ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ โปรแกรมจำลองการทำงานของ MCS-51 และอุปกรณ์ต่อพ่วงผ่านทางพอร์ตอนุกรม
MCS-51 Simulator and Interface Board

ชื่อนักศึกษา

1. นายธีระยุทธ	นาคอนุเคราะห์	รหัสประจำตัว	41031208
2. นายประทีป	หนูรัตน์	รหัสประจำตัว	41031214
3. นางสาวศิรินทร	ตาสาโรจน์	รหัสประจำตัว	41031227
4. นายอมรินทร์	ประสิทธิ์	รหัสประจำตัว	41031236

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** วิศวกรรมโทรคมนาคม

อาจารย์ที่ปรึกษา อาจารย์ไพบูลย์ พวงวงศ์ตระกูล

อาจารย์ที่ปรึกษาร่วม อาจารย์ปิยะ ศุภวราสุวัฒน์

คณะกรรมการสอบปริญญานิพนธ์		ลายมือชื่อ
1. อาจารย์ไพบูลย์	พวงวงศ์ตระกูล	.....
2. อาจารย์ปิยะ	ศุภวราสุวัฒน์	.....
3. ผศ.วิสุทธิ์	อธิพรธรรม	.....
4. อาจารย์พีระวุฒิ	สุวรรณจันทร์	.....
5. อาจารย์อมรชัย	ชัยชนะ	.....

วัน/เดือน/ปีที่สอบ วันศุกร์ที่ 12 พฤษภาคม พ.ศ. 2543 เวลา 14.00 น.

สถานที่สอบ ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่.....เดือน.....พ.ศ.....

ปริญญานิพนธ์

โปรแกรมจำลองการทำงานของไมโครคอนโทรลเลอร์ MCS-51 และอุปกรณ์ต่อพ่วงผ่านทางพอร์ตอนุกรม

MCS-51 SIMULATOR AND INTERFACE BOARD

นายธีระยุทธ นาคอนุเคราะห์

นายประทีป หนูรัตน์

นางสาวศิรินทร ตาสาโรจน์

นายอมรินทร์ ประสิทธิ์

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์

เรื่อง โปรแกรมจำลองการทำงานของไมโครคอนโทรลเลอร์ MCS-51 และอุปกรณ์ต่อพ่วงผ่านทางพอร์ตอนุกรม

Mcs-51 Simulator and Interface Board

วัตถุประสงค์

1. เพื่อศึกษาระบบไมโครคอนโทรลเลอร์เบอร์ 89S53
2. เพื่อออกแบบโปรแกรมจำลองการทำงานของไมโครคอนโทรลเลอร์เบอร์ 89S53
3. เพื่อสร้างโปรแกรมที่ใช้ติดต่อระหว่าง เครื่องคอมพิวเตอร์ และไมโครคอนโทรลเลอร์ผ่านทางพอร์ตอนุกรม
4. เพื่อสร้างโปรแกรมจำลองการทำงานของไมโครคอนโทรลเลอร์ เบอร์ 89S53 และสร้างอุปกรณ์ต่อพ่วงผ่านทางพอร์ตอนุกรม
5. เพื่อทดสอบการจำลองค่าที่เปลี่ยนแปลงหน่วยความจำภายใน และสามารถแสดงผลการทำงานจริง ของอุปกรณ์ต่อพ่วงตามชุดคำสั่งที่เขียนขึ้น
6. เพื่อนำไปใช้ประกอบการศึกษาไมโครคอนโทรลเลอร์ เบอร์ 89S53

ประโยชน์ที่คาดว่าจะได้รับ

1. มีความรู้และทักษะเกี่ยวกับระบบไมโครคอนโทรลเลอร์เบอร์ 89S53
2. มีความรู้ และทักษะเกี่ยวกับ การออกแบบโปรแกรมจำลองการทำงาน ของไมโครคอน - โทรลเลอร์เบอร์ 89S53
3. มีความรู้ และทักษะเกี่ยวกับ การสร้างโปรแกรมที่ติดต่อระหว่าง เครื่องคอมพิวเตอร์และไมโครคอนโทรลเลอร์ ผ่านทางพอร์ตอนุกรม
4. ได้โปรแกรมจำลองการทำงานของ ไมโครคอนโทรลเลอร์เบอร์ 89S53 ตามชุดคำสั่งที่เขียนขึ้น
5. ได้อุปกรณ์ต่อพ่วง และบอร์ดไมโครคอนโทรลเลอร์เบอร์ 89S53 และชุดติดต่อระหว่างคอมพิวเตอร์ และไมโครคอนโทรลเลอร์ผ่านพอร์ตอนุกรม
6. ได้โปรแกรมจำลองการทำงานของไมโครคอนโทรลเลอร์ และอุปกรณ์ต่อพ่วง เพื่อใช้ประกอบการศึกษาไมโครคอนโทรลเลอร์เบอร์ 89S53

ชื่อหัวข้อ	โปรแกรมจำลองการทำงานของ MCS-51 และอุปกรณ์ต่อพ่วงผ่านทางพอร์ตอนุกรม
นักศึกษา	นายธีระยุทธ นาคอนุเคราะห์
	นายประทีป หนูรัตน์
	นางสาวศิรินทร ตาสาโรจน์
	นายอมรินทร์ ประสิทธิ์
อาจารย์ที่ปรึกษา	อาจารย์ไพบูลย์ พวงวงศ์ตระกูล
อาจารย์ที่ปรึกษาร่วม	อาจารย์ปิยะ ศุภวราสุวัฒน์
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชา	วิศวกรรมโทรคมนาคม
ปีการศึกษา	2542

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอการออกแบบและสร้างโปรแกรมจำลองการทำงาน ของไมโครคอนโทรลเลอร์ MCS-51 และอุปกรณ์ต่อพ่วงผ่านทางพอร์ตอนุกรม ชุดทดลองนี้แบ่งเป็น 2 ส่วนคือ ส่วนที่ 1 เป็นส่วนของภาคแสดงผลอุปกรณ์ต่อพ่วงทางพอร์ตอนุกรม ส่วนที่ 2 เป็นส่วนของการจำลองกระบวนการทำงานของภาคแสดงผลจำลองบนคอมพิวเตอร์ ชุดทดลองนี้จะช่วยให้เข้าใจหลักการทำงานของไมโครคอนโทรลเลอร์ MCS-51 และการเขียนโปรแกรมติดต่อใช้งาน MCS-51 ได้ง่ายยิ่งขึ้น สามารถนำไปประยุกต์ใช้งานได้อย่างถูกต้อง

Thesis Title	Mcs-51 Simulator and Interface Board
Students	Mr.Teerayut Nakanukorh
	Mr.Prateep Nurut
	Miss Sirintorn Tasaroj
	Mr.Amarin Pasit
Advisor	Mr.Paiboon Pongwongtragull
Co-Advisor	Mr.Piya Supavarasuwat
Education Level	Bachelor of Science in Industrial Education
Program in	Telecommunication Engineering
Academic Year	1999

ABSTRACT

This thesis presents a designing and implementation of Programmable Microcontroler Mcs-51 Simulation And Interface Board. The Project can be device in to two part ; interface board and level process simulator on computer program. The project helps the user to easier understand the principle of Mcs-51 programmable and can be applied to use.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

การเรียนรู้ที่ดีและมีประสิทธิภาพนั้น ต้องอาศัยปัจจัยหลายอย่าง เพื่อมากระตุ้นประสาทการรับรู้ เช่น มองเห็นภาพการทำงานจำลอง หรือการทำงานจริง ซึ่งจะช่วยให้การเรียนรู้ดีขึ้น และการปฏิบัติจริงๆ หลังจากการศึกษาดูตัวอย่างแล้ว จะช่วยย้ำความรู้ที่รับเข้ามา ในกระบวนการทำงานหลายๆ ด้าน จะสังเกตเห็นว่า ได้มีการนำไมโครคอนโทรลเลอร์เบอร์ 89S53 มาใช้งานทางด้านควบคุมหรืองานอื่นๆ ตามแต่ผู้ใช้ต้องการอยู่หลายๆ งานด้วยกัน ดังนั้น จึงแสดงให้เห็นว่าไมโครคอนโทรลเลอร์มีประโยชน์ และความสำคัญมาก แต่กระบวนการเรียนรู้นอกจากการศึกษาในหนังสือ ตำราเรียนแล้ว สิ่งหนึ่งที่ช่วยให้นักศึกษา มีความเข้าใจมากยิ่งขึ้น ก็คือ การได้ฝึกหัดการเขียนโปรแกรม และทำการทดลองดูผลทั้งบนหน้าจอคอมพิวเตอร์ และผลการทำงานของอุปกรณ์จริงๆ จึงมีความคิดที่จะประดิษฐ์ และสร้างโปรแกรมจำลองการทำงานของหน่วยความจำภายในไมโครคอนโทรลเลอร์ เบอร์ 89S53 และอุปกรณ์ต่อพ่วง เพื่อดูการทำงานจริงขึ้นมา เพื่อตอบสนองต่อความต้องการเรียนรู้ ซึ่งจะทำให้มีความสะดวกในการใช้งานดีขึ้น

1.2 ขีดความสามารถของโครงงาน

1) สามารถจำลองการทำงานของชุดคำสั่งที่เขียนขึ้นได้บนหน้าจอคอมพิวเตอร์

2) สามารถจำลองการทำงานของอุปกรณ์แสดงผล และอุปกรณ์ต่อพ่วงอื่นๆ บนหน้าจอคอมพิวเตอร์

3) สามารถจำลองผลที่เกิดขึ้น ในหน่วยความจำภายในไมโครคอนโทรลเลอร์เบอร์ 89S53 ได้

4) สามารถแสดงการทำงานของอุปกรณ์ต่อพ่วงตามชุดคำสั่งที่เขียนขึ้นได้

5) สามารถติดต่อสื่อสารระหว่าง คอมพิวเตอร์และไมโครคอนโทรลเลอร์เบอร์ 89S53 ผ่านทางพอร์ตอนุกรมได้

6) สามารถแสดงจุดบกพร่องของชุดคำสั่งที่เขียนขึ้นหลังจากคอมไพล์ได้

7) สามารถอ่านคู่มือ (Help) เพื่อช่วยในการใช้โปรแกรมและอุปกรณ์ต่อพ่วงได้

1.3 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้แบ่งออกเป็นบทต่างๆ เพื่อความสะดวกต่อการศึกษาและทำความเข้าใจ ในแต่ละบทจะประกอบด้วยเนื้อหาที่สำคัญดังนี้

บทที่ 2 ทฤษฎีและหลักการประกอบด้วยเนื้อหาในทางทฤษฎีที่เกี่ยวข้องกับวงจร ซึ่งได้แก่ ทฤษฎีเกี่ยวกับไดโอดเปล่งแสง, จอแสดงผลแบบผลึกเหลว, มอเตอร์ไฟฟ้ากระแสตรง, การแสดงผลด้วยจอแสดงผลแบบ 7 ส่วน, สเต็ปปิ้งมอเตอร์, แป้นพิมพ์, ไมโครโปรเซสเซอร์ และไมโครคอนโทรลเลอร์เบอร์ 89S53 และ การสื่อสารแบบอนุกรม

บทที่ 3 การออกแบบการสร้าง และการทำงาน กล่าวถึงการสร้างการออกแบบฮาร์ดแวร์ ซึ่งได้แก่ วงจรแสดงผลแอลอีดีลอจิก, วงจรแสดงผลแอลอีดีดอตเมตริกซ์, วงจรแสดงผลตัวเลข 7 ส่วน, วงจรตรวจจับการกดสวิตช์แบบแถวคูณหลัก, วงจรแสดงผลแบบจอผลึกเหลว, วงจรควบคุมรีเลย์, วงจรมอเตอร์ไฟฟ้ากระแสตรง, วงจรสเต็ปปิ้งมอเตอร์, วงจรขยายสัญญาณเสียงและวงจรตรวจจับทางแสง รวมถึงการสร้างและออกแบบโปรแกรมจำลองการทำงาน MCS-51

บทที่ 4 การทดลองและผลการทดลอง

บทที่ 5 บทสรุป ปัญหา แนวทางแก้ไข และพัฒนา

ภาคผนวก ก เครื่องต้นแบบ

ภาคผนวก ข แผ่นวงจรพิมพ์

ภาคผนวก ค คู่มือการใช้โปรแกรม

ภาคผนวก ง ใบงาน

ภาคผนวก จ เฉลยใบงาน

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

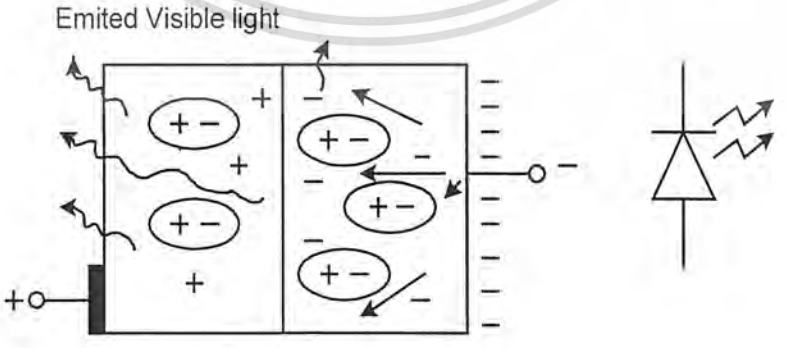
เนื้อหาของในบทนี้เป็นทฤษฎี และหลักการที่นำมาใช้ประกอบการสร้างโครงงาน โดยประกอบด้วยทฤษฎีเกี่ยวข้องกับวงจรแต่ละวงจรที่ใช้ในการสร้างโครงงาน ดังต่อไปนี้

2.2 อุปกรณ์แสดงผลทางแสง

2.2.1 ไดโอดเปล่งแสง

ไดโอดเปล่งแสงเป็นไดโอดที่สามารถเปล่งแสงสว่างออกมาด้วยคลื่นความถี่เดียว และมีเฟสต่อเนื่องได้ ซึ่งต่างไปจากแสงธรรมดาที่คนเรามองเห็น ประกอบไปด้วยคลื่นที่มีความถี่และเฟสต่างๆ มารวมกัน

โครงสร้างของไดโอดเปล่งแสงเหมือนกับไดโอดทั่วๆ ไปที่ประกอบขึ้นมาจากการเอาสารพี และสารเอ็นมาประกบกัน โดยผิวข้างหน้าเป็นมันคล้ายกระจก เมื่อเราให้ฟอร์เวิร์ดแก่ไดโอดเปล่งแสงจะทำให้อิเล็กตรอนที่สารกึ่งตัวนำชนิดเอ็นมีพลังงานสูงขึ้น จนสามารถวิ่งข้ามรอยต่อไปรวมกับโฮลในสารพีได้ ก่อให้เกิดพลังงานที่เราเรียกกันว่าพลังงาน “โฟตอน” เปล่งออกมา โครงสร้าง และสัญญลักษณ์ของไดโอดเปล่งแสง แสดงดังรูปที่ 2.1



รูปที่ 2.1 โครงสร้างและสัญลักษณ์ของไดโอดเปล่งแสง

2.2.2 จอแสดงผลแบบผลึกเหลว

จอแสดงผลแบบผลึกเหลวซึ่งใช้สำหรับวงจรแสดงผลระบบตัวเลขในเครื่องใช้อิเล็กทรอนิกส์ ในปัจจุบันซึ่งอยู่ในภาคแสดงผลของเครื่องคิดเลข, นาฬิกา, หน้าปัดเครื่องรับวิทยุ-โทรทัศน์, เครื่องมือวัดและทดสอบระบบดิจิตอล, วงจรแสดงผลทางดิจิตอล ฯลฯ

ผลึกเหลวเป็นตัวหลักของจอแสดงผลแบบผลึกเหลวนี้ เป็นสสารชนิดหนึ่งซึ่งมีสภาพการไหลเหมือนอย่างของเหลว แต่การรวมตัวของโมเลกุล จะรวมตัวกันแบบโมเลกุลของของแข็ง ซึ่งการรวมตัวแบบนี้เราเรียกว่า การรวมตัวแบบนีแมติกลิควิดคริสตอล

จอแสดงผลแบบผลึกเหลว ที่ใช้อยู่ทั่วไปในปัจจุบันมีอยู่ 2 ชนิด คือ ชนิดไดนามิคสแกทเตอร์ริ่ง (Dynamic Scattering) กับชนิดฟิลด์เอฟเฟ็ค (Field-effect) จอแสดงผลแบบผลึกเหลวชนิดหลังนิยมใช้มากกว่าชนิดแรก เพราะกินกระแสเพียง 20 เปอร์เซนต์ ของชนิดแรก และยังสามารถให้แสงได้ชัดแม้ในขณะที่แสงสว่างจากภายนอกจะมาก

2.2.3 จอแสดงผลแบบผลึกเหลวแบบดอตเมตริกซ์

1) ชนิดของจอแสดงผลแบบผลึกเหลว สามารถแบ่งได้เป็น

1.1) คุณสมบัติ (Character LCD Module)

1.2) กราฟฟิก (Graphic LCD Module)

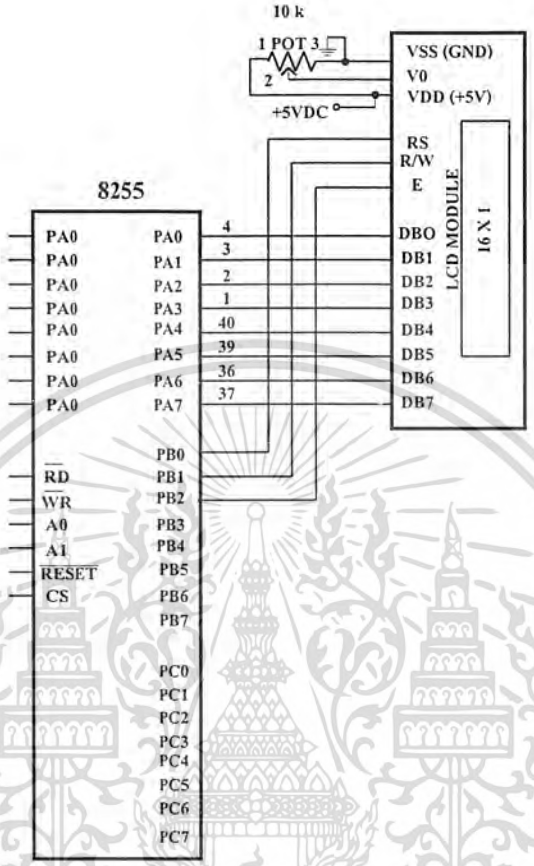
1.3) ส่วนแสดงผล (Segment Display Type LCD Module)

2) มีส่วนประกอบใหญ่ๆ ของจอแสดงผลแบบผลึกเหลว แบ่งได้เป็น

2.1) ดอตเมตริกซ์เป็นตัวแสดงผลให้เรามองเห็นในลักษณะการปิด และเปิดตัวเองกับแสง นั่นคือ ส่วนที่เป็นตัวกระจกบรรจุผลึก

2.2) ตัวขับ เป็นตัวรับสัญญาณจากส่วนควบคุม เพื่อใช้ขับผลึกจอแสดงผลแบบผลึกเหลว

2.3) ตัวควบคุม เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอก และจัดการควบคุมจอแสดงผลแบบผลึกเหลวโมดูล ให้ทำงานแสดงผลต่างๆ เช่น การลงจอภาพ, การเกิดตัวอักษร, การเลื่อนข้อมูลบนหน้าจอแสดงผลแบบผลึกเหลว เป็นต้น โดยมีเบอร์ไอซีที่นิยมใช้ คือ HD44780 ซึ่งใช้ในแบบคุณสมบัติ คือจอแสดงผลแบบผลึกเหลวที่แสดงข้อมูลในรูปตัวอักษร ส่วนเบอร์ HD61830 ซึ่งจะใช้ในแบบกราฟฟิก คือจอแสดงผลแบบผลึกเหลว ที่แสดงข้อมูลในลักษณะรูปภาพ หรือตัวอักษรก็ได้



รูปที่ 2.2 การต่อใช้งานของจอแสดงผลแบบผลึกเหลว

จากวงจรดังรูปที่ 2.2 เป็นการต่อ 8255 ให้เข้าใช้กับจอแสดงผลแบบผลึกเหลว โดยเจำลองสัญญาณต่างๆ ขึ้นมาโดยการใช้พอร์ต A และพอร์ต B โดยพอร์ต A นั้นให้เป็นพอร์ตข้อมูล และพอร์ต B นั้นให้เป็นสัญญาณควบคุม เมื่อเริ่มเปิดไฟป้อนให้ HD44780 นั้นก็จะทำการรีเซตตัวเอง โดยจะให้เวลาประมาณ 10 มิลลิวินาที หลังจากไฟ VDD ถึง 4.5 โวลต์แล้ว โดยเซตตัวเองดังนี้

1) DISPLAY CLEAR จะทำการลบข้อมูลจอภาพแสดงผลแบบผลึกเหลว

2) FUNCTION SET โดยจะเซตค่าภายใน

DL = 1: เป็นการรีเซตให้การติดต่อแบบ 8 บิต

N = 0 : เซตเป็น 1 บรรทัดการแสดงผล

F = 0 : 5X7 จุดต่อหนึ่งต่ออักษร

3) DISPLAY ON/OFF

D = 0 : DISPLAY OFF

C = 0 : CURSOR OFF

B = 0 : BLINK OFF

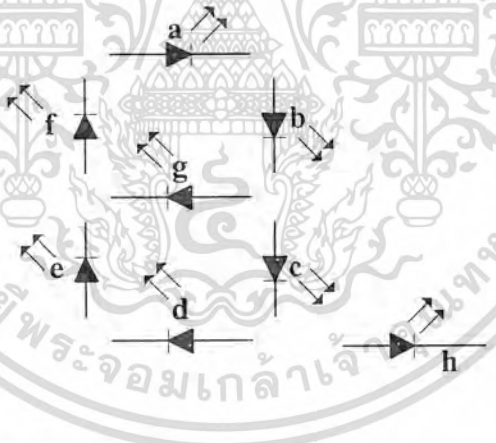
4) ENTRY MODE SET I/D = 1 : + 1 (เพิ่มค่าตัวเองโดยนับขึ้น 1)

S = 0 : NO SHIFT

2.2.4 จอแสดงผลแบบ 7 ส่วน

จอแสดงผลแบบ 7 ส่วน ประกอบด้วยไดโอดเปล่งแสงต่อกันเป็นรูปเลข 8 มีขาร่วม 1 ขา มีทั้งคาโถดร่วม และอาโนดร่วม

ไดโอดเปล่งแสง เป็นอุปกรณ์ที่ใช้แสดงผลของเอาต์พุตอย่างง่ายๆ อย่างหนึ่ง และไดโอดเปล่งแสงแบบ 7 ส่วนจะใช้สำหรับแสดงตัวเลขฐานสิบ 0 ถึง 9 หรือแสดงเลขฐานสิบหก 0 ถึง F การกำหนดรูปแบบของตัวเลข และชื่อของแต่ละส่วน แสดงดังรูปที่ 2.3



รูปที่ 2.3 ชื่อตำแหน่งของไดโอดเปล่งแสงในจอแสดงผลแบบ 7 ส่วน

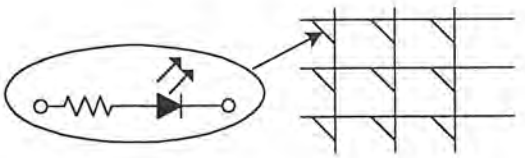
ถ้ากำหนดให้การแสดงผลแบบ 7 ส่วนนี้เป็นแบบคาโถดร่วม และถ้าให้พอร์ตเอาต์พุตบิต D0 ต่ออยู่กับส่วน a บิต D1 ต่ออยู่กับส่วน b ตามลำดับ เมื่อบิตใดมีระดับเป็น 1 จะทำให้ส่วนนั้นติด ดังนั้นการแสดงตัวเลข สามารถเขียนเป็นรหัสเลขฐานสิบหกได้ วิธีการคือเมื่อต้องการให้การแสดงผลแบบ 7 ส่วนติดเป็นเลขใด ก็ส่งรูปแบบนั้นออกทางพอร์ตเอาต์พุต ซึ่งแสดงในตารางที่ 2.1

ตารางที่ 2.1 รูปแบบของรหัสเพื่อการแสดงผลตัวเลขแบบ 7 ส่วน

เลข	D7 .	D6 g	D5 f	D4 e	D3 d	D2 c	D1 b	D0 a	รหัส
0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	0	1	1	0	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F
9	0	1	1	0	1	1	1	1	6F
A	0	1	1	1	0	1	1	1	77
B	0	1	1	1	1	1	0	0	7C
C	0	0	1	1	1	0	0	1	39
D	0	1	0	1	1	1	1	0	5E
E	0	1	1	1	1	0	0	1	79
F	0	1	1	1	0	0	0	1	71

2.2.5 ดอตเมตริกซ์

ดอตเมตริกซ์ เป็นการแสดงผลแบบแถวเดียวโดยควบคุมสัญญาณเพียงด้านเดียว คือ ส่งแต่ข้อมูลแล้วหน่วงเวลาไว้ระยะหนึ่ง แล้วจึงส่งสัญญาณตัวต่อไป ทำให้เห็นเป็นไฟวิ่งได้ สามารถเห็นเป็นตัวอักษรหรือรูปภาพได้ เพิ่มลอจิกการควบคุมที่จะควบคุมการติดดับเพียงด้านเดียว เป็น 2 ด้าน ดังรูปที่ 2.4



รูปที่ 2.4 การต่อไดโอดเปล่งแสงแบบเมตริกซ์

จากรูปที่ 2.4 ถ้าส่งข้อมูลออกไปทางแถวและให้คอลัมน์ใดเป็น 0 จะทำให้ไดโอดเปล่งแสงถูกไบอัสตรง และติดสว่างตามบิตข้อมูลที่ส่งออกไปทางแถว จากหลักการนี้ จึงสามารถควบคุมไดโอดเปล่งแสงทุกดวงในแผงเมตริกซ์ ให้ติดหรือดับได้ตามต้องการ

2.2.6 หลักการเกิดภาพบนไดโอดเปล่งแสงแบบเมตริกซ์

เนื่องจากการมองเห็นของนัยตาของคนนั้น จะมีการคงภาพคือเห็นภาพนั้นอยู่บน 1/6 วินาที ถึงแม้ว่าภาพนั้นจะไม่ปรากฏแล้วก็ตาม จากหลักการของการมองเห็นนี้ ถ้าทำให้ไดโอดเปล่งแสงติดและดับตามจุดต่างๆ ได้เร็วกว่า 1/16 จะเห็นว่าไดโอดเปล่งแสงที่ตำแหน่งนั้นติดค้างอยู่

แต่เนื่องจากซีพียู ทำงานด้วยความเร็วสูง การที่จะไดโอดเปล่งแสงติดและเปลี่ยนตำแหน่งการติดว่างไปเรื่อยๆ ตามความเร็วของซีพียู แล้วจะทำให้ไม่ปรากฏรูปภาพ แต่จะเห็นว่า ไดโอดเปล่งแสงนั้นติดสว่างเรืองๆ ทุกดวง เหตุที่เป็นเช่นนี้เพราะว่า ความสัมพันธ์ของกระแสความอิ่มตัวของทรานซิสเตอร์ รวมทั้งกระแสที่ทำให้ไดโอดเปล่งแสงติดสว่าง เป็นปัจจัยทำให้เห็นไดโอดเปล่งแสงติดทุกดวง ด้วยเหตุนี้เมื่อทำให้ไดโอดเปล่งแสงดวงใดติดแล้ว ควรทำการหน่วงเวลาเพื่อให้ มีกระแสเพียงพอที่จะทำให้ไดโอดเปล่งแสงติดสว่าง ยิ่งหน่วงเวลานานเท่าไร ยิ่งทำให้กระแสเข้าใกล้จุดสูงสุด แต่เนื่องจากต้องทำการกวาดหลายๆ คอลัมน์ถ้าทำการหน่วงเวลามากเกินไป จะทำให้ไดโอดเปล่งแสงในลำดับถัดไปมีวงรอบในการนำกระแสเป็นช่วงๆ ซึ่งระยะเวลาในการนำกระแสของไดโอดเปล่งแสงแต่ละดวง ห่างไกลจากความคงแสงของตามนุษย์ ซึ่งมีผลทำให้เห็นภาพกระพริบหรือพลิ้ว ในการกวาดแบบคอลัมน์นี้เวลาที่จะทำให้ไดโอดเปล่งแสง แต่ละดวงติดนานเท่าไรนั้นขึ้นอยู่กับจำนวนคอลัมน์ของไดโอดเปล่งแสงด้วย

2.2.7 ปัญหาในการกวาดทางคอลัมน์

ถ้าจำนวนคอลัมน์มีมาก เวลาในการนำกระแสของไดโอดเปล่งแสง แต่ละคอลัมน์มีน้อยตามไปด้วยทำให้เกิดการพริ้ว

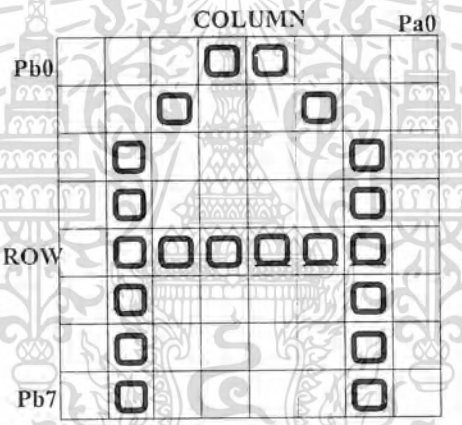
ดังนั้นสามารถแก้ไขได้โดยเมื่อจำนวนคอลัมน์มากเกินไป ควรเปลี่ยนมาเป็นการกวาดทางแถวแทน ซึ่งจะทำให้เวลาของไดโอดเปล่งแสงแต่ละดวงที่จะติด มีเวลาคงที่มากยิ่งขึ้น และความสว่างคงที่มากขึ้น

2.2.8 การคิดตัวอักษร

การคิดข้อมูลที่จะนำมาแสดงเป็นตัวอักษร มิได้ถูกกำหนดเป็นมาตรฐานหรือเป็นทฤษฎี แต่ขึ้นอยู่กับ ฮาร์ดแวร์นั้นๆ เมื่อศึกษาวงจรแสดง ขนาด 8x8 ตัว จะพบว่ากำหนดให้พอร์ต A ของ 8255 เป็นพอร์ตสำหรับจ่ายแหล่งจ่ายกระแส และทางพอร์ต B เป็นตัวรับกระแสซิงค์

ฉะนั้นข้อมูลที่ทำให้ไดโอดเปล่งแสงติดได้ คือ ข้อมูลที่ออกมาจากพอร์ต A ซึ่งต้องเป็น Hi และข้อมูลทางพอร์ต B ต้องเป็น Low แต่เนื่องจากทางพอร์ต B มีทรานซิสเตอร์อยู่ 1 ตัว เป็นตัวรับกระแสซิงค์ จึงต้องคิดข้อมูลเป็นลอจิกบวก คือ 1 = ไดโอดเปล่งแสงติด

ตัวอย่างข้อมูลที่จะแสดง ตัวอักษร A บนแผงแสดงผลไดโอดเปล่งแสง โดยให้ข้อมูลที่หานั้นออกทางพอร์ต B และทำการควาดทางพอร์ต A ดังรูปที่ 2.5

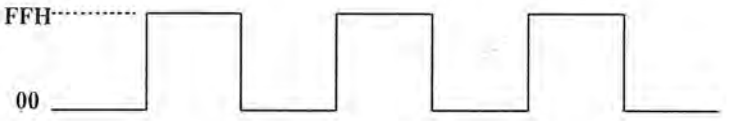


รูปที่ 2.5 ตัวอักษร A บนแผงแสดงผลไดโอดเปล่งแสง

2.3 การขยายออกทางลำโพง

หลักการกำเนิดเสียงโดยใช้ไมโครโปรเซสเซอร์ ก็สามารถทำได้โดยการส่งเอาต์พุตให้มีการสลับไปมาอย่างรวดเร็วออกไปที่ลำโพง จนเกิดเป็นความถี่เสียงขึ้นมา การกำเนินเสียงสามารถแยกระดับของสัญญาณได้ถึง 255 ระดับ ซึ่งทำให้มีความละเอียดของเสียงมากยิ่งขึ้น

การส่งให้มีการสลับไปมานี้ ก็คือ การส่ง “0” กับ “1” สลับกันนั่นเอง ในกรณีของอีทีบอร์ด เราจะมีการส่งข้อมูลสลับกันระหว่าง “0” กับ “FFH” ซึ่งหมายถึงระดับสูงสุดที่มีได้ ผลที่ได้บนลำโพง แสดงดังรูปที่ 2.6



รูปที่ 2.6 ระดับผลที่ได้จากลำโพง

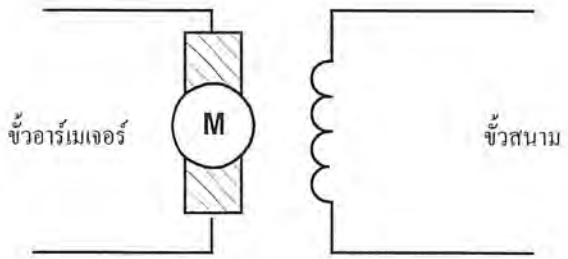
จากรูปที่ 2.6 สังเกตว่า ถ้าสามารถกำหนดช่วงเวลาของการเกิดเสียง (T) ได้ คือเสามารถกำหนดความถี่ของเสียงได้นั่นเอง โดย T = I/F จึงสามารถทำให้ค่า T มีค่าน้อยมากๆ ได้พอที่จะครอบคลุมช่วงความถี่ที่หูคนเราจะได้ยินได้อย่างสบาย (หูคนเราจะได้ยินเสียงที่ช่วงความถี่ประมาณ 20 เฮิรตซ์ ถึง 20 กิโลเฮิรตซ์) แต่อย่างไรก็ตาม เสียงตามตัวโน้ต มีความถี่ที่กำหนดไว้แน่นอนตายตัวแล้ว

เสียงที่เกิดขึ้นจากการสังเคราะห์ทางโปรแกรมนี้ คงไม่สามารถทำให้นุ่มนวลเหมือนเสียงธรรมชาติได้ เพราะรูปคลื่นที่เกิดจากโปรแกรมทำขึ้นนั้น จะเป็นรูปคลื่นสี่เหลี่ยม แต่การสังเคราะห์มีประโยชน์หลายด้านด้วยกัน เช่น เป็นสัญญาณเตือน หรือเป็นสัญญาณที่สื่อความหมายใดๆ หรือเป็นตัวแปลงข้อมูลให้อยู่ในรูปของเสียง

2.4 มอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรงนั้นสามารถจำแนกออกไปได้อีกหลายประเภท ซึ่งขึ้นอยู่กับวิธีการสร้าง ที่รู้จักกันเป็นส่วนใหญ่ในปัจจุบัน คือมอเตอร์ไฟฟ้ากระแสตรงแบบขนาน, แบบอนุกรม, แบบผสมและแบบแม่เหล็กถาวร ลักษณะโครงสร้างของมอเตอร์ชนิดต่างๆ มีดังนี้

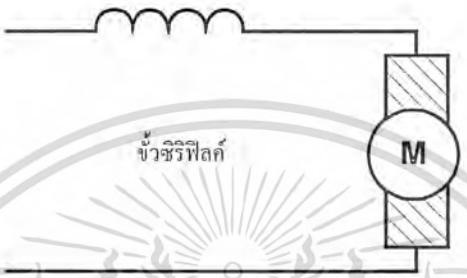
2.4.1 ลักษณะโครงสร้างของมอเตอร์แบบขนาน



รูปที่ 2.7 ลักษณะโครงสร้างของมอเตอร์แบบขนาน

จากรูปที่ 2.7 มอเตอร์แบบนี้สามารถปรับเส้นแรงแม่เหล็กได้อย่างอิสระต่อกระแสของอาร์เมเจอร์ เป็นผลให้สามารถควบคุมพารามิเตอร์ ให้มีค่าคงที่ได้ตลอดช่วงพิสัยที่กว้าง มอเตอร์ชนิดนี้จักมักใช้ในงานระบบควบคุมการเคลื่อนที่ที่ต้องการแรงบิดสูง

2.4.2 ลักษณะโครงสร้างของมอเตอร์แบบอนุกรม



รูปที่ 2.8 ลักษณะโครงสร้างของมอเตอร์แบบอนุกรม

จากรูปที่ 2.8 มอเตอร์แบบนี้ จะมีเส้นแรงแม่เหล็กเป็นสัดส่วนกับกระแส ดังนั้นเส้นแรงของสนามแม่เหล็กจึงสามารถปรับค่าได้ ซึ่งเราจะได้ความสัมพันธ์ระหว่างความเร็ว และแรงบิดไม่เป็นเชิงเส้น จึงเหมาะที่จะนำไปใช้งานในภาวะเฉพาะคือ เมื่อต้องการแรงบิดสูงที่ความเร็วต่ำ และแรงบิดต่ำที่ความเร็วสูง เช่น ระบบการขับเคลื่อนของรถลาก

2.4.3 ลักษณะโครงสร้างของมอเตอร์แบบแม่เหล็กถาวร

รูปที่ 2.9 ลักษณะโครงสร้างของมอเตอร์แบบแม่เหล็กถาวร

จากรูปที่ 2.9 มอเตอร์แบบนี้จะใช้การกระตุ้นฟิลด์ของมอเตอร์ เป็นแม่เหล็กถาวร ซึ่งต่างจากที่กล่าวมาข้างต้นที่ใช้ขดลวดแบบนี้ จะใช้เส้นแรงของฟิลด์มีค่าคงที่ อัตราส่วนระหว่างกระแสอาร์เมเจอร์ และแรงบิดจะมีค่าคงที่ด้วย ซึ่งมีข้อดีคือไม่มีกำลังสูญเสียในฟิลด์ มีประสิทธิภาพสูงกว่า

และมีขนาดเล็กกว่าเมื่อเทียบกับมอเตอร์แบบใช้ขดลวดในการกระตุ้น ที่มีขนาดกำลังม้าเท่ากัน จึงเหมาะกับการที่ต้องการแรงบิดของโหลดสูง

2.5 สเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์ ถือว่าเป็นอุปกรณ์เอาต์พุตอย่างหนึ่ง ซึ่งสามารถควบคุมได้ด้วยไมโครคอมพิวเตอร์ ลักษณะการทำงานของสเต็ปปิ้งมอเตอร์ จะเคลื่อนที่เป็นขั้นซึ่งอาจเป็นขั้นละ 18, 5, 7.5 องศา แล้วแต่ชนิดของมอเตอร์ ส่วนใหญ่สเต็ปปิ้งมอเตอร์จะใช้ในงานควบคุมระบบดิจิตอล เช่น อุปกรณ์ประกอบคอมพิวเตอร์ต่างๆ เช่น พรินซ์เตอร์, X-Y พล็อตเตอร์, ดิสก์ไดรว์ตลอดจนอุปกรณ์ในระบบงานอิเล็กทรอนิกส์อุตสาหกรรม หรือเครื่องมือวัด และระบบควบคุมอื่นๆ

1) สเต็ปปิ้งมอเตอร์ มีส่วนประกอบสำคัญ 2 ส่วนคือ

1.1) โรเตอร์ (Rotor ส่วนที่หมุนได้) จะเป็นแม่เหล็กถาวร หรืออื่นๆ

1.2) สเตเตอร์ (Stator ส่วนที่อยู่กับที่) เป็นขดลวดหลายๆ ขด

2) สเต็ปปิ้งมอเตอร์ ในปัจจุบัน แบ่งได้ 3 แบบคือ

2.1) แบบแม่เหล็กถาวร (PM = Permanent Magnet)

2.2) แบบแปรค่ารีลักแตนซ์ (VR = Variable Reluctance)

2.3) แบบลูกผสม (H = Hybrid)

2.6 แป้นพิมพ์

แป้นพิมพ์นับเป็นสิ่งสำคัญอย่างหนึ่ง เพราะแป้นพิมพ์เป็นอุปกรณ์อินพุตที่ทำให้ผู้ใช้สามารถติดต่อกับเครื่องควบคุมในการทำงานต่างๆ ได้

1) แป้นพิมพ์ ประกอบด้วยส่วนสำคัญ 3 ส่วนคือ

1.1) ส่วนของสวิตซ์แป้นพิมพ์

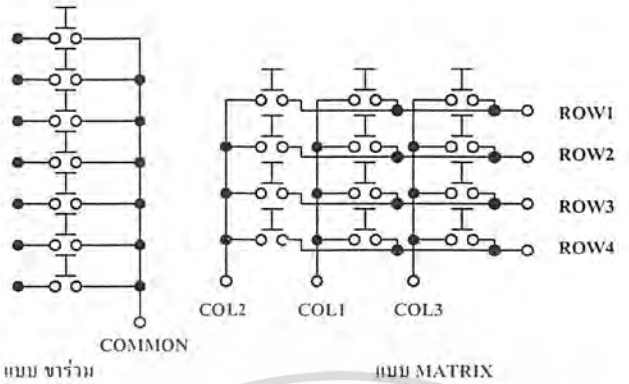
1.2) ส่วนของวงจรเข้ารหัส โดย ส่วนนี้ทำหน้าที่เข้ารหัสเพื่อรับรู้ตำแหน่งตัวอักษรบนแป้น และสามารถให้ค่าตัวอักษรบนแป้นของแต่ละตัว เมื่อมีการกดตัวอักษรบนแป้น

1.3) ส่วนของวงจรถอดรหัส ทำหน้าที่เปลี่ยนรหัสตัวอักษรบนแป้น ให้เป็นรหัสที่นำไปใช้งานได้ เช่น รหัส ASCII, รหัส BCD หรือ รหัส HEX เป็นต้น

2) วงจรแป้นพิมพ์ แบ่งได้เป็น 2 ลักษณะใหญ่ๆ คือ

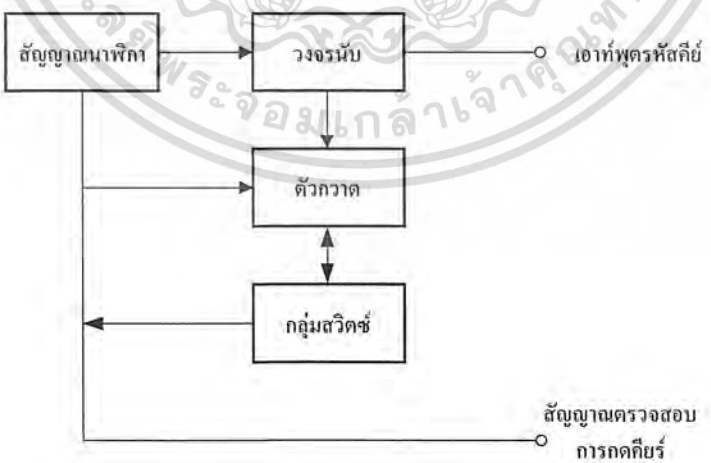
2.1) แบบขาร่วม

2.2) แบบเมตริกซ์ (Matrix)



รูปที่ 2.10 วงจรแป้นพิมพ์แบบขาร่วมและแบบเมตริกซ์

จากรูปที่ 2.10 จะเห็นว่าแป้นพิมพ์แบบขาร่วม จะเป็นแบบที่เข้าใจง่าย และการรับรู้การกดตัวอักษรบนแป้นจะเป็นไปในลักษณะเฉพาะตัว ของแต่ละตัวอักษรบนแป้นซึ่งทำให้ง่ายต่อการออกแบบ สะดวกและประหยัดในกรณีที่สวิตช์ไม่มากนัก ส่วนแบบเมตริกซ์ จะใช้สายในการต่อเข้ากับวงจรเข้ารหัสได้น้อยกว่า โดยให้ระบุจำนวนจุดของตัวอักษรบนแป้น ได้จำนวนมากซึ่งจำนวนตัวอักษรบนแป้นขึ้นอยู่กับสายทางด้านแถว และสายทางด้านคอลัมน์ แต่การออกแบบวงจรเข้ารหัสจะยุ่งยากขึ้น โดยสามารถแสดงแผนผังการทำงานของวงจรเข้ารหัสแป้นพิมพ์ ได้ดังรูปที่ 2.11

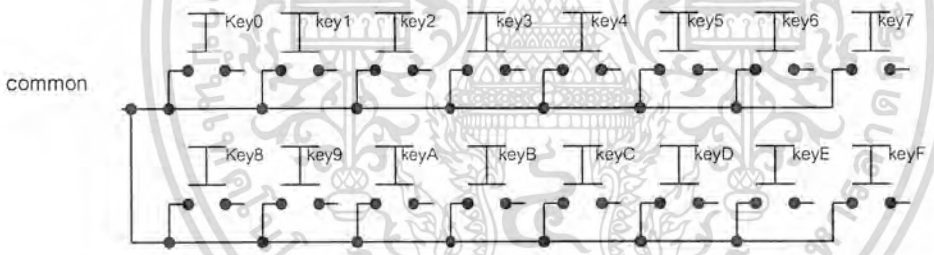


รูปที่ 2.11 แผนผังการทำงานของวงจรเข้ารหัสแป้นพิมพ์

จากรูปที่ 2.11 สัญญาณนาฬิกาใช้ตรวจว่า ตัวอักษรบนแป้นใดถูกกด โดยเมื่อไม่มีการกดตัวอักษรบนแป้น ตัวกวาด จะส่งสัญญาณกวาดไปที่สวิตช์ และเมื่อมีการกดตัวอักษรบนแป้น สัญญาณนาฬิกาจะหยุดการทำงาน ทำให้ตัวกวาดหยุดการกวาด วงจรนับจะหยุดนับ และคงค่าสุดท้ายเอาไว้ จนกว่าจะมีการปล่อย จึงจะรับค่าแป้นพิมพ์ตัวใหม่ได้ ซึ่งลักษณะการกวาดแบบเมตริกซ์นี้ จะเริ่มกวาดจากแถวแรกไปยังแถวสุดท้าย และคอลัมน์แรกไปยังคอลัมน์สุดท้ายพร้อมกัน ดังนั้น เมื่อตัวอักษรบนแป้นใดถูกกด จะทำให้ทราบตัวอักษรบนแป้นที่ถูกที่ถูกกดว่า อยู่แถว และคอลัมน์ใดได้จากส่วนของวงจรนับ

2.6.1 องค์ประกอบอันเกิดจากสวิตช์ตัวอักษรบนแป้น

ตัวอักษรบนแป้นโดยทั่วไป หน้าสัมผัสจะมีการสั่นทั้งขณะเปิด และขณะปิด ซึ่งมีเวลาเป็นมิลลิวินาที เมื่อมีการกดปล่อยตัวอักษรบนแป้นอย่างรวดเร็วจะต้องตรวจสอบได้ด้วยไมโครคอนโทรลเลอร์ ตัวอักษรบนแป้นอาจทำงานเพียงสัมผัส หรือสร้างสัญญาณการกดตัวอักษรบนแป้นด้วยอาร์เอสฟลิปฟลอป หรือซอฟท์แวร์ รูปที่ 2.12 เป็นตัวอย่างแป้นพิมพ์แบบไฮโพธีติคอล



รูปที่ 2.12 ลักษณะแป้นพิมพ์ไฮโพธีติคอล

2.6.2 โปรแกรมสำหรับแป้นพิมพ์

โปรแกรมที่ดีจะต้องทำให้การทำงานของมนุษย์ผ่านทางตัวอักษรบนแป้นบอร์ด ดังนี้

1) สำหรับตัวอักษรบนแป้นหลายๆ ตัว ตัวอักษรบนแป้นที่ได้รับการกดก่อนเท่านั้น จะได้รับการอ่านค่าตัวอักษรบนแป้น

2) การค้างของตัวอักษรบนแป้นตัวอักษรบนแป้นที่กดก่อน จะได้รับการอ่านค่า หลังจับเวลาการกดตัวอักษรบนแป้น จะไม่มีตัวอักษรบนแป้นใดได้รับการอ่านค่าจนกว่าจะพบ ว่าตัวอักษรบนแป้นทั้งหมดกลับสู่สภาวะเดิม

3) การกดปล่อยตัวอักษรบนแป้นอย่างรวดเร็ว ในการออกแบบตัวอักษรบนแป้น จะได้รับการกวาดด้วยอัตราที่สูงกว่าปฏิกิริยาการกระทำของมนุษย์ ลักษณะโปรแกรมที่สำหรับอ่านค่าตัวอักษรบนแป้น มีอยู่ 2 ลักษณะคือ

3.1) การหน่วงเวลาโดยใช้โปรแกรมลูป เพื่อรอรับค่าตัวอักษรบนแป้น

3.2) การอ่านจะมีขึ้นก็ต่อเมื่อมีการกดตัวอักษรบนแป้นเท่านั้น

2.6.3 การอ่านข้อมูลจากสวิตซ์จำนวนมาก

การต่อสวิตซ์แบบปกติจะใช้ 1 บิตต่อสวิตซ์ 1 ตัว แต่ในการใช้งานจริงของระบบไมโครคอมพิวเตอร์ ต้องการใช้สวิตซ์หรือตัวอักษรบนแป้นเป็นจำนวนมาก ดังนั้นถ้าใช้ 1 สวิตซ์ต่อ 1 บิต จะทำให้สิ้นเปลืองพอร์ต วิธีการต่อสวิตซ์แบบประหยัดพอร์ต และเป็นที่นิยมใช้แบบหนึ่งคือ การต่อแบบเมตริกซ์ ซึ่งจะทำให้ลดจำนวนพอร์ตลงถ้าพิจารณาจำนวนของสวิตซ์ และจำนวนของสาย

ตารางที่ 2.2 เปรียบเทียบจำนวนของสายเมื่อต่อแบบธรรมดาและเมตริกซ์

จำนวนสวิตซ์	จำนวนสาย	
	ต่อแบบธรรมดา	ต่อแบบเมตริกซ์
1x1	1	1
2x2	4	4
3x3	9	6
4x4	16	8
5x5	25	10
8x8	64	16

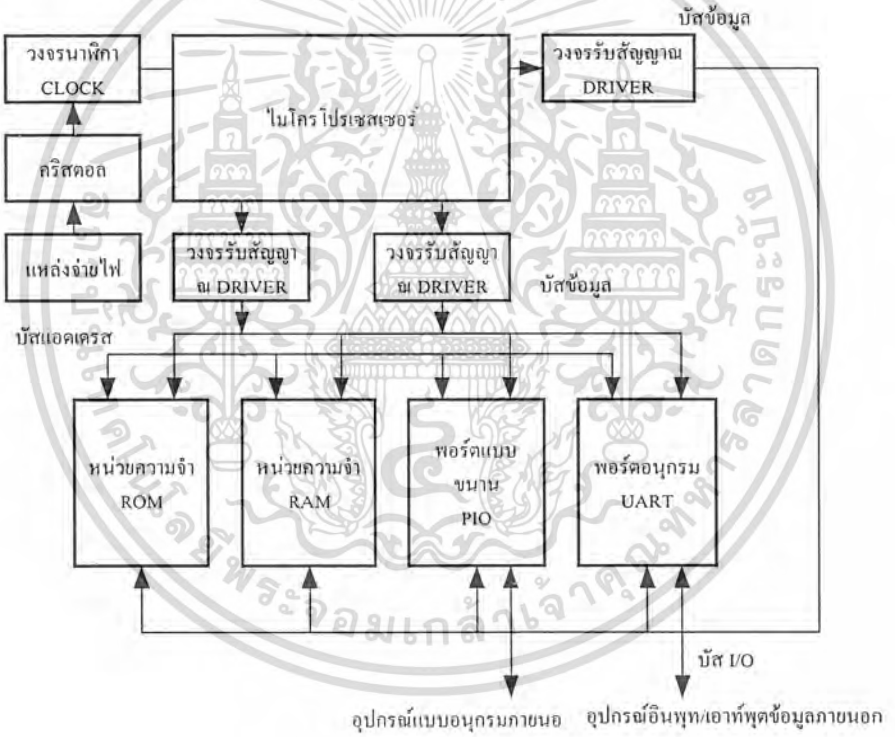
จากตารางที่ 2.2 เห็นได้ชัดว่าเมื่อจำนวนของสวิตซ์มากขึ้น จะทำให้ประหยัดสายข้อมูลมากขึ้นตามไปด้วย ข้อยุ่งยากทางด้านฮาร์ดแวร์จะลดลงและการตรวจสอบสวิตซ์ที่กดจะใช้วิธีการทางซอฟท์แวร์เพื่อจัดการ

2.7 ไมโครโปรเซสเซอร์และไมโครคอนโทรลเลอร์ตระกูล 8051

เป็นตัวประมวลผลกลางของระบบเมื่อมีการนำไปใช้งาน จำเป็นจะต้องมีไอซีประกอบภายนอกเพิ่มเข้าไปให้ระบบที่สมบูรณ์ เช่น หน่วยความจำ และพอร์ตควบคุมโดยองค์ประกอบของ

ไมโครโปรเซสเซอร์ เป็นไอซีภายนอกที่จะต้องเพิ่มเติมเข้าไปให้ครบตามหน้าที่ที่ต้องการ สำหรับไมโครคอนโทรลเลอร์มีหลักการพื้นฐานเช่นเดียวกัน เพียงแต่องค์ประกอบการทำงานหลายส่วนได้รับการออกแบบให้บรรจุอยู่ภายในไอซี เพียงตัวเดียวเท่านั้น

เมื่อเปรียบเทียบกับไมโครโปรเซสเซอร์แล้ว หน่วยการทำงานภายในไมโครคอนโทรลเลอร์นั้นสามารถใช้งานได้จำกัดมากกว่า เช่น มีหน่วยความจำแบบรอม หรือ อีพรอมภายในไม่เกิน 4 กิโลไบต์, หน่วยความจำแบบแรมขนาด 256 ไบต์เท่านั้น และมีพอร์ตแบบขนานประมาณ 3 ถึง 4 พอร์ตเป็นต้น แผนผังการทำงานทั่วไปของระบบไมโครโปรเซสเซอร์แสดงในรูปที่ 2.13



รูปที่ 2.13 แผนผังการทำงานทั่วไปของระบบไมโครโปรเซสเซอร์

2.7.1 ไมโครคอนโทรลเลอร์ตระกูล 8051

ไมโครคอนโทรลเลอร์จากบริษัท อินเทลในตระกูล MCS-51 ได้มีการนำไปใช้งานแพร่หลายมากนับตั้งแต่ปี ค.ศ. 1980 ในระยะที่ผ่านมา ได้มีอีกหลายบริษัท เช่นบริษัทฟิลลิปส์ และซีเมนส์ ได้รับลิขสิทธิ์ในการผลิต,จำหน่าย และได้มีการเพิ่มประสิทธิภาพของหน่วยการทำงานต่างๆ มากขึ้น

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ประกอบด้วย ไมโครคอนโทรลเลอร์หลายรุ่นซึ่งมีสถาปัตยกรรมพื้นฐานที่เหมือนกัน เพียงแต่มีขนาดหรือจำนวนของหน่วยทำงานภายในที่แตกต่างกันเพื่อความเหมาะสมในงานประยุกต์ต่างๆ ตามความต้องการ มีทั้งลักษณะที่ใช้เทคโนโลยีการผลิตไอซีวงจรรวมความจุสูงมากแบบ HMOS หรือ CHMOS สิ้นเปลืองกำลังไฟฟ้าน้อยกว่ามาก การอ้างถึงไมโครคอนโทรลเลอร์ MCS-51 จะเรียกรวมกันว่า 8051 แทน

2.7.2 คุณลักษณะพื้นฐานของ 8051

หน่วยการทำงานพื้นฐานของไมโครคอนโทรลเลอร์เบอร์ต่างๆ ที่จัดอยู่ภายในตระกูล MCS-51 ประกอบด้วย

1) หน่วยประมวลผลกลางขนาด 8 บิต

2) หน่วยประมวลผลสำหรับข้อมูลแบบบิต

3) ความสามารถในการอ้างตำแหน่งของหน่วยความจำโปรแกรม 64 กิโลไบต์

4) ความสามารถในการอ้างตำแหน่งของหน่วยความจำข้อมูล 64 กิโลไบต์

5) หน่วยความจำโปรแกรมภายใน 4 กิโลไบต์ แบบอีพรอม (เบอร์ 8751) หรือแบบรอม (เบอร์ 8051)

6) หน่วยความจำแบบ แรมภายในจำนวน 128 ไบต์

7) พอร์ตอินพุต/เอาต์พุต แบบขนานจำนวน 32 เส้น ซึ่งสามารถแยกทำงานได้อย่างอิสระ

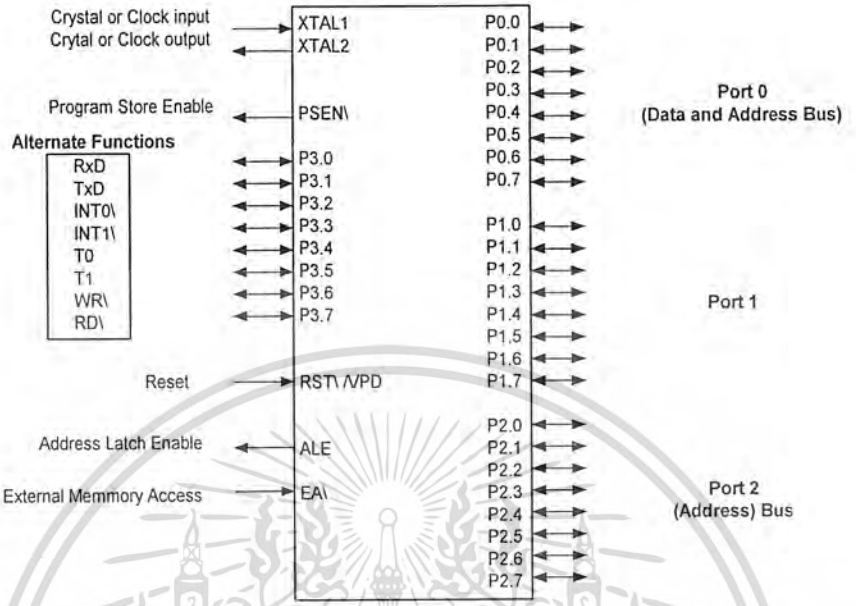
8) วงจรนับ/จับเวลา ขนาน 16 บิต จำนวน 2 วงจร

9) วงจรสื่อสารแบบอนุกรมแบบฟูลดูเพล็กซ์

10) วงจรควบคุมการอินเตอร์รัพต์ จากตำแหน่งกำเนิดสัญญาณ 6 ประเภทพร้อมการกำหนดลำดับความสำคัญได้ 2 ระดับ

11) วงจรออสซิเลเตอร์ภายใน

โดยมากไมโครคอนโทรลเลอร์ตระกูลนี้จะเป็นไอซีแบบดิป ขนาด 40 ขา ดังรูปที่ 2.14 ซึ่งแต่ละขาสัญญาณ จะมีหน้าที่ระบุชัดเจนตามสัญลักษณ์ชื่อย่อที่กำกับในแต่ละขา จะมีบางขาสัญญาณ ที่อาจจะมีหน้าที่ได้มากกว่า 1 อย่าง ซึ่งจะไม่สามารถใช้งานในเวลาเดียวกันได้ ตัวอย่างเช่น ขาสัญญาณบิต 0 ของพอร์ต 3 (ใช้ตัวย่อว่า P3.0) อาจจะใช้เป็นขาสัญญาณเอาต์พุต/อินพุตตามปกติ หรืออาจทำหน้าที่เป็นขาสัญญาณอินพุตของข้อมูลสื่อสารแบบอนุกรม ให้กับวงจรสื่อสารแบบอนุกรมของ 8051 ซึ่งการกำหนดว่าจะทำงานในลักษณะใด ขึ้นอยู่กับการเชื่อมต่อวงจรเข้ากับขาสัญญาณ และโปรแกรมควบคุมของระบบนั้น



รูปที่ 2.14 การกำหนดหน้าที่ขาสัญญาณของ ไอซี 8051

2.7.3 ฐานเวลาในการทำงานของ CPU ภายใน 8051

8051 มีวงจรคริสตอลอยู่ภายใน สำหรับการสร้างพัลส์ของสัญญาณนาฬิกา ซึ่งจะนำไปเป็นฐานเวลาหรือการกำหนดจังหวะการทำงานของหน่วยการทำงานทั้งหมด ให้สอดคล้องกันโดยปกติจะกำหนดการใช้คริสตอลเชื่อมต่อเข้ากับขาสัญญาณ XTAL1 และ XTAL2 พร้อมกับตัวเก็บประจุ หรือ อาจเป็นสัญญาณนาฬิกาจากภายนอกก็ได้

พัลส์ความถี่ของสัญญาณนาฬิกาจะเรียกว่าพัลส์ (ใช้สัญลักษณ์ เป็นตัวอักษร P) และคาบของสัญญาณนาฬิกานี้เรียกว่า คาบเวลาออสซิเลเตอร์ จำนวน 2 คาบ เรียกว่าสเตท ใช้สัญลักษณ์เป็นตัวอักษร S ซึ่งนำไปใช้เป็นช่วงเวลาพื้นฐานการทำงานย่อยของไมโครคอนโทรลเลอร์ เช่น การนำคำสั่ง (Fetch), การถอดความหมาย (Decode), การประมวลผล (Execute) และการเขียนข้อมูล (Write) เป็นต้น ช่วงเวลาของสเตทจำนวน 6 ครั้ง เรียกว่า แมชชีนไซเคิล ดังนั้นค่าหนึ่งแมชชีนไซเคิลจะใช้เวลา 12 คาบเวลา ออสซิเลเตอร์ค่าของแมชชีนไซเคิลนี้จัดว่าเป็นช่วงเวลาที่น้อยที่สุดในการทำคำสั่งใดคำสั่งหนึ่ง หากเป็นคำสั่งที่ซับซ้อนมากจะต้องใช้เวลานาน 2 ถึง 3 แมชชีนไซเคิล

การคำนวณหาว่าเวลาที่ใช้ในการทำคำสั่งใดจนเสร็จสิ้น จะต้องดูว่าคำสั่งนั้นใช้จำนวนแมชชีนไซเคิลเท่าไรในการประมวลผล เวลาที่ใช้ คำนวณได้ตามสูตรที่ 2.1

$$T = \frac{C \times 12}{Cystal\ Frequency} \qquad (2.1)$$

โดย C เป็นค่าจำนวนแมชชีนไซเคิลของคำสั่ง

Crystal Frequency เป็นค่าจำนวนของคริสตอลที่ใช้กับ 8051

2.8 มาตรฐานพอร์ตอนุกรมแบบ RS-232

มาตรฐานการเชื่อมต่อแบบอนุกรม RS-232 เป็นมาตรฐานอุตสาหกรรมที่ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทาง โดยมาตรฐาน RS-232 ในอดีตนั้นถูกออกแบบมาเพื่อการส่งผ่านข้อมูลคอมพิวเตอร์ ไปยังโมเด็มเพียงอย่างเดียว เพื่อที่จะนำข้อมูลจากโมเด็มนี้ส่งผ่านสายโทรศัพท์ไปยังคอมพิวเตอร์อีกชุดซึ่งอยู่ห่างไกลกัน โดยสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association :EIA) ได้วางมาตรฐานที่มีชื่อเรียกว่า EIA RS-232 มาตรฐานนี้ในช่วงแรกจะใช้คอนเน็กเตอร์เป็นแบบ DB-25 โดยกำหนดความยาวสูงสุดของสายสัญญาณไว้ที่ 50 ฟุต มีระดับสัญญาณตั้งแต่ –3V จนถึง –12V แสดงว่า มีข้อมูล และ +3V ถึง +12V แสดงว่าเป็น ช่องว่าง

มาตรฐาน RS-232 ถูกใช้ในการกำหนดรูปแบบการสื่อสารข้อมูลกันระหว่างอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment:DTE) กับวงจรรับข้อมูลปลายทาง (Data Circuit Termination : DCE) อุปกรณ์เชื่อมต่อข้อมูล จะต้องเป็นอุปกรณ์ที่มีการประมวลผลในตัวเช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลแบบอนุกรมได้ ส่วนอุปกรณ์รับข้อมูล ทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาเท่านั้น

สำหรับการใช้งานในคอมพิวเตอร์ พอร์ตอนุกรม RS-232 ถูกใช้เพื่อเชื่อมต่อกับโมเด็ม, เมาส์ และเครื่องพิมพ์ ที่มีสามารถติดต่อทางพอร์ตอนุกรมได้

2.8.1 คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้ หรือ DB-9 ตัวผู้ ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งาน เพียง 9 เส้นเช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่นๆ ไม่มีความสำคัญมากนัก หน้าที่ของขาต่างๆ มีดังนี้

1) ขา Data Carrier Delete : DCD หรืออาจเรียกว่า Carrier Detect : CD ขานี้จะแอกตีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งาน

2) ขา Receive Data : RD หรือ RxD ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยจะนำข้อมูลที่ผ่านได้ไปเก็บไว้ในรีจิสเตอร์บัฟเฟอร์

3) ขา Transmitted Data : TD หรือ TxD ขานี้ใช้เพื่อส่งข้อมูลอนุกรมออกจากคอมพิวเตอร์ โดยการนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป

4) ขา Data Terminal Ready : DTR เป็นขาเอาต์พุต ที่ใช้สำหรับส่งสัญญาณออกจากคอมพิวเตอร์ เพื่อให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่ออุปกรณ์ปลายทางโดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ และถ้าใช้การเชื่อมต่อแบบ 3 สาย ต้องเชื่อมต่อขา DTR และ DSR ของพอร์ตอนุกรมเข้าด้วยกัน และต้องต่อเชื่อมเข้ากับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์

5) ขา Signal Ground : GND เป็นขากราวด์ของสัญญาณ

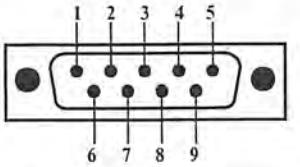
6) ขา Data Set Ready : DSR ขานี้จะใช้ควบคู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอก

7) ขา Request To Send : RTS เป็นขาเอาต์พุตสำหรับส่งข้อมูลร้องขอ ให้อุปกรณ์ปลายทางส่งข้อมูลมาให้คอมพิวเตอร์โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ซึ่งในกรณีที่มีการเชื่อมต่อแบบ 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS เข้าด้วยกัน เพื่อให้การรับ และส่งข้อมูลสามารถเกิดขึ้นได้

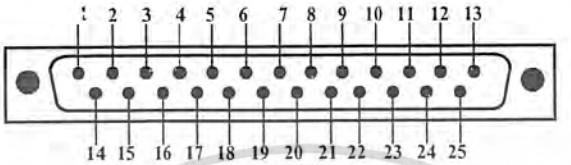
8) ขา Clear To Send : CTS เป็นขาอินพุตทำหน้าที่รอรับสัญญาณที่ส่งเข้ามา เมื่อมีการส่งสัญญาณเข้ามาที่ขานี้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ขานี้จะใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลแล้วหรือยัง

9) ขา Ring Indicator : RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานต่อเมื่อมีการเชื่อมต่อกับโมเด็มแล้ว ยังมีความต้องการตรวจสอบสัญญาณเรียกจากสายโทรศัพท์

คอนเน็กเตอร์อนุกรม 9 ขา หรือ DB-9 และแบบ 25 ขา หรือ DB-25 แสดงในรูปที่ 2.15 ส่วนการจัดขาสัญญาณของพอร์ตอนุกรมในแบบต่างๆ และหน้าที่การทำงานแสดงดังตารางที่ 2.3



(ก) คอนเน็กเตอร์อนุกรม 9 ขาหรือแบบ DB-9 (มองจากด้านหลังคอมพิวเตอร์)



(ข) คอนเน็กเตอร์อนุกรม 25 ขาหรือแบบ DB-9 (มองจากด้านหลังคอมพิวเตอร์)

รูปที่ 2.15 คอนเน็กเตอร์อนุกรม

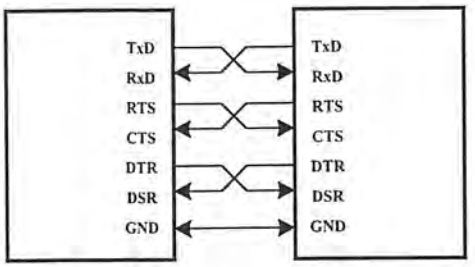
ตารางที่ 2.3 การจัดขาสัญญาณของพอร์ตอนุกรมในแบบต่างๆ และหน้าที่การทำงาน

คอนเน็กเตอร์ DB-9	คอนเน็กเตอร์ DB-25	ชื่อของสายสัญญาณ	ชนิดของสายสัญญาณ
1	8	Data Carrier Dectec : DCD	อินพุต
2	3	Received Data : RxD	อินพุต
3	2	Transmitted Data : TxD	เอาต์พุต
4	20	Data Terminal Ready : DTR	เอาต์พุต
5	7	Signal Ground : GND	-
6	5	Data Set Ready : DSR	อินพุต
7	4	Request To Send : RTS	เอาต์พุต
8	5	Clear to Send : CTS	อินพุต
9	22	Ring indicator : Rt	อินพุต

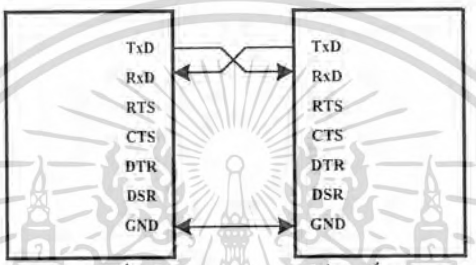
2.8.2 พอร์ตอนุกรม (UART)

UART มาจากคำว่า Universal Asynchronous Receiver Transmitter ซึ่งหมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส ถือว่าเป็นหัวใจสำคัญของการสื่อสารอนุกรม

การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม ในรูปแบบต่างๆ แสดงในรูปที่ 2.16



(ก) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์แบบ Null Modem



(ข) การต่ออุปกรณ์ภายนอกเข้ากับอุปกรณ์คอมพิวเตอร์แบบ RS-232
ในลักษณะที่ใช้สายสัญญาณน้อยที่สุดเพียง 3 เส้น

รูปที่ 2.16 การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในรูปแบบต่าง ๆ

หน้าที่หลักของ UART คือแปลงข้อมูลที่อยู่ในรูปแบบขนานจากซีพียู ให้อยู่ในรูปแบบอนุกรมแบบอะซิงโครนัส แล้วส่งทำการส่งออกไป และแปลงสัญญาณอนุกรมแบบอะซิงโครนัสที่ป้อนเข้ามายัง UART ให้เป็นแบบขนานก่อนที่จะส่งเข้าสู่ซีพียู ซึ่งนอกจาก UART จะส่งข้อมูลไปยังซีพียูแล้ว ยังแจ้งรายละเอียดอื่นๆ ของข้อมูลให้คอมพิวเตอร์ รับทราบด้วย อาทิ อัตราเร็วในการรับส่งข้อมูลหรือบอดเรต, รูปแบบการส่งข้อมูล, ความผิดพลาดที่เกิดขึ้นระหว่างการส่งข้อมูล เช่น ผิดพลาดจากพาริตี้, เฟรมข้อมูล, โอเวอร์รัน เป็นต้น

ภายใน UART มีวงจรสร้างบอดเรตโปรแกรมได้ โดยการกำหนดค่าตัวหารให้กับสัญญาณนาฬิกาของ UART โดยตัวหารนี้จะมีขนาด 16 บิต สามารถกำหนดตัวหารอยู่ในช่วง 1–65,535

1) UART ที่ใช้ในคอมพิวเตอร์

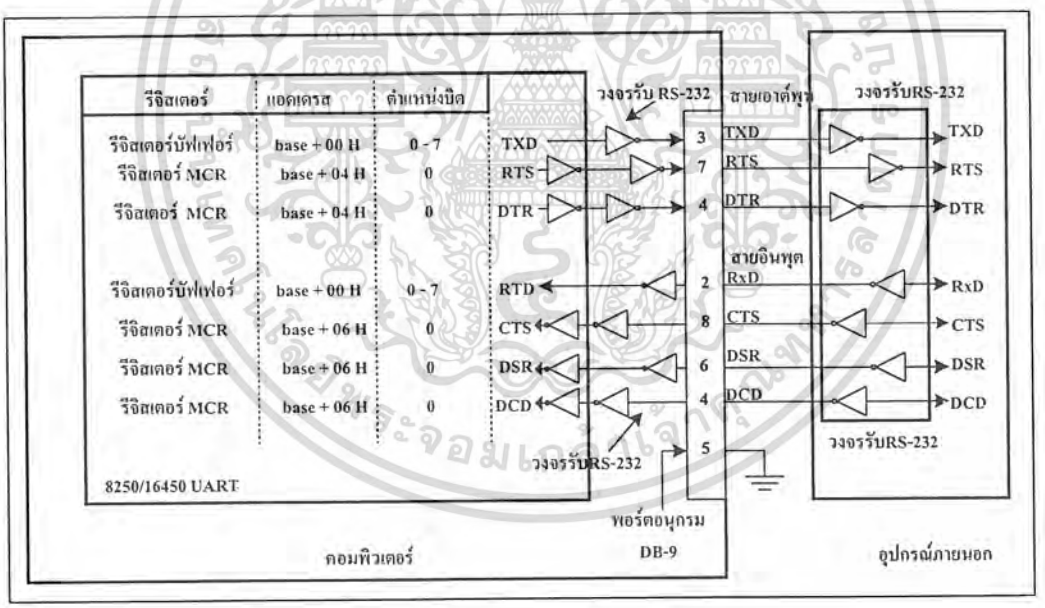
ในเครื่องคอมพิวเตอร์ทั่วไปจะมี UART ที่ใช้งานกันอยู่ 2 เบอร์ คือ เบอร์ 8250 และ 16550 สำหรับ UART เบอร์ 8250 เป็น UART มาตรฐานที่มีมานาน UART มีบัฟเฟอร์สำหรับรับและส่งข้อมูลเป็นตำแหน่งเดียวกัน ทำให้การรับ และส่งข้อมูลถูกจำกัดความเร็วอยู่ที่ 57.6 กิโลบิตต่อวินาที สำหรับ UART เบอร์ 16550 จะเพิ่มส่วนของชิฟต์รีจิสเตอร์แบบ FIFO (First In First Out)

ขนาด 16 ไบต์เข้าไป ทำให้สามารถสนับสนุนความเร็วในการรับส่งข้อมูลที่ระดับ 256 กิโลบิตต่อวินาทีได้

2.8.3 ลักษณะสัญญาณอินพุตและเอาต์พุตของพอร์ต RS-232

สัญญาณเอาต์พุตที่ใช้ควบคุม ได้แก่ RTS และ DTR รวมทั้งสัญญาณแสดงสถานะอินพุต (CTS, DSR และDCD) จะถูกกลับสถานะภายในตัว UART ส่วนสัญญาณข้อมูลทั้งภาคส่ง และรับจะไม่ถูกกลับสถานะและ UART จะให้ระดับสัญญาณเอาต์พุตออกมาเป็นแบบทีทีแอล ดังนั้นเมื่อสัญญาณถูกส่งออกมาจาก UART จะส่งเข้าสู่วงจรขับเพื่อแปลงระดับสัญญาณให้เป็นไปตามมาตรฐาน RS-232 ก่อนส่งออกจากคอมพิวเตอร์

สำหรับอุปกรณ์ต่อเชื่อมปลายทางจะต้องมีวงจรขับในลักษณะนี้ เพื่อให้ได้สัญญาณในระดับเดียวกันวงจรขับที่ใช้ทั้งภายในคอมพิวเตอร์และอุปกรณ์ต่อเชื่อมปลายทางนั้นจะถูกกลับสถานะ แผนผังการทำงานภายในของขาสัญญาณต่างๆ ของพอร์ตอนุกรม แสดงดังรูปที่ 2.17



รูปที่ 2.17 แผนผังการทำงานภายในของขาสัญญาณต่างๆ ของพอร์ตอนุกรม

ตารางที่ 2.4 รายละเอียดการแจ้งจำนวนพอร์ตอนุกรมของหน่วยความจำตำแหน่ง 0000:0411

บิต 3	บิต 2	บิต 1	จำนวนพอร์ต
0	0	0	ไม่มีพอร์ตอนุกรม
0	0	1	มีพอร์ตอนุกรม 1 พอร์ต
0	1	0	มีพอร์ตอนุกรม 2 พอร์ต
0	1	1	มีพอร์ตอนุกรม 3 พอร์ต
1	0	0	มีพอร์ตอนุกรม 4 พอร์ต

ตำแหน่งแอดเดรสพื้นฐานของพอร์ตอนุกรมมีตำแหน่งดังนี้คือ

COM 1 มีแอดเดรสพื้นฐาน (Base) อยู่ที่ 3F8H

COM 2 มีแอดเดรสพื้นฐาน (Base) อยู่ที่ 2F8H

COM 3 มีแอดเดรสพื้นฐาน (Base) อยู่ที่ 3E8H

COM 4 มีแอดเดรสพื้นฐาน (Base) อยู่ที่ 2E8H

เมื่อเริ่มเปิดเครื่องเพื่อใช้งานคอมพิวเตอร์ ไบออสจะทำการตรวจสอบตำแหน่งแอดเดรสของพอร์ต อนุกรมทั้งหมด ถ้าไบออสตรวจพบแอดเดรสของพอร์ตอนุกรม ไบออสจะนำแอดเดรสที่ตรวจพบไปเก็บไว้ในหน่วยความจำขนาด 2 ไบต์ สำหรับ COM1 จะเก็บไว้ที่แอดเดรสตำแหน่ง 0000:0400H และ 0000:0401H ส่วนตำแหน่งอื่นๆ มีรายละเอียดดังนี้

COM 2 เก็บไว้ที่ 0000:0402H – 0000:0403H

COM 3 เก็บไว้ที่ 0000:0404H – 0000:0405H

COM 4 เก็บไว้ที่ 0000:0406H – 0000:0407H

นอกจากนั้นแอดเดรสของหน่วยความจำที่ตำแหน่ง 0000:0411H ยังใช้เพื่อแสดงจำนวนของพอร์ตอนุกรม ที่มีใช้อยู่ในคอมพิวเตอร์อีกด้วย โดยมีรายละเอียดแสดงในตารางที่ 2.4

2.8.4 วงจรภายในและรีจิสเตอร์ของพอร์ตอนุกรม RS-232

เครื่องคอมพิวเตอร์ PC โดยทั่วไปสามารถต่อพอร์ตอนุกรม RS-232 สูงสุดได้ 4 พอร์ต ซึ่งจะมีชื่อเรียกเป็น COM1, COM2 , COM3 และ COM4 ซึ่งพอร์ตอนุกรมแต่ละตัวจะใช้ UART ภายในคอมพิวเตอร์ในการติดต่อกับอุปกรณ์ภายนอก

ในรูปที่ 2.17 แสดงบล็อกไดอะแกรมของวงจรภายในของพอร์ตอนุกรม ประกอบด้วยรีจิสเตอร์ขนาด 8 บิต 8 ตัว ที่ใช้งานร่วมกับ UART ตำแหน่งแอดเดรสของรีจิสเตอร์ภายในพอร์ตอนุกรม สามารถคำนวณได้จากค่ารีจิสเตอร์พื้นฐานของพอร์ตอนุกรมตัวอย่างเช่น พอร์ตอนุกรม

COM1 มีแอดเดรสพื้นฐานอยู่ที่ 3F8H ตำแหน่งของรีจิสเตอร์ต่างๆ จะเป็นตำแหน่งที่บวกเข้าไปกับค่า 3F8H โดยรีจิสเตอร์ที่ใช้งานกับพอร์ตอนุกรม มีดังนี้

1) รีจิสเตอร์ 00H

เป็นรีจิสเตอร์บัฟเฟอร์สำหรับเก็บข้อมูลที่รับเข้ามาหรือเตรียมข้อมูลก่อน ที่จะส่งออกไป มีแอดเดรสอยู่ที่ Base+00H ถ้าเป็น COM1 จะมีแอดเดรสอยู่ที่ 3F8H การติดต่อกับรีจิสเตอร์นี้เพื่อเก็บข้อมูลที่ต้องการจะส่งจะต้องกำหนดให้บิต DLAB ในรีจิสเตอร์กำหนดรูปแบบข้อมูล (03H) มีสถานะเป็น "0" ซึ่งการเขียนข้อมูลไปยังแอดเดรสนี้ เป็นการส่งข้อมูลไปยังรีจิสเตอร์และข้อมูลจะถูกส่งออกไปแบบอนุกรมในกรณีรับข้อมูล เมื่อรับข้อมูลเข้ามาและแปลงเป็นแบบขนานแล้ว ข้อมูลแบบขนานจะถูกส่งไปยังรีจิสเตอร์เก็บข้อมูล หลังจากมีการอ่านค่าจากรีจิสเตอร์นี้แล้ว รีจิสเตอร์นี้จะถูกเคลียร์เพื่อให้พร้อมสำหรับการรับข้อมูลในไบต์ต่อๆ ไป

2) รีจิสเตอร์ 01H

เป็นรีจิสเตอร์อีนาเบิ้ลการอินเตอร์รัพต์ ซึ่งจะใช้เพื่อเซตโหมดในการอินเตอร์รัพต์ของพอร์ตอนุกรม ซึ่งเป็นการกำหนดให้ UART สร้างสัญญาณอินเตอร์รัปต์ขึ้นมา มีแอดเดรสอยู่ที่ Base+01H ถ้าเป็น COM 1 จะมีแอดเดรสอยู่ที่ 3F9H รายละเอียดในแต่ละบิตของรีจิสเตอร์ 01H มีดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	SINP	ERBK	TBE	RxRD

บิต 4-7	บิตเหล่านี้ไม่ถูกใช้งาน มีสถานะเป็น "0"
SINP	"1" เกิดการอินเตอร์รัพต์เมื่อมีการเปลี่ยนสถานะที่ขาอินพุต CTS, DSR, DCD หรือ ขา RI "0" ไม่มีการใช้อินเตอร์รัพต์รูปแบบนี้
ERBK	"1" เกิดการอินเตอร์รัพต์เมื่อเกิดความผิดพลาดขึ้นจากพาริตี้, โอเวอร์รัน, เฟรมข้อมูล หรือการหยุดข้อมูล "0" ไม่มีการใช้อินเตอร์รัพต์รูปแบบนี้
TBE	"1" เกิดการอินเตอร์รัพต์เมื่อรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง "0" ไม่มีการใช้อินเตอร์รัพต์รูปแบบนี้
RxRD	"1" เกิดอินเตอร์รัพต์เมื่อข้อมูล 1 ไบต์ถูกเก็บลงในรีจิสเตอร์บัฟเฟอร์แล้ว "0" ไม่มีการใช้อินเตอร์รัพต์รูปแบบนี้

3) รีจิสเตอร์ 02H

เป็นรีจิสเตอร์แสดงโหมดการอินเตอร์รัพท์ ใช้เพื่อตรวจสอบโหมดของการอินเตอร์รัพต์เมื่อมีการอินเตอร์รัพต์เกิดขึ้น Base+02H โดยมีรายละเอียดของแต่ละบิตดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	0	0	ID1	ID0	PND

บิต 3-7 ไม่ได้ใช้งาน กำหนดเป็น "0"

ID1 , ID0 "00" เกิดการอินเตอร์รัพต์เนื่องจากการเปลี่ยนแปลงของขาอินพุต (มีนัยสำคัญเป็นอันดับ 4 หรือนัยสำคัญต่ำสุด)

"01" เกิดการอินเตอร์รัพต์เนื่องจากรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง (มีนัยสำคัญอันดับ 3)

"10" เกิดการอินเตอร์รัพต์เนื่องจากข้อมูลเก็บลงในรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลแล้ว (มีนัยสำคัญอันดับ 2)

"11" เกิดการอินเตอร์รัพต์เนื่องจากความผิดพลาด ในการส่งข้อมูลหรือเกิดการหยุดกระทันหัน (มีนัยสำคัญเป็นอันดับ 1 หรือนัยสำคัญสูงสุด)

เมื่อมีการสร้างสัญญาณอินเตอร์รัพต์ขึ้น จะต้องมีการเคลียร์ค่าก่อนที่จะให้เกิดอินเตอร์รัพต์ตัวต่อไป โดยสามารถทำได้ดังนี้

3.1) กรณีเกิดอินเตอร์รัพต์เนื่องจากการเปลี่ยนแปลง ของขาอินพุต (ID1 = 0 , ID0 = 0) จะต้องอ่านค่าจากรีจิสเตอร์แสดงสถานะของโมเด็มหรือรีจิสเตอร์ 06H เพื่อเคลียร์ค่าการอินเตอร์รัพปต์

3.2) กรณีเกิดการอินเตอร์รัพต์เนื่องจากบัฟเฟอร์สำหรับส่งข้อมูลระหว่าง (ID1 = 0 , ID0 = 1) จะต้องเขียนข้อมูลไปยังรีจิสเตอร์บัฟเฟอร์ส่งข้อมูล (รีจิสเตอร์ 00H) หรืออ่านค่ารีจิสเตอร์แสดงสถานะอินเตอร์รัพต์ (รีจิสเตอร์ 02H) เพื่อเคลียร์ค่าการอินเตอร์รัพต์

3.3) กรณีเกิดอินเตอร์รัพต์เนื่องจากเก็บข้อมูลลงในรีจิสเตอร์บัฟเฟอร์สำหรับข้อมูล เรียบร้อย สามารถเคลียร์ค่าอินเตอร์รัพต์โดยการอ่านข้อมูลจากรีจิสเตอร์บัฟเฟอร์ สำหรับเก็บข้อมูล (รีจิสเตอร์ 00H)

3.4) กรณีเกิดอินเตอร์รัพต์เนื่องจากความผิดพลาดในการรับส่งข้อมูล หรือเกิดการหยุดกระทันหัน จะต้องเคลียร์ค่าอินเตอร์รัพต์ โดยการอ่านค่ารีจิสเตอร์ แสดงสถานะการรับและส่งข้อมูล แบบอนุกรม (รีจิสเตอร์ 05H)

PND "1" ไม่มีการอินเตอร์รัพต์

"0" มีการอินเตอร์รัพต์เกิดขึ้น

4) รีจิสเตอร์ 03H

เป็นรีจิสเตอร์กำหนดรูปแบบของข้อมูล มีรายละเอียดของแต่ละบิตดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
DLAB	BRK	PAR2	PAR1	PAR0	STOP	DAB1	DAB0

DLAB "1" เข้าสู่โหมดการหารค่าบอดเรต

"0" เป็นการเข้าถึงรีจิสเตอร์บัฟเฟอร์ สำหรับเก็บข้อมูลที่รับเข้ามา และส่งออกไป (รีจิสเตอร์ 00H) และรีจิสเตอร์อีนาเบิ้ลอินเตอร์รัพต์ (รีจิสเตอร์ 01H)

เมื่อบิต DLAB มีสถานะเป็น "1" นั้น รีจิสเตอร์บัฟเฟอร์สำหรับเก็บข้อมูลที่รับเข้ามาและส่งออกไป (รีจิสเตอร์ 00H) และรีจิสเตอร์สำหรับอีนาเบิ้ลอินเตอร์รัพป์ (รีจิสเตอร์ 01H) จะถูกใช้สำหรับการโหลดค่าการหารความถี่สำหรับกำหนดค่าบอดเรต โดยรีจิสเตอร์ 00H ใช้ในการกำหนดค่าการหารในไบต์ต่ำ ส่วนรีจิสเตอร์ 01H ใช้ในการกำหนดค่าการหารในไบต์สูง การหาค่าบอดเรตสามารถเขียนเป็นสมการได้ดังนี้

ค่าบอดเรต = 115,200 / ค่าตัวหาร 16 บิต

ค่าตัวเลข 115,200 มาจากค่าความถี่ของคริสตอลที่ใช้อยู่ในวงจร UART ภายในคอมพิวเตอร์ โดย คริสตอลที่ใช้มีความถี่ 1.8432 MHz จากนั้นภายใน UART จะทำการหารค่าความถี่นี้ด้วย 16 ทำให้ได้ค่าความถี่ 115,200 Hz ออกมา

ค่าตัวหาร 16 บิต = ข้อมูลในรีจิสเตอร์ 00H + (256 x ข้อมูลในรีจิสเตอร์ 01H)

ยกตัวอย่าง ต้องการค่าบอดเรทเท่ากับ 9,600 บิตต่อวินาที ค่าตัวหารที่ใช้จะต้องมีค่าเท่ากับ 12 ซึ่งค่านี้จะต้องถูกโหลดลงในรีจิสเตอร์ 00H และต้องกำหนดค่า 0 ให้แก่รีจิสเตอร์ 01H ค่าตัวหารต่ำสุดที่สามารถเกิดขึ้นได้ นั่นคือ รีจิสเตอร์ 00H มีค่าเท่ากับ "1" และรีจิสเตอร์ 01H มีค่าเท่ากับ "0" ทำให้ได้ค่าบอดเรตสูงสุดที่ 115,200

BRK "1" กำหนดให้สามารถหยุดการรับส่งข้อมูลกระทันหันได้

"0" ไม่มีการเบรก

PAR2 , PAR1 , PAR 0 ใช้กำหนดบิตพาริตี้

000 – ไม่ใช่บิตพาริตี้

001 – กำหนดพาริตี้คี่

011 – กำหนดพาริตี้คู่

101 – มาร์ก (mark)

111 – ช่องว่าง (space)

STOP "1" มีบิตปิดท้าย 2 บิต

"0" มีบิตปิดท้าย 1 บิต

DAB1 , DAB0 ใช้ในการกำหนดจำนวนบิตของข้อมูล

00 – จำนวนบิตข้อมูลเท่ากับ 5 บิต

01 – จำนวนบิตข้อมูลเท่ากับ 6 บิต

10 – จำนวนบิตข้อมูลเท่ากับ 7 บิต

11 – จำนวนบิตข้อมูลเท่ากับ 8 บิต

5) รีจิสเตอร์ 04H

เป็นรีจิสเตอร์ควบคุมโมเด็ม ใช้ตรวจสอบบิตสำหรับการติดต่อกับโมเด็ม เช่น RTS หรือ DTR มีแอดเดรสอยู่ที่ Base+04H มีรายละเอียดหน้าที่ของแต่ละบิตดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	0	0	LOOP	OUT2	OUT1	RTS	DTR

บิต 5-7 ไม่ได้ใช้งาน กำหนดเป็น "0"

LOOP "1" เป็นการอีนาเบิลการส่งค่ากลับ

"0" เป็นการดิสเอเบิล

OUT2 , OUT1 "1" เป็นการอีนาเบิลการใช้งานภายใน

"0" เป็นการดิสเอเบิล

RTS "1" เป็นการอีนาเบิลขา RTS สำหรับคอนเน็กเตอร์ของพอร์ตอนุกรม RS-232

"0" เป็นการดินเอเบิล

DTR "1" เป็นการอีนาเบิลขา DTR สำหรับคอนเน็กเตอร์ของพอร์ตอนุกรม RS-232

"0" เป็นการดิสเอเบิล

6) รีจิสเตอร์ 05H

เป็นรีจิสเตอร์แสดงสถานะการรับและส่งข้อมูลแบบอนุกรมของ UART ซึ่งจะใช้งานร่วมกับรีจิสเตอร์สำหรับแสดงโหมด และสถานะของการอินเตอร์รัปต์ (รีจิสเตอร์ 02H) เพื่อแสดงสาเหตุของการเกิดอินเตอร์รัปต์ มีรายละเอียดหน้าที่ของแต่ละบิตดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
0	TXE	TBE	BREK	FRME	PARE	OVFE	RxRD

TXE (Transmitter Empty) "1" แสดงว่าทั้งในรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลและชิฟต์รีจิสเตอร์ว่าง
"0" แสดงว่ามีข้อมูล 1 ไบต์เก็บอยู่ในรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลและชิฟต์รีจิสเตอร์

TBE(Transmitter Buffer Empty) "1" แสดงว่าภายในรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูลว่าง
"0" แสดงว่าข้อมูล 1 ไบต์เก็บอยู่ในรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล

BREK(Break) "1" แสดงว่า UART ตรวจพบการหยุดกระทันหัน
"0" แสดงว่าไม่มีการหยุดรับส่งข้อมูลกระทันหัน

FRME(Frame Error) "1" แสดงว่า UART ตรวจพบความผิดพลาดแบบเฟรม
"0" แสดงว่าไม่มีข้อผิดพลาด

OVRE(Overrun Error) "1" แสดงว่า UART ตรวจพบความผิดพลาดแบบโอเวอร์รัน
"0" แสดงว่าไม่มีข้อผิดพลาด

RxRD(Received Data Ready) "1" แสดงว่ามีการรับข้อมูลเข้ามาเก็บไว้ในบัฟเฟอร์สำหรับรับข้อมูล
"0" แสดงว่าไม่มีข้อมูลมาเก็บ

7) รีจิสเตอร์ 06H

เป็นรีจิสเตอร์แสดงสถานะโมเด็ม (Modem Status Register:MSR) ใช้แสดงสถานะของขา DCD , R1 , DSR และ CTS มีแอดเดรสอยู่ที่ Base+06H มีรายละเอียดหน้าที่ของแต่ละบิตดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
DCD	RI	DSR	CTR	DDCD	DRI	DDSR	DCTS

DCD "1" แสดงว่าสัญญาณที่ขา DCD เป็น "1"

"0" แสดงสัญญาณที่ขา DCD เป็น "0"

RI "1" แสดงว่าสัญญาณที่ขา RI เป็น "1"

"0" แสดงว่าสัญญาณที่ขา RI เป็น "0"

DSR "1" แสดงว่าสัญญาณที่ขา DSR เป็น "1"

"0" แสดงว่าสัญญาณที่ขา DSR เป็น "0"

CTS "1" แสดงว่าสัญญาณที่ขา CTS เป็น "1"

"0" แสดงว่าสัญญาณที่ขา CTS เป็น "0"

DDCD (Delta Data Carrier Dectect)

"1" แสดงว่าบิต DCD มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DRT (Delta Ring Indicator)

"1" บิต RI มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DDSR(Delta Data Ready)

"1" บิต DSR มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

DCTS (Delta Clear To Send)

"1" บิต CTS มีการเปลี่ยนแปลงเมื่อเทียบจากการอ่านค่าครั้งที่แล้ว

"0" แสดงว่าไม่มีการเปลี่ยนแปลงเมื่อเทียบกับการอ่านค่าครั้งที่แล้ว

8) ริจิสเตอร์ 07H

เป็นรีจิสเตอร์สำหรับการเก็บข้อมูลชั่วคราวหรือใช้ทำหน้าที่เป็นหน่วยความจำแรมขนาด 1 ไบต์ ซึ่งจะไม่มีผลใดๆ กับการใช้งาน UART

บทที่ 3
การออกแบบการสร้างและการทำงาน

3.1 กล่าวนำ

การออกแบบและการสร้างโปรแกรม ได้แบ่งการทำงานของโปรแกรมออกเป็นส่วนๆ เพื่อให้ง่ายต่อการเขียนโปรแกรม และการพัฒนาโปรแกรมต่อไปในอนาคต ซึ่งเราได้แบ่งโปรแกรมออกเป็น 3 ส่วนใหญ่ๆ คือ

1) ส่วนที่ใช้เชื่อมต่อกับส่วนอื่นๆ ซึ่งเป็นหน้าจอหลักและใช้เป็น Editor ด้วย

2) ส่วนของการจำลองผลการทำงานของ Program ใน เครื่องคอมพิวเตอร์

3) ส่วนของการติดต่อกับ MCS-51 ผ่านทางพอร์ตอนุกรม

การจัดทำด้านฮาร์ดแวร์ประกอบด้วยส่วนของวงจรไมโครคอนโทรลเลอร์ เพื่อใช้เชื่อมต่อและทำงานร่วมกับส่วนของโปรแกรมในคอมพิวเตอร์ และมีส่วนของโปรแกรมมอนิเตอร์ เป็นตัวเชื่อมต่อการทำงานไปสู่ฮาร์ดแวร์ ก็คืออุปกรณ์ต่อพ่วงนั่นเอง

3.2 การออกแบบโปรแกรม

การออกแบบโปรแกรมแบ่งออกเป็นส่วนต่างๆ ดังนี้

1) ฟอร์มหลัก FrmMainCommu ใช้ติดต่อกับโปรแกรมจำลองการทำงานภายใน MCS-5 1 และโปรแกรมไมโครคอนโทรลเลอร์ผ่านทางพอร์ตอนุกรม

2) ฟอร์มจำลองการทำงาน Frmsimulate ทำหน้าที่จำลองการทำงานของ MCS-51 และอุปกรณ์ต่อพ่วง ตามคำสั่งภาษาแอสแซมบลีที่เขียนขึ้น

3) ฟอร์มติดต่อทางพอร์ตอนุกรม Frmconect ทำหน้าที่เป็นตัวส่งข้อมูลใน Heximal Files ทางพอร์ตอนุกรมออกไปยังหน่วยความจำแรม ในบอร์ดคอนโทรลเลอร์ภายนอกสามารถทำงานอย่างต่อเนื่องกับฟอร์มหลัก FrmMainCommu

4) ฟอร์มเลือกอุปกรณ์ต่อพ่วงจำลอง frmselectzone ใช้เลือกอุปกรณ์ต่อพ่วง ที่อยู่บนเครื่องคอมพิวเตอร์ที่ต้องการดูผลจำลองการทำงาน

3.2.1 ฟอร์มหลัก FrmMainCommu

ภายในฟอร์มนี้จะทำหน้าที่ติดต่อกับฟอร์มต่างๆ ที่เกี่ยวเนื่องกับโปรแกรมจำลองการทำงานภายใน MCS-51 และยังทำหน้าที่แก้ไขหรือเขียนภาษาแอสแซมบลีเบื้องต้นได้ เมื่อแก้ไข

หรือเขียนเสร็จก็สามารถแปลงเป็นแฟ้มเลขฐานสิบหกได้ ในการแปลงหากเกิดข้อผิดพลาดขึ้นจะมีหน้าต่างหน้าต่างหนึ่ง แสดงจุดบกพร่องเพื่อเทียบเคียงในการแก้ไข

การเลือกรูปแบบการทำงานระหว่างการใช้งานทั่วไป และการศึกษาตามใบงานก็สามารถกระทำได้ในฟอร์มนี้เช่นกัน และในละรูปแบบการทำงานยังสามารถเลือกเพื่อไปทำงานในฟอร์มต่าง ๆ ทั้งการจำลองการทำงานบนคอมพิวเตอร์และอุปกรณ์ต่อพ่วงได้

3.2.2 ฟอร์มติดต่อทางพอร์ตอนุกรม Frmconect

ทำหน้าที่เป็นตัวส่งข้อมูลใน Heximal Files ทางพอร์ตอนุกรมออกไปยังหน่วยความจำแรมในบอร์ดคอนโทรลเลอร์ภายนอกสามารถทำงานอย่างต่อเนื่องกับฟอร์มหลัก FrmMainCommu ได้ กล่าวคือในตัว Edit หากมีการเปิดไฟล์แอสแซมบลีมาทำการแก้ไข และแปลงเมื่อเข้าสู่หน้าต่างนี้ก็จะทำการเปิดไฟล์เดียวกันแต่เป็น Heximal Files ขึ้นมาโดยอัตโนมัติเพื่อพร้อมที่จะส่งออก และสามารถเปลี่ยนไฟล์ใหม่ได้

3.2.3 ฟอร์มเลือกอุปกรณ์ต่อพ่วงจำลอง Frmselectzone

เป็นตัวช่วยฟอร์มของฟอร์ม Frmsimulate ในการเลือกอุปกรณ์ต่อพ่วง ที่อยู่บนเครื่องคอมพิวเตอร์ที่ต้องการดูผลจำลองการทำงานโดยมีอุปกรณ์ให้เลือกทั้งหมด 10 ชิ้น

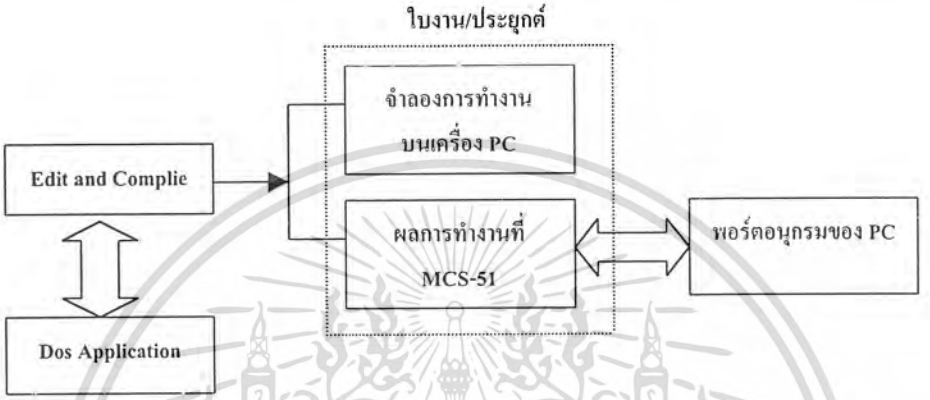
3.2.4 ฟอร์มจำลองการทำงาน Frmsimulate

โดยหน้าที่หลักคือจำลองการทำงานทั้งหน่วยความจำภายในที่มีค่าเปลี่ยนแปลงและหน่วยความจำโปรแกรมภายนอก และจุดสำคัญการเลือกอุปกรณ์ต่อพ่วง เพื่อดูการทำงานของโปรแกรมที่เขียนขึ้น กับอุปกรณ์ต่อพ่วงต่างๆ ที่เลือก การทำงานของหน้าต่างเป็นลักษณะของการจำลองการทำงานจริงที่จะเกิดขึ้นกับอุปกรณ์จริง ดังนั้นขั้นตอนต่างๆ จึงต้องเป็นไปตามลักษณะการใช้อุปกรณ์จริง

หมายเหตุ รูปแบบการทำงานทั้งใบงาน และประยุกต์ มีการใช้หน้าต่างและเครื่องมือที่ใกล้เคียงกัน ต่างกันที่ใบงานจะเป็นในลักษณะอัตโนมัติ

3.3 โครงสร้างของโปรแกรม

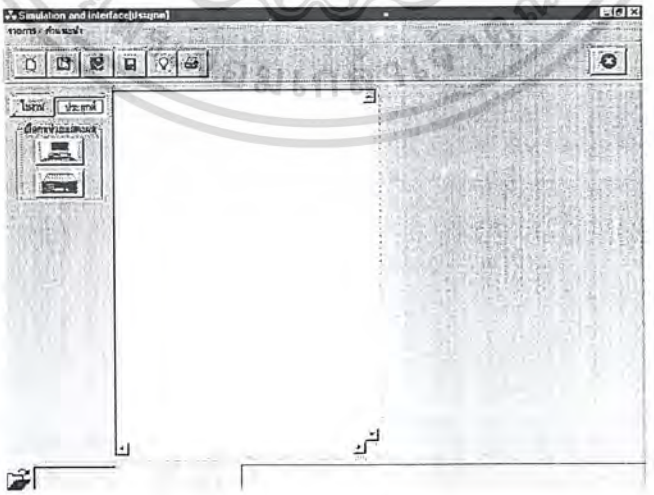
การออกแบบและการสร้างโปรแกรม ได้แบ่งการทำงานของโปรแกรมออกเป็นส่วนๆ เพื่อให้ง่ายต่อการเขียนโปรแกรม คือ ส่วนที่ใช้เชื่อมต่อกับส่วนอื่นๆ ซึ่งเป็นหน้าจอหลัก และ Editor, ส่วนของการจำลองการทำงาน และส่วนของติดต่อกับ MCS-51 ผ่านทางพอร์ตอนุกรม ดังรูปที่ 3.1



รูปที่ 3.1 แผนผังการทำงานของโปรแกรม

3.3.1 การสร้างฟอร์มหลัก FrmMainCommu

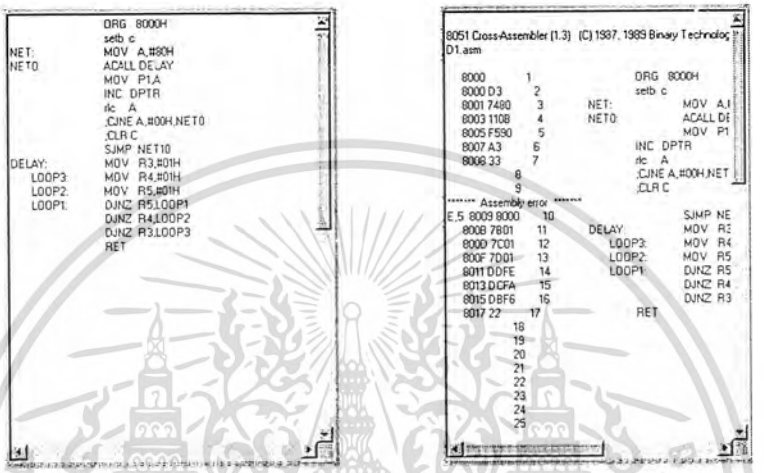
ฟอร์มหลักนี้มีหน้าที่ในการแก้ไขและแปลงแฟ้มข้อมูลภาษาแอสแซมบลีที่ผู้ใช้ต้องการ และยังเป็นจุดเชื่อมต่อไปยังฟอร์มต่างๆ ที่มีอยู่ในโปรแกรม ลักษณะของหน้าจอดังรูปที่ 3.2



รูปที่ 3.2 หน้าจอฟอร์มหลัก FrmMainCommu.frm

รายละเอียดในการสร้างฟอร์มหลัก FrmMainCommu

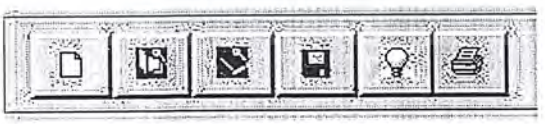
1) วางชิ้นอุปกรณ์ต่างๆ ลงบนฟอร์มว่างๆ พร้อมทั้งตั้งชื่อดังรูปที่ 3.1 หน้าจอ Edit และหน้าจอ Error, รูปที่ 3.4 เมนูบาร์, รูปที่ 3.5 แถบเครื่องมือ, รูปที่ 3.6 แถบแสดงสถานะของโปรแกรมขณะนั้น และรูปที่ 3.7 ปุ่มเลือกโหมดการทำงาน



(ก) หน้าจอ Edit (ข) หน้าจอ Error

รูปที่ 3.3 หน้าจอ Edit และ หน้าจอ Error

รูปที่ 3.4 เมนูบาร์



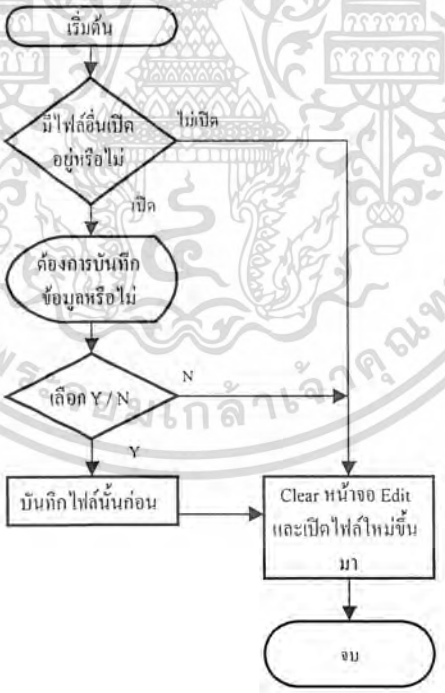
รูปที่ 3.5 แถบเครื่องมือ

รูปที่ 3.6 แถบแสดงสถานะการทำงานของโปรแกรมในขณะนั้น

รูปที่ 3.7 ปุ่มเลือกโหมดการทำงาน

2) เขียนโปรแกรมควบคุมชิ้นอุปกรณ์ต่างๆ ให้มีความสอดคล้องและเป็นไปตามลักษณะการทำงานของโปรแกรมที่ได้กำหนดไว้ ดังนี้รายละเอียดดังนี้

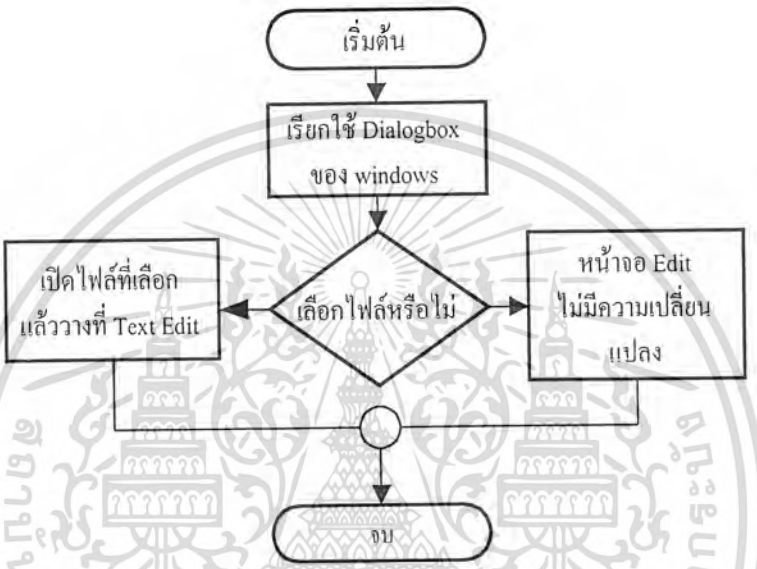
2.1) ฟังก์ชันการทำงานโปรแกรมย่อยของ Menu mnunew หรือ Menu Bar cmdnew



รูปที่ 3.8 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu mnunew หรือ Menu Bar cmdnew

จากรูปที่ 3.8 เมื่อเกิดเหตุการณ์คลิก mnunew หรือ cmdnew จะตรวจสอบว่าขณะนี้มีแฟ้มใดใช้งานอยู่หรือไม่ถ้ามีก็จะแสดงกล่องข้อความขึ้นมาถามก่อนที่จะสร้างไฟล์ขึ้นมาใหม่ แต่ถ้าขณะนั้นไม่มีแฟ้มใดเปิดอยู่ก็จะทำการสร้างไฟล์และลบหน้าจอ Edit

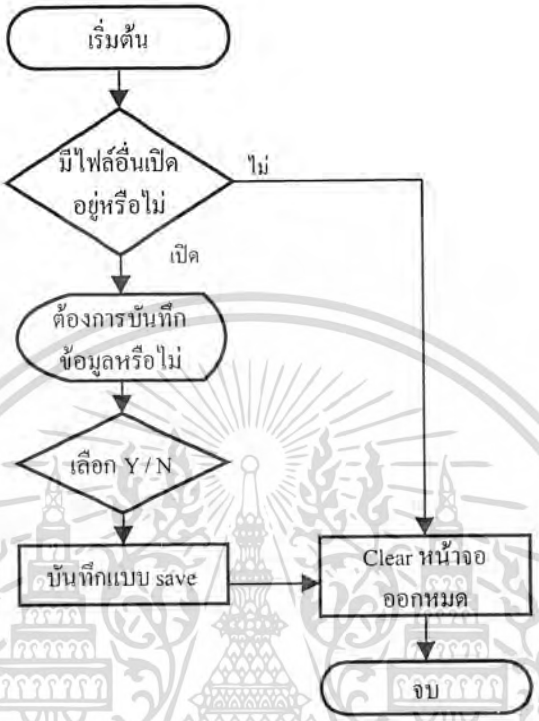
2.2) ฟังก์ชันการทำงานโปรแกรมย่อยของ Menu mnuopen หรือ Menu Bar cmdopen



รูปที่ 3.9 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu mnuopen หรือ Menu Bar cmdopen

จากรูปที่ 3.9 เมื่อเกิดเหตุการณ์คลิกที่ mnuopen หรือ cmdopen จะแสดง Dialog Box Control ซึ่งเป็นตัวคอนโทรลสำเร็จรูปที่ใช้สำหรับเปิดหรือบันทึกแฟ้มในวินโดวส์ จากนั้นก็จะตรวจสอบว่ามีการเลือกแฟ้มหรือไม่ หากไม่มีการเลือกก็จะไม่มีการเปลี่ยนแปลงใดๆ ที่ Text Edit แต่ถ้าเลือกก็จะวางตัวอักษรในแฟ้มนั้นบน Text Edit

2.3) ฟังก์ชันการทำงานโปรแกรมย่อยของ Menu mnuclose หรือ Menu Bar cmdclose



รูปที่ 3.10 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu mnuclose หรือ Menu Bar cmdclose

จากรูปที่ 3.10 เมื่อเกิดเหตุการณ์คลิก mnuclose หรือ cmdclose จะตรวจสอบทันทีว่าในแฟ้มข้อมูลกำลังเปิดอยู่ที่หน้าจอแก้ไขหรือไม่ หากมีจะแสดงกล่องข้อความ ว่าจะบันทึกหรือไม่แต่ถ้าไม่มีแฟ้มข้อมูลใดเปิด อยู่ก็จะทำการล้างหน้าจอแก้ไขข้อความให้ว่าง

2.4) ฟังก์ชันการทำงานโปรแกรมย่อยของ Menu mnusave หรือ Menu Bar cmdsave

รูปที่ 3.11 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu mnusave หรือ Menu Bar cmdsave

จากรูปที่ 3.11 เมื่อเกิดเหตุการณ์คลิก mnusave หรือ cmdsave จะตรวจสอบว่าแฟ้มที่จะบันทึกเป็นแฟ้มที่สร้างเองโดยอัตโนมัติหรือไม่หากเป็นแฟ้มที่สร้างเองอัตโนมัติ ก็จะแสดง Dialog Box เพื่อให้แก้ไขชื่อไฟล์เสียใหม่หรือไม่แก้ไขก็ได้ แต่ถ้าเป็นแฟ้มอื่นก็จะบันทึกตามชื่อแฟ้มนั้นเลย

2.5) ฟังก์ชันการทำงานโปรแกรมย่อยของ Menu mnusaveas

รูปที่ 3.12 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu mnusaveas

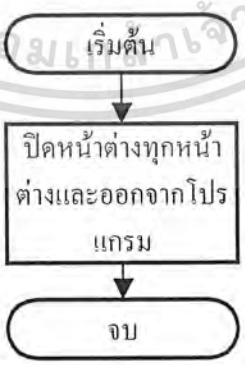
จากรูปที่ 3.12 เมื่อเกิดเหตุการณ์คลิก mnusaveas จะแสดง Dialog Box เพื่อตั้งชื่อแฟ้มตามต้องการ และจะ บันทึกแฟ้มตามชื่อที่ตั้งนั้นทันที

2.6) ฟังก์ชันการทำงานโปรแกรมย่อย Menu mnuprint หรือ Menu Bar cmdprint

รูปที่ 3.13 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu mnuprint หรือ Menu Bar cmdprint

จากรูที่ 3.13 เมื่อเกิดเหตุการณ์คลิก mnuprint หรือ cmdprint จะแสดง Dialog Box เพื่อกำหนดคุณสมบัติเกี่ยวกับการจัดพิมพ์ต่างๆ ก่อนทำการพิมพ์ออกทางเครื่องพิมพ์

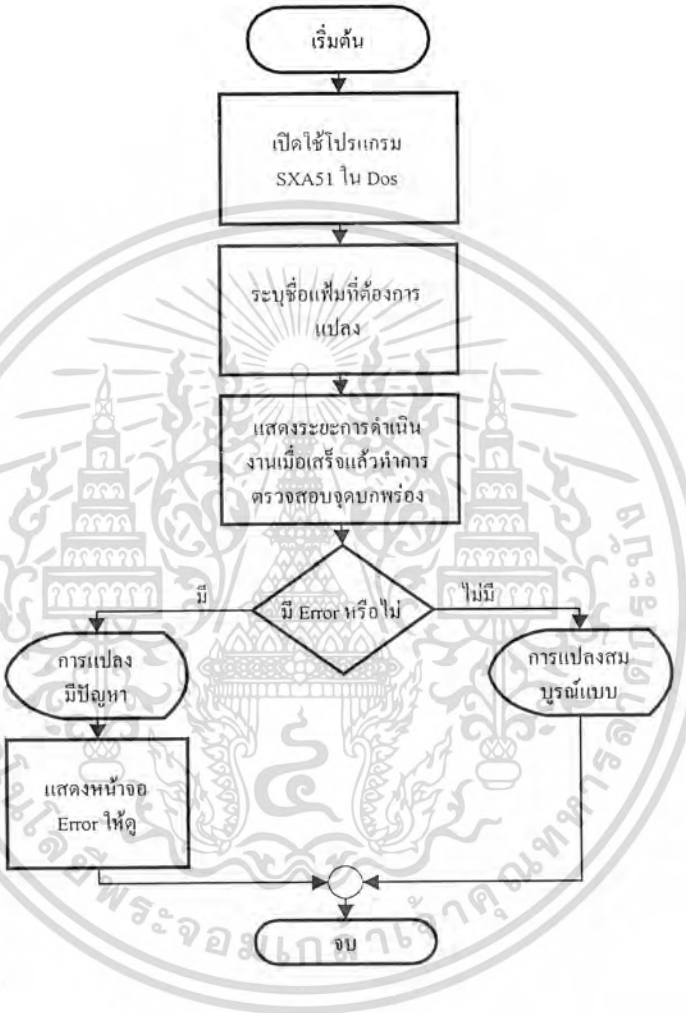
2.7) ฟังก์ชันการทำงานโปรแกรมย่อย Menu mnuend หรือ Menu Bar cmdend



รูปที่ 3.14 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu mnuend หรือ Menu Bar cmdend

จากรูปที่ 3.14 เมื่อเกิดเหตุการณ์คลิก mnuend หรือ cmdend จะปิดหน้าต่างทุกหน้าต่างออกจากโปรแกรม

2.8) ฟังก์ชันการทำงานโปรแกรมย่อยของ Menu Bar cmdcomplie



รูปที่ 3.15 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu Bar cmdcomplie

จารรูปที่ 3.15 เมื่อเกิดเหตุการณ์คลิก cmdcomplie จะเรียกใช้คำสั่ง SXA51 ที่เป็นโปรแกรมทำงานภายใต้ระบบ Dos เพื่อทำการแปลงนามสกุล .asm แปลงและสร้างแฟ้มนามสกุล .hex และ .lst ขึ้นมาจากนั้นจะทำการทำแฟ้มนามสกุล .lst มาตรวจดูหาจุด Error ซึ่งถ้ามีจะปรากฏหน้าจอ Error ขึ้นมาเพื่อใช้เทียบเคียงในการแก้ไขจุดต่างๆ หากไม่พบจุดบกพร่องจะปรากฏข้อความ เพื่อบอกว่าการแปลงไม่พบจุดบกพร่องใดๆ

2.9) cmdmcs51app หรือ cmdmcs51lab เป็นปุ่มที่ใช้เรียกฟอร์ม frmconect ขึ้นมาเพื่อดูการทำงานของโปรแกรมบน MCS-51 และชุดอินเตอร์เฟสภายนอก

2.10) cmdpcapp หรือ cmdpclab เป็นปุ่มที่ใช้เรียกฟอร์ม frmsimulate ขึ้นมาเพื่อดูการทำงานของโปรแกรมจำลองการทำงาน บนหน้าจอคอมพิวเตอร์

2.11) optapp เป็นปุ่มเลือกรูปแบบการทำงานแบบประยุกต์

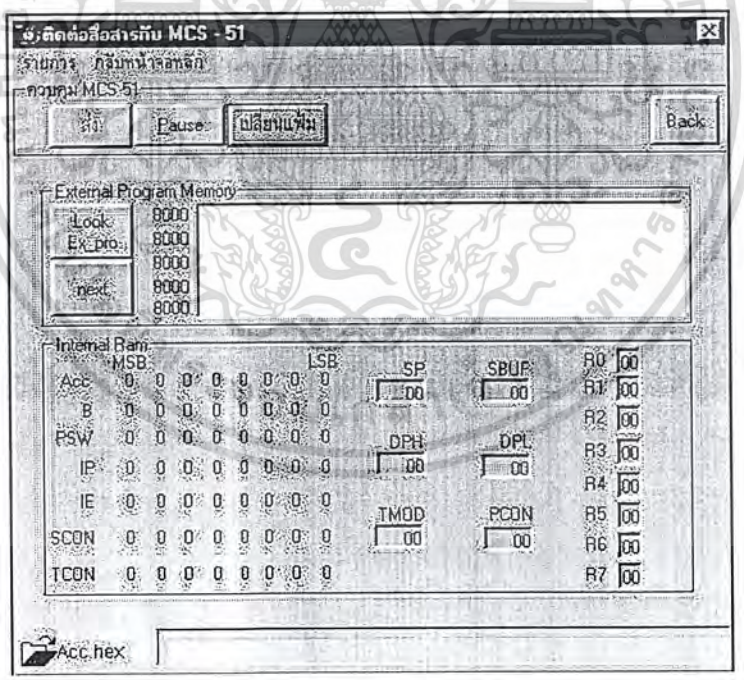
2.12) oplab เป็นปุ่มเลือกรูปแบบการทำงานแบบการทดลอง

2.13) pic 2 ใช้แสดงชื่อแฟ้มที่เปิดอยู่ในขณะนั้นๆ

2.14) pgb2 ใช้บอกผลการดำเนินการแต่ละอย่างว่า คืบหน้าเพียงใด

3.3.2 การสร้างฟอร์มติดต่อทางพอร์ตอนุกรม frmconect

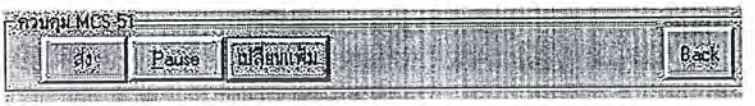
ทำหน้าที่เป็นตัวส่งข้อมูลใน Heximal Files ทางพอร์ตอนุกรมออกไปยังหน่วยความจำแรมในบอร์ดคอนโทรลเลอร์ภายนอกสามารถทำงานอย่างต่อเนื่องกับฟอร์มหลัก FrmMainCommu หน้าต่างฟอร์ม frmconet แสดงดังรูปที่ 3.16



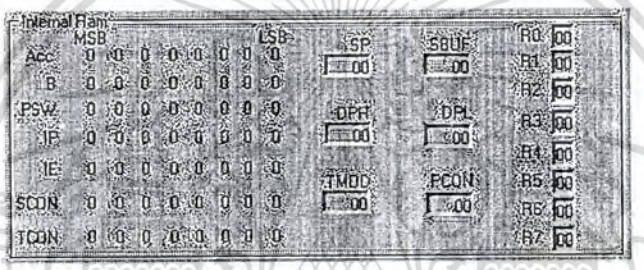
รูปที่ 3.16 หน้าจอฟอร์มติดต่อสื่อสารกับ MCS-51 ทางพอร์ตอนุกรม

รายละเอียดในการสร้างฟอร์มติดต่อทางพอร์ตอนุกรม frmconect

1) วางชิ้นอุปกรณ์ต่างๆ ลงบนฟอร์มว่างๆ พร้อมทั้งตั้งชื่อดังรูปที่ 3.17 แถบเครื่องมือควบคุมการทำงาน MCS-51, รูปที่ 3.18 หน้าต่างแสดงหน่วยความจำภายในคอนโทรเลอร์, รูปที่ 3.19 หน้าต่างแสดงหน่วยความจำภายนอก, รูปที่ 3.20 เมนูบาร์



รูปที่ **3.17** แถบเครื่องมือควบคุมการทำงาน MCS-51



รูปที่ **3.18** หน้าต่างหน่วยความจำภายในคอนโทรลเลอร์

รูปที่ **3.19** หน้าต่างหน่วยความจำข้อมูลภายนอก

รูปที่ **3.20** เมนูบาร์ (Frm.conect)

2) เขียนโปรแกรมควบคุมชิ้นอุปกรณ์ต่างๆ ให้มีความสอดคล้องและเป็นไปตามลักษณะการทำงานของโปรแกรมที่ได้กำหนดไว้ มีรายละเอียดดังนี้

2.1) ฟังก์ชันการทำงานโปรแกรมย่อยของ form_load

รูปที่ 3.21 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย form_load

จากรูปที่ 3.21 ในขณะโหลดฟอร์มขึ้นมาจะทำการตรวจสอบแฟ้มที่เป็นแฟ้มนามสกุล .asm ว่าเปิดอยู่ที่หน้าจอหลักหรือไม่ หากเปิดอยู่ก็จะเปิดแฟ้มที่เป็นแฟ้มนามสกุล .hex ให้โดยอัตโนมัติ แต่ถ้าไม่ได้เปิดไว้ ก็จะทำงานตามปกติคือโหลดฟอร์มขึ้นมา

2.2) ฟังก์ชันการทำงานโปรแกรมย่อย Menu mnuopen

รูปที่ 3.22 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu mnuopen

จากรูปที่ 3.22 เมื่อเกิดการคลิก Menu open จะแสดง Dialog Box Control เลือกแฟ้มที่จะส่งไปยัง MCS-51

2.3) ฟังก์ชันการทำงานโปรแกรมย่อย Menu Bar cmdsend

รูปที่ 3.23 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu Bar cmdend

จากรูปที่ 3.23 เมื่อเกิดเหตุการณ์คลิก จะทำการตั้งค่าพอร์ตอนุกรม และส่งข้อมูลที่เปิดไว้ออกไปทางพอร์ตอนุกรมทุกตัวเข้า MCS-51

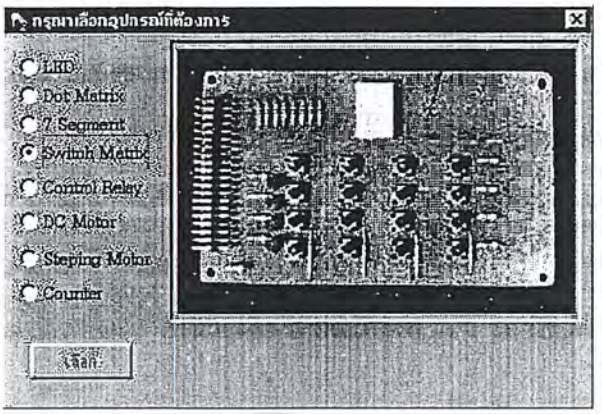
2.4) ฟังก์ชันการทำงานโปรแกรมย่อย Menu Bar cmdreset

รูปที่ 3.24 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Menu Bar cmdreset

จากรูปที่ 3.24 เมื่อเกิดเหตุการณ์คลิก cmdreset จะตั้งค่าให้กับพอร์ตอนุกรมจากนั้นจะส่งค่ารีเซตออกไปทางพอร์ตอนุกรมและลบแฟ้มข้อมูลที่เปิดอยู่ออกไป

3.3.3 การสร้างฟอร์มเลือกอุปกรณ์ต่อพ่วงจำลอง frmSelectzone

ใช้เลือกอุปกรณ์ต่อพ่วงจำลอง ที่อยู่บนเครื่องคอมพิวเตอร์ที่ต้องการดูผลจำลองการทำงานหน้าต่างฟอร์มเลือกอุปกรณีต่อพ่วงจำลอง frmSelectzone แสดงดังรูปที่ 3.25

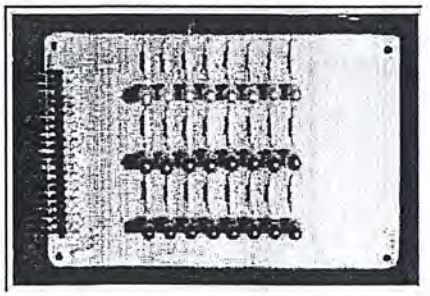


รูปที่ 3.25 หน้าจอฟอร์มเลือกอุปกรณ์ต่อพ่วงจำลอง (frmSelectzone)

รายละเอียดในการสร้างฟอร์มเลือกอุปกรณ์ต่อพ่วงจำลอง frmSelectzone

1) วางชิ้นอุปกรณ์ต่างๆ ลงบนฟอร์มว่างๆ พร้อมทั้งตั้งชื่อดังรูปที่ 3.26 หน้าต่างเลือกอุปกรณ์ต่อพ่วง และรูปที่ 3.27 แสดงอุปกรณ์ต่อพ่วงที่เลือก

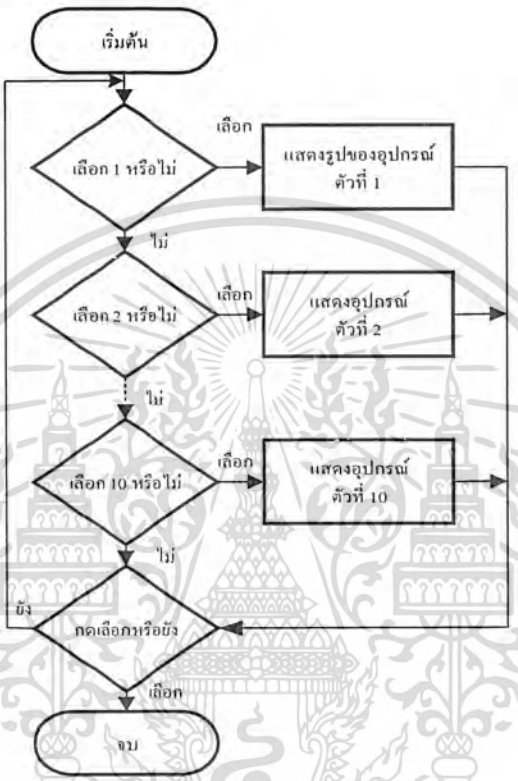
รูปที่ 3.26 หน้าต่างตัวเลือกอุปกรณ์ต่อพ่วง



รูปที่ 3.27 หน้าต่างอุปกรณ์ต่อพ่วงตามที่เลือก

2) เขียนโปรแกรมควบคุมชิ้นอุปกรณ์ต่างๆ ให้มีความสอดคล้องและเป็นไปตามลักษณะการทำงานของโปรแกรมที่ได้กำหนดไว้ ดังนี้รายละเอียดดังนี้

2.1) ฟังก์ชันการทำงานโปรแกรมย่อย Selecter option1



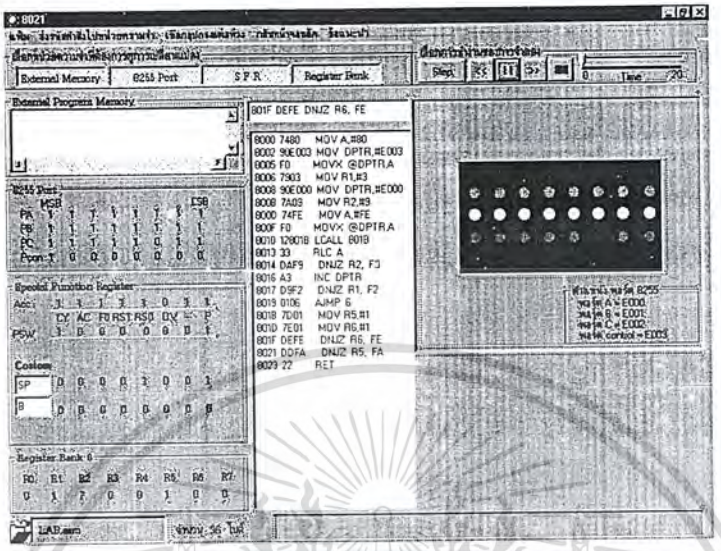
รูปที่ 3.28 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย Selecter option1

จากรูปที่ 3.28 เมื่อเกิดเหตุการณ์คลิก Selecter option1 สามารถเลือกได้เพียงหนึ่งเท่านั้น หากมีการคลิกที่ตัวใดจะมีภาพอุปกรณ์ที่เลือกขึ้นและส่งค่าไปยัง formsimulate ต่อไปเพื่อใช้อ้างอิงชนิดอุปกรณ์ที่เลือก

2.2) ปุ่ม cmdselect เพื่อยืนยันการเลือกตัวอุปกรณ์ที่เลือก

3.3.4 การสร้างฟอร์มจำลองการทำงาน frmsimulate

ทำหน้าที่จำลองการทำงาน ของ MCS-51 และอุปกรณ์ต่อพ่วงตามคำสั่งภาษาแอสแซมบลีที่เขียนขึ้น หน้าจอฟอร์มจำลองการทำงาน frmsimlate ดังรูปที่ 3.29

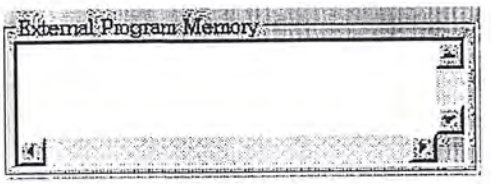


รูปที่ 3.29 หน้าจอฟอร์มจำลองการทำงาน frmsimulate

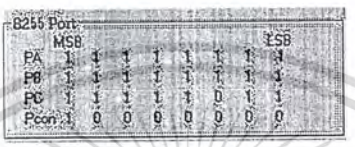
รายละเอียดในการสร้างฟอร์มจำลองการทำงาน frmsimulate

1) วางชิ้นอุปกรณ์ต่างๆ ลงบนฟอร์มว่างๆ พร้อมทั้งตั้งชื่อดังรูปที่ 3.30 เมนูบาร์, รูปที่ 3.31 หน้าต่างหน่วยความจำโปรแกรมภายใน, รูปที่ 3.32 หน้าต่างพอร์ต 8255, รูปที่ 3.33 หน้าต่างรีจิสเตอร์พิเศษ, รูปที่ 3.34 แถบแสดงสภาวะการทำงานต่างๆ, รูปที่ 3.35 หน้าต่างหน่วยความจำรีจิสเตอร์แบงค์, รูปที่ 3.36 แถบเครื่องมือเลือกการทำงานของการจำลอง, รูปที่ 3.37 หน้าต่างเลือกหน่วยความจำที่ต้องการดูการเปลี่ยนแปลง, รูปที่ 3.38 หน้าต่างแสดงรหัส Memonic, รูปที่ 3.39 หน้าต่างแสดงสภาวะการทำงานของอุปกรณ์ต่อพ่วง

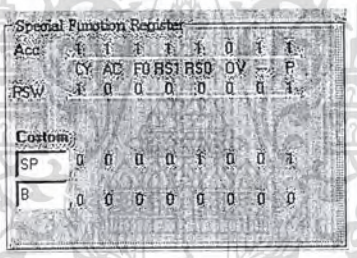
รูปที่ 3.30 เมนูบาร์



รูปที่ 3.31 หน้าต่างหน่วยความจำโปรแกรมภายใน

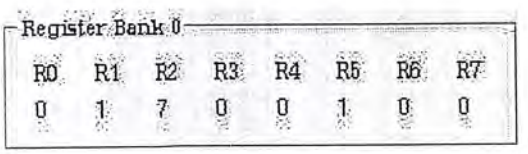


รูปที่ 3.32 หน้าต่างพอร์ต 8255



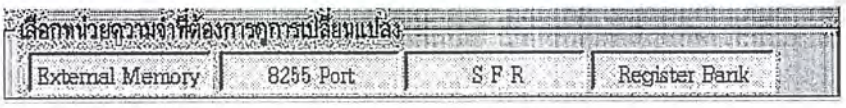
รูปที่ 3.33 หน้าต่างรีจิสเตอร์หน้าที่พิเศษ

รูปที่ 3.34 แถบแสดงสภาวะการทำงานต่างๆ

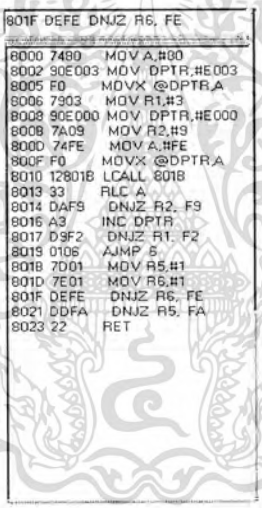


รูปที่ 3.35 หน้าต่างหน่วยความจำรีจิสเตอร์แบงค์

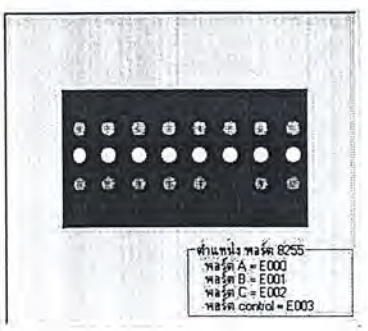
รูปที่ 3.36 แถบเครื่องมือการทำงานของการจำลอง



รูปที่ 3.37 แถบเครื่องมือเลือกหน่วยความจำที่ต้องการดูการเปลี่ยนแปลง



รูปที่ 3.38 หน้าต่างรหัส Memonic



รูปที่ 3.39 หน้าต่างสภาวะการทำงานของอุปกรณ์ต่อพ่วงที่เลือก

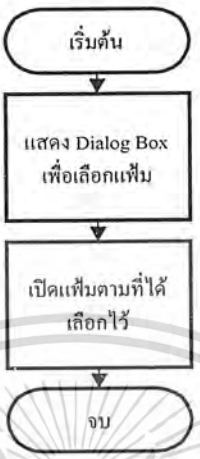
2) เขียนโปรแกรมควบคุมอุปกรณ์ต่างๆ ให้มีความสอดคล้อง และเป็นไปตามลักษณะการทำงานของโปรแกรมที่ได้กำหนดไว้ มีรายละเอียดดังนี้

2.1) ฟังก์ชันการทำงานโปรแกรมย่อยของ formload

รูปที่ 3.40 ผังงานของฟังก์ชันการทำงานโปรแกรมย่อย formload

จากรูปที่ 3.40 เมื่อโหลดฟอร์มขึ้นมาจะทำการตั้งค่าตัวอุปกรณ์และตัวแปรต่างๆ ที่สร้างไว้ จากนั้นจะตรวจสอบว่ามีการเปิดแฟ้มนามสกุล .asm ไว้หรือไม่หากมีการเปิดอยู่จะทำการเปิดแฟ้มเดียวกันแต่เป็น แฟ้มนามสกุล .hex ให้โดยอัตโนมัติหากไม่มีก็จะทำงานตามปกติ

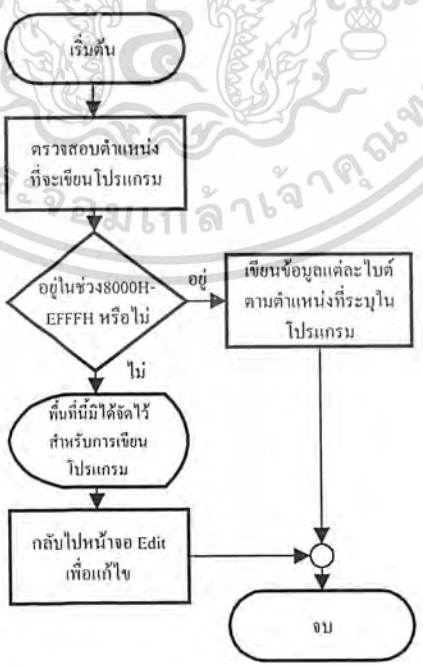
2.2) ฟังก์ชั่นการทำงานโปรแกรมย่อย Menu mnuopen



รูปที่ 3.41 ผังงานของฟังก์ชั่นการทำงาน โปรแกรมย่อย Menu mnuopen

จากรูปที่ 3.41 เมื่อเกิดเหตุการณ์คลิก mnuopen จะแสดง Control Dialog Box เพื่อกำหนดชื่อแฟ้มที่ต้องการ

2.3) ฟังก์ชั่นการทำงาน โปรแกรมย่อย Menu mnusend



รูปที่ 3.42 ผังงานของฟังก์ชั่นการทำงาน โปรแกรมย่อย Menu mnusend

จากรูปที่ 3.42 เมื่อเกิดเหตุการณ์คลิกจะทำการตรวจสอบตำแหน่งที่จะเขียนลงในตัวแปรที่จัดไว้หากตำแหน่ง ของโปรแกรมอยู่นอกเหนือจากที่จัดไว้ให้จะแสดงกล่องข้อความเตือนเพื่อให้กลับไป แก้ไขอีกครั้ง แต่ถ้าตำแหน่งของโปรแกรมอยู่ภายในที่จัดไว้ให้จะบันทึกโปรแกรมที่ระบุอยู่ในโปรแกรมจนครบทุกตำแหน่ง

3.4 การสร้างชุดคำสั่งของ MCS-51

เขียนคำสั่งแยกออกมา Frmsimulate Code เพื่อสะดวกต่อการดูแลเรียกใช้โดยให้ตัวแปรหนึ่งเป็นตัวเก็บรหัสคำสั่ง เพื่อเปรียบเทียบว่าเป็นรหัสคำสั่งใด และทำการประมวลผลตามคำสั่งนั้นๆ ซึ่งสามารถแยกลักษณะการประมวลผลได้ทั้งหมด 5 ประเภท ดังนี้

1) คำสั่งเคลื่อนย้ายข้อมูล (Data Transfer Instructions)

2) คำสั่งที่ใช้คำนวณทางคณิตศาสตร์ (Arithmatic Instructions)

3) คำสั่งกระทำระดับบิต (Logical Instructions)

4) คำสั่งคำนวณทางตรรก (Boolean Instructions)

5) คำสั่งควบคุมการอ่านโปรแกรม (Program Counter Control Instructions)

5.1) คำสั่ง 1 ไบต์

5.2) คำสั่ง 2 ไบต์

5.3) คำสั่ง 3 ไบต์

3.4.1 รายละเอียดการทำงานตามชุดคำสั่งของ MCS-51 ที่เป็นคำสั่ง 1 ไบต์

1) คำสั่งเคลื่อนย้ายข้อมูล

1.1) คำสั่ง MOV

รูปแบบคำสั่ง

' "MOV A,Rn"

' "MOV Rn,A"

หน้าที่

คัดลอกค่าที่อยู่ในรีจิสเตอร์แบงค์ นำมาใส่ในรีจิสเตอร์ A หรือคัดลอกค่าที่อยู่ในรีจิสเตอร์ A นำมาใส่ในรีจิสเตอร์แบงค์

การทำงาน

ตรวจสอบค่าในรีจิสเตอร์ PSW เพื่อเลือกใช้งานรีจิสเตอร์แบงค์จากนั้นให้ค่าในรีจิสเตอร์แบงค์ มีค่าเท่ากับค่าในรีจิสเตอร์ A หรือให้ค่าในรีจิสเตอร์ A มีค่าเท่ากับค่าในรีจิสเตอร์แบงค์ สามารถเขียนโปรแกรมจำลองการทำงานดังรูปที่ 3.43

```
'"MOV  A,Rn"
Internal_ram(&HE0) = Internal_ram(bank + n)

' "MOV  Rn,A"
Internal_ram(bank + n) = Internal_ram(&HE0)
```

รูปที่ 3.43 รูปแบบคำสั่ง MOV

1.2) คำสั่ง MOV

รูปแบบคำสั่ง

' "MOV @Ri,A"

' "MOV A,@Ri"

หน้าที่

คัดลอกค่าในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่มาใส่ไว้ในรีจิสเตอร์ A หรือคัดลอกค่าในรีจิสเตอร์ A มาใส่ไว้ในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่

การทำงาน

ตรวจสอบค่าในรีจิสเตอร์ PSW เพื่อเลือกใช้งานรีจิสเตอร์แบงค์ จากนั้นให้ค่าในรีจิสเตอร์ A มีค่าเท่ากับค่าในตำแหน่งที่รีจิสเตอร์แบงค์เชี้อยู่ หรือ ให้ค่าในตำแหน่งที่รีจิสเตอร์แบงค์เป็นตัวชี้อยู่ที่ค่าเท่ากับค่าในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานดังรูปที่ 3.44

```
'"MOV  @Ri,A"
Internal_ram(Internal_ram(bank + i)) = Internal_ram(&HE0)

' "MOV  A,@Ri"
Internal_ram(&HE0) = Internal_ram(Internal_ram(bank + i))
```


รูปที่ 3.44 รูปแบบคำสั่ง MOV

1.3) คำสั่ง MOVC

รูปแบบคำสั่ง

"MOVC A,@A + DPTR"

หน้าที่

คัดลอกค่าในตำแหน่งที่ชี้โดย DPTR ที่บวกตำแหน่งเพิ่มไปเท่ากับค่าในรีจิสเตอร์ A แล้วนำค่ามาไว้ในรีจิสเตอร์ A

การทำงาน

รวมค่าของ DPTR และค่าที่อยู่ในรีจิสเตอร์ A เพื่อเป็นตำแหน่งที่ต้องการนำข้อมูลมาใส่ในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานดังรูปที่ 3.45

```
"MOVC  A,@A + DPTR"
Internal_ram(&HE0) = External_ram(DPTR + Internal_ram(&HE0) + 1)
```

รูปที่ 3.45 รูปแบบคำสั่ง MOVC

1.4) คำสั่ง MOVC

รูปแบบคำสั่ง

"MOVC A,@A + PC"

หน้าที่

คัดลอกค่าในตำแหน่งที่ชี้โดย PC ที่บวกตำแหน่งเพิ่มไปเท่ากับค่าในรีจิสเตอร์ แล้วนำค่ามาไว้ในรีจิสเตอร์ A

การทำงาน

เพิ่มค่าของ PC ขึ้น 1 แล้วบวกค่าในรีจิสเตอร์ A แล้วให้ค่าที่รีจิสเตอร์ A มีค่าเท่ากับค่าในตำแหน่งที่ PC เปลี่ยนแปลงเป็นตัวชี้สามารถเขียนโปรแกรม จำลองการทำงานดังรูปที่ 3.46

```
"MOVC  A,@A + PC"
Internal_ram(&HE0) = External_ram(PC + Internal_ram(&HE0) + 1)
```

รูปที่ 3.46 รูปแบบคำสั่ง MOVC

1.5) คำสั่ง MOVX

รูปแบบคำสั่ง

' "MOVX @DPTR,A"

' "MOVX A,@DPTR"

หน้าที่

คัดลอกค่าในตำแหน่งที่ DPTR ชี้อยู่มาเก็บไว้ในรีจิสเตอร์ A หรือคัดลอกค่าในรีจิสเตอร์ A ไปใส่ในตำแหน่งที่ DPTR ชี้อยู่

การทำงาน

ให้ค่าในรีจิสเตอร์ A มีค่าเท่ากับค่าในตำแหน่งที่ DPTR ชี้อยู่ หรือให้ค่าในตำแหน่งที่ DPTR ชี้อยู่มีค่าเท่ากับค่าในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานดังรูปที่ 3.47

```
'"MOVX  @DPTR,A"
External_ram(DPTR) = Internal_ram(&HE0)

' "MOVX  A,@DPTR"
Internal_ram(&HE0) = External_ram(DPTR)
```

รูปที่ 3.47 รูปแบบคำสั่ง MOVX

1.6) คำสั่ง XCH

รูปแบบคำสั่ง

' "XCH A,Rn"

หน้าที่

สลับข้อมูลระหว่างรีจิสเตอร์ A และรีจิสเตอร์แบงค์

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานรีจิสเตอร์แบงค์ พักข้อมูลของรีจิสเตอร์ A และรีจิสเตอร์แบงค์ไว้ แล้วให้ค่าในรีจิสเตอร์ A มีค่าเท่ากับข้อมูลของรีจิสเตอร์แบงค์ที่พักไว้อีกที่หนึ่ง และให้ค่าในรีจิสเตอร์แบงค์มีค่าเท่ากับข้อมูลของรีจิสเตอร์ A ที่พักไว้อีกที่หนึ่ง โปรแกรมดังรูปที่ 3.48

```
'"XCH  A,Rn"
swapA = Internal_ram(&HE0)
swapbank = Internal_ram(bank + n)
Internal_ram(&HE0) = swapbank
Internal_ram(bank + n) = swapA
```


รูปที่ 3.48 รูปแบบคำสั่ง XCH

1.7) คำสั่ง XCH

รูปแบบคำสั่ง

' "XCH A,@Ri"

หน้าที่

สลับข้อมูลระหว่างรีจิสเตอร์ A และรีจิสเตอร์แบงค์หรือสลับข้อมูลเฉพาะ 4 บิตล่าง ระหว่างรีจิสเตอร์ A และรีจิสเตอร์แบงค์

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์ พักข้อมูลของรีจิสเตอร์ A และรีจิสเตอร์แบงค์ชี้อยู่ แล้วให้ค่าในรีจิสเตอร์ A มีค่าเท่ากับข้อมูลของตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่ หรือเฉพาะ 4 บิตล่าง และให้ค่าในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่ มีค่าเท่ากับค่าที่พักไว้ของรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานดังรูปที่ 3.49

```
' "XCH  A,@Ri"
swapA = Internal_ram(&HE0)
swapbank = Internal_ram(Internal_ram(bank + i))
Internal_ram(&HE0) = swapbank
Internal_ram(Internal_ram(bank + i)) = swapA
```

รูปที่ 3.49 รูปแบบคำสั่ง XCH

1.8) คำสั่ง SWAP

รูปแบบคำสั่ง

' "SWAP A"

หน้าที่

สลับข้อมูลระหว่าง 4 บิตบนและ 4 บิตล่าง

การทำงาน

นำข้อมูลในรีจิสเตอร์ A แปลงเป็นฐาน 16 แล้วแยกเก็บทีละตัวอักษร จากนั้นนำมาเรียงกลับสลับตำแหน่งใหม่ และแปลงข้อมูลเป็นฐาน 10 ใส่ค่าในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.50

```
' "SWAP  A"
If Internal_ram(&HE0) < 16 Then
    swap = "0" & Hex(Internal_ram(&HE0))
Else
    swap = Hex(Internal_ram(&HE0))
End If
swap1 = Mid(swap, 1, 1)
swap2 = Mid(swap, 2, 1)
swap = swap2 & swap1
swap = "&H" & swap
Internal_ram(&HE0) = Val(swap)
```

รูปที่ 3.50 รูปแบบคำสั่ง SWAP

2) คำสั่งที่ใช้คำนวณทางคณิตศาสตร์

2.1) คำสั่ง INC หรือ DEC

รูปแบบคำสั่ง

"INC Rn"

' "DEC Rn"

หน้าที่

เพิ่มค่าในรีจิสเตอร์แบงค์ขึ้นอีก 1 หรือลดค่าในรีจิสเตอร์แบงค์ลงทีละ 1

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์ นำค่าเดิมของรีจิสเตอร์แบงค์มาบวกกับ 1 หรือลบด้วย 1 ค่า กับไปไว้ที่ตำแหน่งเดิมสามารถเขียนโปรแกรมของการทำงานได้ดังรูปที่ 3.51

```
"INC Rn"
Internal_ram(bank+n) = Val(Internal_ram(bank + n)) + 1
location_byte = Val(bank + n)
Call check_up(location_byte)

' "DEC Rn"
Internal_ram(bank + n) = Val(Internal_ram(bank + n)) - 1
location_byte = Val(bank + n)
Call check_down(location_byte)
```

รูปที่ 3.51 รูปแบบคำสั่ง INC หรือ DEC

2.2) คำสั่ง INC หรือ DEC

รูปแบบคำสั่ง

' "INC A"

' "DEC A"

หน้าที่

เพิ่มค่าในรีจิสเตอร์ A ขึ้นอีก 1 หรือลดค่าใน A ลงอีก 1

การทำงาน

นำค่าเดิมของรีจิสเตอร์ A มาบวกกับ 1 แล้วนำค่าที่ได้ไปใส่ในรีจิสเตอร์ A สามารถเขียนโปรแกรมการทำงานดังรูปที่ 3.52

```
' "INC A"
Internal_ram(&HE0) = Val(Internal_ram(&HE0)) + 1
location_byte = Val(&HE0)
Call check_up(location_byte)

' "DEC  A"
Internal_ram(&HE0) = Val(Internal_ram(&HE0)) - 1
location_byte = Val(&HE0)
Call check_down(location_byte)
```

รูปที่ 3.52 รูปแบบคำสั่ง INC หรือ DEC

2.3) คำสั่ง INC หรือ DEC

รูปแบบคำสั่ง

' "INC @Ri"

' "DEC @Ri"

หน้าที่

เพิ่มค่าในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่ขึ้น 1 หรือลดค่าในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่ลง 1

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์นำข้อมูล ในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่มาบวกกับ 1 หรือลบกับ 1 แล้วนำค่าไปใส่ไว้ที่เดิมสามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.53

```
' "INC @Ri"
Internal_ram(Internal_ram(bank + i)) = Internal_ram(Internal_ram(bank + i)) + 1
location_byte = Val(Internal_ram(bank + i))
Call check_up(location_byte)

' "DEC  @Ri"
Internal_ram(Internal_ram(bank + i)) = Internal_ram(Internal_ram(bank + i)) - 1
location_byte = Val(Internal_ram(bank + i))
Call check_down(location_byte)
```

รูปที่ 3.53 รูปแบบคำสั่ง INC หรือ DEC

2.4) คำสั่ง ADD

รูปแบบคำสั่ง

' "ADD A,Rn"

หน้าที่

บวกค่าของรีจิสเตอร์ A กับรีจิสเตอร์ Rn ผลลัพธ์ที่ได้เก็บไว้ใน รีจิสเตอร์ A

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์ นำข้อมูลของรีจิสเตอร์แบงค์มาบวกกับข้อมูลของรีจิสเตอร์ A นำค่าที่ได้มาเปรียบเทียบว่ามีค่าเกิน 255 หรือไม่ถ้าเกิน ให้ลบออกจาก 256 แล้วนำค่าที่ได้เก็บในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.54

```
' "ADD A,Rn"
Internal_ram(&HE0) = Val(Internal_ram(&HE0)) + Val(Internal_ram(bank + n))
location_byte = Val(&HE0)
Call check_up(location_byte)
```

รูปที่ 3.54 รูปแบบคำสั่ง ADD

2.5) คำสั่ง ADDC

รูปแบบคำสั่ง

' "ADDC A,Rn"

หน้าที่

บวกค่าของรีจิสเตอร์ A, รีจิสเตอร์ Rn และ Carry Flag ค่าที่ได้เก็บไว้ไรีจิสเตอร์ A

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์ นำข้อมูลของรีจิสเตอร์ A และบิตใน Carry Flag นำค่าที่ได้มาเปรียบเทียบว่ามีค่าเกิน 255 หรือไม่ถ้าเกิน ให้ลบออกจาก 256 แล้วนำค่าที่ได้เก็บในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.55

```
' "ADDC  A,Rn"
Internal_ram(&HE0) = Val(Internal_ram(&HE0)) + Val(Internal_ram(bank + n)) + CF
location_byte = Val(&HE0)
Call check_up(location_byte)
```

รูปที่ 3.55 รูปแบบคำสั่ง ADDC

2.6) คำสั่ง ADD

รูปแบบคำสั่ง

' "ADD A,@Ri"

หน้าที่

บวกค่าในรีจิสเตอร์ A กับค่าในตำแหน่งในรีจิสเตอร์แบงค์ที่ชี้อยู่ค่าที่ได้เก็บไว้ที่รีจิสเตอร์ A

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์ นำข้อมูลในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่บวกกับค่าในรีจิสเตอร์ A นำค่าที่ได้มาเปรียบเทียบว่ามีค่าเกิน 255 หรือไม่ถ้าเกิน ให้ลบออกจาก 256 แล้วนำค่าที่ได้เก็บในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.56

```
' "ADD A,@Ri"
Internal_ram(&HE0) = Internal_ram(&HE0) + _
Internal_ram(Internal_ram(bank + i))
location_byte = Val(&HE0)
Call check_up(location_byte)
```

รูปที่ 3.56 รูปแบบคำสั่ง ADD

2.7) คำสั่ง ADDC

รูปแบบคำสั่ง

' "ADDC A,@Ri"

หน้าที่

บวกค่าในรีจิสเตอร์ A กับค่าในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่และ Carry Flag ค่าที่ได้นำไปเก็บไว้ในรีจิสเตอร์ A

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์ นำข้อมูลในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่ บวกกับค่าในริจิสเตอร์A และบิตใน Carry Flag นำค่าที่ได้มาเปรียบเทียบว่ามีค่าเกิน 255 หรือไม่ถ้าเกิน ให้ลบออกจาก 256 แล้วนำค่าที่ได้เก็บในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.57

```
' "ADDC  A,@Ri"
Internal_ram(&HE0) = Internal_ram(&HE0) + _
Internal_ram(Internal_ram(bank + i)) + CF
location_byte = Val(&HE0)
Call check_up(location_byte)
```

รูปที่ 3.57 รูปแบบคำสั่ง ADDC

2.8) คำสั่ง SUBB

รูปแบบคำสั่ง

' "SUBB A,Rn"

หน้าที่

ลบค่าในรีจิสเตอร์ A ด้วยค่าในรีจิสเตอร์แบงค์ ค่า Carry Flag เก็บไว้ในรีจิสเตอร์ A

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์นำข้อมูลในรีจิสเตอร์แบงค์มาลบกับค่าในรีจิสเตอร์ A และ Carry Flag นำค่าที่ได้มาเปรียบเทียบว่ามีค่าน้อยกว่า 0 หรือไม่ ถ้าน้อยกว่าให้บวกกับ 256 และนำค่าที่ได้มาเก็บไว้ในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.58

```
' "SUBB  A,Rn"
Internal_ram(&HE0) = Val(Internal_ram(&HE0)) - Val(Internal_ram(bank + n))
If Internal_ram(&HE0) < 0 Then
    Internal_ram(&HE0) = 256 + Internal_ram(&HE0)
End If
```

รูปที่ 3.58 รูปแบบคำสั่ง SUBB

2.9) คำสั่ง SUBB

รูปแบบคำสั่ง

' "SUBB A,@Ri"

หน้าที่

ลบค่าในรีจิสเตอร์ A ด้วยค่าในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่ และค่าของ Carry Flag ค่าที่ได้ไปเก็บในรีจิสเตอร์ A

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์ นำข้อมูลในตำแหน่งรีจิสเตอร์แบงค์ชี้อยู่ไปลบกับค่าในรีจิสเตอร์ A และค่าของ Carry Flag นำค่าที่ได้มาเปรียบเทียบว่ามีค่าน้อยกว่า 0 หรือไม่ ถ้าน้อยกว่าให้บวกกับ 256 และนำค่าที่ได้มาเก็บไว้ในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.59

```
'"SUBB  A,@Ri"
Internal_ram(&HE0) = Val(Internal_ram(&HE0)) -
Val(Internal_ram(Internal_ram(bank + i))) - CF
If Internal_ram(&HE0) < 0 Then
    Internal_ram(&HE0) = 256 + Internal_ram(&HE0)
End If
```

รูปที่ 3.59 รูปแบบคำสั่ง SUBB

2.10) คำสั่ง INC

รูปแบบคำสั่ง

' "INC DPTR"

หน้าที่

เพิ่มค่าของ DPTR ขึ้นทีละ 1

การทำงาน

ดึงค่าจากรีจิสเตอร์ DPH และ DPL ออกมารวมกันเป็นเลข 16 บิต แล้วบวกค่าเพิ่มขึ้น 1 แล้วแยกเก็บข้อมูลโดย DPH เก็บข้อมูลไบต์สูงส่วน DPL เก็บข้อมูลไบต์ต่ำ สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.60

```
' "INC  DPTR"
If Internal_ram(&H82) < 15 Then
davedpl = "0" & Hex(Internal_ram(&H82))
Else
davedpl = Hex(Internal_ram(&H82))
End If
davedptr = Hex(Internal_ram(&H83)) & davedpl
Call StringAddress_Address(davedptr)
DPTR = Val(davedptr) + 1
davedptr = Hex(DPTR)
dave = "&H" & Mid(davedptr, 1, 2)
Internal_ram(&H83) = Val(dave)
dave = "&H" & Mid(davedptr, 3, 1)
If dave = "&H0" Then
dave = "&H" & Mid(davedptr, 4, 1)
End If
Internal_ram(&H82) = Val(dave)
```

รูปที่ 3.60 รูปแบบคำสั่ง INC

2.11) คำสั่ง CLR

รูปแบบคำสั่ง

' "CLR A"

หน้าที่

ทำให้ค่าในรีจิสเตอร์ A มีค่าเท่ากับ 0

การทำงาน

ทำให้ค่าในรีจิสเตอร์ A มีค่าเท่ากับ 0 สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.61

```
' "CLR  A"
Internal_ram(&HE0) = 0
```


รูปที่ 3. 61 รูปแบบคำสั่ง CLR

2.12) คำสั่ง DIV

รูปแบบคำสั่ง

' "DIV AB"

หน้าที่

นำข้อมูลในรีจิสเตอร์ A มาหารด้วยค่าในรีจิสเตอร์ B

การทำงาน

นำค่าในรีจิสเตอร์ A เป็นตัวตั้งหารด้วยค่าในรีจิสเตอร์ B จำนวนเต็มที่ได้เก็บไว้ในรีจิสเตอร์ A ส่วนเศษที่ได้จากการหารเก็บไว้ในรีจิสเตอร์ B สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.62

```
' "DIV  AB"
Dim A As Byte, B As Byte
A = Internal_ram(&HE0)
B = Internal_ram(&HF0)
Internal_ram(&HE0) = Int(A / B)
Internal_ram(&HF0) = Int(A Mod B)
```

รูปที่ 3.62 รูปแบบคำสั่ง DIV

2.13) คำสั่ง MUL

รูปแบบคำสั่ง

' "MUL AB"

หน้าที่

นำค่าในรีจิสเตอร์ A มาคูณกับค่าในรีจิสเตอร์ B

การทำงาน

นำค่าในรีจิสเตอร์ A เป็นตัวตั้งคูณด้วยค่าในรีจิสเตอร์ B ผลคูณที่ได้เป็นค่า 16 บิต 8 บิตบนเก็บในรีจิสเตอร์ A และ 8 บิตล่างเก็บในรีจิสเตอร์ B สามารถเขียนโปรแกนมจำลองการทำงานได้ดังรูปที่ 3.63

```
' "MUL  AB"
Internal_ram(&HE0) = Internal_ram(&HE0) * Internal_ram(&HF0)
davedptr = Hex(Internal_ram(&HE0))
If Len(davedptr) = 3 Then
davedptr = "0" & davedptr
End If
dave = "&H" & Mid(davedptr, 3, 2)
Internal_ram(&HE0) = Val(dave)
dave = "&H" & Mid(davedptr, 1, 1)
If dave = "0" Then
    dave = "&H" & Mid(davedptr, 2, 1)
Else
    dave = "&H" & Mid(davedptr, 1, 2)
End If
Internal_ram(&HF0) = Val(dave)
```

รูปที่ 3.63 รูปแบบคำสั่ง MUL

3) คำสั่งกระทำระดับบิต

3.1) คำสั่ง CLR หรือ SETB

รูปแบบคำสั่ง

' "CLR C"

' "SETB C"

หน้าที่

ทำให้บิต Carry Flag เป็น 0 หรือเป็น 1

การทำงาน

นำ Carry Flag มาทำให้เป็น 0 หรือเป็น 1 แล้วใส่ผลลงไปในรีจิสเตอร์ PSW สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.64

```
' "CLR  C"
   CF = 0

' "SETB  C"
   CF = 1
```

รูปที่ 3.64 รูปแบบคำสั่ง CLR หรือ SETB

3.2) คำสั่ง RL หรือ RR

รูปแบบคำสั่ง

' "RL A"

' "RR A"

หน้าที่

เลื่อนบิตทุกบิตในรีจิสเตอร์ A ไปทางซ้ายและให้บิต 0 เท่ากับ บิตที่ 7 หรือเลื่อนบิตทุกบิตไปทางขวามือให้บิตที่ 7 เท่ากับบิตที่ 0

การทำงาน

นำค่าในรีจิสเตอร์ออกมาแยกบิตแล้วทำการเลื่อนตำแหน่งโดยให้บิต 0 เท่ากับบิต 7, บิต 1 เท่ากับบิต 0, บิต 2 เท่ากับบิต 1 ไปเรื่อยๆ หรือเลื่อนตำแหน่งโดยให้บิต 7 เท่ากับบิต 0, บิต 6 เท่ากับบิต 7, บิต 5 เท่ากับบิต 6 ไปเรื่อยๆ แล้วแปลงเป็นเลขฐาน 10 นำค่าใส่ในรีจิสเตอร์ดังเดิม สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.65

```
' "RL  A"
acc = Internal_ram(&HE0)
Call hex_to_binary(acc)
For i = 0 To 6
        old(i + 1) = binary_number(i)
Next i
        old(0) = binary_number(7)
Internal_ram(&HE0) = 0
For i = 0 To 7
        Internal_ram(&HE0) = Internal_ram(&HE0) + (old(i) * (2 ^ i))
Next i

' "RR A"
acc = Internal_ram(&HE0)
Call hex_to_binary(acc)
For i = 1 To 7
        old(i - 1) = binary_number(i)
Next i
        old(7) = binary_number(0)
Internal_ram(&HE0) = 0
For i = 0 To 7
        Internal_ram(&HE0) = Internal_ram(&HE0) + (old(i) * (2 ^ i))
Next i
```

รูปที่ 3.65 รูปแบบคำสั่ง RL หรือ RR

3.3) คำสั่ง RRC หรือ RLC

รูปแบบคำสั่ง

' "RRC A"

' "RLC A"

หน้าที่

เลื่อนบิตทุกบิตในรีจิสเตอร์ A ไปทางซ้ายและบิตที่ 0 เท่ากับบิต Carry Flag และให้บิต Carry Flag มีค่าเท่ากับบิตที่ 7 หรือเลื่อนตำแหน่งโดยให้บิตทุกบิตไปทางขวา ให้บิตที่ 7 มีค่าเท่ากับ Carry Flag และบิต Carry Flag มีค่าเท่ากับบิต 0

การทำงาน

เลื่อนบิตทุกบิตในรีจิสเตอร์ A ไปทางซ้ายและบิตที่ 0 เท่ากับบิต Carry Flag และให้บิต Carry Flag มีค่าเท่ากับบิตที่ 7 หรือเลื่อนตำแหน่งโดยให้บิตทุกบิตไปทางขวา ให้บิตที่ 7 มีค่าเท่ากับ Carry Flag และบิต Carry Flag มีค่าเท่ากับบิต 0, บิตที่ 7 เท่ากับ Carry Flag หรือบิตที่ 0 เท่ากับ Carry Flag สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.66

```
' "RRC  A"
acc = Internal_ram(&HE0)
Call hex_to_binary(acc)
For i = 1 To 7
        old(i - 1) = binary_number(i)
Next i
        old(7) = CF
        CF = binary_number(0)
Internal_ram(&HE0) = 0

For i = 0 To 7
        Internal_ram(&HE0) = Internal_ram(&HE0) + (old(i) * (2 ^ i))
Next i

' "RLC  A"
acc = Internal_ram(&HE0)
Call hex_to_binary(acc)
For i = 0 To 6
         old(i + 1) = binary_number(i)
Next i
        old(0) = CF
        CF = binary_number(7)
        Internal_ram(&HE0) = 0
For i = 0 To 7
        Internal_ram(&HE0) = Internal_ram(&HE0) + (old(i) * (2 ^ i))
Next i
```

รูปที่ 3.66 รูปแบบคำสั่ง RRC หรือ RLC

3.4) คำสั่ง CPL

รูปแบบคำสั่ง

' "CPL C"

หน้าที่

สลับบิต C ให้มีค่าตรงกันข้าม

การทำงาน

ตรวจสภาวะลอจิกปัจจุบันของ Carry Flag แล้วทำการแปลงให้เป็นบิตตรงกันข้ามกับสภาวะลอจิกเดิม สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.67

```
' "CPL  C"
If CF = 1 Then
   CF = 0
Else
   CF = 1
End If
```

รูปที่ 3.67 รูปแบบคำสั่ง CPL

3.5) คำสั่ง CPL

รูปแบบคำสั่ง

'"CPL A"

หน้าที่

สลับบิตในรีจิสเตอร์ A ทุกบิตให้เป็นตรงกันข้าม

การทำงาน

นำข้อมูลเดิมมาแยกบิตและตรวจสภาวะลอจิกทั้ง 8 แล้วทำการแปลงให้เป็นบิตตรงกันข้ามจากนั้น รวมค่าเข้ามาเป็นฐาน 16 กับไปที่รีจิสเตอร์ A อีกครั้ง สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.68

```
' "CPL  A"
acc = Internal_ram(&HE0)
For i = 0 To 7
    binary_number(i) = acc Mod 2
    acc = Int(acc / 2)
    acc = Int(acc)
Next i
For i = 0 To 7
    If binary_number(i) = 0 Then
        binary_number(i) = 1
    Else
        binary_number(i) = 0
    End If
Next i
Internal_ram(&HE0) = 0
For i = 0 To 7
        Internal_ram(&HE0) = Internal_ram(&HE0) + (binary_number(i) * (2 ^ i))
Next i
```


รูปที่ 3. 68 รูปแบบคำสั่ง CPL

4) คำสั่งคำนวณทางตรรก

4.1) คำสั่ง ORL, ANL หรือ XRL

รูปแบบคำสั่ง

' "ORL A,Rn"

' "ANL A,Rn"

' "XRL A,Rn"

หน้าที่

นำข้อมูลในรีจิสเตอร์ A กระทำการ AND, OR หรือ XOR กับข้อมูลในรีจิสเตอร์แบงค์ แล้วผลเก็บไว้ที่รีจิสเตอร์ A

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์นำข้อมูลในรีจิสเตอร์ A กระทำการ AND กับข้อมูลในรีจิสเตอร์แบงค์ ผลที่ได้ไปเก็บไว้รีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.69

```
' "ORL  A,Rn"
Internal_ram(&HE0) = Internal_ram(&HE0) Or Internal_ram(bank + n)

'"ANL  A,Rn"
Internal_ram(&HE0) = Internal_ram(&HE0) And Internal_ram(bank + n)

' "XRL A,Rn"
Internal_ram(&HE0) = Internal_ram(&HE0) Xor Internal_ram(bank + n)
```

รูปที่ 3.69 รูปแบบคำสั่ง ORL, ANL หรือ XRL

4.2) คำสั่ง ORL, ANL หรือ XRL

รูปแบบคำสั่ง

' "ORL A,@Ri"

' "ANL A,@Ri"

' "XRL A,@Ri"

หน้าที่

นำข้อมูลรีจิสเตอร์ A กระทำการ AND, OR และ XOR กับข้อมูลในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่ แล้วผลเก็บไว้ในรีจิสเตอร์ A

การทำงาน

ตรวจค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์นำข้อมูลในรีจิสเตอร์ A กระทำการ AND กับข้อมูลในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่แล้วนำผลลัพธ์มาเก็บไว้ในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.70

```
' "ORL  A,@Ri"
Internal_ram(&HE0) = Internal_ram(&HE0) Or Internal_ram(Internal_ram(bank + i))

' "ANL  A,@Ri"
Internal_ram(&HE0) = Internal_ram(&HE0) And Internal_ram(Internal_ram(bank +
i))

' "XRL A,@Ri"
Internal_ram(&HE0) = Internal_ram(&HE0) Xor Internal_ram(Internal_ram(bank +
i))
```

รูปที่ 3.70 รูปแบบคำสั่ง ORL, ANL หรือ XRL

5) คำสั่งควบคุมการอ่านโปรแกรม

5.1) คำสั่ง LCALL หรือ ACALL

หน้าที่

ย้อนกลับไปยังตำแหน่งที่กระโดดมาทำฟังก์ชันย่อย

การทำงาน

นำค่าในตำแหน่งที่ SP ชี้อยู่ออกมาพักไว้ แล้วลดตำแหน่งที่ SP ชี้อยู่ลง 1 แล้วนำค่าออกมาอีกครั้ง นำค่าที่ได้มาเรียงต่อกันเป็นตำแหน่ง 2 ไบต์แล้วแปลงเป็นตำแหน่งของเลขฐาน 10 ใส่ค่าตำแหน่งที่ได้ใน PC เพื่อย้อนกลับไปทำงานต่อ สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.71

```
LcallS1 = Hex(Internal_ram(Internal_ram(&H81)))
If Len(LcallS1) < 2 Then
    LcallS1 = "0" & LcallS1
End If
Internal_ram(&H81) = Internal_ram(&H81) - 1

LcallS2 = Hex(Internal_ram(Internal_ram(&H81)))
If Len(LcallS2) = 1 Then
    LcallS1 = "0" & LcallS2
End If
Internal_ram(&H81) = Internal_ram(&H81) - 1
lcall = LcallS2 & LcallS1
davedptr = lcall
Call StringAddress_Address(davedptr)
PC = Val(davedptr)
```

รูปที่ 3.71 รูปแบบคำสั่ง LCALL หรือ ACALL

5.2) คำสั่ง JMP

รูปแบบคำสั่ง

' "JMP @A + DPTR"

หน้าที่

กระโดดไปยังตำแหน่งที่ DPTR และค่าในรีจิสเตอร์มาบวกกัน

การทำงาน

ให้ค่า PC มีค่าเท่ากับค่าของ DPTR และค่าในรีจิสเตอร์ A บวกกันสามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.72

```
'"JMP @A + DPTR"
PC = Internal_ram(&HE0) + DPTR
```

รูปที่ 3.72 รูปแบบคำสั่ง JMP

3.4.2 รายละเอียดการทำงานตามชุดคำสั่ง MCS-51 ที่เป็นคำสั่ง 2 ไบต์

1) คำสั่งเคลื่อนย้ายข้อมูล

1.1) คำสั่ง MOV

รูปแบบคำสั่ง

' "MOV A,#data"

' "MOV A,addr"

' "MOV addr,A"

หน้าที่

ให้ค่าในรีจิสเตอร์ A มีค่าเท่ากับค่าในตำแหน่ง Address หรือมีค่าเท่ากับ Data ที่กำหนด

การทำงาน

เรียกใช้ฟังก์ชันตรวจสอบ Address หรือ Data ถ้าเป็น Address นำค่าที่มีอยู่ใน Address มาใส่ไว้ในรีจิสเตอร์ A ถ้าเป็น Data ให้ค่าในรีจิสเตอร์มีค่าเท่ากับ Data สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.73

```
' "MOV A,#data"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = data

' "MOV  A,addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(addr)

' "MOV  addr,A"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addr) = Internal_ram(&HE0)
```

รูปที่ 3.73 รูปแบบคำสั่ง MOV

1.2) คำสั่ง MOV

รูปแบบคำสั่ง

' "MOV Rn,addr"

' "MOV addr,Rn"

' "MOV Rn,#data"

หน้าที่

ให้ค่าในรีจิสเตอร์แบงค์เท่ากับค่าที่มีอยู่ใน Address หรือให้ค่าใน Address เท่ากับค่าในรีจิสเตอร์แบงค์ หรือให้ค่าในรีจิสเตอร์แบงค์มีค่า Data ที่กำหนด

การทำงาน

เรียกใช้ฟังก์ชันตรวจสอบค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์ให้ค่าในรีจิสเตอร์แบงค์เท่ากับค่าที่อยู่ใน Address หรือมีค่าเท่ากับ Data ที่กำหนดหรือให้ค่าที่มีอยู่ใน Address เท่ากับค่าในรีจิสเตอร์แบงค์ สามารถเขียนโปรแกรมจำลองการทำงานได้ ดังรูปที่ 3.74

```
' "MOV  Rn,addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(bank + n) = Internal_ram(addr)

' "MOV addr,Rn"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addr) = Internal_ram(bank + n)


' "MOV Rn,#data"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(bank + n) = data
```

รูปที่ 3.74 รูปแบบคำสั่ง MOV

1.3) คำสั่ง MOV

รูปแบบคำสั่ง

' "MOV addr,@Ri"

' "MOV @Ri,#data"

' "MOV @Ri,addr"

หน้าที่

ให้ค่าในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่เท่ากับค่าในตำแหน่ง Address หรือ Data ที่กำหนดหรือให้ค่าในตำแหน่ง Address เท่ากับค่าในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่

การทำงาน

เรียกใช้ฟังก์ชันตรวจสอบค่าใน PSW เพื่อเลือกใช้งานในรีจิสเตอร์แบงค์ให้ค่าในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่เท่ากับ ค่าในตำแหน่ง Address หรือเท่ากับค่าของ Data ที่กำหนด

หรือให้ค่าที่อยู่ใน Address เท่ากับค่าในตำแหน่งที่รีจิสเตอร์แบงค์ชี้อยู่สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.75

```
'"MOV addr,@Ri"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addr) = Internal_ram(Internal_ram(bank + i))

' "MOV @Ri,#data"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(Internal_ram(bank + i)) = data

'"MOV @Ri,addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(Internal_ram(bank + i)) = Internal_ram(addr)
```

รูปที่ 3.75 รูปแบบคำสั่ง MOV

1.4) คำสั่ง MOV

รูปแบบคำสั่ง

'"MOV C,bit addr"

หน้าที่

ให้ค่าบิตใน Carry Flag มีค่าเท่ากับบิตในตำแหน่งบิต Address หรือมีค่าบิต Address เท่ากับบิตใน Carry Flag

การทำงาน

เรียกฟังก์ชันการตรวจสอบตำแหน่งบิต เพื่อตรวจตำแหน่งบิตที่ต้องการให้ค่าบิตใน Carry Flag มีค่าเท่ากับบิตในตำแหน่งบิต Address หรือให้ค่าในบิต Address เท่ากับค่าบิตใน Carry Flag สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.76

```
' "MOV  C,bit addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
    If bit_addr < 128 Then
        location_byte = Val(&H20)
        For i = 0 To bit_addr
            location_bit = location_bit + 1
            If location_bit = 8 Then
                location_byte = location_byte + 1
                location_bit = 0
            End If
        Next i
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
    Next i
        CF = old(location_bit)
        Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
Else
    dave_byte = Mid(Hex(bit_addr), 1, 1)
    dave_bit = Mid(Hex(bit_addr), 2, 1)
        If dave_bit > 7 Then
            dave_byte = "&H" & dave_byte & "8"
            location_byte = Val(dave_byte)
            dave_bit = "&H" & dave_bit
            location_bit = Val(dave_bit)
        Else
            dave_byte = "&H" & dave_byte & "0"
            location_byte = Val(dave_byte)
            location_bit = Val(dave_bit)
        End If

    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2
    Next i
            CF = old(location_bit)
            Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
End If
```

รูปที่ 3.76 รูปแบบคำสั่ง MOV

1.5) คำสั่ง PUSH หรือ POP

รูปแบบคำสั่ง

' "PUSH addr"

' "POP addr"

หน้าที่

เพิ่มค่า SP ขึ้น 1 แล้วนำข้อมูลที่ตำแหน่ง Address ไปใส่ไว้ในตำแหน่งที่ SP ชี้อยู่ หรือนำข้อมูลที่ตำแหน่งที่ SP ชี้อยู่ ไปเก็บในตำแหน่ง Address แล้วลดค่า SP ลง 1

การทำงาน

ตรวจสอบ Address เรียกฟังก์ชันการตรวจสอบหาค่า Address เพิ่มค่า SP ขึ้น 1 แล้วนำข้อมูลที่ตำแหน่ง Address ไปใส่ในตำแหน่งที่ SP ชี้อยู่หรือนำข้อมูลในตำแหน่งที่ SP ชี้อยู่ไปเก็บที่ตำแหน่ง Address แล้วลดค่า SP ลง 1 สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.77

```
' "PUSH  addr"
Internal_ram(&H81) = Internal_ram(&H81) + 1
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(Internal_ram(&H81)) = Internal_ram(addr)

' "POP  addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addr) = Internal_ram(Internal_ram(&H81))
Internal_ram(&H81) = Internal_ram(&H81) - 1
```

รูปที่ 3.77 รูปแบบคำสั่ง PUSH หรือ POP

1.6) คำสั่ง XCH

รูปแบบคำสั่ง

' "XCH A,addr"

หน้าที่

สลับข้อมูลในรีจิสเตอร์ A กับข้อมูลในตำแหน่ง Address

การทำงาน

ตรวจสอบ Address เรียกใช้ฟังก์ชันที่ตรวจสอบหาค่า Address แล้วพักข้อมูลของรีจิสเตอร์ และข้อมูลที่ตำแหน่ง Address ไว้แล้วให้สลับข้อมูลผ่านตัวพักข้อมูล โดยให้ค่าในรีจิสเตอร์ A มีค่าเท่ากับค่าในตำแหน่ง Address หรือให้ค่าในตำแหน่ง Address มีค่าเท่ากับค่าในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.78

```
' "XCH  A,addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
swapA = Internal_ram(&HE0)
swapbank = Internal_ram(addr)
Internal_ram(&HE0) = swapbank
Internal_ram(addr) = swapA
```

รูปที่ 3.78 รูปแบบคำสั่ง XCH

2) คำสั่งใช้คำนวณทางคณิตศาสตร์

2.1) คำสั่ง ADD

รูปแบบคำสั่ง

' "ADD A,#data"

' "ADD A,addr "

หน้าที่

บวก-ลบค่าในรีจิสเตอร์ A กับค่าในตำแหน่ง Address หรือค่าของ Data ผลที่ได้นำไปเก็บไว้ในรีจิสเตอร์ A

การทำงาน

เรียกฟังก์ชันตรวจสอบ Address หรือ Data ให้ค่าในรีจิสเตอร์ A มีค่าเท่ากับผลบวกหรือลบของรีจิสเตอร์ A หรือค่าในตำแหน่ง Address หรือค่าของ Data นำค่าที่ได้มาเปรียบเทียบว่ามีค่ามากกว่า 255 หรือไม่ถ้ามากกว่าให้นำมาลบกับ 256 แล้วนำผลลัพธ์ไปใส่ในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.79

```
'"ADD  A,#data"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) + data
location_byte = Val(&HE0)
Call check_up(location_byte)
' "ADD A,addr "
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) + Internal_ram(addr)
location_byte = Val(&HE0)
Call check_up(location_byte)
```


รูปที่ 3.79 รูปแบบคำสั่ง ADD

2.2) คำสั่ง ADDC หรือ SUBB

รูปแบบคำสั่ง

' "ADDC A,#data"

' "ADDC A,addr"

' "SUBB A,#data"

' "SUBB A,addr"

หน้าที่

บวกค่าในรีจิสเตอร์ A กับค่าในตำแหน่ง Address หรือค่าของ Data และบิต Carry Flag ผลที่ได้นำไปเก็บไว้ในรีจิสเตอร์ A

การทำงาน

เรียกใช้ฟังก์ชันตรวจสอบ Address หรือ Data ให้ค่าใน รีจิสเตอร์ A มีค่าเท่ากับผลบวกหรือผลลบของรีจิสเตอร์ A กับค่าในตำแหน่ง Address หรือค่าของ Data และค่าบิต Carry Flag นำค่าที่ได้มาเปรียบเทียบว่ามีค่ามากกว่า 255 หรือไม่ ถ้ามากกว่าให้นำมาลบกับ 256 แล้วนำผลลัพธ์ไปใส่ในรีจิสเตอร์ A สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.80

```
' "ADDC  A,#data"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) + data + CF
location_byte = Val(&HE0)
Call check_up(location_byte)

' "ADDC  A,addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) + Internal_ram(addr) + CF
location_byte = Val(&HE0)
Call check_up(location_byte)

' "SUBB  A,#data"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) - data - CF
If Internal_ram(&HE0) < 0 Then
    Internal_ram(&HE0) = 256 + Internal_ram(&HE0)
End If

' "SUBB  A,addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) - Internal_ram(addr) - CF
If Internal_ram(&HE0) < 0 Then
    Internal_ram(&HE0) = 256 + Internal_ram(&HE0)
End If
```


รูปที่ 3.80 รูปแบบคำสั่ง ADDC หรือ SUBB

2.3) คำสั่ง INC หรือ DEC

รูปแบบคำสั่ง

' "INC addr"

' "DEC addr"

หน้าที่

เพิ่มค่าในตำแหน่ง Address ขึ้น 1 หรือลดค่าในตำแหน่ง Address ลง 1

การทำงาน

เรียกใช้ฟังก์ชันตรวจสอบ Address นำข้อมูลเดิมในตำแหน่ง Address มาบวกเพิ่มกับ 1 หรือลบกับ 1 แล้วนำค่าที่ได้ไปใส่ในตำแหน่ง Address เดิม สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.81

```
' "INC addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addr) = Internal_ram(addr) + 1
location_byte = addr
Call check_up(location_byte)

' "DEC addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addr) = Internal_ram(addr) - 1
location_byte = addr
Call check_down(location_byte)
```

รูปที่ 3.81 รูปแบบคำสั่ง INC หรือ DEC

3) คำสั่งกระทำระดับบิต

3.1) คำสั่ง ORL, ANL หรือ XRL

รูปแบบคำสั่ง

"ORL addr,A"

' "ORL A,#data"

' "ORL A,addr"

' "ANL addr,A"

' "XRL A,addr"

' "ANL A,#data"

' "ANL A,addr"

' "XRL addr,A"

' "XRL A,#data"

หน้าที่

ให้ตำแหน่งบิต Address เท่ากับ 0 หรือเท่ากับ 1

การทำงาน

เรียกฟังก์ชันเพื่อตรวจสอบบิต Address แล้วทำให้ตำแหน่งบิต Address มีลอจิกเป็น "0" หรือเป็น "1" สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.82

```
"ORL  addr,A"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addr) = Internal_ram(addr) Or Internal_ram(&HE0)

'"ORL  A,#data"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) Or data

'"ORL  A,addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) Or Internal_ram(addr)

' "ANL addr,A"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addr) = Internal_ram(addr) Or Internal_ram(&HE0)

' "ANL  A,#data"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) Or data

' "ANL  A,addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addr) = Internal_ram(addr) Or Internal_ram(&HE0)

' "XRL addr,A"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addr) = Internal_ram(addr) Or Internal_ram(&HE0)

'"XRL A,#data"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) Xor data

' "XRL A,addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(&HE0) = Internal_ram(&HE0) Xor Internal_ram(addr)
```

รูปที่ 3.82 รูปแบบคำสั่ง ORL, ANL หรือ XRL

```
    Next i
        CF = CF Or old(location_bit)
        location_byte = Val(&HD0)
            Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
End If

'"ANL  C,bit addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
    If bit_addr < 128 Then
        location_byte = Val(&H20)
        location_byte = Val(&H20)
        For i = 0 To bit_addr
            location_bit = location_bit + 1
            If location_bit = 8 Then
                location_byte = location_byte + 1
                location_bit = 0
            End If
        Next i
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
    Next i
        CF = CF And old(location_bit)
        location_byte = Val(&HD0)
        Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
Else
    dave_byte = Mid(Hex(bit_addr), 1, 1)
    dave_bit = Mid(Hex(bit_addr), 2, 1)
        If dave_bit > 7 Then
            dave_byte = "&H" & dave_byte & "8"
            location_byte = Val(dave_byte)
            dave_bit = "&H" & dave_bit
            location_bit = Val(dave_bit)
        Else
            dave_byte = "&H" & dave_byte & "0"
            location_byte = Val(dave_byte)
            location_bit = Val(dave_bit)
        End If
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
    Next i
        CF = CF And old(location_bit)
        location_byte = Val(&HD0)
            Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
```

รูปที่ 3.83 (ต่อ) รูปแบบคำสั่ง ORL หรือ ANL

```
        Next i
End If

'"ORL  C,/bit addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
    If bit_addr < 128 Then
        location_byte = Val(&H20)
        location_byte = Val(&H20)
        For i = 0 To bit_addr

            location_bit = location_bit + 1
            If location_bit = 8 Then
                location_byte = location_byte + 1
                location_bit = 0
            End If

         Next i
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
    Next i
        If old(location_bit) = 0 Then
            old(location_bit) = 1
        Else
            old(location_bit) = 0
        End If
        CF = CF Or old(location_bit)
        location_byte = Val(&HD0)
        Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
Else
    dave_byte = Mid(Hex(bit_addr), 1, 1)
    dave_bit = Mid(Hex(bit_addr), 2, 1)
        If dave_bit > 7 Then
            dave_byte = "&H" & dave_byte & "8"
            location_byte = Val(dave_byte)
            dave_bit = "&H" & dave_bit
            location_bit = Val(dave_bit)
        Else
            dave_byte = "&H" & dave_byte & "0"
            location_byte = Val(dave_byte)
            location_bit = Val(dave_bit)
        End If
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
    Next i
        If old(location_bit) = 0 Then
            old(location_bit) = 1
        Else
            old(location_bit) = 0
        End If
        CF = CF Or old(location_bit)
        location_byte = Val(&HD0)
            Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
```


รูปที่ 3.83 (ต่อ) รูปแบบคำสั่ง ORL หรือ ANL

```
        Next i
End If

'"ANL  C,/bit addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
    If bit_addr < 128 Then
        location_byte = Val(&H20)
        location_byte = Val(&H20)
        For i = 0 To bit_addr
            location_bit = location_bit + 1


       If location_bit = 8 Then
                location_byte = location_byte + 1
                location_bit = 0
            End If
        Next i
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
    Next i

        CF = CF Or old(location_bit)
        location_byte = Val(&HD0)
        Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
Else
    dave_byte = Mid(Hex(bit_addr), 1, 1)
    dave_bit = Mid(Hex(bit_addr), 2, 1)
        If dave_bit > 7 Then
            dave_byte = "&H" & dave_byte & "8"
            location_byte = Val(dave_byte)
            dave_bit = "&H" & dave_bit
            location_bit = Val(dave_bit)
        Else
            dave_byte = "&H" & dave_byte & "0"
            location_byte = Val(dave_byte)
            location_bit = Val(dave_bit)
        End If
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
    Next i
        CF = CF Or old(location_bit)
        location_byte = Val(&HD0)
            Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
End If
```

รูปที่ 3.83 (ต่อ) รูปแบบคำสั่ง ORL หรือ ANL

4) คำสั่งกระทำระดับบิต

4.1) คำสั่ง CLR หรือ SETB

รูปแบบคำสั่ง

' "CLR bit addr"

' "SETB bit addr"

หน้าที่

AND, OR หรือ XOR ระหว่างค่าในรีจิสเตอร์ A กับค่าในตำแหน่ง Address หรือค่าของ Data ค่าที่ได้เก็บไว้ในรีจิสเตอร์ A หรือระหว่างค่าในตำแหน่ง Address กับรีจิสเตอร์ A หรือค่าของ Data แล้วนำค่าที่ได้เก็บไว้ในตำแหน่ง Address

การทำงาน

เรียกใช้ฟังก์ชันตรวจสอบ Address หรือ Data AND, OR หรือ XOR ระหว่างค่าในรีจิสเตอร์ A กับค่าในตำแหน่ง Address หรือค่าของ Data ค่าที่ได้เก็บไว้ที่รีจิสเตอร์ A หรือระหว่างค่าในตำแหน่ง Address กับรีจิสเตอร์ A หรือค่าของ Data และนำค่าที่ได้เก็บไว้ในตำแหน่ง Address สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.84

```
' "CLR  bit addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
    If bit_addr < 128 Then
        location_byte = Val(&H20)
        location_byte = Val(&H20)
        For i = 0 To bit_addr
            location_bit = location_bit + 1
            If location_bit = 8 Then
                location_byte = location_byte + 1
                location_bit = 0
            End If
        Next i
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
    Next i
        old(location_bit) = 0
        Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
Else
    dave_byte = Mid(Hex(bit_addr), 1, 1)
    dave_bit = Mid(Hex(bit_addr), 2, 1)
```


รูปที่ 3.84 รูปแบบคำสั่ง CLR หรือ SETB

```
        If dave_bit > 7 Then
            dave_byte = "&H" & dave_byte & "8"
            location_byte = Val(dave_byte)
            dave_bit = "&H" & dave_bit
            location_bit = Val(dave_bit)
        Else
            dave_byte = "&H" & dave_byte & "0"
            location_byte = Val(dave_byte)
            location_bit = Val(dave_bit)
        End If
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)

    Next i
            old(location_bit) = 0
            Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i

' "SETB  bit addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
    If bit_addr < 128 Then
        location_byte = Val(&H20)
        For i = 0 To bit_addr
            location_bit = location_bit + 1
            If location_bit = 8 Then
                location_byte = location_byte + 1
                location_bit = 0
            End If
        Next i
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)

        Next i
        old(location_bit) = 1
        Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
Else
    dave_byte = Mid(Hex(bit_addr), 1, 1)
    dave_bit = Mid(Hex(bit_addr), 2, 1)
        If dave_bit > 7 Then
            dave_byte = "&H" & dave_byte & "8"
            location_byte = Val(dave_byte)
            dave_bit = "&H" & dave_bit
            location_bit = Val(dave_bit)
        Else
            dave_byte = "&H" & dave_byte & "0"
            location_byte = Val(dave_byte)
            location_bit = Val(dave_bit)
        End If
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
```


รูปที่ 3.84 (ต่อ) รูปแบบคำสั่ง CLR หรือ SETB

```
    Next i
            old(location_bit) = 1
            Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
End If

End Select
```

รูปที่ 3.84 (ต่อ) รูปแบบคำสั่ง CLR หรือ SETB

4.2) คำสั่ง CPL

รูปแบบคำสั่ง

'"CPL bit addr"

หน้าที่

AND หรือ OR บิตระหว่าง Carry Flag กับบิตตำแหน่ง Bit Address ผลที่ได้เก็บไว้ที่ Carry Flag

การทำงาน

เรียกใช้ฟังก์ชันเพื่อตรวจสอบ Bit Address แล้วทำการ AND หรือ OR บิตระหว่าง Carry Flag กับบิตที่ตำแหน่ง Bit Address ผลที่ได้จากการประมวลจะใส่ไว้ใน Carry Flag สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.85

```
'"CPL  bit addr "
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
    If bit_addr < 128 Then
        location_byte = Val(&H20)
        For i = 0 To bit_addr
            location_bit = location_bit + 1
            If location_bit = 8 Then
                location_byte = location_byte + 1
                location_bit = 0
            End If
        Next i
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
```


รูปที่ 3.85 รูปแบบคำสั่ง CPL

```
    Next i
        If old(location_bit) = 0 Then
            old(location_bit) = 1

        Else
            old(location_bit) = 0
        End If
        Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
Else
    dave_byte = Mid(Hex(bit_addr), 1, 1)
    dave_bit = Mid(Hex(bit_addr), 2, 1)
        If dave_bit > 7 Then
            dave_byte = "&H" & dave_byte & "8"
            location_byte = Val(dave_byte)
            dave_bit = "&H" & dave_bit
            location_bit = Val(dave_bit)
        Else
            dave_byte = "&H" & dave_byte & "0"
            location_byte = Val(dave_byte)
            location_bit = Val(dave_bit)
        End If
    acc = Internal_ram(location_byte)
    For i = 0 To 7
        old(i) = acc Mod 2
        acc = Int(acc / 2)
    Next i
        If old(location_bit) = 0 Then
            old(location_bit) = 1
        Else
            old(location_bit) = 0
        End If
            Internal_ram(location_byte) = 0
        For i = 0 To 7
            Internal_ram(location_byte) = Internal_ram(location_byte) + (old(i)
* (2 ^ i))
        Next i
```

รูปที่ 3.85 (ต่อ) รูปแบบคำสั่ง CPL

5) คำสั่งควบคุมการอ่านโปรแกรม

5.1) คำสั่ง ACALL

รูปแบบคำสั่ง

' "ACALL addr"

หน้าที่

กระโดดไปทำคำสั่งย่อยตามตำแหน่งที่ Address

การทำงาน

เรียกใช้ฟังก์ชันเพื่อตรวจสอบบิต Address แล้วกระโดดไปทำคำสั่งย่อยตามตำแหน่งที่ Address ระบุสามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.86

```
' "ACALL addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)

lcall = Hex(PC)
LcallS1 = "&H" & Mid(lcall, 1, 2)
Lcall1 = Val(LcallS1)
Internal_ram(&H81) = Internal_ram(&H81) + 1
Internal_ram(Internal_ram(&H81)) = Lcall1
LcallS2 = "&H" & Mid(lcall, 3, 2)
If Mid(LcallS2, 3, 1) = "0" Then
    LcallS2 = "&H" & Mid(lcall, 4, 1)
End If
Lcall2 = Val(LcallS2)
Internal_ram(&H81) = Internal_ram(&H81) + 1
Internal_ram(Internal_ram(&H81)) = Lcall2

If addr < 16 Then
davedptr = Mid(Hex(PC), 1, 2) & "0" & Hex(addr)
Else
davedptr = Mid(Hex(PC), 1, 2) & Hex(addr)
End If

Call StringAddress_Address(davedptr)
PC = Val(davedptr)
```

รูปที่ 3.86 รูปแบบคำสั่ง ACALL

5.2) คำสั่ง AJMP

รูปแบบคำสั่ง

' "AJMP addr"

หน้าที่

กระโดดไปทำคำสั่งตามตำแหน่งที่ Address ระบุไว้โดยปราศจากข้อแม้

การทำงาน

ฟังก์ชันตรวจสอบบิต Address แล้วกระโดดไปทำคำสั่งตามตำแหน่งที่ Address ระบุ สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.87

```
' "AJMP addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
davedptr = Mid(Hex(PC), 1, 2)
If addr < 16 Then
davedptr = davedptr & "0" & Hex(addr)
Else
davedptr = davedptr & Hex(addr)
End If

Call StringAddress_Address(davedptr)
PC = Val(davedptr)
```


รูปที่ 3.87 รูปแบบคำสั่ง AJMP

5.3) คำสั่ง DNJZ

รูปแบบคำสั่ง

' "DNJZ Rn,addr"

หน้าที่

ลดค่าในรีจิสเตอร์แบงค์จนมีค่าเท่ากับ 0 หากยังไม่เป็น 0 ให้กระโดดไปยังตำแหน่ง Address

การทำงาน

เรียกใช้ฟังก์ชันเพื่อตรวจสอบบิต Address ลดค่าในรีจิสเตอร์แบงค์ลงจนกว่าจะเท่ากับ 0 หากมากกว่า 127 ให้กระโดดไปด้านหลังตรงข้ามกัน หาก Address น้อยกว่า 127 ให้กระโดดไปข้างหน้าตามตำแหน่งที่ระบุและทำคำสั่งต่อไปสามารถเขียนโปรแกรม จำลองการทำงานได้ดังรูปที่ 3.88

```
' "DNJZ Rn,addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(bank + n) = Internal_ram(bank + n) - 1

If Internal_ram(bank + n) < 0 Then
    Internal_ram(bank + n) = 256 + Internal_ram(bank + n)
    Internal_ram(bank + n) = Internal_ram(bank + n) - 1
End If

If Internal_ram(bank + n) > 0 Then
    If addr >= Val(&H80) Then
        addr = 256 - addr
        PC = PC - addr
    Else
        PC = PC + addr
    End If
End If
```

รูปที่ 3.88 รูปแบบคำสั่ง DNJZ

5.4) คำสั่ง JC, JNC, JZ หรือ JNZ

รูปแบบคำสั่ง

' "JC addr"

' "JNC addr"

' "JZ addr"

' "JNZ addr"

หน้าที่

กระโดดไปทำงานหากบิตตำแหน่ง, บิต Address, Carry Flag หรือค่าในรีจิสเตอร์ A เป็น 0 หรือ 1 ยังตำแหน่ง Address

การทำงาน

เรียกใช้ฟังก์ชันเพื่อตรวจสอบบิต Address กระโดดไปทำงานหากบิตตำแหน่ง, บิต Address, Carry Flag หรือค่าในรีจิสเตอร์ A เป็น 0 หรือ 1 ยังตำแหน่ง Address หาก Address น้อยกว่า 127 ให้กระโดดไปข้างหน้า ตามตำแหน่งที่ระบุสามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.89

```
' "JC  addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
If CF = 1 Then
     If addr >= Val(&H80) Then
         addr = 256 - addr
         PC = PC - addr
     Else
         PC = PC + addr
     End If
End If

' "JNC addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
If CF = 0 Then
     If addr >= Val(&H80) Then
         addr = 256 - addr
         PC = PC - addr
     Else
         PC = PC + addr
     End If
End If

' "JZ addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
If Internal_ram(&HE0) = 0 Then
     If addr >= Val(&H80) Then
         addr = 256 - addr
         PC = PC - addr
     Else
         PC = PC + addr
     End If
End If

' "JNZ addr"
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
```

รูปที่ 3.89 รูปแบบคำสั่ง JC, JNC, JZ หรือ JNZ

```
If Internal_ram(&HE0) <> 0 Then
    If addr >= Val(&H80) Then
        addr = 256 - addr
        PC = PC - addr
    Else
        PC = PC + addr
    End If
End If
```

รูปที่ 3.89 (ต่อ) รูปแบบคำสั่ง JC, JNC, JZ หรือ JNZ

5.5) คำสั่งกระโดด

หน้าที่

กระโดดไปยังตำแหน่งที่ระบุ

การทำงาน

เรียกใช้ฟังก์ชันเพื่อตรวจสอบบิต Address แล้วกระโดดไปทำคำสั่งที่ระบุโดย Address สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.90

```
Call check_operand(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
    If addr >= Val(&H80) Then
        addr = 256 - addr
        PC = PC - addr
    Else
        PC = PC + addr
    End If
```

รูปที่ 3.90 รูปแบบคำสั่งกระโดด

3.4.3 รายละเอียดการทำงานตามชุดคำสั่ง MCS-51 ที่เป็นคำสั่ง 3 ไบต์

1) คำสั่งเคลื่อนย้ายข้อมูล

1.1) คำสั่ง MOV

รูปแบบคำสั่ง

"MOV addr,#data"

"MOV addr,addr"

หน้าที่

ให้ข้อมูลในตำแหน่ง Address2 มีค่าเท่ากับข้อมูลใน Address1 หรือให้ข้อมูลในตำแหน่ง Address มีค่าเท่ากับค่าของ Data

การทำงาน

เรียกใช้ฟังก์ชันเพื่อตรวจสอบ Data และ Address และให้ข้อมูลใน Address2 มีค่าเท่ากับ ข้อมูลใน Address1 หรือ Address มีค่าเท่ากับค่าของ Data สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.91

```
' "MOV addr,#data"
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addrA) = data

'"MOV addr,addr"
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addrB) = Internal_ram(addrA
```

รูปที่ 3.91 รูปแบบคำสั่ง MOV

1.2) คำสั่ง MOV

รูปแบบคำสั่ง

' "MOV DPTR,#data16"

หน้าที่

ให้ค่าใน DPTR มีค่าเท่ากับข้อมูล 16 บิต

การทำงาน

เรียกใช้ฟังก์ชันเพื่อตรวจสอบ Data1 และเรียกใช้ฟังก์ชันเพื่อตรวจสอบ Data ซ้ำอีกครั้งนำ Data ที่เรียงต่อกันเป็นข้อมูลขนาด 16 บิต เก็บไว้ใน DPTR สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.92

```
' "MOV  DPTR,#data16"
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
dave = "&H" & data16
Internal_ram(&H83) = Val(dave)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
dave = "&H" & Right(data16, 2)
Internal_ram(&H82) = Val(dave)
davedptr = data16
Call StringAddress_Address(davedptr)
DPTR = Val(davedptr)
```

รูปที่ 3.92 รูปแบบคำสั่ง MOV

2) คำสั่งคำนวณทางคณิตศาสตร์

2.1) คำสั่ง ORL, ANL หรือ XRL

รูปแบบคำสั่ง

' "ORL addr,#data"

' "ANL addr,#data"

' "XRL addr,#data"

หน้าที่

ทำการ AND, OR หรือ XOR ข้อมูลที่อยู่ในตำแหน่ง Address กับค่าของ Data ผลที่ได้เก็บไว้ในตำแหน่ง Address

การทำงาน

เรียกฟังก์ชันเพื่อตรวจสอบ Address และ Data นำค่าในตำแหน่ง Address มา AND, OR หรือ XOR กับค่าของ Data แล้วนำค่าเก็บไว้ในตำแหน่ง Address สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.93

```
'"ORL addr,#data"
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addrA) = Internal_ram(addrA) Or data

' "ANL addr,#data"
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addrA) = Internal_ram(addrA) And data

' "XRL addr,#data"
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addrA) = Internal_ram(addrA) Xor data
```

รูปที่ 3.93 รูปแบบคำสั่ง ORL, ANL หรือ XRL

3) คำสั่งควบคุมการอ่านโปรแกรม

3.1) คำสั่ง CJNE

รูปแบบคำสั่ง

' "CJNE A,#data,addr"

' "CJNE A,addr,addr"

' "CJNE Rn,#data,addr"

หน้าที่

ทำการเปรียบเทียบค่าที่กำหนดไว้ หรือค่าในตำแหน่งที่กำหนดไว้กับอีกตำแหน่ง ว่าเท่ากันหรือไม่หากไม่เท่ากันให้กระโดดไปยังตำแหน่งที่ระบุ

การทำงาน

เรียกใช้ฟังก์ชันเพื่อตรวจสอบ Address และ Data และทำการเปรียบเทียบค่าที่กำหนดไว้หรือค่าในตำแหน่งที่ระบุกับตำแหน่งว่าเท่ากันหรือไม่ หากไม่เท่ากันให้กระโดดไปยังตำแหน่งปัจจุบันเป็นตำแหน่งใหม่ ที่ได้จากการคำนวณ สามารถเขียนโปรแกรมจำลองการได้ดังรูปที่ 3.94

```
' "CJNE A,#data,addr"
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
If Not Internal_ram(&HE0) = data Then
    If addrB >= Val(&H80) Then
        addr = 256 - addrB
        PC = PC - addr
    Else
        PC = PC + addrB
    End If
Else
    CF = 1
End If

' "CJNE A,addr,addr"
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
If Internal_ram(&HE0) <> Internal_ram(addrA) Then

    If addrB >= Val(&H80) Then
        addr = 256 - addrB
        PC = PC - addr
    Else
        PC = PC + addrB
    End If
    CF = 1
End If

' "CJNE Rn,#data,addr"
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
If Internal_ram(bank + n) <> data Then

    If addrB >= Val(&H80) Then
        addr = 256 - addrB
        PC = PC - addr
    Else
        PC = PC + addrB
    End If
    CF = 1
End If
```

รูปที่ 3.94 รูปแบบคำสั่ง CJNE

3.2) คำสั่ง DJNZ

รูปแบบคำสั่ง

' "DJNZ addr,addr"

หน้าที่

ลดค่าในตำแหน่ง Address ลงเรื่อยๆ จนเป็น 0 หากไม่เป็น 0 ให้กระโดดไปทำคำสั่งในตำแหน่งที่ระบุ จากการคำนวณ

การทำงาน

เรียกใช้ฟังก์ชันเพื่อตรวจสอบ Address และลดค่าในตำแหน่ง Address ลงเรื่อยๆ จนกว่าจะเท่ากับ 0 หากไม่เท่าให้กระโดดไปยังตำแหน่งที่สัมพัทธ์กับตำแหน่งปัจจุบัน สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.95

```
' "DJNZ  addr,addr"
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Internal_ram(addrA) = Internal_ram(addrA) - 1

If Internal_ram(addrA) < 0 Then
    Internal_ram(addrA) = Internal_ram(addrA) + 1
End If


If Internal_ram(addrA) > 0 Then
    If addrB >= Val(&H80) Then
        addr = 256 - addrB
        PC = PC - addr
    Else
        PC = PC + addrB
    End If
    CF = 1

End If
```

รูปที่ 3.95 รูปแบบคำสั่ง DJNZ

3.3) คำสั่งกระโดดไปทำคำสั่งเกี่ยวกับ Address 16

หน้าที่

ให้ PC กระโดดไปอ่านคำสั่งตามที่ Address 16 ระบุ

การทำงาน

เรียกใช้ฟังก์ชันตรวจสอบ Address 16 นำค่า Address 16 ใส่เข้าไปในค่า PC จากนั้น PC จะกระโดดไปอ่านคำสั่งที่ตำแหน่ง Address 16 สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.96

```
Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
davedptr = addr16
Call StringAddress_Address(davedptr)
PC = Val(davedptr)
```

รูปที่ 3.96 รูปแบบคำสั่งกระโดดไปทำคำสั่งเกี่ยวกับ Address 16

3.4) คำสั่งกระโดดไปทำคำสั่งย่อยเกี่ยวกับ Address 16

หน้าที่

ให้ PC กระโดดไปอ่านคำสั่งตามที่ตำแหน่ง Address 16 ระบุและเมื่ออ่านจบก็จะกระโดดมาตำแหน่งที่กระโดดไป

การทำงาน

บันทึกตำแหน่งที่ 2 ไบต์ โดยใช้การทำงานของ SP และกระโดดไปยังตำแหน่ง Address 16 กระทำคำสั่งย่อยจนกว่าจะครบคำสั่งย้อนกลับสามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.97

```
lcall = Hex(PC + 2)
LcallS1 = "&H" & Mid(lcall, 1, 2)
Lcall1 = Val(LcallS1)
Internal_ram(&H81) = Internal_ram(&H81) + 1
Internal_ram(Internal_ram(&H81)) = Lcall1

LcallS2 = "&H" & Mid(lcall, 3, 2)
Lcall2 = Val(LcallS2)
Internal_ram(&H81) = Internal_ram(&H81) + 1
Internal_ram(Internal_ram(&H81)) = Lcall2

Call check_operand31(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
Call check_operand32(program, i, data, data16, addr, addr16, bit_addr, _
dave, addrA, addrB)
davedptr = addr16
Call StringAddress_Address(davedptr)
PC = Val(davedptr)
```

รูปที่ 3.97 รูปแบบคำสั่งกระโดดไปทำคำสั่งย่อยเกี่ยวกับ Address 16

3.4.4 คำสั่งแปลงเลขฐาน 10 เ ป็นเลขฐาน 2

หน้าที่

นำเลขฐาน 10 มาแปลงเป็นเลขฐาน 2 จำนวน 8 บิต

การทำงาน

นำเลขฐาน 10 มาหารแบบคิดเศษ ซึ่งเศษในครั้งแรกจะใส่ในบิตที่ 0 โดยจะทำการหารทั้งหมด 8 ครั้ง สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.98

```
Dim i As Byte
acc = internal_ram(address)
For i = 0 To 7
    binary_number(i) = acc Mod 2
    acc = Int(acc / 2)
Next i
```

รูปที่ 3.98 รูปแบบคำสั่งแปลงเลขฐาน 10 เป็นเลขฐาน 2

3.4.5 คำสั่งแปลงข้อความของเลขฐาน 16 (2 ไบต์) เป็นฐาน 10

หน้าที่

นำข้อความที่แทนตำแหน่งหน่วยความจำโปรแกรมภายนอก แปลงเป็นตำแหน่งหน่วยความจำโปรแกรมที่เป็นเลขฐาน 10

การทำงาน

ใช้ฟังก์ชั่นที่มีอยู่ใน Visual Basic โดยทำให้ข้อความแปลงเป็นจำนวนของเลขฐาน 10 ตรวจดูค่าที่ได้ว่ามีค่าน้อยกว่า 0 หรือไม่หากน้อยกว่า ให้นำจำนวนที่ได้มารวมกับค่า 32769 ก่อนแล้วนำไปบวกเพิ่มกับ 32767 สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูปที่ 3.99

```
Dim dave As Variant
davedptr = String Address
dave = "&H" & davedptr
If Val(dave) < 0 Then
    dave = 32767 + (32769 + Val(dave))
End If
davedptr = Val(dave)
```


รูปที่ 3.99 รูปแบบคำสั่งแปลงข้อความจากเลขฐาน 16 (2 ไบต์) เป็นฐาน 10

3.4.6 คำสั่งตรวจสอบความเปลี่ยนแปลงใน PSW

หน้าที่

ตรวจสภาวะลอจิกในตำแหน่ง PSW (D0H) แล้วแสดงผล

การทำงาน

นำค่าในตำแหน่ง PSW ออกมาแปลงเป็นเลขฐาน 2 โดยเรียกใช้ฟังก์ชันแปลงเลขฐาน 2 นำสภาวะของบิตต่างๆ ที่เปลี่ยนแปลงแต่ละคำสั่งแทนที่บิตข้อมูล ที่เปิดขึ้นมา และหาการใช้งานรีจิสเตอร์แบงค์จากนั้น นำค่าใหม่ที่ได้ใส่กับคืนไปที่ตำแหน่ง PSW (D0H) ตามเดิม สามารถเขียนโปรแกรมจำลองการทำงานได้ดังรูป 3.100

```
Public Sub check_psw(bank)
Dim i As Byte

accumu = Internal_ram(&HD0)
For i = 0 To 7
    binary_number(i) = accumu Mod 2
    accumu = accumu / 2
    accumu = Int(accumu)
Next i

'ตรวจ parity
For i = 0 To 7
    parity_bit = parity_bit + binary_number(i)
Next i
    parity_bit = parity_bit Mod 2
    binary_number(0) = parity_bit
'*********************************************

'ตรวจ carry flag
    binary_number(7) = CF
'*********************************************

'ตรวจ Auxilily carry
    binary_number(6) = AC
'*********************************************

'ตรวจ bank
    If binary_number(3) = 0 Then

        If binary_number(4) = 0 Then
            bank = bank0
        Else
            bank = bank2
        End If

    Else
```

รูปที่ 3.100 รูปแบบคำสั่งตรวจสอบความเปลี่ยนแปลงใน PSW

```
    If binary_number(4) = 0 Then
            bank = bank1
        Else
            bank = bank3
        End If

    End If
'****************************************
Internal_ram(&HD0) = 0
For i = 0 To 7
    Internal_ram(&HD0) = Internal_ram(&HD0) + (binary_number(i) * (2 ^ i))
Next i

End Sub
```

รูปที่ 3.100 (ต่อ) รูปแบบคำสั่งตรวจสอบความเปลี่ยนแปลงใน PSW

3.5 การออกแบบอุปกรณ์ต่อพ่วงและส่วนของ MCS-51

3.5.1 การออกแบบและการสร้างบอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I)

การออกแบบและการสร้างบอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) นั้นได้คำนึงถึงการใช้งานหลักใหญ่ๆ 2 ประการคือ

1) เพื่อใช้ติดต่อสื่อสารระหว่างบอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) และคอมพิว - เตอร์ในการส่งโปรแกรมจากคอมพิวเตอร์ ไปยังบอร์ดไมโครคอนโทรลเลอร์(TEPSA-I) และสั่งให้บอร์ดไมโครคอนโทรลเลอร์(TEPSA-I)

2) เพื่อให้บอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) ทำงานตามโปรแกรมที่ได้รับจากบอร์ดไมโครคอนโทรลเลอร์(TEPSA-I) โดยทำงานร่วมกันกับอุปกรณ์ต่อพ่วงที่สร้างขึ้น

ดังนั้นในการออกแบบและการสร้างบอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) จึงจะต้องออกแบบตามการใหญ่ๆ สองประการนี้ได้แก่

3.5.2 การออกแบบโปรแกรมรีโมทมอร์นิเตอร์

โปรแกรมรีโมทมอร์นิเตอร์ (RemoteMonitor) จะเป็นโปรแกรมที่ใช้ในการติดต่อสื่อสารระหว่างบอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) และคอมพิวเตอร์ ขั้นตอนการออกแบบโปรแกรมรีโมทมอนิเตอร์นั้นเริ่มจากต้องเข้าใจมาตรฐานของข้อมูลที่ใช้ในการส่งผ่านพอร์ตอนุกรม ในโครงงานนี้ใช้ลักษณะการจัดเรียงข้อมูลแบบฟอร์แมตแบบอินเทล (Intel Hex'-Format) โปรแกรมโค้ดทั้งหมดจะถูกแบ่งออกเป็นบล็อกๆ ในแต่ละบล็อกจะประกอบไปด้วยบรรทัดหนึ่งซึ่งกำหนดจำนวนไบต์ นอกจากนั้นยังมีการเพิ่มจำนวนไบต์ต่อไปอีก ลักษณะดังตัวอย่างต่อไปนี้

: 04 0000 00 740C 24 05 53 บรรทัดของตัวข้อมูล

: 00 0000 01 FF บรรทัดจบ

สามารถอธิบายความหมายของแต่ละไบต์ของรูปแบบข้อมูลของอินเทลได้ดังนี้

: อักษรตัวแรกดับเบอร์พอยต์ (Double Point)

04 ขนาดหรือความยาวของไบต์ข้อมูล

0000 แอดเดรสขนาด 16 บิต (2ไบต์) สำหรับข้อมูลไบต์แรก

00 เป็นตัวบอกว่าบรรทัดนี้คือข้อมูลถ้าเป็น 01 คือบรรทัดสิ้นสุด (จบ)

740C 2405 ไบต์ข้อมูล (ตัวโปรแกรมโค้ดไบนารี)

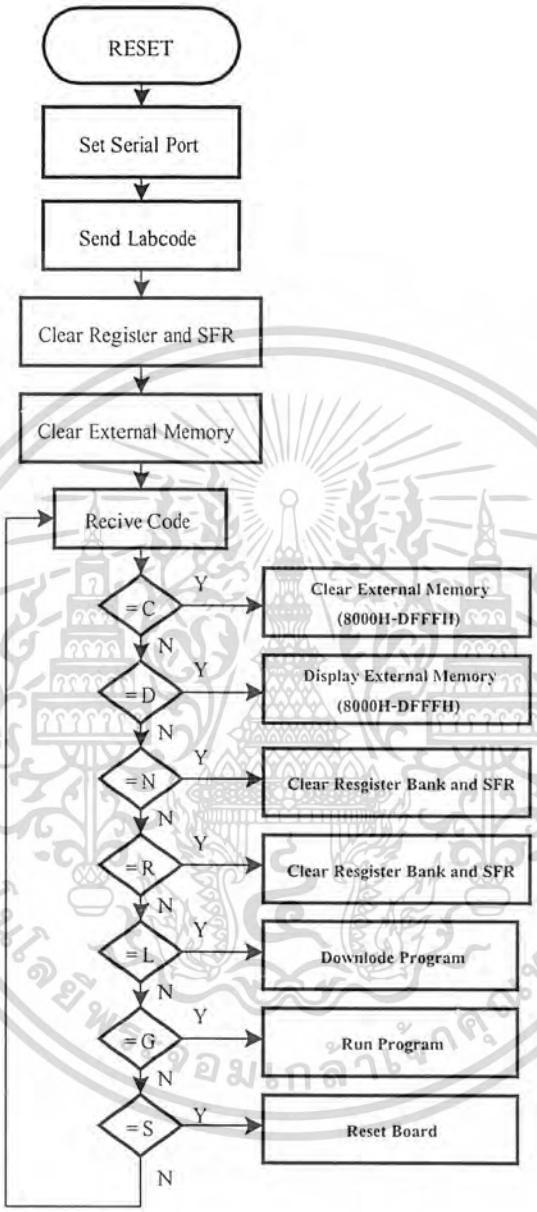
53 ตรวจสอบผลรวม (ผลรวมทุกๆ ไบต์ในบรรทัดนี้ต้องมีค่าเท่ากับ XX00)

ทุกๆ บรรทัดจะขึ้นต้นด้วยดับเบิลพอร์ตไบต์ถัดไปจะจะบอกถึงความยาวของไบต์ข้อมูลในบรรทัดนั้นๆ ตามตัวอย่างนี้ไบต์ข้อมูลคือ 74, 0C, 24, และ 05 ความยาวข้อมูลคือ 04 จากนั้นจึงตามด้วยตำแหน่งแอดเดรสเริ่มต้นของไบต์ข้อมูลแรก (74)

สำหรับโปรแกรมรีโมทมอนิเตอร์นี้ได้สร้างให้มีความสามารถติดต่อกันระหว่างบอร์ดบอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) และคอมพิวเตอร์ตามสัญลักษณ์นำหน้าเพื่อเลือกบริการของโปรแกรมรีโมทมอนิเตอร์ได้ดังนี้

สัญลักษณ์รหัสแอสกี้	บริการของโปรแกรมรีโมทมอนิเตอร์
“L”	Download Program
“G”	Run Program
“D”	Display External Memory (8000H-DFFFH)
“C”	Clear External Memory (8000H-DFFFH)
“R”	Display Register Bank and SFK
“S”	Reset Board

การทำงานของโปรแกรมรีโมทมอนิเตอร์เขียนลำดับขั้นตอนการทำงานได้ดังรูปที่ 3.101

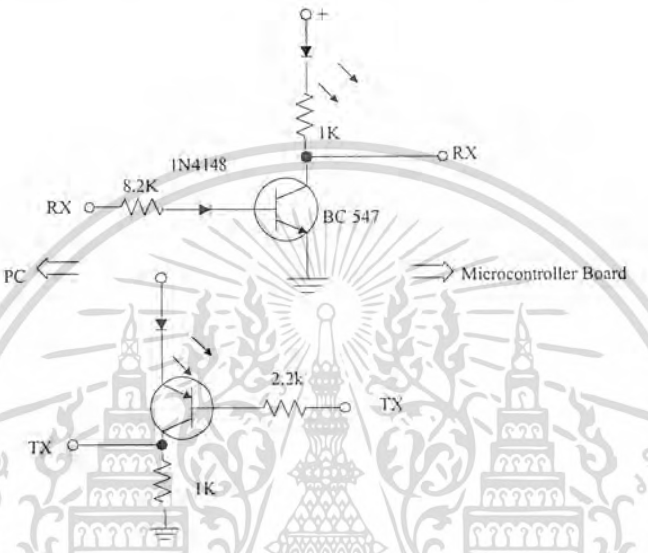


รูปที่ 3.101 ผังงานของโปรแกรมรีโมทมอนิเตอร์

3.5.3 การออกแบบวงจรบอร์ดไมโครคอนโทรลเลอร์

การออกแบบต้องกำหนดมาตรฐานหรือขีดความสามารถของวงวรบอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) ว่าต้องการให้ทำงานในขอบเขตใดบ้าง สำหรับในโครงานนี้ได้กำหนดขีดความสามารถของวงจรบอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) ให้มีคุณสมบัติดังต่อไปนี้

1) สามารถติดต่อสื่อสารกับคอมพิวเตอร์ได้ทางพอร์ตอนุกรม โดยเลือกมาตรฐานการติดต่อแบบ RS-232 ใช้อุปกรณ์ทรานซิสเตอร์เป็นตัวปรับแรงดันระหว่างคอมพิวเตอร์ และวงจรบอร์ดไมโครคอนโทรลเลอร์(TEPSA-I)ให้เป็นมาตรฐานเดียวกัน วงจรแสดงดังรูปที่ 3.102



รูปที่ 3.102 วงจรปรับค่าแรงดันทางพอร์ตอนุกรม

2) เลือกใช้หน่วยประมวลผลกลางเป็นไอซีตระกูล MCS-51 เบอร์ 89S53 มีคุณสมบัติเฉพาะดังนี้

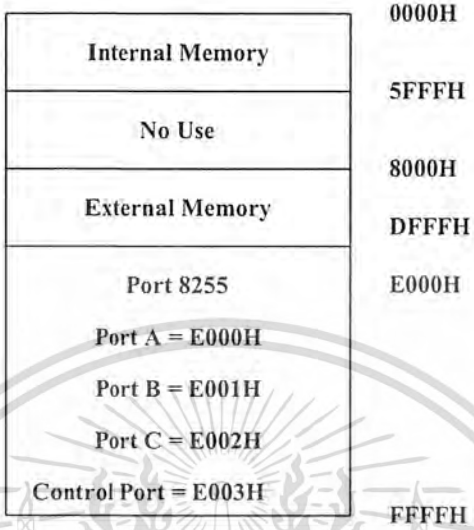
2.1) ทำงานในมาตรฐานตระกูล MCS-51

2.2) หน่วยความจำโปรแกรมภายในจำนวน 12 กิโลไบต์ และสามารถเขียนโปรแกรมผ่านทางพอร์ตอนุกรมได้

2.3) EEPROM จำนวน 2 กิโลไบต์

3) หน่วยความจำภายนอก เลือกใช้ชนิดแรมเบอร์ 62256 มีหน่วยความจำที่ใช้งานได้จำนวน 32 กิโลไบต์

4) เนื่องจากมีการแบ่งช่วงของหน่วยความจำภายใน และหน่วยความจำภายนอก ออกจากกันจึงต้องมีอุปกรณ์ทำหน้าที่ถอดรหัสเพื่อแบ่งแยกหน่วยความจำ เลือกใช้ไอซีเบอร์ 74LS138 สามารถเขียนเป็นแผนที่หน่วยความจำได้ดังรูปที่ 3.103

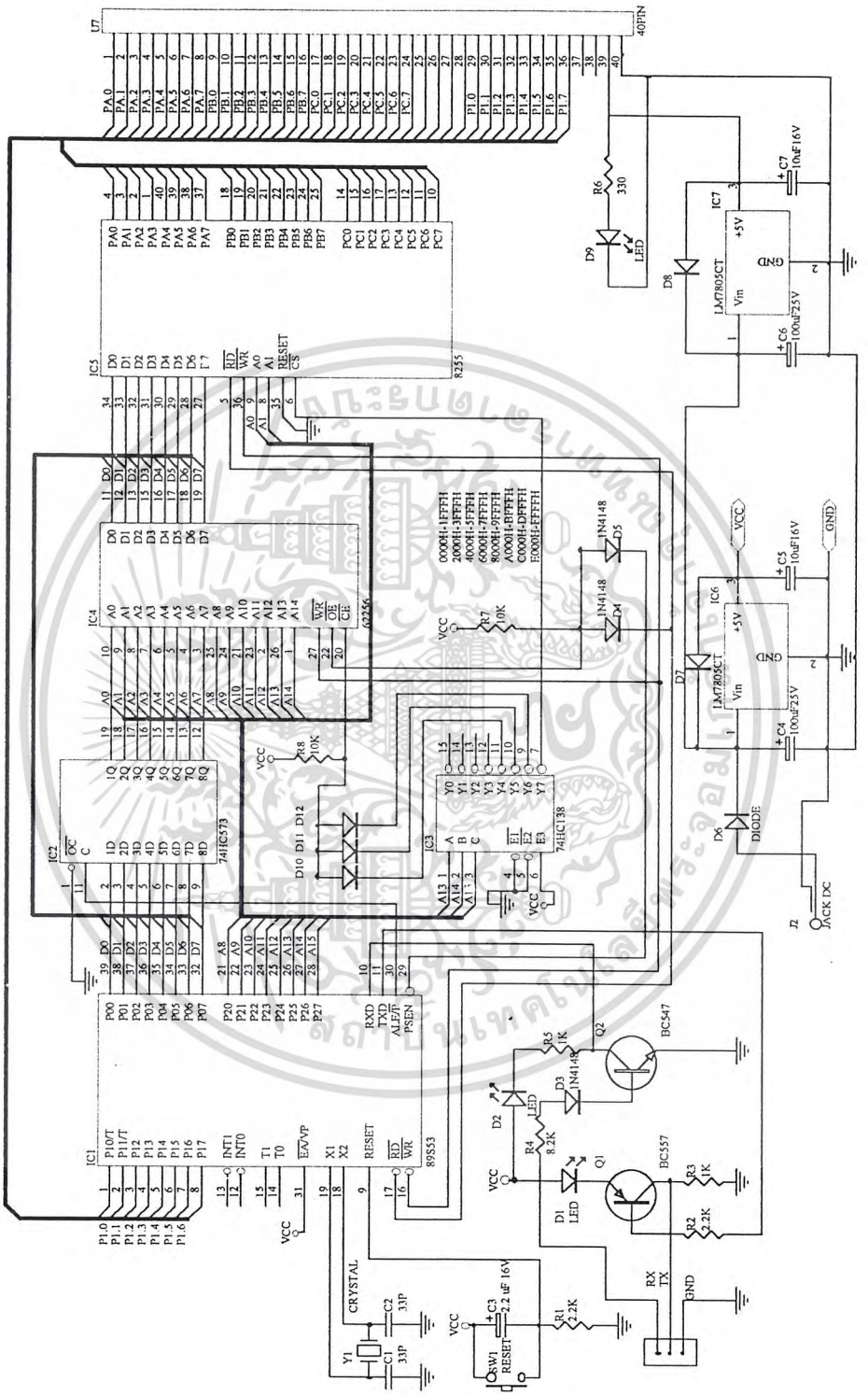


รูปที่ 3.103 แผนที่หน่วยความจำ ไอซีเบอร์ 74LS138

5) ใช้อุปกรณ์ขยายพอร์ตเบอร์ 8255 สามารถขยายให้วงจรไมโครคอนโทรลเลอร์ใช้งานควบคุมได้ถึง 3 พอร์ต แต่ละพอร์ตประกอบด้วย 8 บิต จากรูปที่ 3.103 จะเห็นได้ว่ามีการกำหนดพอร์ตของ 8255 เริ่มต้นที่ตำแหน่งหน่วยความจำ E000H จนถึง FFFFH

ในตำแหน่งของอุปกรณ์ขยายพอร์ตเบอร์ 8255 มีความจำเป็นต้องแยกออกมาจากตำแหน่งหน่วยความจำภายนอกไม่สามารถซ้อนทับได้ ดังนั้นในการอ้างอิงตำแหน่งหน่วยความจำภายนอกตั้งแต่ที่ตำแหน่ง E000H จนถึง FFFFH จึงไม่สามารถทำได้ เพราะตำแหน่งของพอร์ตทั้ง 3 พอร์ตแล้วเลือกใช้เพียงควบคุมหรือติดต่อกับอุปกรณ์ภายนอกเท่านั้น

บอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) ออกแบบให้มีแหล่งจ่ายไฟ 2 แหล่งคือจ่ายให้กับวงจรบอร์ดไมโครคอนโทรลเลอร์เองและอีกส่วนหนึ่งจ่ายให้กับวงจรของอุปกรณ์ต่อพ่วงวงจรสมบูรณ์ของวงจรบอร์ดไมโครคอนโทรลเลอร์แสดงในรูปที่ 3.104

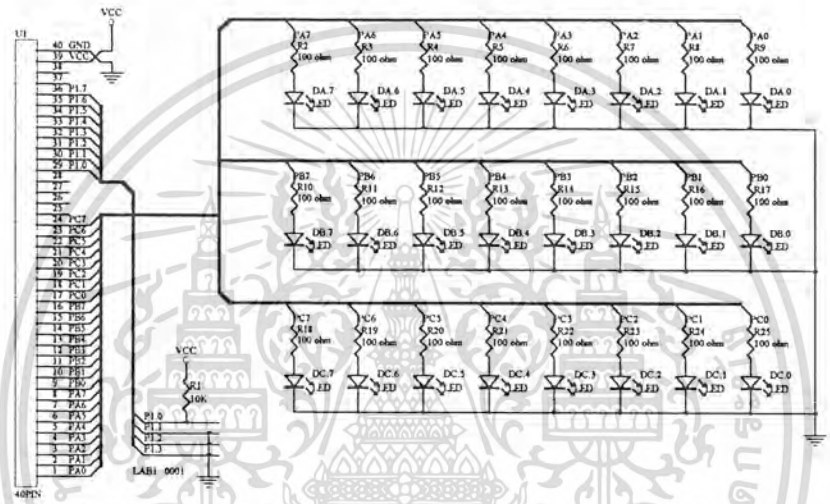


รูปที่ 3.104 วงจรของบอร์ดไมโครคอนโทรลเลอร์

3.5.4 การออกแบบและการสร้างอุปกรณ์ต่อพ่วง

1) การออกแบบวงจรทดลองที่ 1 เรื่องการเชื่อมต่อกับแอลอีดีลอจิก

วงจรทดลองที่ 1 ออกแบบให้ใช้งานในการทดลองสถานะพอร์ตทั้ง 3 พอร์ต ของวงจรไมโครคอนโทรลเลอร์คือ พอร์ต A, พอร์ต B และพอร์ต C แต่ละพอร์ตจะแสดงสถานะด้วยแอลอีดี 1 พอร์ต มี 8 บิต ดังนั้นจึงใช้ แอลอีดี 8 ดวง ต่อ 1 พอร์ต ดังแสดงในวงจรรูปที่ 3.105

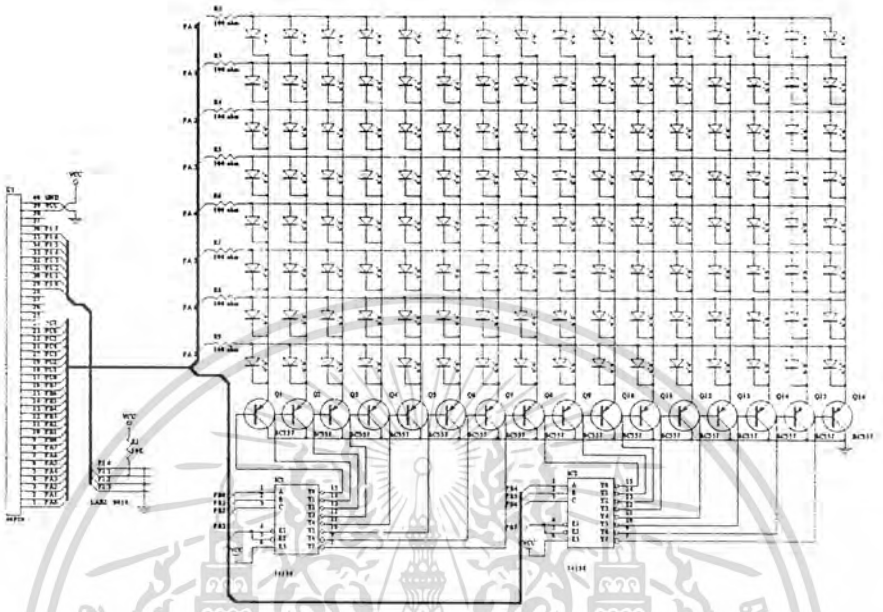


รูปที่ 3.105 วงจรทดลองที่ 1 เรื่องการเชื่อมต่อกับแอลอีดีลอจิก

จากรูปที่ 3.105 สถานะลอจิกเท่ากับ "1" มีแรงดันเท่ากับ 4.2–4.8 โวลต์ แอลอีดีต้องการแรงดันเท่ากับ 2.2โวลต์ จึงจำเป็นต้องต่อตัวต้านทาน 100โอห์ม เพื่อจำกัดกระแสและแบ่งแรงดันให้เหมาะสมกับแอลอีดี

2) การออกแบบวงจรทดลองที่ 2 เรื่องการเชื่อมต่อกับแอลอีดีดอทเมตริกซ์

วงจรทดลองที่ 2 ออกแบบเพื่อใช้เป็นอุปกรณ์แสดงรูปภาพหรือสัญลักษณ์ ในการสร้างภาพด้วยจุดทั้งแนวนอนและแนวตั้งรวมกันหมายถึง หากต้องการให้แอลอีดี ดวงใดติดสว่างจำเป็นต้องทำให้ครบวงจรทั้งแนวนอน และแนวตั้ง กำหนดให้ ลอจิก "1" เป็นแรงดันทางแนวนอน และลอจิก "0" เป็นแรงดันในแนวตั้งจึงเกิดการครบวงจร แสดงวงจรดังรูปที่ 3.106



รูปที่ 3.106 วงจรทดลองที่ 2 เรื่องการเชื่อมต่อกับแอลอีดีดอทเมตริกซ์

จากวงจรในรูปที่ 3.106 การให้ลอจิก "1" ทางแนวนอนกำหนดโดยพอร์ต A ของไมโครคอนโทรลเลอร์ ส่วนลอจิก "0" ที่จะทำให้แอลอีดีติดสว่างได้ ถูกควบคุมด้วยทรานซิสเตอร์ชนิดพีเอ็นพี จะทำงานด้วยลอจิก "0" จากไอซีดีโคดเดอร์เบอร์ 74138 รับค่าลอจิกจากพอร์ต B ของวงจรไมโครคอนโทรลเลอร์

การควบคุมแนวนอน ถูกควบคุมด้วยพอร์ต A แบ่งออกเป็นแถว 1 แถว ต่อสภาวะ 1 บิต ของพอร์ต A มีวิธีเรียงลำดับ ดังนี้ พอร์ต A0-A7 เรียงจากแถวด้านบนคือแถวที่ 1 จนถึงแถวด้านล่างคือแถวสุดท้ายแถวที่ 8

การควบคุมทางด้านแนวตั้ง มีทั้งหมด 16 หลัก เริ่มจากหลักที่ 1 ทางด้านซ้าย เรียงลำดับไปถึงหลักที่ 16 ทางด้านขวา เพื่อให้ง่ายต่อการควบคุมจึงใช้ไอซีดีโคดเบอร์ 74138 ควบคุมด้วยพอร์ต B 4 บิตล่าง สำหรับหลักที่ 1-8 และ 4 บิตบนสำหรับหลักที่ 9-6 ดังตารางที่ 3.1

ตารางที่ 3.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 2

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	หลักที่	ฐาน 16
1	0	0	0	0	0	0	0	1	80
1	0	0	0	0	0	0	1	2	81
1	0	0	0	0	0	1	0	3	82
1	0	0	0	0	0	1	1	4	83
1	0	0	0	0	1	0	0	5	84
1	0	0	0	0	1	0	1	6	85
1	0	0	0	0	1	1	0	7	86
1	0	0	0	0	1	1	1	8	87
0	0	0	0	1	0	0	0	9	08
0	0	0	1	1	0	0	0	10	18
0	0	1	0	1	0	0	0	11	28
0	0	1	1	1	0	0	0	12	38
0	1	0	0	1	0	0	0	13	48
0	1	0	1	1	0	0	0	14	58
0	1	1	0	1	0	0	0	15	68
0	1	1	1	1	0	0	0	16	78
0	0	0	0	0	0	0	0	ไม่แสดงหลัก	00

3) การออกแบบวงจรทดลองที่ 3 เรื่องการเชื่อมต่อกับจอแสดงผลแบบ 7 ส่วน

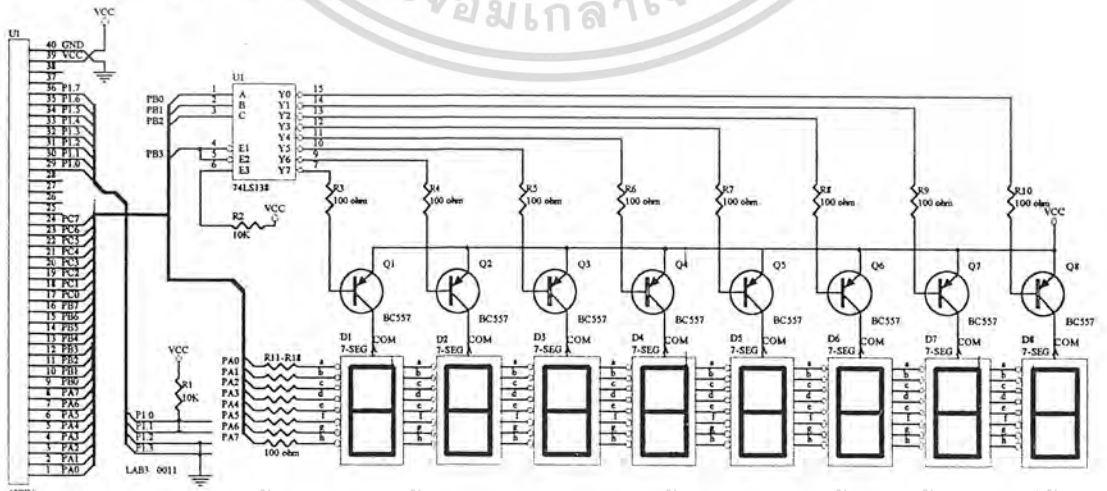
วงจรทดลองที่ 3 ทดลองการใช้งานจอแสดงผลแบบ 7 ส่วน ที่เลือกใช้ในวงจรทดลองที่ 3 นี้เลือกใช้แบบมีขั้วบวกร่วมกันดังนั้นเมื่อจะทำให้จอแสดงผลแบบ 7 ส่วน ส่วนใดติดสว่างต้องให้ส่วนนั้นต้องได้รับลอจิก "0" สำหรับการออกแบบวงจรการทดลองที่ 3 นี้ ใช้จอแสดงผลแบบ 7 ส่วนจำนวน 8 ตัวจอแสดงผลแบบ 7 ส่วน แต่ละส่วนของทุกตัวเลข จะต่อถึงกันและควบคุมด้วยพอร์ต A ของวงจรไมโครคอนโทรลเลอร์ หากต้องการให้ส่วนใดของตัวเลขติดสว่าง ก็ให้ลอจิกนั้นมีสถานะเป็น "0" แต่จะติดสว่างที่ตัวเลขหลักใดนั้นขึ้นอยู่กับจะให้หลักใดครบวงจร ในวงจรนี้เลือกใช้ทรานซิสเตอร์ชนิดพีเอ็นพี เป็นอุปกรณ์สวิตช์โดยควบคุมการเลือกหลักของตัวเลขด้วย

พอร์ต B ผ่านทางไอซีดีโค้ดเดอร์เบอร์ 74138 ใช้พอร์ต B เพียง 4 เส้นในการควบคุมหลักของจอแสดงผลแบบ 7 ส่วน เลือกใช้พอร์ต B ไบต์ต่ำเพื่อควบคุมหลักที่ใช้แสดงผล เริ่มต้นหลักที่ 1 ทางด้านขวา จนถึงหลักที่ 8 ทางด้านซ้าย การกำหนดค่าแสดงดังตารางที่ 3.2 และแสดงวงจรดังรูปที่ 3.107

ตารางที่ 3.2 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 3

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	หลักที่	ฐาน 16
X	X	X	X	0	0	0	0	1	X0
X	X	X	X	0	0	0	1	2	X1
X	X	X	X	0	0	1	0	3	X2
X	X	X	X	0	0	1	1	4	X3
X	X	X	X	0	1	0	0	5	X4
X	X	X	X	0	1	0	1	6	X5
X	X	X	X	0	1	1	0	7	X6
X	X	X	X	0	1	1	1	8	X7
X	X	X	X	1	0	0	0	ไม่แสดงหลัก	-

หมายเหตุ : X หมายถึงค่าใดๆ



รูปที่ 3.107 วงจรทดลองที่ 3 เรื่องการเชื่อมต่อกับจอแสดงผลแบบ 7 ส่วน

4) การออกแบบวงจรทดลองที่ 4 เรื่องการเชื่อมต่อกับสวิตช์เมตริกซ์

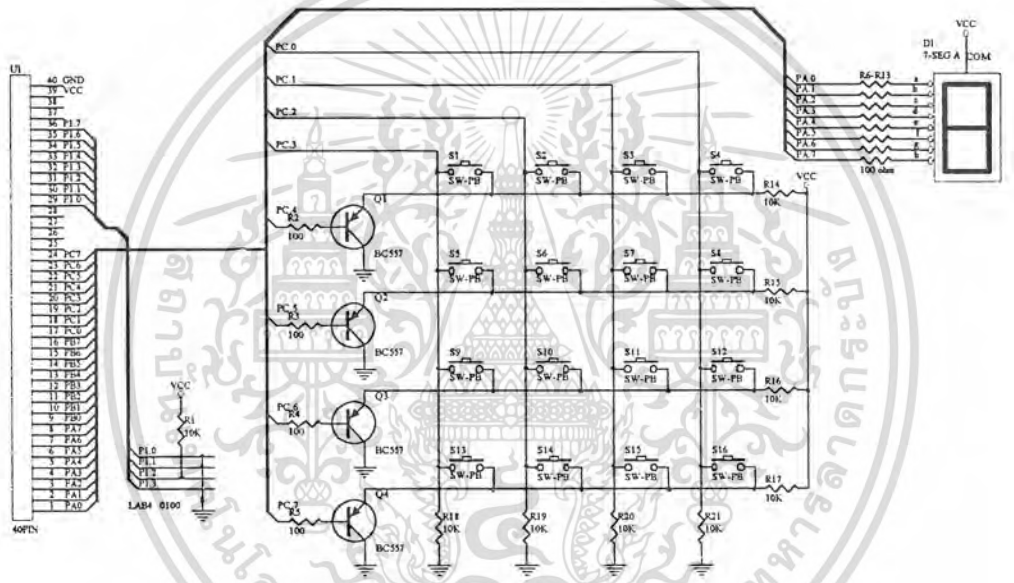
วงจรทดลองที่ 4 ทดลองการทำงานของสวิตช์ที่ต่อใช้งานแบบแถวคูณหลักควบคุมและตรวจสอบค่าด้วยสภาวะของพอร์ต C สภาวะเริ่มต้นกำหนดให้ค่าพอร์ตทุกบิตเป็น "1" และใช้การทดสอบทีละ 1 แถว โดยกำหนดค่าออกด้วยพอร์ต C ไบต์สูง คือ PC4–PC7 และรับค่าเข้าเพื่อพิจารณาค่าด้วยพอร์ต C ไบต์ต่ำ คือ PC0–PC3 คือทดสอบทีละ 1 แถวนั่นเอง 1 แถวพิจารณาสวิตช์ 4 ตัว ดังนั้นสวิตช์ 4 แถวทดสอบสวิตช์ได้ทั้งหมด 16 ตัว 16 กรณี ดังตารางที่ 3.3

ตารางที่ 3.3 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 4

PC7	PC6	PC5	PC4	PC3	PC2	PC1	CP0	สวิตช์	ฐาน 16
1	1	1	0	1	1	1	0	0	EE
1	1	1	0	1	1	0	1	1	ED
1	1	1	0	1	0	1	1	2	EB
1	1	1	0	0	1	1	1	3	E7
1	1	0	1	1	1	1	0	4	DE
1	1	0	1	1	1	0	1	5	DD
1	1	0	1	1	0	1	1	6	DB
1	1	0	1	0	1	1	1	7	D7
1	0	1	1	1	1	1	0	8	BA
1	0	1	1	1	1	0	1	9	BD
1	0	1	1	1	0	1	1	A	BB
1	0	1	1	0	1	1	1	B	B7
0	1	1	1	1	1	1	0	C	7E
0	1	1	1	1	1	0	1	D	7D
0	1	1	1	1	0	1	1	E	7B
0	1	1	1	0	1	1	1	F	77

จากตารางที่ 3.3 นั้น กรณีทั้ง 16 กรณีจะเกิดขึ้น เมื่อมีการกดสวิตช์พอร์ต C ไบต์ต่ำ จะเปลี่ยนสภาวะจาก "1" เป็น "0" ส่วนพอร์ต C ไบต์สูง จะกำหนดว่าจะพิจารณาแถวใดก็ให้แถวนั้นมีสภาวะเป็น "0"

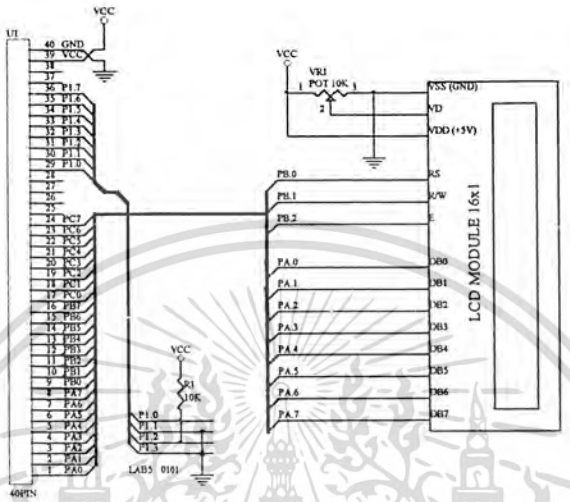
ในวงจรการทดลองนี้ได้ใช้ตัวเลข 7 ส่วนเพื่อใช้แสดงตำแหน่งของสวิตช์ที่ถูกกด ตัวเลข 7 ส่วนควบคุมด้วยสภาวะของพอร์ต A คือต้องการให้ส่วนของตัวเลขติดสว่าง ก็ให้บิตของตัวเลขส่วนนั้นมีสภาวะเป็น "0" แสดงวงจรดังรูปที่ 3.108



รูปที่ 3.108 วงจรทดลองที่ 4 เรื่องการเชื่อมต่อกับสวิตช์เมตริกซ์

5) การออกแบบวงจรทดลองที่ 5 เรื่องการเชื่อมต่อกับอุปกรณ์แสดงผลชนิดผลึกเหลว

วงจรทดลองที่ 5 เป็นการทดลองการใช้งานอุปกรณ์แสดงผลชนิดผลึกเหลวซึ่งการควบคุมประกอบด้วย 14 ขา ควบคุมและส่งข้อมูลขาควบคุมการเขียนการอ่านข้อมูล RS, R/Z และ E ควบคุมด้วยพอร์ต B คือ B0, B1 และ B2 ตามลำดับ ส่วนขาควบคุมการส่งข้อมูล DB0–DB7 ควบคุมด้วยพอร์ต A0–A7 ตามลำดับ แสดงวงจรดังรูปที่ 3.109



รูปที่ 3.109 วงจรทดลองที่ 5 เรื่องการต่อใช้งานกับอุปกรณ์แสดงผลชนิดผลึกเหลว

6) การออกแบบวงจรทดลองที่ 6 เรื่องการเชื่อมต่อกับคอนโทรลสวิตช์

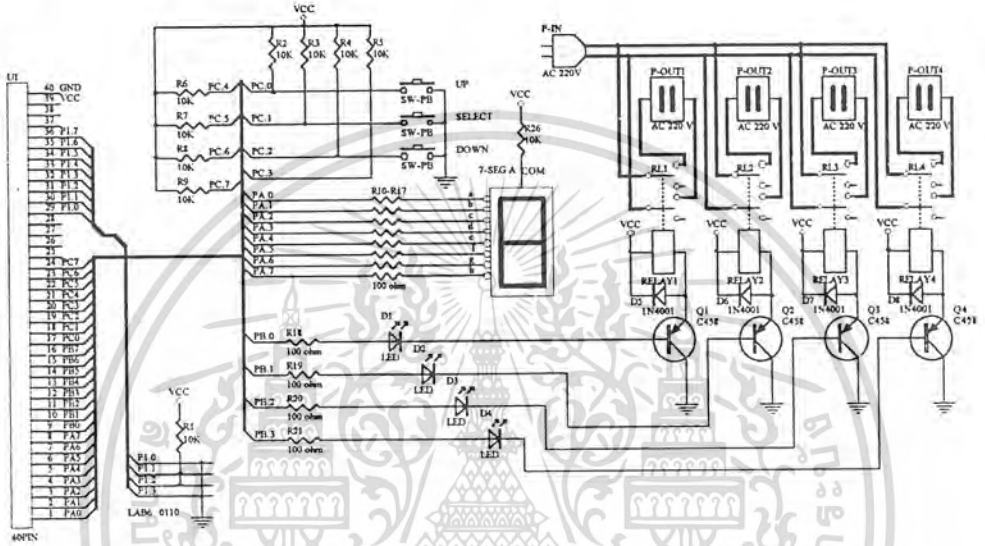
วงจรทดลองที่ 6 ออกแบบเพื่อควบคุมการทำงานของรีเลย์โดยอาศัยสภาวะของพอร์ต B เป็นตัวกำหนดรีเลย์ทำงานเมื่อได้รับสภาวะลอจิก "1" ผ่านแอลอีดีแสดงผลใช้รีเลย์จำนวน 4 ตัว รีเลย์ 1–4 ควบคุมด้วยพอร์ต B0–B3 ตามลำดับ

สวิตช์ควบคุมสำหรับวงจรทดลองนี้มี 3 สวิตช์ ถูกกำหนดและตรวจสอบด้วยพอร์ต C เมื่อกดสวิตช์ต่างๆ จะมีค่าที่พอร์ต C ดังตารางที่ 3.4

ตารางที่ 3.4 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 6

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	1	1	1	1	0	1	FE
1	1	1	1	1	1	0	1	2	FD
1	1	1	1	1	0	1	1	3	FB

นอกจากนั้นยังใช้จอแสดงผลแบบ 7 ส่วนเป็นอุปกรณ์แสดงค่าควบคุมด้วยพอร์ต A เนื่องจากเป็นตัวเลข 7 ส่วนที่มีขั้วบวกร่วมกัน การแสดงตัวเลขจึงต้องการลอจิก "0" เมื่อกำหนดให้ตัวเลขส่วนใดติดสว่าง แสดงวงจรดังรูปที่ 3.110



รูปที่ 3.110 วงจรทดลองที่ 6 เรื่องการเชื่อมต่อกับคอนโทรลสวิตช์

7) การออกแบบวงจรทดลองที่ 7 เรื่องการเชื่อมต่อกับมอเตอร์ไฟฟ้ากระแสตรง

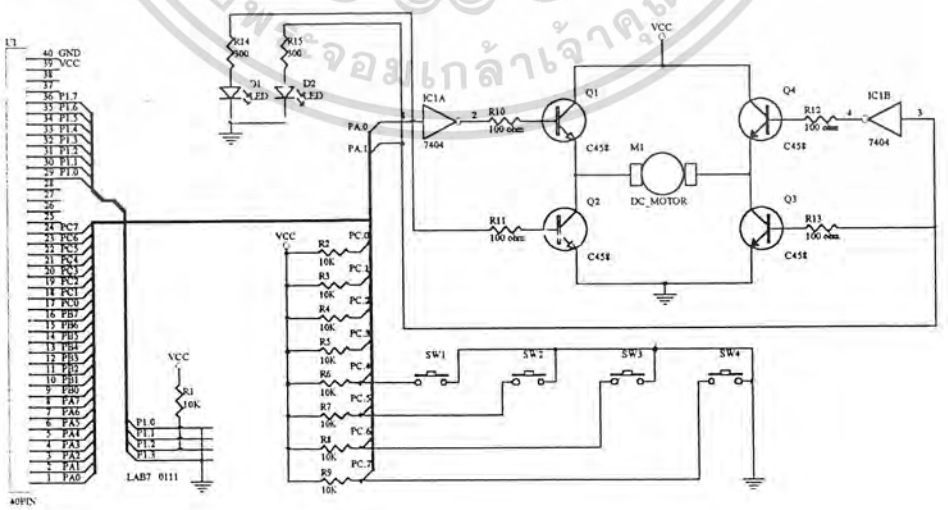
วงจรทดลองที่ 7 เป็นวงจรควบคุมทิศทางการหมุนของมอเตอร์ไฟฟ้ากระแสตรงเลือกใช้ทรใช้ทรานซิสเตอร์ชนิดเอ็นพีเอ็น 4 ตัวต่อกันในลักษณะกากบาทโดยให้มอเตอร์ไฟฟ้ากระแสตรงอยู่ตรงกลาง และให้สภาวะของทรานซิสเตอร์ที่อยู่คนละด้านคือ Q1 และ Q3 มีสภาวะเดียวกันเช่นเดียวกับ Q2 และ Q4 และสามารถควบคุมสภาวะของทรานซิสเตอร์ 2 คู่นี้ด้วยพอร์ต A0 และ A1 โดยมีไอซีนอตเกทเป็นตัวกลับลอจิกให้ตรงข้ามกัน แสดงความสัมพันธ์ของพอร์ต A0 และ A1 และทิศทางการหมุนของมอเตอร์ได้ดังนี้

ทิศทางการหมุน	พอร์ต A0	พอร์ต A1
ไม่หมุน	0	0
หมุนขวา	0	1
หมุนซ้าย	1	0
ไม่หมุน	1	1

ส่วนสวิตช์ที่ใช้ในการควบคุมการหมุนเลือกใช้พอร์ต C ไบต์สูงคือ PC4–PC7 เป็นอุปกรณ์ตรวจจับการกดเมื่อกดสวิตช์ 1-4 จะทำให้สภาวะของพอร์ต C เปลี่ยนแปลงไปดังตารางที่ 3.5 และวงจรแสดงดังรูปที่ 3.111

ตารางที่ 3.5 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 7

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	0	1	1	1	1	1	EF
1	0	0	1	1	1	1	1	2	DF
1	0	1	1	1	1	1	1	3	BF
0	1	1	1	1	1	1	1	4	7F



รูปที่ 3.111 วงจรทดลองที่ 7 เรื่องการเชื่อมต่อกับดีซีมอเตอร์

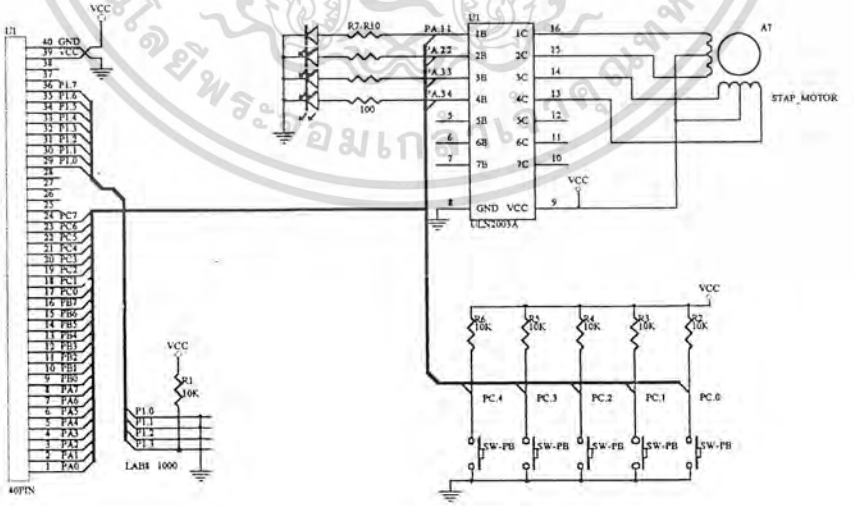
8) การออกแบบวงจรทดลองที่ 8 เรื่องการเชื่อมต่อกับสเต็ปปิ้งมอเตอร์

วงจรทดลองที่ 8 เป็นวงจรขับมอเตอร์ชนิดสเต็ปเลือกใช้สเต็ปมอเตอร์แบบ 4 เฟส กำหนดให้พอร์ต A0–A4 ควบคุมเฟส 1–4 ของสเต็ปมอเตอร์ตามลำดับ และมีแอลอีดีเพื่อแสดงสภาวะแต่ละเฟส แสดงวงจรดังรูปที่ 3.112

ควบคุมการใช้พอร์ต C0–C4 ตรวจจับการกดสวิตช์ 1-5 โดยเมื่อกดสวิตช์ 1-5 จะเกิดสถานะของพอร์ต C ดังตารางที่ 3.6

ตารางที่ 3.6 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 8

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	1	1	1	1	0	1	FE
1	1	1	1	1	1	0	1	2	FD
1	1	1	1	1	0	1	1	3	FB
1	1	1	1	0	1	1	1	4	F7
1	1	1	0	1	1	1	1	5	EF



รูปที่ 3.112 วงจรทดลองที่ 8 เรื่องการเชื่อมต่อกับสเต็ปปิ้งมอเตอร์

9) การออกแบบวงจรที่ 9 เรื่องการสร้างสัญญาณเสียง

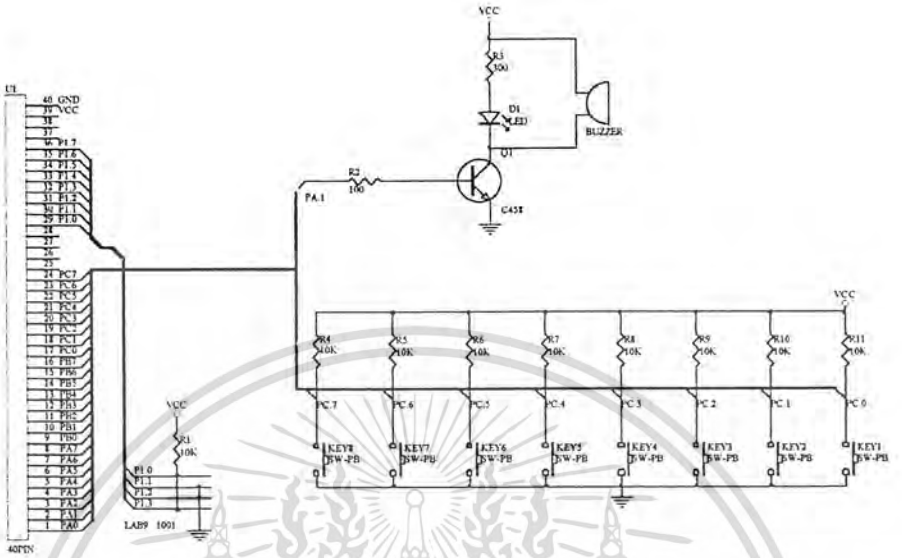
วงจรทดลองที่ 9 ออกแบบเพื่อแสดงการสร้างสัญญาณพัลส์ด้วยพอร์ต A ควบคุมอัตราความถี่ของเสียงด้วยโปรแกรมและควบคุมโปรแกรมด้วยสวิตช์ 8 ตัวคือ คีย์ร์ 8 ถึงคีย์ร์ 1 เรียงลำดับจากซ้ายไปขวา

ค่าที่เกิดขึ้นขณะกดสวิตช์จะทำให้บิตของพอร์ต C เปลี่ยนสภาวะจาก "1" เป็น "0" เขียนดังตารางที่ 3.7

ตารางที่ 3.7 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 9

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	1	1	1	1	0	1	FE
1	1	1	1	1	1	0	1	2	FD
1	1	1	1	1	0	1	1	3	FB
1	1	1	1	0	1	1	1	4	F7
1	1	1	0	1	1	1	1	5	EF
1	1	0	1	1	1	1	1	6	DF
1	0	1	1	1	1	1	1	7	BF
0	1	1	1	1	1	1	1	8	7F

ในส่วนของการขยายเสียงใช้ทรานซิสเตอร์ชนิดเอ็นพีเอ็นรับลอจิกไบอัสจากพอร์ต A1 ขับลำโพงชนิดเปียสโซให้เสียงดังในระดับพอประมาณขึ้นอยู่กับความถี่ที่ป้อนไบอัส ให้กับทรานซิสเตอร์ วงจรแสดงดังรูปที่ 3.113

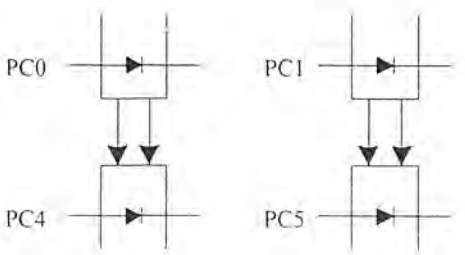


รูปที่ 3.113 วงจรทดลองที่ 9 เรื่องการเชื่อมต่อกับวงจรสร้างสัญญาณเสียง

10) การออกแบบทดลองที่ 10 เรื่องการเชื่อมต่อกับเซ็นเซอร์และเคาน์เตอร์

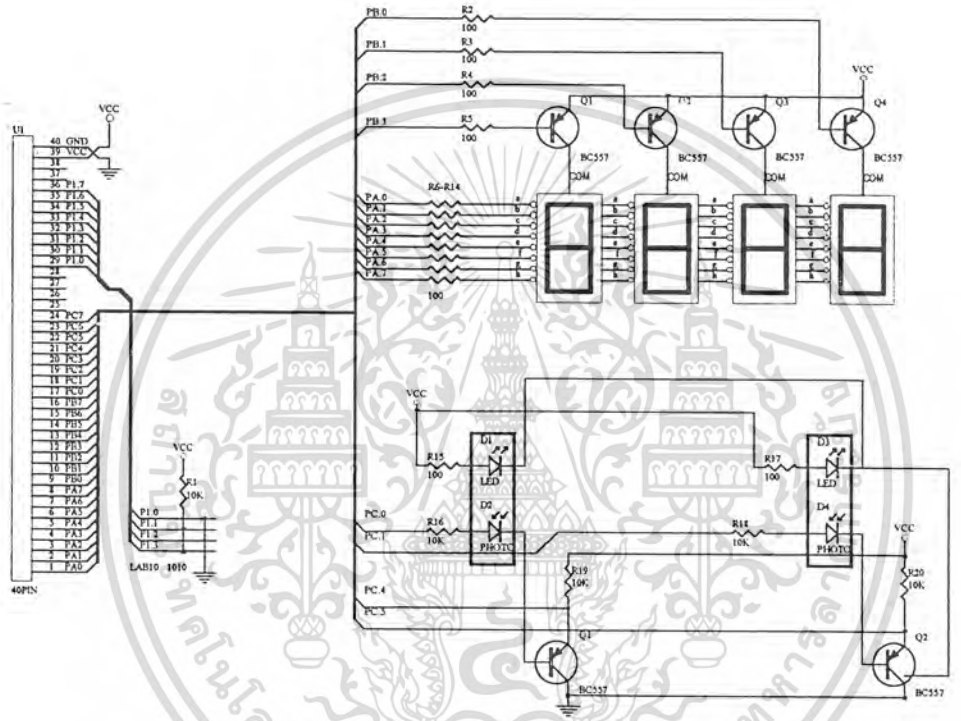
วงจรทดลองที่ 10 เป็นการใช้งานร่วมกันกับอุปกรณ์ตรวจจับแสง ความเปลี่ยนแปลงสถานะของอุปกรณ์ตรวจจับแสงนำมาประมวลผลและแสดงออกทางตัวเลข 7 ส่วน จำนวน 4 หลัก

อุปกรณ์ตรวจจับแสงใช้พอร์ต C เป็นตัวตรวจจับสภาวะ ให้ทางภาคส่งแสงเป็นพอร์ต C0 และ C1 ส่วนตัวรับแสงใช้พอร์ต C4 และ C5 ตามลำดับ การรับค่าสภาวะของลอจิกจะรับเข้าทางพอร์ต C4 และ C5 วางตำแหน่งของอุปกรณ์ตรวจจับแสงดังรูปที่ 3.114



รูปที่ 3.114 ตำแหน่งของอุปกรณ์ตรวจจับแสง

จากรูปที่ 3.114 แสงจากภาคส่งจะส่งออกมาได้นั้น ต้องได้รับสภาวะลอจิก “1” จากพอร์ต C0 และ C1 ทางด้านรับเมื่อได้รับแสง จะทำให้มีสภาวะทางพอร์ต C4 และพอร์ต C5 เป็น “0” ดังนั้นหากมีสิ่งกีดขวาดทางแสง สภาวะที่ได้จากพอร์ต C4 และพอร์ต C5 จะเป็น “1” ทันที จะเกิดการเปลี่ยนแปลงทางลอจิกสามารถนำการเปลี่ยนแปลงไปประมวลผลได้ วงจรแสดงดังรูปที่ 3.115



รูปที่ 3.115 วงจรทดลองที่ 10 เรื่องการเชื่อมต่อกับเซ็นเซอร์และเคาน์เตอร์

3.6 การสร้าง Help File

3.6.1 ขั้นตอนการสร้าง Help File

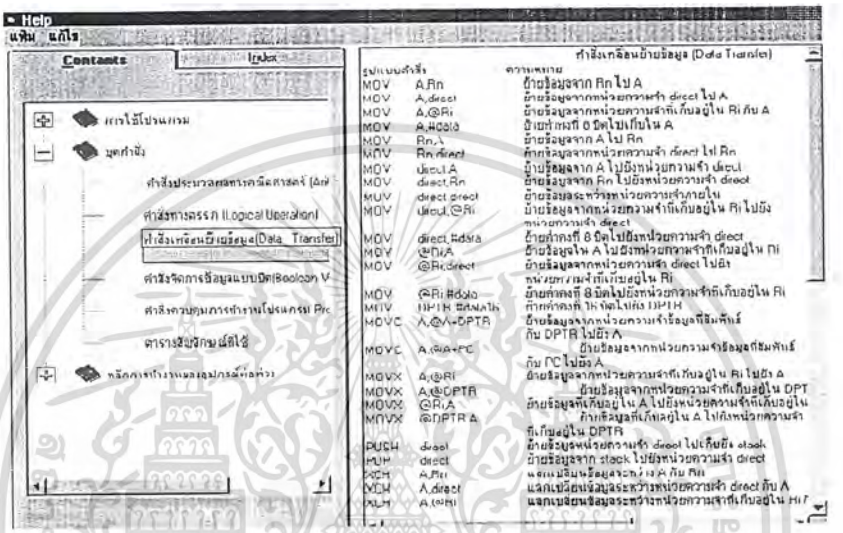
1) กำหนดรูปแบบของหัวข้อหลักและย่อย ดังรูปที่ 3.116

- ชุดคำสั่ง
  - คำสั่งประมวลผลทางคณิตศาสตร์ (Arithemation Operation)
  - คำสั่งทางตรรก (Logical Operations)
  - คำสั่งเคลื่อนย้ายข้อมูล (Data Transfer)
  - คำสั่งจัดการข้อมูลแบบบิต (Boolean Variable Manipulation)
  - คำสั่งควบคุมการทำงานโปรแกรม (Program and Machine Control)
  - ตารางสัญลักษณ์ที่ใช้
- การใช้โปรแกรม
  - การสร้างไฟล์ .ASM
  - การคอมไพล์ สร้างไฟล์ .HEX
- การเลือกโหมดการทำงาน
  - จำลองการทำงานบน PC
  - ทำงานโดยชุดอินเตอร์เฟส (อุปกรณ์ต่อพ่วง)
- หลักการทำงานของอุปกรณ์ต่อพ่วง
  - อุปกรณ์ต่อพ่วงที่ 1
  - อุปกรณ์ต่อพ่วงที่ 2
  - อุปกรณ์ต่อพ่วงที่ 3
  - อุปกรณ์ต่อพ่วงที่ 4
  - อุปกรณ์ต่อพ่วงที่ 5
  - อุปกรณ์ต่อพ่วงที่ 6
  - อุปกรณ์ต่อพ่วงที่ 7
  - อุปกรณ์ต่อพ่วงที่ 8
  - อุปกรณ์ต่อพ่วงที่ 9
  - อุปกรณ์ต่อพ่วงที่ 10

รูปที่ 3.116 รูปแบบของหัวข้อหลักและหัวข้อย่อย

2) สร้างไฟล์ Help Contants โดยกำหนดฟอร์มของวิชวล ดังนี้

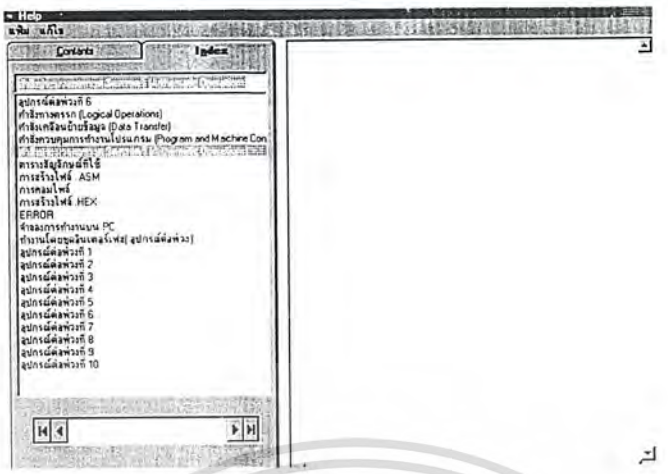
SSTab, TreeView และ TextBox วางฟอร์มบนวิชวลดังแสดงในรูปที่ 3.117 การกำหนดหัวข้อหลักและย่อยนั้น จะกำหนดลงใน TreeView และเมื่อมีการกดเลือกหัวข้อใด ก็จะมีการแสดงข้อมูลที่ TextBox จะแสดงข้อมูลในรูปแบบไฟล์ .TXT



รูปที่ 3.117 การกำหนดรูปแบบของ Contants File

3) สร้างไฟล์ Index โดยการกำหนดฟอร์มของวิชวล ดังนี้

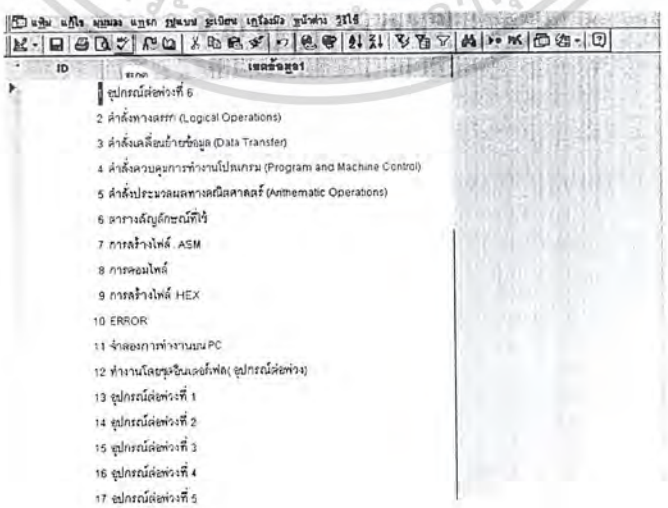
SSTab (ใช้งานร่วมกับ Contants), DBCombo, Data Control และ TextBox (ใช้งานร่วมกับ Contants) วางฟอร์มบนวิชวลดังแสดงในรูปที่ 1.118 โดยที่หัวข้อหลักและย่อยรวมถึงรายละเอียดของแต่ละหัวข้อจะถูกเก็บไว้ที่ Data Control เมื่อเราต้องการดูข้อมูลให้พิมพ์หรือเลือกได้จาก DBCombo โดยการเก็บข้อมูลใน Data Control จะเก็บในรูปแบบไฟล์ .MDB



รูปที่ 3.118 การกำหนดรูปแบบของไฟล์ Index

4) การสร้างไฟล์ .TXT และ ไฟล์ .MDB

โดยที่การสร้างไฟล์ .TXT นั้นสามารถสร้างได้ทั้งจาก Microsoft Word และ จาก Notepad เริ่มจากการเปิดเอกสารใหม่ขึ้นมาหนึ่งเอกสารพิมพ์ข้อความลงไป ตามรายละเอียดที่ต้องการ เมื่อเสร็จแล้วทำการบันทึกเป็นไฟล์ .TXT ส่วนการสร้างไฟล์ .MDB นั้นเราจะทำสร้างใน Microsoft Access โดยเริ่มจากเปิดที่แฟ้ม/สร้างฐานข้อมูล จากนั้นเลือกสร้างตาราง/Datasheet View จากนั้นใส่หัวข้อ, รายละเอียดข้อมูล และ กำหนด ID ให้กับแต่ละหัวข้อ ดังแสดงในรูปที่ 3.119 เพื่อใช้ในการติดต่อข้อมูลกับ Data Control



รูปที่ 3.119 การกำหนดค่า ID ให้กับข้อมูล

5) การบันทึก

เมื่อทำการพิมพ์ข้อมูลทุกอย่างครบแล้วก็ทำการบันทึกข้อมูล โดยหากเป็น Notepad สามารถเลือกที่ File/Save และตั้งชื่อบันทึกได้เลยส่วน Microsoft Word นั้นให้เลือกที่ แฟ้ม/บันทึกแล้วเลือกจัดเก็บเป็นชนิด MS-DOS Text จากนั้นตั้งชื่อแล้วบันทึกได้ ส่วนการจัดเก็บไฟล์ .MDB นั้นให้เลือกแฟ้ม/บันทึกจากนั้นตั้งชื่อและบันทึกจากนั้นเลือกบันทึกเป็น/การส่งออก กำหนดจัดเก็บเป็นชนิด Microsoft Access และจัดเก็บในแฟ้มที่ต้อง

6) การเรียกใช้งาน

เมื่อเปิดใช้โปรแกรมและเรียก Help แล้วก็จะสามารถดูรายละเอียดของข้อมูลได้จากฟอร์มในข้อ 2 และ 3

บทที่ 4

การทดลองและผลการทดลอง

4.1 บทนำ

ในส่วนของบทนี้กล่าวถึงการทดลอง และผลการทดลองโปรแกรมจำลองการทำงานของ MCS-51และอุปกรณ์ต่อพ่วงผ่านทางพอร์ตอนุกรมติดต่อ โปรแกรมที่สร้างขึ้นมานี้มีการใช้งานหลักๆ ด้วยกัน 2 ส่วนคือ ในส่วนของการจำลองการทำงานด้วยโปรแกรมคอมพิวเตอร์และในส่วนของการใช้อุปกรณ์ต่อพ่วงซึ่งรายละเอียดของการทดลองและผลการทดลอง ซึ่งมีรายละเอียดดังนี้

4.2 การทดลองวงจรไมโครคอนโทรลเลอร์ (TEPSA-I)

โครงสร้างการทำงานของวงจรไมโครคอนโทรลเลอร์ได้กล่าวมาแล้วในบทที่ 3 ซึ่งในที่นี้จะเป็นการทดลองการทำงานของวงจรไมโครคอนโทรลเลอร์ หน้าที่หลักคือรับ-ส่งและแลกเปลี่ยนข้อมูลซึ่งทำงานตามลำดับขั้นตอนที่ได้รับจากคอมพิวเตอร์โดยทำงานร่วมกันกับโปรแกรมมอนิเตอร์ที่บันทึกอยู่ไอซีประมวลผลกลางของวงจรไมโครคอนโทรลเลอร์

4.2.1 การทดลองส่งข้อมูล

การทดลอง

1) เข้าหน้าจอหลักของโปรแกรม
2) เลือกที่ประยุกต์ใช้งาน
3) เขียนโปรแกรมภาษาแอสแซมบลีในอีดิเตอร์
4) บันทึกโปรแกรมที่สร้างขึ้น
5) ทำการคอมไพล์โปรแกรมเป็นไฟล์ฐานสิบหก
6) เลือกการทำงานกับอุปกรณ์ต่อพ่วงที่ 1
7) ส่งข้อมูลไปยังหน่วยความจำภายนอกของวงจรไมโครคอนโทรลเลอร์
8) สั่งให้โปรแกรมทำงาน

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

อุปกรณ์ต่อพ่วงที่ 1 ทำงานตามโปรแกรมที่เขียนขึ้นได้อย่างถูกต้อง แสดงว่าการส่งข้อมูลจากคอมพิวเตอร์ไปยังหน่วยความจำภายนอกของวงจรไมโครคอนโทรลเลอร์ เป็นไปอย่างถูกต้อง

ผลการทดลองทางด้านโปรแกรม

สามารถทำงานได้ตามคำสั่งที่เขียนขึ้นมีลักษณะการทำงานคล้ายกับอุปกรณ์จริง แต่จะแสดงผลช้ากว่าอุปกรณ์จริงอยู่มาก

4.2.2 การทดลองรับข้อมูล

การทดลอง

1) หลังจากส่งข้อมูลโปรแกรมไปยังหน่วยความจำภายนอก ของวงจรไมโครคอนโทรลเลอร์ ยังไม่ต้องสั่งให้โปรแกรมทำงาน

2) ขอดูค่าโปรแกรมที่ส่งไปจากตำแหน่งหน่วยความจำภายนอกของวงจรไมโครคอนโทรลเลอร์เพื่อกลับมาแสดงผลบนคอมพิวเตอร์

3) สั่งให้โปรแกรมทำงาน

4) ขณะที่โปรแกรมทำงานอยู่กดปุ่มหยุดเพื่อหยุดการทำงาน

5) ขอดูรีจิสเตอร์แบงค์และรีจิสเตอร์ฟังก์ชันพิเศษจากหน่วยความจำภายในของวงจรไมโครคอนโทรลเลอร์และส่งค่าเหล่านั้นกลับมาแสดงบนคอมพิวเตอร์

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

หน้าจอคอมพิวเตอร์สามารถแสดงค่าของรีจิสเตอร์แบงค์ และรีจิสเตอร์ฟังก์ชันพิเศษจากหน่วยความจำภายในของวงจรไมโครคอนโทรลเลอร์ และส่งค่าตำแหน่งต่างๆ ที่โปรแกรมของหน่วยความจำภายนอกจากวงจรไมโครคอนโทรลเลอร์ได้อย่างถูกต้อง

ผลการทดลองทางด้านโปรแกรม

สามารถทำงานได้ตามคำสั่งที่เขียนขึ้นมีลักษณะการทำงานคล้ายกับอุปกรณ์จริง แต่จะแสดงผลช้ากว่าอุปกรณ์จริงอยู่มาก

4.2.3 การทดลองอุปกรณ์ต่อพ่วงที่ 1 เรื่องการเชื่อมต่อกับแอลอีดีลอจิก

การทดลอง

อุปกรณ์ต่อพ่วงที่ 1 แอลอีดีลอจิกเป็นวงจรทดสอบสภาวะของพอร์ตทั้ง 3 ด้วยแอลอีดีกำหนดให้สภาวะลอจิก “1” แอลอีดีติดสว่างและสภาวะลอจิก “0” แอลอีดีดับเขียนโปรแกรมทดสอบได้ดังรูปที่ 4.1

```
;*******************************************
;     LAB 1 LED LOGIC
;*******************************************

ORG 8000H
PORT_A          EQU     0E000H
PORT_B          EQU     0E001H
PORT_C          EQU     0E002H
C_PORT          EQU     0E003H

;-------------------------------------------

SET_8255:       MOV     A,#80H
                MOV     DPTR,#C_PORT
                MOVX    @DPTR,A
;-------------------------------------------
START:          MOV     R1,#03H
                MOV     DPTR,#PORT_A
START0:         MOV     R2,#09H
                MOV     A,#01H
NEXT:           MOVX    @DPTR,A
                LCALL   DELAY_SC
                RL      A
                DJNZ    R2,NEXT
                INC     DPTR
                DJNZ    R1,START0
                AJMP    START
;-------------------------------------------
DELAY_SC:       MOV     R5,#65H
LOOP1:          MOV     R6,#30H
LOOP2:          NOP
                DJNZ    R6,LOOP2
                DJNZ    R5,LOOP1

RET
;-------------------------------------------
END
```

รูปที่ 4.1 โปรแกรมการทดลองที่ 1 เรื่องการเชื่อมต่อกับแอลอีดีลอจิก

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

จากโปรแกรมที่กำหนดให้แอลอีดีติดสว่างเพิ่มขึ้นที่ละ 1 ดวง และจะทำงานทีละ 1 แถว ปรากฏว่าวงจรสามารถทำงานได้ตามโปรแกรมที่กำหนดไว้

ผลการทดลองทางด้านโปรแกรม

สามารถทำงานได้ตามคำสั่งที่เขียนขึ้นมีลักษณะการทำงานคล้ายกับอุปกรณ์จริง แต่จะแสดงผลช้ากว่าอุปกรณ์จริงอยู่มาก

4.2.4 การทดลองอุปกรณ์ต่อพ่วงที่ 2 เรื่องการเชื่อมต่อกับแอลอีดีดอตเมตริกซ์

การทดลอง

อุปกรณ์ต่อพ่วงที่ 2 เป็นวงจรแสดงรูปภาพ, ข้อความหรือสัญลักษณ์ ด้วยแอลอีดีแบบแถวคูณหลัก มีขนาด 8 แถวคูณ16 หลัก เขียนโปรแกรมทดสอบดังรูปที่ 4.2

```
;************************************************************
;LAB 2 LED DOTMATRIX
;************************************************************
ORG 8000H
PORT_A    EQU   0E000H
PORT_B    EQU   0E001H
PORT_C    EQU   0E002H
C_PORT    EQU   0E003H
;************************************************************
SET_8255: MOV   A,#80H
          MOV   DPTR,#C_PORT
          MOVX @DPTR,A
;************************************************************
        ST0:    MOV     R1,#00H
        ST1:    MOV     R2,#07H
ST2:    MOV     R3,#00H
        MOV     A,R1
        MOV     R3,A
;************************************************************
        MOV     DPTR,#PORT_A
        MOV     A,#00H
        MOVX    @DPTR,A
DIGIDA: MOV     DPTR,#PORT_B
        MOV     A,#0F0H
LA:     LCALL   PATTERN
        MOVX    @DPTR,A
        LCALL   DELAY
        INC     A
        CJNE    A,#0F8H,LA
DIGIDB: MOV     A,#0FH
LB:     LCALL   PATTERN
        MOVX    @DPTR,A
        LCALL   DELAY
        SWAP    A
        INC     A
        SWAP    A
        CJNE    A,#8FH,LB
;************************************************************
        DJNZ    R2,ST2
        INC     R1
        CJNE    R1,#30H,ST1
        AJMP    ST0
;************************************************************
        DELAY:  MOV     R5,#01H
        LOOP1:  MOV     R6,#01H
        LOOP2:  NOP
        DJNZ    R6,LOOP2
        DJNZ    R5,LOOP1
        RET
;************************************************************
        PATTERN: PUSH   DPH
        PUSH    DPL
        PUSH    ACC
        MOV     A,R3
        MOV     DPTR,#DATA
        MOVC    A,@A+DPTR
        MOV     DPTR,#PORT_A
        MOVX    @DPTR,A
        INC     R3
        POP     ACC
        POP     DPL
        POP     DPH
        RET
;************************************************************
```


รูปที่ 4.2 โปรแกรมการทดลองที่ 2 เรื่องการเชื่อมต่อกับแอลอีดีดอตเมตริกซ์

```
DATA:   DB      00H,00H,00H,00H     ;_
        DB      00H,00H,00H,00H     ;
        DB      00H,00H,00H,00H     ;_
        DB      00H,00H,00H,00H     ;_
        DB      04H,7CH,04H,00H     ;T
        DB      7CH,54H,44H,00H     ;E
        DB      7CH,14H,1CH,00H     ;P
        DB      5CH,54H,74H,00H     ;S
        DB      7CH,14H,7CH,00H     ;A
        DB      10H,10H,10H,00H     ; -
        DB      44H,7CH,44H,00H     ; I
        DB      00H,00H,00H,00H     ;_
        DB      00H,00H,00H,00H     ;_
        DB      00H,00H,00H,00H     ;_
        DB      00H,00H,00H,00H     ;_
;*****************************************************************
END
```

รูปที่ 4.2 (ต่อ) โปรแกรมการทดลองที่ 2 เรื่องการเชื่อมต่อกับแอลอีดีดอตเมตริกซ์

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

จากโปรแกรมกำหนดให้แสดงข้อความ โดยเกิดทางด้านขวามือแล้วเลื่อนจนถึงทางด้านซ้ายมือและวนกลับมาแสดงผลใหม่ทางด้านขวามืออีกครั้ง ปรากฏว่าวงจรสามารถทำงานได้ตามโปรแกรมที่กำหนดไว้

ผลการทดลองทางด้านโปรแกรม

สามารถทำงานได้ตามคำสั่งที่เขียนขึ้นมีลักษณะการทำงานคล้ายกับอุปกรณ์จริง แต่จะแสดงผลช้ากว่าอุปกรณ์จริงอยู่มาก

4.2.5 การทดลองอุปกรณ์ต่อพ่วงที่ 3 เรื่องการเชื่อมต่อกับจอแสดงผล 7 ส่วน

การทดลอง

การทดลองอุปกรณ์ต่อพ่วงที่ 3 เป็นจอแสดงผลแบบ 7 ส่วนมีด้วยกันทั้งหมด 8 ตำแหน่ง เป็นตัวเลข 7 ส่วนชนิดขั้วบวกร่วมกันดังนั้นหากต้องการให้ส่วนใดติดสว่างต้องให้ส่วนนั้นได้รับสภาวะลอจิก “0” เขียนโปรแกรมทดสอบได้ดังรูปที่ 4.3

```
;************************************************************
;   LAB3  LED 7 SEGMENT (COMMON ANOD)
;   PORT_A  SEND DATA
;   PORT_B  SEND DIGID
;************************************************************
                ORG   8000H
PORT_A          EQU   0E000H
PORT_B          EQU   0E001H
CP              EQU   0E003H
;************************************************************
SET_8255:       MOV   DPTR,#CP
                MOV   A,#80H
                MOVX  @DPTR,A
;************************************************************
ST0:            MOV     R1,#00H
                MOV     DPTR,#PORT_B
ST1:            MOVX    @DPTR,A
                LCALL   OUT
                LCALL   DELAY
                INC     A
                INC     R1
                CJNE    R1,#08H,ST1
                SJMP    ST0
;************************************************************
OUT:            PUSH    DPH
                PUSH    DPL
                PUSH    ACC
                MOV     DPTR,#DATA
                MOV     A,R1
                MOVC    A,@A+DPTR
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
                POP     ACC
                POP     DPL
                POP     DPH
                RET
DATA:           DB      0C0H,0F9H,0A4H,0B0H
                DB      99H,92H,82H,0F8H
                DB      80H,90H,88H,83H
                DB      0C6H,0A1H,86H,8EH
;************************************
DELAY:  MOV  R5,#30H
LOOP3:  MOV  R6,#25H
LOOP2:  MOV  R7,#20H
LOOP1:  NOP
        NOP
        DJNZ R7,LOOP1
        DJNZ R6,LOOP2
        DJNZ R5,LOOP3
        RET
;************************************
 END
```

รูปที่ 4.3 โปรแกรมการทดลองที่ 3 เรื่องการเชื่อมต่อกับจอแสดงผล 7 ส่วน

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

จากโปรแกรมกำหนดให้ตัวเลข 7 ส่วนติดสว่าง แสดงตัวเลข "0"ในหลักที่ 1 ทางด้านขวามือแล้วแสดงตัวเลข "1" ในหลักที่ 2 แสดงตัวเลข "2" ในหลักที่ 3 แสดงตัวเลข "3" ในหลักที่ 4 แสดงตัวเลข "4" ในหลักที่ 5 แสดงตัวเลข "5" ในหลักที่ 6 แสดงตัวเลข "6" ในหลักที่ 7 แสดงตัวเลข "7"ในหลักที่ 8 และแสดงเรียงติดต่อกันไปจากหลักที่ 1 ถึงหลักที่ 8 แล้วกลับมาแสดงในหลักที่ 1 ใหม่ วงจรสามารถทำงานได้ตามโปรแกรมที่กำหนดไว้

ผลการทดลองทางด้านโปรแกรม

สามารถทำงานได้ตามคำสั่งที่เขียนขึ้นมีลักษณะการทำงานคล้ายกับอุปกรณ์จริง แต่จะแสดงผลช้ากว่าอุปกรณ์จริงอยู่มาก

4.2.6 การทดลองอุปกรณ์ต่อพ่วงที่ 4 เรื่องการเชื่อมต่อกับสวิตช์เมตริกซ์

การทดลอง

อุปกรณ์ต่อพ่วงที่ 4 เป็นวงจรทดสอบการใช้งานการกด สวิตช์แบบแถวคูณหลักรับค่าและส่งค่าการกด สวิตช์ด้วย พอร์ต C จอแสดงผลแบบ 7 ส่วนจากพอร์ต A เขียนโปรแกรมทดสอบดังรูปที่ 4.4

```
;*************************************************************
;    LAB4  SWITCH MATRIX
;    PORT_A          = OUT   7-SEGMENT
;    PORT_C  .0 - .3 = SCAN ROW
;            .4 - .7 = SCAN CORLUM
;*************************************************************
             ORG   8000H
PORT_A       EQU   0E000H
PORT_C       EQU   0E002H
CP           EQU   0E003H
;*************************************************************
SET_8255:    MOV   DPTR,#CP
             MOV   A,#81H
             MOVX  @DPTR,A
;*************************************************************
             MOV     DPTR,#PORT_A
             MOV     A,#0FFH
             MOVX    @DPTR,A
SCAN1:       MOV     DPTR,#PORT_C
             MOV     A,#0EFH
             MOVX    @DPTR,A
             MOVX    A,@DPT
SW:          CJNE    A,#0E7H,SW1
             MOV     R1,#00H
             LJMP    OUT
```

รูปที่ 4.4 โปรแกรมการทดลองที่ 4 เรื่องการเชื่อมต่อกับสวิตช์เมตริกซ์

```
SW1:        CJNE    A,#0EBH,SW2
            MOV     R1,#01H
            LJMP    OUT
SW2:        CJNE    A,#0EDH,SW3
            MOV     R1,#02H
            LJMP    OUT

SW3:        CJNE    A,#0EEH,SW4
            MOV     R1,#03H
            LJMP    OUT
SW4:        NOP
SCAN2:      MOV     DPTR,#PORT_C
            MOV     A,#0DFH
            MOVX    @DPTR,A
            MOVX    A,@DPTR
            CJNE    A,#0D7H,SW5
            MOV     R1,#04H
            LJMP    OUT
SW5:        CJNE    A,#0DBH,SW6
            MOV     R1,#05H
            LJMP    OUT
SW6:        CJNE    A,#0DDH,SW7
            MOV     R1,#06H
            LJMP    OUT
SW7:        CJNE    A,#0DEH,SW8
            MOV     R1,#07H
            LJMP    OUT
SW8:        NOP
SCAN3:      MOV     DPTR,#PORT_C
            MOV     A,#0AFH
            MOVX    @DPTR,A
            MOVX    A,@DPTR
            CJNE    A,#0A7H,SW9
            MOV     R1,#08H
            LJMP    OUT
SW9:        CJNE    A,#0ABH,SWA
            MOV     R1,#09H
            LJMP    OUT
SWA:        CJNE    A,#0ADH,SWB
            MOV     R1,#0AH
            LJMP    OUT
SWB:        CJNE    A,#0AEH,SWC
            MOV     R1,#0BH
            LJMP    OUT
SWC:        NOP
SCAN4:      MOV     DPTR,#PORT_C
            MOV     A,#7FH
            MOVX    @DPTR,A
            MOVX    A,@DPTR
            CJNE    A,#077H,SWD
            MOV     R1,#0CH
            LJMP    OUT
SWD:        CJNE    A,#7BH,SWE
            MOV     R1,#0DH
            LJMP    OUT
SWE:        CJNE    A,#7DH,SWF
            MOV     R1,#0EH
            LJMP    OUT
SWF:        CJNE    A,#7EH,SW0
            MOV     R1,#0FH
            LJMP    OUT
SW0:        LJMP    SCAN1
```

รูปที่ 4.4 (ต่อ) โปรแกรมการทดลองที่ 4 เรื่องการเชื่อมต่อกับสวิตช์เมตริกซ์

```
;************************************************************
OUT:            MOV     DPTR,#DATA
                MOV     A,R1
                MOVC    A,@A+DPTR
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
                LJMP    SCAN1
DATA:           DB      0C0H,0F9H,0A4H,0B0H
                DB      99H,92H,82H,0F8H
                DB      80H,90H,88H,83H
                DB      0C6H,0A1H,86H,8EH
;*************************************
DELAY:          MOV     R5,#30H
LOOP3:          MOV     R6,#20H
LOOP2:          MOV     R7,#20H
LOOP1:          NOP
                NOP
                DJNZ    R7,LOOP1
                DJNZ    R6,LOOP2
                DJNZ    R5,LOOP3
                RET
;*************************************
END
```

รูปที่ 4.4 (ต่อ) โปรแกรมการทดลองที่ 4 เรื่องการเชื่อมต่อกับสวิตช์เมตริกซ์

ผลการทดลอง

จากโปรแกรมกำหนดให้มีการตรวจการกดสวิตช์ ให้สวิตช์แถวบนเป็นแถวที่ 1 และสวิตช์ซ้ายมือสุด เป็นสวิตช์ตำแหน่ง "0" เรียงลำดับจนถึงตำแหน่ง "F" ที่สวิตช์ตัวขวามือตัวสุดท้าย เมื่อกดสวิตช์ใด ค่าตำแหน่งของสวิตช์นั้น จะแสดงออกที่เลข 7 ส่วน ปรากฏว่าวงจรสามารถทำงานได้ตามโปรแกรมที่กำหนดไว้

4.2.7 การทดลองอุปกรณ์ต่อพ่วงที่ 5 เรื่องการเชื่อมต่อกับจอแสดงผลแบบผลึกเหลว

การทดลอง

อุปกรณ์ต่อพ่วงที่ 5 เป็นวงจรแสดงผลด้วย จอแสดงผลแบบผลึกเหลวสามารถปรับความเข้มหรือสว่างของข้อความได้ด้วยการปรับ VR1 เขียนโปรแกรมทดสอบได้ดังรูปที่ 4.5

```
;************************************************************
;   LAB 5  CONTROL LCD
;   DOT MATRIX LCD MODULD 16X1
;************************************************************
ORG  8000H
PORT_A    EQU 0E000H
PORT_B    EQU 0E001H
C_PORT    EQU 0E003H
;************************************************************
SET:      MOV    PSW,#00H
          ACALL  DELAY  ;WAIT FOR 15mS
;************************************************************
SET_8255: MOV    DPTR,#C_PORT
          MOV    A,#80H
          MOVX   @DPTR,A
;************************************************************
SET_LCD:  ACALL  INTI
;************************************************************
LOOP:     MOV    DPTR,#DATA1
                 ACALL SWORD        ;WRITE
                 ACALL DELX         ;DELAY
                 MOV   DPTR,#DATA2
                 ACALL SWORD        ;WRITE
                 ACALL DELX         ;DELAY
                 MOV   DPTR,#DATA3
                 ACALL SWORD        ;WRITE
                 ACALL DELX         ;DELAY
                 MOV   DPTR,#DATA4
                 ACALL SWORD        ;WRITE
                 ACALL DELX         ;DELAY
                 MOV   DPTR,#DATA5
                 ACALL SWORD        ;WRITE
                 ACALL DELX         ;DELAY
                 MOV   DPTR,#DATA6
                 ACALL SWORD        ;WRITE
                 ACALL DELX         ;DELAY
                 LJMP  LOOP
;************************************************************
DELX:  MOV   R1,#0FFH
DEL1:  MOV   R2,#0FFH
DEL2:  MOV   R3,#07H
DEL3:  DJNZ  R3,DEL3
       DJNZ  R2,DEL2
       DJNZ  R1,DEL1
       RET
;************************************************************
DATA1:  DB   " !!!!HELLO !!!! "
DATA2:  DB   "     LAB  5     "
DATA3:  DB   " LCD  DOTMATRIX "
DATA4:  DB   " MCS-51 PROJECT "
DATA5:  DB   "       BY       "
DATA6:  DB   "    TEPSA - I   "
;************************************************************

SWORD:  MOV    R6,#01H
ACALL   LCODE
MOVX    A,@DPTR      ;CLEAR DISPLAY
MOV     R5,#00H
SWORD1: MOV    R0,A
ACALL   WRBYTE
```

รูปที่ 4.5 โปรแกรมการทดลองที่ 5 เรื่องการเชื่อมต่อกับจอแสดงผลแบบผลึกเหลว

```
        INC     DPTR
        INC     R5
        MOVX    A,@DPTR
        CJNE    R5,#16,SWORD2
        RET
SWORD2: CJNE    R5,#8,SWORD1
        MOV     R6,#0C0H
        ACALL   LCODE
        SJMP    SWORD1
;***********************************************************
WRBYTE: PUSH    DPL
        PUSH    DPH
        PUSH    ACC
        MOV     A,#01H
        MOV     DPTR,#PORT_B
        MOVX    @DPTR,A         ;CLEAR DISPLAY
        MOV     A,R0
        MOV     DPTR,#PORT_A
        MOVX    @DPTR,A         ;GET DATA
        ACALL   PLUSE
        POP     ACC
        POP     DPH
        POP     DPL
        RET
;***********************************************************
INTI:   MOV     A,#00H
        MOV     DPTR,#PORT_B
        MOVX    @DPTR,A
        MOV     R6,#38H         ;FUNCTION SET 8 BIT/5X7
        ACALL   LCODE
        ACALL   DELAY
        MOV     R6,#0CH         ;DISPLAY ON/OFF
        ACALL   LCODE
        MOV     R6,#01H         ;CLEAR DISPLAY
        ACALL   LCODE
        ACALL   DELAY
        RET
;***********************************************************
LCODE:  PUSH    DPH
        PUSH    DPL
        PUSH    ACC
        CLR     A
        MOV     DPTR,#PORT_B
        MOVX    @DPTR,A
        MOV     A,R6
        MOV     DPTR,#PORT_A
        MOVX    @DPTR,A
        ACALL   PLUSE
        POP     ACC
        POP     DPL
        POP     DPH
        RET
;***********************************************************
PLUSE:  MOV     DPTR,#PORT_B    ;PLUSE FOR ENABLE
        MOVX    A,@DPTR
        SETB    ACC.2           ;ENABLE=1
        MOV     DPTR,#PORT_B
        MOVX    @DPTR,A
        ACALL   DELAY
```

รูปที่ 4.5 (ต่อ) โปรแกรมการทดลองที่ 5 เรื่องการเชื่อมต่อกับจอแสดงผลแบบผลึกเหลว

```
        CLR     ACC.2       ;ENABLE=0
        MOV     DPTR,#PORT_B
        MOVX    @DPTR,A
        RET
;************************************************************
DELAY:  MOV     R1,#50H     ;DELAY TIME
DEL:    MOV     R2,#00H
DEL0:   DJNZ    R2,DEL0
        DJNZ    R1,DEL
        RET
;************************************************************
END
```

รูปที่ 4.5 (ต่อ) โปรแกรมการทดลองที่ 5 เรื่องการเชื่อมต่อกับจอแสดงผลแบบผลึกเหลว

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

จากโปรแกรมกำหนดให้แสดงข้อความทั้งหมด 5 ข้อความโดยเริ่มจากข้อความที่ 1 แล้วดับลง แสดงข้อความที่ 2 แล้วดับลง สลับกันไปถึงข้อความที่ 5 จึงวนกลับมาแสดงข้อความที่ 1 อีกครั้ง วงจรสามารถทำงานได้ตามโปรแกรมที่กำหนดไว้

ผลการทดลองทางด้านโปรแกรม

สามารถทำงานได้ตามคำสั่งที่เขียนขึ้นมีลักษณะการทำงานคล้ายกับอุปกรณ์จริง แต่จะแสดงผลช้ากว่าอุปกรณ์จริงอยู่มาก

4.2.8 การทดลองอุปกรณ์ต่อพ่วงที่ 6 เรื่องการเชื่อมต่อกับคอนโทรลสวิตซ์

การทดลอง

อุปกรณ์ต่อพ่วงที่ 6 เป็นวงจรควบคุมการทำงานของรีเลย์จำนวน 4 ตัว มีสวิตช์ควบคุมด้วยพอร์ต C จำนวน 3 ตัว และแสดงผลด้วยตัวเลข 7 ส่วน จากพอร์ต A เขียนโปรแกรมทดสอบการควบคุมได้ดังรูปที่ 4.6

```
;************************************************************
;LAB 6 CONTROL SWITCH
;PORT_A
;PORT_B
;PORT_C  1 - 3
;************************************************************
ORG 8000H
PORT_A          EQU   0E000H
PORT_B          EQU   0E001H
PORT_C          EQU   0E002H
C_PORT          EQU   0E003H

;************************************************************
SET_8255:       MOV   DPTR,#C_PORT
                MOV   A,#81H
                MOVX  @DPTR,A
;************************************************************
                MOV     R1,#00H
                MOV     R2,#00H
                MOV     R3,#00H
SCAN:           LCALL   OUT
                MOV     DPTR,#PORT_C
                MOV     A,#0FFH
                MOVX    @DPTR,A
                MOVX    A,@DPTR
                CJNE    A,#0FEH,SW2
                LJMP    SELECT
SW2:            CJNE    A,#0FDH,SW3
                LJMP    ON
SW3:            CJNE    A,#0FBH,SW4
                LJMP    OFF
SW4:            LJMP    SCAN
;************************************************************
SELECT:         LCALL   DELAY
                INC     R2
                CJNE    R2,#06H,SCAN
                MOV     R2,#00H
                LJMP    SCAN
;************************************************************
OUT:            MOV     DPTR,#DATA
                MOV     A,R2
                MOVC    A,@A+DPTR
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
                RET
DATA:           DB      0C0H,0F9H,0A4H,0B0H,99H,0C0H
;************************************************************
ON:             CJNE    R2,#01H,ON2
                LJMP    CON1_ON
ON2:            CJNE    R2,#02H,ON3
                LJMP    CON2_ON
ON3:            CJNE    R2,#03H,ON4
                LJMP    CON3_ON
ON4:            CJNE    R2,#04H,SCAN
                LJMP    CON4_ON
OFF:            CJNE    R2,#01H,OFF2
                LJMP    CON1_OFF
OFF2:           CJNE    R2,#02H,OFF3
                LJMP    CON2_OFF
OFF3:           CJNE    R2,#03H,OFF4
                LJMP    CON3_OFF
OFF4:           CJNE    R2,#04H,SCAN
                LJMP    CON4_OFF
;************************************************************
```


รูปที่ 4.6 โปรแกรมการทดลองที่ 6 เรื่องการเชื่อมต่อกับคอนโทรลสวิตช์

```
CON1_ON:   MOV  A,R1
           SETB    ACC.0
           MOV     R1,A
           LCALL   SET_OUT
           LJMP    SCAN
CON1_OFF:MOV    A,R1
           CLR     ACC.0
           MOV     R1,A
           LCALL   SET_OUT
           LJMP    SCAN
CON2_ON: MOV    A,R1
           SETB    ACC.1
           MOV     R1,A
           LCALL   SET_OUT
           LJMP    SCAN
CON2_OFF:MOV       A,R1
           CLR        ACC.1
           MOV        R1,A
           LCALL    SET_OUT
           LJMP     SCAN
CON3_ON: MOV       A,R1
           SETB     ACC.2
           MOV      R1,A
           LCALL   SET_OUT
           LJMP     SCAN
CON3_OFF:MOV       A,R1
           CLR        ACC.2
           MOV        R1,A
           LCALL    SET_OUT
           LJMP     SCAN
CON4_ON:MOV        A,R1
           SETB     ACC.3
           MOV      R1,A
           LCALL   SET_OUT
           LJMP     SCAN
CON4_OFF:MOV       A,R1
           CLR        ACC.3
           MOV        R1,A
           LCALL    SET_OUT
           LJMP     SCAN
;*************************************************************
SET_OUT:LCALL      OUT
           MOV       A,R1
           MOV       DPTR,#PORT_B
           MOVX     @DPTR,A
           RET
;*****************************************
DELAY:     MOV R6,#0FFH
LOOP1:     MOV R5,#0F0H
LOOP2:     NOP
           JNZ R5,LOOP2
DJNZ R6,LOOP1
           RET
;*****************************************
           END
```

รูปที่ 4.6 (ต่อ) โปรแกรมการทดลองที่ 6 เรื่องการเชื่อมต่อกับคอนโทรลสวิตช์

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

จากโปรแกรมกำหนดให้สวิตช์ 1 ทำงานเพื่อเลือกลำดับของรีเลย์ ว่าต้องการควบคุมรีเลย์ตัวใดโดยแสดงตำแหน่งของรีเลย์ด้วยจอแสดงผลแบบ 7 ส่วน ส่วนสวิตช์ 2 สั่งให้รีเลย์ตัวที่เลือกมีสภาวะทำงานและสวิตช์ 3 สั่งให้รีเลย์ตัวที่เลือกมีสภาวะไม่ทำงาน ปรากฏผลคือวงจรสามารถทำงานได้ตามโปรแกรมที่กำหนดไว้

ผลการทดลองทางด้านโปรแกรม

สามารถทำงานได้ตามคำสั่งที่เขียนขึ้นมีลักษณะการทำงานคล้ายกับอุปกรณ์จริง แต่จะแสดงผลช้ากว่าอุปกรณ์จริงอยู่มาก

4.2.9 การทดลองอุปกรณ์ต่อพ่วงที่ 7 เรื่องการเชื่อมต่อกับดีซีมอเตอร์

การทดลอง

อุปกรณ์ต่อพ่วงที่ 7 เป็นวงจรควบคุมทิศทางการหมุนของมอเตอร์ไฟฟ้ากระแสตรง ควบคุมด้วยพอร์ต A0 และพอร์ต A1 มีสวิตช์ควบคุมทั้งหมด 4 ตัว ควบคุมด้วยพอร์ต C ไบต์สูงเขียนโปรแกรมทดลองการควบคุมได้ดังรูปที่ 4.7

```
;*************************************************************
;LAB 7     DC MOTOR
;PORT A
;PORT C
;*************************************************************
ORG 8000H
PORT_A          EQU   0E000H
PORT_B          EQU   0E001H
PORT_C          EQU   0E002H
C_PORT          EQU   0E003H
;*************************************************************
SET_8255:       MOV   DPTR,#C_PORT
                MOV   A,#8BH
                MOVX  @DPTR,A
;*************************************************************
                MOV       R1,#00H
ST:             LCALL     SCAN
                LJMP      ST
;*************************************************************
SCAN:           MOV       DPTR,#PORT_C
                MOVX      A,@DPTR
SW1:            CJNE      A,#07FH,SW2
                LJMP      RW
SW2:            CJNE      A,#0BFH,SW3
                LJMP      FW
SW3:            CJNE      A,#0DFH,SW4
                JMP       INV
SW4:            CJNE      A,#0EFH,SW0
                LJMP      STOP
SW0:            RET
```


รูปที่ 4.7 โปรแกรมการทดลองที่ 7 เรื่องการเชื่อมต่อกับดีซีมอเตอร์

```
;************************************************************
STOP:           MOV     A,#00H
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
                LJMP    ST
;************************************************************
INV:            MOV     A,#00H
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
INV0:           MOV     A,R1
                CPL     A
                LCALL   DELAY
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
                MOV     R1,A
                LJMP    ST
;************************************************************
RW:             MOV     A,#00H
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
                LCALL   DELAY
                MOV     A,#01H
                MOVX    @DPTR,A
                MOV     R1,A
                LJMP    ST
;************************************************************
FW:             MOV     A,#00H
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
                LCALL   DELAY
                MOV     A,#02H
                MOVX    @DPTR,A
                MOV     R1,A
                LJMP    ST
;************************************************************
DELAY:          MOV     R6,#50H
LOOP1:          MOV     R5,#40H
LOOP2:          MOV     R4,#30H
LOOP3:          NOP
                DJNZ    R4,LOOP3
                DJNZ    R5,LOOP2
                DJNZ    R6,LOOP1
                RET
;************************************************************
                END
```

รูปที่ 4.7 (ต่อ) โปรแกรมการทดลองที่ 7 เรื่องการเชื่อมต่อกับดีซีมอเตอร์

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

จากโปรแกรมที่กำหนดให้สวิตช์ที่ 1 ทางด้านขวามือสั่งให้มอเตอร์หมุนไปทางขวา สวิตช์ที่ 2 สั่งให้มอเตอร์หมุนไปทางซ้าย สวิตช์ที่ 3 สั่งให้มอเตอร์หมุนไปทิศทางตรงกันข้าม และสวิตช์ที่ 4 สั่งให้มอเตอร์หยุดหมุนปรากฏว่าวงจรทำงานได้ตามโปรแกรมที่กำหนดไว้

ผลการทดลองทางด้านโปรแกรม

สามารถทำงานได้ตามคำสั่งที่เขียนขึ้นมีลักษณะการทำงานคล้ายกับอุปกรณ์จริง แต่จะแสดงผลช้ากว่าอุปกรณ์จริงอยู่มาก

4.2.10 การทดลองอุปกรณ์ต่อพ่วงที่ 8 เรื่องการเชื่อมต่อกับสเต็ปปิ้งมอเตอร์

การทดลอง

อุปกรณ์ต่อพ่วงที่ 8 เป็นวงจรควบคุมการหมุนของสเต็ปปิ้งมอเตอร์ควบคุมด้วยพอร์ต A และมีสวิตช์ทั้งหมด 5 ตัว ควบคุมสภาวะด้วยพอร์ต C เขียนโปรแกรมทดลองได้ดังรูปที่ 4.8

```
;*****************************************************
;   LAB 8  STAPPING MOTOR
;*****************************************************
ORG 8000H
PORT_A          EQU   0E000H
PORT_B          EQU   0E001H
PORT_C          EQU   0E002H
C_PORT          EQU   0E003H
;*************************************
SET_8255:       MOV   DPTR,#C_PORT
                MOV   A,#89H
                MOVX  @DPTR,A
;*************************************
                MOV     PSW,#00H
                MOV     R1,#00H
ST:             LCALL   SCAN
                LJMP    ST
;*************************************
SCAN:           MOV     PSW,#00H
                MOV     DPTR,#PORT_C
                MOVX    A,@DPTR
                CJNE    A,#0EFH,SW2
                LCALL   STOP
SW2:            CJNE    A,#0F7H,SW3
                LJMP    FW
SW3:            CJNE    A,#0FBH,SW4
                LJMP    F
SW4:            CJNE    A,#0FDH,SW5
                LJMP    R
SW5:            CJNE    A,#0FEH,SW0
                LJMP    RW
SW0:            RET
;*************************************************************
STOP:         LJMP      SET_8255
;*************************************************************
F:              LCALL   OUT_F
                LCALL   DELAY
                LCALL   SCAN
                INC     R1
                CJNE    R1,#08H,RF
                MOV     R1,#00H
RF:             LJMP    ST
FW:             LCALL   OUT_F
```


รูปที่ 4.8 โปรแกรมการทดลองที่ 8 เรื่องการเชื่อมต่อกับสเต็ปปิ้งมอเตอร์

```
                LCALL   DELAY
                LCALL   SCAN
                INC     R1
                CJNE    R1,#08H,FW
                MOV     R1,#00H
                LJMP    FW
;*************************************************************
R:              LCALL   OUT_R
                LCALL   DELAY
                LCALL   SCAN
                INC     R1
                CJNE    R1,#08H,RR
                MOV     R1,#00H
RR:             LJMP    ST
RW:             LCALL   OUT_R
                LCALL   DELAY
                LCALL   SCAN
                INC     R1
                CJNE    R1,#08H,RW
                MOV     R1,#00H
                LJMP    RW
;*************************************************************
OUT_F:          PUSH    DPH
                PUSH    DPL
                PUSH    ACC
                MOV     DPTR,#DATAF
                MOV     A,R1
                MOVC    A,@A+DPTR
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
                POP     ACC
                POP     DPL
                POP     DPH
                RET
;*************************************************************
OUT_R:          PUSH    DPH
                PUSH    DPL
                PUSH    ACC
                MOV     DPTR,#DATAR
                MOV     A,R1
                MOVC    A,@A+DPTR
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
                POP     ACC
                POP     DPL
                POP     DPH
                RET
;*************************************************************
DATAR:          DB              9H,08H,0CH,04H
                DB              06H,02H,03H,01H
;*************************************************************
DATAF:          DB              1H,03H,02H,06H
                DB              4H,0CH,08H,09H
;*************************************************************
DELAY:          MOV  R5,#10H
LOOP6:          MOV  R6,#10H
LOOP5:          MOV  R7,#10H
LOOP4:          NOP
                NOP
                DJNZ R7,LOOP4
                DJNZ R6,LOOP5
                DJNZ R5,LOOP6
                RET
```


รูปที่ 4.8 (ต่อ) โปรแกรมการทดลองที่ 8 เรื่องการเชื่อมต่อกับสเต็ปปิ้งมอเตอร์

```
;************************************************************
DELAY_          L:MOV  R5,#30H
LOOPC:          MOV  R6,#20H
LOOPB:          MOV  R7,#10H
LOOPA:          NOP
                NOP
                DJNZ R7,LOOPA
                DJNZ R6,LOOPB
                DJNZ R5,LOOPC
                RET
;************************************************************
                END
```

รูปที่ 4.8 (ต่อ) โปรแกรมการทดลองที่ 8 เรื่องการเชื่อมต่อกับสเต็ปปิ้งมอเตอร์

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

จากโปรแกรมกำหนดให้สวิตช์ที่ 1 ทางด้านขวามือสั่งให้สเต็ปปิ้งมอเตอร์ หมุนวนขวาอย่างต่อเนื่อง สวิตช์ที่ 2 สั่งให้สเต็ปปิ้งมอเตอร์หมุนวนขวา 1 จังหวะ สวิตช์ที่ 3 สั่งให้สเต็ปปิ้งมอเตอร์หมุนวนซ้าย 1 จังหวะ สวิตช์ที่ 4 สั่งให้ สเต็ปปิ้งมอเตอร์หมุนวนซ้ายอย่างต่อเนื่อง และสวิตช์ที่ 5 สั่งให้สเต็ปปิ้งมอเตอร์หยุดหมุนปรากฏว่าวงจรสามารถทำงานได้ตามโปรแกรมที่กำหนดไว้

ผลการทดลองทางด้านโปรแกรม

สามารถทำงานได้ตามคำสั่งที่เขียนขึ้นมีลักษณะการทำงานคล้ายกับอุปกรณ์จริง แต่จะแสดงผลช้ากว่าอุปกรณ์จริงอยู่มาก

4.2.11 การทดลองอุปกรณ์ต่อพ่วงที่ 9 เรื่องการสร้างเสียงสัญญาณ

การทดลอง

อุปกรณ์ต่อพ่วงที่ 9 เป็นวงจรขยายสัญญาณพัลส์ ที่ได้จากพอร์ต A ออกทางลำโพงแบบเปียสโซด้วยทรานซิสเตอร์ชนิดเอ็นพีเอ็น โดยมีสวิตช์ให้เลือกใช้งานทั้งหมด 8 สวิตช์ ควบคุมสภาวะด้วยพอร์ต C เขียนโปรแกรมทดลองได้ดังรูปที่ 4.9

```
;*********************************
***********
; LAB 9 SOUND
;*****************************************
; MAIN PROGRAM
ORG 8000H
PORT_A    EQU   0E000H
PORT_B    EQU   0E001H
PORT_C    EQU   0E002H
C_PORT    EQU   0E003H
;-----------------------------------------
SET_8255:         MOV     A,#89H
                  MOV     DPTR,#C_PORT
                  MOVX    @DPTR,A
;-----------------------------------------
SCAN:             MOV     A,#0FFH
                  MOV     DPTR,#PORT_C
                  MOVX    @DPTR,A
                  MOVX    A,@DPTR
                  CJNE    A,#0FEH,SW2
                  LCALL   SOUND1
SW2:              CJNE    A,#0FDH,SW3
                  LCALL   SOUND2
SW3:              CJNE    A,#0FBH,SW4
                  LCALL   SOUND3
SW4:              CJNE    A,#0F7H,SW5
                  LCALL   SOUND4
SW5:              CJNE    A,#0EFH,SW6
                  LCALL   SOUND5
SW6:              CJNE    A,#0DFH,SW7
                  LCALL   SOUND6
SW7:              CJNE    A,#0BFH,SW8
                  LCALL   SSOUND
SW8:              CJNE    A,#7FH,SCAN
                  LCALL   SOUND8
                  SJMP    SCAN
;*************************************************************
SOUND1:           MOV     R7,#0FH
                  MOV     DPTR,#PORT_A
                  MOV     A,#01H
S1:               MOVX    @DPTR,A
                  LCALL   DELAY
                  RL      A
                  DJNZ    R7,S1
RET
;*************************************************************
SOUND2:           MOV     R7,#0FH
                  MOV     DPTR,#PORT_A
                  MOV     A,#01H
S2:               MOVX    @DPTR,A
                  LCALL   DELAY
                  LCALL   DELAY
                  RL      A
                  DJNZ    R7,S2
                  RET
;*************************************************************
SOUND3:           MOV     R7,#0FH
                  MOV     DPTR,#PORT_A
                  MOV     A,#01H
S3:               MOVX    @DPTR,A
                  LCALL   DELAY
                  LCALL   DELAY
                  LCALL   DELAY
                  RL      A
                  DJNZ    R7,S3
                  RET
```


รูปที่ 4.9 โปรแกรมการทดลองที่ 9 เรื่องการสร้างสัญญาณเสียง

```
;************************************************************
SOUND4:         MOV     R7,#0FH
                MOV     DPTR,#PORT_A
                MOV     A,#01H
S4:             MOVX    @DPTR,A
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                RL      A
                DJNZ    R7,S4
                RET
;************************************************************
SOUND5:         MOV     R7,#0FH
                MOV     DPTR,#PORT_A
                MOV     A,#01H
S5:             MOVX    @DPTR,A
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                RL      A
                DJNZ    R7,S5
                RET
;************************************************************
SOUND6:         MOV     R7,#0FH
                MOV     DPTR,#PORT_A
                MOV     A,#01H
S6:             MOVX    @DPTR,A
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                RL      A
                DJNZ    R7,S6
                RET
;************************************************************
SOUND7:         MOV     R7,#0FH
                MOV     DPTR,#PORT_A
                MOV     A,#01H
S7:             MOVX    @DPTR,A
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                RL      A
                DJNZ    R7,S7
                RET
;************************************************************
SOUND8:         MOV     R7,#0FH
                MOV     DPTR,#PORT_A
                MOV     A,#01H
S8:             MOVX    @DPTR,A
```

รูปที่ 4.9 (ต่อ) โปรแกรมการทดลองที่ 9 เรื่องการสร้างสัญญาณเสียง

```
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                LCALL   DELAY
                RL      A
                DJNZ    R7,S8
                RET
;*************************************************************
DELAY:          MOV     R5,#05H
LOOPZ:          NOP
                NOP
                DJNZ    R5,LOOPZ
                RET
;*************************************************************
END
```

รูปที่ 4.9 (ต่อ) โปรแกรมการทดลองที่ 9 เรื่องการสร้างสัญญาณเสียง

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

จากโปรแกรมกำหนดให้สร้างสัญญาณพัลส์ทุกครั้งที่มีการกดสวิตซ์ในแต่ละสวิตซ์ มีความถี่ที่แตกต่างกันปรากฏว่า เมื่อกดสวิตซ์แต่ละสวิตซ์จะได้ยินเสียงที่แตกต่างกันจริง แสดงว่าวงจรสามารถทำงานได้ตามโปรแกรมที่กำหนดไว้

4.2.12 การทดลองอุปกรณ์ต่อพ่วงที่ 10 เรื่องการเชื่อมต่อกับเซ็นเซอร์และเคาน์เตอร์

การทดลอง

อุปกรณ์ต่อพ่วงที่ 10 เป็นวงจรตรวจจับทางแสงเมื่อมีวัตถุขวาง และไม่มีวัตถุขวางให้ผลสภาวะทางลอจิกที่ต่างกัน มีด้วยกัน 2 ชุด ควบคุมการทำงานด้วยพอร์ต C และมีตัวเลข 7 ส่วน จำนวน 4 หลักควบคุมด้วยพอร์ต A และพอร์ต B เขียนโปรแกรมทดลองได้ดังรูปที่ 4.10

```
;************************************************************
;   LAB10       LIGHT SENSOR
;   PORT_A      SEND DATA
;   PORT_B      SEND DIGID
;   PORT_C      .1-.2     LED
;               .4-.5     OPTO
;************************************************************
                ORG   8000H
PORT_A          EQU   0E000H
PORT_B          EQU   0E001H
PORT_C          EQU   0E002H
CP              EQU   0E003H
;************************************************************
SET_8255:       MOV   DPTR,#CP
                MOV   A,#88H
                MOVX  @DPTR,A
;************************************************************
RUN:            MOV     R1,#00H
                MOV     R2,#00H
                MOV     R3,#00H
                MOV     R4,#00H
NEW:            MOV     R5,#10H
GO:             MOV     DPTR,#PORT_B
                MOV     A,#0FEH
                MOVX    @DPTR,A
                LCALL   DIGID1
                LCALL   DELAY
                RL      A
                MOVX    @DPTR,A
                LCALL   DIGID2
                LCALL   DELAY
                RL      A
                MOVX    @DPTR,A
                LCALL   DIGID3
                LCALL   DELAY
                RL      A
                MOVX    @DPTR,A
                LCALL   DIGID4
                LCALL   DELAY
                MOV     DPTR,#PORT_C
                MOV     A,#0FFH
                MOVX    @DPTR,A
                MOVX    A,@DPTR
                CJNE    A,#0FFH,GO
                DJNZ    R5,GO
                INC     R1
                CJNE    R1,#0AH,NEXT
                MOV     R1,#00H
                INC     R2
NEXT:           CJNE    R2,#0AH,NEXT1
                MOV     R2,#00H
                INC     R3
NEXT1:          CJNE    R3,#0AH,NEXT2
                MOV     R3,#00H
                INC     R4
NEXT2:          CJNE    R4,#0AH,NEXT3
                MOV     R4,#00H
NEXT3:          SJMP    NEW
;************************************************************
```

รูปที่ 4.10 โปรแกรมการทดลองที่ 10 การเชื่อมต่อกับเซ็นเซอร์และเคาน์เตอร์

```
DIGID1:         PUSH    DPH
                PUSH    DPL
                PUSH    ACC
                MOV     A,R1
                MOV     DPTR,#DATA
                MOVC    A,@A+DPTR
                MOV     DPTR,#PORT_A
                MOVX    @DPTR,A
                    POP     ACC
                    POP     DPL
                    POP     DPH
                    RET
;*************************************************************
DIGID2:             PUSH    DPH
                    PUSH    DPL
                    PUSH    ACC
                    MOV     A,R2
                    MOV     DPTR,#DATA
                    MOVC    A,@A+DPTR
                    MOV     DPTR,#PORT_A
                    MOVX    @DPTR,A
                    POP     ACC
                    POP     DPL
                    POP     DPH
                    RET
;*************************************************************
DIGID3:             PUSH    DPH
                    PUSH    DPL
                    PUSH    ACC
                    MOV     A,R3
                    MOV     DPTR,#DATA
                    MOVC    A,@A+DPTR
                    MOV     DPTR,#PORT_A
                    MOVX    @DPTR,A
                    POP     ACC
                    POP     DPL
                    POP     DPH
                    RET
;*************************************************************
DIGID4:             PUSH    DPH
                    PUSH    DPL
                    PUSH    ACC
                    MOV     A,R4
                    MOV     DPTR,#DATA
                    MOVC    A,@A+DPTR
                    MOV     DPTR,#PORT_A
                    MOVX    @DPTR,A
                    POP     ACC
                    POP     DPL
                    POP     DPH
                    RET
;*************************************************************
DATA:               DB          0C0H,0F9H,0A4H,0B0H, 99H
                    DB          92H  ,  82H, 0F8H, 80H,90H
DATA0:              DB          90H,80H,0F8H,82H,92H
                    DB          99H,0B0H,0A4H,0F9H,0C0H
;*************************************
```

รูปที่ 4.10 (ต่อ) โปรแกรมการทดลองที่ 10 การเชื่อมต่อกับเซ็นเซอร์และเคาน์เตอร์

```
DELAY:          MOV  R6,#20H
LOOP2:          MOV  R7,#10H
LOOP1:          NOP
                NOP
                DJNZ R7,LOOP1
                DJNZ R6,LOOP2
                RET
;*************************************
                END
```

รูปที่ 4.10 (ต่อ) โปรแกรมการทดลองที่ 10 การเชื่อมต่อกับเซ็นเซอร์และเคาน์เตอร์

ผลการทดลองทางด้านอุปกรณ์ต่อพ่วง

จากโปรแกรมกำหนดให้ตัวเลข 7 ส่วน ทำงานเป็นตัวนับ 4 หลัก เริ่มจาก 0000 จนถึง 9999 แล้วกลับมาเป็น 0000 อีกครั้ง โดยมีข้อแม้ว่าจะนับเพิ่มค่าขึ้น 1 ค่าทุกครั้งที่มีการปิดกั้นอุปกรณ์ตรวจจับทางแสงทั้ง 2 ชุดพร้อมกัน และจะนับต่อเนื่องเมื่อยังปิดกั้นอุปกรณ์ตรวจจับแสงนั้นอยู่ ผลปรากฏว่าวงจรสามารถทำงานได้ตามโปรแกรมที่กำหนดไว้ได้อย่างถูกต้อง

ผลการทดลองทางด้านโปรแกรม

สามารถทำงานได้ตามคำสั่งที่เขียนขึ้นมีลักษณะการทำงานคล้ายกับอุปกรณ์จริง แต่จะแสดงผลช้ากว่าอุปกรณ์จริงอยู่มาก

4.3 การทดลองโปรแกรม

วิธีทำการทดลองชุดคำสั่งของไมโครคอนโทรลเลอร์ MCS-51 โดยแบ่งการทดสอบได้ 5 โปรแกรม แต่ละโปรแกรมจะครอบคลุมคำสั่งต่างๆ ที่มีอยู่ใน MCS-51 เกือบทั้งหมดและทดลองอุปกรณ์ต่อพ่วงที่สร้างขึ้นอีก 6 อย่างโดยมีรายละเอียดดังนี้

การทดลองโปรแกรมที่ 1

เป็นโปรแกรมที่รวมกลุ่มคำสั่งการเคลื่อนย้ายข้อมูล และการประมวลผลทางคณิตศาสตร์ไว้ด้วยกัน โดยมีคำสั่งดังรูปที่ 4.11

```
ORG 8000H
START:
                MOV  A,#88H
                MOV R0,A
                MOV @R0,#31H
                MOV 0F0H,88H
                MOV A,#96H
                DIV  AB
                MOV R6,A
                MOV R7,0F0H
                SJMP MULTIPLIE
                SUBB A,R0
                DEC  0F0H
MULTIPLIE:
                MOV R4,06H
                MOV A,R7
                ADDC A,R4
                MOV 0F0H,R7
                MUL AB
                AJMP START
```

รูปที่ 4.11 คำสั่งที่ใช้ในการทดลองโปรแกรมที่ 1

ผลการทดลองโปรแกรมที่ 1

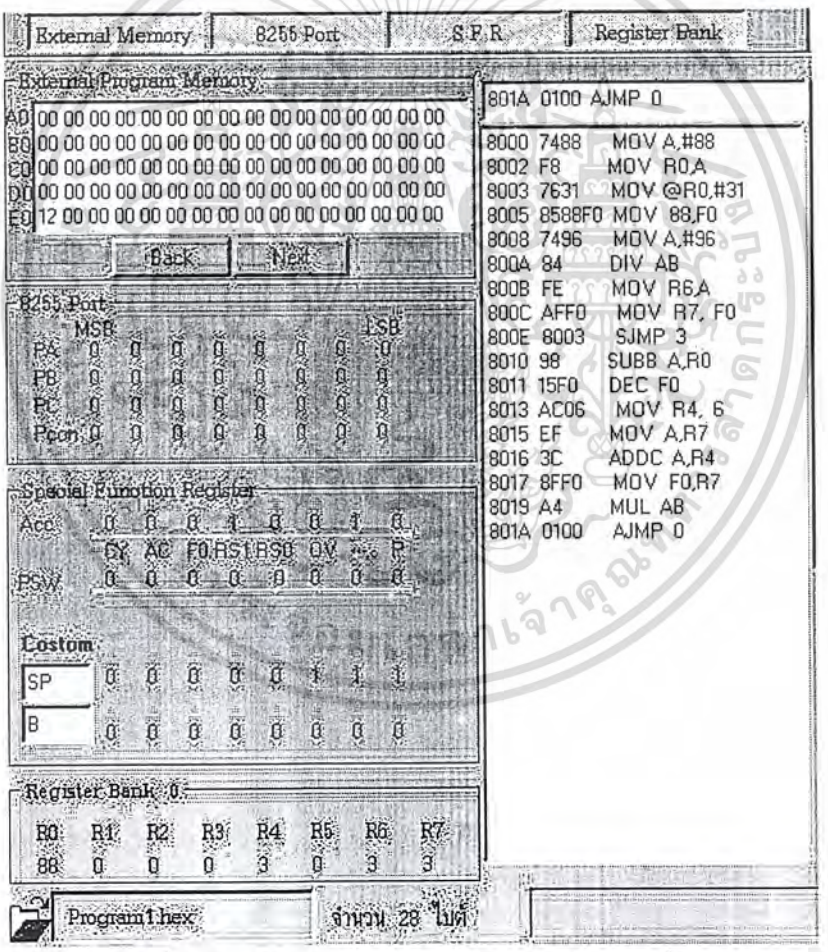
ผลการทำงานที่ได้จากการคำนวณและจากการทำงานของโปรแกรม มีผลดังตารางที่ 4.1 และหน้าจอแสดงผลการทดลองโปรแกรมที่ 1 ดังรูปที่ 4.12

ตารางที่ 4.1 การเปรียบเทียบการทำงานการทดลองโปรแกรมที่ 1

คำสั่ง	การคำนวณ	การทำงานของโปรแกรม
MOV A,#88H	Acc = 88H	Acc = 88H
MOV RO,A	R0 = 88H	R0 = 88H
MOV @RO, #31H	Tcon = 31H	Tcon = 31H
MOV 88H, OFOH	B = 31H	B = 31H
MOV A, #96H	Acc = 96H	Acc = 96H
DIV AB	3 เศษ 3	Acc = 3 B = 3
MOV R6, A	R6 = 3	R6 = 3
MOV R7, 0F0H	R7 = 3	R7 = 3
SJMP 3	PC = 8010 3 = 8013H	PC = 8013H
MOV R4, 06H	R4 = 3	R4 = 3
MOV A, R7	Acc = 3	Acc = 3

ตารางที่ 4.1 (ต่อ) การเปรียบเทียบการทำงานการทดลองโปรแกรมที่ 1

คำสั่ง	การคำนวณ	การทำงานของโปรแกรม
ADDC A, R4	Acc = 6	Acc = 6
MOV 0F0H, R7	B = 3	B = 3
MUL AB	12 H	Acc = 12H B = 0
AJMP 00	PC = 8000H	PC = 8000H



รูปที่ 4.12 หน้าจอแสดงผลการทดลองโปรแกรมที่ 1

การทดลองโปรแกรมที่ 2

โปรแกรมนี้จะเน้นการทำงานในระดับบิต และตรวจสอบสภาวะการทำงาน ของคำสั่งที่มีผลต่อค่าภายใน PSW รวมทั้งการกระโดดทำงานในลักษณะมีข้อแม้ด้วย โดยมีคำสั่งดังนี้

```
ORG 8000H
START:
                    MOV A,#80H
SE1:              RR  A
                    CJNE A,#10H,SE1
                    MOV A,#01H
SE2:              RLC A
                    MOV R3,A
                    CJNE R3,#0CH,SE2
                    CLR C
                    CPL A
                    SETB C
                    ADDC A,#14H
                    SJMP START
```

รูปที่ 4.13 คำสั่งที่ใช้ในการทดลองโปรแกรมที่ 2

ผลการทดลองโปรแกรม

ผลการทำงานที่ได้จากการคำนวณและจากการทำงานของโปรแกรม มีผลดังตารางที่4.2 และหน้าจอแสดงผลการทดลองโปรแกรมที่ 2 ดังรูปที่ 4.14

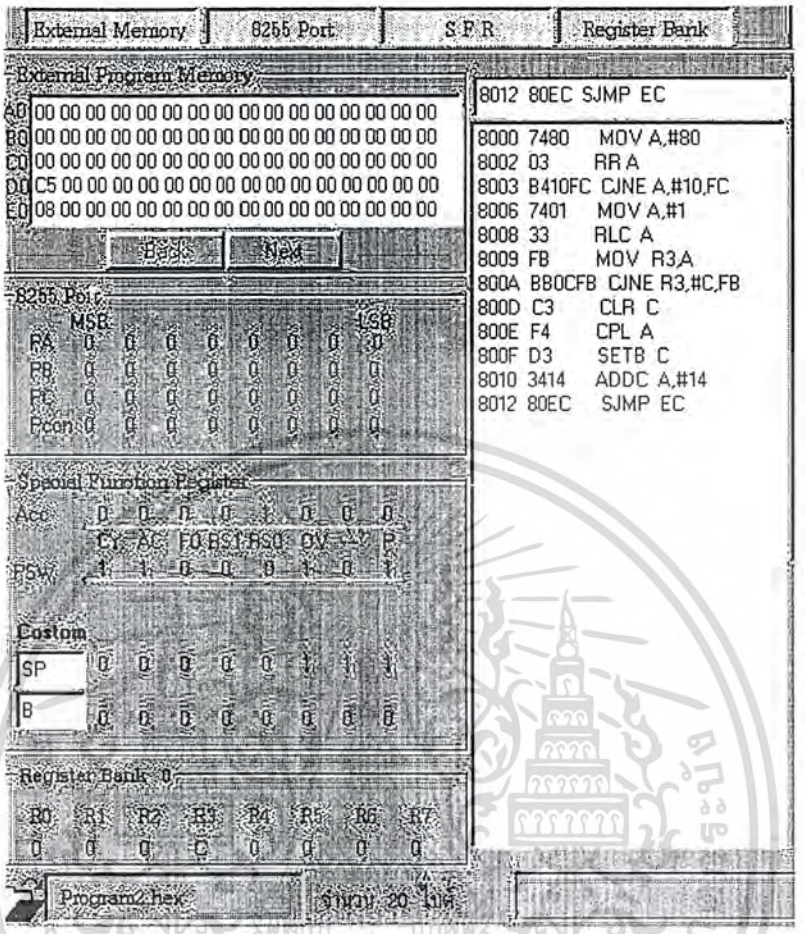
ตารางที่ 4.2 การเปรียบเทียบการทำงานการทดลองโปรแกรมที่ 2

คำสั่ง	การคำนวณ	การทำงานของโปรแกรม
MOV A, #80H	Acc = 80H	Acc = 80H
RR A	10000000 เป็น 0100000 = 40	Acc = 40H
CJNE A, #10, FC	PC = 8005H3 = 8002H	PC = 8002H
RR A	01000000 เป็น 00100000	Acc = 20H
CJNEA, #10, FC	PC = 8005H-3 = 8002H	PC 8002H
RR A	00100000 เป็น 00010000	Acc = 10H
CJNEA, #10, FC	PC = 8005H+8006H	PC = 8006, PSW = 81H
MOV A, #01H	Acc = 1	Acc = 1

ตารางที่ 4.2 (ต่อ) การเปรียบเทียบการทำงาน

การทดลองโปรแกรมที่ 2

คำสั่ง	การคำนวณ	การทำงานของโปรแกรม
RLC A	00000001 เป็น 00000011	Acc = 3, PSW = 0
MOV R3, A	R3 = 3	R3 = 3
CJNE R3, #0CH, FB	PC = 800CH –4 = 8008H	PC = 8008H
RLC A	00000011 เป็น 00000110	Acc = 06H
MOV R3, A	R3 = 6	R3 = 6
CJNE R3, #0CH, FB	PC = 800CH-4 = 8008H	PC = 8008H
RLC A	00001101 เป็น 00001100	Acc = 0CH
MOV R3, A	R3 = 0CH	R3 = 0CH
CJNE R3, #0CH, FB	PC = 800CH+1 = 800DH	PC = 800D, PSW = 81H
CLR C	PSW = 0	PSW = 0
CPL A	Acc = F3H	Acc F3H
SETB C	PSW = 81	PSW = 81
ADDC A, #14H	Acc = F3H+01+14H = 108	Acc = 18, PSW = C5
SJMP EC	PC = 8013H-13H = 8000H	PC = 8000H



รูปที่ 4.14 หน้าจอแสดงผลการทดลองโปรแกรมที่ 2

การทดลองโปรแกรมที่ 3

คำสั่งการกระทำในทางตรรกศาสตร์ ในรูปแบบข้อมูลแบบไบต์และแบบบิต และการเปรียบเทียบในระดับบิต รวมถึงการส่งลดค่าและเทียบค่าโดยมีคำสั่งดังรูปที่ 4.15 และหน้าจอแสดงผลการทดลองโปรแกรมที่ 3 ดังรูปที่ 4.16

```
ORG 8000H
START:
                MOV 00H,#55H
                MOV 0E0H,#0AAH
                ORL  A,R0
                MOV R7,#0FFH
SE1:          DEC A
                XRL A,R7
                JB 0E0H,SE2
                RR A
                RLC A
SE2:          MOV R1,#0E0H
                INC @R1
                SWAP A
```

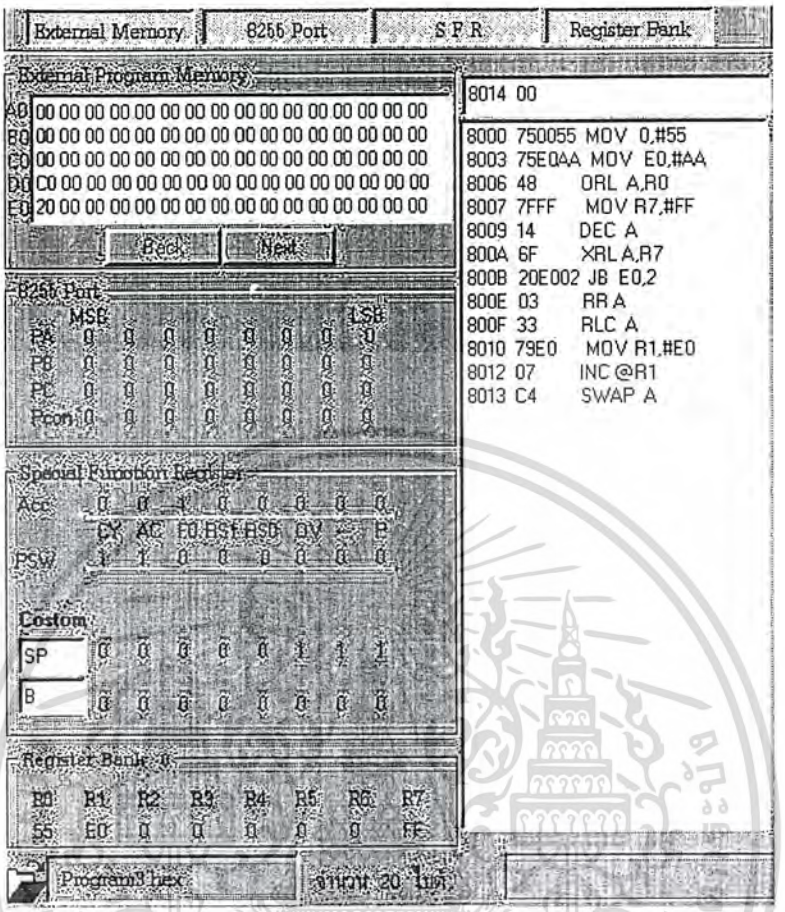
รูปที่ 4.15 คำสั่งที่ใช้ในการทดลองโปรแกรมที่ 3

ผลการทดลองโปรแกรม

ผลการทำงานที่ได้จากการคำนวณและจากการทำงานของโปรแกรม มีผลดังตารางที่ 4.3

ตารางที่ 4.3 การเปรียบเทียบการทำงานการทดลองโปรแกรมที่ 3

คำสั่ง	การคำนวณ	การทำงานของโปรแกรม
MOV 00H, #55H	R0 = 55H	R0 = 55H
MOV 0E0H, #0AAH	Acc = AAH	Acc = AAH
ORL A, R0	10101010O8O1O1O101 = 11111111	Acc = FFH
MOV R7, #FFH	R7 = FFH	R7 = FFH
DEC A	FFH-1 = FEH	Acc = FEH
XRL A, R7	11111110X or 11111111 = 00000001	Acc = 01H
JB 0E0, 2	PC = 800EH+2 = 80L0H	PC = 8010H
MOV R1, 0E0H	R1 = E0H	R1 = ECH
INC @R1	Acc = C2H	Acc = 02H
SWAP A	Acc = 20H	Acc = 20H



รูปที่ 4.16 หน้าจอแสดงผลการทดลองโปรแกรมที่ 3

การทดลองโปรแกรมที่ 4

คำสั่งเรียกใช้โปรแกรมย่อยและคำสั่งที่ให้ SP เป็นตัวชี้ที่เก็บข้อมูล และการเคลื่อนย้ายข้อมูล 16 บิต และการเรียกตำแหน่งแบบ 16 บิต และการย้อนกลับมาทำโปรแกรมเดิมจากตำแหน่งที่กระโดดมาด้วย โดยมีคำสั่งดังรูปที่ 4.17

```
ORG 8000H
START:
                MOV PSW,#10H
                MOV R0,#03H
                MOV R1,#15H
                MOV R2,#0F0H
                LCALL SE
                MOV DPTR,#1234H
                INC DPTR
                PUSH 10H
                PUSH 11H
                PUSH 12H
                POP    10H
                POP    11H
                POP    12H
SE:
                MOV A,R0
                ADD   A,#02H
                MOV R0,A
                MOV   A,R1
                SUBB A,#05H
                MOV R1,A
                MOV A,R2
                SUBB A,#0A0H
                MOV R2,A
                RET
```

รูปที่ 4.17 คำสั่งที่ใช้ในการทดลองโปรแกรมที่ 4

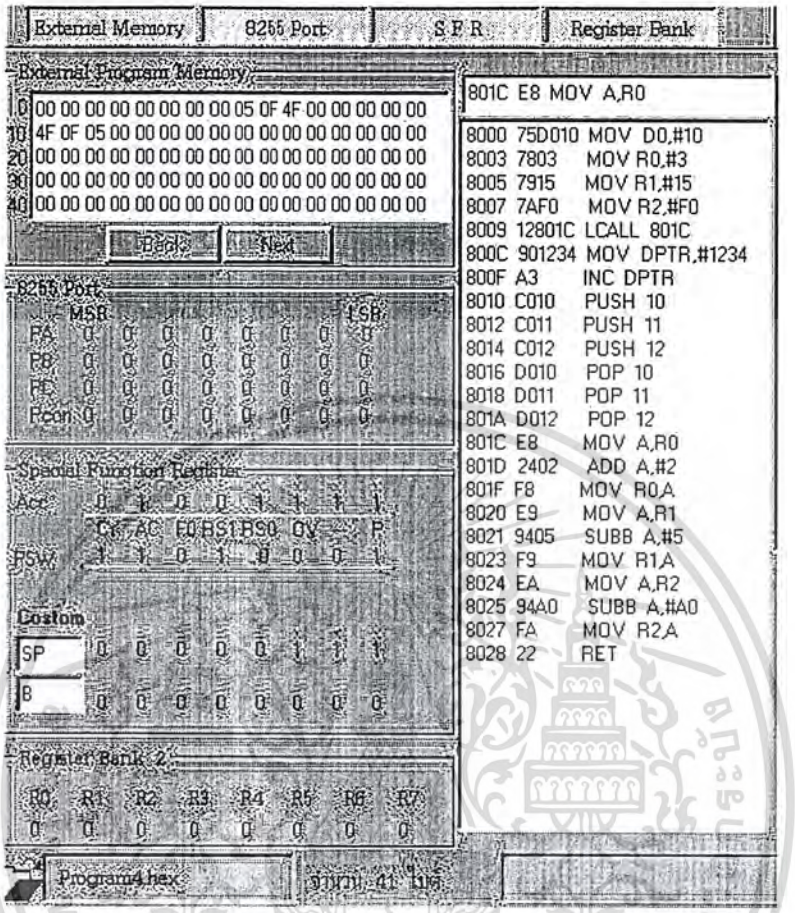
ผลการทดลองโปรแกรม

ผลการทำงานที่ได้จากการคำนวณและจากการทำงานของโปรแกรม มีผลดังตารางที่ 4.4 และหน้าจอแสดงผลการทดลองโปรแกรมที่ 4 ดังรูปที่ 4.18

ตารางที่ 4.4 การเปรียบเทียบการทำงานการ

ทดลองโปรแกรมที่ 4

คำสั่ง	การคำนวณ	การทำงานของโปรแกรม
MOV D0H, #10	PSW = 10H Bank =2	PSW = 10H
MOV R0, #03H	R0 = 3H	R0 (bank 2) = 3H
MOV R1, #15H	R1 = 15H	R1 (bank 2) = 15H
MOV R2, #0F0h	R2 = F0H	R2 (bank 2) = F0H
LCALL	PC = 801C SP = 9	SP = 20/C SP = 9
MOV A, R0	Acc = 3H	Acc = 3H
ADD A,#C2H	Acc = 3+2 = 5	Acc = 5H
MOV RO, A	R0 = 54	R0 (bank 2) = 5H
MOV A,R1	Acc = 15H	Acc = 15H
SUBB A, #05H	Acc = 15H-5H = 10H	Acc = 10H
MOV R1, A	R1 = 10H	R1 (bank 2) = 10H
MOV A, R2	Acc = F0H	Acc = F0H
SUBB A, #0A0H	Acc = F0H-ACH = 50H	Acc = 50H
MOV R2, A	R2 = 50H	R2 (bank 2) = 50H
RET	PC = 800C, SP = 7	PC = 800C, SP = 7
MOV DPTR, #1234H	DPL = 34H, DPH = 12H	DPL = 34H, DPH = 12H
IWC DPTR	DPL = 35H, DPH = 12H	DPL = 35H, DPH = 12H
PUSH 10H	SP = 8, R0 = 05H	SP = 8, RO (bank1) = 05H
PUSH 11H	SP = 9, R1 = 10H	SP = 9, R1(bank1) = 10H
PUSH 12H	SP = A, R2 = 50H	SP = A, R2(bank1) = 50H
POP 10H	R0 = 50H, SP = 9	RO(bank2) = 50H, SP = 9
POP 11H	R1 = 10H, SP = 8	R1 (bank2) = 10H, SP = 8
POP 12H	R2 = 05H, SP = 7	R2(bank2) = 05H, SP=7



รูปที่ 4.18 หน้าจอแสดงผลการทดลองโปรแกรมที่ 4

การทดลองโปรแกรมที่ 5

คำสั่งการแลกเปลี่ยนข้อมูลระหว่างหน่วยความจำภายใน คำสั่งกระโดดไปตามตำแหน่งด้วยการตรวจค่าในรีจิสเตอร์ A การเคลื่อนย้ายข้อมูลไปยังหน่วยความจำภายนอกและการรับค่าจากหน่วยความจำภายนอกหรือ ตำแหน่งอุปกรณ์ภายนอกโดยมีคำสั่งดังรูปที่ 4.19

```
ORG 8000H
START:
                SETB 0E0H
                SETB 0E3H
                SETB 0E6H
                ADD  A,#25H
                MOV R4,#05H
                XCH  A,R4
                MOV R1,#04H
                XCHD A,@R1
               MOV DPTR,#0E000H
               MOVX @DPTR,A
```

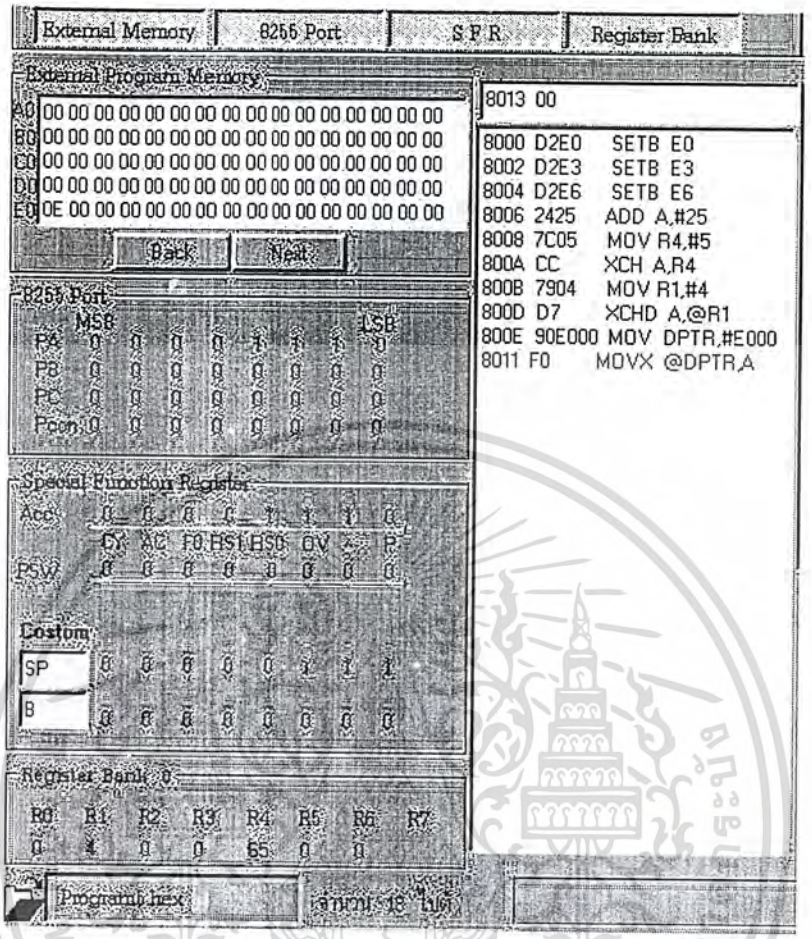
รูปที่ 4.19 คำสั่งที่ใช้ในการทดลองโปรแกรมที่ 5

ผลการทดลอง

ผลการทำงานที่ได้จากการคำนวณและจากการทำงานของโปรแกรม มีผลดังตารางที่ 4.5 และหน้าจอแสดงผลการทดลองดังรูปที่ 4.20

ตารางที่ 4.5 การเปรียบเทียบการทำงานการทดลองโปรแกรมที่ 5

คำสั่ง	การคำนวณ	การทำงานของโปรแกรม
SETB 0E0H	00000000 เป็น 00000001	Acc = 1H
SETB 0E3H	00000001 เป็น 00001001	Acc = 9H
SETB 0E6H	00001001 เป็น 01001001	Acc = 49H
ADD A, #25H	Acc = 49H + 25H = 6EH	Acc = 6EH
MOV RA, #05H	R4 = 5H	R4 = 5H
XCH A, R4	Acc = 5H, R4 = 6EH	Acc = 5H, R4 = 6EH
MOV R1, H04H	R1 = 4H	R1 = 4H
XCHD A,@R1	Acc = EH, R4 = 65H	Acc = EH, R4 = 65H
MOV DPTR, 10E00H	DPH = E0H,DPL = 00H	DPH = E0H, DPL = 00H
MOVX @DPTR, A	Port A = EH	Port A =EH



รูปที่ 4.20 หน้าจอแสดงผลการทดลองโปรแกรมที่ 5

บทที่ 5
บทสรุป ปัญหา แนวทางแก้ไขและพัฒนา

5.1 บทสรุป

ปริญญานิพนธ์ฉบับนี้ เป็นการนำเสนอโปรแกรมจำลองการทำงานของ MCS-51 และอุปกรณ์ต่อพ่วงผ่านทางพอร์ตอนุกรมซึ่งในเนื้อหาโดยรวมจะเป็นโปรแกรมที่ ทำงานบนระบบปฏิบัติการวินโดวส์ความสามารถของโปรแกรมคือสามารถเขียนโปรแกรมภาษาแอสแซมบลี และแปลงให้เป็นไฟล์ฐาน 16 รวมทั้งแสดงข้อผิดพลาดของการเขียนได้ด้วย การแสดงไฟล์นำสกุล .LST ขึ้นมาทั้งหมดนี้จะทำงานบนระบบปฏิบัติการวินโดวส์ทั้งสิ้น หลังจากได้ไฟล์เลขฐาน 16 สามารถนำไปใช้งานได้ 2 ประการคือ ประการแรกส่งไฟล์เลขฐาน 16 นั้นไปยังหน่วยความจำของวงจรไมโครคอนโทรลเลอร์และสั่งให้โปรแกรมทำงานหรือหยุดทำงานได้ด้วยการควบคุมของคอมพิวเตอร์ อีกประการหนึ่งคือนำไฟล์เลขฐาน 16 นั้นไปจำลองการทำงานด้วยโปรแกรมบนคอมพิวเตอร์ สามารถทำงานได้ทีละขั้นตอนทำให้เข้าใจและหาข้อผิดพลาดของโปรแกรม ได้ดียิ่งขึ้นนอกจากนั้นแล้ว ยังมีอุปกรณ์ต่อพ่วงซึ่งใช้งานกับอุปกรณ์ประเภทต่างๆ ครอบคลุมการใช้งานจริงถึง 10 วงจรด้วยกันทั้งหมดที่กล่าวมานี้ เป็นความสามารถที่ได้จากการสร้างโครงงานปริญญานิพนธ์ฉบับนี้ สามารถนำไปใช้งานจริงหรือนำไปประกอบการสอน ในวิชาไมโครโปรเซสเซอร์ได้สะดวกมากขึ้น เพราะทั้งหมดทำงานบนปฏิบัติการวินโดว์ส ซึ่งเป็นมาตรฐานที่นิยมใช้ในปัจจุบัน

อย่างไรก็ตาม ในการจัดทำโปรแกรมจำลองการทำงานของ MCS-51 และอุปกรณ์ต่อพ่วงผ่านทางพอร์ตอนุกรมนี้ ประสบปัญหาในการจัดทำหลายประการ จึงขอเสนอแนวทางแก้ไขปัญหาและการพัฒนาที่น่าเป็นไปได้ดังนี้

5.2 ปัญหาในการสร้าง

1) การติดต่อระหว่างวงจรไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ จำเป็นต้องมีโปรแกรมรองรับการทำงานดังกล่าวนี้คือ โปรแกรมมอนิเตอร์ ซึ่งมีความสลับซับซ้อนมาก

2) การออกแบบอุปกรณ์ต่อพ่วงมีปัญหาและข้อผิดพลาดอยู่เสมอ

3) การเขียนโปรแกรมด้วยวิชวลเบสิก ซึ่งเป็นภาษาที่ไม่คุ้นเคย และไม่ถนัดเนื่องจากไม่ได้ศึกษามาโดยตรง จึงต้องใช้เวลามากในตอนต้น เพื่อทำความเข้าใจการใช้งาน

4) การออกแบบพอร์ตใช้งาน 8255 ใช้งานได้เฉพาะอุปกรณ์ต่อพ่วงทั้ง 10 วงจรนี้เท่านั้น เพราะไม่ได้อ้างอิงมาตรฐานจากบริษัทใด เป็นมาตรฐานเฉพาะ

5) ความเร็วในการทำงานประมวลผลด้วยคอมพิวเตอร์ ช้ากว่าการประมวลของวงจรไมโครคอนโทรลเลอร์มากตัวจำลองอุปกรณ์ต่อพ่วงบางตัว แสดงผลช้ามากเช่น ดอทเมตริกซ์, จอแสดงผล 7 ส่วน เป็นต้น เวลาดูผลการทำงานจึงไม่ค่อยมีความต่อเนื่อง

6) อุปกรณ์จำลองที่เป็นอินพุตเช่น สวิตซ์, ออปโต้เป็นต้น การกระทำต่ออุปกรณ์ตัวนั้นๆ จะต้องกระทำให้ตรงกับโปรแกรมที่เขียนขึ้นด้วย จึงจะได้ผลดี

5.3 การแก้ปัญหา

1) ออกแบบโปรแกรมมอนิเตอร์ขึ้นเอง โดยใช้ข้อกำหนดต่างๆ ร่วมกันกับการออกแบบโปรแกรมด้วยวิชวลเบสิกบนคอมพิวเตอร์ ในการสื่อความหมาย

2) พยายามปรับปรุงและทดลองวงจรก่อน สร้างวงจรจริงทุกครั้ง

3) ใช้เวลาในการศึกษาโปรแกรมให้มากๆ ในช่วงแรกๆ จึงใช้เวลาค่อนข้างมากในการเขียนโปรแกรม และพยายามแยกงานออกเป็นส่วนๆ เพื่อแบ่งงานกันทำได้โดยไม่เสียเวลา

4) มีการแปลงพอร์ตให้สามารถใช้งานร่วมกันกับวงจรมาตรฐานของบริษัทอื่นๆ ได้

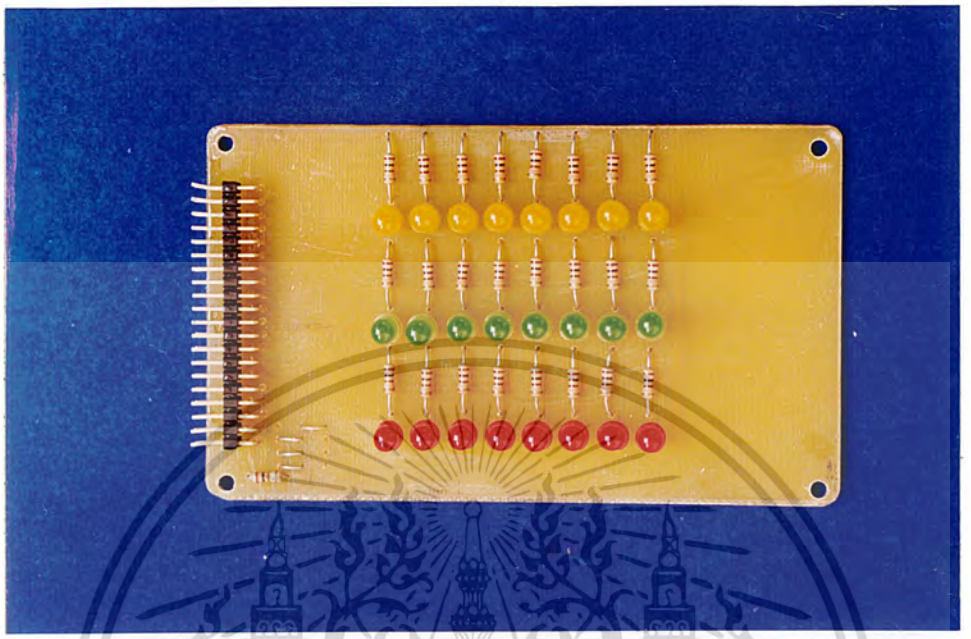
5.4 แนวทางการพัฒนาโครงงาน

1) สามารถสร้างวงจรของอุปกรณ์ต่อพ่วงอื่น มาใช้งานร่วมกับวงจรไมโครคอนโทรลเลอร์ได้อย่างต่อเนื่อง

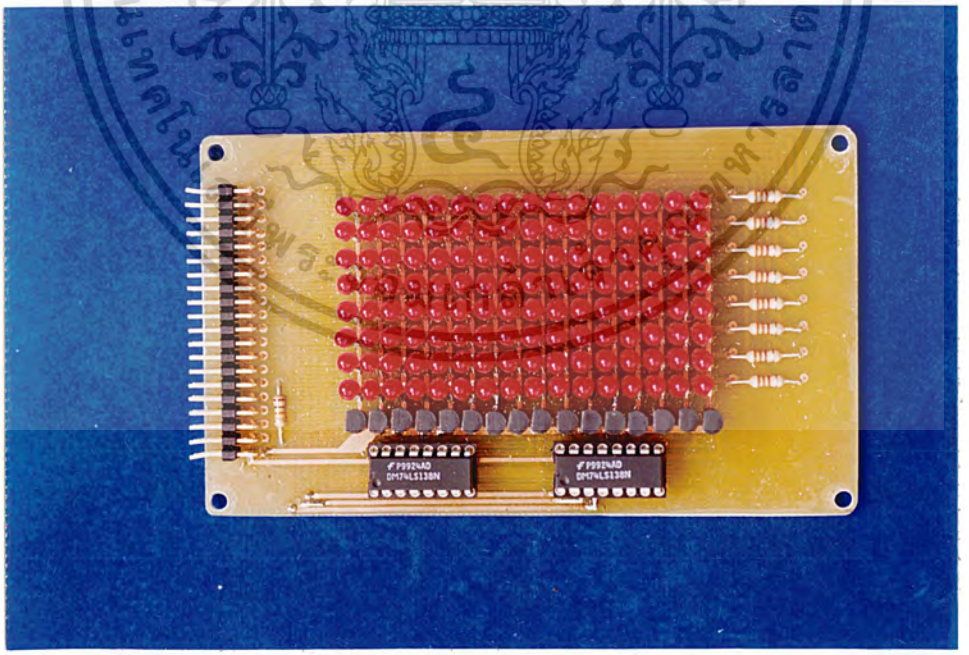
2) หาวิธีในการเขียนโปรแกรมเพื่อให้การจำลองการทำงานบนคอมพิวเตอร์ มีความเร็วเท่ากับการประมวลผลของไมโครคอนโทรลเลอร์

3) สร้างการจำลองอุปกรณ์ต่อพ่วง ให้มีจำนวนมากยิ่งขึ้น

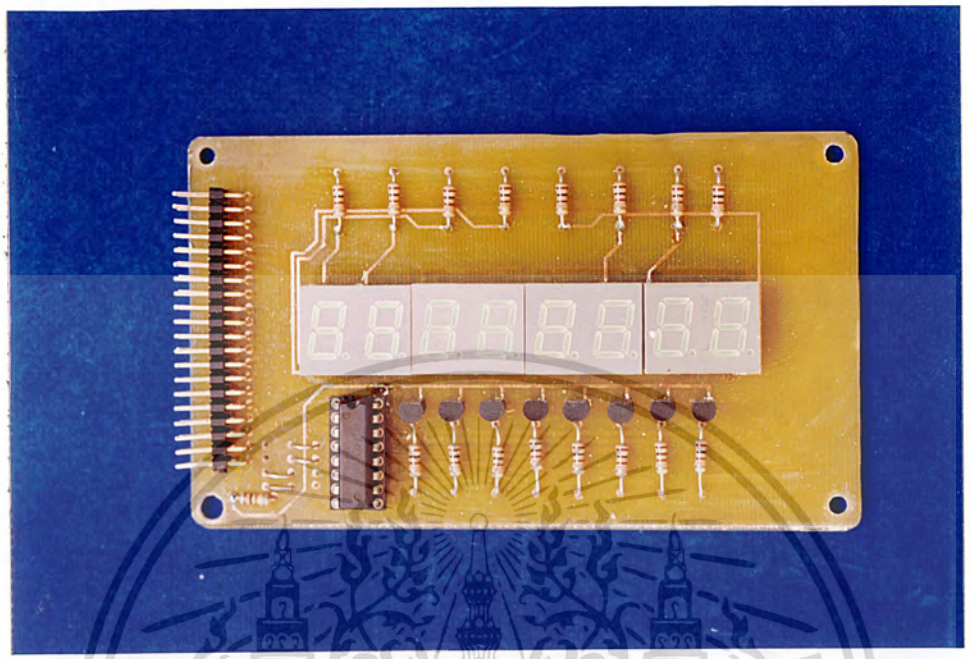
ภาคผนวก ก
เครื่องต้นแบบ



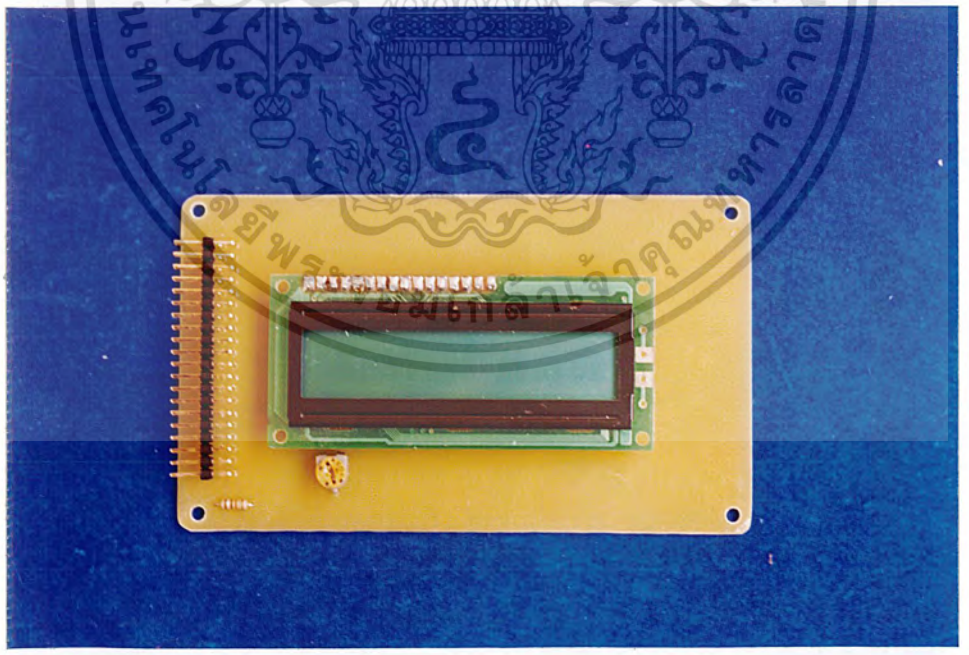
รูปที่ ก.1 แผงอุปกรณ์ต่อพ่วงสำหรับทดลองเรื่อง แอลอีดีลอจิก



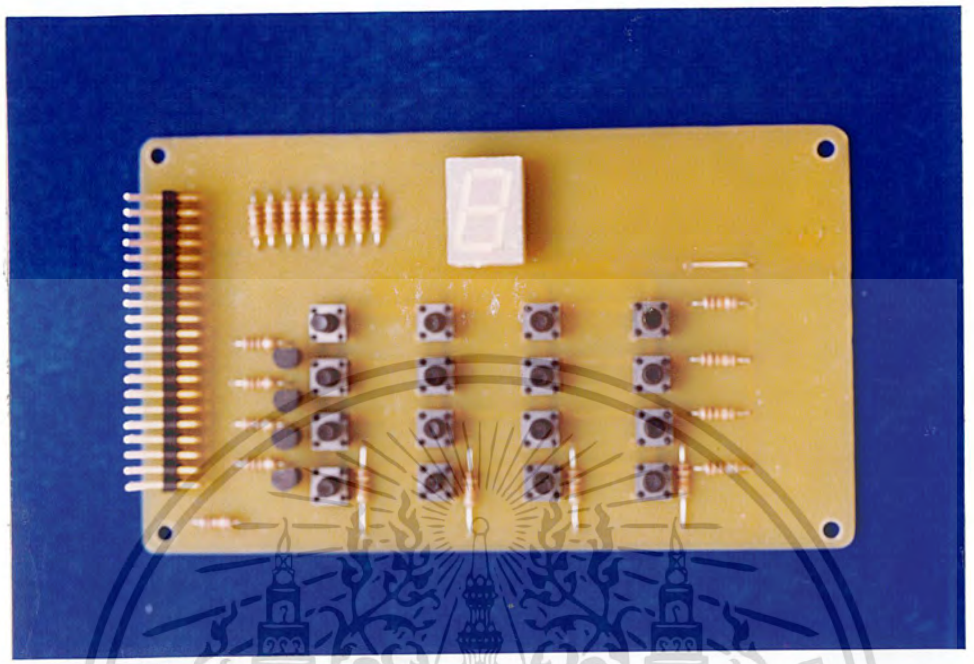
รูปที่ ก.2 แผงอุปกรณ์ต่อพ่วงสำหรับทดลองเรื่อง แอลอีดีดอตเมตริกซ์



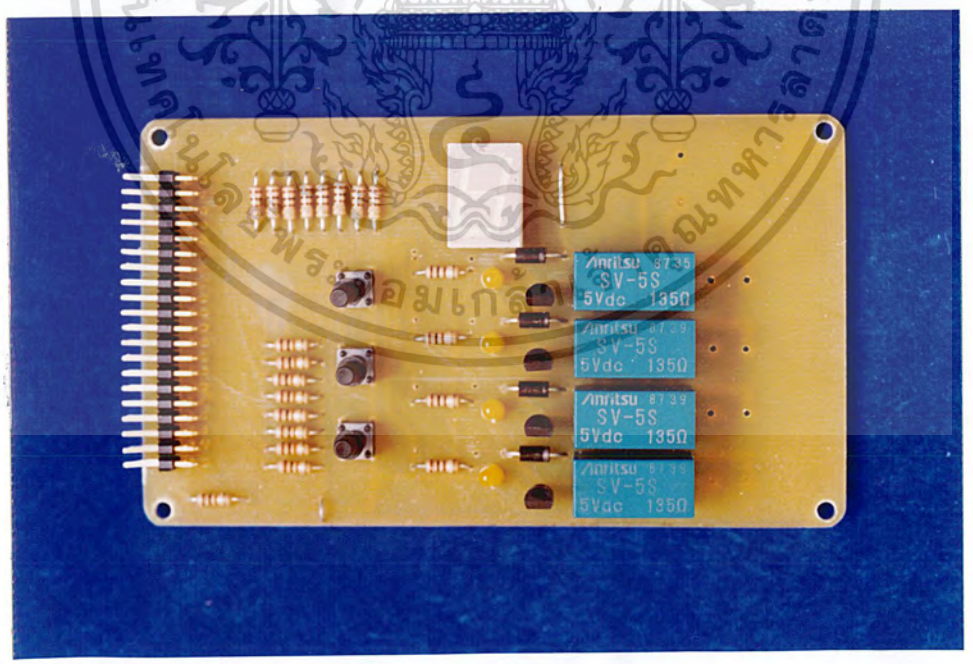
รูปที่ ก.3 แผงอุปกรณ์ต่อพ่วงสำหรับทดลองเรื่อง จอแสดงผลแบบ 7 ส่วน



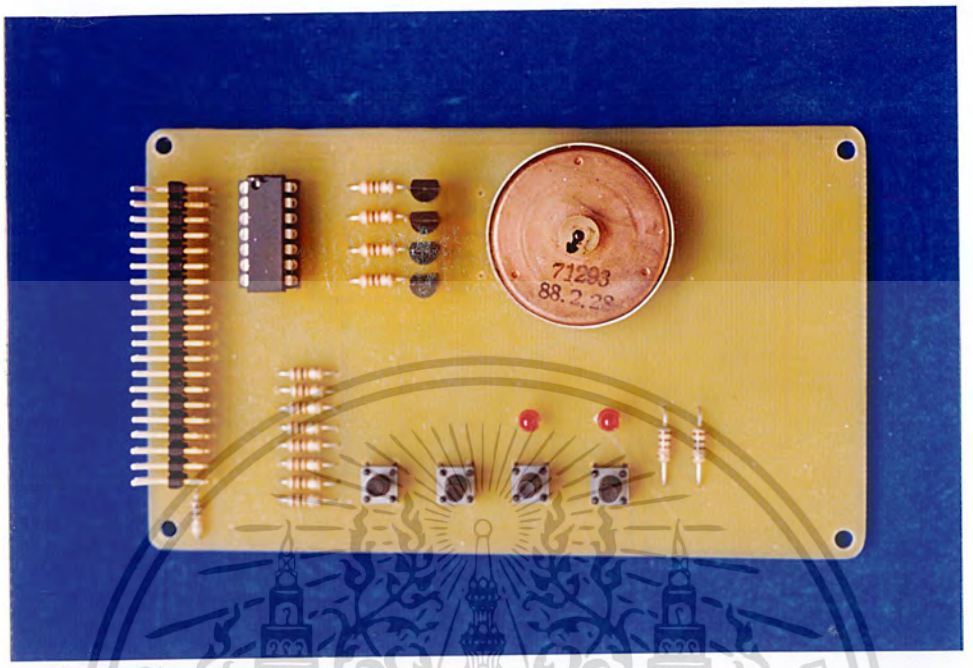
รูปที่ ก.4 แผงอุปกรณ์ต่อพ่วงสำหรับทดลองเรื่อง จอแสดงผลแบบผลึกเหลว



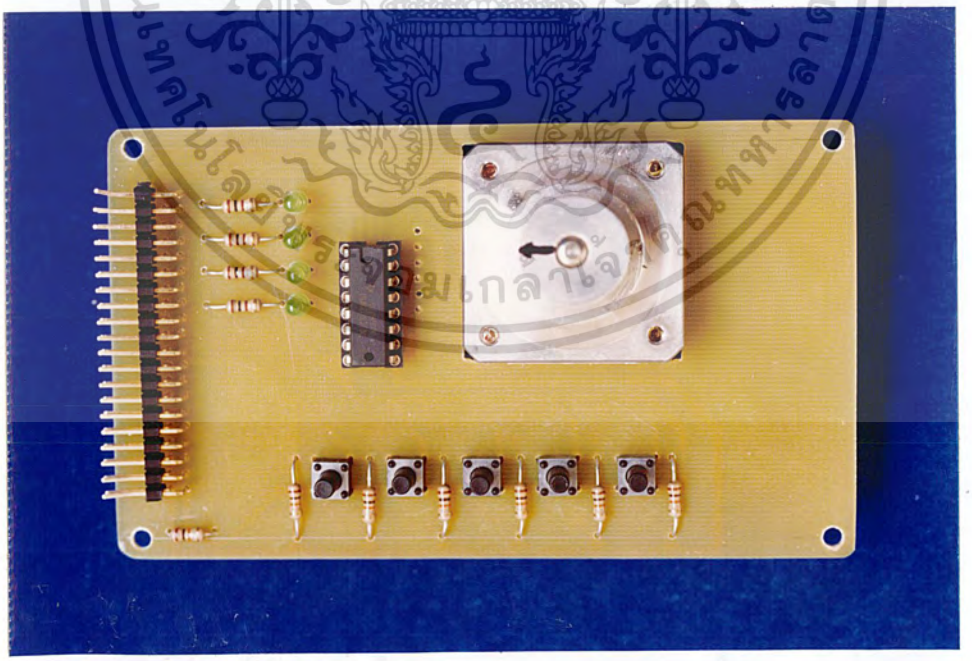
รูปที่ ก.5 แผงอุปกรณ์ต่อพ่วงสำหรับทดลองเรื่อง สวิตช์เมตริกซ์



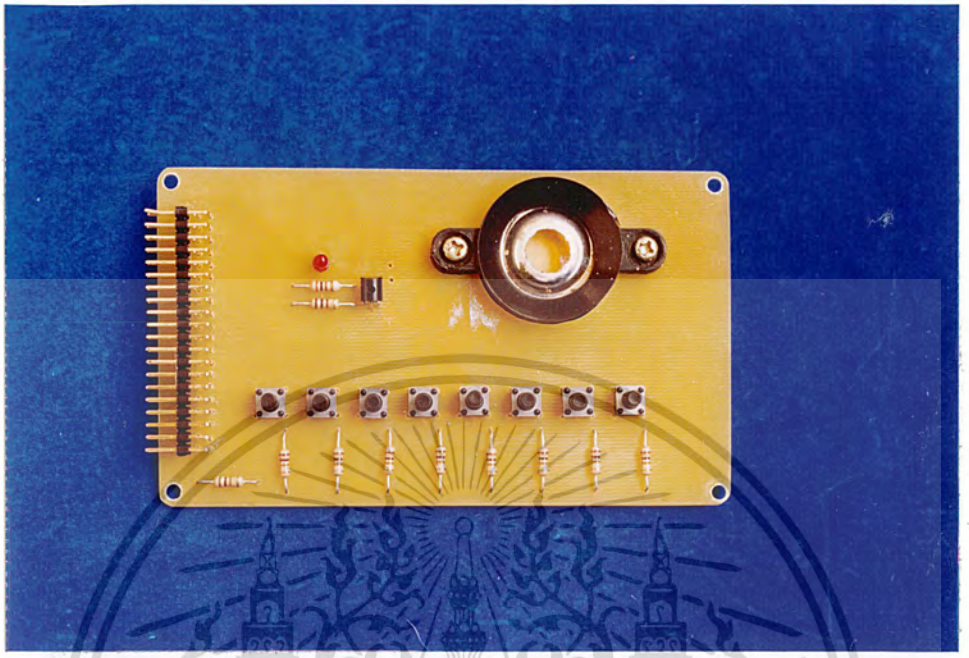
รูปที่ ก.6 แผงอุปกรณ์ต่อพ่วงสำหรับทดลองเรื่อง คอนโทรลสวิตช์



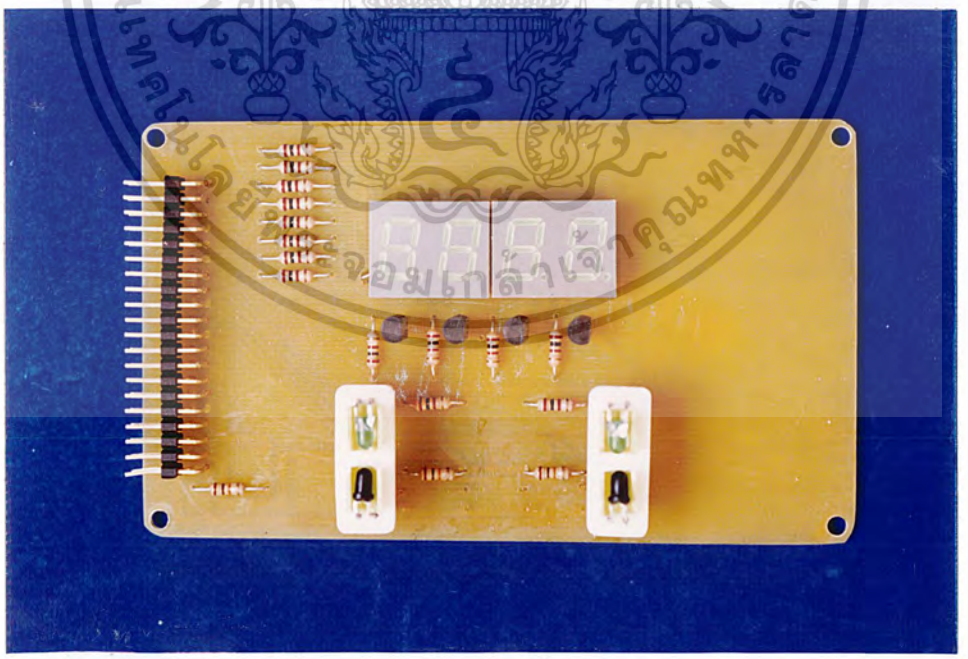
รูปที่ ก.7 แผงอุปกรณ์ต่อพ่วงสำหรับทดลองเรื่อง มอเตอร์ไฟฟ้ากระแสตรง



รูปที่ ก.8 แผงอุปกรณ์ต่อพ่วงสำหรับทดลองเรื่อง สเต็ปปิ้งมอเตอร์



รูปที่ ก.9 แผงอุปกรณ์ต่อพ่วงสำหรับทดลองเรื่อง วงจรกำเนิดเสียง

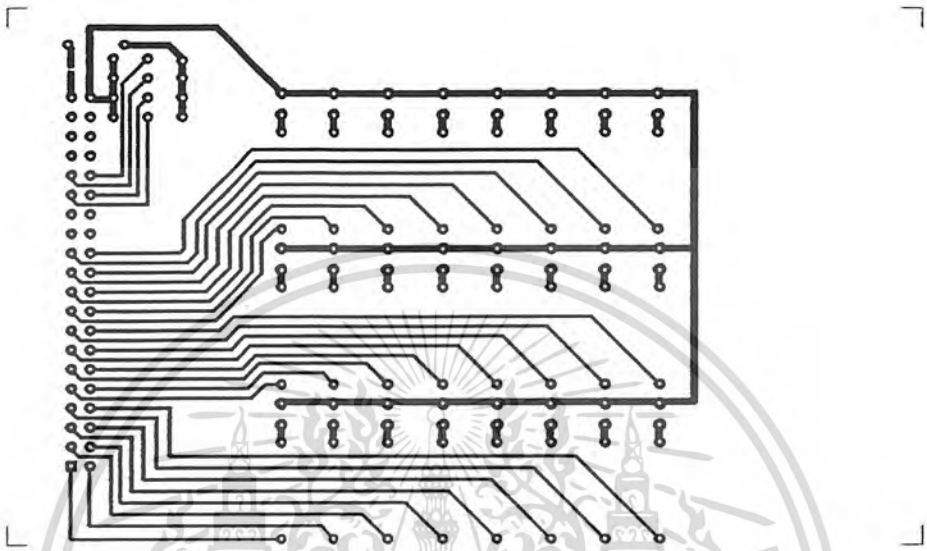


รูปที่ ก.10 แผงอุปกรณ์ต่อพ่วงสำหรับทดลองเรื่อง เซ็นเซอร์และเคาน์เตอร์

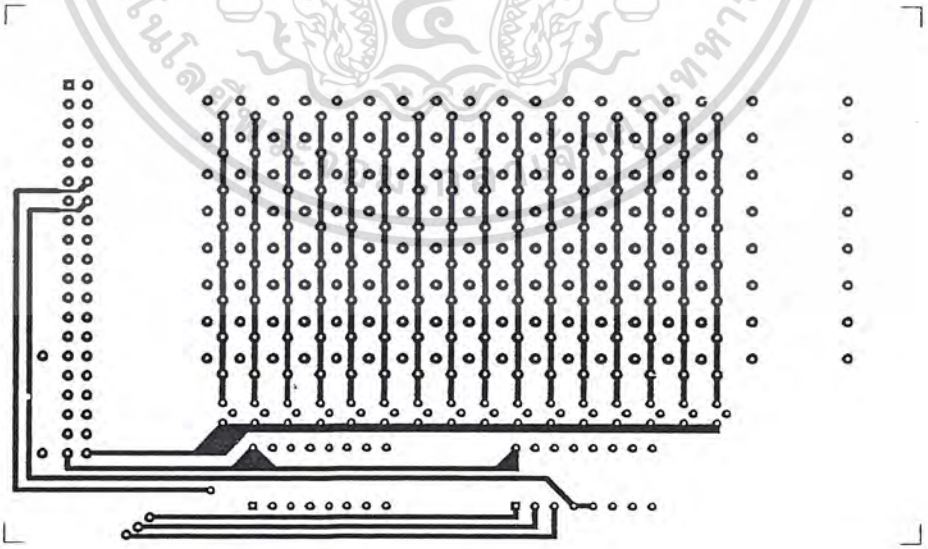
รูปที่ ก.11 แผงอุปกรณ์ต่อพ่วงทั้งหมดในกล่องบรรจุ 10 กล่อง

รูปที่ ก.12 กล่องบรรจุอุปกรณ์ต่อพ่วง ประกอบด้วยบอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) และกล่องอุปกรณ์ต่อพ่วงทั้ง 10 กล่อง

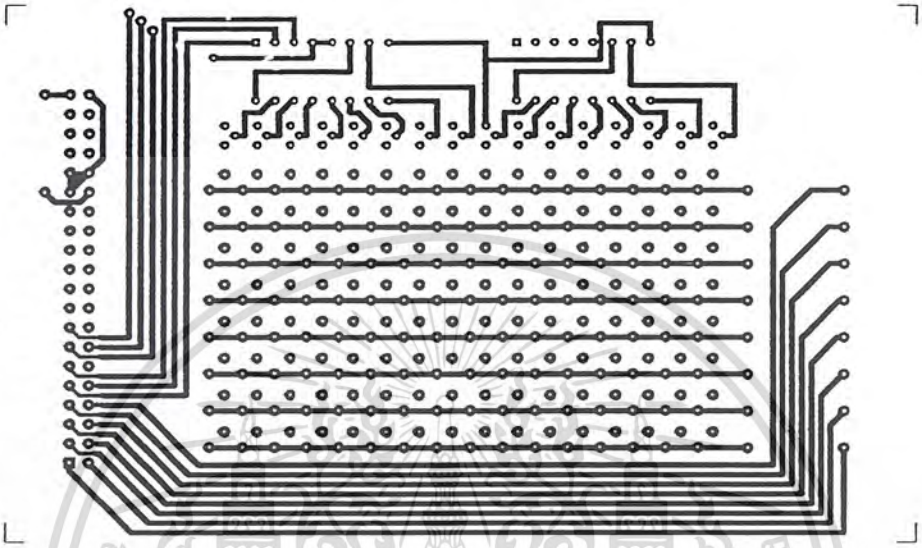
ภาคผนวก ข
แผ่นวงจรพิมพ์



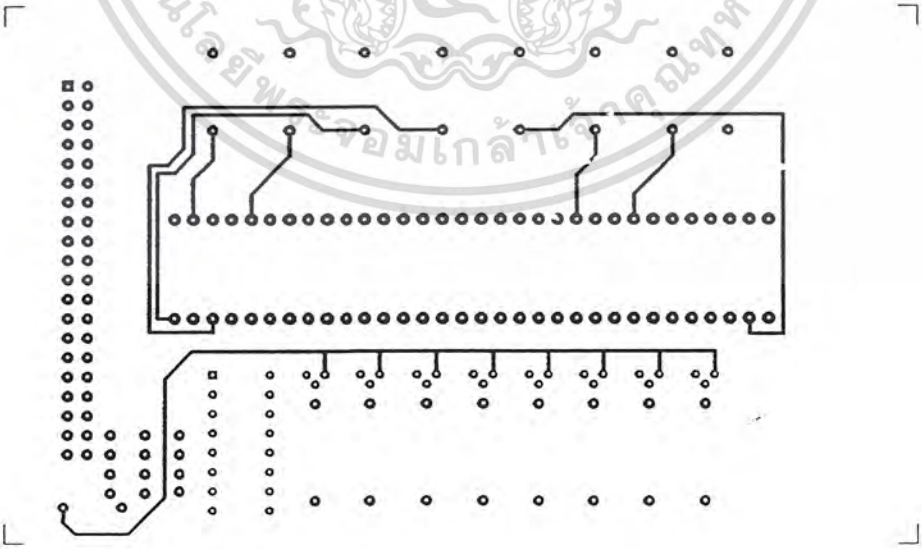
รูปที่ ข.1 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง แอลอีดีลอจิก



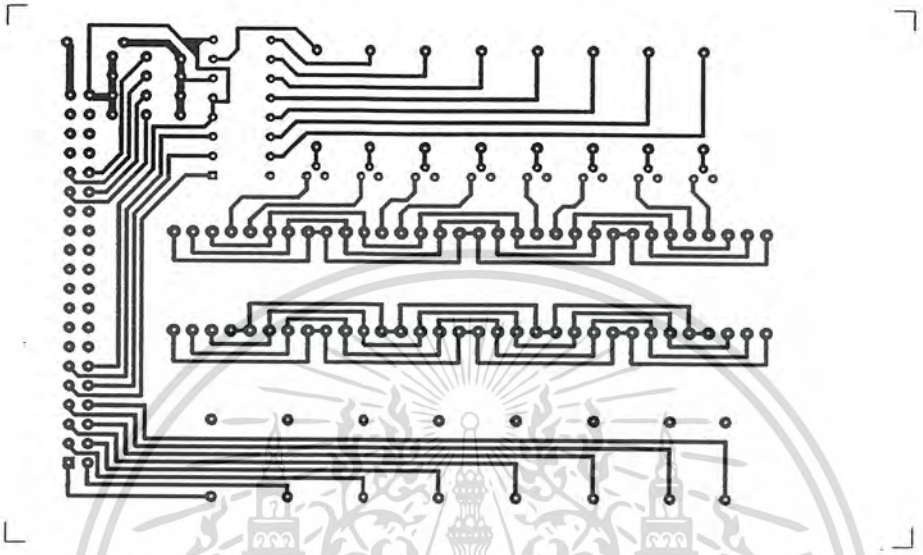
รูปที่ ข.2 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง แอลอีดีดอตเมตริกซ์ (ด้านบน)



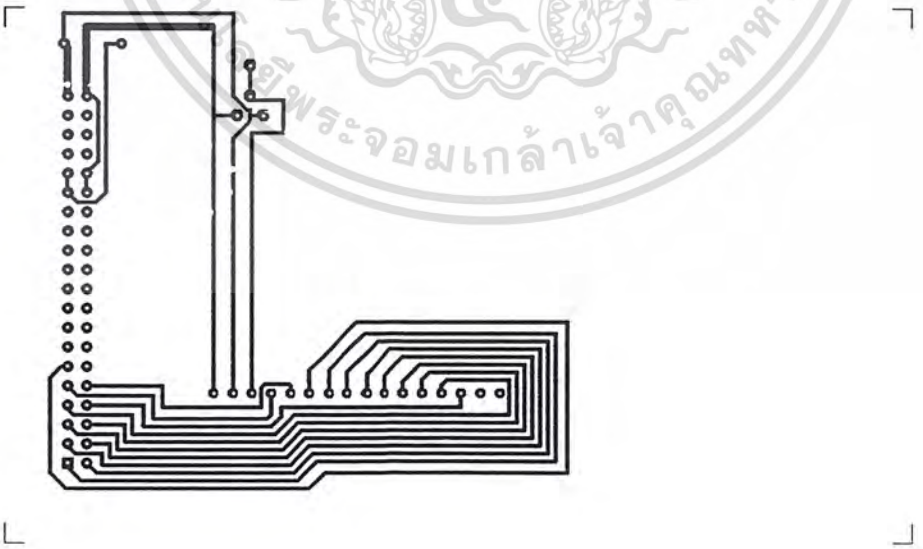
รูปที่ ข.3 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง แอลอีดีดอตเมตริกซ์ (ด้านล่าง)



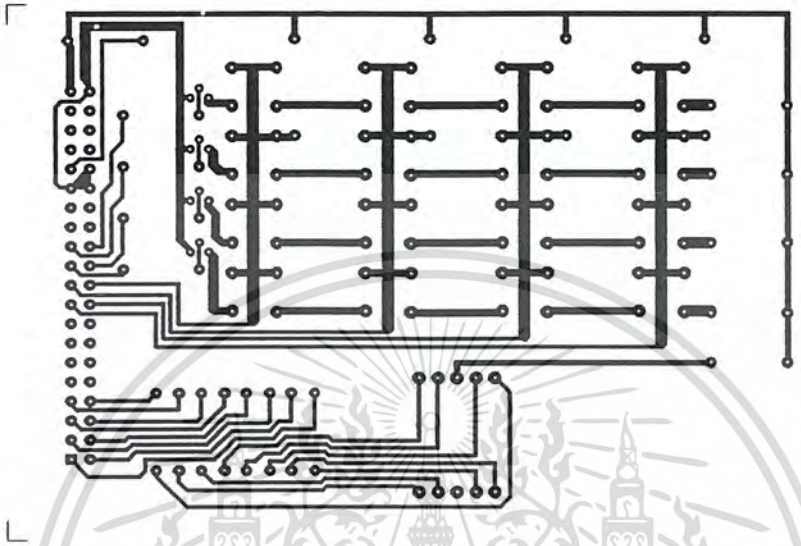
รูปที่ ข.4 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง จอแสดงผลแบบ 7 ส่วน (ด้านบน)



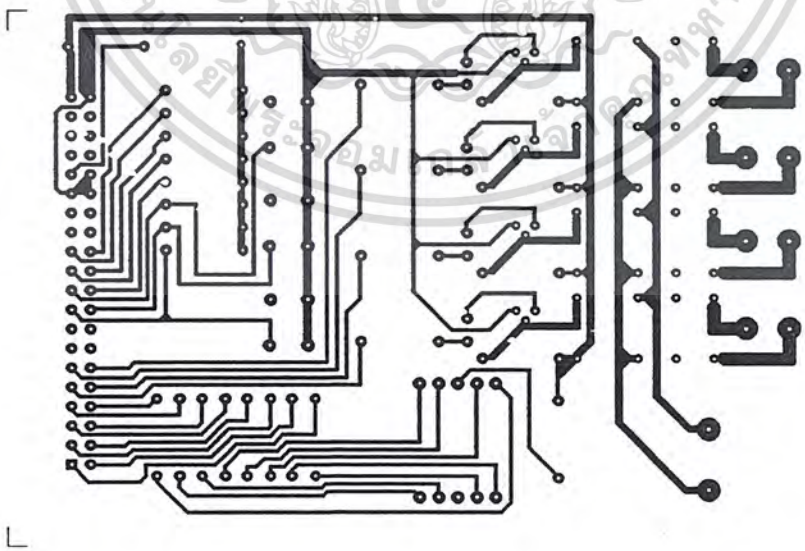
รูปที่ ข.5 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง จอแสดงผลแบบ 7 ส่วน (ด้านล่าง)



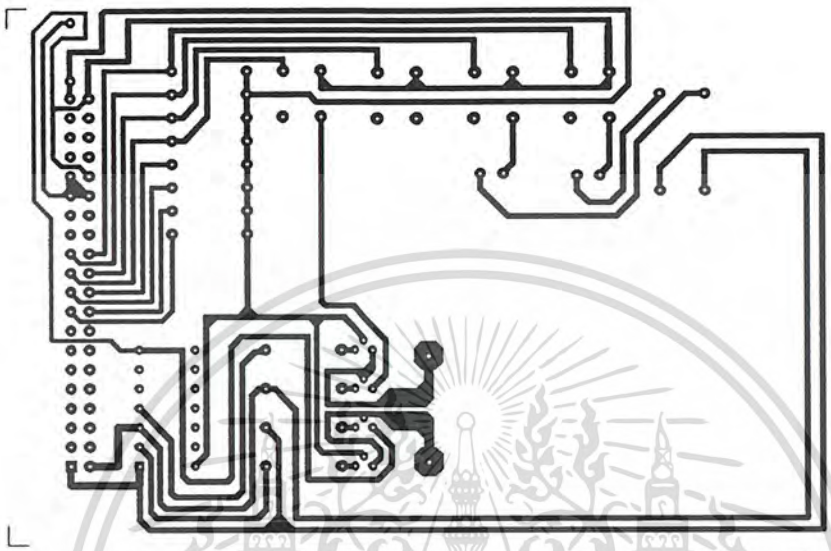
รูปที่ ข.6 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง จอแสดงผลแบบผลึกเหลว



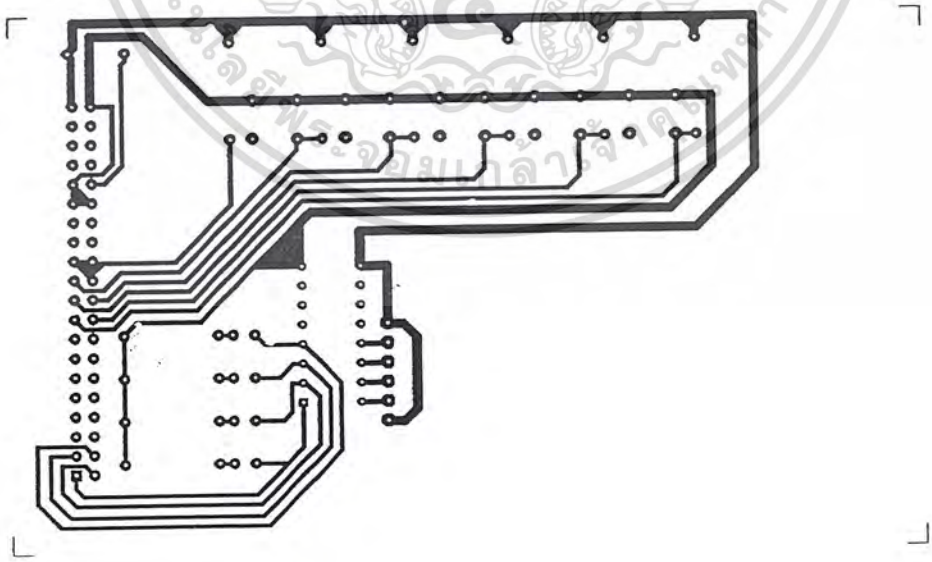
รูปที่ ข.7 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง สวิตช์เมตริกซ์



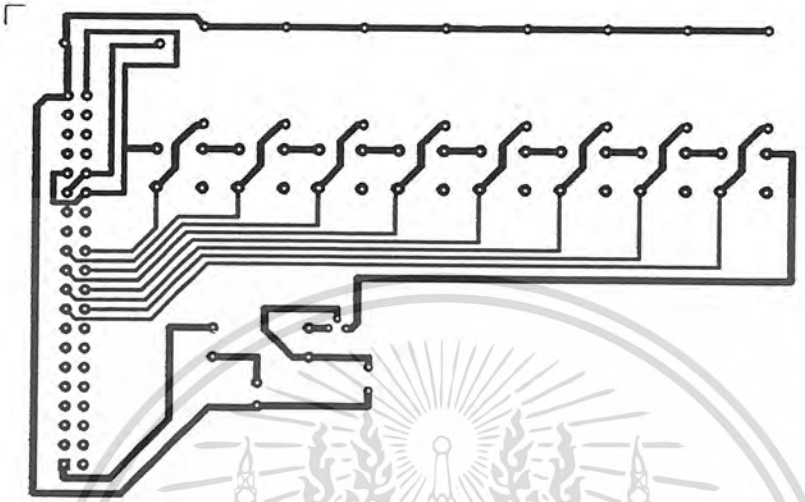
รูปที่ ข.8 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง คอนโทรลสวิตช์



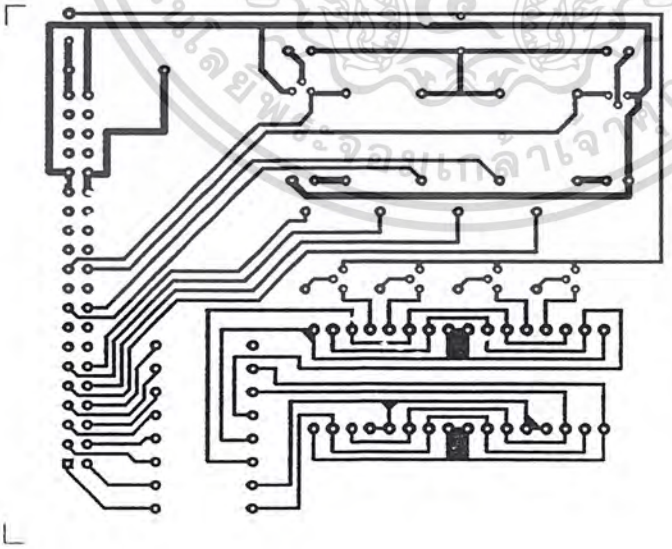
รูปที่ ข.9 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง มอเตอร์ไฟฟ้ากระแสตรง



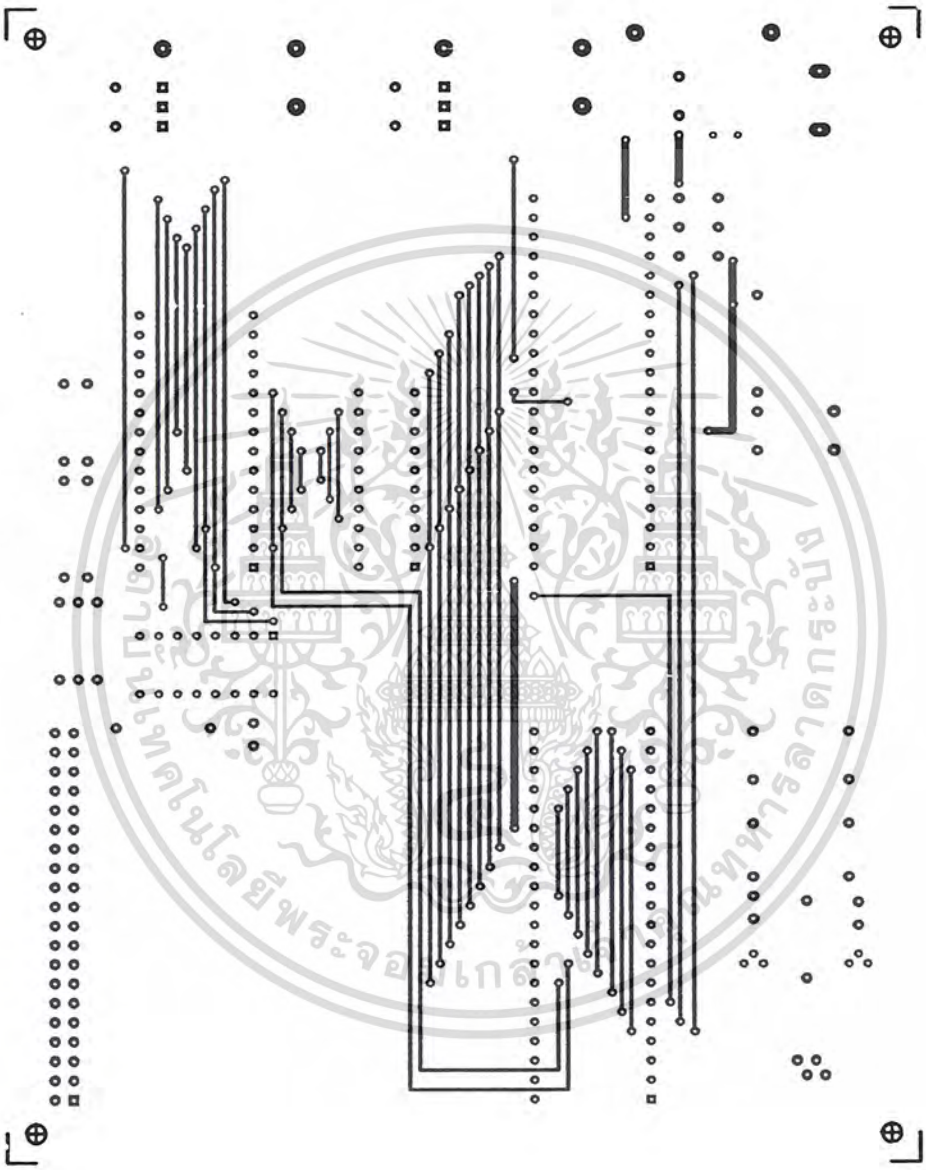
รูปที่ ข.10 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง สเต็ปปิ้งมอเตอร์



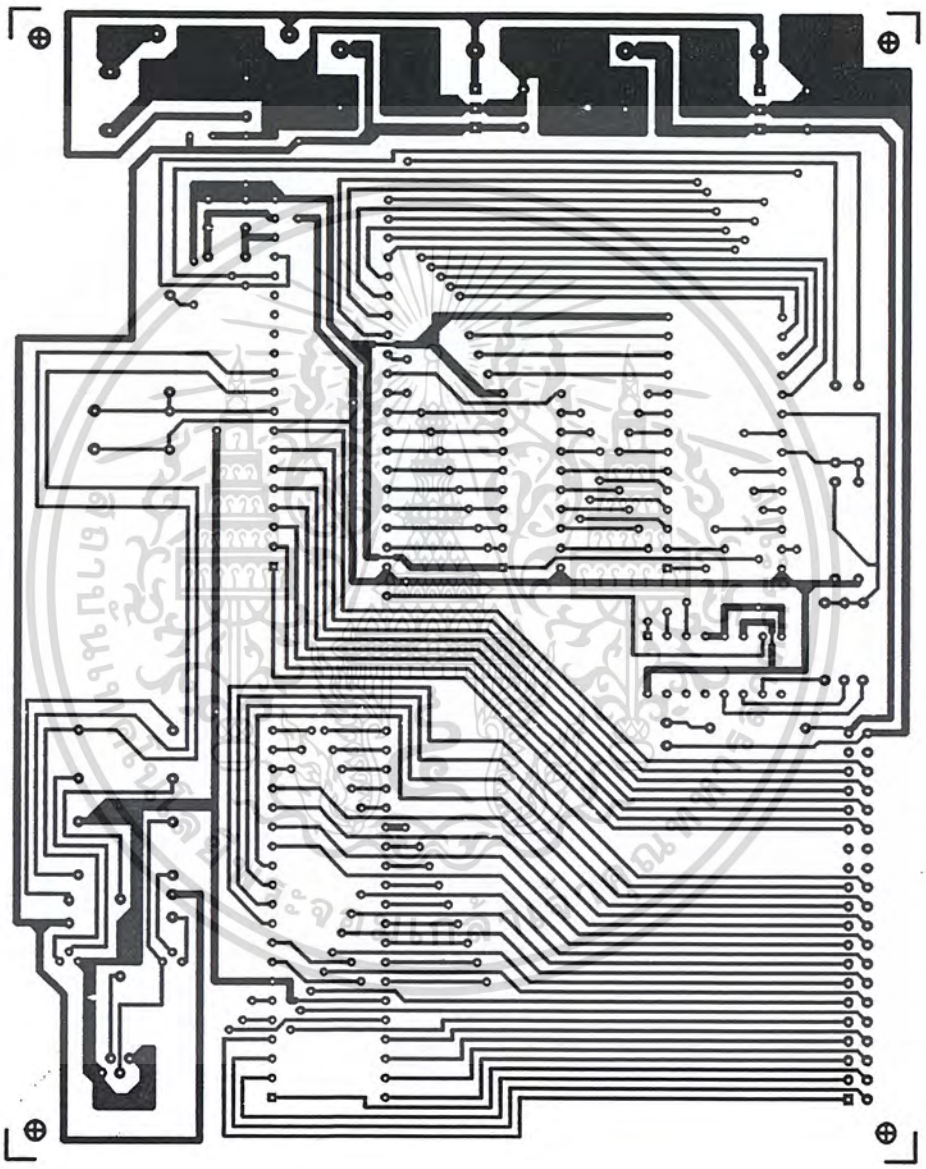
รูปที่ ข.11 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง การกำเนิดเสียง



รูปที่ ข.12 แผ่นวงจรพิมพ์สำหรับทดลองเรื่อง เซ็นเซอร์ และเคาน์เตอร์



รูปที่ ข.13 แผ่นวงจรพิมพ์บอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) ด้านบน



รูปที่ ข.14 แผ่นวงจรพิมพ์บอร์ดไมโครคอนโทรลเลอร์ (TEPSA-I) ด้านล่าง

ภาคผนวก ค
คู่มือการใช้โปรแกรม

MCS-51 Simulator Program

By

TEPSA

คู่มือการใช้งานโปรแกรมจำลองการทำงานของ MCS-51 และอุปกรณ์ต่อพ่วง

1 กล่าวนำ

โปรแกรมจำลองการทำงานของ MCS-51 และอุปกรณ์ต่อพ่วง มีความสามารถในการจำลองการประมวลผลคำสั่ง และความเปลี่ยนแปลงในหน่วยความจำภายใน MCS-51 ได้ ทั้งยังรวมหน้าจอ Editor มาไว้รวมกัน เพื่อความสะดวกในการใช้งาน และหน้าจอ Connect สำหรับการติดต่อกับบอร์ดไมโครคอนโทรลเลอร์

1.1 ความรู้พื้นฐานเกี่ยวกับ MCS-51 Simulate Program

1.1.1 การติดตั้ง MCS-51 Simulate Program

1) อุปกรณ์ทางฮาร์ดแวร์ และซอฟต์แวร์ที่ต้องการ

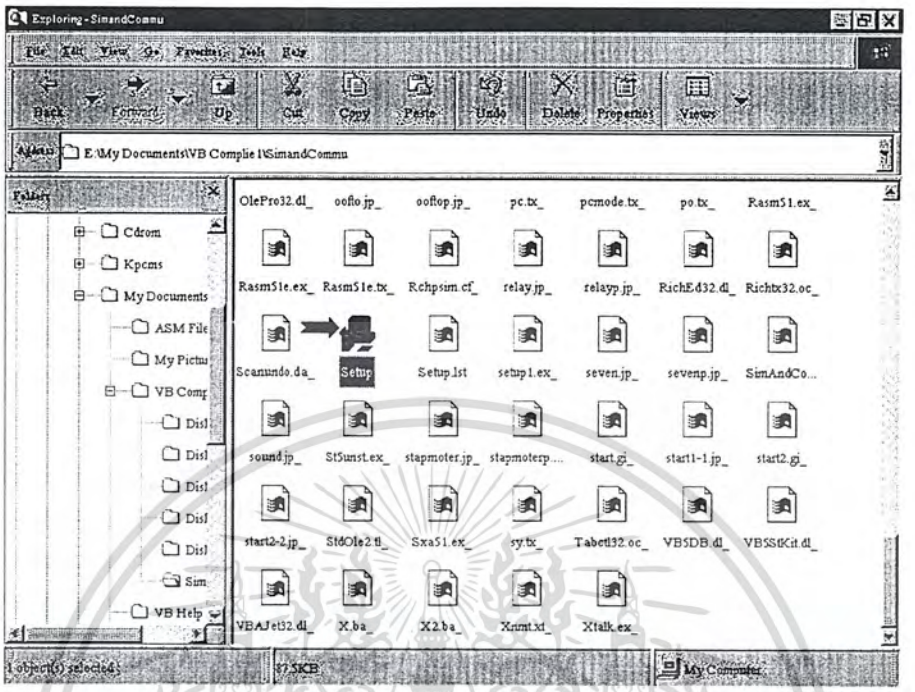
1.1) ระบบปฏิบัติการ Windows 3.11 ขึ้นไป (ภาษาไทย)

1.2) ฮาร์ดดิสก์มีพื้นที่ว่างประมาณ 13 เมกะไบต์ ขึ้นไป

1.3) ใช้ได้ดีในหน้าจอโหมด 800x600 dpi

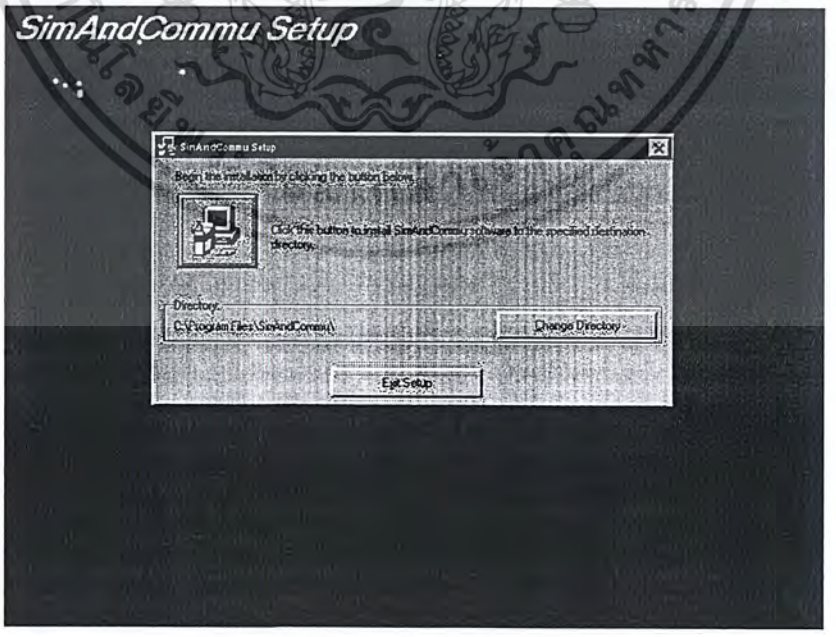
2) ขั้นตอนการติดตั้ง

2.1) เลือกไฟล์ Setup.exe จากโฟลเดอร์ที่มีไฟล์สำหรับการติดตั้งอยู่ ดังรูปที่ ค.1



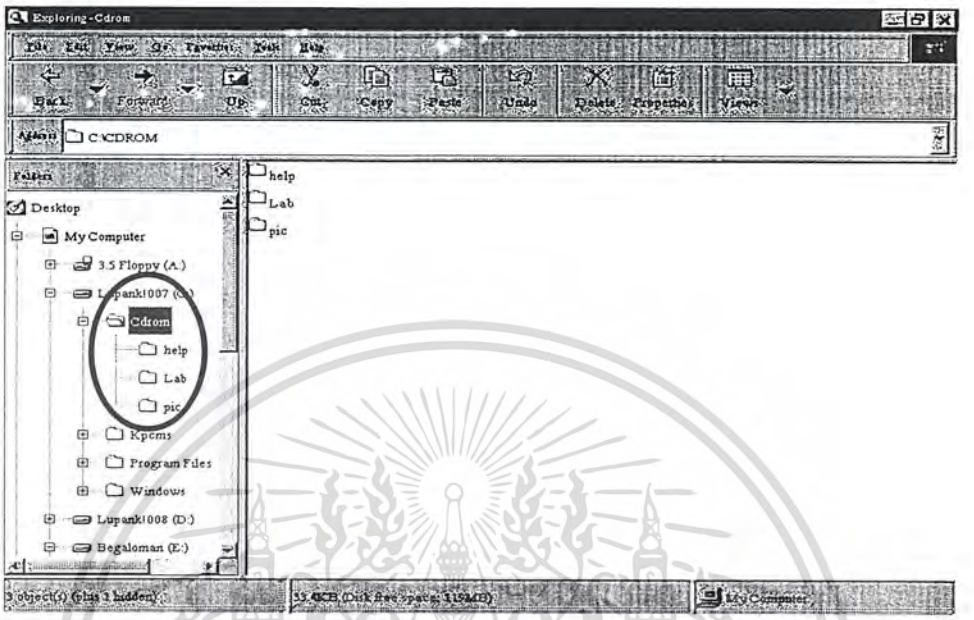
รูปที่ ค.1 ขณะเปิดโฟลเดอร์

2.2) กำหนดตำแหน่งของโฟลเดอร์ที่ต้องการ Install ดังรูปที่ ค.2



รูปที่ ค.2 ขณะ Setup (เลือกโฟลเดอร์)

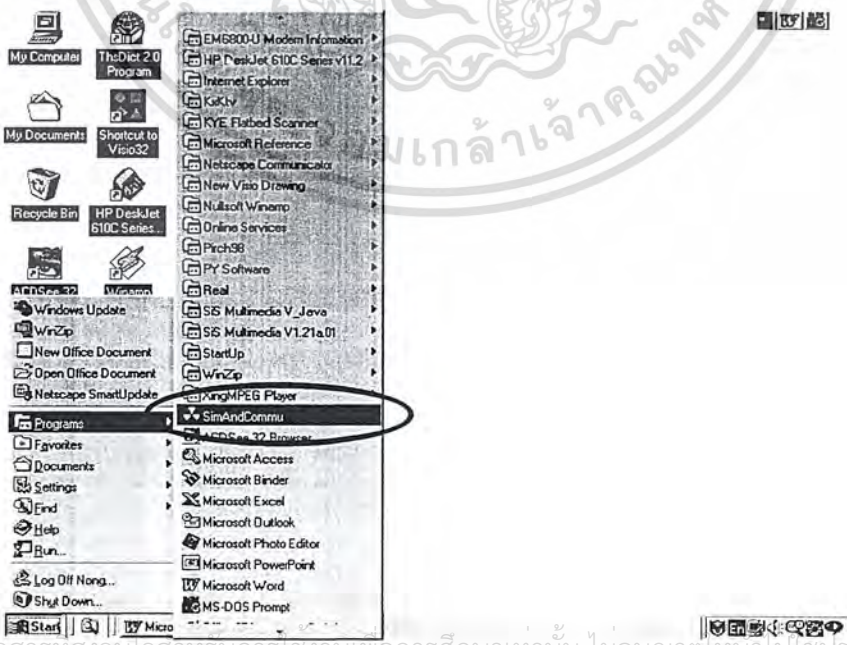
2.3) แนะนำตำแหน่งที่อยู่ของโปรแกรม และแฟ้มข้อมูลที่เกี่ยวข้อง แสดงดังรูปที่ ค1.3



รูปที่ ค.3 โฟลเดอร์ของโปรแกรม และ CDROM

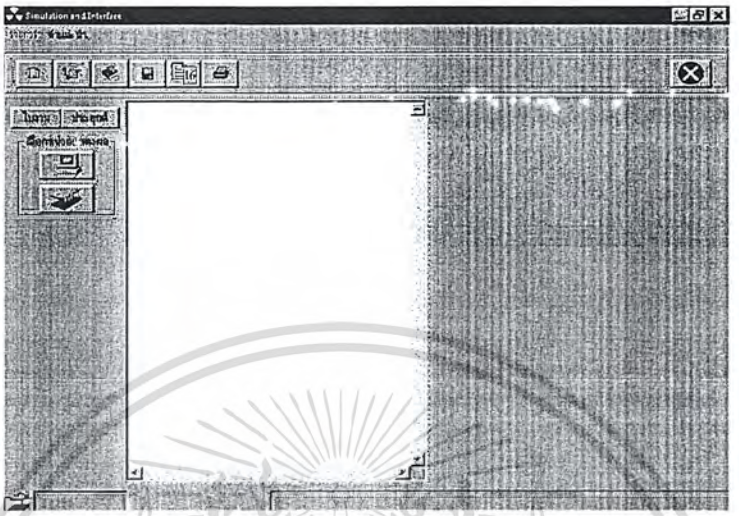
3) เริ่มต้นใช้ MCS-51 Simulate Program

3.1) คลิกเปิดโปรแกรม ดังรูปที่ ค.4



รูปที่ ค.4 การเปิดโปรแกรมจำลองการทำงาน MCS-51 และอุปกรณ์ต่อพ่วง

3.2) เมื่อทำการคลิก จะปรากฏหน้าจอหลัก และ Massage Box ดังรูปที่ ค.5



รูปที่ ค.5 หน้าจอหลัก Main

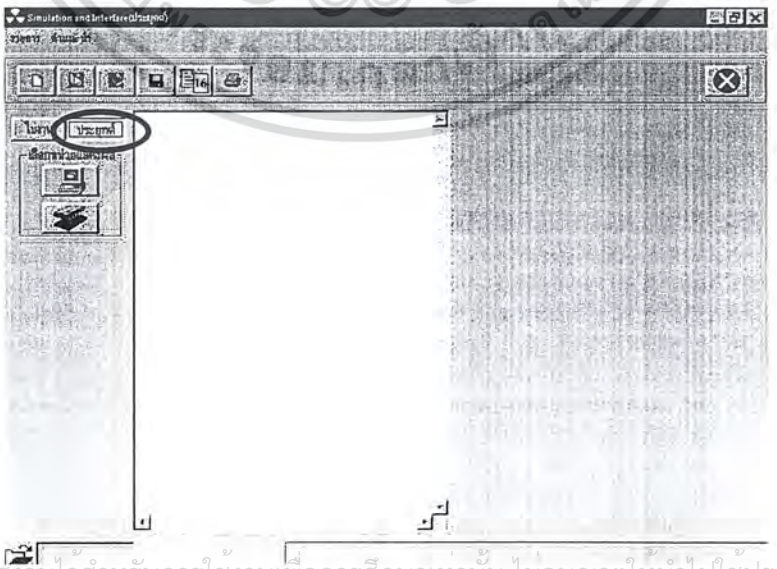
1.1.2 หน้าจอต่างๆ ของโปรแกรมจำลองการทำงานของ MCS-51 และอุปกรณ์ต่อพ่วง

1) หน้าจอหลัก

ภาพหน้าจอหลักแสดงดังรูปที่ ค.5

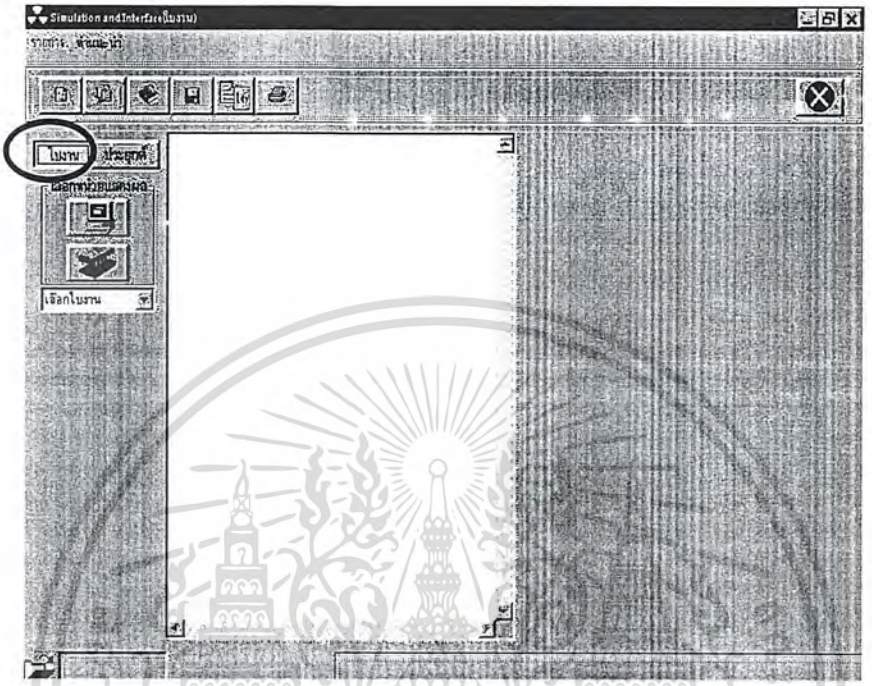
2) หน้าที่ของหน้าจอหลัก

2.1)ใช้เพื่อเป็นหน้าจอ Editor เมื่อคลิกที่ปุ่มประยุกต์ใช้งาน แสดงดังรูปที่ ค.6



รูปที่ ค.6 หน้าจอหลักเมื่อคลิกที่ปุ่มประยุกต์

2.2) ใช้เพื่อศึกษา และทำความเข้าใจใบงานที่มีให้ เมื่อคลิกที่ปุ่มใบงาน ดังรูปที่ ค.7

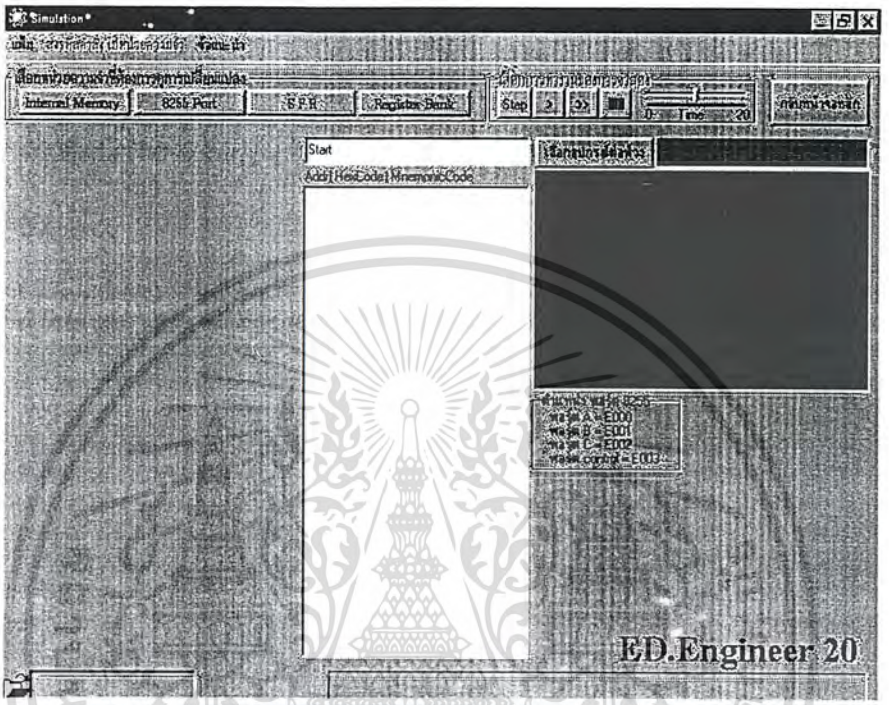


รูปที่ ค.7 หน้าจอหลักเมื่อคลิกที่ปุ่มใบงาน

2.3) ติดต่อกับหน้าจอ Simulate และ Connect โดยคลิกเลือกจากกรอบเลือกหน่วยแสดงผล แสดงดังรูปที่ ค.8

รูปที่ ค.8 กรอบเลือกหน่วยแสดงผล

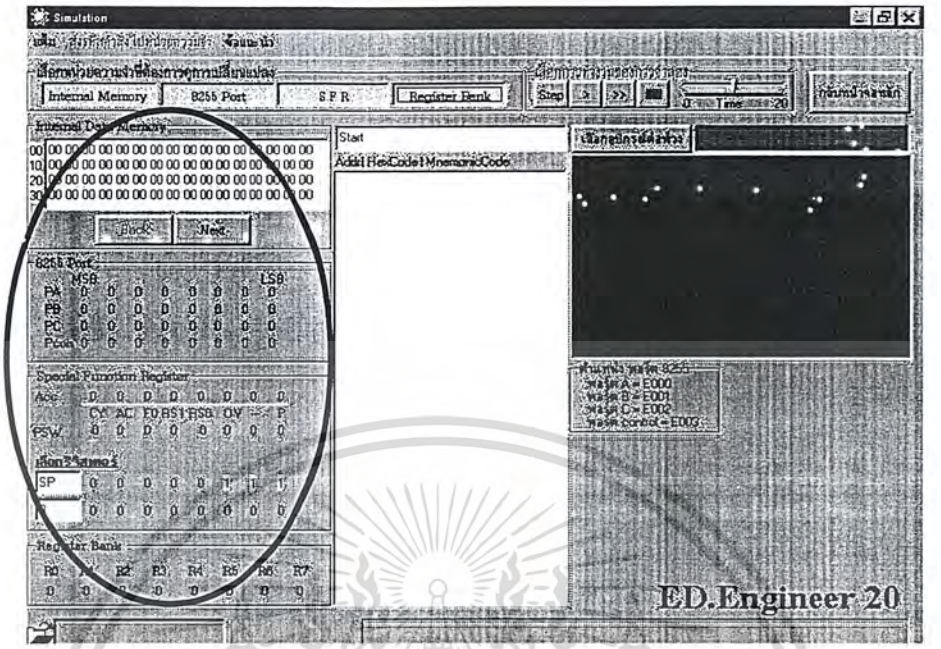
จากรูปที่ ค.8 เมื่อคลิกปุ่มในกรอกเลือกหน่วยแสดงผล (ไอค่อนรูปคอมพิวเตอร์) จะปรากฏหน้าจอ Simulate มีลักษณะจอภาพดังรูปที่ ค.9



รูปที่ ค.9 หน้าจอ Simulate

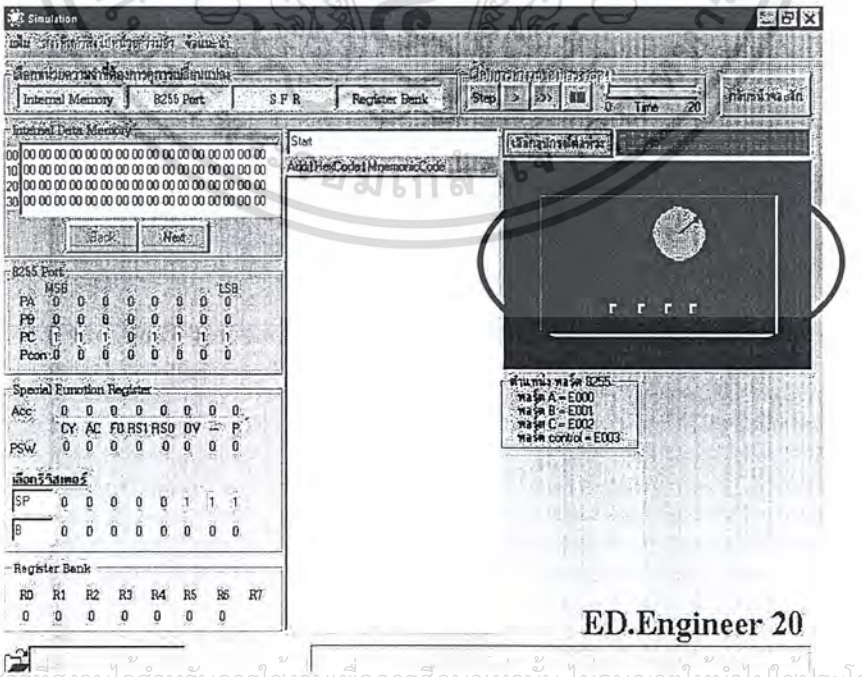
3) หน้าที่ของหน้าจอ simulate

3.1) แสดงความเปลี่ยนแปลงของหน่วยความจำภายใน, รีจิสเตอร์ฟังก์ชันพิเศษ, รีจิสเตอร์แบงค์ และพอร์ต 8255 หน้าจอแสดงดังรูปที่ ค.10



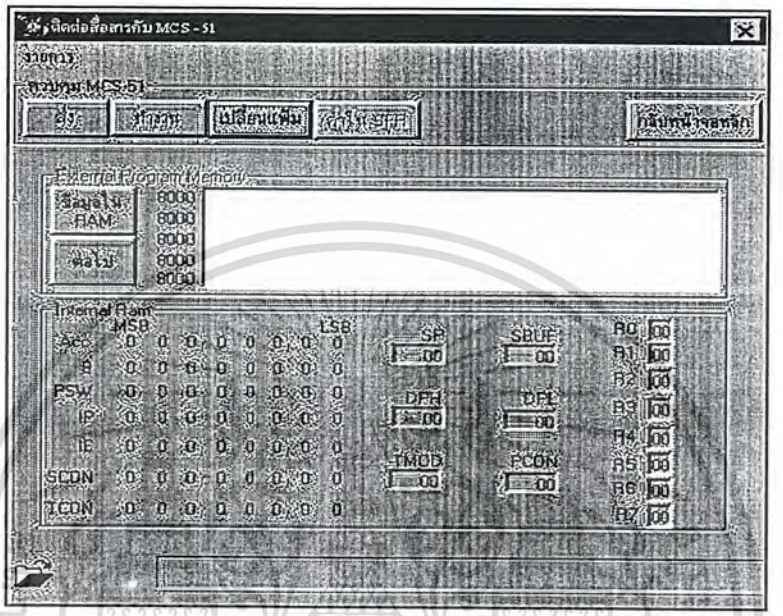
รูปที่ ค.10 หน้าต่างแสดงการเปลี่ยนแปลง ของหน่วยความจำภายใน, รีจิสเตอร์ต่างๆ และพอร์ต 8255

3.2) แสดงความเปลี่ยนแปลง ของอุปกรณ์ต่อพ่วงจำลองที่มีให้ ดังรูปที่ ค.11



รูปที่ ค.11 หน้าต่างอุปกรณ์ต่อพ่วงจำลอง

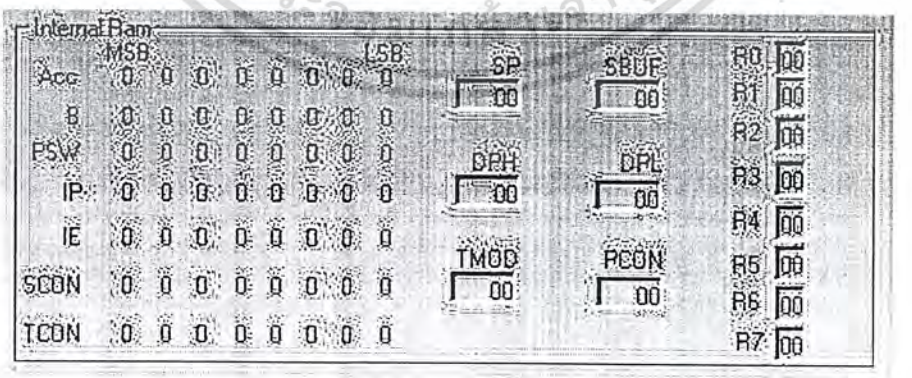
จากรูปที่ ค.11 เมื่อคลิกปุ่มในกรอบเลือกหน่วยแสดงผลไอค่อนรูปไอซี จะปรากฏหน้าจอ Connect มีลักษณะจอภาพดังรูปที่ ค.12



รูปที่ ค.12 หน้าจอ Connect

4) หน้าที่ของหน้าจอ Connect

4.1) แสดงความเปลี่ยนแปลงรีจิสเตอร์ฟังก์ชันพิเศษ และรีจิสเตอร์แบงค์ ดังรูปที่ ค.13



รูปที่ ค.13 หน้าต่างหน่วยความจำ SFR Bank

4.2) ดาวน์โหลดโปรแกรมที่เขียนไปยังไมโครคอนโทรลเลอร์ และสามารถหยุดการทำงานของไมโครคอนโทรลเลอร์ชั่วขณะ เพื่อดูค่าในรีจิสเตอร์ฟังก์ชันพิเศษ และรีจิสเตอร์แบงค์ โดยแถบเครื่องมือควบคุมการทำงานดังรูปที่ ค.14

รูปที่ ค.14 แถบเครื่องมือควบคุมการทำงาน

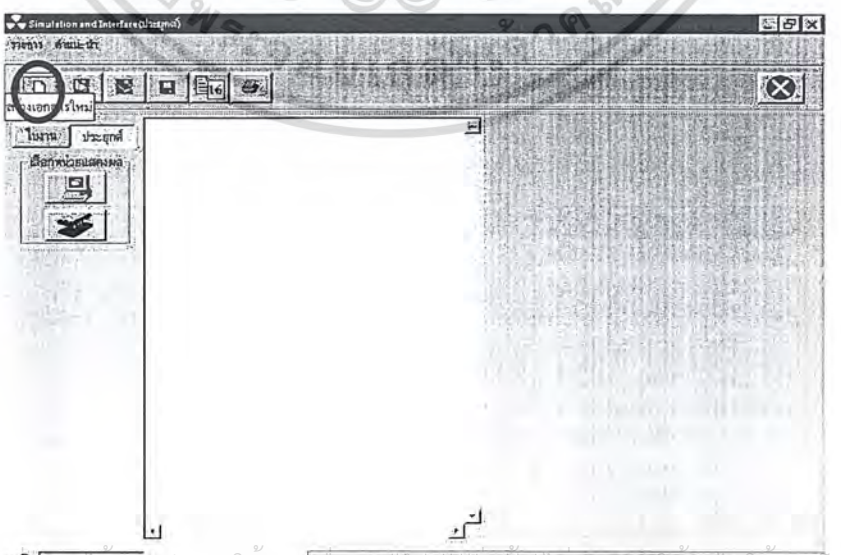
4.3) แสดงความเปลี่ยนแปลงของหน่วยความจำภายนอก (RAM) ดังรูปที่ ค.15

รูปที่ ค.15 หน้าต่างหน่วยความจำภายนอก (RAM)

1.1.3 การเริ่มต้นพิมพ์โปรแกรมใหม่

1) เมื่อเปิดโปรแกรมขึ้นมาให้เลือกโหมดเป็นประยุกต์ใช้งาน

2) คลิกที่ปุ่มทูลบาร์ไอค่อนรูปเอกสารเปล่า หรือเลือกจากเมนูสร้าง ดังรูปที่ ค1.16



รูปที่ ค1.16 หน้าจอหลักประยุกต์ใช้งาน New

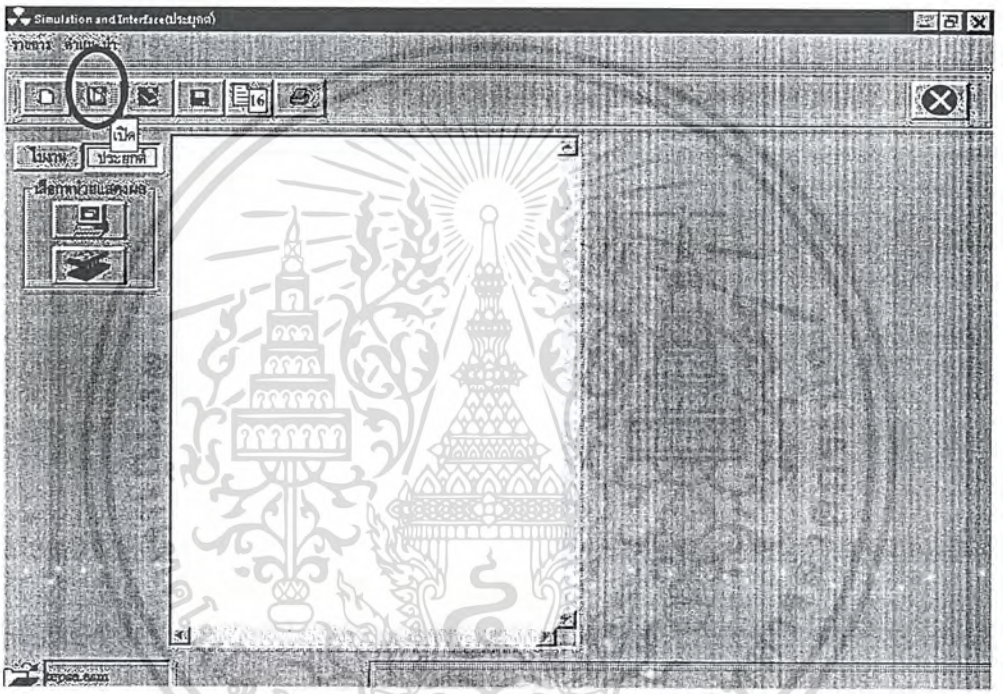
3. เริ่มพิมพ์โปรแกรมที่ออกแบบไว้

หมายเหตุ : การใช้ Tab สามารถทำได้โดยการกด Ctrl + Tab

1.1.4 การเรียกโปรแกรมเดิมมาแก้ไข

1) เมื่อเปิดโปรแกรมขึ้นมาให้เลือกโหมดเป็นประยุกต์ใช้งาน

2) คลิกที่ปุ่มทูลบาร์ ไอค่อนรูปเปิดเอกสาร หรือเลือกจากเมนูเปิด ดังรูปที่ ค1.17

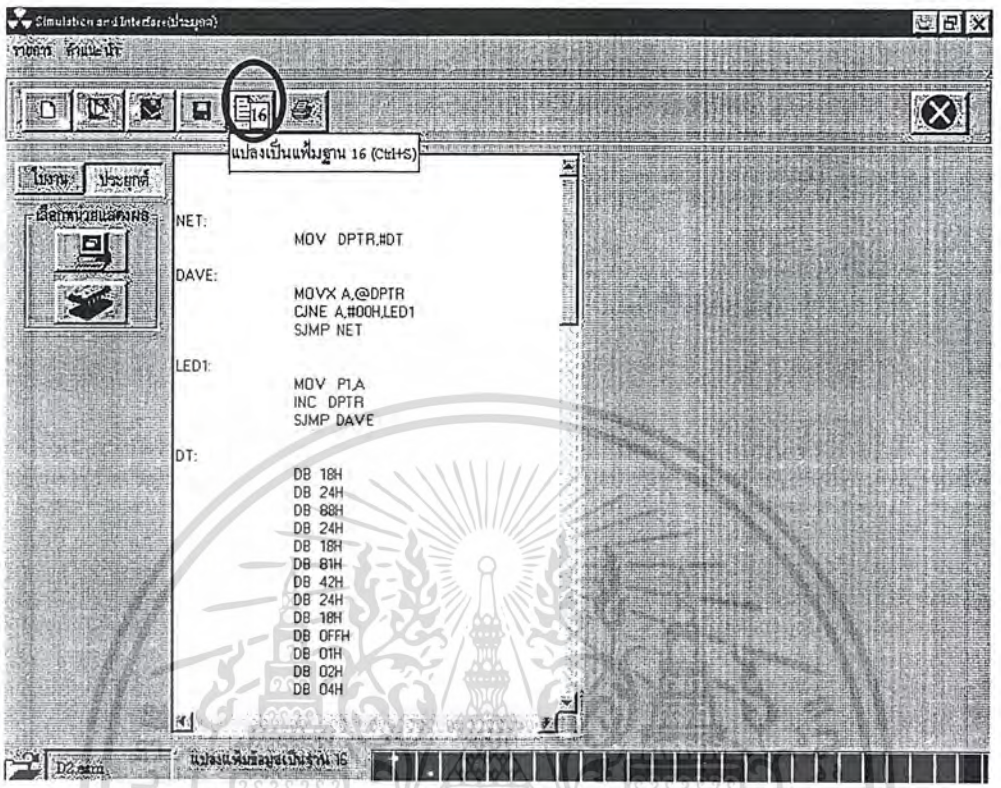


รูปที่ ค1.17 หน้าจอหลักประยุกต์ใช้งาน Open

3) ทำการแก้ไขโปรแกรมเดิมตามที่กำหนดไว้

1.1.5 การแปลงโปรแกรมเป็นแฟ้มฐาน 16

1) เปิดแฟ้มโปรแกรมขึ้นมา คลิกที่ปุ่มไอค่อนรูปเอกสาร 16 แสดงดังรูปที่ ค1.18

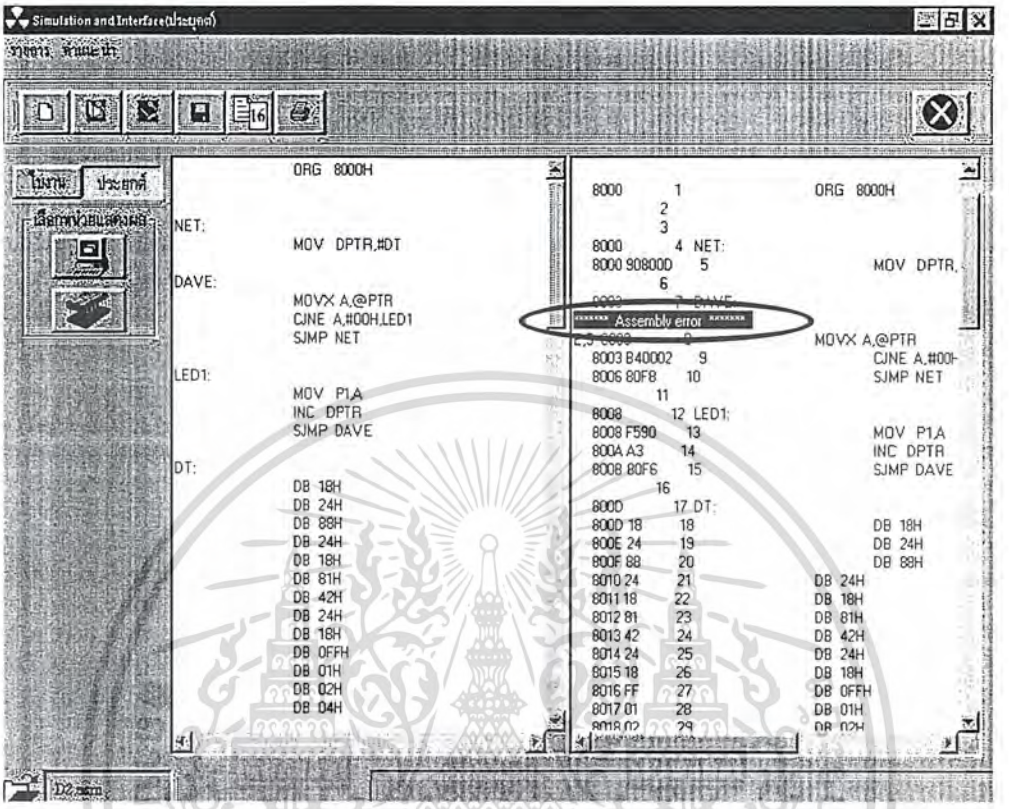


รูปที่ ค1.18 หน้าจอหลัก Compile

หมายเหตุ : ก่อนแปลงให้บันทึกโปรแกรมก่อนเสมอ

1.1.6 เมื่อโปรแกรมที่เขียนมีข้อผิดพลาด

หลังจากที่แปลงโปรแกรมเป็นแฟ้มฐาน 16 แล้วจะมีการตรวจสอบความผิดพลาดที่เกิดจากการเขียนโปรแกรมโดยอัตโนมัติ หากมีข้อผิดพลาด จะแสดงกรอบข้อความขึ้นทางด้านขวามือของกรอบข้อความที่ใช้เขียนโปรแกรม พร้อมแถบแสงแสดงจุดผิดพลาด ดังรูป ค1.19

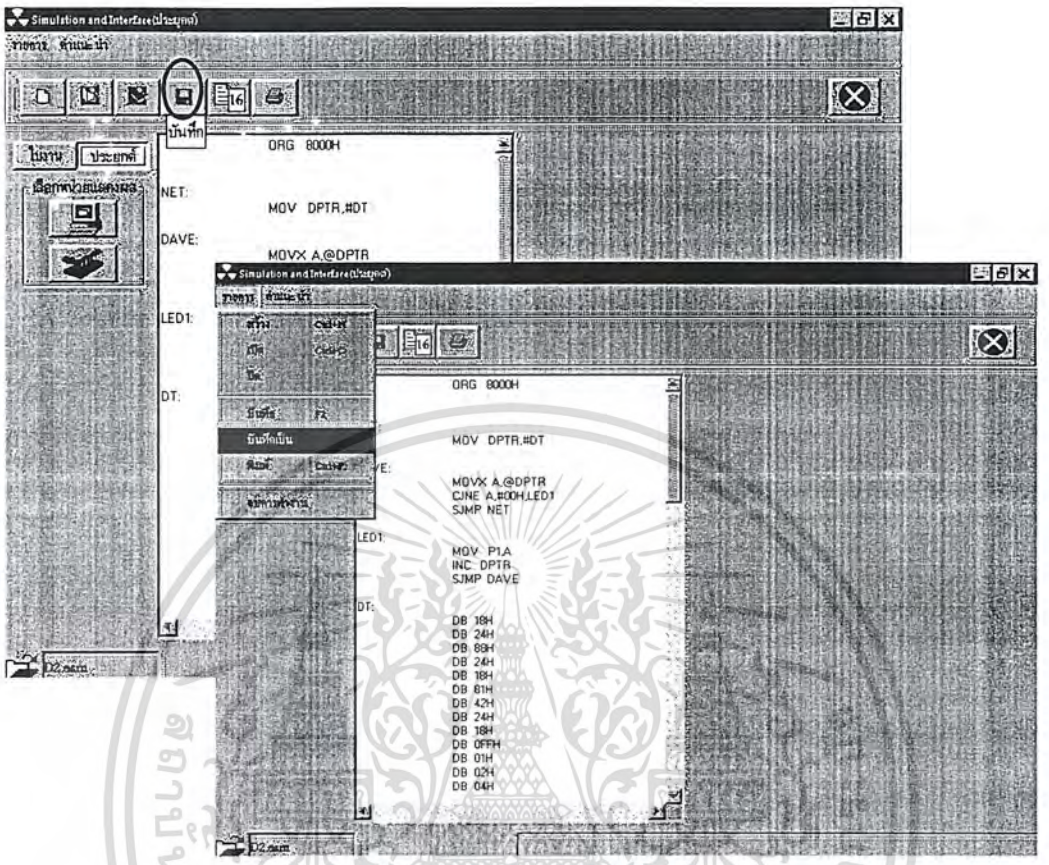


รูปที่ ค1.19 หน้าจอหลักเมื่อมีการแจ้ง Error

จากรูปที่ ค1.19 วิธีการใช้กรอบข้อความแจ้งจุดผิดพลาดคือการแจ้งจุดผิดพลาดจะแจ้งไว้บนบรรทัดที่เกิดจุดผิดพลาด หากต้องการแก้ไขให้แก้ไขได้ที่กรอบข้อความทางด้านซ้ายมือ

1.1.7บันทึกโปรแกรม

คลิกที่ปุ่มทูลบาร์ ไอค่อนรูปแผ่นดิสก์ หรือเลือกที่เมนูบันทึก หรือเมนูบันทึกแฟ้มเป็นดังรูปที่ ค.20 หากเป็นแฟ้มที่สร้างขึ้นใหม่ หรือแฟ้มที่เปิดขึ้นมาแก้ไขแล้วเปลี่ยนชื่อแฟ้มให้ใส่นามสกุล .asm ให้กับแฟ้มนั้นด้วย



รูปที่ ค.20 หน้าจอหลักเมื่อทำการบันทึกโปรแกรม

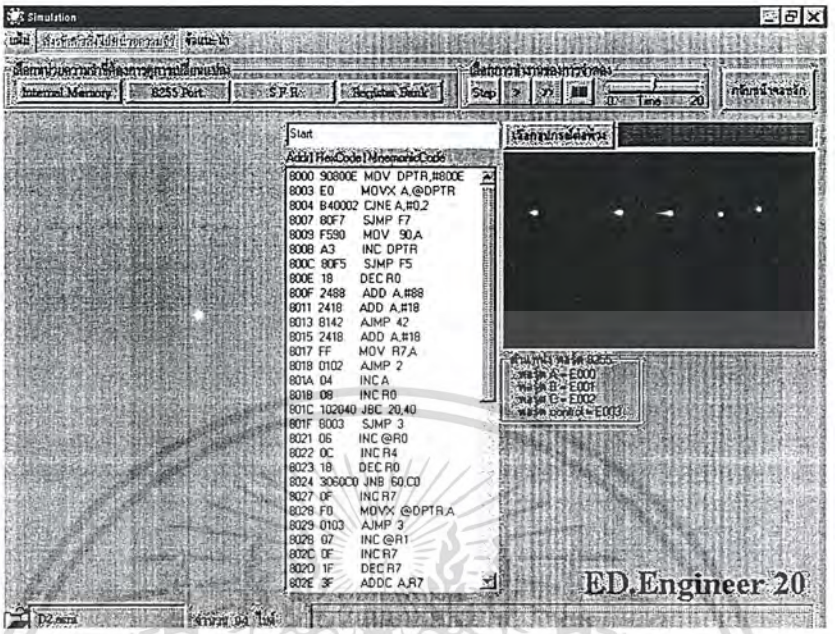
1.2 การใช้งานหน้าจอ Simulate

1) เปิดแฟ้ม

การทำงานของหน้าจอ Simulate จะต่อเนื่องมาจากหน้าจอหลักกล่าวคือ หากหน้าจอหลักเปิดโปรแกรมใดอยู่ จะทำให้หน้าจอ Simulate จะโหลดแฟ้มฐาน 16 ของโปรแกรมที่เปิดอยู่ ให้โดยอัตโนมัติ แต่ถ้าไม่ได้เปิดโปรแกรมในหน้าจอหลักต้องเปิดแฟ้มฐาน 16 ที่ต้องการเอง

2) การโหลดโปรแกรมไปยังหน่วยความจำจำลอง

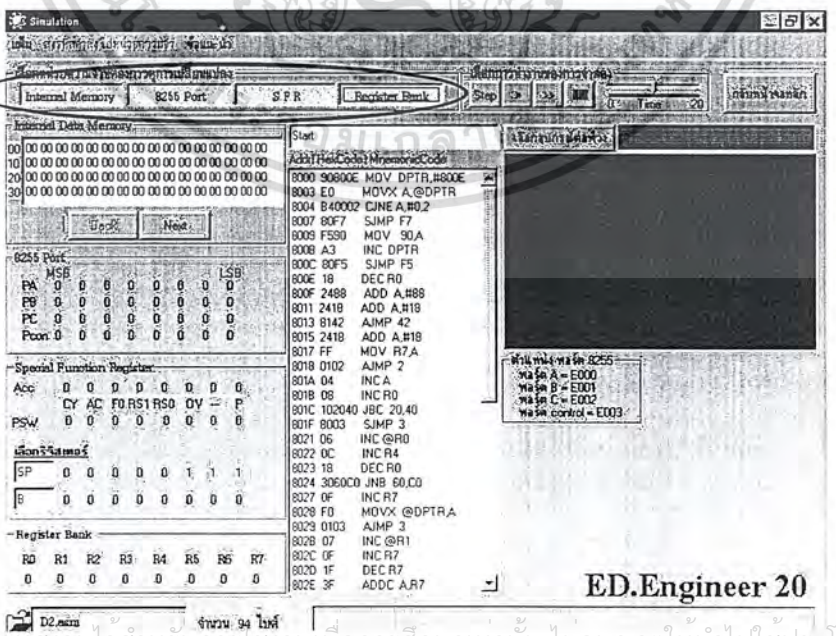
การใช้หน้าจอ Simulate จะจำลองตั้งแต่การโหลดโปรแกรมไปยังหน่วยความจำจนถึงการสั่งให้โปรแกรมทำงาน ดังนั้น เมื่อเปิดแฟ้มที่ต้องการแล้ว ต้องโหลดโปรแกรมไปยังหน่วยความจำจำลองด้วย จึงจะสามารถส่งรันดูผลการทำงานได้ หน้าจอแสดงดังรูปที่ ค.21



รูปที่ ค.21 หน้าจอ Simulateขณะดาวน์โหลดโปรแกรม

3) เรียกดูหน่วยความจำ

เลือกดูการเปลี่ยนแปลงของหน่วยความจำต่างๆ ได้โดยการคลิกที่ปุ่มของหน่วยความจำนั้น หรือให้ซ่อนทำได้โดยการคลิกปุ่มเดิมอีกครั้ง ดังรูปที่ ค.22



รูปที่ ค.22 การเลือกดูหน่วยความจำ Simulate

ในส่วนของรีจิสเตอร์ฟังก์ชันพิเศษ สามารถแสดงได้ทั้งหมด 4 ตัว โดย 2 ตัวแรกจะกำหนดให้ และอีก 2 ตัว ให้ผู้ใช้เป็นผู้เลือก ซึ่งทำได้ 2 วิธีคือ พิมพ์ชื่อรีจิสเตอร์ (ตัวพิมพ์ใหญ่) หรือกดแป้นพิมพ์ลูกศรขึ้นหรือลง (ใช้เมาส์คลิกไว้ที่ช่องก่อน)

4) ควบคุมการ Simulate

4.1) ปุ่ม Step เพื่อเลือกรูปแบบการทำงานตามคำสั่ง โดยเลือกได้ทั้งแบบดูการทำงานทีละคำสั่ง และดูผลการทำงานแบบทั้งโปรแกรม

4.2) ปุ่ม > จะใช้งานได้เมื่อปุ่ม Step ยังไม่ถูกกด เป็นการดูผลการทำงานของโปรแกรม

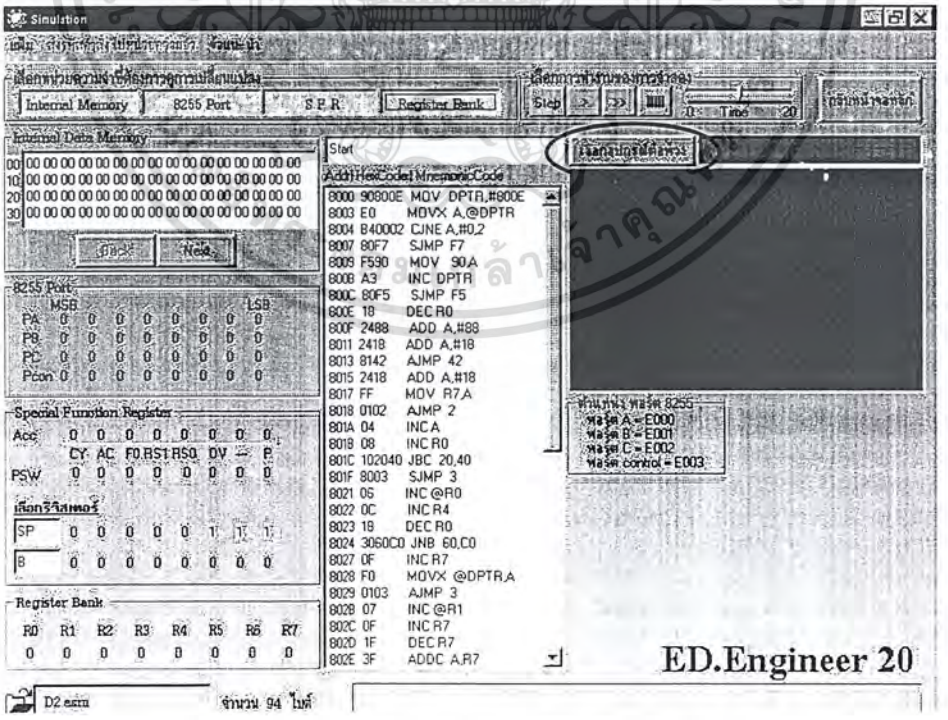
4.3) ปุ่ม >> จะใช้งานได้เมื่อ ปุ่ม Step ถูกกดเป็นการดูผลการทำงานแบบทีละคำสั่ง

4.4) ปุ่ม >| จะใช้งานได้เมื่อปุ่ม Step ยังไม่ถูกกดซึ่งเป็นปุ่มเดียวกันกับปุ่ม > จะปรากฏเมื่อกดปุ่ม > เพื่อดูการทำงานของโปรแกรม ใช้หยุดการทำงานของโปรแกรมชั่วขณะ

4.5) ปุ่ม ■ ใช้สำหรับการให้โปรแกรมที่ดูผลอยู่หยุดทำงาน และรีเซ็ตค่าในหน่วยความจำภายในทั้งหมด ให้อยู่ในสภาวะเริ่มต้น

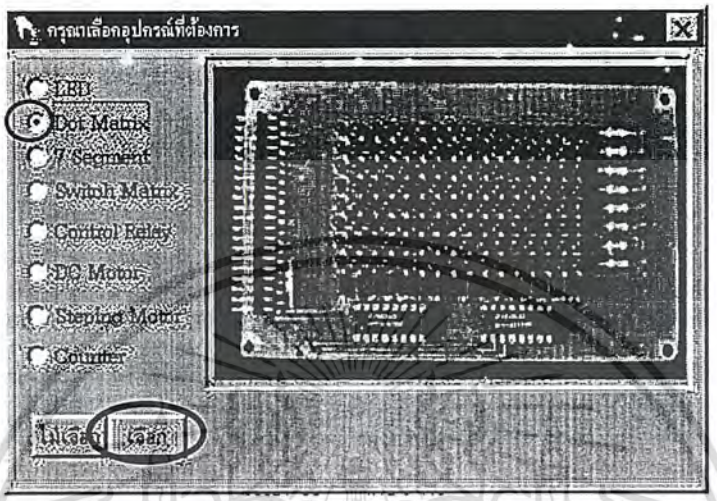
5) อุปกรณ์จำลอง

5.1) สามารถเลือกดูผลจำลองอุปกรณ์จากการคลิกปุ่มเลือกอุปกรณ์ต่อพ่วง ดังรูปที่ ค.23



รูปที่ ค.23 หน้าจอ Simulate เลือกปุ่มอุปกรณ์ต่อพ่วง

5.2) เมื่อปรากฏหน้าจอ เลือกอุปกรณ์ต่อพ่วงแล้ว สามารถเลือกอุปกรณ์ได้โดยการคลิกที่ช่องว่างหน้า ชื่ออุปกรณ์ต่อพ่วงแล้วกดปุ่มเลือก ดังรูปที่ ค.24



รูปที่ ค.24 หน้าจอ Simulate เลือกอุปกรณ์คลิกเลือก

5.3) หากไม่ต้องการดูผลการทำงานของอุปกรณ์ใด ให้คลิกที่ปุ่มไม่เลือก

หมายเหตุ : เมื่อเลือกอุปกรณ์ขึ้นมาแล้ว ต้องการที่จะไม่ดูผลกับอุปกรณ์ต่อพ่วง ให้คลิกที่ปุ่มเลือกอุปกรณ์ต่อพ่วงแล้วคลิกที่ปุ่มไม่เลือก

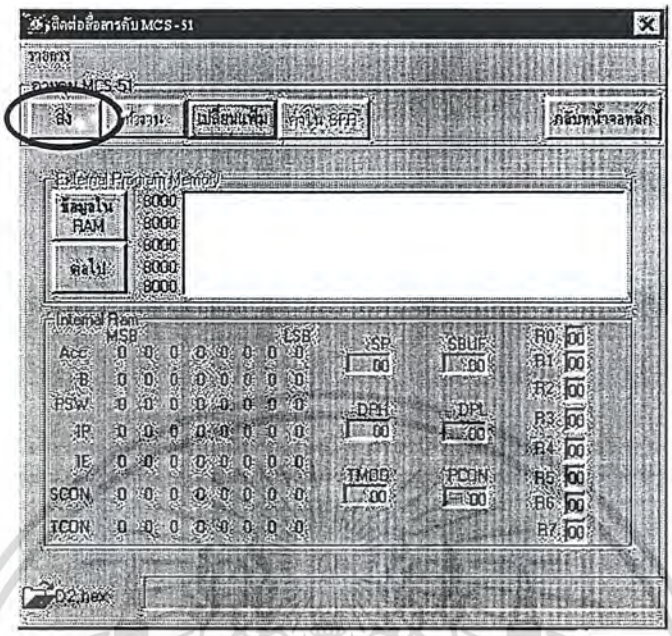
1.3 การใช้งานหน้าจอ Connect

1) เปิดแฟ้ม

การทำงานหน้าจอ Connect จะต่อเนื่องมาจากหน้าจอหลัก กล่าวคือ หากหน้าจอหลักเปิดโปรแกรมใดอยู่ จะทำให้หน้าจอ Connect โหลดแฟ้มฐาน 16 ของโปรแกรมที่เปิดอยู่ให้โดยอัตโนมัติ แต่ถ้าไม่ได้เปิดโปรแกรมในหน้าจอหลัก จะต้องเปิดแฟ้มฐาน 16 ที่ต้องการเอง

2) โหลดโปรแกรมไปยังบอร์ดไมโครคอนโทรลเลอร์

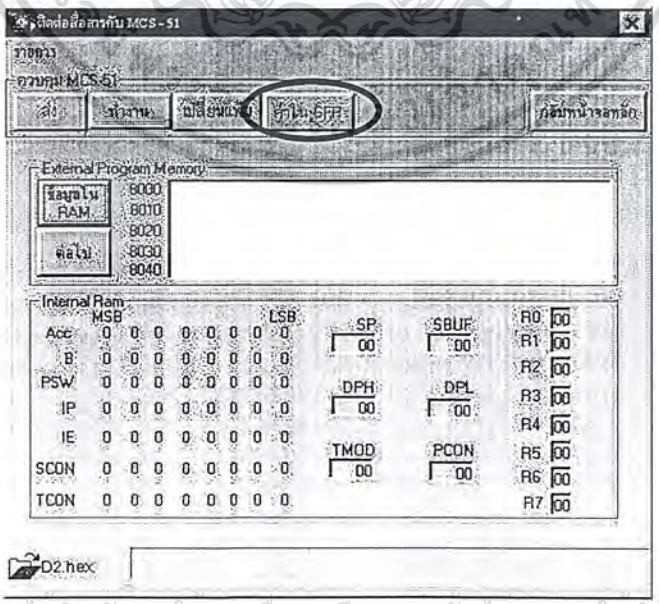
คลิกที่ปุ่มส่ง เพื่อทำการโหลดโปรแกรมไปยังบอร์ดไมโครคอนโทรลเลอร์ ดังรูปที่ ค.25



รูปที่ ค.25 คลิกที่ปุ่มส่งเพื่อโหลดโปรแกรมไปยังบอร์ดไมโครคอนโทรลเลอร์

3) การดูค่าภายในรีจิสเตอร์ของไมโครคอนโทรลเลอร์

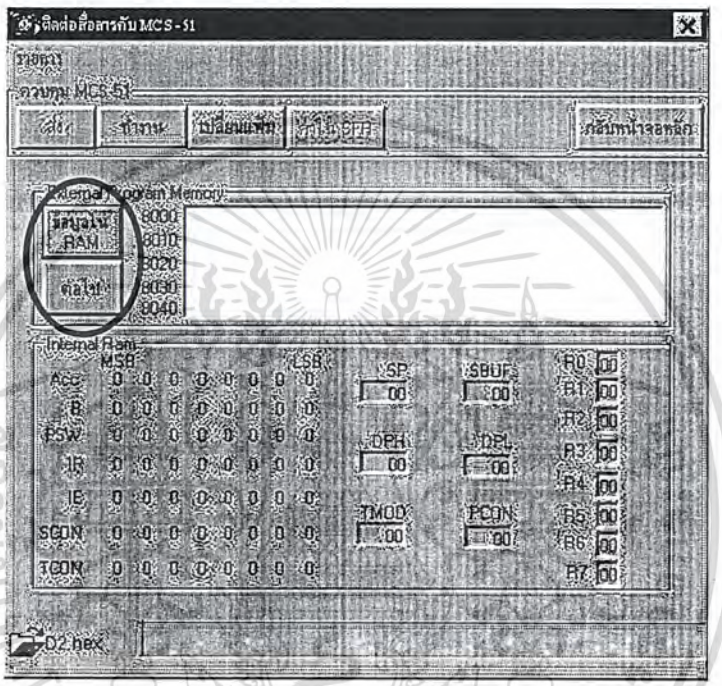
สามารถคลิกได้ต่อเมื่อ การทำงานอยู่ในสภาวะหยุดชั่วขณะ เพื่อดูค่าในรีจิสเตอร์ภายในไมโครคอนโทรลเลอร์ โดยคลิกที่ปุ่มค่าใน SFR ดังรูปที่ ค.26



รูปที่ ค.26 การดูค่าภายในรีจิสเตอร์ของไมโครคอนโทรลเลอร์

4) การดูค่าในหน่วยความจำโปรแกรมบนบอร์ดไมโครคอนโทรลเลอร์

สามารถดูค่าได้ก่อนทำการรันเท่านั้น เพื่อดูค่าในหน่วยความจำโปรแกรมบนบอร์ดไมโครคอนโทรลเลอร์ เมื่อกดปุ่มค่าในหน่วยความจำโปรแกรม จะแสดงค่าในตำแหน่ง 8000H และเมื่อคลิกต่อไปก็จะแสดงค่าในตำแหน่งถัดไปอีก 80 ตัว ดังรูปที่ ค.27



รูปที่ ค.27 การดูค่าหน่วยความจำโปรแกรมบนบอร์ดไมโครคอนโทรลเลอร์

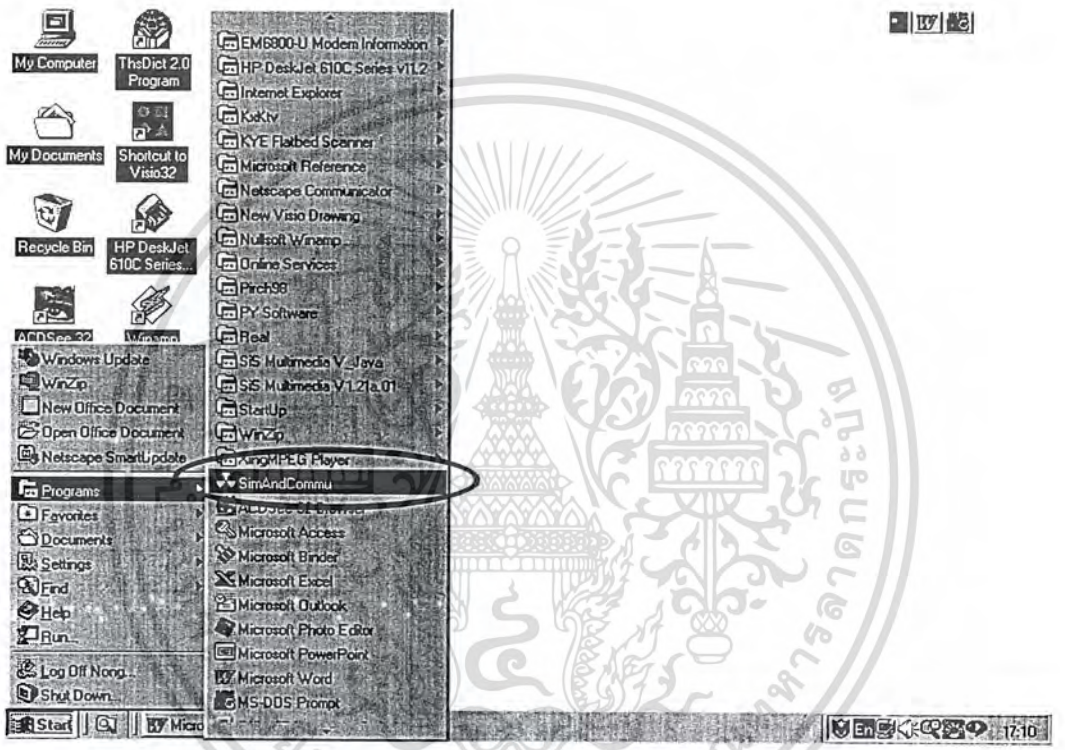
5) เปลี่ยนแฟ้ม

ในกรณีผู้ใช้งานต้องการเปลี่ยนโปรแกรมให้คลิกที่ปุ่มเปลี่ยนแฟ้ม เปรียบเสมือนกับปุ่มรีเซ็ต บนบอร์ดไมโครคอนโทรลเลอร์

2. ตัวอย่างการงานใช้โปรแกรมจำลองการทำงานของ MCS-51 และอุปกรณ์ต่อพ่วง

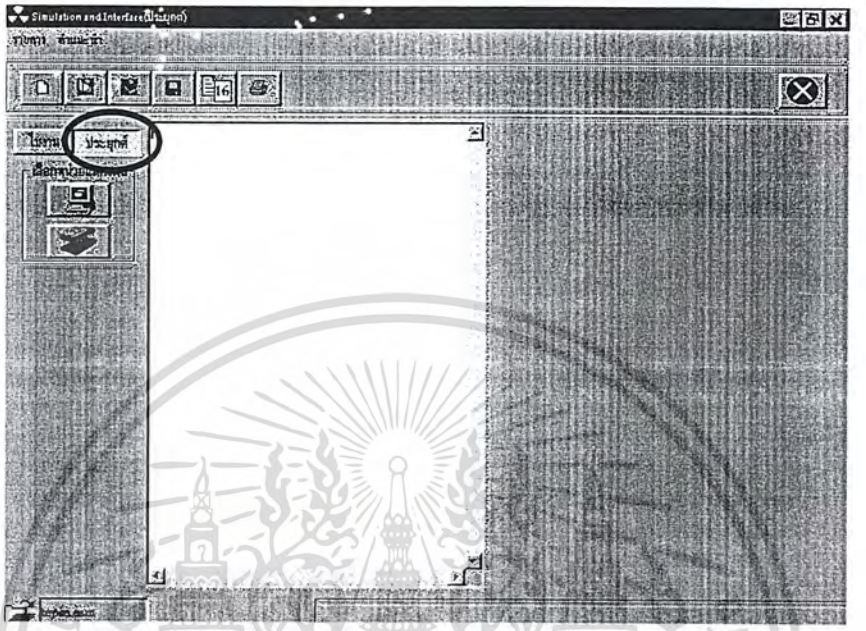
2.1 ขั้นตอนการใช้งาน

1) เปิดโปรแกรม MCS-51 Simulate Program ดังรูปที่ ค.28



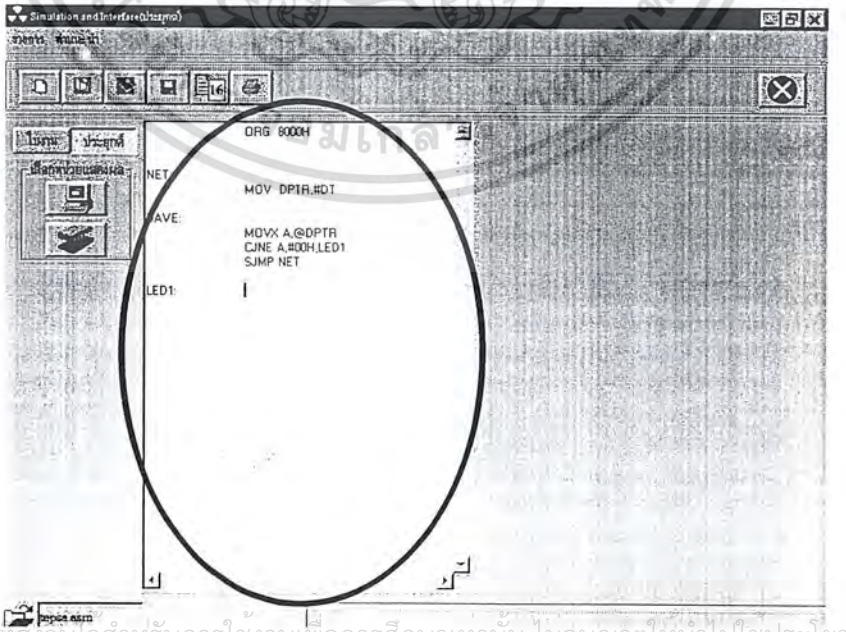
รูปที่ ค.28 การเปิดโปรแกรม

2) เลือกโหมดใช้งานเป็นประยุกต์ใช้งาน โดยคลิกที่ปุ่มประยุกต์ใช้งาน ดังรูปที่ ค.29



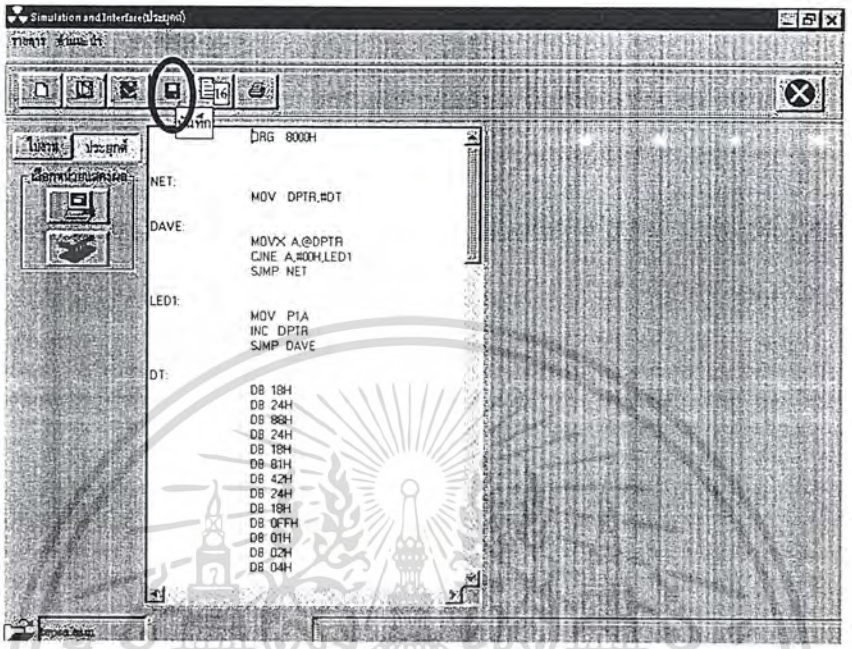
รูปที่ ค.28 คลิกที่ปุ่มประยุกต์ใช้งาน

3) เริ่มพิมพ์โปรแกรมที่ได้ออกแบบไว้ในกรอบข้อความ ดังรูปที่ ค.30



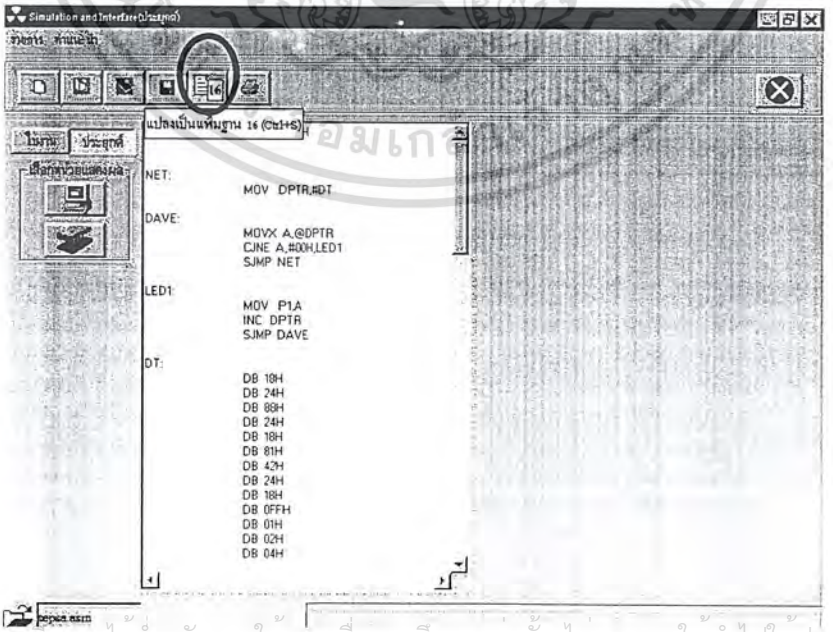
รูปที่ ค.30 การเริ่มพิมพ์โปรแกรม

4) บันทึกแฟ้มที่ทำการพิมพ์โดยคลิกที่ไอค่อนรูปแผ่นดิสก์เกต ดังรูปที่ ค.31



รูปที่ ค.31 การบันทึกข้อมูล (Save)

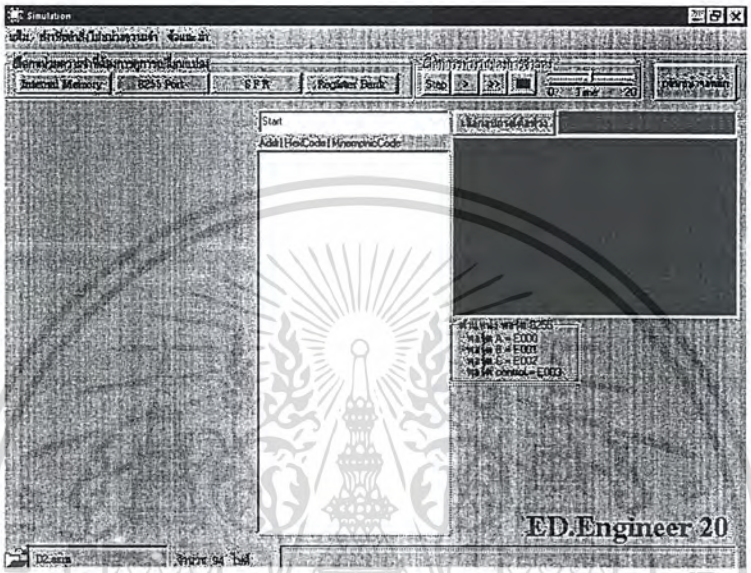
5) แปลงเป็นแฟ้มฐาน 16 (compile)โดยคลิกที่ปุ่มไอค่อนรูปเอกสาร 16 ดังรูปที่ ค.32



รูปที่ ค.32 แปลงแฟ้มข้อมูลเป็นฐาน 16 (Compile)

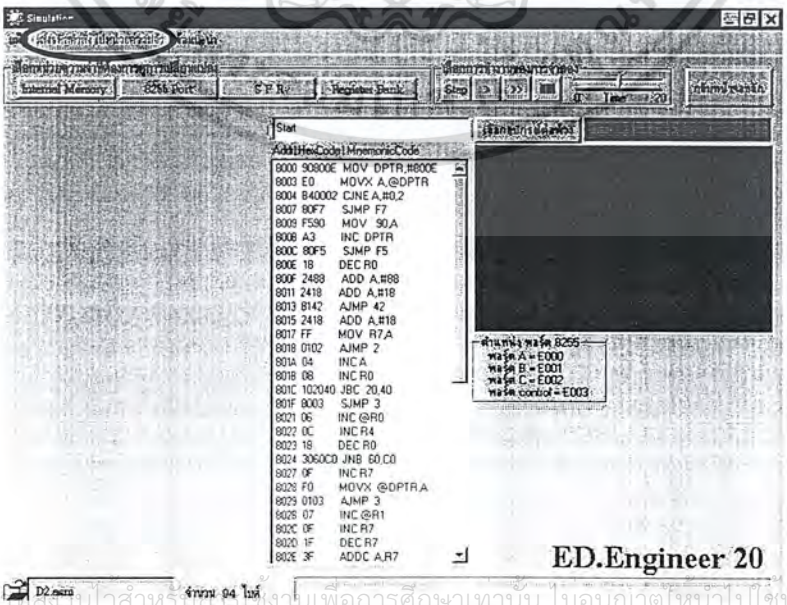
6) เลือกหน่วยแสดงผล

6.1) ดูผลการทำงานบนคอมพิวเตอร์ โดยคลิกที่ปุ่มไอคอนรูปคอมพิวเตอร์ ซึ่งจะปรากฏหน้าจอ ดังรูปที่ ค.33



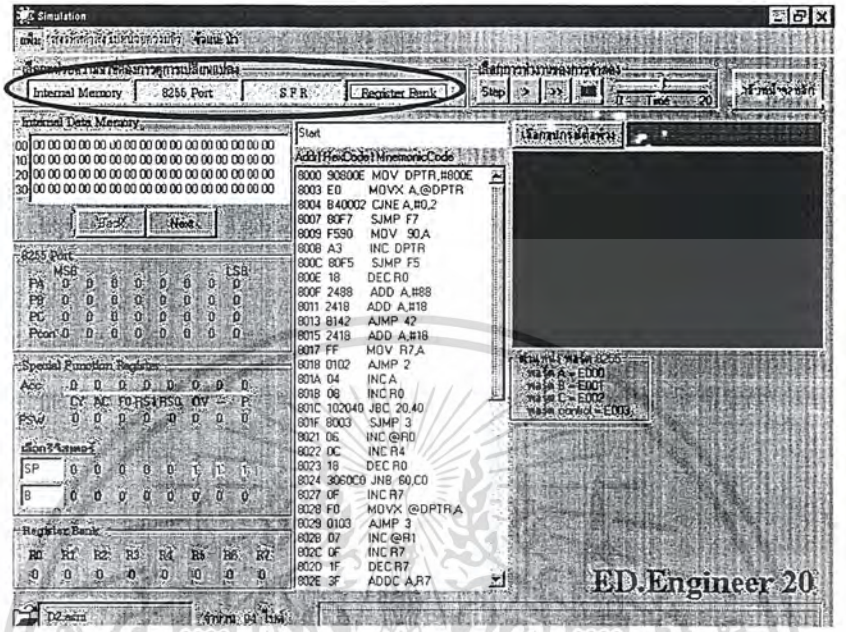
รูปที่ ค.33 เข้าสู่หน้าจอการจำลองการทำงาน

6.1.1) โหลดโปรแกรมไปยังหน่วยความจำโปรแกรมจำลอง ดังรูปที่ ค.34



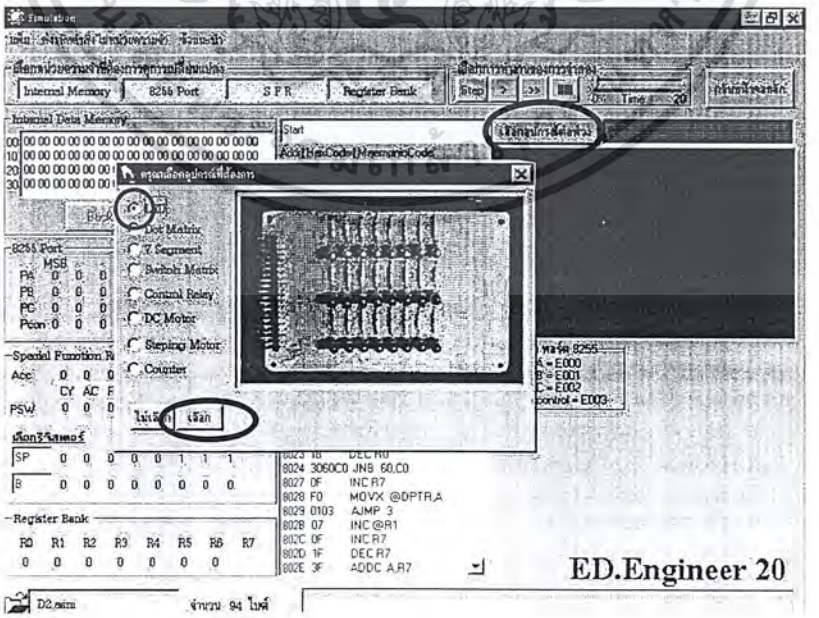
รูปที่ ค.34 การโหลดโปรแกรม

6.1.2) เลือกหน่วยความจำที่ต้องการดูการเปลี่ยนแปลง ดังรูปที่ ค.35



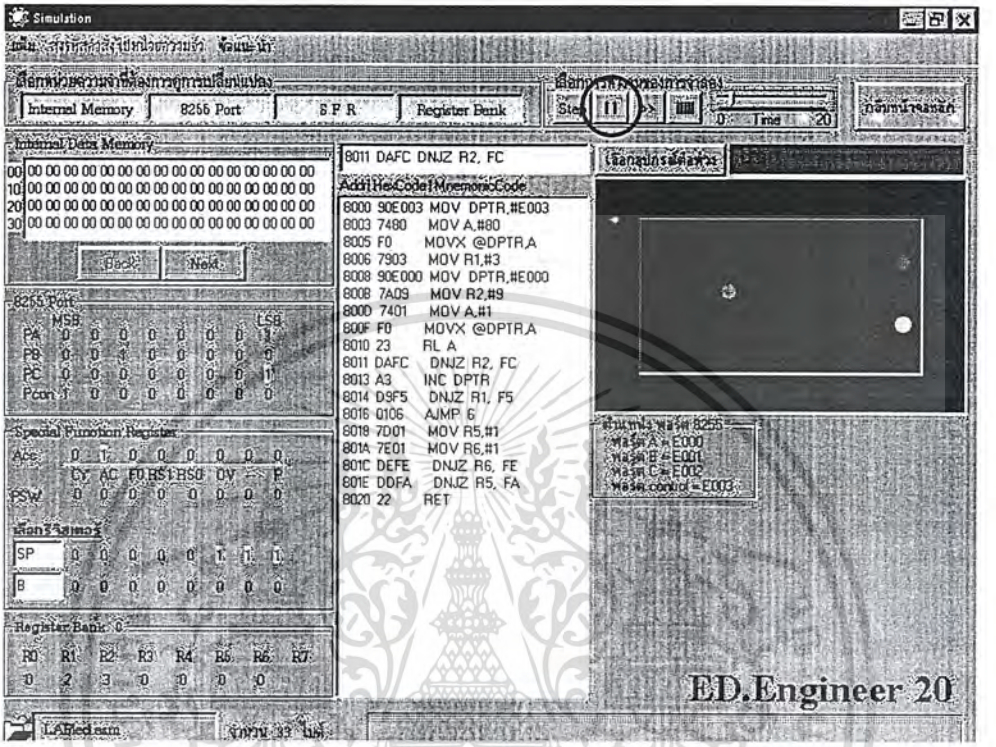
รูปที่ ค.35 เลือกหน่วยความจำที่ต้องการดูการเปลี่ยนแปลง

6.1.3) เลือกอุปกรณ์ต่อพ่วงจำลองที่ต้องการดูการเปลี่ยนแปลง ดังรูปที่ ค.36



รูปที่ ค.36 เลือกอุปกรณ์ต่อพ่วงจำลองที่ต้องการดูการเปลี่ยนแปลง

6.1.4) คลิกปุ่มดูผลการทำงานของโปรแกรมที่ปุ่ม  ปรากฏหน้าจอดังรูป ค.37

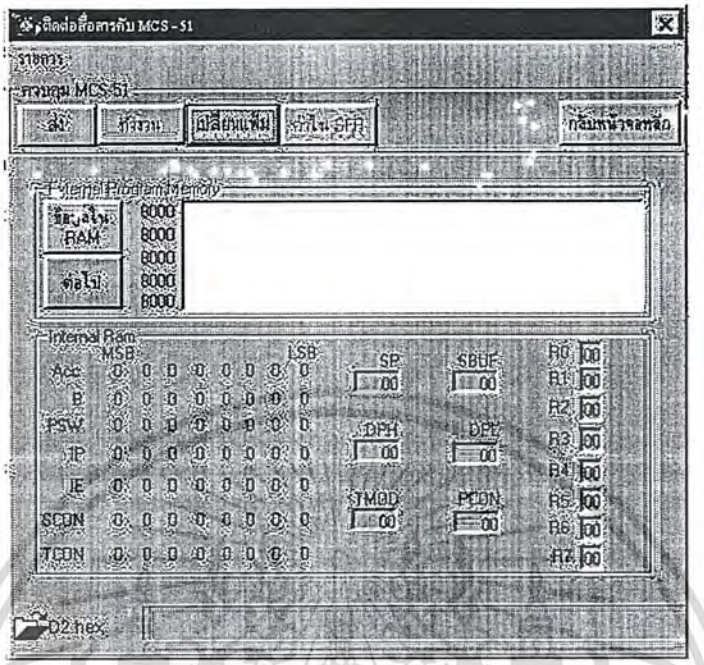


รูปที่ ค.37 กดปุ่มดูผลการทำงานของโปรแกรมที่ปุ่ม 

6.1.5) คลิกที่ปุ่ม  เพื่อหยุดการทำงาน

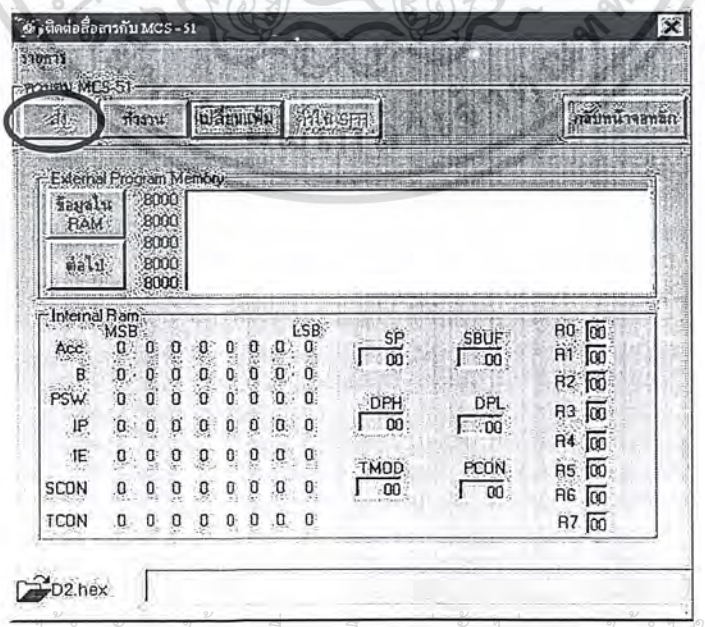
6.1.6) กลับสู่หน้าจอหลัก

6.2) ดูผลการทำงานบนไมโครคอนโทรลเลอร์โดยคลิกที่ปุ่มไอคอนรูปไอซี จะปรากฏหน้าจอการติดต่อกับ MCS-51 ขึ้น ดังรูปที่ ค.38



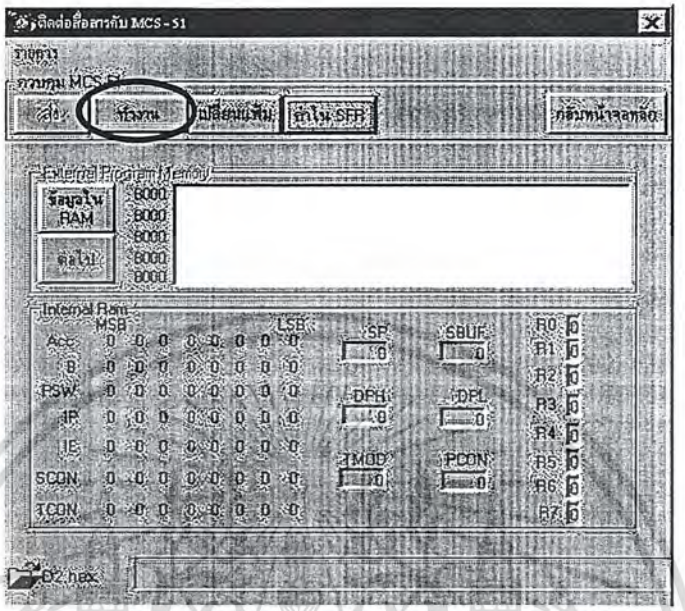
รูปที่ ค.38 เข้าหน้าจอติดต่อกับ MCS-51

6.2.1) โหลดโปแกรมไปยังหน่วยความจำโปรแกรมบนบอร์ดไมโครคอนโทรลเลอร์ โดยกดปุ่มส่ง ดังรูปที่ ค.39



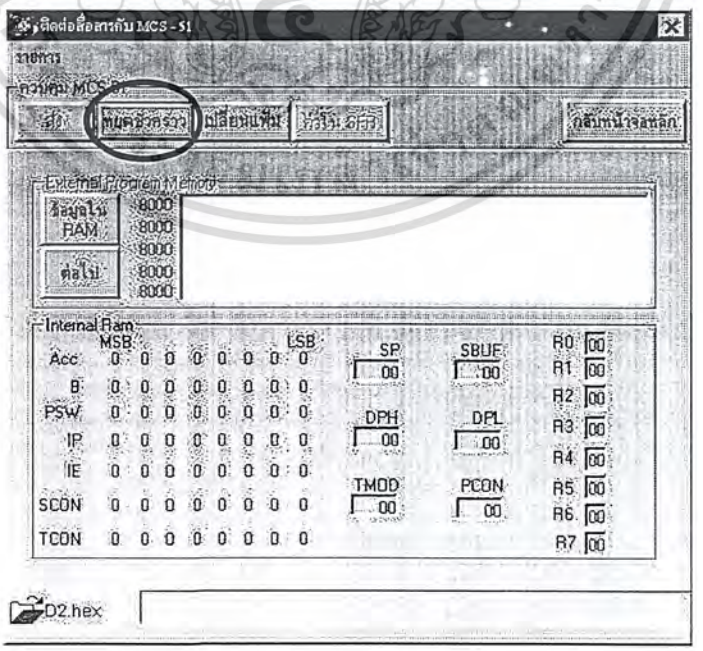
รูปที่ ค.39 โหลดโปรแกรมโดยคลิกที่ปุ่มส่ง

6.2.4) ดูผลการทำงานของโปรแกรมโดยคลิกปุ่มทำงาน ดังรูปที่ ค.40



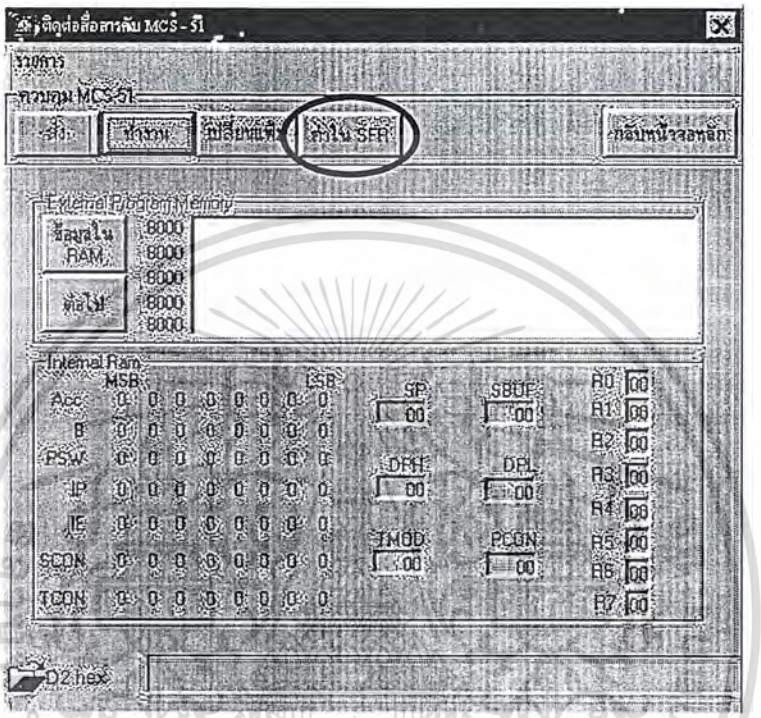
รูปที่ ค.40 ดูผลการทำงานของโปรแกรมโดยคลิกที่ปุ่มทำงาน

6.2.5) หยุดการทำงานของโปรแกรมชั่วขณะโดยกดปุ่มหยุดการทำงาน ดังรูปที่ ค.41



รูปที่ ค.41 หยุดการทำงานของโปรแกรมชั่วขณะโดยคลิกที่ปุ่มหยุดชั่วคราว

6.2.6) เรียกดูค่าในรีจิสเตอร์ภายในไมโครคอนโทรลเลอร์ ในขณะที่หยุดการทำงานชั่วขณะในตอนนั้น โดยคลิกที่ปุ่มค่าใน SFR ดังรูปที่ ค.42



รูปที่ ค.43 การคลิกดูค่ารีจิสเตอร์ภายในไมโครคอนโทรลเลอร์

7) กลับสู่หน้าจอหลัก

8) จบการทำงานโดยคลิกที่ปุ่มรูป  หรือเลือกจากเมนู/จบการทำงาน

ภาคผนวก ง

ใบงาน

ใบงานที่ 1

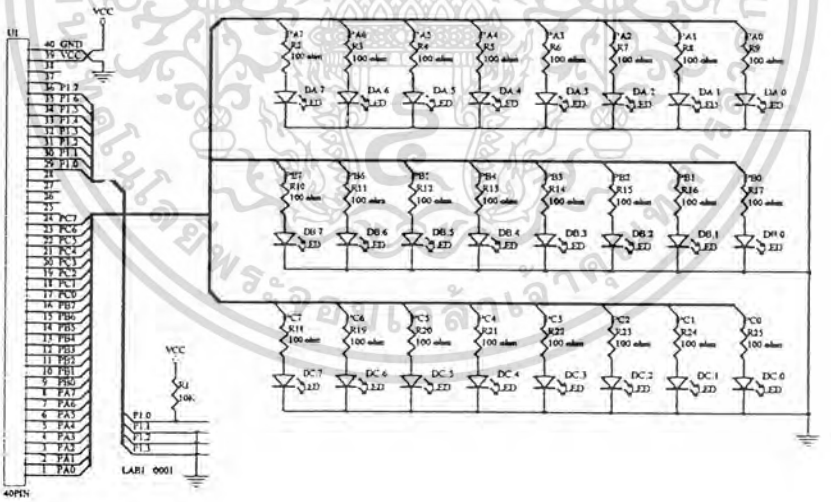
แอลอีดีลอจิก (LED LOGIC)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถ ส่งค่าออกทางพอร์ตทั้ง 3 ของวงจรแอลลีดีลอจิกได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรม เพื่อควบคุมวงจรแอลอีดีลอจิกได้

ทฤษฎี

การทดลองที่ 1 ออกแบบให้ใช้งานในการทดลองสถานะพอร์ตทั้ง 3 พอร์ตของวงจรไมโครคอนโทรลเลอร์คือ พอร์ต A, พอร์ต B และพอร์ต C แต่ละพอร์ตจะแสดงสถานะด้วยหลอดแอลอีดี ใน 1 พอร์ต มี 8 บิต ดังนั้นจึงใช้แอลอีดี 8 ดวงต่อ 1 พอร์ต ดังแสดงในวงจรรูปที่ 1



รูปที่ 1.1 วงจรการเชื่อมต่อกับแอลอีดีลอจิก

จากรูปที่ 1 สถานะลอจิกเท่ากับ "1" มีแรงดันเท่ากับ 4.2–4.8 โวลต์ แอลอีดีต้องการแรงดันเท่ากับ 2.2โวลต์ จึงจำเป็นต้องต่อตัวต้านทาน 100โอห์ม เพื่อจำกัดกระแสและแบ่งแรงดันให้เหมาะสมกับแอลอีดี

ปรับเวลาไว้ที่ 0

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง
2. ในใบงานนี้การทำงานของ SP มีลักษณะการทำงานอย่างไร
3. ข้อมูลที่เปลี่ยนแปลงใน DPH และ DPL เป็นค่าของอะไร
4. หากต้องการเปลี่ยนทิศทางการวิ่งของแอลอีดี จะสามารถแก้ไขโปรแกรมที่ส่วนใดบ้าง
5. จงเขียนโปรแกรมเพื่อให้วงจรแอลดีดีลอจิก ติดสว่างทุกดวง และดับลง สลับไปมาเริ่มจากแถวที่1 ลงมาจนถึงแถวที่3 แล้ววนกลับไปเริ่มแถวที่1ใหม่

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรม คลิกที่ ใบงาน เลือก ใบงานที่ 1
2. เลือกหน่วยแสดงผลเป็นทำงานบนคอนโทรลเลอร์ (แสดงไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

.....

.....

.....

ตอนที่ 2

1. เลือกหน่วยแสดงผลเป็นทำงานบนคอมพิวเตอร์ (แสดงไอค่อนรูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง
3. เลือกการทำงานของการจำลองเป็นแบบทีละขั้น โดยการคลิกที่ปุ่ม STEP
4. คลิกการทำงานทีละขั้นที่ปุ่ม >> และบันทึกผลการทดลองในตาราง
5. เลือกการทำงานของการจำลองโดยคลิกที่ปุ่ม STEP อีกครั้ง แล้วคลิกที่คลิกปุ่ม > เข้าสู่การทำงานปกติ ทดลองปรับความเร็วที่แถบเครื่องสปีดบาร์ที่ตำแหน่งต่อไปนี้ บันทึกผล

ผลการทดลอง(ตอนที่ 2)

ปรับเวลาไว้ที่ 20

.....

.....

.....

ตารางบันทึกผลการทดลอง ที่ .....

ครั้งที่	ตำแหน่ง	รหัสฐาน 16	รหัสคำสั่ง	หน่วยความจำภายใน														หน่วยความจำภายนอก			
				Acc	DPH	DPL	PSW	SP	R0	R1	R2	R3	R4	R5	R6	R7		PA	PB	PC	Pcon
1	8000	90 E0 00	MOV DPTR,#0E000	0 0	E0	0 3	0 0	0 7	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0		0 0	0 0	0 0	0 0

ใบงานที่ 2

แอลอีดีดอตเมตริกซ์ (LED DOT MATRIX)

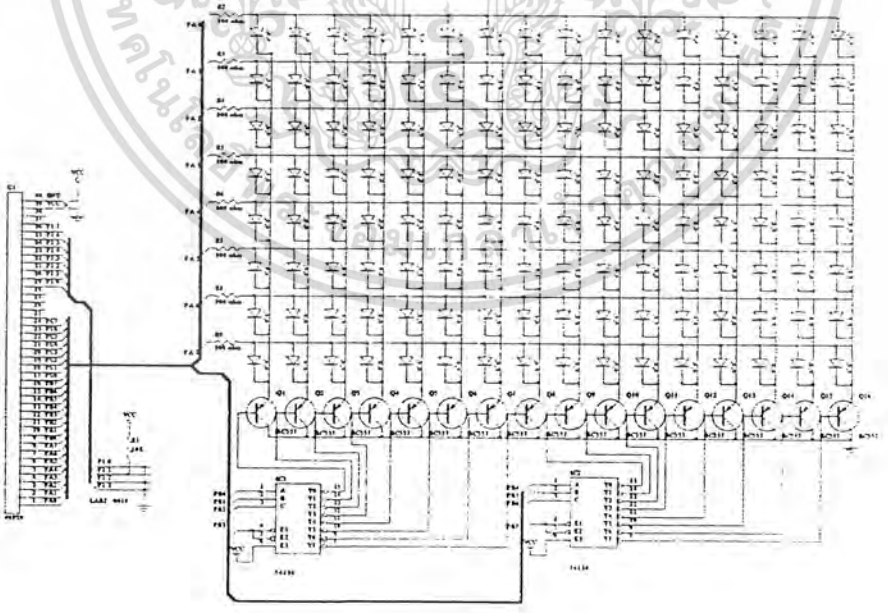
วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถถอดรหัสจากภาพ, ข้อความหรือ สัญลักษณ์ ออกเป็นข้อมูลตัวเลข ฐาน 16 ได้

2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรแอลอีดีดอตเมตริกซ์ได้

ทฤษฎี

การทดลองที่ 2 ออกแบบเพื่อใช้เป็นอุปกรณ์แสดงรูปภาพ, ข้อความ หรือสัญลักษณ์ ในการสร้างภาพด้วยจุดทั้งแนวนอนและแนวตั้งรวมกัน หมายถึง หากต้องการให้แอลอีดีดวงใดติดสว่างจำเป็นต้องทำให้ครบวงจรทั้งแนวนอน และแนวตั้ง กำหนดให้ ลอจิก "1" เป็นแรงดันทางแนวนอนและลอจิก "0" เป็นแรงดันในแนวตั้ง จึงเกิดการครบวงจร



รูปที่ 2.1 วงจรการเชื่อมต่อกับแอลอีดีดอทเมตริกส์

จากวงจรในรูปที่ 2.1 การให้ลอจิก "1" ทางแนวนอนกำหนดโดยพอร์ต A ของไมโครคอนโทรลเลอร์ ส่วนลอจิก "0" ที่ทำให้แอลอีดีติดสว่างได้จะถูกควบคุมด้วยทรานซิสเตอร์ชนิดพีเอ็นพี จะทำงานด้วยลอจิก "0" จากไอซีดีโค้ดเดอร์เบอร์ 74138 รับค่าลอจิกจากพอร์ต B ของวงจรไมโครคอนโทรลเลอร์

การควบคุมแนวนอน ถูกควบคุมด้วยพอร์ต A แบ่งออกเป็นแถว 1 แถว ต่อสภาวะ 1 บิต ของพอร์ต A มีวิธีเรียงลำดับ ดังนี้ Port A0-A7 เรียงจากแถวด้านบนคือแถวที่ 1 จนถึงแถวด้านล่างคือแถวสุดท้ายแถวที่ 8

การควบคุมทางด้านแนวตั้ง มีทั้งหมด 16 หลัก เริ่มจากหลักที่ 1 ทางด้านซ้าย เรียงลำดับไปถึงหลักที่ 16 ทางด้านขวา เพื่อให้ง่ายต่อการควบคุมจึงใช้ไอซีดีโค้ดเดอร์เบอร์ 74138 ควบคุมด้วยพอร์ต B 4 บิตล่าง สำหรับหลักที่ 1-8 และ 4 บิตบนสำหรับ หลักที่ 9-6 ดังตารางที่ 2.1

ตารางที่ 2.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 2

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	หลักที่	ฐาน 16
1	0	0	0	0	0	0	0	1	80
1	0	0	0	0	0	0	1	2	81
1	0	0	0	0	0	1	0	3	82
1	0	0	0	0	0	1	1	4	83
1	0	0	0	0	1	0	0	5	84
1	0	0	0	0	1	0	1	6	85
1	0	0	0	0	1	1	0	7	86
1	0	0	0	0	1	1	1	8	87
0	0	0	0	1	0	0	0	9	08
0	0	0	1	1	0	0	0	10	18
0	0	1	0	1	0	0	0	11	28
0	0	1	1	1	0	0	0	12	38
0	1	0	0	1	0	0	0	13	48
0	1	0	1	1	0	0	0	14	58
0	1	1	0	1	0	0	0	15	68
0	1	1	1	1	0	0	0	16	78
0	0	0	0	0	0	0	0	ไม่แสดงหลัก	00

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรมคลิกที่ปุ่ม [ใบงาน] เลือกใบงานที่ 2
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูป ไอซี)
3. คลิกปุ่ม [ส่ง] เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม [ทำงาน] เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

.....

.....

.....

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (รูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง
3. เลือกการทำงานของการจำลองเป็นแบบทีละขั้น โดยการคลิกที่ปุ่ม [STEP]
4. คลิกการทำงานทีละขั้นที่ปุ่ม [>>] และบันทึกผลการทดลองในตาราง
5. เลือกการทำงานของการจำลองโดยคลิกที่ปุ่ม [STEP] อีกครั้ง แล้วคลิกที่ปุ่ม [>] เพื่อเข้าสู่ การทำงานปกติ ทดลองปรับความเร็วที่แถบเครื่องมือสปีดบาร์ที่ตำแหน่งต่อไปนี้ แล้วบันทึกผล

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

.....

.....

.....

.....

ปรับเวลาไว้ที่ 0

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง
2. จงเขียนโปรแกรมเพื่อให้วงจรแอลอีดีดอทเมตริกส์ แสดงข้อความ “HELLO”

ตารางบันทึกผลการทดลอง ที่ .....

ครั้งที่	ตำแหน่ง	รหัสฐาน 16	รหัสคำสั่ง	หน่วยความจำภายใน													หน่วยความจำภายนอก			
				Acc	DPH	DPL	PSW	SP	R0	R1	R2	R3	R4	R5	R6	R7	PA	PB	PC	Pcon
1	8000	90 E0 00	MOV DPTR,#0E000	0 0	E0	0 3	0 0	0 7	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0

ตารางบันทึกผลการทดลอง ที่ .. ..

ครั้งที่	ตำแหน่ง	รหัสฐาน 16	รหัสคำสั่ง	หน่วยความจำภายใน													หน่วยความจำภายนอก			
				Acc	DPH	DPL	PSW	SP	R0	R1	R2	R3	R4	R5	R6	R7	PA	PB	PC	Pcon

ใบงานที่ 3

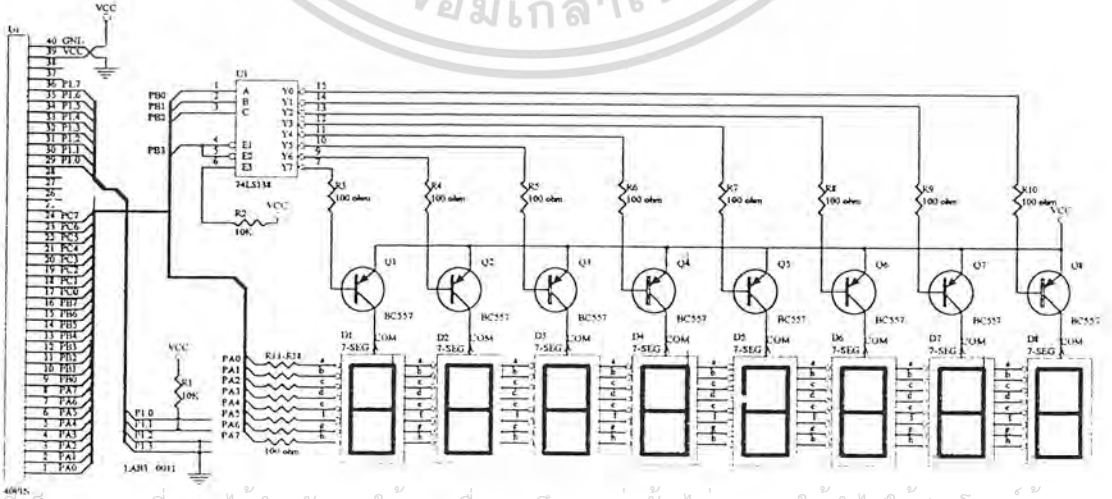
จอแสดงผลแบบ 7 ส่วน (SEVEN-SEGMENT)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถใช้งานจอแสดงผลแบบ 7 ส่วน แบบขั้วบวกร่วมกันได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรเชื่อมต่อกับจอแสดงผลแบบ 7 ส่วนได้

ทฤษฎี

การทดลองที่ 3 เป็นการทดลองใช้งานจอแสดงผลแบบ 7 ส่วน แบบมีขั้วบวกร่วมกัน ดังนั้นเมื่อจะทำให้แอลอีดีของจอแสดงผลแบบ 7 ส่วน ส่วนใดติดสว่างต้องให้ส่วนนั้นได้รับลอจิก "0" สำหรับการออกแบบวงจรการทดลองที่ 3 นี้ ใช้ จอแสดงผลแบบ 7 ส่วนจำนวน 8 ตัวแอลอีดีแต่ละส่วนของทุกตัวเลข จะต่อถึงกันและควบคุมด้วยพอร์ต A ของวงจรไมโครคอนโทรลเลอร์ หากต้องการให้ส่วนใดของตัวเลขติดสว่าง ก็ให้ลอจิกนั้นมีสถานะเป็น "0" แต่จะติดสว่างที่ตัวเลขหลักใดนั้นขึ้นอยู่กับจะให้หลักใดครบวงจร ในวงจรนี้เลือกใช้ทรานซิสเตอร์ชนิด PNP เป็นอุปกรณ์สวิตช์โดยควบคุมการเลือกหลักของตัวเลขด้วยพอร์ต B ผ่านทางไอซีดีโค้ดเดอร์เบอร์ 74138 ใช้พอร์ต B เพียง 4 เส้นในการควบคุมหลักของจอแสดงผลแบบ 7 ส่วน เลือกใช้พอร์ต B ไบต์ต่ำเพื่อควบคุมหลักที่ใช้แสดงผล เริ่มต้นหลักที่ 1 ทางด้านขวา จนถึงหลักที่ 8 ทางด้านซ้าย



รูปที่ 3.1 วงจรการเชื่อมต่อกับจอแสดงผลแบบ 7 ส่วน

ตารางที่ 3.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 3

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	หลักที่	ฐาน 16
X	X	X	X	0	0	0	0	1	X0
X	X	X	X	0	0	0	1	2	X1
X	X	X	X	0	0	1	0	3	X2
X	X	X	X	0	0	1	1	4	X3
X	X	X	X	0	1	0	0	5	X4
X	X	X	X	0	1	0	1	6	X5
X	X	X	X	0	1	1	0	7	X6
X	X	X	X	0	1	1	1	8	X7
X	X	X	X	1	0	0	0	ไม่แสดงหลัก	-

หมายเหตุ : X หมายถึงค่าใดๆ

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรมคลิกที่ ใบงาน เลือก ใบงานที่ 3
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

.....

.....

.....

.....

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (รูปคอมพิวเตอร์)

2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง

3. เลือกการทำงานของการจำลองเป็นแบบทีละขั้น โดยการคลิกที่ปุ่ม [STEP]

4. คลิกการทำงานทีละขั้นที่ปุ่ม [>>] และบันทึกผลการทดลองในตาราง

5. เลือกการทำงานของการจำลองเป็นแบบธรรมดา โดยคลิกที่ปุ่ม [STEP] อีกครั้ง แล้วคลิกที่ปุ่ม [>] เข้าสู่การทำงานปกติ ทดลองปรับความเร็วที่แถบเครื่องมือสปีดบาร์ที่ตำแหน่งต่อไปนี้ แล้วบันทึกผล

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

.....

.....

.....

ปรับเวลาไว้ที่ 0

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

2. จงเขียนโปรแกรมที่ทำให้จอแสดงผลแบบ 7 ส่วน แสดงตัวเลข 0-7 ที่หลัก 0 ถึงหลัก 7 เรียงตามลำดับ

3. ในโปรแกรมการทดลองนี้ค่า Carry Flag ใน PSW จะเปลี่ยนแปลงเป็น 1 เมื่อใด

ตารางบันทึกผลการทดลอง ที่ .....

ครั้งที่	ตำแหน่ง	รหัสฐาน 16	รหัสคำสั่ง	หน่วยความจำภายใน													หน่วยความจำภายนอก			
				Acc	DPH	DPL	PSW	SP	R0	R1	R2	R3	R4	R5	R6	R7	PA	PB	PC	Pcon
1	8000	90 E0 00	MOV DPTR,#0E000	0 0	E0	0 3	0 0	0 7	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0

ตารางบันทึกผลการทดลอง ที่ ....

ครั้งที่	ตำแหน่ง	รหัสฐาน 16	รหัสคำสั่ง	หน่วยความจำภายใน													หน่วยความจำภายนอก			
				Acc	DPH	DPL	PSW	SP	R0	R1	R2	R3	R4	R5	R6	R7	PA	PB	PC	Pcon

ใบงานที่ 4

สวิตช์เมตริกซ์ (SWITCH MATRIX)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถถอดรหัส เมื่อมีการกดสวิตช์ แบบคูณหลักได้
2. เพื่อให้นักศึกษาสมารถเขียนโปรแกรมควบคุมวงจรสวิตช์เมตริกส์ ได้

ทฤษฎี

การทำงานของสวิตช์ ที่ต่อใช้งานแบบแถวคูณหลัก จะควบคุมและตรวจสอบค่าด้วยสภาวะของพอร์ต C สภาวะเริ่มต้นกำหนดให้ค่าพอร์ตทุกบิตเป็น "1" และใช้การทดสอบทีละ 1 แถว โดยกำหนดค่าออกด้วยพอร์ต C ไบต์สูง คือ PC4–PC7 และรับค่าเข้าเพื่อพิจารณาค่าด้วยพอร์ต C ไบต์ต่ำ คือ PC0–PC3 คือทดสอบทีละ 1 แถวนั่นเอง 1 แถวพิจารณาสวิตช์ 4 ตัว ดังนั้นสวิตช์ 4 แถว ทดสอบสวิตช์ได้ทั้งหมด 16 ตัว 16 กรณี สามารถเขียนได้ดังตารางที่ 3.1

ตารางที่ 3.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 4

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์ที่	ฐาน 16
1	1	1	0	1	1	1	0	0	EE
1	1	1	0	1	1	0	1	1	ED
1	1	1	0	1	0	1	1	2	EB
1	1	1	0	0	1	1	1	3	E7
1	1	0	1	1	1	1	0	4	DE
1	1	0	1	1	1	0	1	5	DD
1	1	0	1	1	0	1	1	6	DB
1	1	0	1	0	1	1	1	7	D7
1	0	1	1	1	1	1	0	8	BA
1	0	1	1	1	1	0	1	9	BD
1	0	1	1	1	0	1	1	A	BB
1	0	1	1	0	1	1	1	B	B7
0	1	1	1	1	1	1	0	C	7E
0	1	1	1	1	1	0	1	D	7D
0	1	1	1	1	0	1	1	E	7B
0	1	1	1	0	1	1	1	F	77

ผลการทดลอง (ตอนที่ 1)

.....

.....

.....

.....

.....

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง
3. ทดสอบโดยการคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง บันทึกค่า

3.1 ปุ่ม 1 .....

3.2 ปุ่ม 2 .....

3.3 ปุ่ม 3 .....

3.4 ปุ่ม 4 .....

3.5 ปุ่ม 5 .....

3.6 ปุ่ม 6 .....

3.7 ปุ่ม 7 .....

3.8 ปุ่ม 8 .....

3.9 ปุ่ม 9 .....

3.10 ปุ่ม 10 .....

3.11 ปุ่ม 11 .....

3.12 ปุ่ม 12 .....

3.13 ปุ่ม 13 .....

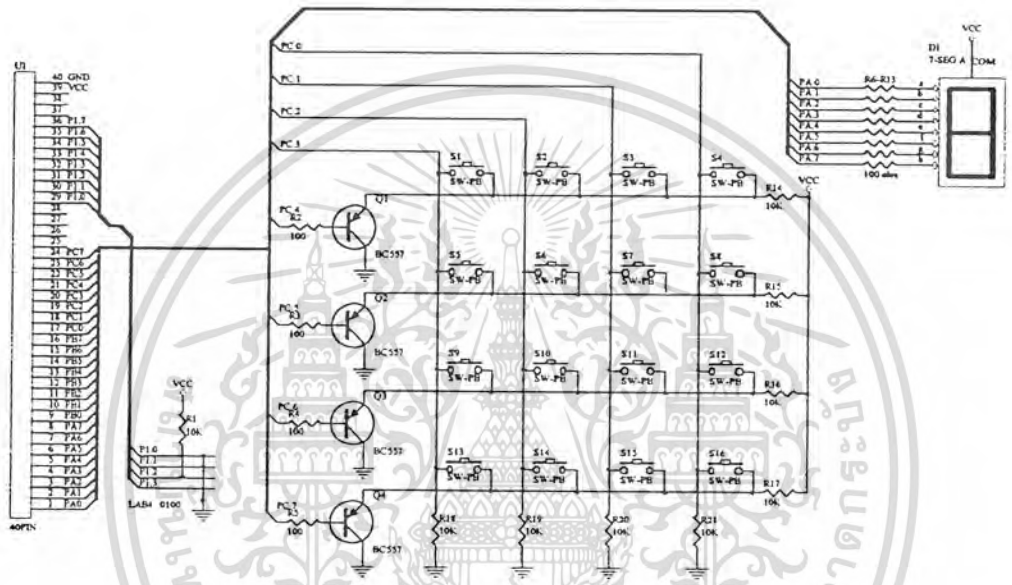
3.14 ปุ่ม 14 .....

3.15 ปุ่ม 15 .....

3.16 ปุ่ม 16 .....

กรณีทั้ง 16 กรณีจะเกิดขึ้น เมื่อมีการกดสวิตช์พอร์ต C ไบต์ต่ำ จะเปลี่ยนสภาวะจาก "1" เป็น "0" ส่วนพอร์ต C ไบต์สูง จะกำหนดว่าจะพิจารณาแถวใดก็ให้แถวนั้นมีสภาวะเป็น "0"

ในวงจรการทดลองนี้ได้ใช้จอแสดงผลแบบ 7 ส่วน เพื่อใช้แสดงตำแหน่งของสวิตช์ที่ถูกกด จอแสดงผลแบบ 7 ส่วน ถูกควบคุมด้วยสภาวะของพอร์ต A คือต้องการให้ส่วนของตัวเลขติดสว่าง ก็ให้บิตของตัวเลขส่วนนั้นมีสภาวะเป็น "0" วงจรการเชื่อมต่อแสดงดังรูปที่ 4.1



รูปที่ 4.1 วงจรการเชื่อมต่อกับสวิตช์เมตริกซ์

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรมคลิกที่ ใบงาน เลือก ใบงานที่ 4
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ใบงานที่ 5

จอแสดงผลชนิดผลึกเหลว (LCD)

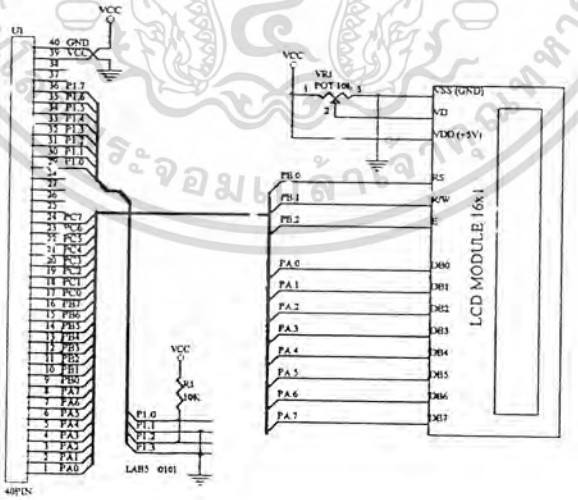
วัตถุประสงค์

1. เพื่อให้นักศึกษาสารมารถสั่งงานวงจรการเชื่อมต่อกับจอผลแบบผลึกเหลวขนาด 16 x1 ได้

2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมการแสดงผล บนจอแสดงผล ชนิดผลึกเหลว ได้

ทฤษฎี

การทดลองที่ 5 เป็นการทดลองการใช้งานอุปกรณ์แสดงผลชนิดผลึกเหลวซึ่งการควบคุมประกอบด้วย 14 ขา ควบคุมและส่งข้อมูลขาควบคุมการเขียนการอ่านข้อมูล RS, R/Z และ E ควบคุมด้วยพอร์ต B คือ B0, B1 และ B2 ตามลำดับ ส่วนขาควบคุมการส่งข้อมูล DB0–DB7 ควบคุมด้วยพอร์ต A0–A7 ตามลำดับ ดังแสดงในรูปที่ 5.1



รูปที่ 5.1 วงจรการเชื่อมต่อกับอุปกรณ์แสดงผลชนิดผลึกเหลว

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม คลิกที่ ใบงาน เลือก ใบงานที่ 5
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูป ไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง

ผลการทดลอง

.....

.....

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงเขียนโปรแกรมแสดงข้อความ "THAILAND"

4. คลิกปุ่ม [>] เพื่อกลับสู่การทำงานปกติ พร้อมทั้งคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง แล้วบันทึกผลการทดลอง

ผลการทดลอง (ตอนที่ 2)

.....

.....

.....

.....

.....

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงเขียนโปรแกรมแสดงค่าการคลิกสวิตช์ โดยกำหนด สวิตช์ 1 เป็นค่า A สวิตช์ 5 มีค่าเป็น B สวิตช์ 9 มีค่าเป็น D

2. เหตุใดเมื่อกดสวิตช์ ในตัวอุปกรณ์จำลองแล้วจึงมีการตอบสนองช้า

3. จงบอกวิธีที่สามารถทำให้ตัวแสดง 7 ส่วนเปลี่ยนแปลงได้รวดเร็วขึ้นเมื่อกดสวิตช์ใด สวิตช์หนึ่ง บนอุปกรณ์จำลอง

ใบงานที่ 6

สวิตซ์ควบคุมการทำงาน (SWITCH CONTROL)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถสั่งงานวงจรควบคุมรีเลย์ได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรสวิตซ์ควบคุมการทำงานได้

ทฤษฎี

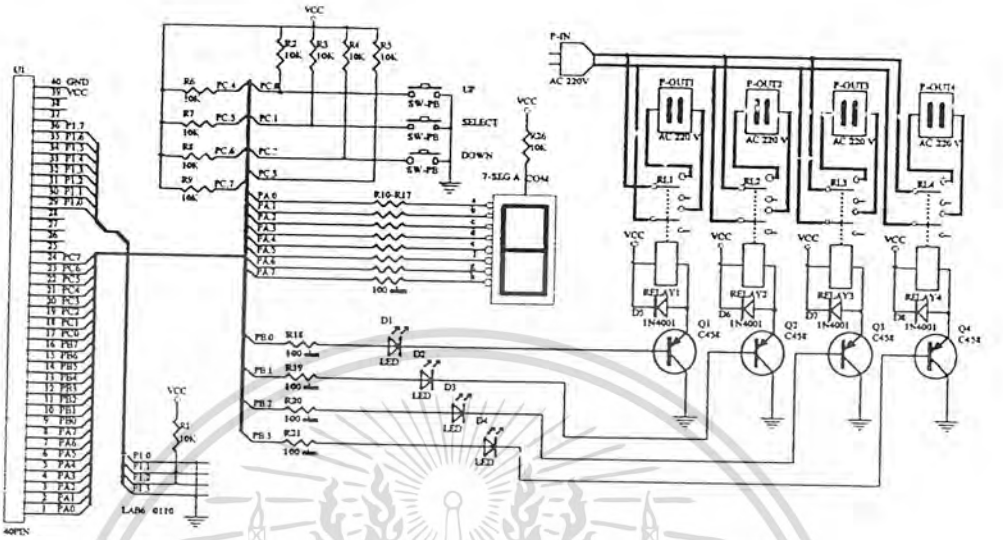
การทดลองที่ 6 ออกแบบเพื่อควบคุมการทำงานของรีเลย์ โดยอาศัยสภาวะของพอร์ต B เป็นตัวกำหนดรีเลย์ทำงาน เมื่อได้รับสภาวะลอจิก "1" ผ่านแอลอีดีแสดงผล ใช้รีเลย์จำนวน 4 ตัว รีเลย์ 1–4 ควบคุมด้วยพอร์ต B0–B3 ตามลำดับ

สวิตช์ควบคุมสำหรับวงจรทดลองนี้ มี 3 สวิตช์ ถูกกำหนดและตรวจสอบด้วยพอร์ต C เมื่อกดสวิตช์ต่างๆ จะมีค่าที่พอร์ต C ดังตารางที่ 6.1

ตารางที่ 6.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 6

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	1	1	1	1	0	1	FE
1	1	1	1	1	1	0	1	2	FD
1	1	1	1	1	0	1	1	3	FB

นอกจากนั้นยังใช้จอแสดงผลแบบ 7 ส่วน เป็นอุปกรณ์แสดงค่าควบคุมด้วยพอร์ต A เนื่องจากเป็นตัวเลข 7 ส่วนที่มีขั้วบวกร่วมกัน การแสดงตัวเลข จึงต้องการลอจิก "0" เมื่อกำหนดให้ตัวเลขส่วนใดติดสว่าง ดังแสดงวงจรในรูปที่ 6.1



รูปที่ 6.1 วงจรการเชื่อมต่อกับสวิตช์ควบคุมการทำงาน

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรม คลิกที่ ใบงาน เลือก ใบงานที่ 6
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผลด้วยการกดปุ่ม 1 ถึง 3
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

.....

.....

.....

.....

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)

2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง

3. ทดสอบโดยการคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง บันทึกค่า

   3.1 ปุ่ม 1 .....

   3.2 ปุ่ม 2 .....

   3.3 ปุ่ม 3 .....

4. คลิกปุ่ม [>] เพื่อกลับสู่การทำงานปกติ พร้อมทั้งคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง แล้วบันทึกผลการทดลองดังนี้

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

.....

.....

.....

ปรับเวลาไว้ที่ 0

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

2. จงเขียนโปรแกรม ให้มีการควบคุมรีเลย์ โดยสวิตช์ 1 ควบคุมการทำงานของรีเลย์ สวิตช์ 2 ควบคุมการทำงานของรีเลย์ตัวที่ 2 และ สวิตช์ 3 ควบคุมการทำงานของรีเลย์ตัวที่ 3 กดสวิตช์ครั้งแรกสั่งให้รีเลย์ทำงาน กดสวิตช์อีกครั้งสั่งให้รีเลย์หยุดทำงาน

ใบงานที่ 7

มอเตอร์ไฟฟ้ากระแสตรง (DC MOTER)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถควบคุมทิศทางการหมุนของมอเตอร์ไฟฟ้ากระแสตรงได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรเชื่อมต่อมอเตอร์ไฟฟ้ากระแสตรงได้

ทฤษฎี

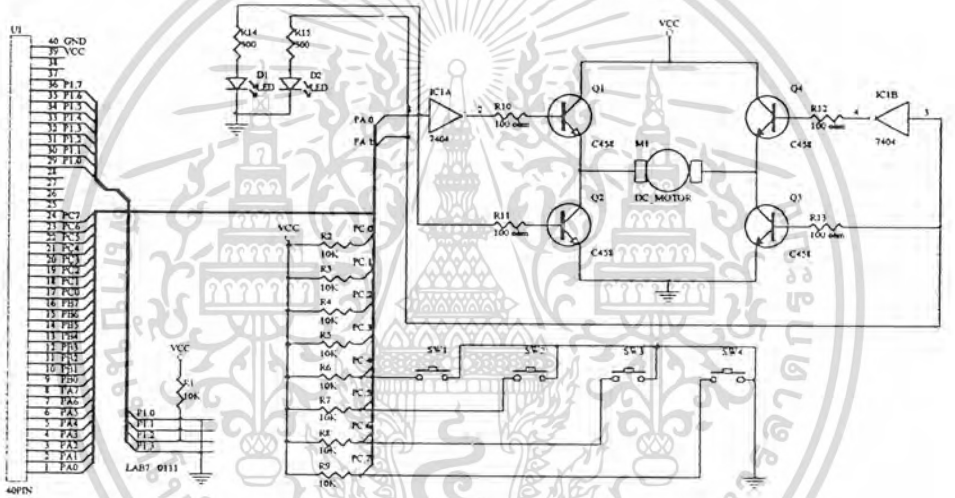
การทดลองที่ 7 เป็นวงจรควบคุมทิศทางการหมุนของมอเตอร์ไฟฟ้ากระแสตรงเลือกใช้ทรานซิสเตอร์ชนิดเอ็นพีเอ็น 4 ตัว ต่อกันในลักษณะกากบาทโดยให้มอเตอร์กระแสตรงอยู่ตรงกลาง และให้สภาวะของทรานซิสเตอร์ที่อยู่คนละด้านคือ Q1 และ Q3 มีสภาวะเดียวกันเช่นเดียวกับ Q2 และ Q4 และสามารถควบคุมสภาวะของทรานซิสเตอร์ 2 คู่นี้ด้วยพอร์ต A0 และ A1 โดยมีไอซีนอตเกทเป็นตัวกลับลอจิกให้ตรงข้ามกัน แสดงความสัมพันธ์ของพอร์ต A0 และ A1 และทิศทางการหมุนของมอเตอร์ได้ดังนี้

ทิศทางการหมุน	พอร์ต A0	พอร์ต A1
ไม่หมุน	0	0
หมุนขวา	0	1
หมุนซ้าย	1	0
ไม่หมุน	1	1

ส่วนสวิตช์ที่ใช้ในการควบคุมการหมุน เลือกใช้พอร์ต C ไบต์สูง คือ PC4 – PC7 เป็นอุปกรณ์ตรวจจับการกดเมื่อกดสวิตช์ 1-4 จะทำให้สภาวะของพอร์ต C เปลี่ยนแปลงไป ดังตารางที่ 5.1

ตารางที่ 5 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 7

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	0	1	1	1	1	1	EF
1	0	0	1	1	1	1	1	2	DF
1	0	1	1	1	1	1	1	3	BF
0	1	1	1	1	1	1	1	4	7F



รูปที่ 7.1 วงจรการเชื่อมต่อกับมอเตอร์ไฟฟ้ากระแสตรง

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรม คลิกที่ ใบงาน เลือก ใบงานที่ 7
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกที่ปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกที่ปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

.....

.....

.....

.....

.....

.....

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง
3. ทดสอบโดยการคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง บันทึกค่า
   - 3.1 ปุ่ม 1 .....
   - 3.2 ปุ่ม 2 .....
   - 3.3 ปุ่ม 3 .....
   - 3.4 ปุ่ม 4 .....
4. คลิกปุ่ม [>] เพื่อให้ทำงานปกติ พร้อมทั้งคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง บันทึกผลการทดลองดังนี้

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

.....

.....

.....

ปรับเวลาไว้ที่ 0

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

2. จงเขียนโปรแกรมให้มอเตอร์ไฟฟ้ากระแสตรง เมื่อกดสวิตช์ 1 และสวิตช์ 2 สั่งให้หยุดหมุน เมื่อกดสวิตช์ 3 สั่งให้หมุนวนทางขวา เป็นเวลาประมาณ 1 วินาที เมื่อกดสวิตช์ 4 สั่งให้หมุนวนซ้ายเป็นเวลาประมาณ 1 วินาที

ใบงานที่ 8

มอเตอร์ชนิดสเต็ป (STEPING MOTOR)

วัตถุประสงค์

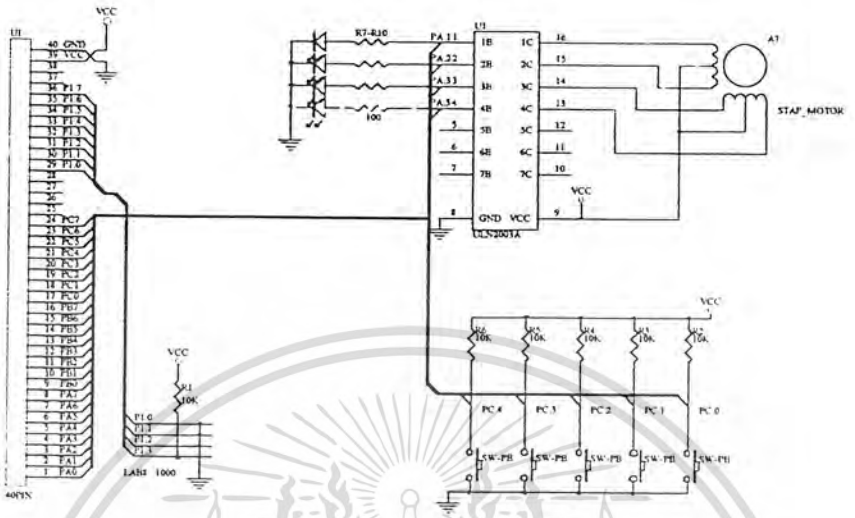
1. เพื่อให้นักศึกษาสามารถใช้งานสเต็ปปิ้งมอเตอร์ได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรสเต็ปปิ้งมอเตอร์ได้

ทฤษฎี

การทดลองที่ 8 เป็นวงจรขับมอเตอร์ชนิดสเต็ปเลือกใช้สเต็ปมอเตอร์แบบ 4 เฟส กำหนดให้พอร์ต A0–A4 ควบคุมเฟส 1–4 ของสเต็ปมอเตอร์ตามลำดับ และมีแอลอีดี เพื่อแสดงสภาวะของแต่ละเฟส ควบคุมการใช้พอร์ต C0–C4 ตรวจจับการกดสวิตช์ 1-5 โดยเมื่อกดสวิตช์ 1-5 จะเกิดสถานะของพอร์ต C ดังนี้

ตารางที่ 8.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 8

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	1	1	1	1	0	1	FE
1	1	1	1	1	1	0	1	2	FD
1	1	1	1	1	0	1	1	3	FB
1	1	1	1	0	1	1	1	4	F7
1	1	1	0	1	1	1	1	5	EF



รูปที่ 8.1 วงจรการเชื่อมต่อกับสเต็ปปิ้งมอเตอร์

ขั้นตอนการทดลอง

ตอนที่ 1

1. เปิดโปรแกรมคลิกปุ่ม ใบงาน เลือก ใบงานที่ 8
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

.....

.....

.....

.....

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)

2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง

3. ทดสอบโดยการกดปุ่มบนอุปกรณ์ต่อพ่วงจำลองบันทึกค่า

   3.1 ปุ่ม 1 .....

   3.2 ปุ่ม 2 .....

   3.3 ปุ่ม 3 .....

   3.4 ปุ่ม 4 .....

   3.5 ปุ่ม 5 .....

4. คลิกปุ่ม [>] เพื่อให้ทำงานปกติ พร้อมทั้งคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง บันทึกผลการทดลอง

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

.....

.....

.....

ปรับเวลาไว้ที่ 0

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

2. จงให้เหตุผลว่าทำไมเมื่อป้อนค่า 01H และ 04H ที่พอร์ต B ตามลำดับแล้วสเตปปิ้งมอเตอร์จึงไม่หมุน

3. จงเขียนโปรแกรมให้เสต็ปปิ้งมอเตอร์ เมื่อกดสวิตช์ 5 สั่งให้หยุดหมุน เมื่อกดสวิตช์ 4 สั่งให้หมุนวนทางซ้ายเร็ว เมื่อกดสวิตช์ 3 สั่งให้หมุนวนซ้ายช้า เมื่อกดสวิตช์ 2 สั่งให้หมุนวนทางขวาช้า เมื่อกดสวิตช์ 3 สั่งให้หมุนวนขวาเร็ว

ใบงานที่ 9

สร้างสัญญาณเสียง (SOUND)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถสร้างสัญญาเสียงจากไมโครคอนโทเลเลอได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรสร้างสัญญาณเสียงได้

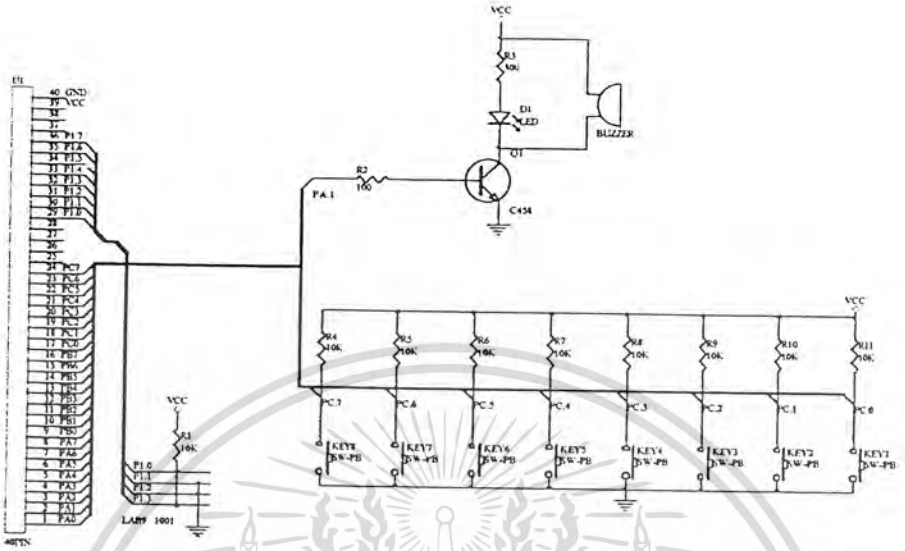
ทฤษฎี

การทดลองที่ 9 ออกแบบเพื่อแสดงการสร้างสัญญาณพัลส์ ด้วยพอร์ต A ควบคุมอัตราความถี่ของเสียงด้วยโปรแกรมและควบคุมโปรแกรมด้วยสวิตช์ 8 ตัวคือ คีย์ร์ 8 ถึง คีย์ร์ 1 เรียงลำดับจากซ้ายไปขวา ค่าที่เกิดขึ้นขณะกดสวิตช์ จะทำให้บิตของพอร์ต C เปลี่ยนสภาวะจาก "1" เป็น "0" เขียนดังตาราง 9.1ที่

ตารางที่ 9.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 9

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	1	1	1	1	0	1	FE
1	1	1	1	1	1	0	1	2	FD
1	1	1	1	1	0	1	1	3	FB
1	1	1	1	0	1	1	1	4	F7
1	1	1	0	1	1	1	1	5	EF
1	1	0	1	1	1	1	1	6	DF
1	0	1	1	1	1	1	1	7	BF
0	1	1	1	1	1	1	1	8	7F

ในส่วนของการขยายเสียงใช้ทรานซิสเตอร์ชนิดเอ็นพีเอ็น รับลอจิกไบอัสจากพอร์ต A1 ขับลำโพงชนิดเปียสโซให้เสียงดังในระดับพอประมาณ ขึ้นอยู่กับความถี่ที่ป้อนไบอัสให้กับทรานซิสเตอร์ดังแสดงในรูปที่ 9.1



รูปที่ 9.1 การเชื่อมต่อกับวงจรสร้างสัญญาณเสียง

ขั้นตอนการทดลอง

ตอนที่ 1

1. เปิดโปรแกรมคลิกปุ่ม ใบงาน เลือก ใบงานที่ 9
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

.....

.....

.....

.....

.....

สรุปผลการทดลอง

.....

.....

.....

.....

.....

.....

คำถามท้ายการทดลอง

1. จงเขียนโปรแกรม ให้สร้างเสียงความถี่ 10 กิโลเฮิรตซ์ออกทางลำโพง กำหนดความถี่ใช้งานของวงจรไมโครคอนโทรลเลอร์ไว้ที่ 12 เมกะเฮิรตซ์ ใช้ไทเมอร์ 0 ในการเขียนโปรแกรมพร้อมทั้งแสดงวิธีคำนวณ

2. เขียนวิธีคำนวณความถี่ 2 KHz

ใบงานที่ 10

อุปกรณ์ตรวจจับแสง (SENSOR and COUNTER)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถใช้งานอุปกรณ์ตรวจจับการแสงได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมอุปกรณ์ตรวจจับแสงและวงจรนับได้

ทฤษฎี

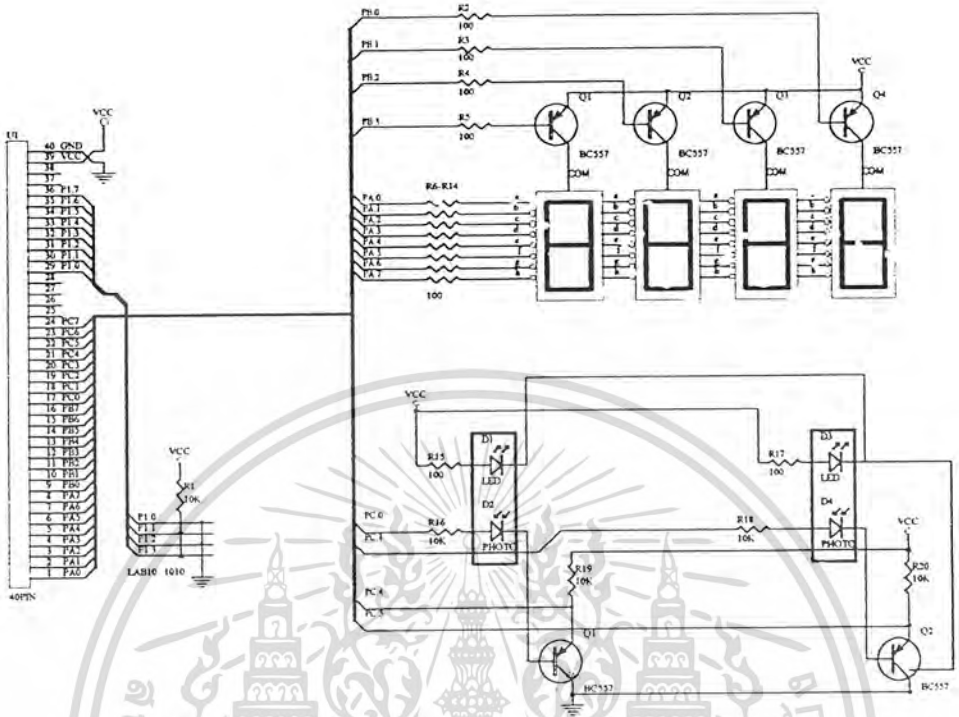
การทดลองที่ 10 เป็นการใช้งานร่วมกันกับอุปกรณ์ตรวจจับแสง ความเปลี่ยนแปลงสถานะของอุปกรณ์ตรวจจับแสงนำมาประมวลผลและแสดงออกทางตัวเลข 7 ส่วน จำนวน 4 หลัก

อุปกรณ์ตรวจจับแสงใช้พอร์ต C เป็นตัวตรวจจับสภาวะจากวงจรรูปที่ 10.1 ให้ทางภาคส่งแสงเป็นพอร์ต C0 และ C1 ส่วนตัวรับแสงใช้พอร์ต C4 และ C5 ตามลำดับ การรับค่าสภาวะของลอจิกจะรับเข้าทางพอร์ต C4 และ C5

ตารางที่ 10.1 แสดงสภาวะของพอร์ต C

PC7	PC6	PC5	PC4	PC5	PC2	PC1	PC0	ฐาน 16
1	1	1	1	1	1	0	0	FC
1	1	1	0	1	1	0	1	ED
1	1	0	1	1	1	1	0	DE
1	1	0	0	1	1	1	1	CF

แสงจากภาคส่งจะส่งออกมาได้นั้น ต้องได้รับสภาวะลอจิก "1" จากพอร์ต C0 และ C1 ทางด้านรับเมื่อได้รับแสง จะทำให้มีสภาวะทางพอร์ต C4 และพอร์ต C5 เป็น "0" ดังนั้นหากมีสิ่งกีดขวาดทางแสง สภาวะที่ได้จากพอร์ต C4 และพอร์ต C5 จะเป็น "1" ทันที จะเกิดการเปลี่ยนแปลงทางลอจิกสามารถนำการเปลี่ยนแปลงไปประมวลผลได้



รูปที่ 10.1 การเชื่อต่อกับวงจรตรวจจับแสง และวงจรนับ

ลำดับขั้นการทดลอง

1. เปิดโปรแกรมคลิกปุ่ม ใบงาน เลือก ใบงานที่ 10
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

.....

.....

.....

.....

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง

3. คลิกปุ่ม [>] เพื่อให้ทำงานปกติ พร้อมทั้งลากเม้าท์ผ่านอุปกรณ์ต่อพ่วงจำลอง บันทึกผลการทดลอง

ผลการทดลอง (ตอนที่ 2)

.....
.....
.....
.....
.....
.....
.....

สรุปผลการทดลอง

.....
.....
.....
.....
.....
.....

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

ภาคผนวก จ
เฉลยใบงาน

เฉลยใบงานที่ 1

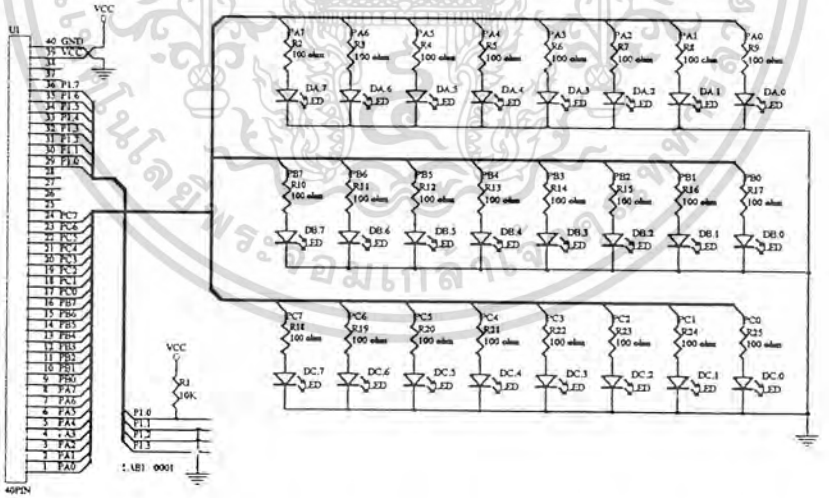
แอลอีดีลอจิก (LED LOGIC)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถ ส่งค่าออกทางพอร์ตทั้ง 3 ของวงจรแอลลีดีลอจิกได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรม เพื่อควบคุมวงจรแอลอีดีลอจิกได้

ทฤษฎี

การทดลองที่ 1 ออกแบบให้ใช้งานในการทดลองสถานะพอร์ตทั้ง 3 พอร์ตของวงจรไมโครคอนโทรลเลอร์คือ พอร์ต A, พอร์ต B และพอร์ต C แต่ละพอร์ตจะแสดงสถานะด้วยหลอดแอลอีดี ใน 1 พอร์ต มี 8 บิต ดังนั้นจึงใช้แอลอีดี 8 ดวงต่อ 1 พอร์ต ดังแสดงในวงจรรูปที่ 1.1



รูปที่ 1.1 วงจรการเชื่อมต่อกับแอลอีดีลอจิก

จากรูปที่ 1 สถานะลอจิกเท่ากับ "1" มีแรงดันเท่ากับ 4.2–4.8 โวลต์ แอลอีดีต้องการแรงดันเท่ากับ 2.2โวลต์ จึงจำเป็นต้องต่อตัวต้านทาน 100โอห์ม เพื่อจำกัดกระแสและแบ่งแรงดันให้เหมาะสมกับแอลอีดี

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรม คลิกที่ ใบงาน เลือก ใบงานที่ 1
2. เลือกหน่วยแสดงผลเป็นทำงานบนคอนโทรลเลอร์ (แสดงไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

หลอดแอลอีดี ติดสว่างทางด้านขวามือของแถวที่ 1 (บน) ติดเรียงมาจนถึงดวงสุดท้ายทางด้านซ้ายมือ และจะติดดวงแรกค้างไว้ หลังจากนั้นจึงไปเริ่มติดแถวที่ 2 (กลาง) และแถวที่ 3 ตาม (ล่าง) ตามลำดับ และวนกลับมาเริ่มแถวที่ 1 (บน) ใหม่อีกครั้งหนึ่ง

ตอนที่ 2

1. เลือกหน่วยแสดงผลเป็นทำงานบนคอมพิวเตอร์ (แสดงไอค่อนรูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง
3. เลือกการทำงานของการจำลองเป็นแบบทีละขั้น โดยการคลิกที่ปุ่ม STEP
4. คลิกการทำงานทีละขั้นที่ปุ่ม >> และบันทึกผลการทดลองในตาราง
5. เลือกการทำงานของการจำลองโดยคลิกที่ปุ่ม STEP อีกครั้ง แล้วคลิกที่คลิกปุ่ม > เข้าสู่การทำงานปกติ ทดลองปรับความเร็วที่แถบเครื่องสปีดบาร์ที่ตำแหน่งต่อไปนี้ บันทึกผล

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

โปรแกรมจะเริ่มทำงานทีละคำสั่งอย่างๆ สามารถอ่านค่าจากรีจิสเตอร์ต่างๆ ได้ แต่การแสดงของอุปกรณ์ต่อพ่วงจำลอง จะเปลี่ยนแปลงช้าด้วย

ปรับเวลาไว้ที่ 0

โปรแกรมที่ละขั้นตอนแต่เป็นไปอย่างรวดเร็วกว่าการปรับเวลาไว้ที่ 20 ค่าของรีจิสเตอร์จะแสดงรวดเร็วเช่นกัน การเปลี่ยนแปลงของอุปกรณ์ต่อพ่วงเร็วขึ้นจนสามารถสังเกตความเปลี่ยนแปลงได้

สรุปผลการทดลอง

เป็นการทดลองส่งค่าออกทางพอร์ตทั้ง 3 พอร์ต คือ พอร์ต A, พอร์ต B และ พอร์ต C สามารถเข้าถึงและควบคุมได้ในระดับบิต ให้ลอจิก "1" หลอดแอลอีดีสว่าง แต่ให้ลอจิก "0" หลอดแอลอีดีดับ

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

ตอบ. การทำงานตามคำสั่งของตัวอุปกรณ์จริงจะทำงานได้รวดเร็วกว่า การทำงานของอุปกรณ์จำลองเนื่องจากการทำงานของอุปกรณ์จำลองได้มาจากการจำลองการทำงานของ MCS-51

2. ในใบงานนี้การทำงานของ SP มีลักษณะการทำงานอย่างไร

ตอบ. จะใช้ในกรณีที่มีการกระโดดไปทำงานโปรแกรมย่อย เพื่อเป็นตัวชี้ตำแหน่งไว้เก็บตำแหน่งตรงที่กระโดดไป

3. ข้อมูลที่เปลี่ยนแปลงใน DPH และ DPL เป็นค่าของอะไร

ตอบ. เป็นค่าของตำแหน่งหน่วยความจำภายนอกหรือตำแหน่งอุปกรณ์ภายนอกโดยใช้ รีจิสเตอร์ DPTR เป็นตัวชี้ ซึ่งค่าใน DPTR เป็นข้อมูล 16 บิต ค่าใน DPH คือ ข้อมูล 8 บิต ด้านบนของ DPTR ส่วนค่าใน DPL คือ ข้อมูล 8 บิตล่างของ DPTR

4. หากต้องการเปลี่ยนทิศทางการวิ่งของแอลอีดี จะสามารถแก้ไขโปรแกรมที่ส่วนใดบ้าง

ตอบ. เปลี่ยนค่าเริ่มต้นของรีจิสเตอร์ A จาก 01H เป็น 80H และเปลี่ยนคำสั่งเลื่อนบิตในรีจิสเตอร์ A จากคำสั่ง RL A เป็น RR A

5. จงเขียนโปรแกรมเพื่อให้วงจรแอลอีดีลอจิก ติดสว่างทุกดวง และดับลง สลับไปมาเริ่มจากแถวที่1 ลงมาจนถึงแถวที่3 แล้ววนกลับไปเริ่มแถวที่1 ใหม่

ตอบ.


```
        ORG 8000H
PORT_A  EQU  0E000H
PORT_B  EQU  0E001H
```

```
PORT_C          EQU     0E002H
C_PORT          EQU     0E003H
;-----------------------------------------
SET_8255: MOV   A,#80H
          MOV   DPTR,#C_PORT
          MOVX  @DPTR,A
;-----------------------------------------
START:    MOV   DPTR,#PORT_A
          LCALL     OUT
          INC       DPTR
          LCALL     OUT
          INC       DPTR
          LCALL     OUT
          AJMP      START
;-----------------------------------------
OUT:      MOV       A,#0FFH
          MOVX  @DPTR,A
          LCALL     DELAY_SC
          CPL   A
          MOVX      @DPTR,A
          LCALL     DELAY_SC
          RET
;-----------------------------------------
DELAY_SC:MOV  R5,#0FFH
LOOP1:    MOV       R6,#0FFH
LOOP2:    NOP
          DJNZ      R6,LOOP2
          DJNZ              R5,LOOP1
          RET
;-----------------------------------------
END
```

ตารางบันทึกผลการทดลอง ที่ 1

ครั้งที่	ตำแหน่ง	รหัสฐาน 16	รหัสคำสั่ง	หน่วยความจำภายใน													หน่วยความจำภายนอก			
				Acc	DPH	DPL	PSW	SP	R0	R1	R2	R3	R4	R5	R6	R7	PA	PB	PC	Pcon
1	8000	90 E0 03	MOV DPTR,C_PORT	00	E0	03	00	07	00	00	00	00	00	00	00	00	00	00	00	00
5	8008	90 E0 00	MOV DPTR,#0E000	80	E0	00		07		03										80
10	8011	DAFC	DJNZ R2,FC	02	E0	00		07		03	08									80
15	8010	23	RLA	08	E0	00		07		03	07									80
20	800F	F0	MOVX @ DPTR, A					07		03	05									80
25	8011	DAFC	DJNZ R2,FC					07		03	03						20			80
30	8010	23	RLA					07		03	02						80			80
35	8013	A3	INC DPTR					07		03	00						01			80
40	8010	23	RLA					07		02	09						01	01		80
45	800F	F0	MOVX @ DPTR, A					07		02	07						01	04		80
50	8011	DAFC	DJNZ R2,FC					07		02	05						01	08		80
55	8010	23	RLA					07		02	04						01	20		80
60	800F	F0	MOVX @ DPTR, A					07		02	02						01	80		80
65	8011	DAFC	DJNZ R2,FC					07		02	00						01	01		80
70	800F	FO	MOVX @ DPTR, A					07		01	09						01	01	01	80
75	8011	DAFC	DJNZ R2,FC					07		01	07						01	01	02	80
80	8010	23	RLA					07		01	06						01	01	08	80
85	800F	F0	MOVX @ DPTR, A					07		01	04						01	01	20	80
90	8011	DAFC	DJNZ R2,FC					07		01	02						01	01	40	80
95	8010	23	RLA					07		01	01						01	01	01	80
100	8006		MOV R1, #3					07		03	00						01	01	01	80
105	8010	RLA	RLA					07		03	09						01	01	01	80

เฉลยใบงานที่ 2

แอลอีดีดอตเมตริกซ์ (LED DOT MATRIX)

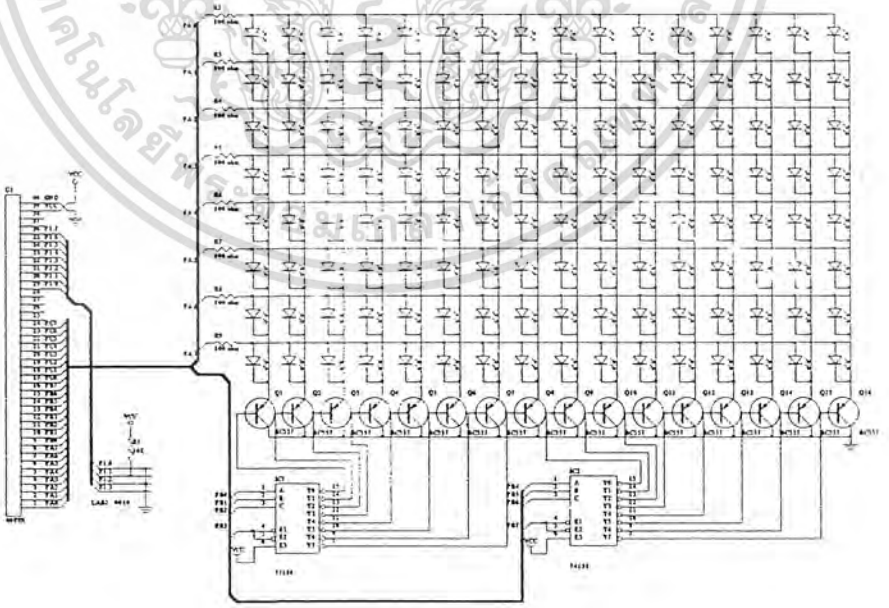
วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถถอดรหัสจากภาพ, ข้อความหรือ สัญลักษณ์ ออกเป็นข้อมูลตัวเลข ฐาน 16 ได้

2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรแอลอีดีดอตเมตริกซ์ได้

ทฤษฎี

การทดลองที่ 2 ออกแบบเพื่อใช้เป็นอุปกรณ์แสดงรูปภาพ, ข้อความ หรือสัญลักษณ์ ในการสร้างภาพด้วยจุดทั้งแนวนอนและแนวตั้งรวมกัน หมายถึง หากต้องการให้แอลอีดีดวงใดติดสว่างจำเป็นต้องทำให้ครบวงจรทั้งแนวนอน และแนวตั้ง กำหนดให้ ลอจิก "1" เป็นแรงดันทางแนวนอนและลอจิก "0" เป็นแรงดันในแนวตั้ง จึงเกิดการครบวงจร วงจรแสดงดังรูปที่ 2.1



รูปที่ 2.1 วงจรการเชื่อมต่อกับแอลอีดีดอตเมตริกส์

จากวงจรในรูปที่ 2.1 การให้ลอจิก "1" ทางแนวนอนกำหนดโดยพอร์ต A ของไมโครคอนโทรลเลอร์ ส่วนลอจิก "0" ที่ทำให้แอลอีดีติดสว่างได้จะถูกควบคุมด้วยทรานซิสเตอร์ชนิดพีเอ็นพี จะทำงานด้วยลอจิก "0" จากไอซีดีโค้ดเดอร์เบอร์ 74138 รับค่าลอจิกจากพอร์ต B ของวงจรไมโครคอนโทรลเลอร์

การควบคุมแนวนอน ถูกควบคุมด้วยพอร์ต A แบ่งออกเป็นแถว 1 แถว ต่อสภาวะ 1 บิต ของพอร์ต A มีวิธีเรียงลำดับ ดังนี้ Port A0-A7 เรียงจากแถวด้านบนคือแถวที่ 1 จนถึงแถวด้านล่างคือแถวสุดท้ายแถวที่ 8

การควบคุมทางด้านแนวตั้ง มีทั้งหมด 16 หลัก เริ่มจากหลักที่ 1 ทางด้านซ้าย เรียงลำดับไปถึงหลักที่ 16 ทางด้านขวา เพื่อให้ง่ายต่อการควบคุมจึงใช้ไอซีดีโค้ดเดอร์เบอร์ 74138 ควบคุมด้วยพอร์ต B 4 บิตล่าง สำหรับหลักที่ 1-8 และ 4 บิตบนสำหรับ หลักที่ 9-6 ดังตารางที่ 2.1

ตารางที่ 2.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 2

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	หลักที่	ฐาน 16
1	0	0	0	0	0	0	0	1	80
1	0	0	0	0	0	0	1	2	81
1	0	0	0	0	0	1	0	3	82
1	0	0	0	0	0	1	1	4	83
1	0	0	0	0	1	0	0	5	84
1	0	0	0	0	1	0	1	6	85
1	0	0	0	0	1	1	0	7	86
1	0	0	0	0	1	1	1	8	87
0	0	0	0	1	0	0	0	9	08
0	0	0	1	1	0	0	0	10	18
0	0	1	0	1	0	0	0	11	28
0	0	1	1	1	0	0	0	12	38
0	1	0	0	1	0	0	0	13	48
0	1	0	1	1	0	0	0	14	58
0	1	1	0	1	0	0	0	15	68
0	1	1	1	1	0	0	0	16	78
0	0	0	0	0	0	0	0	ไม่แสดงหลัก	00

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรมคลิกที่ปุ่ม [ใบงาน] เลือกใบงานที่ 2
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูป ไอซี)
3. คลิกปุ่ม [ส่ง] เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม [ทำงาน] เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

ที่หลอดแอลอีดีดอตเมทริกซ์แสดงตัวอักษรวิ่งเป็นคำว่า "TEPSA – 1" โดยวิ่งจากทางด้านซ้ายมาทางด้านขวามือ

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (แสดงไอค่อนรูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง
3. เลือกการทำงานของการจำลองเป็นแบบทีละขั้น โดยการคลิกที่ปุ่ม [STEP]
4. คลิกการทำงานทีละขั้นที่ปุ่ม [>>] และบันทึกผลการทดลองในตาราง
5. เลือกการทำงานของการจำลองโดยคลิกที่ปุ่ม [STEP] อีกครั้ง แล้วคลิกที่ปุ่ม [>] เพื่อเข้าสู่ การทำงานปกติ ทดลองปรับความเร็วที่แถบเครื่องมือสปีดบาร์ที่ตำแหน่งต่อไปนี้ แล้วบันทึกผล

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

โปรแกรมทำงานทีละขั้นตอน การแสดงผลของอุปกรณ์ต่อพ่วงจำลองจะแสดงทีละขั้นตอนโดยจะมีแอลอีดีติดมาทางด้านซ้ายมือทีละแถวไปจนถึงแถวที่ 8 ซึ่งแต่ละแถวจะมีติดไม่เหมือนกันยังไม่สามารถอ่านเป็นตัวอักษรได้

ปรับเวลาไว้ที่ 0

การแสดงผลของอุปกรณ์ต่อพ่วงจำลองเป็นเช่นเดียวกับในขณะที่ปรับเวลาไว้ที่ 20 แต่จะเปลี่ยนแปลงเร็ว จนสามารถคาดเดาว่าแอลอีดีดอทเมทริกซ์เป็นตัวอักษร "A"

สรุปผลการทดลอง

การแสดงผลของหลอดแอลอีดีดอทเมทริกซ์ แสดงที่อุปกรณ์ต่อพ่วงจริง นั้นในความเป็นจริงแล้วเป็นการแสดงออกทีละหลักเช่นเดียวกับที่ปรากฏในอุปกรณ์ต่อพ่วงจำลอง แต่ว่าความเร็วทำให้มองเห็นเป็นตัวอักษรได้

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

ตอบ. การแสดงอุปกรณ์จริงทำได้รวดเร็วกว่าการแสดงผลด้วยอุปกรณ์จำลอง

2. จงเขียนโปรแกรมเพื่อให้วงจรแอลอีดีดอทเมตริกซ์ แสดงข้อความ "HELLO"

ตอบ.

```
        ORG 8000H
        PORT_A  EQU     0E000H
        PORT_B  EQU     0E001H
        PORT_C  EQU     0E002H
        C_PORT  EQU     0E003H
        ;-------------------------------------------------------
        SET_8255: MOV   A,#80H          ;ออกทั้งหมด
                  MOV   DPTR,#C_PORT
                  MOVX  @DPTR,A
        ;-------------------------------------------------------
        ST0:      MOV   PSW,#00H
                  MOV   R3,#00H     ;ควบคุมรูปแบบ
        ;-------------------------------------------------------
                  MOV   DPTR,#PORT_A
                  MOV   A,#00H
                  MOVX  @DPTR,A
        DIGITA:   MOV   DPTR,#PORT_B
                  MOV   A,#0F0H
        LA:       LCALL PATTERN           ;ส่งรูปแบบ
                  MOVX  @DPTR,A
                  ACALL DELAY
                  INC   A
                  CJNE  A,#0F8H,LA
```

```
DIGITB:  MOV    A,#0FH
LB:      LCALL  PATTERN        ;ส่งรูปแบบ
         MOVX   @DPTR,A
         ACALL  DELAY
         SWAP   A
         INC    A
         SWAP   A
         CJNE   A,#8FH,LB
;-------------------------------------------------------
         AJMP   ST0
;-------------------------------------------------------
PATTERN: PUSH   DPH
         PUSH   DPL
         PUSH   ACC
         MOV    A,R3
         MOV    DPTR,#DATA
         MOVC   A,@A+DPTR
         MOV    DPTR,#PORT_A
         MOVX   @DPTR,A
         INC    R3
         POP    ACC
         POP    DPL
         POP    DPH
         RET
;-------------------------------------------------------
DATA:    DB     3EH,08H,3EH    ;H
         DB     3EH,2AH,22H    ;E
         DB     3EH,20H,20H    ;L
         DB     3EH,20H,20H    ;L
         DB     3EH,22H,3EH,00H ;O
;-------------------------------------------------------
DELAY:   MOV    R4,#10H
LOOP2:   MOV    R5,#10H
LOOP1:   NOP
         DJNZ   R5,LOOP1
         DJNZ   R4,LOOP2
         RET
;-------------------------------------------------------
         END
```

ตารางบันทึกผลการทดลอง ที่ 2/1

ครั้งที่	ตำแหน่ง	รหัสฐาน 16	รหัสคำสั่ง	หน่วยความจำภายใน													หน่วยความจำภายนอก			
				Acc	DPH	DPL	PSW	SP	R0	R1	R2	R3	R4	R5	R6	R7	PA	PB	PC	Pcon
1	8000	90 E0 03	MOV DPTR,C_PORT	0 0	E0	0 3	0 0	0 7	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
5	8008	74 80	MOV A, # 80	8 0	E0	0 3		0 7												8 0
10	8013	C0 83	PUSH 83	8 0	E0	0 1		0 B										80		8 0
15	8025	93	MOVC A, @A+DPTR	F8	80	32		0 D										80		8 0
20	802D	D0 82	POP 82	80	E0	0 1		0 B				0 1					F8	80		8 0
25	8016	B4 88 F1	CJNE A,#88,F1	81	E0	0 1		0 7				0 1					F8	80		8 0
30	801B	C0 83	PUSH 83	81	E0	0 1		0 B				0 1					F8	81		8 0
35	8025	93	MOVC A, @A+DPTR	FC	80	32		0 D				0 1					F8	81		8 0
40	802D	D0 82	POP 82	82	E0	0 1		0 B				0 2					FC	81		8 0
45	8016	B4 88 F1	CJNE A,#88,F1	82	E0	0 1		0 7				0 2					FC	81		8 0
50	801B	C0 83	PUSH 83	82	E0	0 1		0 B				0 2					FC	82		8 0
55	8025	93	MOVC A, @A+DPTR	36	80	32		0 D				0 2					FC	82		8 0
60	802D	D0 D3282	POP 82	82	E0	0 1		0 B				0 3					36	82		8 0
65	8016	B4 88 F1	CJNE A,#88,F1	83	E0	0 1		0 7				0 3					36	82		8 0
70	801B	C083	PUSH 83	83	E0	0 1		0 B				0 3					36	83		8 0
75	8025	93	MOVC A, @A+DPTR	33	80	32		0 D				0 3					36	83		8 0
80	802D	D082	POP 82	83	E0	0 1		0 B				0 4					33	83		8 0
85	8016	B4 88 F1	CJNE A,#88,F1	84	E0	0 1		0 7				0 4					33	83		8 0
90	801B	C0 83	PUSH 83	84	E0	0 1		0 B				0 4					33	84		8 0
95	8025	93	MOVC A, @A+DPTR	36	80	32		0 D				0 4					33	84		8 0
100	802D	D0 82	POP 82	84	E0	0 1		0 B				0 5					36	84		8 0
105	8016	B4 88 F1	CJNE A,#88,F1	85	E0	0 1		0 7				0 5					36	84		8 0
110	801B	C0 83	PUSH 83	85	E0	0 1		0 B				0 5					36	85		80
115	8025	93	MOVC A, @A+DPTR	FC	80	32		0 D				0 5					36	85		80
120	802D	D0 82	POP 82	85	E0	0 1		0 B				0 6					FC	85		80

ตารางบันทึกผลการทดลอง ที่ 2/2

ครั้งที่	ตำแหน่ง	รหัสฐาน 16	รหัสคำสั่ง	หน่วยความจำภายใน												หน่วยความจำภายนอก			
				Acc	DPH	DPL	PSW	SP	R0	R1	R2	R3	R4	R5	R6	R7	PA	PB	PC	Pcon
125	8016	B4 88 F1	CJNE A,#88,F1	86	E0	0 1						0 6					36	85		80
130	801B	C0 83	PUSH 83	86	E0	0 1						0 6					FC	85		80
135	8025	93	MOVC A ,@ A+DPTR	F8	E0	0 1						0 6					FC	86		80
140	802D	D0 82	POP 82	86	E0	0 1						0 7					FC	86		80
145	8016	B4 88 F1	CJNE A,#88,F1	87	E0	0 1						0 7					F8	86		80
150	801B	C0 83	PUSH 83	87	E0	0 1						0 7					F8	86		80
155	8025	93	MOVC A ,@ A+DPTR	0 0	80	32						0 7					F8	87		80
160	802D	D0 82	POP 82	87	E0	0 1						0 8					0 0	87		80
164	8016	B4 88 F1	CJNE A,#88,F1	88	E0	0 1						0 8					0 0	87		80
170	800D	F0	MOVC A ,@ A+DPTR	80	E0	0 1	81					0 0					0 0	87		80
175	801F	C0 E0	PUSH E0	80	E0	0 1	81					0 0					0 0	80		80
180	8029	F0	MOVX @ DPTR, A	F8	E0	0 0	81					0 0					F8	80		80

เฉลยใบงานที่ 3

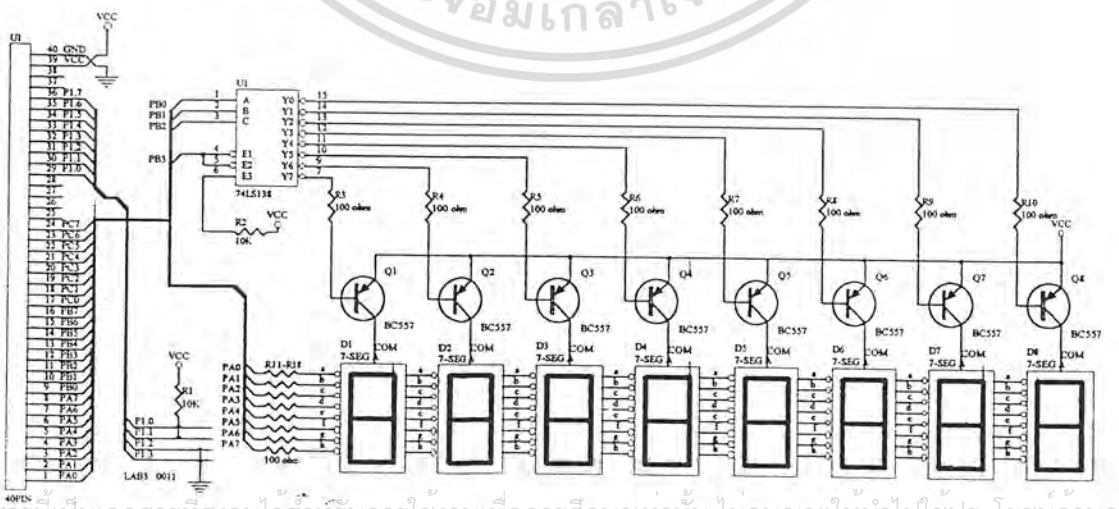
จอแสดงผลแบบ 7 ส่วน (SEVEN-SEGMANT)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถใช้งานจอแสดงผลแบบ 7 ส่วน แบบขั้วบวกร่วมกันได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรเชื่อมต่อกับจอแสดงผลแบบ 7 ส่วนได้

ทฤษฎี

การทดลองที่ 3 เป็นการทดลองใช้งานจอแสดงผลแบบ 7 ส่วน แบบมีขั้วบวกร่วมกัน ดังนั้นเมื่อจะทำให้แอลอีดีของจอแสดงผลแบบ 7 ส่วน ส่วนใดติดสว่างต้องให้ส่วนนั้นได้รับลอจิก "0" สำหรับการออกแบบวงจรการทดลองที่ 3 นี้ ใช้ จอแสดงผลแบบ 7 ส่วนจำนวน 8 ตัวแอลอีดีแต่ละส่วนของทุกตัวเลข จะต่อถึงกันและควบคุมด้วยพอร์ต A ของวงจรไมโครคอนโทรลเลอร์ หากต้องการให้ส่วนใดของตัวเลขติดสว่าง ก็ให้ลอจิกนั้นมีสถานะเป็น "0" แต่จะติดสว่างที่ตัวเลขหลักใดนั้นขึ้นอยู่กับจะให้หลักใดครบวงจร ในวงจรนี้เลือกใช้ทรานซิสเตอร์ชนิด PNP เป็นอุปกรณ์สวิตช์โดยควบคุมการเลือกหลักของตัวเลขด้วยพอร์ต B ผ่านทางไอซีดีโค้ดเดอร์เบอร์ 74138 ใช้พอร์ต B เพียง 4 เส้นในการควบคุมหลักของจอแสดงผลแบบ 7 ส่วน เลือกใช้พอร์ต B ไบต์ต่ำเพื่อควบคุมหลักที่ใช้แสดงผล เริ่มต้นหลักที่ 1 ทางด้านขวา จนถึงหลักที่ 8 ทางด้านซ้าย



รูปที่ 3.1 วงจรการเชื่อมต่อกับจอแสดงผลแบบ 7 ส่วน

ตารางที่ 3.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 3

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	หลักที่	ฐาน 16
X	X	X	X	0	0	0	0	1	X0
X	X	X	X	0	0	0	1	2	X1
X	X	X	X	0	0	1	0	3	X2
X	X	X	X	0	0	1	1	4	X3
X	X	X	X	0	1	0	0	5	X4
X	X	X	X	0	1	0	1	6	X5
X	X	X	X	0	1	1	0	7	X6
X	X	X	X	0	1	1	1	8	X7
X	X	X	X	1	0	0	0	ไม่แสดงหลัก	-

หมายเหตุ : X หมายถึงค่าใดๆ

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรมคลิกที่ ใบงาน เลือก ใบงานที่ 3
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

ที่อุปกรณ์จอแสดงผลตัวเลข 7 ส่วนจะแสดงตัวเลขทีละค่าและทีละหลัก เริ่มจากหลักแรกทางด้านขวามือโดยแสดงตัวเลข "0" ไปจนถึงหลักสุดท้ายทางด้านซ้ายมือ โดยแสดงตัวเลข "7" เรียงตามลำดับกันไป

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)

2. ส่งรหัสคำสั่งไปยังหน่วยลวามจำโปรแกรมจำลอง

3. เลือกการทำงานของการจำลองเป็นแบบทีละขั้น โดยการคลิกที่ปุ่ม STEP

4. คลิกการทำงานทีละขั้นที่ปุ่ม >> และบันทึกผลการทดลองในตาราง

5. เลือกการทำงานของการจำลองเป็นแบบธรรมดา โดยคลิกที่ปุ่ม STEP อีกครั้ง แล้วคลิกที่ปุ่ม > เข้าสู่การทำงานปกติ ทดลองปรับความเร็วที่แถบเครื่องมือสปีดบาร์ที่ตำแหน่งต่อไปนี้ แล้วบันทึกผล

ผลการทดลอง (ตอนที่ 2)

ปรับความเร็วไว้ที่ 20

ที่อุปกรณ์จอแสดงผลตัวเลข 7 ส่วนจำลองจะแสดงค่าตัวเลขได้เหมือนกับอุปกรณ์จริง เพียงแต่ทำงานได้ช้ากว่า คือจะทำงานทีละขั้นตอน จนเห็นการเปลี่ยนแปลงของตัวเลขและหลักได้ชัดเจน

ปรับความเร็วไว้ที่ 0

ได้ผลเช่นเดียวกับการแสดงผลตอนปรับความเร็วไว้ที่ 20 แต่ความเร็วในการแสดงผลจะเร็วกว่า

สรุปผลการทดลอง

การแสดงผลของตัวเลข 7 ส่วนมากกว่า 1 หลักไม่สามารถแสดงได้พร้อมกันทุก ๆ หลักแต่สามารถทำให้เห็นว่าแสดงพร้อมกันได้ด้วยการอาศัยความเร็วในการแสดงผลของไมโครคอนโทรลเลอร์

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

ตอบ. อุปกรณ์จริงแสดงผลได้เร็วกว่าอุปกรณ์จำลอง

2. จงเขียนโปรแกรมที่ทำให้จอแสดงผลแบบ 7 ส่วน แสดงตัวเลข 0-7 ที่หลัก 0 ถึงหลัก 7 เรียงตามลำดับ

ตอบ.

```
        ORG  8000H
PA      EQU  0E000H
PB      EQU  0E001H
CP      EQU  0E003H
;*************************************
SET_8255: MOV  DPTR,#CP
        MOV  A,#80H
        MOVX @DPTR,A
;*************************************
ST:     MOV   R1,#00H
        MOV   R2,#00H
OUT:    MOV   A,R1
        MOV   DPTR,#DATA
        MOVC  A,@A+DPTR
        MOV   DPTR,#PA
        MOVX  @DPTR,A
        INC   R1
        MOV   A,R2
        MOV   DPTR,#PB
        MOVX  @DPTR,A
        INC   R2
        ACALL DELAY
        CJNE  R2,#08H,OUT
        JMP   ST
;*************************************
DATA:   DB    0C0H,0F9H,0A4H,0B0H
        DB    99H,92H,82H,0F8H
        DB    80H,90H,88H,83H
        DB    0C6H,0A1H,86H,8EH
;*************************************
DELAY:  MOV R5,#10H
LOOP2:  MOV R6,#05H
LOOP1:  NOP
        NOP
```

```
        DJNZ R6,LOOP1
        DJNZ R5,LOOP2
        RET
;.....

        END
```

3. ในโปรแกรมการทดลองนี้ค่า Carry Flag ใน PSW จะเปลี่ยนแปลงเป็น 1 เมื่อใด

ตอบ. เปลี่ยนแปลงเมื่ออ่านคำสั่ง CJNE แล้ว ค่าในรีจิสเตอร์ R1 เป็น 8

ตารางบันทึกผลการทดลอง ที่ 3/1

ครั้งที่	ตำแหน่ง	รหัสฐาน 16	รหัสคำสั่ง	หน่วยความจำภายใน													หน่วยความจำภายนอก			
				Acc	DPH	DPL	PSW	SP	R0	R1	R2	R3	R4	R5	R6	R7	PA	PB	PC	Pcon
1	8000	90 E0 03	MOV DPTR,C_PORT	0 0	E0	0 3	0 0	0 7	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
5	800B	74 80	MOVX @ DPTR, A	8 0	E0	0 1		0 7										80		8 0
10	801C	C0 83	MOV DPTR,# 802C	8 0	80	2C		0C										80		8 0
15	8025	93	POP E0	80	E0	0 0		0B									C0	80		8 0
20	8010	D0 82	INC R1	81	E 0	0 1		0 7		0 1							C0	80		8 0
25	8018	B4 88 F1	PUSH 82	81	E0	0 1		0B		0 1							C0	81		8 0
30	8021	C0 83	MOV DPTR,# E0 00	F 9	E 0	0 0		0C		0 1							C0	81		8 0
35	802B	93	RET	FC	80	32		0 7		0 1							F9	81		8 0
40	800C	D0 82	LCALL 8016	82	E0	0 1		0 9		0 2							F9	82		8 0
45	801F	B4 88 F1	MOV A, R1	82	E0	0 1		0C		0 2							F9	82		8 0
50	8027	C0 83	POP 82	82	E0	0 1		0A		0 2							A4	82		8 0
55	8011	93	CJNE R1,# 8, F7	36	80	32		0 7		0 3							A4	82		8 0
60	801A	D0 D3282	PUSH E0	82	E0	0 1		0C		0 3							A4	83		8 0
65	8024	B4 88 F1	MOVX @ DPTR, A	83	E0	0 1		0C		0 3							B0	83		8 0
70	800F	C083	INC A	83	E0	0 1		0 7		0 3							B0	83		8 0
75	8016	93	PUSH 83	33	80	32		0A		0 4							B0	84		8 0
80	8020	D082	MOVC A, @A+DPTR	83	E0	0 1		0C		0 4							B0	84		8 0
85	8029	B4 88 F1	POP 83	84	E0	0 1		0 9		0 4							99	84		8 0
90	800B	C0 83	MOVX @ DPTR, A	84	E0	0 1		0 B		0 5							99	85		8 0
95	801C	93	MOV DPTR,# 802C	36	80	32		0 7		0 5							99	85		8 0
100	8025	C0 E0	POP E0	85	E0	0 0		0C		0 5							92	85		8 0
105	8010	0 9	INC R1	86	E0	0 1		0 7		0 6							92	85		8 0
110	8018	C0 82	PUSH 82	86	E0	0 1		0 B		0 6							92	86		80
115	8021	90 E0 00	MOV DPTR,#E0 00	82	E0	0 0		0 C		0 6							92	86		80
120	802B	22	RET	86	E0	0 1		0 7		0 6							82	86		80

ตารางบันทึกผลการทดลอง ที่ 3/2

ครั้งที่	ตำแหน่ง	รหัสฐาน 16	รหัสคำสั่ง	หน่วยความจำภายใน													หน่วยความจำภายนอก			
				Acc	DPH	DPL	PSW	SP	R0	R1	R2	R3	R4	R5	R6	R7	PA	PB	PC	Pcon
125	800C	12 80 16	LCALL 8016	87	E0	2 C		0 9		0 7							82	87		80
130	801F	E9	MOV A,R1	0 7	80	0 1		0C		0 7							82	87		80
135	8027	C0 82	POP 82	87	E0	0 1		0A		0 7							F8	87		80
140	8011	B9 08 F7	CJNE R1, #8,F7	88	E0	0 1		0 7		0 8							F8	87		80
145	800C	12 80 16	LCALL 8016	88	E0	0 1		0 9		0 0							F8	88		80
150	801F	E9	MOV A,R1	0	80	2 C	81	0 C		0 0							F8	88		80
155	8027	D0 82	POP 82	88	E0	0 1	81	0A		0 0							C 0	88		80
160	8011	B9 08 F7	CJNE R1, #8,F7	89	E0	0 1	81	0 7		0 1							C 0	88		80

เฉลยใบงานที่ 4

สวิตช์เมตริกซ์ (SWITCH MATRIX)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถถอดรหัส เมื่อมีการกดสวิตช์ แบบคูณหลักได้
2. เพื่อให้นักศึกษาสมารถเขียนโปรแกรมควบคุมวงจรสวิตช์เมตริกส์ ได้

ทฤษฎี

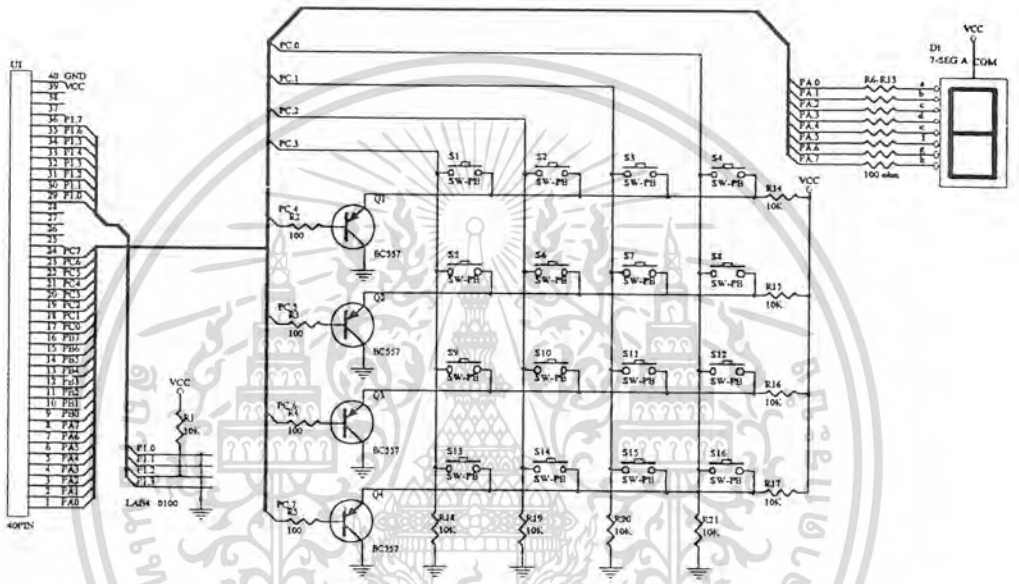
การทำงานของสวิตช์ ที่ต่อใช้งานแบบแถวคูณหลัก จะควบคุมและตรวจสอบค่าด้วยสภาวะของพอร์ต C สภาวะเริ่มต้นกำหนดให้ค่าพอร์ตทุกบิตเป็น "1" และใช้การทดสอบทีละ 1 แถว โดยกำหนดค่าออกด้วยพอร์ต C ไบต์สูง คือ PC4–PC7 และรับค่าเข้าเพื่อพิจารณาค่าด้วยพอร์ต C ไบต์ต่ำ คือ PC0–PC3 คือทดสอบทีละ 1 แถวนั่นเอง 1 แถวพิจารณาสวิตช์ 4 ตัว ดังนั้นสวิตช์ 4 แถว ทดสอบสวิตช์ได้ทั้งหมด 16 ตัว 16 กรณี สามารถเขียนได้ดังตารางที่ 3.1

ตารางที่ 3.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 4

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์ที่	ฐาน 16
1	1	1	0	1	1	1	0	0	EE
1	1	1	0	1	1	0	1	1	ED
1	1	1	0	1	0	1	1	2	EB
1	1	1	0	0	1	1	1	3	E7
1	1	0	1	1	1	1	0	4	DE
1	1	0	1	1	1	0	1	5	DD
1	1	0	1	1	0	1	1	6	DB
1	1	0	1	0	1	1	1	7	D7
1	0	1	1	1	1	1	0	8	BA
1	0	1	1	1	1	0	1	9	BD
1	0	1	1	1	0	1	1	A	BB
1	0	1	1	0	1	1	1	B	B7
0	1	1	1	1	1	1	0	C	7E
0	1	1	1	1	1	0	1	D	7D
0	1	1	1	1	0	1	1	E	7B
0	1	1	1	0	1	1	1	F	77

กรณีทั้ง 16 กรณีจะเกิดขึ้น เมื่อมีการกดสวิตช์พอร์ต C ไบต์ต่ำ จะเปลี่ยนสภาวะจาก "1" เป็น "0" ส่วนพอร์ต C ไบต์สูง จะกำหนดว่าจะพิจารณาแถวใดก็ให้แถวนั้นมีสภาวะเป็น "0"

ในวงจรการทดลองนี้ได้ใช้จอแสดงผลแบบ 7 ส่วน เพื่อใช้แสดงตำแหน่งของสวิตช์ที่ถูกกด จอแสดงผลแบบ 7 ส่วน ถูกควบคุมด้วยสภาวะของพอร์ต A คือต้องการให้ส่วนของตัวเลขติดสว่าง ก็ให้บิตของตัวเลขส่วนนั้นมีสภาวะเป็น "0" วงจรการเชื่อมต่อแสดงดังรูปที่ 4.1



รูปที่ 4.1 วงจรการเชื่อมต่อกับสวิตช์เมตริกซ์

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรมคลิกที่ ใบงาน เลือก ใบงานที่ 4
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

เมื่อกดปุ่มใด จะแสดงค่าปุ่มนั้น ออกทางจอแสดงผล 7 ส่วน สำหรับสวิตช์ที่ 10 จนถึง 15 จะแสดงด้วยตัวเลขฐาน 16 คือ A จนถึง F ตามลำดับ

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง
3. ทดสอบโดยการคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง บันทึกค่า
   - 3.1 ปุ่ม 1 ค่าใน Port C ของ 8255 มีค่าเป็น E7
   - 3.2 ปุ่ม 2 ค่าใน Port C ของ 8255 มีค่าเป็น EB
   - 3.3 ปุ่ม 3 ค่าใน Port C ของ 8255 มีค่าเป็น ED
   - 3.4 ปุ่ม 4 ค่าใน Port C ของ 8255 มีค่าเป็น EE
   - 3.5 ปุ่ม 5 ค่าใน Port C ของ 8255 มีค่าเป็น D7
   - 3.6 ปุ่ม 6 ค่าใน Port C ของ 8255 มีค่าเป็น DB
   - 3.7 ปุ่ม 7 ค่าใน Port C ของ 8255 มีค่าเป็น DD
   - 3.8 ปุ่ม 8 ค่าใน Port C ของ 8255 มีค่าเป็น DE
   - 3.9 ปุ่ม 9 ค่าใน Port C ของ 8255 มีค่าเป็น B7
   - 3.10 ปุ่ม 10 ค่าใน Port C ของ 8255 มีค่าเป็น BB
   - 3.11 ปุ่ม 11 ค่าใน Port C ของ 8255 มีค่าเป็น BD
   - 3.12 ปุ่ม 12 ค่าใน Port C ของ 8255 มีค่าเป็น BE
   - 3.13 ปุ่ม 13 ค่าใน Port C ของ 8255 มีค่าเป็น 77
   - 3.14 ปุ่ม 14 ค่าใน Port C ของ 8255 มีค่าเป็น 7B
   - 3.15 ปุ่ม 15 ค่าใน Port C ของ 8255 มีค่าเป็น 7D
   - 3.16 ปุ่ม 16 ค่าใน Port C ของ 8255 มีค่าเป็น 7E
4. คลิกปุ่ม [>] เพื่อกลับสู่การทำงานปกติ พร้อมทั้งคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง แล้วบันทึกผลการทดลอง

ผลการทดลอง (ตอนที่ 2)

การแสดงผลและการตอบสนองมีความเหมือนกับอุปกรณ์จริงเพียงแต่เวลาในการรับค่าและการแสดงผลจะช้ากว่าอุปกรณ์จริงอยู่มาก และต้องใช้จังหวะในการกดปุ่มบนอุปกรณ์จำลอง

สรุปผลการทดลอง

สวิตช์เมทริกซ์สามารถเขียนโปรแกรมควบคุมการสแกนได้ แต่จะต้องสแกนทีละ 1 แถวเท่านั้น แต่ด้วยความเร็วในการประมวลผลของไมโครคอนโทรลเลอร์จึงสามารถสแกนได้ทั้ง 16 สวิตช์ ในเวลาที่ใกล้เคียงกัน แต่จะมีผลเวลาที่ใช้กับอุปกรณ์ต่อพ่วงจำลอง คือการสแกนจะไม่สามารถสแกน ทั้ง 16 สวิตช์ ในเวลาเดียวกันได้จึงต้องอาศัยจังหวะในการกด

คำถามท้ายการทดลอง

1. จงเขียนโปรแกรมแสดงค่าการคลิกสวิตช์ โดยกำหนด สวิตช์ 1 เป็นค่า A สวิตช์ 5 มีค่าเป็น B สวิตช์ 9 มีค่าเป็น D

ตอบ.

```
                ORG  8000H
PORT_A    EQU  0E000H
PORT_C    EQU  0E002H
CP        EQU  0E003H
;-------------------------------------------------------
SET_8255: MOV  DPTR,#CP
          MOV  A,#81H
          MOVX @DPTR,A
;-------------------------------------------------------
          MOV   DPTR,#PORT_A
          MOV   A,#0FFH
          MOVX  @DPTR,A
SCAN1:    MOV   DPTR,#PORT_C
          MOV   A,#0EFH
          MOVX  @DPTR,A
          MOVX  A,@DPTR

SW1:      CJNE  A,#0EBH,SCAN2
          MOV   R1,#0AH
          AJMP  OUT
```

```
SCAN2:   MOV   DPTR,#PORT_C
         MOV   A,#0DFH
         MOVX  @DPTR,A
         MOVX  A,@DPTR
SW5:     CJNE  A,#0DBH,SCAN3
         MOV   R1,#0BH
         AJMP  OUT
SCAN3:   MOV   DPTR,#PORT_C
         MOV   A,#0AFH
         MOVX  @DPTR,A
         MOVX  A,@DPTR
SW9:     CJNE  A,#0ABH,SW0
         MOV   R1,#0DH
         AJMP  OUT
SW0:     AJMP  SCAN1
;*************************************************
OUT:     MOV   DPTR,#DATA
         MOV   A,R1
         MOVC  A,@A+DPTR
         MOV   DPTR,#PORT_A
         MOVX  @DPTR,A
         AJMP  SCAN1
DATA:    DB    0C0H,0F9H,0A4H,0B0H
         DB    99H,92H,82H,0F8H
         DB    80H,90H,88H,83H
         DB    0C6H,0A1H,86H,8EH
;*************************************
DELAY:   MOV R5,#30H
LOOP3:   MOV R6,#20H
LOOP2:   MOV R7,#20H
LOOP1:   NOP
         NOP
         DJNZ R7,LOOP1
         DJNZ R6,LOOP2
         DJNZ R5,LOOP3
         RET
;*************************************
         END
```

2. เหตุใดเมื่อกดสวิตช์ ในตัวอุปกรณ์จำลองแล้วจึงมีการตอบสนองช้า

ตอบ. เพราะการประมวลผลของอุปกรณ์จำลอง ทำงานที่ละขั้นตอนด้วยเวลาที่นานกว่าการทำงานจริงมาก

3. จงบอกวิธีที่สามารถทำให้ตัวแสดง 7 ส่วนเปลี่ยนแปลงได้รวดเร็วขึ้นเมื่อกดสวิตช์ใดสวิตช์หนึ่ง บนอุปกรณ์จำลอง

ตอบ. พัฒนาโปรแกรมในการสแกนสวิตช์ให้สั้นลงอีกจึงจะทำให้การจำลองการทำงานดูเหมือนอุปกรณ์จริงมากที่สุด

เฉลยใบงานที่ 5

จอแสดงผลชนิดผลึกเหลว (LCD)

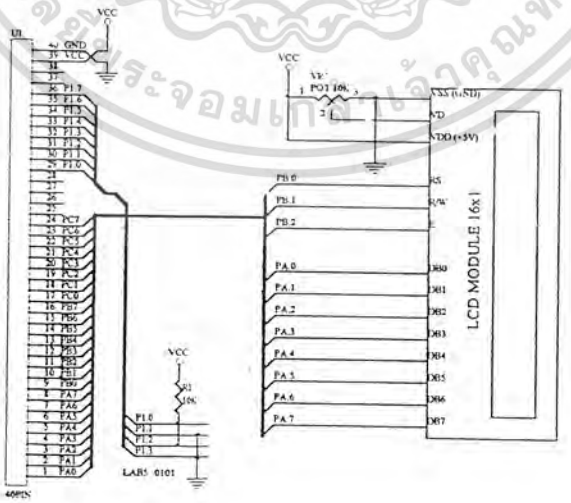
วัตถุประสงค์

1. เพื่อให้นักศึกษาสารมารถสั่งงานวงจรการเชื่อมต่อกับจอผลแบบผลึกเหลวขนาด 16 x1 ได้

2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมการแสดงผล บนจอแสดงผล ชนิดผลึกเหลว ได้

ทฤษฎี

การทดลองที่ 5 เป็นการทดลองการใช้งานอุปกรณ์แสดงผลชนิดผลึกเหลวซึ่งการควบคุมประกอบด้วย 14 ขา ควบคุมและส่งข้อมูลขาควบคุมการเขียนการอ่านข้อมูล RS, R/Z และ E ควบคุมด้วยพอร์ต B คือ B0, B1 และ B2 ตามลำดับ ส่วนขาควบคุมการส่งข้อมูล DB0–DB7 ควบคุมด้วยพอร์ต A0–A7 ตามลำดับ ดังแสดงในรูปที่ 5.1



รูปที่ 5.1 วงจรการเชื่อมต่อกับอุปกรณ์แสดงผลชนิดผลึกเหลว

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม คลิกที่ ใบงาน เลือก ใบงานที่ 5
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูป ไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง

ผลการทดลอง

จอแสดงผลผลึกเหลวแสดงข้อความ ดังนี้ "!!!HELLO!!!", "LAB5", "LCD DOT MATRIX", "MCS-51 PROJECT", "BY", "TEPSA-1" โดยจะแสดงสลับกันไปเรื่อยๆ จนครบทั้งหมดและวนกลับมาแสดงอีก

สรุปผลการทดลอง

จอแสดงผลชนิดผลึกเหลว แบบ 1 แถว 16 ตัวอักษร สามารถควบคุมการทำงานและการแสดงข้อความในแต่ละครั้งได้ แต่ในจำนวนไม่เกิน 16 ตัวอักษร

คำถามท้ายการทดลอง

1. จงเขียนโปรแกรมแสดงข้อความ "THAILAND"

ตอบ.

```
        ORG  8000H
PORT_A  EQU 0E000H
PORT_B  EQU 0E001H
C_PORT  EQU 0E003H
;-------------------------------------------------------
SET:    MOV  PSW,#00H
        ACALL DELAY    ;WAIT FOR 15mS
;-------------------------------------------------------
SET_8255: MOV  DPTR,#C_PORT
        MOV  A,#80H
        MOVX @DPTR,A   ;ออกทั้งหมด
;-------------------------------------------------------
SET_LCD: ACALL  INTI
;-------------------------------------------------------
```

```
LOOP:    MOV  DPTR,#DATA
         ACALL SWORD      ;WRITE
         ACALL DELX       ;DELAY
         LJMP LOOP
;*************************************************
DELX:  MOV  R1,#0FFH
DEL1:  MOV  R2,#0FFH
DEL2:  MOV  R3,#07H
DEL3:  DJNZ R3,DEL3
       DJNZ R2,DEL2
       DJNZ R1,DEL1
       RET
;*************************************************
DATA:  DB   "  THAILAND  "
;*************************************************
SWORD:        MOV     R6,#01H
        ACALL   LCODE
        MOVX    A,@DPTR     ;CLEAR DISPLAY
        MOV     R5,#00H
SWORD1: MOV     R0,A
        ACALL   WRBYTE
        INC     DPTR
        INC     R5
        MOVX    A,@DPTR
        CJNE    R5,#16,SWORD2
        RET
SWORD2: CJNE    R5,#8,SWORD1
        MOV     R6,#0C0H
        ACALL   LCODE
        SJMP    SWORD1
;*************************************************
WRBYTE: PUSH    DPL
          PUSH  DPH
         PUSH   ACC
         MOV    A,#01H
         MOV    DPTR,#PORT_B
         MOVX   @DPTR,A     ;CLEAR DISPLAY
         MOV    A,R0
         MOV    DPTR,#PORT_A
```

```
        MOVX    @DPTR,A     ;GET DATA
        ACALL   PLUSE
        POP     ACC
        POP     DPH
        POP     DPL
        RET
;-------------------------------------------------------
INTI:   MOV     A,#00H
        MOV     DPTR,#PORT_B
        MOVX    @DPTR,A
        MOV     R6,#38H     ;FUNCTION SET 8 BIT/5X7
        ACALL   LCODE
        ACALL   DELAY
        MOV     R6,#0CH     ;DISPLAY ON/OFF
        ACALL   LCODE
        MOV     R6,#01H     ;CLEAR DISPLAY
        ACALL   LCODE
        ACALL   DELAY
        RET
;-------------------------------------------------------
LCODE:  PUSH    DPH
        PUSH    DPL
        PUSH    ACC
        CLR     A
        MOV     DPTR,#PORT_B
        MOVX    @DPTR,A
        MOV     A,R6
        MOV     DPTR,#PORT_A
        MOVX    @DPTR,A
        ACALL   PLUSE
        POP     ACC
        POP     DPL
        POP     DPH
        RET
;-------------------------------------------------------
PLUSE:  MOV     DPTR,#PORT_B    ;PLUSE FOR ENABLE
        MOVX    A,@DPTR
        SETB    ACC.2       ;ENABLE=1
        MOV     DPTR,#PORT_B
```

```
        MOVX    @DPTR,A
        ACALL   DELAY
        CLR     ACC.2       ;ENABLE=0
        MOV     DPTR,#PORT_B
        MOVX    @DPTR,A
        RET
;*************************************************************
DELAY: MOV      R1,#50H     ;DELAY TIME
DEL:    MOV     R2,#00H
DEL0:   DJNZ    R2,DEL0
        DJNZ    R1,DEL
        RET
;*************************************************************
END
```

เฉลยใบงานที่ 6

สวิตซ์ควบคุมการทำงาน (SWITCH CONTROL)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถสั่งงานวงจรควบคุมรีเลย์ได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรสวิตซ์ควบคุมการทำงานได้

ทฤษฎี

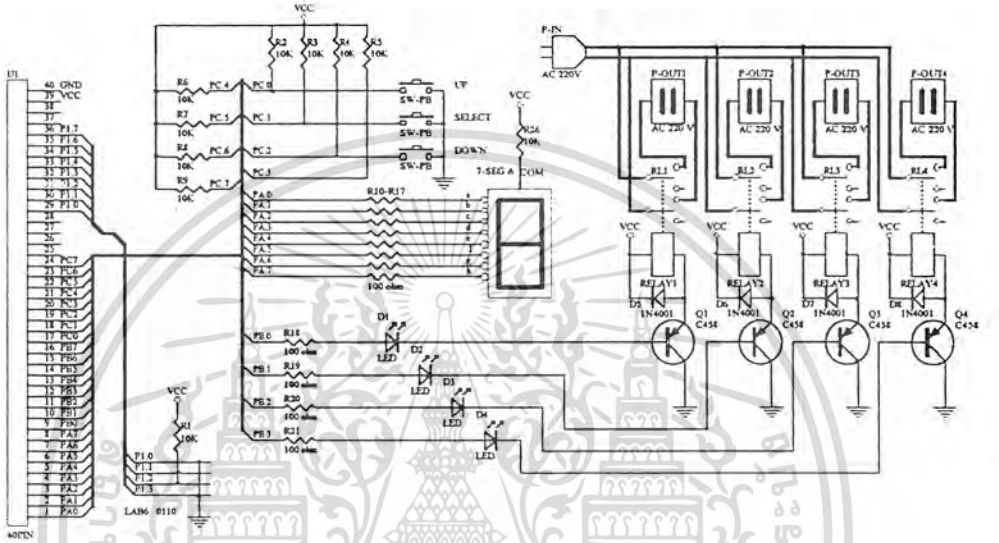
การทดลองที่ 6 ออกแบบเพื่อควบคุมการทำงานของรีเลย์ โดยอาศัยสภาวะของพอร์ต B เป็นตัวกำหนดรีเลย์ทำงาน เมื่อได้รับสภาวะลอจิก "1" ผ่านแอลอีดีแสดงผล ใช้รีเลย์จำนวน 4 ตัว รีเลย์ 1–4 ควบคุมด้วยพอร์ต B0–B3 ตามลำดับ

สวิตช์ควบคุมสำหรับวงจรทดลองนี้ มี 3 สวิตช์ ถูกกำหนดและตรวจสอบด้วยพอร์ต C เมื่อกดสวิตช์ต่างๆ จะมีค่าที่พอร์ต C ดังตารางที่ 6.1

ตารางที่ 6.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 6

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	1	1	1	1	0	1	FE
1	1	1	1	1	1	0	1	2	FD
1	1	1	1	1	0	1	1	3	FB

นอกจากนั้นยังใช้จอแสดงผลแบบ 7 ส่วน เป็นอุปกรณ์แสดงค่าควบคุมด้วยพอร์ต A เนื่องจากเป็นตัวเลข 7 ส่วนที่มีขั้วบวกร่วมกัน การแสดงตัวเลข จึงต้องการลอจิก "0" เมื่อกำหนดให้ตัวเลขส่วนใดติดสว่าง ดังแสดงวงจรในรูปที่ 6.1



รูปที่ 6.1 วงจรการเชื่อมต่อกับสวิตช์ควบคุมการทำงาน

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรม คลิกที่ ใบงาน เลือก ใบงานที่ 6
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผลด้วยการกดปุ่ม 1 ถึง 3
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

กดปุ่ม 1 จะทำหน้าที่เลือกรีเลย์ที่ต้องการควบคุม ทราบตำแหน่งรีเลย์จากตัวเลข 7 ส่วน

กดปุ่ม 2 สั่งให้รีเลย์ที่กำลังควบคุมอยู่นั้นอยู่ในสภาวะทำงาน

กดปุ่ม 3 สั่งให้รีเลย์ที่กำลังควบคุมอยู่นั้นอยู่ในสภาวะหยุดทำงาน

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง
3. ทดสอบโดยการคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง บันทึกค่า
   3.1 ปุ่ม 1 ค่าใน Port C ของ 8255 มีค่าเป็น FEH
   3.2 ปุ่ม 2 ค่าใน Port C ของ 8255 มีค่าเป็น FDH
   3.3 ปุ่ม 3 ค่าใน Port C ของ 8255 มีค่าเป็น FBH
4. คลิกปุ่ม [>] เพื่อกลับสู่การทำงานปกติ พร้อมทั้งคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง แล้วบันทึกผลการทดลองดังนี้

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

การกดปุ่มต้องใช้เวลาในการกดให้นานจนกว่าคำสั่งจะถึงการตรวจรับค่าปุ่ม เมื่อกดปุ่มที่1 จะทำให้จอแสดงผล 7 ส่วนแสดงค่าเลขจาก 0-4 และวนกลับมาอีก เมื่อจอแสดงผล 7 ส่วนแสดงเลขใดอยู่หากกดปุ่ม 2 จะทำให้แอลอีดีของเลขนั้น (นับจากบน) สว่างขึ้นแต่ถ้ากดปุ่ม 3 จะทำให้แอลอีดีที่สว่างหรือดับของดวงนั้นดับ

ปรับเวลาไว้ที่ 0

ผลที่ได้คล้ายกับผลการทดลองเมื่อปรับความเร็วไว้ที่ 20 ต่างตรงที่ใช้เวลาในการกดปุ่มต่างๆ น้อยกว่าก็สามารถดูผลการเปลี่ยนแปลงของจอแสดงผล 7 ส่วนได้

สรุปผลการทดลอง

การกดปุ่มบนอุปกรณ์จำลอง ต้องอาศัยจังหวะในการกดจึงจะได้ผลตามต้องการ ในส่วนของตัวแสดงผล 7 ส่วน เป็นอุปกรณ์เสริมที่ช่วยให้การใช้งานดูง่ายขึ้น การประยุกต์ใช้งานสามารถนำไปประยุกต์ใช้กับการสั่ง ปิด-เปิด อุปกรณ์ไฟฟ้าได้

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

ตอบ. ลักษณะโครงสร้างการทำงานของโปรแกรมจะมีลักษณะคล้ายกันทั้งแบบจำลองการทำงานและการทำงานจริง แต่ในการทำงานของไมโครคอนโทรลเลอร์ จะมีความเร็วในการอ่านและทำตามคำสั่งที่เร็วกว่ามาก

2. จงเขียนโปรแกรม ให้มีการควบคุมรีเลย์ โดยสวิตช์ 1 ควบคุมการทำงานของรีเลย์ สวิตช์ 2 ควบคุมการทำงานของรีเลย์ตัวที่ 2 และ สวิตช์ 3 ควบคุมการทำงานของรีเลย์ตัวที่ 3 กดสวิตช์ครั้งแรกสั่งให้รีเลย์ทำงาน กดสวิตช์อีกครั้งสั้งให้รีเลย์หยุดทำงาน

ตอบ.

```
        ORG 8000H
PORT_A  EQU  0E000H
PORT_B  EQU  0E001H
PORT_C  EQU  0E002H
C_PORT  EQU  0E003H
;-------------------------------------------------
SET_8255: MOV  DPTR,#C_PORT
        MOV  A,#81H
        MOVX @DPTR,A
;-------------------------------------------------
        MOV   R1,#00H
        MOV   R2,#00H
        MOV   R3,#00H
SCAN:   LCALL  OUT
        MOV   DPTR,#PORT_C
        MOV   A,#0FFH
        MOVX  @DPTR,A
        MOVX  A,@DPTR
        CJNE   A,#0FEH,SW2
        LJMP   CON1
SW2:    CJNE   A,#0FDH,SW3
        LJMP   CON2
SW3:    CJNE   A,#0FBH,SW4
        LJMP   CON3
SW4:    MOV   DPTR,#PORT_C
        MOVX  A,@DPTR
        MOV   A,#0FFH
        MOVX  @DPTR,A
        LJMP   SCAN
CON1:   MOV    R2,#01H
```

```
        MOV     A,R1
        CPL     ACC.0
        MOV     R1,A
        ACALL   SET_OUT
        ACALL   DELAY
        SJMP    SCAN
CON2:   MOV     R2,#02H
        MOV     A,R1
        CPL     ACC.1
        MOV     R1,A
        ACALL   SET_OUT
        ACALL   DELAY
        SJMP    SCAN
CON3:   MOV     R2,#03H
        MOV     A,R1
        CPL     ACC.2
        MOV     R1,A
        ACALL   SET_OUT
        ACALL   DELAY
        SJMP    SCAN
OUT:    MOV     DPTR,#DATA
        MOV     A,R2
        MOVC    A,@A+DPTR
        MOV     DPTR,#PORT_A
        MOVX    @DPTR,A
        RET
DATA:   DB      0C0H,0F9H,0A4H,0B0H,99H,0C0H
SET_OUT:LCALL   OUT
        MOV     A,R1
        MOV     DPTR,#PORT_B
        MOVX    @DPTR,A
        RET
DELAY:  MOV R6,#0FFH
LOOP1:  MOV R5,#0F0H
LOOP2:  NOP
        DJNZ R5,LOOP2
        DJNZ R6,LOOP1
        RET
        END.
```

เฉลยใบงานที่ 7
มอเตอร์ไฟฟ้ากระแสตรง (DC MOTER)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถควบคุมทิศทางการหมุนของมอเตอร์ไฟฟ้ากระแสตรงได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรเชื่อมต่อมอเตอร์ไฟฟ้ากระแสตรงได้

ทฤษฎี

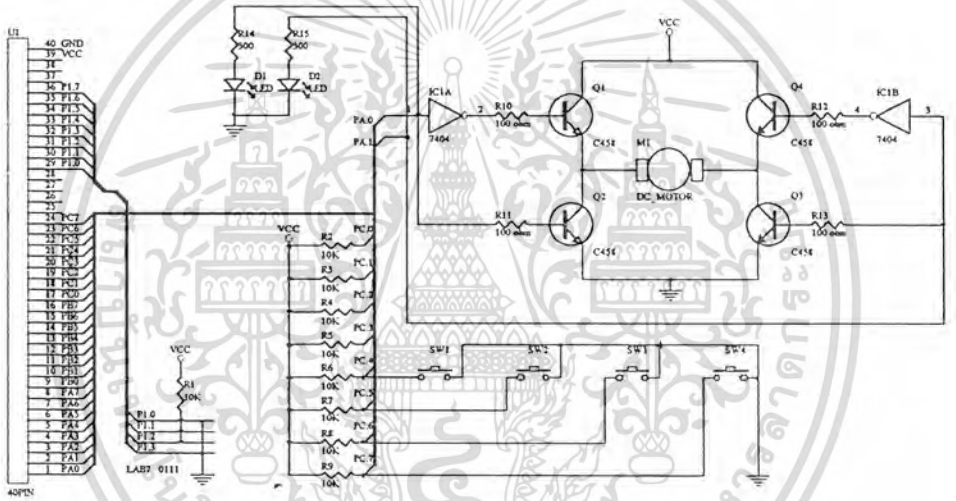
การทดลองที่ 7 เป็นวงจรควบคุมทิศทางการหมุนของมอเตอร์ไฟฟ้ากระแสตรงเลือกใช้ทรานซิสเตอร์ชนิดเอ็นพีเอ็น 4 ตัว ต่อกันในลักษณะกากบาทโดยให้มอเตอร์กระแสตรงอยู่ตรงกลาง และให้สภาวะของทรานซิสเตอร์ที่อยู่คนละด้านคือ Q1 และ Q3 มีสภาวะเดียวกันเช่นเดียวกับ Q2 และ Q4 และสามารถควบคุมสภาวะของทรานซิสเตอร์ 2 คู่นี้ด้วยพอร์ต A0 และ A1 โดยมีไอซีนอตเกทเป็นตัวกลับลอจิกให้ตรงข้ามกัน แสดงความสัมพันธ์ของพอร์ต A0 และ A1 และทิศทางการหมุนของมอเตอร์ได้ดังนี้

ทิศทางการหมุน	พอร์ต A0	พอร์ต A1
ไม่หมุน	0	0
หมุนขวา	0	1
หมุนซ้าย	1	0
ไม่หมุน	1	1

ส่วนสวิตช์ที่ใช้ในการควบคุมการหมุน เลือกใช้พอร์ต C ไบต์สูง คือ PC4 – PC7 เป็นอุปกรณ์ตรวจจับการกดเมื่อกดสวิตช์ 1-4 จะทำให้สภาวะของพอร์ต C เปลี่ยนแปลงไป ดังตารางที่ 5.1

ตารางที่ 5 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 7

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	0	1	1	1	1	1	EF
1	0	0	1	1	1	1	1	2	DF
1	0	1	1	1	1	1	1	3	BF
0	1	1	1	1	1	1	1	4	7F



รูปที่ 7.1 วงจรการเชื่อมต่อกับมอเตอร์ไฟฟ้ากระแสตรง

ลำดับขั้นการทดลอง

ตอนที่ 1

1. เปิดโปรแกรม คลิกที่ ใบงาน เลือก ใบงานที่ 7
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกที่ปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกที่ปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

เมื่อกดปุ่ม 4 ทิศทางของมอเตอร์จำลองจะหมุนตามเข็มนาฬิกา ปุ่ม 3 ทิศทางการหมุนของมอเตอร์จำลองจะหมุนทวนเข็มนาฬิกา ปุ่ม 2 จะสลับทิศทางการหมุนในขณะที่หมุนอยู่เป็นตรงกันข้าม และปุ่ม 1 จะหยุดการหมุนของมอเตอร์

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง
3. ทดสอบโดยการคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง บันทึกค่า
   3.1 ปุ่ม 1 ค่าใน Port C ของ 8255 มีค่าเป็น EFH
   3.2 ปุ่ม 2 ค่าใน Port C ของ 8255 มีค่าเป็น DFH
   3.3 ปุ่ม 3 ค่าใน Port C ของ 8255 มีค่าเป็น BFH
   3.4 ปุ่ม 4 ค่าใน Port C ของ 8255 มีค่าเป็น 7FH
4. คลิกปุ่ม [>] เพื่อให้ทำงานปกติ พร้อมทั้งคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง บันทึกผลการทดลองดังนี้

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

เมื่อกดปุ่มใดก็ทำให้อุปกรณ์จำลองมอเตอร์หมุน ภาพที่ปรากฏจะช้าอย่างเห็นได้ชัดและผลตอบสนองของปุ่มบนอุปกรณ์จำลองก็ช้า เมื่อกดปุ่ม 4 ทิศทางของมอเตอร์จำลองจะหมุนตามเข็มนาฬิกา ปุ่ม 3 ทิศทางการหมุนของมอเตอร์จำลองจะหมุนทวนเข็มนาฬิกา ปุ่ม 2 จะสลับทิศทางการหมุนในขณะที่หมุนอยู่เป็นตรงกันข้าม และปุ่ม 1 จะหยุดการหมุนของมอเตอร์

ปรับเวลาไว้ที่ 0

ผลที่ได้คล้ายกับผลการทดลองเมื่อปรับเวลาไว้ที่ 20 ต่างกันที่ใช้เวลาในการกดปุ่มต่างๆ น้อยกว่าก็สามารถดูผลการเปลี่ยนแปลงของจอแสดงผล มอเตอร์จำลองและแอลอีดีได้

สรุปผลการทดลอง

การทำงานของอุปกรณ์จำลองมีลักษณะคล้ายกับอุปกรณ์ตัวจริง ต่างกันตรงที่เรื่องของปุ่มกล่าวคือผลตอลสนองของปุ่มช้ากว่าอุปกรณ์จริงทั้งนี้ขึ้นอยู่กับความเร็วในการประมวลผลคำสั่ง ค่าหน่วยความจำภายในจำลอง ตั้งแต่ 09H ขึ้นไปจะปรากฏเลขซ้ำกันคือ 80 0B ไปเรื่อยๆ และเมื่อกด

ปุ่มใดปุ่มหนึ่ง ค่าในรีจีสเตอร์ A จะเปลี่ยนแปลง แต่ถ้าไม่มีการกดปุ่มใด ค่าในรีจีสเตอร์ A จะเป็น FFH เสมอ

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

ตอบ. โครงสร้างการทำงานของอุปกรณ์จำลองและอุปกรณ์ที่มีลักษณะคล้ายกัน แต่ในการทดลองอุปกรณ์จำลองภายในโปรแกรมที่เขียนขึ้นจะไม่มีคำสั่งหน่วงเวลาเหมือนอุปกรณ์จริง

2. จงเขียนโปรแกรมให้มอเตอร์ไฟฟ้ากระแสตรง เมื่อกดสวิตช์ 1 และสวิตช์ 2 สั่งให้หยุดหมุน เมื่อกดสวิตช์ 3 สั่งให้หมุนวนทางขวา เป็นเวลาประมาณ 1 วินาที เมื่อกดสวิตช์ 4 สั่งให้หมุนวนซ้ายเป็นเวลาประมาณ 1 วินาที

ตอบ.

```
        ORG 8000H
        PA      EQU  0E000H
        PB      EQU  0E001H
        P_C     EQU  0E002H
        CP      EQU  0E003H
        ;-------------------------------------
        SET_8255: MOV  DPTR,#CP
                  MOV  A,#8BH
                  MOVX @DPTR,A
        ;-------------------------------------
        ST:       LCALL  SCAN
                  LJMP   ST
        ;-------------------------------------
        SCAN:     MOV    DPTR,#P_C
                  MOVX   A,@DPTR
        SW1:      CJNE   A,#0EFH,SW2
                  LJMP   STOP
        SW2:      CJNE   A,#0DFH,SW3
                  LJMP   STOP
        SW3:      CJNE   A,#0BFH,SW4
                  LJMP   FW
        SW4:      CJNE   A,#7FH,SW0
                  LJMP   RW
        SW0:      RET
        ;-------------------------------------
```

```
STOP:   MOV   A,#00H
        MOV   DPTR,#PA
        MOVX  @DPTR,A
        LJMP  ST
;.....
;
RW:     MOV   A,#01H
        MOV   DPTR,#PA
        MOVX  @DPTR,A
        ACALL DELAY
        AJMP  STOP
;.....
;
FW:     MOV   A,#02H
        MOV   DPTR,#PA
        MOVX  @DPTR,A
        ACALL DELAY
        AJMP  STOP
;.....
;
DELAY:  MOV   R1,#50H
LOOP1:  MOV   R2,#50H
LOOP2:  MOV   R3,#50H
LOOP3:  NOP
        DJNZ  R3,LOOP3
        DJNZ  R2,LOOP2
        DJNZ  R1,LOOP1
        RET
;.....
        END
```

เฉลยใบงานที่ 8

มอเตอร์ชนิดสเต็ป (STEPING MOTOR)

วัตถุประสงค์

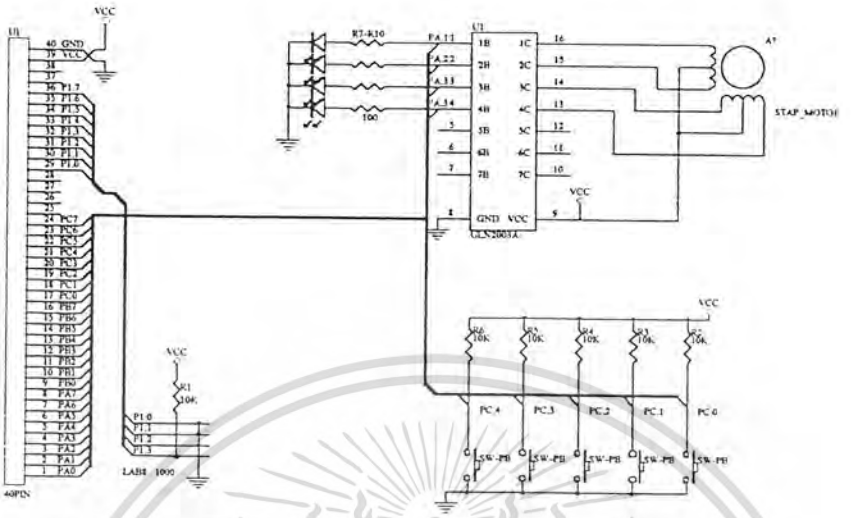
1. เพื่อให้นักศึกษาสามารถใช้งานสเต็ปปิ้งมอเตอร์ได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรสเต็ปปิ้งมอเตอร์ได้

ทฤษฎี

การทดลองที่ 8 เป็นวงจรขับมอเตอร์ชนิดสเต็ปเลือกใช้สเต็ปมอเตอร์แบบ 4 เฟส กำหนดให้พอร์ต A0–A4 ควบคุมเฟส 1–4 ของสเต็ปมอเตอร์ตามลำดับ และมีแอลอีดี เพื่อแสดงสภาวะของแต่ละเฟส ควบคุมการใช้พอร์ต C0–C4 ตรวจจับการกดสวิตช์ 1-5 โดยเมื่อกดสวิตช์ 1-5 จะเกิดสถานะของพอร์ต C ดังนี้

ตารางที่ 8.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 8

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	1	1	1	1	0	1	FE
1	1	1	1	1	1	0	1	2	FD
1	1	1	1	1	0	1	1	3	FB
1	1	1	1	0	1	1	1	4	F7
1	1	1	0	1	1	1	1	5	EF



รูปที่ 8.1 วงจรการเชื่อมต่อกับสเต็ปปิ้งมอเตอร์

ขั้นตอนการทดลอง

ตอนที่ 1

1. เปิดโปรแกรมคลิกปุ่ม ใบงาน เลือก ใบงานที่ 8
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

เมื่อกดปุ่ม 5 เป็นการสั่งมอเตอร์แบบสเตปหยุดหมุน ปุ่ม 2 ให้มอเตอร์แบบสเตปหมุนทวนเข็มตลอดปุ่ม 3 ให้มอเตอร์แบบสเตปหมุนทวนเข็ม 1 จังหวะ ปุ่มที่ 2 ให้มอเตอร์แบบสเตปหมุนตามเข็ม 1 จังหวะ และปุ่มที่ 1 ให้มอเตอร์แบบสเตปหมุนตามเข็มตลอด

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)
2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง

3. ทดสอบโดยการกดปุ่มบนอุปกรณ์ต่อพ่วงจำลองบันทึกค่า

3.1 ปุ่ม 1 ค่าใน Port C ของ 8255 มีค่าเป็น EFH

3.2 ปุ่ม 2 ค่าใน Port C ของ 8255 มีค่าเป็น F7H

3.3 ปุ่ม 3 ค่าใน Port C ของ 8255 มีค่าเป็น FBH

3.4 ปุ่ม 4 ค่าใน Port C ของ 8255 มีค่าเป็น FCH

3.5 ปุ่ม 5 ค่าใน Port C ของ 8255 มีค่าเป็น FEH

4. คลิกปุ่ม [>] เพื่อให้ทำงานปกติ พร้อมทั้งคลิกปุ่มบนอุปกรณ์ต่อพ่วงจำลอง บันทึกผลการทดลอง

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

เมื่อกดปุ่มใดปุ่มหนึ่งทั้ง 5 ปุ่ม ผลตอบสนองต่อปุ่มช้ามากต้องกดแช่ไว้จนกว่าคำสั่งการทำงานจะผ่านไปถึงคำสั่งรับค่าปุ่ม

ปรับเวลาไว้ที่ 0

โครงสร้างการแสดงผลมีลักษณะคล้ายกับการปรับเวลาไว้ที่ 20 แต่ผลการตอบสนองของปุ่มจะรวดเร็วกว่า

สรุปผลการทดลอง

เมื่อกดปุ่ม 5 เป็นการสั่งให้อุปกรณ์จำลองมอเตอร์แบบสเตปหยุดหมุน ปุ่ม 2 ให้มอเตอร์แบบสเตปหมุนทวนเข็มตลอด ปุ่ม 3 ให้มอเตอร์แบบสเตปหมุนทวนเข็ม 1 จังหวะ ปุ่มที่ 2 ให้มอเตอร์แบบสเตปหมุนตามเข็ม 1 จังหวะ และปุ่มที่ 1 ให้มอเตอร์แบบสเตปหมุนตามเข็มตลอด

คำถามท้ายการทดลอง

1. อธิบายเปรียบเทียบระหว่างการทำงานตามคำสั่งในตัวอุปกรณ์จริงและอุปกรณ์จำลอง

ตอบ. เวลาในการตอบสนองและแสดงผลต่างกันคือ เวลาในการตอบสนองและแสดงผลของอุปกรณ์จริงจะเร็วกว่าอุปกรณ์จำลองมาก และในอุปกรณ์จริงจะมีสเตปของมอเตอร์มากกว่าอุปกรณ์จำลองจึงใช้เวลาการครบรอบเร็วกว่าอุปกรณ์จริง

2. จงให้เหตุผลว่าทำไมเมื่อป้อนค่า 01H และ 04H ที่พอร์ต B ตามลำดับแล้วสเตปปิ้งมอเตอร์จึงไม่หมุน

ตอบ. เพราะว่าการป้อนข้อมูลให้มอเตอร์แบบสเตป ต้องเรียงกันในลักษณะที่ให้เฟสชิดกัน หรือห่างกันไม่เกิน 2 เฟส เช่น 01H, 03H, 02H, 06H หรือ 09H, 08H, 0C, 04H เป็นต้น ซึ่งค่าที่ป้อนเข้ามานั้นมีเฟสต่างกันถึง 3 เฟส จึงทำให้มอเตอร์แบบสเตปไม่หมุน

3. จงเขียนโปรแกรมให้เสต็บปิ้งมอเตอร์ เมื่อกดสวิตช์ 5 สั่งให้หยุดหมุน เมื่อกดสวิตช์ 4 สั่งให้หมุนวนทางซ้ายเร็ว เมื่อกดสวิตช์ 3 สั่งให้หมุนวนซ้ายช้า เมื่อกดสวิตช์ 2 สั่งให้หมุนวนทางขวาช้า เมื่อกดสวิตช์ 3 สั่งให้หมุนวนขวาเร็ว

ตอบ.

```
ORG 8000H
PA      EQU  0E000H
PB      EQU  0E001H
P_C     EQU  0E002H
CP      EQU  0E003H
;-----------------------------------
SET_8255: MOV  DPTR,#CP
          MOV  A,#8BH
          MOVX @DPTR,A
;-----------------------------------
ST:       LCALL  SCAN
          LJMP   ST
;-----------------------------------
SCAN:     MOV    DPTR,#P_C
          MOVX   A,@DPTR
          CJNE   A,#0FEH,SW2
          LJMP   RW
SW2:      CJNE   A,#0FDH,SW3
          LJMP   RWW
SW3:      CJNE   A,#0FBH,SW4
          LJMP   FWW
SW4:      CJNE   A,#0F7H,SW5
          LJMP   FW
SW5:      CJNE   A,#0EFH,SW0
          LJMP   STOP
SW0:      RET
;-----------------------------------
STOP:     MOV    A,#00H
          MOV    DPTR,#PA
          MOVX @DPTR,A
```

```
        LJMP   ST
;.....
FWW:    MOV    A,#00H
        MOV    R2,#08H
        MOV    R3,#00H
FWW1:   NOP
        LCALL  SCAN
        LCALL  DATA1
        MOV    DPTR,#PA
        MOVX   @DPTR,A
        LCALL  DELAY_2
        INC    R3
        DJNZ   R2,FWW1
        SJMP   FWW
;.....
RWW:    MOV    A,#00H
        MOV    R2,#08H
        MOV    R3,#00H
RWW1:   NOP
        LCALL  SCAN
        LCALL  DATA
        MOV    DPTR,#PA
        MOVX   @DPTR,A
        LCALL  DELAY_2
        INC    R3
        DJNZ   R2,RWW1
        SJMP   RWW
RW:     MOV    A,#00H
        MOV    R2,#08H
        MOV    R3,#00H
RW1:    NOP
        LCALL  SCAN
        LCALL  DATA
        MOV    DPTR,#PA
        MOVX   @DPTR,A
        LCALL  DELAY_1
        INC    R3
        DJNZ   R2,RW1
        SJMP   RW
```

```
DATA:   MOV     A,R3
        INC     A
        MOVC    A,@A+PC
        RET
DB      01H,03H,02H,06H,04H,0CH,08H,09H
FW:     MOV     A,#00H
        MOV     R2,#08H
        MOV     R3,#00H
FW1:    NOP
        LCALL   SCAN
        LCALL   DATA1
        MOV     DPTR,#PA
        MOVX    @DPTR,A
        LCALL   DELAY_1
        INC     R3
        DJNZ    R2,FW1
        SJMP    FW
;-------------------------------------------------------
DATA1:  MOV     A,R3
        INC     A
        MOVC    A,@A+PC
        RET
DB      09H,08H,0CH,04H,06H,02H,03H,01H
;-------------------------------------------------------
DELAY_1: MOV  R5,#10H
LOOPC:  MOV  R6,#10H
LOOPB:  MOV  R7,#10H
LOOPA:  NOP
        NOP
        DJNZ R7,LOOPA
        DJNZ R6,LOOPB
        DJNZ R5,LOOPC
        RET
```

```
DELAY_2: MOV  R5,#20H
LOOP6:   MOV  R6,#20H
LOOP5:   MOV  R7,#20H
LOOP4:   NOP
         NOP
         DJNZ R7,LOOP4
         DJNZ R6,LOOP5
         DJNZ R5,LOOP6
         RET
;-----------------------------------------------------------
         END.
```

เฉลยใบงานที่ 9

สร้างสัญญาณเสียง (SOUND)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถสร้างสัญญาเสียงจากไมโครคอนโทเลเลอได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมวงจรสร้างสัญญาณเสียงได้

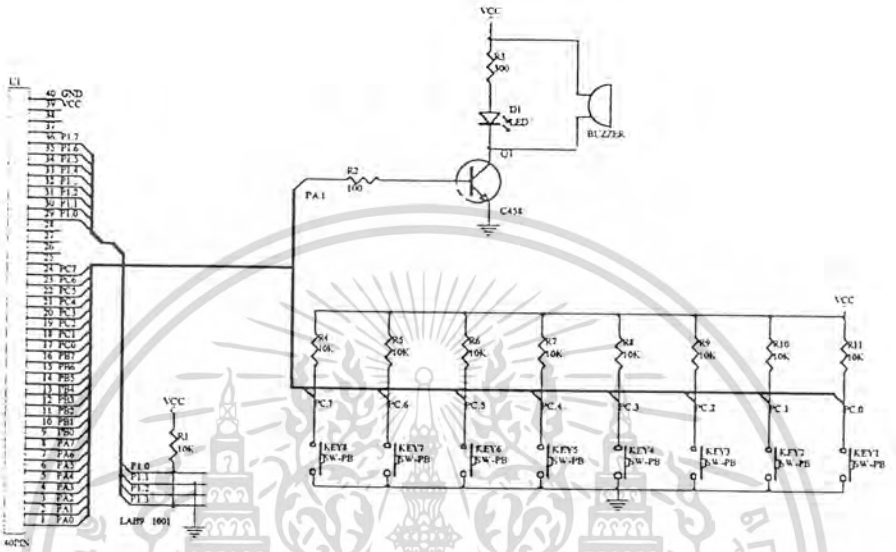
ทฤษฎี

การทดลองที่ 9 ออกแบบเพื่อแสดงการสร้างสัญญาณพัลส์ ด้วยพอร์ต A ควบคุมอัตราความถี่ของเสียงด้วยโปรแกรมและควบคุมโปรแกรมด้วยสวิตช์ 8 ตัวคือ คีย์ร์ 8 ถึง คีย์ร์ 1 เรียงลำดับจากซ้ายไปขวา ค่าที่เกิดขึ้นขณะกดสวิตช์ จะทำให้บิตของพอร์ต C เปลี่ยนสภาวะจาก "1" เป็น "0" เขียนดังตาราง 9.1ที่

ตารางที่ 9.1 การกำหนดค่าเพื่อแสดงผลการทดลองที่ 9

PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	สวิตช์	ฐาน 16
1	1	1	1	1	1	1	0	1	FE
1	1	1	1	1	1	0	1	2	FD
1	1	1	1	1	0	1	1	3	FB
1	1	1	1	0	1	1	1	4	F7
1	1	1	0	1	1	1	1	5	EF
1	1	0	1	1	1	1	1	6	DF
1	0	1	1	1	1	1	1	7	BF
0	1	1	1	1	1	1	1	8	7F

ในส่วนของการขยายเสียงใช้ทรานซิสเตอร์ชนิดเอ็นพีเอ็น รับลอจิกไบอัสจากพอร์ต A1 ขับลำโพงชนิดเปียสโซให้เสียงดังในระดับพอประมาณ ขึ้นอยู่กับความถี่ที่ป้อนไบอัสให้กับทรานซิสเตอร์ดังแสดงในรูปที่ 9.1



รูปที่ 9.1 การเชื่อมต่อกับวงจรสร้างสัญญาณเสียง

ขั้นตอนการทดลอง

ตอนที่ 1

1. เปิดโปรแกรมคลิกปุ่ม ใบงาน เลือก ใบงานที่ 9
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม ส่ง เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม ทำงาน เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่1)

เมื่อกดปุ่มที่ 1 จนถึงปุ่มที่ 8 จะได้ยินเสียงออกทางลำโพงในความถี่ที่แตกต่างกันไล่จากความถี่สูงไปยังความถี่ต่ำตามลำดับ

สรุปผลการทดลอง

การสร้างสัญญาณเสียงออกทางลำโพงทำได้โดยการสลับสภาวะของพอร์ตจากลอจิก 1 ไปเป็นลอจิก 0 กลับไปมาช่วงเวลาระหว่างการเปลี่ยนสภาวะคือการกำหนดความถี่ของสัญญาณ

คำถามท้ายการทดลอง

1. จงเขียนโปรแกรมสร้างสัญญาณเสียงความถี่ 10 กิโลเฮอร์ ออกทางลำโพงโดยกำหนดความถี่คริสตอลของวงจรไมโครคอนโทรลเลอร์ไว้ที่ 11.0592 เมกะเฮิร์ต กำหนดให้ใช้ไทเมอร์ 0

ตอบ.

```
        ORG 8000H
PORT_A  EQU   0E000H
PORT_B  EQU   0E001H
PORT_C  EQU   0E002H
C_PORT  EQU   0E003H
;-------------------------------------------------------
;
SET_8255: MOV  A,#80H
        MOV   DPTR,#C_PORT
        MOVX  @DPTR,A
;-------------------------------------------------------
;
        MOV   A,#00H
        MOV   TMOD,#02
        MOV   TH0,#0CEH
        SETB  TR0
LOOP:   JNB   TF0,LOOP
        CLR   TF0
        MOV   DPTR,#PORT_A
        CPL   A
        MOVX  @DPTR,A
        SJMP  LOOP
;-------------------------------------------------------
END
```

เฉลยใบงานที่ 10

อุปกรณ์ตรวจจับแสง (SENSOR and COUNTER)

วัตถุประสงค์

1. เพื่อให้นักศึกษาสามารถใช้งานอุปกรณ์ตรวจจับการแสงได้
2. เพื่อให้นักศึกษาสามารถเขียนโปรแกรมควบคุมอุปกรณ์ตรวจจับแสงและวงจรนับได้

ทฤษฎี

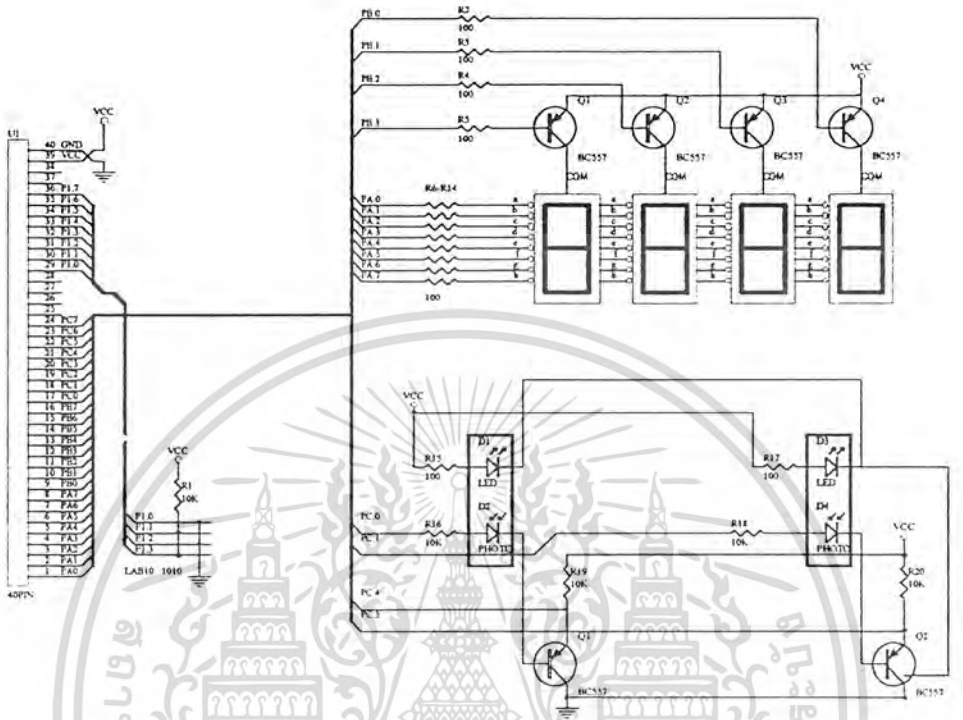
การทดลองที่ 10 เป็นการใช้งานร่วมกันกับอุปกรณ์ตรวจจับแสง ความเปลี่ยนแปลงสถานะของอุปกรณ์ตรวจจับแสงนำมาประมวลผลและแสดงออกทางตัวเลข 7 ส่วน จำนวน 4 หลัก

อุปกรณ์ตรวจจับแสงใช้พอร์ต C เป็นตัวตรวจจับสภาวะจากวงจรรูปที่ 10.1 ให้ทางภาคส่งแสงเป็นพอร์ต C0 และ C1 ส่วนตัวรับแสงใช้พอร์ต C4 และ C5 ตามลำดับ การรับค่าสภาวะของลอจิกจะรับเข้าทางพอร์ต C4 และ C5

ตารางที่ 10.1 แสดงสภาวะของพอร์ต C

PC7	PC6	PC5	PC4	PC5	PC2	PC1	PC0	ฐาน 16
1	1	1	1	1	1	0	0	FC
1	1	1	0	1	1	0	1	ED
1	1	0	1	1	1	1	0	DE
1	1	0	0	1	1	1	1	CF

แสงจากภาคส่งจะส่งออกมาได้นั้น ต้องได้รับสภาวะลอจิก "1" จากพอร์ต C0 และ C1 ทางด้านรับเมื่อได้รับแสง จะทำให้มีสภาวะทางพอร์ต C4 และพอร์ต C5 เป็น "0" ดังนั้นหากมีสิ่งกีดขวาดทางแสง สภาวะที่ได้จากพอร์ต C4 และพอร์ต C5 จะเป็น "1" ทันที จะเกิดการเปลี่ยนแปลงทางลอจิกสามารถนำการเปลี่ยนแปลงไปประมวลผลได้



รูปที่ 10.1 การเชื่อต่อกับวงจรตรวจจับแสง และวงจรนับ

ลำดับขั้นการทดลอง

1. เปิดโปรแกรมคลิกปุ่ม [ใบงาน] เลือก ใบงานที่ 10
2. เลือกหน่วยแสดงผล เป็นทำงานบนคอนโทรลเลอร์ (ไอค่อนรูปไอซี)
3. คลิกปุ่ม [ส่ง] เพื่อส่งค่าให้กับคอนโทรลเลอร์
4. คลิกปุ่ม [ทำงาน] เพื่อดูการเปลี่ยนแปลง และการแสดงผล
5. บันทึกผลการทดลอง และกลับหน้าจอหลัก

ผลการทดลอง (ตอนที่ 1)

เมื่อโปรแกรมเริ่มทำงานปรากฏตัวเลข 3 หลัก จะเริ่มนับเมื่อมีวัตถุกีดขวางทางของแสงทั้ง 2 จุด จะนับขึ้นครั้งละ 1

ตอนที่ 2

1. เลือกหน่วยแสดงผล เป็นทำงานบนคอมพิวเตอร์ (ไอค่อนรูปคอมพิวเตอร์)

2. ส่งรหัสคำสั่งไปยังหน่วยความจำโปรแกรมจำลอง

3. คลิกปุ่ม [>] เพื่อให้ทำงานปกติ พร้อมทั้งลากเม้าท์ผ่านอุปกรณ์ต่อพ่วงจำลอง และบันทึกผลการทดลอง

ผลการทดลอง (ตอนที่ 2)

ปรับเวลาไว้ที่ 20

เมื่อลากเม้าท์ผ่านระหว่างตัวตรวจจับแสงจำลอง ผลตอบสนองที่ได้ช้ามากจึงต้องลากเม้าท์ผ่านให้บ่อยขึ้นและนานมากก่อนจะเกิดผลตอบสนอง

ปรับเวลาไว้ที่ 0

โครงสร้างของผลตอบสนองของโปรแกรมและส่วนแสดงผลมีลักษณะคล้ายกันแต่ผลตอบสนองจะรวดเร็วกว่า

สรุปผลการทดลอง

เมื่อลากเม้าท์ผ่านตัวตรวจจับแสงจำลอง ตัวซ้ายมือค่าใน Port C ของ 8255 มีค่าเท่ากับ EDH ส่วนตัวขวามือมีค่าเท่ากับ DEH ในใบงานนี้ ตัวตรวจจับแสงที่มีผลต่อโครงสร้างโปรแกรม คือ ตัวซ้ายมือ เมื่อลากผ่านจะเพิ่มค่าที่แสดงในจอแสดงผล 7 ส่วน ขึ้นทีละ 1

คำถามท้ายการทดลอง

1. จงอธิบายข้อแตกต่างของการทำงานจากอุปกรณ์จริงและอุปกรณ์จำลอง

ตอบ. การทำงานของอุปกรณ์จริงจะตอบสนองได้รวดเร็วกว่าการทำงานของอุปกรณ์จำลองเนื่องจากอุปกรณ์จำลองจะประมวลผลบนคอมพิวเตอร์ขณะทดลองต้องใช้เม้าเลื่อนไปขวางทางแสงแทนการใช้วัสดุกีดขวางทางแสง

บรรณานุกรม

ฉัททวุฒิ พืชผล และพิชิต สันติกุลานนท์. **คู่มือเรียน Visual Basic 6.** กรุงเทพฯ : บริษัทโปรวิชั่น จำกัด.2539

รัชนัย อินทุใส และไตรภพ อินทุใส. **ไมโครคอนโทรลเลอร์ 8051.** กรุงเทพฯ : หจก.สำนักพิมพ์ฟิสิกส์เซ็นเตอร์.2536

สัจจะ จรัสรุ่งรวีวร. **คู่มือการสร้างแอพพริเคชันด้วย Visual Basic 6.** นนทบุรี : สำนักพิมพ์อินโฟเพรส. 2540

สุทธิศักดิ์ พงศ์ทนาพานิช. **Visual Basic 4.0 Professional.** กรุงเทพฯ : บริษัทซีเอ็ดยูเคชั่นจำกัด (มหาชน). 2338

สมยศ จุณณะปิยะ. **การใช้งานไมโครคอนโทรลเลอร์ ตระกูล MCS-51.** กรุงเทพฯ : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังจัดพิมพ์. 2537.

สุนทร วิทูสุรพจน์. **การใช้งานไมโครคอนโทรล เลอร์ ตระกูล 8051.** กรุงเทพฯ : บริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน). 2337

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายธีระยุทธ นาคอนุเคราะห์
วันเดือนปีเกิด	24 กุมภาพันธ์ พ.ศ. 2520
สถานที่เกิด	จังหวัดกรุงเทพมหานคร
ภูมิลำเนาเดิม	50/28 แขวงวังทองหลาง เขตวังทองหลาง จังหวัดกรุงเทพมหานคร 10310
ที่อยู่ปัจจุบัน	64/18 หมู่ 2 แขวงคันนายาว เขตคันนายาว จังหวัดกรุงเทพมหานคร 10230
โทรศัพท์	(02) 9198552

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนบางกอกศึกษา (ป.1-ป.6) จังหวัดกรุงเทพมหานคร
มัธยมศึกษาตอนต้น	โรงเรียนบางกะปิ (ม.1-ม.3) จังหวัดกรุงเทพมหานคร
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคมีนบุรี
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคมีนบุรี
ปริญญาตรี	สาขา วิศวกรรมโทรคมนาคม ภาควิชา ครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
คติพจน์	เป็นคนดีก็พอแล้ว

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายประทีป หนูรัตน์
วันเดือนปีเกิด	18 กันยายน พ.ศ. 2520
สถานที่เกิด	จังหวัดพัทลุง
ภูมิลำเนาเดิม	13/64 หมู่ 3 แขวงคลองจั่น เขตบางกะปิ จังหวัดกรุงเทพมหานคร 10240
ที่อยู่ปัจจุบัน	4/171 แขวงบึงกุ่ม เขตบึงกุ่ม จังหวัดกรุงเทพมหานคร
โทรศัพท์	(02) 3797750

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนวัดศรีบุญเรือง (ป.1-ป.6) จังหวัดกรุงเทพมหานคร
มัธยมศึกษาตอนต้น	โรงเรียนบางกะปี (ม.1-ม.3) จังหวัดกรุงเทพมหานคร
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคมีนบุรี
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคมีนบุรี
ปริญญาตรี	สาขา วิศวกรรมโทรคมนาคม ภาควิชา ครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

คติพจน์ อย่าทำให้ชาวบ้านเดือดร้อน

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นางสาวศิรินทร ตาสาโรจน์
วันเดือนปีเกิด	19 มกราคม พ.ศ. 2520
สถานที่เกิด	จังหวัดกรุงเทพมหานคร
ภูมิลำเนาเดิม	131 หมู่ 7 แขวงทับยาว เขตลาดกระบัง จังหวัดกรุงเทพมหานคร 10520
ที่อยู่ปัจจุบัน	68/6 หมู่ 7 แขวงลำปลาทิว เขตลาดกระบัง จังหวัดกรุงเทพมหานคร 10520
โทรศัพท์	(02) 7373153

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนวัดทิพพาวาส (ป.1-ป.6) จังหวัดกรุงเทพมหานคร
มัธยมศึกษาตอนต้น	โรงเรียนพรตพิทยพยัต (ม.1-ม.3) จังหวัดกรุงเทพมหานคร
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคมีนบุรี
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคมีนบุรี
ปริญญาตรี	สาขา วิศวกรรมโทรคมนาคม ภาควิชา ครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
คติพจน์	ไม่มีผิดถ้าผิดแล้วแก้ไข ไม่มีถูกถ้าไม่ได้ลองทำ

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายอมรินทร์ ประสิทธิ์
วันเดือนปีเกิด	10 กุมภาพันธ์ พ.ศ. 2521
สถานที่เกิด	จังหวัดกรุงเทพมหานคร
ภูมิลำเนาเดิม	206/2 หมู่ 4 แขวงลาดกระบัง เขตลาดกระบัง จังหวัดกรุงเทพมหานคร 10520
ที่อยู่ปัจจุบัน	385/3 หมู่ 4 แขวงลาดกระบัง เขตลาดกระบัง จังหวัดกรุงเทพมหานคร 10520
โทรศัพท์	(02) 3266751, 7370902

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนมาเรียลัย (ป.1-ป.6) จังหวัดกรุงเทพมหานคร
มัธยมศึกษาตอนต้น	โรงเรียนเทพศิรินทร์ร่มเกล้า (ม.1-ม.3) จังหวัดกรุงเทพมหานคร
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคมีนบุรี
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคมีนบุรี
ปริญญาตรี	สาขา วิศวกรรมโทรคมนาคม ภาควิชา ครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
คติพจน์	ยินดีในสิ่งที่ได้มา ยอมรับในสิ่งที่เสียไป