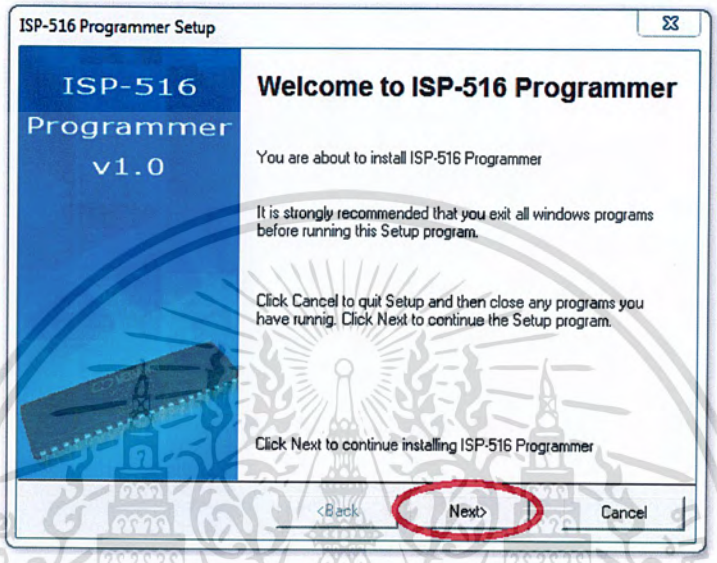
หลังจากโปรแกรมติดตั้งเสร็จแล้ว กด "Finish"เป็นอันเสร็จสิ้นการติดตั้งโปรแกรม



รูปที่ ก.2.12 หน้าจอแสดงการติดตั้งโปรแกรมเสร็จสิ้น

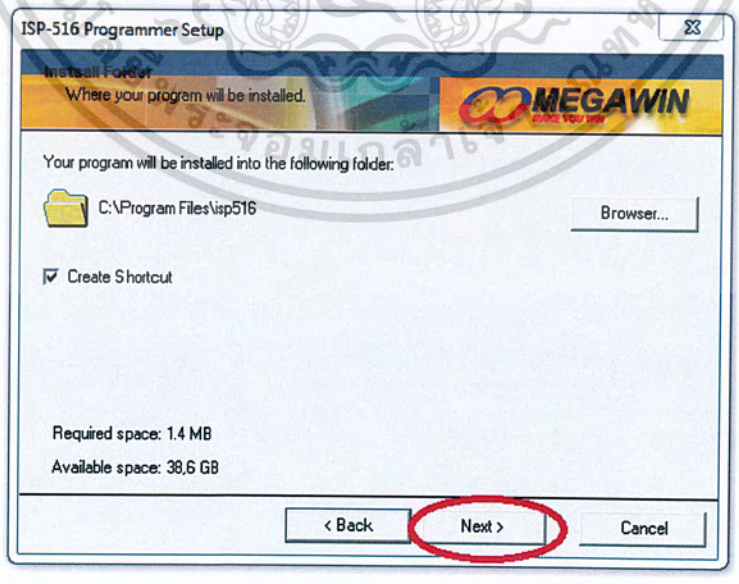
ก.3 ลงโปรแกรม ISP-516

เข้าไฟล์ติดตั้งจะขึ้นมาดังรูปแล้วทำการกด “Next” เพื่อติดตั้งโปรแกรม



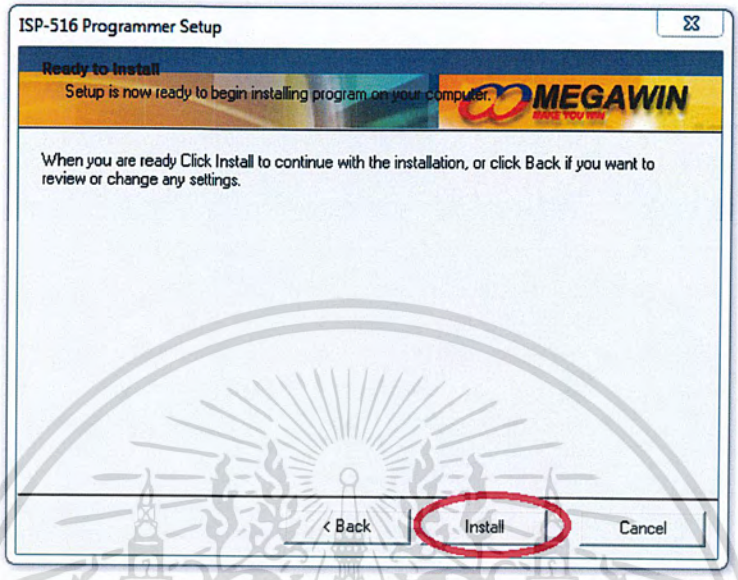
รูปที่ ก.3.1 หน้าจอแสดงการเริ่มต้นการติดตั้งโปรแกรม

หลังจากนั้นเลือกตำแหน่งที่จะติดตั้งโปรแกรมแล้วกด “Next”



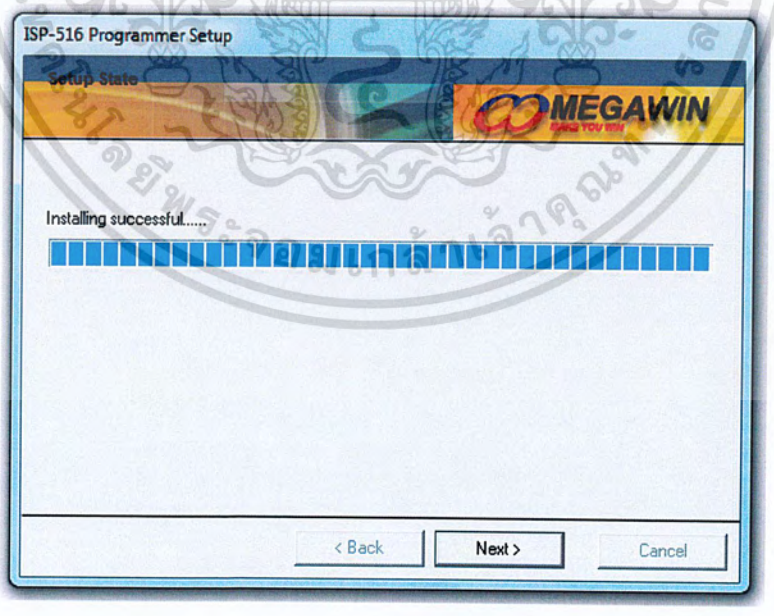
รูปที่ ก.3.2 หน้าจอแสดงการเลือกโฟลเดอร์สำหรับการติดตั้งโปรแกรม

กด Install เพื่อทำการติดตั้งโปรแกรม



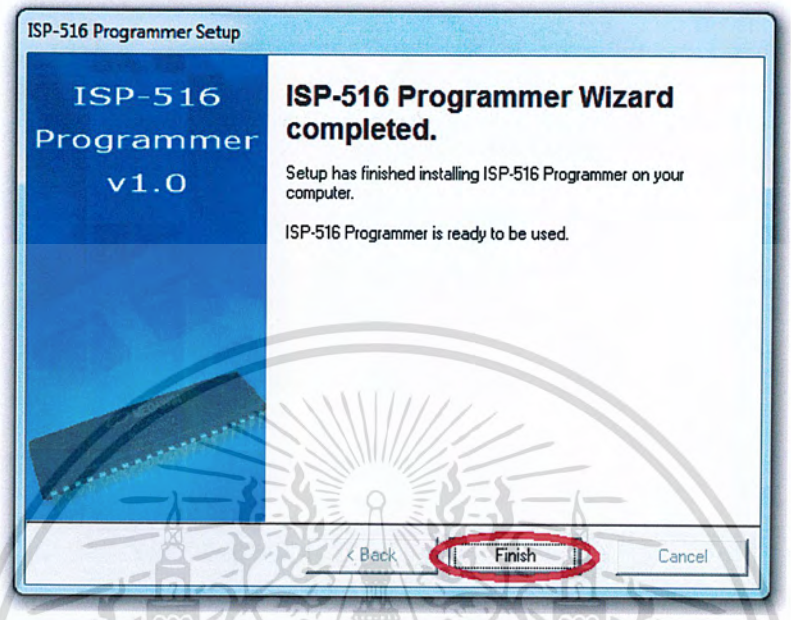
รูปที่ ก.3.3 หน้าจอแสดงการเตรียมพร้อมสำหรับติดตั้งโปรแกรม

โปรแกรมจะทำการติดตั้งดังรูป



รูปที่ ก.3.4 หน้าจอแสดงการติดตั้งโปรแกรม

โปรแกรมติดตั้งสร็จสิ้นและพร้อมที่จะใช้งานให้กด “Finish”

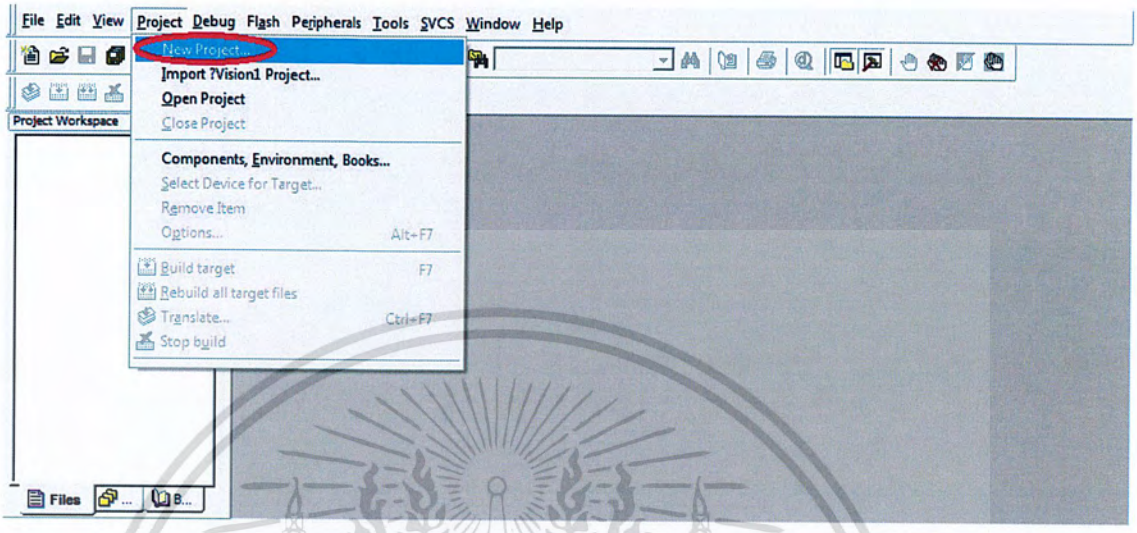


รูปที่ ก.3.5 หน้าจอแสดงการติดตั้งโปรแกรมเสร็จสิ้น

ก.4 ขั้นตอนการเขียนโปรแกรมภาษา C โดยใช้ Keil μVision 2

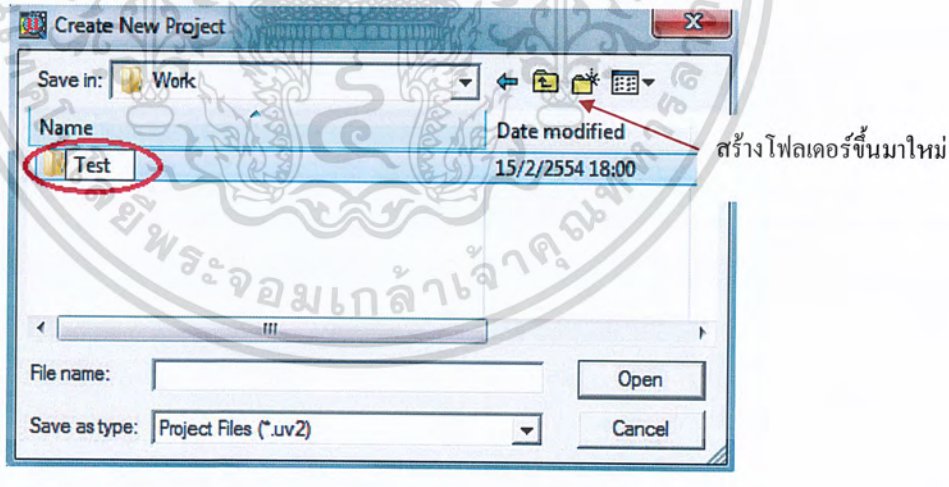
รูปที่ ก.4.1 แสดงโปรแกรม Keil μVision 2

สร้าง Project โดย New Project เพื่อสร้างโปรเจคใหม่ขึ้นมา



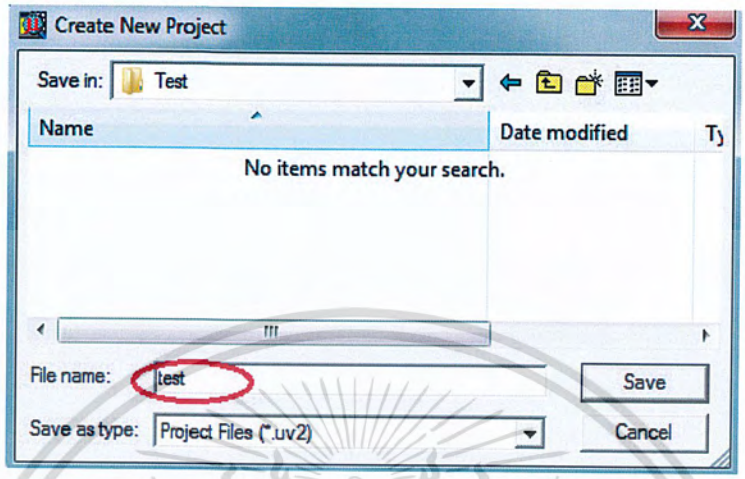
รูปที่ ก.4.2 หน้าจอแสดงการสร้าง Project

ทำการสร้างโฟลเดอร์ขึ้นมาใหม่ ตั้งชื่อโฟลเดอร์ แล้วคลิกเข้าไปในโฟลเดอร์นั้น



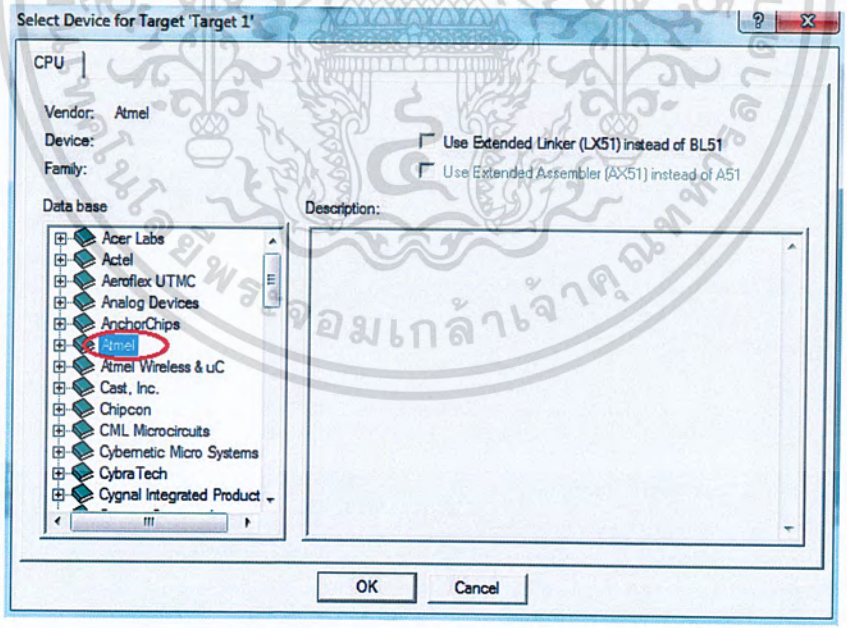
รูปที่ ก.4.3 หน้าจอแสดงการเลือกโฟลเดอร์ที่จะทำการสร้าง Project

จากนั้น ตั้งชื่อไฟล์ ซึ่งควรจะสอดคล้องกับชื่อของโปรเจค

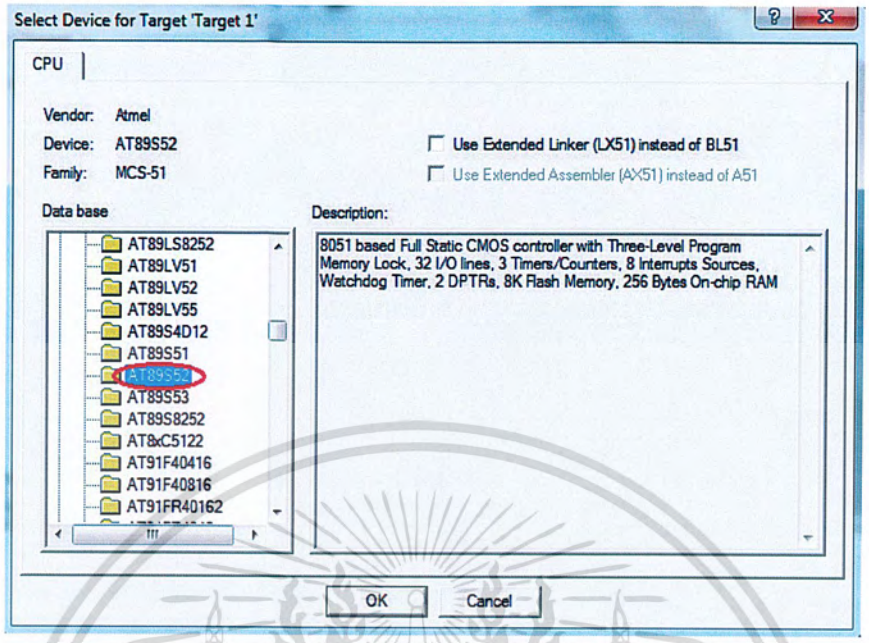


รูปที่ ก.4.4 หน้าจอแสดงการตั้งชื่อ Project

เลือกเบอร์ชิปที่เราจะใช้ โดยไปที่ Atmel แล้วเลือก AT89S52 กด OK

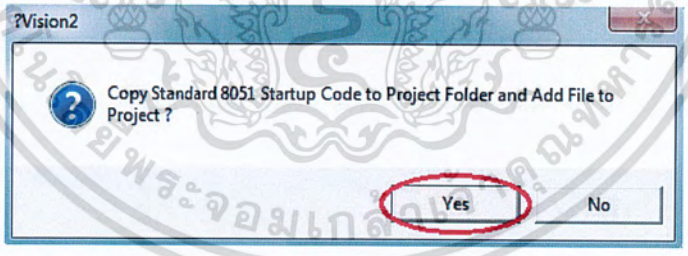


รูปที่ ก.4.5 หน้าจอแสดงการเลือกตระกูลชิปที่ใช้



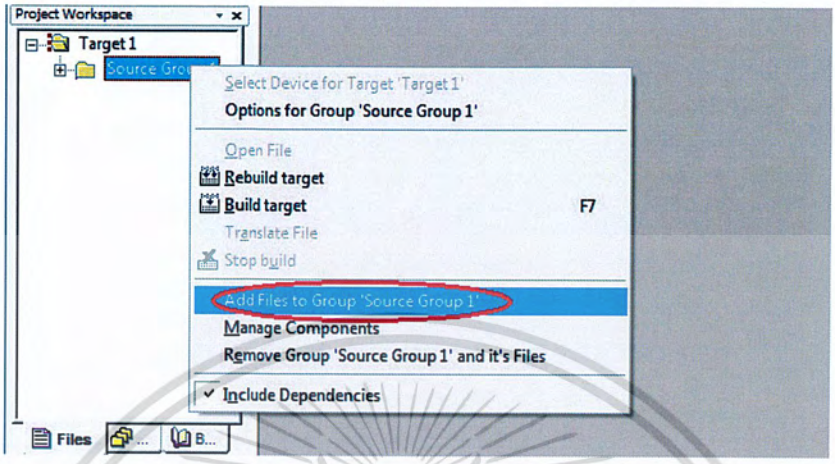
รูปที่ ก.4.6 หน้าจอแสดงการเลือกเบอร์ชิปที่ใช้

ในภาษา C ตัวแปรจะมี Startup code มาให้อยู่แล้ว ซึ่งสามารถตอบ No ได้ ซึ่งจะเป็นการใช้ตัวมาตรฐานของมันเลย แต่ในกรณีที่ใช้บอร์ดพิเศษ ก็จะเลือกตอบ Yes



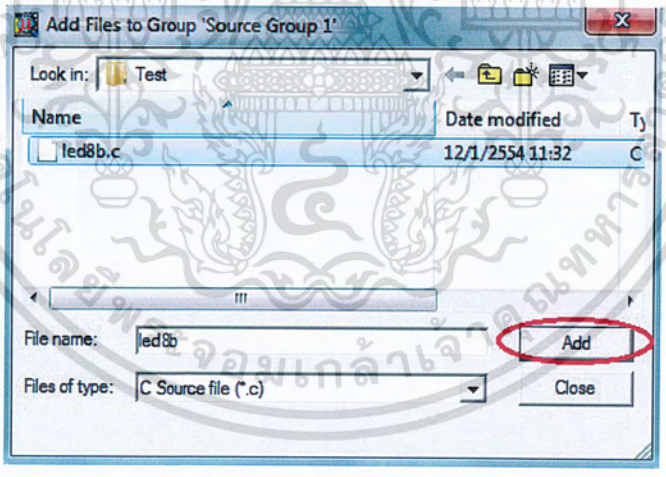
รูปที่ ก.4.7 หน้าจอแสดงการเลือก startup code

จากนั้น คลิกขวาที่ Source Group แล้วเลือก Add Files to Group



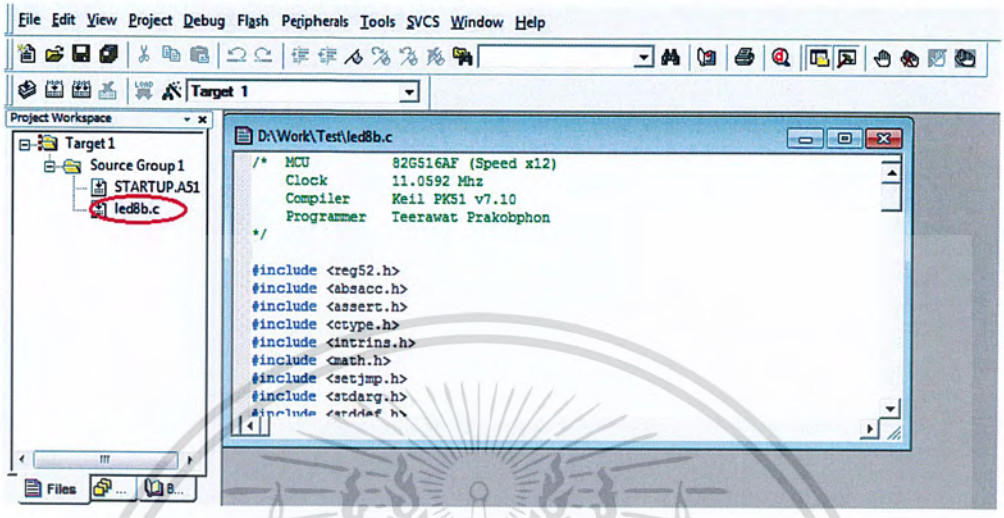
รูปที่ ก.4.8 หน้าจอแสดงการนำไฟล์ภาษาซีเข้ามา

เลือกไฟล์ที่ต้องการ แล้วกด Add



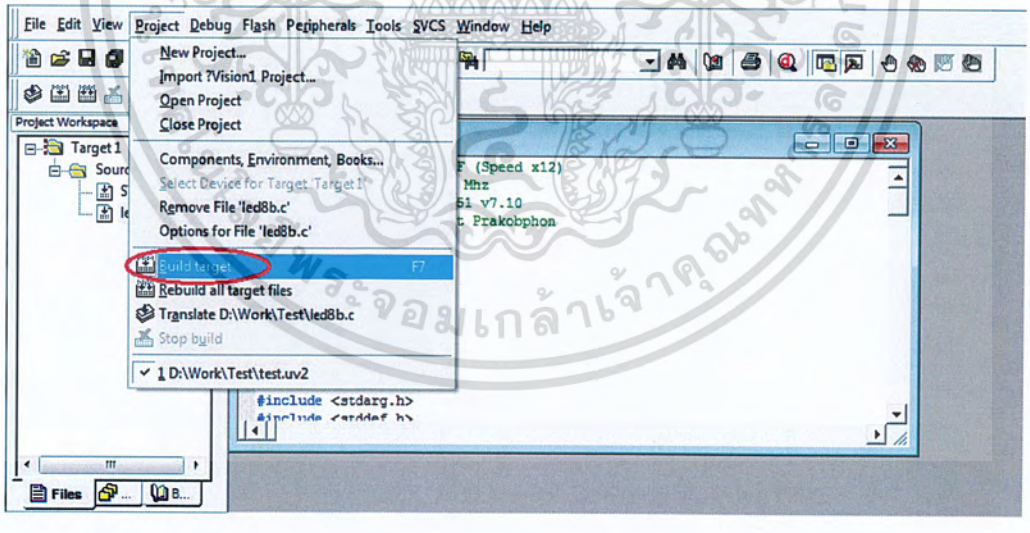
รูปที่ ก.4.9 หน้าจอแสดงการเลือกไฟล์ภาษาซี

จะมีไฟล์ที่เลือกไว้ปรากฏขึ้นทางซ้ายมือ



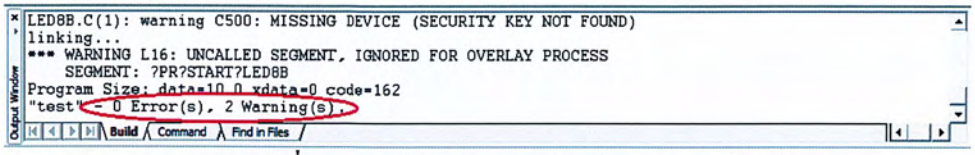
รูปที่ ก.4.10 หน้าจอแสดงไฟล์ภาษาซีที่เลือกไว้

ทำการ Compile โดยไปที่ Project เลือก Build target



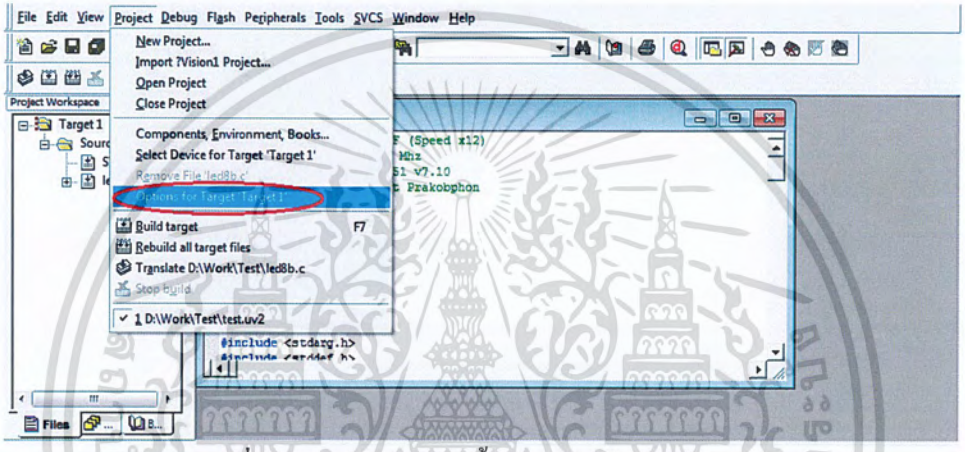
รูปที่ ก.4.11 หน้าจอแสดงการคอมไพล์โปรแกรม

ถ้ามันขึ้น Error แสดงว่าโปรแกรมมีจุดที่ผิด ต้องทำการแก้ไข แต่ถ้า Error เป็น 0 แสดงว่าโปรแกรมนั้นถูกต้อง



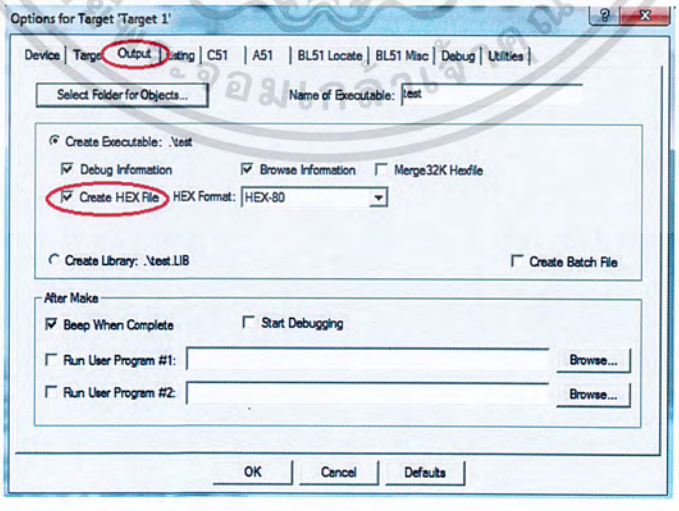
รูปที่ ก.4.12 หน้าจอแสดงผลลัพธ์จากการคอมไพล์

หลังจากนั้น ไปที่ Project เลือก Options for Target



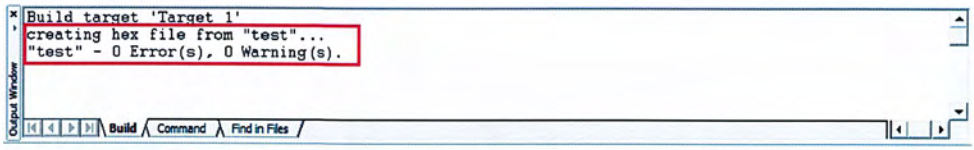
รูปที่ ก.4.13 หน้าจอแสดงขั้นตอนการเลือก HEX file

กด Output แล้วทำเครื่องหมายในช่องของ Create HEX file เพื่อสร้าง hex file แล้วนำไปโหลดลงบอร์ด



รูปที่ ก.4.14 หน้าจอแสดงการเลือกไฟล์ .HEX

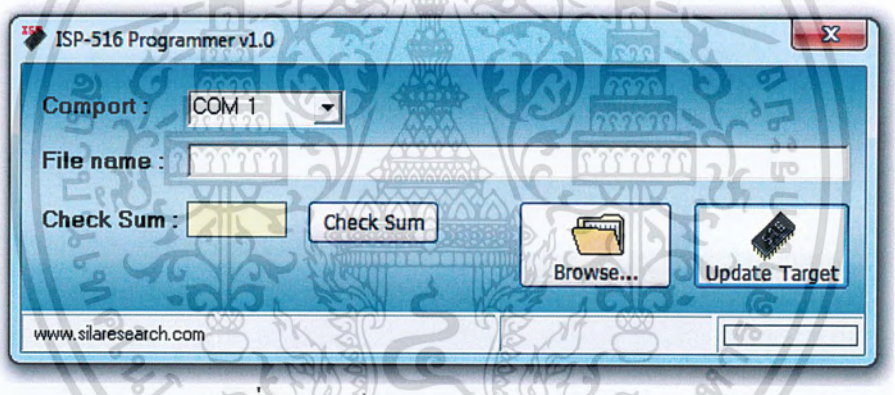
จากนั้น Compile อีกครั้งหนึ่ง



รูปที่ ก.4.15 หน้าจอแสดงการคอมไพล์โปรแกรมเมื่อเลือกไฟล์ .HEX แล้ว

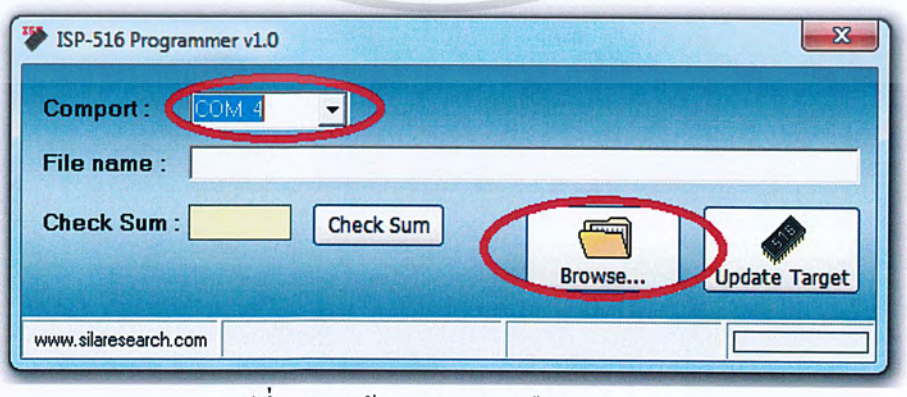
ก.5 การโหลดโปรแกรมลงบอร์ดด้วยโปรแกรม ISP-516

เปิดโปรแกรม ISP-516 ขึ้นมา จะแสดงดังรูป



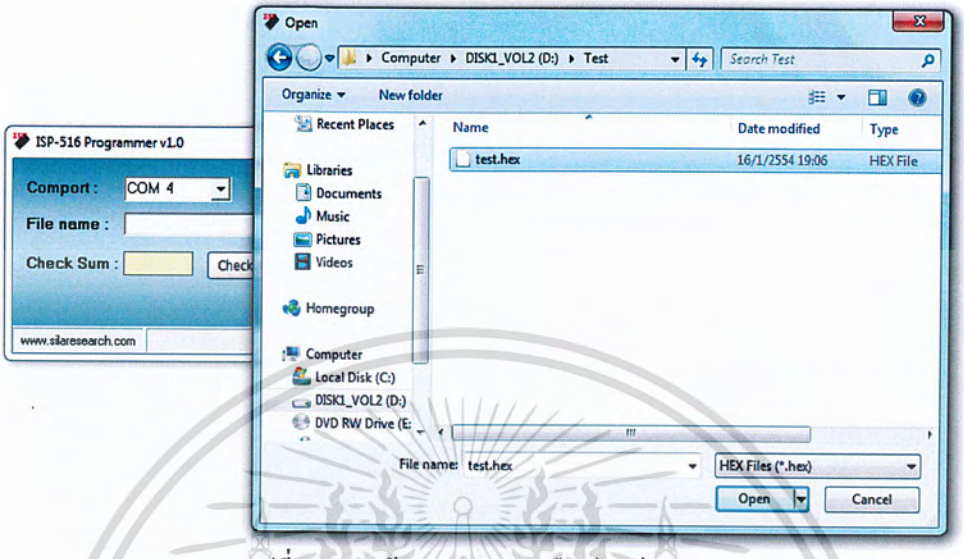
รูปที่ ก.5.1 หน้าจอแสดงโปรแกรม ISP-516

ทำการเลือก Comport หลังจากนั้นกด Browse เพื่อเลือกไฟล์ .hex ที่ต้องการ



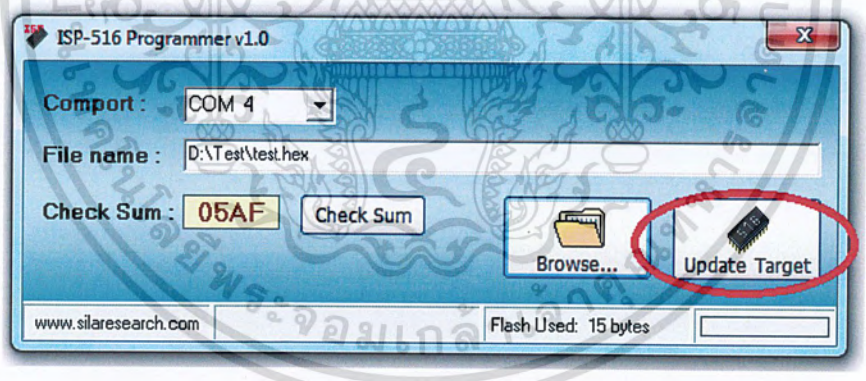
รูปที่ ก.5.2 หน้าจอแสดงการเลือก comport

หลังจากกด Browse จะแสดงดังรูป ทำการเลือกไฟล์แล้วกด open



รูปที่ ก.5.3 หน้าจอแสดงการเลือกไฟล์ .HEX

กด Update Target เพื่อทำการโหลดโปรแกรมลงชิป แล้วกด reset ที่ตัวบอร์ด เป็นอันเสร็จเรียบร้อย



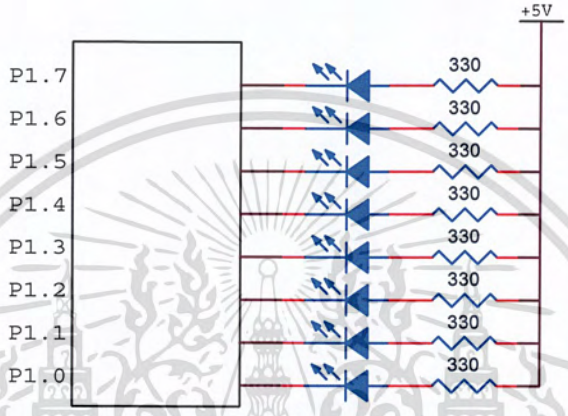
รูปที่ ก.5.4 หน้าจอแสดงการโหลดโปรแกรมลงบอร์ด

ภาคผนวก ข.

โปรแกรมและการทดลอง

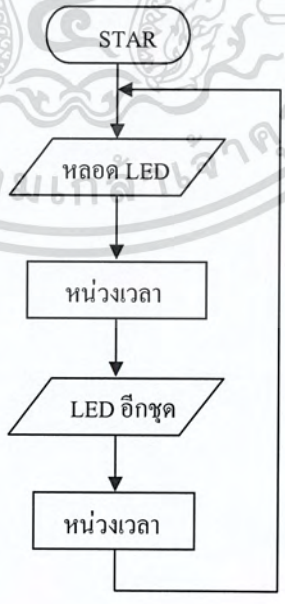
ข.1 การทดลองที่ 1 : การเขียนโปรแกรม LED ไฟวิ่ง 8 ดวง

การทดลองเขียนโปรแกรมสำหรับไมโครคอนโทรลเลอร์อย่างง่ายคือ การเขียนโปรแกรมควบคุมหลอด LED ที่ต่ออยู่กับพอร์ตให้สว่างหรือดับ ถ้าหากหลอด LED หนึ่งหลอด ใช้พอร์ตควบคุม 1 บิต การทำให้หลอดสว่างหรือดับทำได้โดยการควบคุมบิตที่ใช้งานให้มีลอจิกเป็น 0 หรือ 1 จะทำให้ควบคุมกระแสไฟฟ้าที่ไหลผ่านหลอด LED ได้ ถ้าหากมีวงจรต่ออยู่ดังรูป



รูปที่ ข.1.1 แสดงวงจรที่มีหลอด LED ต่ออยู่

จากวงจร ถ้าหากบิตใดเป็นลอจิก “0” จะทำให้กระแสไฟฟ้าไหลผ่านหลอด LED ได้ ทำให้หลอดไฟสว่าง ถ้าหากต้องการเขียนโปรแกรมให้หลอด LED สว่างติด ดับสลับกันไป จะออกแบบโปรแกรมได้ดังนี้



รูปที่ ข.1.2 แสดงการทำงานการติด ดับ ของหลอด LED สลับกัน

สำหรับการกำหนดบิตของพอร์ตที่ต่ออยู่กับหลอด LED นั้น สามารถใช้คำสั่ง sbit ในการนิยามบิตได้ ถ้าหากกำหนดค่าลอจิกให้ตัวแปรของบิตที่กำหนดไว้ จะทำให้บิตดังกล่าวมีค่าเป็นไปตามที่กำหนดทันที ตัวอย่างเช่น

```
sbit    P10    =    P1^0;          //ให้ P10 แทนพอร์ต 1 บิต 0
        P10    =    1;             //กำหนดให้ P10 เป็นลอจิก 1 ดังนั้น LED ดับ

        P10    =    0;             //กำหนดให้ P10 เป็นลอจิก 0 ดังนั้น LED สว่าง
```

จากการต่อวงจรดังรูป เมื่อให้เอาต์พุตของบิตพอร์ตเป็น “0” จะทำให้กระแสไฟฟ้าจากแหล่งจ่ายไฟ ไหลผ่านตัวต้านทานขนาด 330 โอห์ม ผ่านหลอด LED เข้าสู่บิตพอร์ตของไมโครคอนโทรลเลอร์ ทำให้หลอด LED สว่าง

บอร์ตไมโครคอนโทรลเลอร์ที่ใช้ในการทดลองนี้ยังได้นำแหล่งกำเนิดเสียงมาติดตั้งไว้ด้วย การควบคุมให้ส่งเสียงทำได้โดยส่งลอจิก “1” เป็นช่วงเวลาสั้น ๆ ไปยังบิตพอร์ตที่ต่ออยู่กับลำโพง

สำหรับการเขียนโปรแกรมหน่วงเวลานั้น อาจทำได้โดยสร้างโปรแกรมย่อยขึ้นมา แล้วให้ไมโครคอนโทรลเลอร์วนลูปให้เสียเวลาไป ช่วงเวลาที่เสียไปจะขึ้นอยู่กับจำนวนครั้งของการวนลูป ถ้าหากต้องการให้หลอด LED สว่างติดดับสลับกันไป อาจเขียนโปรแกรมได้ดังนี้

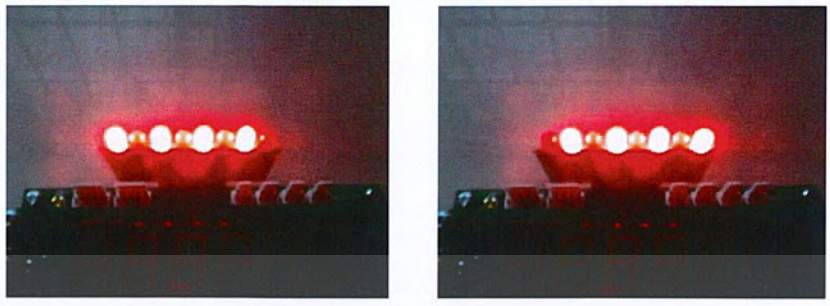
```
#include <reg52.h>
#inclund <dmsec.h>

/********** I/O PORT **********/
sbit   P10      =     P1^0;
sbit   P11      =     P1^1;
sbit   P14      =     P1^4;
sbit   P15      =     P1^5;
sbit   P16      =     P1^6;
sbit   P17      =     P1^7;
sbit   P12      =     P1^2;
sbit   P13      =     P1^3;

sbit   BUZZER   = P2^3;  // buzzer
```

ประกาศตัวแปร ตามบิตพอร์ตที่กำหนด

```
void beep (void) {BUZZER = 0; dmsec (250); BUZZER = 1;}  // ส่งเสียง beep สั้นๆ

void test_port()                          //โปรแกรมย่อยแสดงไฟกระพริบ
{
   P10 = 1;
   P11 = 0;
   P12 = 1;
   P13 = 0;
   P14 = 1;
   P15 = 0;
   P16 = 1;
   P17 = 0;
  dmsec(500);                 //หน่วงเวลา ครึ่งวินาที
   P10 = 0;
   P11 = 1;
   P12 = 0;
   P13 = 1;
   P14 = 0;
   P15 = 1;
   P16 = 0;
   P17 = 1;
  dmsec(500);
}
void start (void) {              // start process
      dmsec (250);
      beep ();
}
void main (void) {
     start();
     test_port();
}
```

กำหนดรูปแบบการสว่างและดับให้กับหลอด LED

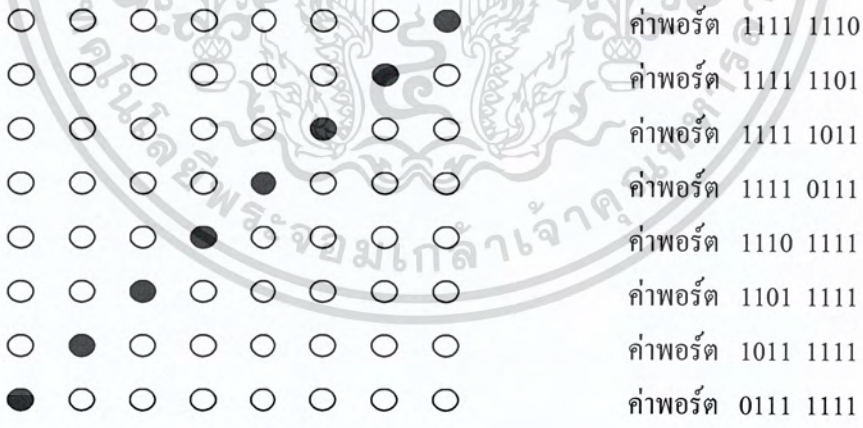
เมื่อรันโปรแกรมจะพบว่าหลอด LED จะติดดับสลับกันไป ดังรูป



รูปที่ ข.1.3 แสดงภาพของหลอด LED ที่ติดดับสลับกัน

ถ้าหากต้องการให้หลอด LED สว่างทีละดวงเรียงกันไปดังรูป

การเขียนโปรแกรมลักษณะนี้สามารถนำวิธีการเลื่อนบิตมาใช้ได้ โดยขั้นแรกจะส่งค่า FEH หรือ 1111 1110 ออกไปก่อน เพื่อให้หลอด LED หลอดแรกสว่าง จากนั้นหน่วงเวลา แล้วเลื่อนบิตทั้งหมดไปทางซ้าย ทำซ้ำไปเรื่อย ๆ โปรแกรมจะแสดงผลเป็นไฟวิ่งทีละหลัก



จากหลักการที่กล่าวมา การเขียนโปรแกรมทำได้โดยประกาศตัวแปรขึ้นมาหนึ่งตัวให้มีขนาดเป็นไบต์ ในที่นี้ให้ชื่อว่า BITM โดยไบต์ข้อมูลจะให้อ้างข้อมูลเป็นระดับบิตได้ ให้ชื่อว่า B0 ถึง B7

B7	B6	B5	B4	B3	B2	B1	B0

ตัวแปร BITM

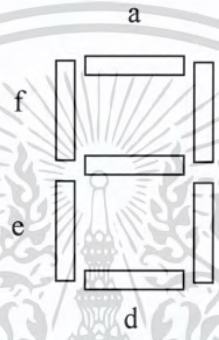
ในการเลื่อนบิตจะใช้วิธีเลื่อนข้อมูลในไบต์ BITM ไป 8 ครั้ง ตัวอย่างนี้สามารถเขียนเป็นโปรแกรมได้ดังนี้

```
#include <reg52.h>
#inclund <dmsec.h>

/********** I/O PORT **********/
sbit    P10         = P1^0;   // 8B Port
sbit    P11         = P1^1;
sbit    P14         = P1^4;
sbit    P15         = P1^5;
sbit    P16         = P1^6;
sbit    P17         = P1^7;
sbit    P12         = P1^2;
sbit    P13         = P1^3;

sbit    BUZZER    = P2^3;   // buzzer

/********** INT-RAM WORKING AREA **********/
unsigned char bdata    BITM;
sbit            B0 = BITM^0;
sbit            B1 = BITM^1;
sbit            B2 = BITM^2;
sbit            B3 = BITM^3;
sbit            B4 = BITM^4;
sbit            B5 = BITM^5;
sbit            B6 = BITM^6;
sbit            B7 = BITM^7;

/********** BASIC FUNCTION **********/

void beep (void) {BUZZER = 0; dmsec (250); BUZZER = 1;}    // beep short
```

```
void test8bsub (void) {
   P10 = B0;
   P11 = B1;
   P12 = B2;
   P13 = B3;
   P14 = B4;
   P15 = B5;
   P16 = B6;
   P17 = B7;
}

void test8b (void) {
   unsigned char i,k;
   for (k=1;k<=10;k++) {
      BITM = 0xfe;
      for (i=1;i<=8;i++) {
         test8bsub ();
         dmsec (200);
         BITM = (BITM << 1) | 0x01;
      }
   }
}

void start (void) {              // start process
   dmsec (250);
   beep ();
}

void main (void) {
     start();
     test8b ();
}
```

ข.2 การทดลองที่ 2 : การเขียนโปรแกรมแสดงผลตัวเลขทาง LED 7 ส่วน

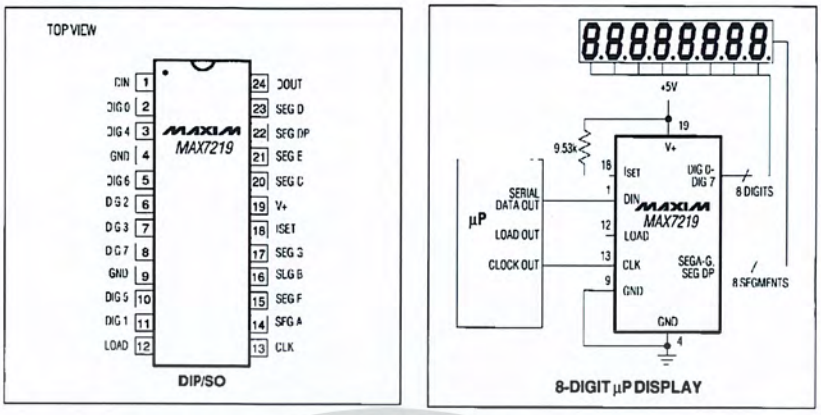
หลอด LED 7 ส่วน (LED 7-segment) จะเกิดจากการนำหลอด LED จำนวน 7 หลอดมาต่อเรียงกันเพื่อให้แสดงผลเป็นตัวเลขได้ ตัว LED แต่ละหลอดหรือแต่ละเซกเมนต์จะมีชื่อประจำคือ a, b, c, d, e, f และ g การนำหลอด LED มาต่อลักษณะนี้จะทำให้แสดงเป็นตัวเลขหรืออักขระบางตัวได้ ถ้าหากต้องการให้หลอด LED สว่างเป็นเลขใดทำโดยให้หลอดแต่ละหลอดสว่างประกอบกันให้เป็นเลขนั้น ถ้าหากให้หลอด LED ทุกหลอดสว่างพร้อมกันหมดจะแสดงเป็นเลข 8



รูปที่ ข.2.1 แสดงหลอดแต่ละหลอดของ LED 7 ส่วน

จากโครงสร้างของหลอด LED จะพบว่าการให้แสดงผลเป็นตัวเลขจะต้องใช้บิตพอร์ตอย่างน้อย 7 บิต ในการแสดงผลหนึ่งหลัก ถ้าหากต้องการสร้างหน่วยแสดงผลให้แสดงหลายหลักจะต้องใช้บิตพอร์ตจำนวนมาก หรือใช้เทคนิคการสแกนแทน แต่การเขียนโปรแกรมสำหรับให้สแกนเป็นเลขต่าง ๆ นั้น โปรแกรมเขียนได้ยาก และไมโครคอนโทรลเลอร์ต้องเสียเวลาไปในการสแกนหน่วยแสดงผล ในปัจจุบันได้มีการผลิตฮาร์ดแวร์ออกมาในรูปของวงจรรวม (IC) ให้สแกนหลอด LED หลาย ๆ หลักได้เอง โดยไม่ต้องเขียนโปรแกรมสแกน เพียงแต่ผู้ใช้งานสั่งให้ไมโครคอนโทรลเลอร์ส่งข้อมูลที่ต้องการแสดงผลไปให้ไอซีแบบอนุกรม จากนั้นจะเป็นหน้าที่ของไอซีแสดงหน่วยแสดงผลเอง

ในการทดลองนี้เราจะเขียนโปรแกรมควบคุมการแสดงผล LED 7 เซกเมนต์ โดยผ่านตัวสแกนที่มีชื่อว่า MAX7219



รูปที่ ข.2.2 แสดงส่วนประกอบต่าง ๆและตัววงจรของ MAX7219

จากวงจร ไมโครคอนโทรเลอร์จะส่งข้อมูลที่ต้องแสดงผลทั้งหมดไปให้ MAX7219 แบบอนุกรม โดยใช้สายสัญญาณ 3 บิต โดยข้อมูลจะส่งออกไปทีละบิตทาเข้ามาทางขา DIN โดยมีสัญญาณนาฬิกาเป็นตัวควบคุมทางขา CLK การให้ตัวเลขสว่างหนึ่งหลักจะต้องส่งข้อมูลออกไปทั้งหมด 16 บิต ในการส่งข้อมูลนั้นจะต้องทำการส่งไปทีละบิต สลับกับการส่งสัญญาณนาฬิกา (CLK) เข้าไปในรีจิสเตอร์ด้วย ซึ่งข้อมูลที่ส่งไปนั้นจะถูกควบคุมโดยขา LOAD

รูปที่ ข.2.3 แสดงไดอะแกรมเวลาการส่งข้อมูลให้ไอซี MAX7219

ในการใช้งานไอซี MAX7219 นั้น เริ่มต้นจะต้องส่งข้อมูลค่าเริ่มต้นให้กับรีจีสเตอร์ต่าง ๆ ภายในตัวมันก่อน เพื่อกำหนดรูปแบบการทำงาน โดยรีจีสเตอร์ต่าง ๆ จะมีทั้งหมด 14 ตัว เป็นรีจีสเตอร์สำหรับเก็บข้อมูลที่จะให้แสดงผลแต่ละหลักจำนวน 8 ตัว ส่วนที่เหลือเป็นรีจีสเตอร์ควบคุม เช่น กำหนดความสว่างของ LED กำหนดว่าการสแกนนั้นจะให้สแกนถึงหลักใด กำหนดว่าจะให้วงจรถอดรหัสภายในทำการถอดรหัสสำหรับ LED 7 ส่วนหรือไม่ รีจีสเตอร์แต่ละตัวจะมีแอดเดรสประจำตัวมัน โดยหน้าที่ของแอดเดรสแต่ละตัวเป็นดังตารางต่อไปนี้

ตารางที่ ข.2.1 แสดงรีจีสเตอร์และแอดเดรสใน MAX7219

รีจีสเตอร์	แอดเดรส	การทำงาน
NO-OP	00	ส่งข้อมูลให้ไอซีที่ต่อกันหลายตัว
DIGIT 0	01	ส่งข้อมูลให้หลักที่ 0
DIGIT 1	02	ส่งข้อมูลให้หลักที่ 1
DIGIT 2	03	ส่งข้อมูลให้หลักที่ 2
DIGIT 3	04	ส่งข้อมูลให้หลักที่ 3
DIGIT 4	05	ส่งข้อมูลให้หลักที่ 4
DIGIT 5	06	ส่งข้อมูลให้หลักที่ 5
DIGIT 6	07	ส่งข้อมูลให้หลักที่ 6
DIGIT 7	08	ส่งข้อมูลให้หลักที่ 7
DECODE MODE	09	กำหนดว่าจะมีการถอดรหัสข้อมูลหรือไม่
INTENSITY	0A	กำหนดความสว่างของหลอด LED
SCAN LIMIT	0B	กำหนดจำนวนหลักที่ต้องการให้การสแกนไปถึง
SHUTDOWN	0C	กำหนดการ Shutdown
DISPLAY TEST	0F	ทดสอบการแสดงผล

ถ้าหากนำ MAX7219 ไปต่อกับ LED 7 ส่วน การให้ LED แต่ละหลักแสดงเป็นตัวเลขต่าง ๆ นั้นจะต้องส่งไบต์ข้อมูลให้กับหน่วยความจำภายในของ MAX7219 ดังตารางต่อไปนี้

ตารางที่ ข.2.2 ข้อมูลที่ส่งให้กับรีจีสเตอร์ภายใน โดยแสดงถึงการแสดงผลในแต่ละเซกเมนต์

รูปตัวเลข	A	B	C	D	E	F	G	ค่าไบต์ข้อมูล
0	1	1	1	1	1	1	0	7EH
1	0	1	1	0	0	0	0	30H
2	1	1	0	1	1	0	1	6DH
3	1	1	1	1	0	0	1	79H
4	0	1	1	0	0	1	1	33H
5	1	0	1	1	0	1	1	5BH
6	1	0	1	1	1	1	1	5FH
7	1	1	1	0	0	0	0	70H
8	1	1	1	1	1	1	1	7FH
9	1	1	1	1	0	1	1	7BH

-	0	0	0	0	0	0	1	01H
E	1	0	0	1	1	1	1	4FH
H	0	1	1	0	1	1	1	37H
L	0	0	0	1	1	1	0	0EH
P	1	1	0	0	1	1	1	67H
Blank	0	0	0	0	0	0	0	00H

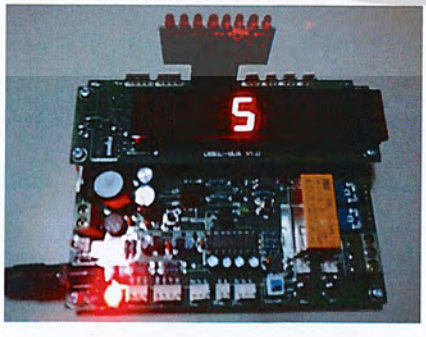
การนำไอซี MAX7219 ไปใช้งานควรแบ่งหน่วยความจำ RAM สำหรับเก็บข้อมูลที่ต้องการให้ MAX7219 สแกน ในที่นี้เรียกว่า DISBUF โดยให้หนึ่งไบต์แทนหนึ่งหลักของการแสดงผล เมื่อต้องการแสดงผลค่าตัวเลขใดจะนำรูปแบบการแสดงผลไปเก็บไว้ในหน่วยความจำ DISBUF ก่อน จากนั้นจะโหลดข้อมูลทั้งหมดลงไปในชิพอีกครั้งหนึ่ง

สำหรับในชุดทดลองนี้ได้พัฒนาโปรแกรมย่อยสำหรับควบคุมการแสดงผลทางแผง LED ขนาด 8 หลักไว้ให้ใช้งานแล้ว โดยได้สร้างฟังก์ชันต่าง ๆ สำหรับควบคุมการแสดงผลให้ใช้งานได้ง่าย โดยรวมไว้ในไฟล์ชื่อ shownum.h ดังนั้นการเขียนโปรแกรมแสดงผลตัวเลขทางหลอด LED 7 เซกเมนต์เราจะต้อง include ไฟล์ shownum.h ขึ้นมาก่อน ซึ่งในไฟล์ประกอบไปด้วยฟังก์ชันที่สำคัญ 3 ฟังก์ชันดังนี้

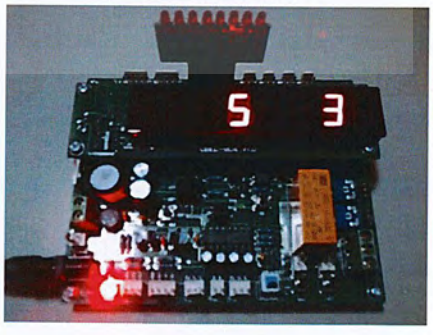
- dis_num_digit(x,y);

ฟังก์ชันนี้จะแสดงผลตัวเลขทางหลอด LED 7 เซกเมนต์ โดยที่พารามิเตอร์ x คือ หลัก(1-8)ที่ต้องการแสดงผลและพารามิเตอร์ y คือ ค่า(0-9)ที่ต้องการแสดงผล ดังตัวอย่าง เช่น

dis_num_digit(4,5);



dis_num_digit(4,5);
dis_num_digit(3,7);

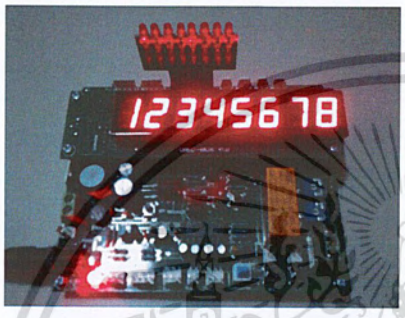


รูปที่ ข.2.4 แสดงผลตัวเลขและหลักที่ต้องการทาง LED 7 ส่วน

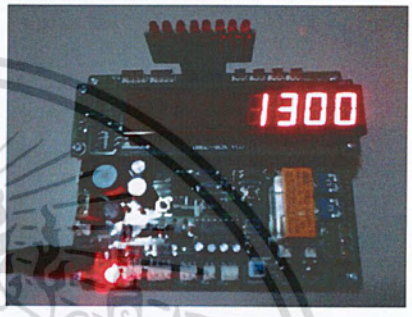
- dis_num(x);

ฟังก์ชัน dis_num() จะแสดงผลตัวเลข LED 7 เซกเมนต์ โดยที่ค่า x คือค่าที่ต้องการจะให้แสดงผลโดยค่าที่จะแสดงนั้นจะแสดงได้ตั้งแต่ 1- 8 หลัก และแต่ละหลักจะเป็นตัวเลข 0 – 9 เช่น ตัวอย่างเช่น

dis_num(12345678);



dis_num(1300);

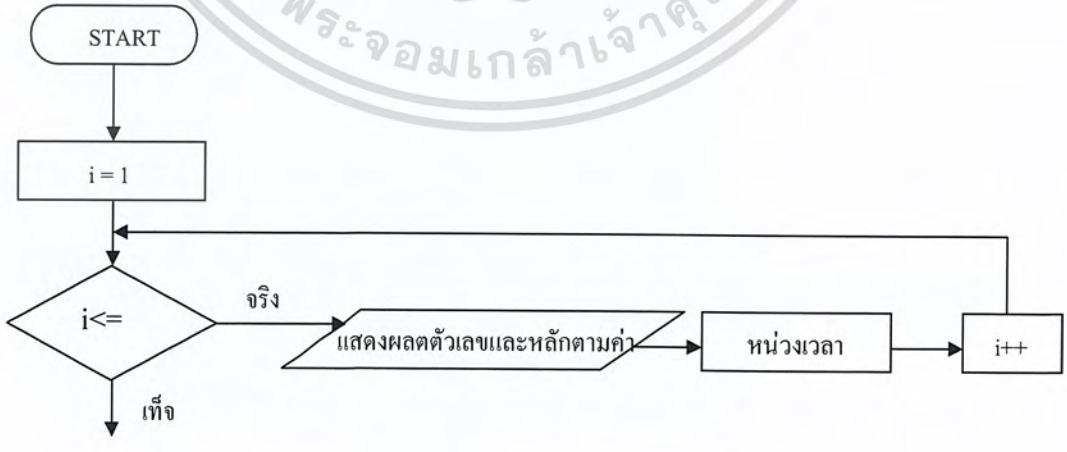


รูปที่ ข.2.5 แสดงผลตัวเลขหลายหลักทาง LED 7 ส่วน

- disclear();

เป็นฟังก์ชันสำหรับเคลียร์ค่าต่างๆที่อยู่ใน DISBUF ให้มีค่าเป็น 0

ถ้าหากต้องการเขียนโปรแกรมให้หลอด LED สว่างเป็นตัวเลขทีละหลัก จะออกแบบโปรแกรมได้ดังนี้



รูปที่ ข.2.6 แสดงการทำงานของโปรแกรมหลอด LED สว่างเป็นตัวเลขทีละหลัก

สำหรับตัวอย่างโปรแกรมให้แสดงตัวเลข 12345678 สว่างทีละหลักสามารถเขียนโปรแกรมได้ดังนี้

```
#include <reg52.h>
#include<shownum.h>
#inclund <dmsec.h>

sbit    BUZZER   =   P2^3;  // buzzer

void beep (void) {BUZZER = 0; dmsec (250); BUZZER = 1;}    // ส่งเสียง beep สั้น ๆ

void test_disdigit(void){
        unsigned char i;
        for(i=1;i<=8;i++){
            dis_num_digit(i,i);
            dsmsec(1000);
        }
}

void start (void) {             // start process
      dmsec (250);
      beep ();
}

void main (void) {
     start();
    disclear();
     test_disdigit();
}
```

รูปแบบการแสดงผลตัวเลข ตั้งแต่ 1 – 8 ตามค่าและหลัก

เมื่อรันโปรแกรมจะพบว่าหลอด LED จะสว่างเป็นตัวเลขตามค่าและหลัก ดังรูป

1								i = 1
	2							i = 2
		3						i = 3
			4					i = 4
				5				i = 5
					6			i = 6
						7		i = 7
							8	i = 8

รูปที่ ข.2.7 แสดงผลหลอด LED สว่างเป็นตัวเลขตามค่าและหลัก

ถ้าหากเราต้องการแสดงผลตัวเลขพร้อมกันทุกหลักแล้วเลื่อนค่าไปทีละหลักเราสามารถนำฟังก์ชัน dis_num(); มาใช้ดังรูป

1	2	3	4	5	6	7	8
8	1	2	3	4	5	6	7
7	8	1	2	2	4	5	6
6	7	8	1	2	3	4	5
5	6	7	8	1	2	3	4
4	5	6	7	8	1	2	3
3	4	5	6	7	8	1	2
2	3	4	5	6	7	8	1

รูปที่ ข.2.8 แสดงผลตัวเลขทุกตัวพร้อมกันแล้วเลื่อนไปทีละหลัก

สามารถเขียนโปรแกรมได้ดังนี้

```
#include <reg52.h>
#include<shownum.h>
#inclund <dmsec.h>

sbit    BUZZER   =   P2^3;   // buzzer
```

```
void beep (void) {BUZZER = 0; dmsec (250); BUZZER = 1;}    // ส่งเสียง beep สั้น ๆ

void test_disnum (void){
        dis_num(12345678);
        dmsec(1000);
        dis_num(81234567);
        dmsec(1000);
        dis_num(78123456);
        dmsec(1000);
        dis_num(67812345);
        dmsec(1000);
        dis_num(56781234);
        dmsec(1000);
        dis_num(45678123);
        dmsec(1000);
        dis_num(34567812);
        dmsec(1000);
        dis_num(23456781);
        dmsec(1000);
}
void start (void) {          // start process
      dmsec (250);
        beep ();
}
void main (void) {
    start();
   disclear();
    test_disnum();
}
```

การแสดงผลของ dis_num() ตามที่กำหนด

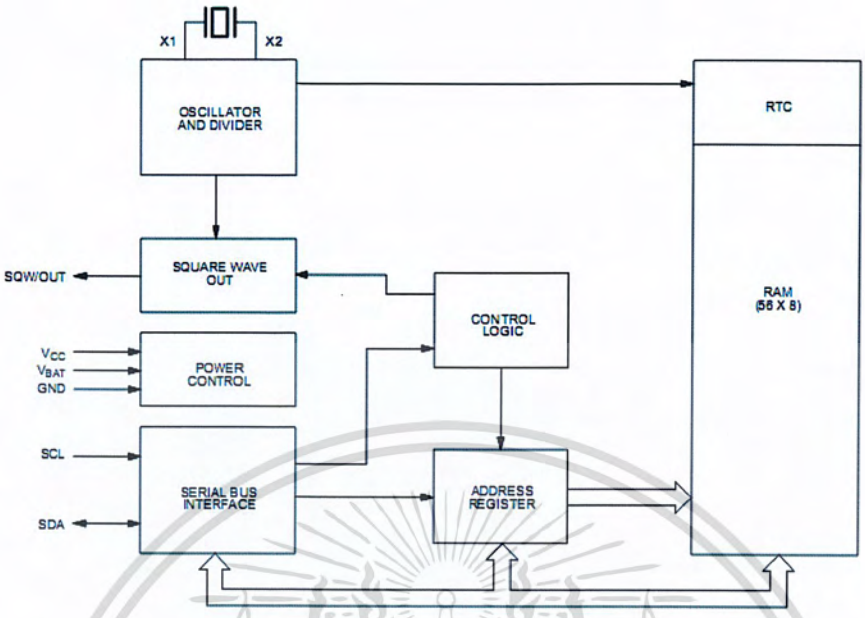
ข.3 การทดลองที่ 3 : การเขียนโปรแกรมระบบเวลาจริง

ระบบที่ใช้ไมโครคอนโทรลเลอร์ในการควบคุมบางระบบจะมีการนำเวลาเข้ามาเกี่ยวข้องด้วย เช่นระบบตั้งเวลาปิดเปิดอุปกรณ์ต่าง ๆ โดยอาจใช้ในการดูวินาที นาที ชั่วโมง หรือดูวัน เดือน ปี โดยค่าเวลาจะต้องมีค่าถูกต้องแน่นอน การสร้างฐานเวลาให้กับไมโครคอนโทรลเลอร์มี 2 วิธี วิธีแรกจะใช้วงจรกำเนิดสัญญาณนาฬิกาป้อนให้กับตัวไทเมอร์ 0 หรือไทเมอร์ 1 เพื่อให้ไทเมอร์ส่งสัญญาณมาอินเทอร์รัปต์ซีพียูในช่วงเวลาที่กำหนด อีกวิธีหนึ่งจะใช้ชิปภายนอกสร้างฐานเวลาปัจจุบันให้กับระบบไมโครคอนโทรลเลอร์ แต่ในชุดทดลองที่เราจะเขียนโปรมแกรมนั้นจะใช้ชิปภายนอกสร้างฐานเวลาให้กับไมโครคอนโทรลเลอร์ โดยใช้ชิป DS1307

ไอซีฐานเวลาจริง DS1307

ชิปไอซี DS1307 เป็นชิปที่สร้างฐานเวลาจริง โดยสามารถรับส่งข้อมูลให้กับไมโครคอนโทรลเลอร์ได้โดยใช้บัสแบบ I²C สามารถให้ข้อมูลเกี่ยวกับเวลา เช่น วินาที นาที ชั่วโมง (ทั้งแบบ 24 ชั่วโมง และแบบ 12 ชั่วโมง พร้อมระบุค่า AM/PM) และบอกวัน เดือน ปีได้ คุณสมบัติที่สำคัญมีดังนี้

- เป็นไอซีแบบ 8 ขา กินพลังงานต่ำมาก โดยกินกระแสน้อยกว่า 500 นาโนแอมป์ ในโหมดแบตเตอรี่สำรอง
- นับสัญญาณนาฬิกาเป็นวินาที นาที ชั่วโมง วัน วันที่ เดือน และปีได้อย่างถูกต้องไปถึงปี ค.ศ. 2100
- มีหน่วยความจำภายในแบบ nonvolatile RAM ขนาด 56 ไบต์ ไว้เก็บข้อมูลเวลาภายใน
- เชื่อมต่อกับไมโครคอนโทรลเลอร์โดยใช้บัสแบบ I²C
- สามารถโปรแกรมให้สร้างคลื่นรูปสี่เหลี่ยม (squarewave) ออกมาได้



รูปที่ ข.3.1 โครงสร้างภายใน DS1307

การเขียนโปรแกรมแสดงเวลาจริง เราจะต้อง include ไฟล์ rtc.h ซึ่งไฟล์ rtc.h จะประกอบไปด้วย ฟังก์ชันดังนี้

- setrtc(d,mo,y,h,mi,se)

  ฟังก์ชันนี้จะเป็นการตั้งค่าวันเวลา ในการตั้งค่านั้นเราจะใส่เลขฐาน16 ให้กับพารามิเตอร์แต่ละตัวโดยพารามิเตอร์แต่ละตัวมีค่า ดังนี้

  d คือ วันที่ ตัวอย่างเช่น 0x10 คือวันที่ 10

  mo คือ เดือน ตัวอย่างเช่น 0x02 คือ เดือนที่2(เดือนกุมภาพันธ์)

  y คือ ปี เช่น 0x10 คือ ปีที่ 10

  h คือ ชั่วโมง เช่น 0x11 คือ 11 นาฬิกา

  mi คือ นาที เช่น 0x55 คือ 55 นาที

  se คือ วินาที เช่น 0x59 คือ 59 วินาที

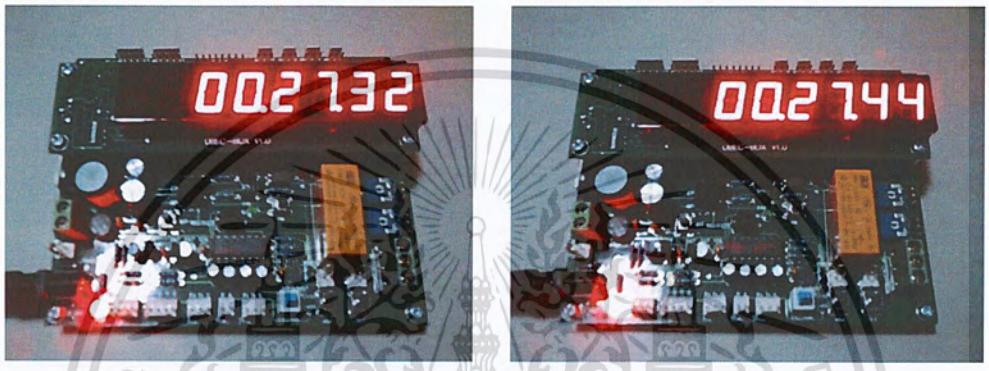
หลังจากการตั้งค่านี้ไฟล์ rtc.h จะมีฟังก์ชันที่ทำการเขียนข้อมูลลงบนชิป DS1307 ทำให้เวลาเดินตลอดเวลาเนื่องจากใช้พลังงานจากแบตเตอร์รี่ และเราสามารถเรียกวันเวลาปัจจุบันดูได้ตามที่ต้องการ ตัวอย่างการใช้งานฟังก์ชัน setrtc();

setrtc(0x02,0x02,0x10,0x00,0x27,0x32);// คือ วันที่ 2 เดือน 2 ปีที่ 10 เวลา 00:27:32

- dis_rtc()

ฟังก์ชันนี้เป็นการแสดงผลนาฬิกาเวลาปัจจุบัน เราสามารถเรียกใช้ฟังก์ชันนี้ได้เลย โดยการแสดงค่าของเวลานั้นจะทำการอ่านข้อมูลจากชิป DS 1307 แล้วแสดงผลออกทาง LED 7 เซกเมนต์ ตัวอย่างการใช้งานฟังก์ชัน dis_rtc()

dis_rtc();//เรียกใช้ฟังก์ชันเพื่อแสดงเวลาขณะนั้นทาง หลอด LED 7เซกเมนต์



รูปที่ ข.3.2 แสดงผลเวลาทาง LED 7 ส่วน

- rt_rtc(m)

คำสั่ง rt_rtc(m) เป็นคำสั่งคืนค่าของเวลามาเป็นข้อมูลประเภท char ซึ่งการขอคืนค่าจะต้องประกาศตัวแปร unsigned char sec = 0,min = 1,hour = 2,day = 4,month = 5,year = 6; พารามิเตอร์ที่ใช้ในการขอคืนค่าแต่ละตัวจะให้ค่าที่คืนกลับมาต่างกัน ดังนี้

rt_rtc(sec) คืนค่าของวินาที
rt_rtc(min) คืนค่าของนาที
rt_rtc(hour) คืนค่าของชั่วโมง
rt_rtc(day) คืนค่าของวัน
rt_rtc(month) คืนค่าของเดือน
rt_rtc(year) คืนค่าของปี

ส่วนมากเราจะคืนค่าของเวลามาใช้ในการเขียนโปรแกรมกำหนดการทำงานของชุดทดลอง เช่น กำหนด รีเลย์ เปิด-ปิดไฟ ตั้งเป็นนาฬิกาปลุก ตัวอย่างเช่น

```
if (rt_rtc(min) = = 0x00)    //min เป็นการคืนค่านาที ดังนั้นเมื่อนาทีถึง 00 มีเสียงดัง beep
    beep();
```

ตัวอย่างโปรแกรม ให้เขียนโปรแกรมเปิดปิดรีเลย์ เปิดเมื่อเวลาเดินถึง 18:00 และปิดเมื่อเวลาเดินถึง 05:00 ในการเขียนโปรแกรมนี้เราจะต้องกำหนดบิตใดบิตหนึ่งเป็นรีเลย์ก่อน ในชุดทดลองนี้เชื่อมต่อกับพอร์ต P0.4

```
#include <reg52.h>
#include <intrins.h>       //เรียกใช้ฟังก์ชัน เพื่อจะเรียก _nop_(); มาใช้
#include<shownum.h>    //include ไฟล์ shownnum.h เพื่อที่จะสามารถเรียกใช้ฟังก์ชันที่อยู่ภายในไฟล์ได้
#include<rtc.h>            //include ไฟล์ rtc.h เพื่อที่จะสามารถเรียกใช้ฟังก์ชันที่อยู่ภายในไฟล์ได้
#inclund <dmsec.h>

sbit    BUZZER    =    P2^3;   // buzzer
sbit     RELAY      =    P0^4;         //กำหนดให้ P0.4 เป็นรีเลย์

void relay_on (void){ RELAY = 0; }   //relay  on
void relay_off (void){ RELAY = 1; }   //relay  off
void beep (void) {BUZZER = 0; dmsec (250); BUZZER = 1;}    // ส่งเสียง beep สั้น ๆ

void start (void) {                  // start process
       dmsec (250);
          beep ();
}
void main (void) {
     unsigned char min= 1 ,hour = 2,day = 4,month = 5,year = 6;
     start();
     while(1){
          dis_rtc();
          if(rt_rtc(min) = =0x00 && rt_rtc(hour) ==0x18)
                    relay_on();
          if(rt_rtc(min) = =0x00 && rt_rtc(hour) ==0x05)
                    relay_off();
   }
}
```

ส่วนแสดงเวลาและเงื่อนไขในการเปิด-ปิด รีเลย์
เงื่อนไขแรกเปิดรีเลย์ เมื่อเวลา 18:00
เงื่อนไขที่สองปิดรีเลย์ เมื่อถึงเวลา 05:00

สำหรับตัวอย่างโปรแกรมให้ตั้งค่าวันที่และเวลาในขณะนั้น แล้วแสดงผลออกมาทางLED 7 เซกเมนต์ สามารถเขียนโปรแกรมได้ดังนี้

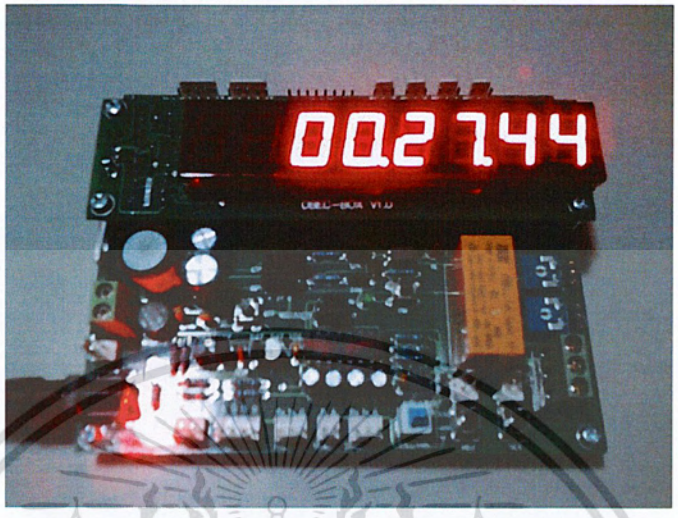
```
#include <reg52.h>
#include <intrins.h>      //เรียกใช้ฟังก์ชัน เพื่อจะเรียก _nop_(); มาใช้
#include<shownum.h>       //include ไฟล์ shownnum.h เพื่อที่จะสามารถเรียกใช้ฟังก์ชันที่อยู่ภายในไฟล์ได้
#include<rtc.h>           //include ไฟล์ rtc.h เพื่อที่จะสามารถเรียกใช้ฟังก์ชันที่อยู่ภายในไฟล์ได้
#inclund <dmsec.h>

sbit   BUZZER  =  P2^3;  // buzzer

void beep (void) {BUZZER = 0; dmsec (250); BUZZER = 1;}   // ส่งเสียง beep สั้น ๆ

void start (void) {              // start process
        dmsec (250);
        disclear ();
        mxload ();
        dmsec (250);
        beep ();
}

void main (void) {
    start();
    setrtc(0x02,0x02,0x10,0x00,0x27,0x44);
}
```

เรียกใช้ฟังก์ชันจากไฟล์ rtc.h เป็นการตั้งค่าวันเวลา

```
while(1){
        dis_rtc();
        }
}//endmain
```

เรียกใช้ฟังก์ชันจากไฟล์ rtc.h เป็นการแสดงผลของเวลา

เมื่อรันโปรแกรมผลที่ได้จะออกมาดังรูป



รูปที่ ข.3.3 แสดงเวลาขณะนั้นออกมาทางจอ LED 7 เซกเมนต์และเวลาจะเดินไปเรื่อยๆ