พีแอลซี

PLC (PROGRAMMABLE LOGIC CONTROLLER)

โดย

นายขรรค์ชัย นิยมอดุลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

พีแอลซี

PLC (PROGRAMMABLE LOGIC CONTROLLER)

โดย

นายขรรค์ชัย นิยมอดุลย์ เลขประจำตัว 40013204

อาจารย์ที่ปรึกษา

อาจารย์ชินภัทร นันทจิวากรชัย

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2544

ปริญญานิพนธ์ ปีการศึกษา 2544

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าคุณทหารลาดกระบัง

เรื่อง พีแอลซี

ผู้จัดทำ

นายขรรค์ชัย นิยมอดุลย์ เลขประจำตัว 40013204

.....อาจารย์ที่ปรึกษา

(.....)

โครงงานเรื่อง พีแอลซี

PROGRAMMABLE LOGIC CONTROLLER

จัดทำโดย

นายขรรค์ชัย นิยมดุลย์ เลขประจำตัว 40013204

โครงงานนี้ได้รับการตรวจสอบแล้วพร้อมที่จะทำการสอบได้

.....

(.....)

อาจารย์ที่ปรึกษา

พีแอลซี

นายขรรค์ชัย นิยมอดุลย์
อาจารย์ชินภัทร นันทจิวากรชัย อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

พีแอลซี เป็นอุปกรณ์ที่ใช้ควบคุมที่นำมาใช้แทนวงจรรีเลย์ในงานอุตสาหกรรม โดยมีการนำไมโครคอนโทรลเลอร์ตระกูล MCS-51 มาใช้ควบคุมการทำงานเป็นพีแอลซีโดยใช้ภาษาแอสเซมบลี้ ในการทำงานด้านฮาร์ดแวร์ ส่วนคำสั่งที่จะนำมาใช้ควบคุมพีแอลซี จะถูกเขียนขึ้นบนภาษาแลดเดอร์ ซึ่งจะถูกพัฒนาขึ้นโดยการนำไปใช้บนระบบปฏิบัติการวินโดวส์ ที่ถูกสร้างขึ้นจากโปรแกรมวิชวลเบสิก (Visual Basic) ทำให้สามารถใช้และพัฒนาต่อไปได้ง่ายขึ้นในอนาคต

PLC

(PROGRAMMABLE LOGIC CONTROLLER)

Mr. Khanchai Niyomadul

Mr. Chinnapat Nanthajiwakornchai Advisor

2001

ABSTRACT

PLC (PROGRAMMABLE LOGIC CONTROLLER) is used instead of relay for controlling equipment or machines in the factories. This project use a microcontroller MCS-51 control Hardware of PLC by assembly language and uses Visual Basic language program to translate the symbols of ladder language from users to objectcode file of microcontroller. Then users send the program to microcontroller by RS-232. The microcontroller will store and process the program. This semester users can use PLC with Windows operation system that users can use it easily

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่1
บทนำ
Programmable Logic Controller (PLC)

Programmable Logic Controller หรือ PLC เป็นอุปกรณ์ควบคุมการทำงานของเครื่องจักรและอุปกรณ์ในโรงงาน ซึ่งเดิม จะใช้วงจรไฟฟ้าที่เรียกว่า " Relay" คือการเดินสายไฟโดยตรง (Hard-Wired) ซึ่งมีข้อเสียคือ เมื่อจำเป็นจะต้องเปลี่ยนกระบวนการผลิตหรือลำดับการทำงานใหม่ก็ต้องเดินสายไฟฟ้าใหม่ซึ่งเสียเวลาและเสียค่าใช้จ่ายสูง แต่ถ้าใช้ PLC จะทำให้สามารถเปลี่ยนกระบวนการได้โดยการโปรแกรมเข้าไปใหม่ซึ่งสะดวกกว่า พีแอลซี นี้ยังมีชื่อเรียกอย่างอื่นอีก เช่น PC ซึ่งย่อมาจาก “Programmable Controller”

PC หรือโปรแกรมเมเบิลคอนโทรลเลอร์ (PROGRAMMABLE CONTROLLER) เป็นอุปกรณ์ที่ควบคุมเครื่องจักรทำงานเรียง ตามลำดับแบบอัตโนมัติ โดยภายในมี Microprocessor เป็นสมองสั่งการหรือซอฟแวร์ PC จะสร้างอุปกรณ์ควบคุมต่าง ๆ ได้ภายในตัวเอง เช่น รีเลย์ ตัวตั้งเวลา ตัวนับ อุปกรณ์ที่กล่าวนี้อยู่ในรูปของโปรแกรม สิ่งเหล่านี้ที่กล่าวมาไม่มีตัวตนในรูปวัตถุ แต่ปรากฏในรูปของฟังก์ชั่น

ประวัติความเป็นมา

เมื่อ พ.ศ. 2511 บริษัท General Motors ฝ่าย Hydromatic ประเทศสหรัฐอเมริกาได้ทำการคิดประดิษฐ์อุปกรณ์ควบคุมใหม่มา แทนวงจรเดิมที่มีใช้ในงานอุตสหกรรม ในปีพ.ศ. 2512 PLC ได้ถูกผลิตขึ้นจำหน่ายในประเทศสหรัฐอเมริกาเป็นแห่งแรก ในประเทศญี่ปุ่น PLC ได้ถูกพัฒนาขึ้นมาภายหลัง จากบริษัทออมรอน (OMROM) ประเทศญี่ปุ่นประสบความสำเร็จในการผลิต โซลิด-สเตท รีเลย์

ความแตกต่างระหว่าง PC กับ PLC

PLC หรือ PROGRAMMABLE LOGIC CONTROLLER จะเป็นอุปกรณ์ควบคุมการทำงานของเครื่องจักร หรือกระบวนการต่าง ๆ ที่มีลักษณะการทำงานเป็นแบบลอจิก หรือเป็นแบบซีเควันซ์ คือ เปิด-ปิด (ON-OFF) หรือ “0” กับ “1” เท่านั้นแต่ PC หรือ PROGRAMMABLE CONTROLER จะรวมเอาการควบคุมที่มีสัญญาณเป็นแบบอนาล็อก ตัวเลข การควบคุมตำแหน่ง การควบคุม PID สรุป PLC มีขนาดเล็กกว่า PC หรือ PLC จะเป็นส่วนหนึ่งของ PC นั่นเอง

วัตถุประสงค์

1. สามารถสร้าง พีแอลซี ขึ้นมาใช้ได้เอง เพื่อการศึกษา และนำไปใช้งานได้จริง
2. สามารถสร้างซอฟแวร์ของ พีแอลซี ที่สามารถเขียนโปรแกรมในรูปของภาษาบูลีนในขณะเดียวกันสามารถแสดงภาพโปรแกรมเป็นภาษาแลดเดอร์ได้ซึ่งเมื่อเขียนโปรแกรมเสร็จแปลงเป็นภาษาแอสเซมบลี้ของไมโครคอนโทรเลอร์ ในขณะแปลงภาษาสามารถตรวจสอบความถูกต้องได้ และเมื่อแปลงเสร็จสามารถส่งโปรแกรมไปเก็บข้อมูลบนหน่วยความจำของเครื่อง พีแอลซี ได้
3. สามารถสร้าง พีแอลซี ที่ใช้ต้นทุนต่ำ

ขอบข่ายของงาน

1. สามารถเขียนโปรแกรมได้ง่าย, ใช้งานได้ง่าย
2. การใช้งานโดยการเขียนโปรแกรมควบคุมบนเครื่องคอมพิวเตอร์(PC) และส่งผ่านโปรแกรมโดยผ่านการสื่อสารของพอร์ตอนุกรม RS-232 ไปยังไมโครคอนโทรเลอร์
3. ในส่วนของฮาร์ดแวร์ มีอุปกรณ์อินพุทและเอาท์พุทหลักบนตัวเครื่อง อย่างละ 8 ชุด และมี Auxiliary Relay
4. โปรแกรมสามารถประมวลผลภาษาบูลีนของพีแอลซี เป็นภาษาของไมโครคอนโทรเลอร์ ในรูปแบบ Hex File ได้
5. โปรแกรมมีความสามารถทำคำสั่งหลักๆ ของภาษา พีแอลซีได้

ประโยชน์หรือผลที่คาดว่าจะได้รับ

1. สามารถนำ พีแอลซี ไปใช้แทนระบบเก่าที่ใช้รีเลย์ซึ่งติดตั้งและดัดแปลงแก้ไขลำบากมาเป็น ระบบควบคุมที่ใช้วงจรอีเล็กทรอนิกส์แทนรีเลย์ โดยสามารถเขียนโปรแกรมควบคุมกำหนดเงื่อนไขในการทำงานแทนการเดินสายเชื่อมต่อวงจรไฟฟ้าแบบเก่าเพื่อความสะดวก
2. สร้างความเข้าใจและความหมายในเรื่องของ พีแอลซี แก่ผู้ที่ทำการศึกษา
3. เป็นแนวทางการศึกษา และพัฒนา แก่ผู้ที่สนใจทำการพัฒนาต่อไป

ขั้นตอนการทำโครงงาน

1. ค้นคว้าหาข้อมูลที่เกี่ยวข้องของพีแอลซี
2. ออกแบบฮาร์ดแวร์(Hard ware) โดยใช้โปรแกรม Protel
3. ออกแบบโปรแกรมควบคุมทดสอบการทำงาน

4. สร้างและทดลองฮาร์ดแวร์ตามที่ออกแบบเอาไว้
5. ออกแบบการเขียนโปรแกรมโดยใช้โปรแกรมวิชวลเบสิก เพื่อสร้างซอฟต์แวร์
6. ทดลองใช้งานและแก้ไขข้อผิดพลาด
7. ประเมินผลการใช้การใช้โปรแกรมและสรุปผลการทำงาน

บทที่ 2
โครงสร้างการทำงานของพีแอลซี

PLC เป็นอุปกรณ์ทางอิเล็กทรอนิกส์ ทำหน้าที่ควบคุมเครื่องจักรหรือกระบวนการผลิต โดยใช้โปรแกรมในหน่วยความจำ กำหนดเงื่อนไขการควบคุมผ่านทางหน่วยอินพท/เอาท์พุท PLC ประกอบด้วยส่วนสำคัญ 2 ส่วนคือ หน่วยประมวลผลกลาง หรือ CPU และหน่วยอินพุท/เอาท์พุท

2.1 ลักษณะการทำงานของ PLC

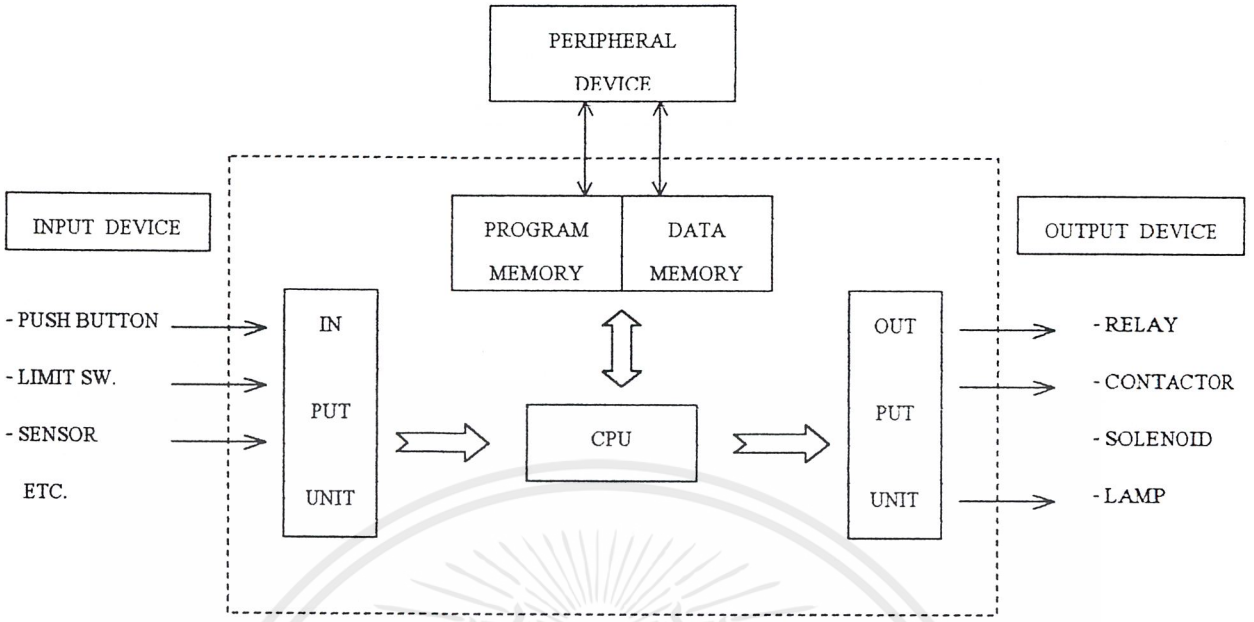
CPU เป็นส่วนประกอบสำคัญของ PLC ทำหน้าที่ตัดสินใจ และควบคุมการทำงานทั้งหมดของ PLC โดยการรับค่าสภาวะ ต่างๆของเครื่องจักรหรือกระบวนการผลิตผ่านทางหน่วยอินพุท ประมวลผลตามโปรแกรมของผู้ใช้ที่เก็บไว้ในหน่วยความจำ จากนั้นจึงนำผลลัพธ์ที่ได้ส่งไปควบคุมเครื่องจักรทางหน่วยเอาท์พุท การทำงานของ PLC ทั้งหมดนี้เรียกว่า การสแกน (scanning) หน่วยจ่ายกำลังมีหน้าที่จ่ายพลังงานไฟฟ้าให้ CPU และหน่วยความจำทำงานตามปกติ

หน่วยอินพุต/เอาท์พุททำหน้าที่ติดต่อระหว่าง PLC กับเครื่องจักร กระบวนการผลิต หรืออุปกรณ์ ภายนอก หน่วยอินพุท ทำหน้าที่รับค่าสัญญาณอินพุทในรูปแบบต่างๆ จากภายนอก เช่น สวิตซ์ต่างๆหรืออุปกรณ์หรืออุปกรณ์ตรวจจับ(sensor) ที่วัดค่าสัญญาณอนาล็อก แล้วปรับระดับของสัญญาณให้เหมาะสมกับ PLC หน่วยเอาท์พุททำหน้าที่ส่งสัญญาณเอาท์พุท ออกไปควบคุมเครื่องจักรหรืออุปกรณ์ภายนอกต่างๆเช่น หลอดไฟฟ้า กระดิ่ง มอเตอร์ไฟฟ้า และวาล์วควบคุม (control valve)

นอกจาก PLC จะประกอบด้วยหน่วยประมวลผลกลางหรือ CPU หน่วยอินพุท/เอาต์พุท แล้วยัง ประกอบด้วยหน่วยป้อน โปรแกรมที่เขียนขึ้นไว้เก็บในหน่วยความจำ ปกติหน่วยป้อนโปรแกรมจะเชื่อมต่อกับ PLC เมื่อผู้ใช้ต้องการป้อน ตรวจสอบ หรือแก้ไขโปรแกรมเท่านั้น และ PLC เองก็สามารถทำงานได้โดยไม่ต้องพึ่งหน่วยป้อนโปรแกรม หน่วยป้อนโปรแกรมจึงไม่ได้ ถูกจัดเป็นส่วนประกอบของ PLC

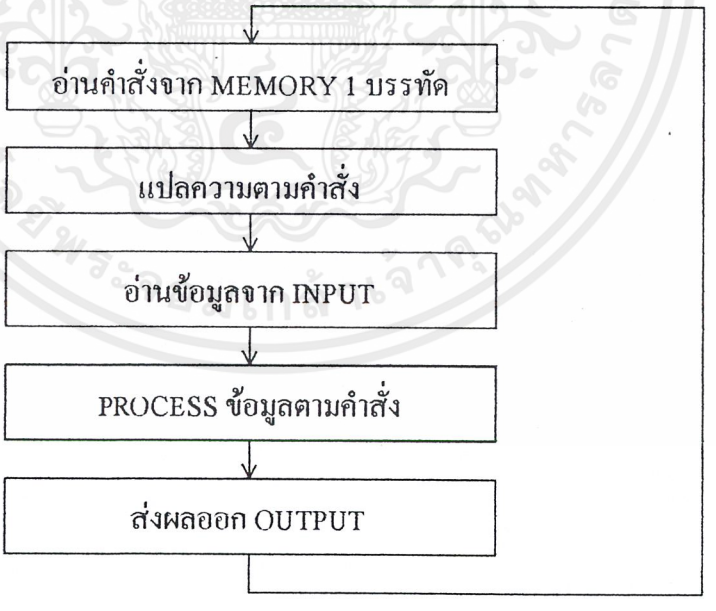
2.2 กระบวนการทำงานของ PLC

- หน่วยประมวลผลกลางCPU (The microprocessor or processor module)
- หน่วยความจำ(Memory)
- หน่วยรับข้อมูล (Input Module)
- หน่วยเอาท์พุต(Output Module)



รูปที่ 2.1 แสดงโครงสร้างของ PLC

2.2.1 หน่วยประมวลผลกลางCPU (The microprocessor or processor module)



รูปที่ 2.2 Flow Chat แสดงการทำงานของ CPU

เป็นส่วนมันสมองของระบบ ภายใน CPU ประกอบไปด้วยวงจร Logic Gate ชนิดต่างๆ หลายชนิดและมี Microprocessor-based ใช้สำหรับแทนอุปกรณ์จำพวก Relay Counter Timer และ

Sequencer เพื่อให้ผู้ใช้ได้ออกแบบได้สะดวกขึ้น การประมวลผลของ CPU จากโปรแกรมทำได้โดยรับข้อมูลและค่าของสภาวะของอุปกรณ์จาก Input มาเก็บไว้ในหน่วยความจำ เสร็จแล้วจะทำการปฏิบัติตามโปรแกรมที่เขียนไว้ทีละคำสั่งจากหน่วยความจำนั้นจนสิ้นสุดแล้วส่งไปที่ Output

2.2.2 หน่วยความจำ(Memory

หน่วยความจำทำหน้าที่สำคัญ คือ จะทำหน้าที่เก็บโปรแกรมของผู้ใช้และข้อมูล สำหรับใช้ในการปฏิบัติงานของ PLC โดยที่ PLC แบ่งหน่วยความจำที่สำคัญออกเป็น 2 ส่วนด้วยกันคือ

1.1 **หน่วยความจำระบบ(Systm Memory)**

หน่วยความจำระบบเป็นส่วนที่เก็บโปรแกรมบริหารระบบ และระบบข้อมูลที่ไม่อนุญาติให้ผู้ใช้ทำการแก้ไขและเปลี่ยนแปลงได้

1.2 **หน่วยความจำผู้ใช้(User Memory)**

หน่วยความจำส่วนที่ทำหน้าที่เก็บโปรแกรมผู้ใช้เก็บข้อมูล อินพุต เอาพุต และอุปกรณ์ภายในสามารถทำ การแก้ไข เปลี่ยนแปลง ข้อมูลได้ส่วนใหญ่เป็นชนิดหน่วยความจำRAM

ชนิดของหน่วยความจำ

หน่วยความจำ RAM(Ramdom Access Memory)

เป็นหน่วยความจำที่สามารเปลี่ยนแปลงแก้ไขข้อมูลได้ หน่วยความจำนี้จะมีแบตเตอร์รี่เล็กๆต่อไว้เพื่อนำไปเลี้ยงข้อมูลเมื่อเกิดไฟดับข้อมูลจะสูญหาย โดยการอ่านและเขียนโปรแกรมจะทำได้ง่ายเหมาะกับงานที่มีการเปลี่ยนข้อมูลบ่อยๆส่วนใหญ่เป็นการเก็บโปรแกรมในช่วงพัฒนา

หน่วยความจำROM(Read Memory)

เป็นหน่วยความจำไม่สามารถเปลี่ยนแปลงแก้ไขข้อมูลภายในได้ง่าย หน่วยความจำนี้เมื่อไม่มีกระแสไฟเลี้ยงก็ยังคงเก็บโปรแกรมข้อมูลไว้ได้หน่วยความจำนี้จะอ่านโปรแกรมได้อย่างเดียวผู้ใช้ไม่สามารถเขียนโปรแกรมเข้าไปได้เหมาะสมสำหรับใช้เป็นโปรแกรมบริหารระบบเป็นโปรแกรมที่เสร็จสิ้นสมบูรณ์

หน่วยความจำ EPROM(Erable Programmable Read Only)

หน่วยความจำชนิดนี้ต้องใช้เครื่องมือพิเศษ ในเขียนโปรแกรมเครื่องมือเขียนโปรแกรมนี้เรียกว่าPROM WRITEหน่วยความจำEPROMใช้เก็บโปรแกรมที่มีการพัฒนา ที่สมบรูณ์ดีแล้วที่ไม่

มีการเปลี่ยนแปลงแก้ไขอีกเมื่อไฟดับโปรแกรมที่เก็บไว้จะไม่มีการสูญหายถ้าจะลบโปรแกรมต้องใช้ เครื่องลบโปรแกรม หรือใช้แสงอุลตร้าไวโอเลต

หน่วยความจำEEPROM(Electrical Erasable Programable)

หน่วยความจำชนิดนี้เหมาะสำหรับเก็บโปรแกรม ที่พัฒนาสมบรูณ์แบบแล้วหน่วยความจำนี้มีลักษณะคล้ายหน่วยความจำEPROMเมื่อเกิดไฟดับโปแกรมนี้จะไม่สูญหายเช่นกัน การอัดโปรแกรมเข้าไปใช้เครื่องอัดชนิดพิเศษ ในการลบโปแกรมและข้อมูลนั้นทำได้โดยใช้วิธีการป้อนสัญญานพัลส์

2.2.2 หน่วยรับข้อมูล (Input Module)

หน่วยอินพุตทำหน้าที่เชื่อมต่อระหว่างอุปกรณ์ภายนอกกับPLC

สรุปหน้าที่ของหน่วยอินพุตได้ดังนี้

- แปลงระดับสัญญาณจากภายนอกให้เหมาะสมกับระบบการทำงานของCPU
- แบ่งสัญญาณภายนอกและภายในออกจากกัน เพื่อป้องกันไม่ให้หน่วยประเมิลผลเกิดการเสียหายเมื่อหน่วยอินพุตลัดวงจร
- แก้ปัญหาการสั่นสะเทือนของหน้าสัมผัสได้
- หน่วยอินพุตมี 2 ประเภทด้วยกันคือ หน่วยอินพุตกระแสตรงและหน่วยอินพุตกระแสสลับ

2.2.4 หน่วยเอาท์พุต(Output Module)

หน่วยเอาท์พุตจะทำหน้าที่เชื่อมต่อระหว่าง CPU อุปกรณ์ภายนอกโดยให้ค่าสภาวะต่างๆ แก่อุปกรณ์ด้านเอาท์พุตซึ่งอยู่ในลักษณะของการเปิดและปิดเท่านั้นหน้าที่หลักของเอาพต์พุตก็คือรับสัญญาณจากCPU แยกสัญญาณระหว่าง CPU กับอุปกรณ์ภายนอก ขยายสัญญาณให้มีค่าสูงส่งออกไปยังเอาท์พุท หน่วยเอาพุตแบ่งได้เป็น 2 ประเภทคือ หน่วยเอาท์พุตกระแสตรง และหน่วยเอาท์พุตกระแสสลับหน่วยเอาต์พุตบางประเภทใช้ได้ทั้งไฟฟ้ากระแสตรงและสลับ

2.3 คุณสมบัติของ PLC

1. ขนาดของระบบเล็กลง

ภายในของ PLC จะใช้อุปกรณ์ทางอิเล็กทรอนิกส์และซอฟแวร์แทนรีเลย์ ตัวตั้งเวลา ตัวนับ และองค์ประกอบของวงจรซีเคว็นอื่น ๆ อีกมากมายซึ่งจำนวนของอุปกรณ์ต่าง ๆ เหล่านี้จะ

ไม่ขึ้นอยู่กับขนาดของ PLC

2. ใช้โปรแกรมแทนการเดินสาย

วงจรรีเลย์ต้องการการเดินสายระหว่างรีเลย์และอุปกรณ์ต่าง ๆ เพื่อประกอบวงจรการควบคุม แต่ PLC ใช้โปแกรมรูปวงจรในหน่วยความจำจึงไม่มีการเดินสายระหว่างอุปกรณ์จริง ๆ ให้ยุ่งยาก

3. เปลี่ยนวงจรและขยายระบบง่าย

โปรแกรมใน PLC สามารถเปลี่ยนแปลงแก้ไขได้โดยง่ายแล้ถ้าต้องการขยายระบบก็ง่ายเช่นเดียวกัน

4. ลดเวลาในการออกแบบการสร้าง

อุปกรณ์ของ PLC เป็นมาตรฐานสามารถประกอบใส่ตู้ควบคุมได้รวดเร็วการออกแบบวงจรและการโปรแกรมทำได้รวดเร็ว นอกจากนั้นยังสามารถทดสอบวงจรโดยทดลองใน PC ได้ด้วย ทำให้การทดลองวงจรเป็นได้โดยรวดเร็ว

5. PLC มีเสถียรภาพมากกว่าแบบรีเลย์

ชิ้นส่วนภายในของ PLC เป็น SOLIC STATE วงจรควบคุมไม่มีการเดินสายอย่างเช่นรีเลย์ จึงไม่มีปัญหาเรื่องสายขาด หน้าสัมผัสหลวม, หน้าสัมผัสไม่ดี นอกจากนั้นใน PLC ยังมีโปรแกรมที่สามารถทดสอบตัวเองได้อีกด้วย

6. มีหน่วยอินพุท/เอาท์พุท หลายแบบ

ในปัจจุบัน PLC มีหน่วยอินพุท/เอาท์พุท หลายแบบสามารถเลือกใช้ให้เหมาะสมกับสภาพของงาน เช่น LOGIC INPUT/OUTPUT, ANALOG INPUT/OUTPUT เป็นต้น

ปัจจุบัน พีแอลซี มีหลายขนาด ผู้ผลิตหลายบริษัทพยามผลิตและออกแบบให้เหมาะสมกับงานแต่ละประเภท การแบ่งขนาดของ พีแอลซี ในที่นี้จะแบ่งตามขนาดของหน่วยอินพุท/เอาท์พุท ซึ่งสามารถแบ่งออกได้ 4 ขนาดด้วยกัน

1. PLC ขนาดเล็ก คือ มีอุปกาณ์อินพุท/เอาท์พุทและหน่วยความจำ ที่ใช้แทนอุปกรณ์รีเลย์ในการควบคุมแบบ ON/OFF ไม่เชื่อมโยงกับคอมพิวเตอร์และ PC ระบบอื่น จำนวนอินพุท/เอาท์พุทไม่เกิน 128 จุด
2. PLC ขนาดกลาง คือ มีหน่วยอินพุท/เอาท์พุท ประมาณ 64-1024 จุด มีการควบคุมแบบอนาล็อก การคำนวณพื้นฐานทางคณิตศาสตร์ การจัดการข้อมูลสามารถเชื่อมโยงกับคอมพิวเตอร์
3. PLC ขนาดใหญ่ คือ ใช้กับระบบควบคุมขนาดใหญ่ มีข้อมูลจำนวนมากและการคำนวณมีการซับซ้อน มีจำนวนอินพุท/เอาท์พุท ประมาณ 2048 จุด
4. PLC ขนาดใหญ่พิเศษ คือ ประกอบด้วยหน่วยอินพุท/เอาท์พุท ประมาณ 4096 จุด ทำหน้าที่เป็นส่วนควบคุมหลักแทนคอมพิวเตอร์ ในระบบควบคุมส่วนใหญ่จะใช้ PC หลายเครื่องทำงานร่วมกัน

2.4 ไมโครคอนโทรเลอร์

เป็นไอซีที่มีการยืดหยุ่นในการทำงานสูงจึงเหมาะสมที่จะใช้ในการนำมาควบคุม การทำงานของพีแอลซี ซึ่งมีความหลายหลายในการใช้งานจะขอกล่าวถึงเฉพาะไมโครคอนโทรเลอร์ MCS-51 ที่นำมาใช้งานจริง

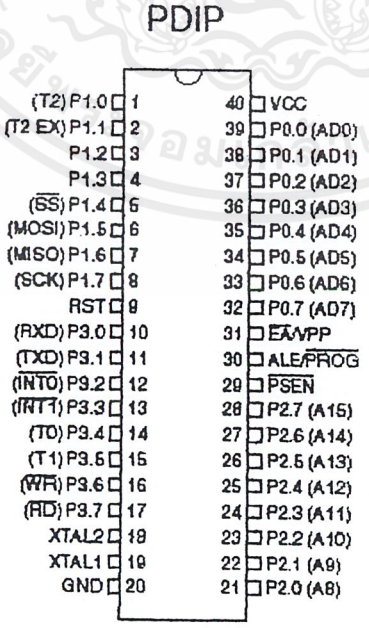
คุณสมบัติทั่วไปที่สำคัญของไมโครคอนโทรเลอร์ตะกูล MCS-51 มีดังนี้

- เป็นไมโครคอนโทรเลอร์ขนาด 8 บิท

- มีวงจรออสซิเลเตอร์และวงจรผลิตสัญญาณนาฬิกาภายในไอซี
- สัญญาณอินพุทเอาท์พุทจำนวน 32 บิต
- สามารถเชื่อมต่อหน่วยความจำข้อมูลภายนอก (external program memory) โดยอ้างตำแหน่งแอดเดรสได้ถึง 64 k
- มีหน่วยความจำโปรแกรมภายในตัว (on-chip program memory) ขนาด 4 K
- มีหน่วยความจำข้อมูลภายในตัว (on-chip data memory) ขนาด 128 ไบต์
- มีหน่วยความจำข้อมูลบางส่วนสามารถเข้าถึงข้อมูลระดับบิตได้ด้วย ทำให้การควบคุมหรือการตรวจสอบสถานะบิตทำได้ง่าย ส่งผลให้เขียนโปรแกรมได้ง่ายขึ้น
- มีไทม์เมอร์/เคาท์เตอร์ ขนาด 16 บิต จำนวน 2 ตัว
- การอินเตอร์รัปต์สามารถทำได้จาก 5 แหล่งกำเนิด โดยการอินเตอร์รัปต์ยังสามารถจัดระดับความสำคัญได้เป็น 2 ระดับ
- มีพอร์ตสื่อสารอนุกรมภายในตัวเอง ซึ่งทำงานเป็นแบบฟูลดูเพล็กซ์ (Full duplex)
- มีคำสั่งในการคำนาณทางคณิตศาสตร์
- คำสั่งส่วนใหญ่ใช้เวลาเพียง 1 ไมโครวินาที เมื่อใช้คริสตอลความถี่ 12 เมกะเฮริตซ์
- ต้องการแหล่งจ่ายไฟ 5 โวลท์ เพียงชุดเดียว

โครงสร้างภายนอกของ MCS-51

ไมโครคอนโทรเลอร์ตระกูล MCS-51 ทุกเบอร์จะมีตำแหน่งขาพื้นฐานที่เหมือนกัน ดังแสดงในรูปที่ 2.3 สำหรับหน้าที่การใช้งานของแต่ละขามีดังนี้



รูปที่ 2.3 แสดงการจัดตำแหน่งขาต่างๆ ของไมโครคอนโทรเลอร์ตระกูล MCS-51

- ขา Vcc เป็นขาป้อนแรงดันไฟเลี้ยง + 5 โวลท์
- ขา Vss เป็นขากราวด์
- ขาพอร์ต 0 (Port 0) มีขา 8 ขา ได้แก่ P0.0 – P 0.7 เป็นขาอินพุทแบบสองทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุทพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดให้ขาพอร์ตเหล่านั้นอยู่ในสภาวะปล่อยลอย ซึ่งในสถานะนี้เองที่สามารถนำมาใช้เป็นพอร์ตอินพุทอิมพีแดนซ์สูงได้ นอกจากพอร์ตนี้จะใช้งานเป็นพอร์ตอินพุทเอาท์พุทแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วยโดยทำหน้าที่ในการกำหนดตำแหน่งแอดเดรสไบท์ต่ำ (A0- A7) ซึ่งจะใช้งานเป็นแบบมัลติเพล็กซ์กับการรับส่งขนาด 8 บิต (D0- D7)
- ขาพอร์ต 1 (Port 1) มี 8 ขา ได้แก่ขา P1.0 – P1.7 เป็นขาพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทาง สำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุทพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ต เพื่อกำหนดพอร์ตให้เป็นพอร์ตอินพุท
- ขาพอร์ต 2 (Port 2) มี 8 ขา ได้แก่ขา P2.0 – P2.7 เป็นขาพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทางสำหรับใช้งานทั่วไป โดยถ้าใช้งานเป็นอินพุทพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ตเพื่อกำหนดให้เป็นพอร์ตอินพุท นอกจากจะใช้งานเป็นพอร์ตอินพุทเอาท์พุทแล้วมันยังถูกใช้งานในการติดต่อกับหน่วยความจำภายนอกด้วยโดยทำหน้าที่ในการกำหนดตำแหน่งแอดเดรสไบท์สูง (A8- A15)
- ขาพอร์ต 3 (Port 3) มี 8 ขา ได้แก่ขา P3.0 – P3.7 เป็นขาพอร์ตอินพุทเอาท์พุทแบบ 2 ทิศทางสำหรับการใช้งานทั่วไป โดยถ้าใช้งานอินพุทพอร์ตต้องทำการเขียนค่า 1 ไปยังแต่ละบิตของพอร์ตเพื่อกำหนดให้เป็นพอร์ตอินพุท นอกจากจะใช้งานเป็นพอร์ตอินพุทเอาท์พุทแล้วมันยังถูกใช้งานในหน้าที่พิเศษต่าง ๆ ดังต่อไปนี้

  - ขารีเซต (RST) ใช้สำหรับการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยการรีเซตต้องคงสถานะเป็นอย่างน้อย นาน 2 แมชชีนรีไซเคิล ในขณะที่ออสซิล-เตอร์ยังทำงานอยู่

  - ขา ALE / $\overline{\text{PROG}}$ เป็นขาสัญญาณเพื่อทำหน้าที่ควบคุมการแลตซ์ (Latch) ค่าตำแหน่งแอดเดรสไบต์ต่ำ (Address Latch Enable)เมื่อต้องการติดต่อกับหน่วยความจำภายนอก นอกจากนี้ขานี้ยังทำหน้าที่เป็นอินพุทรับสัญญาณพัลส์ในการโปรแกรม (Program pluse input) ในส่วนของหน่วยความจำ EPROM สำหรับไมโครคอนโทรลเลอร์ในตระกูล MCS – 51 ทีมีหน่วยความจำโปรแกรมภายในเป็น EPROM

  - ขา $\overline{\text{PSEN}}$ (Program Store Enable)ทำหน้าที่เป็นสัญญาณสโตรปเพื่ออ่านคำสั่งจากหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ประมวลผลคำ

สั่งจากหน่วยความจำภายนอก ขานี้จะส่งสัญญาณสโตรปจำนวน 2 ครั้งในแต่ละแมชชีนไซเคิล แต่ในขณะที่ติดต่อกับหน่วยความจำข้อมูลภายนอกจะไม่มีการส่งสัญญาณสโตรปแต่อย่างใด

- ขา EA / VPP (External Access enable / VPP)เป็นขาสำหรับการเลือกใช้หน่วยความจำโปรแกรมภายในหรือจากภายนอก โดยถ้ามีสถานะเป็น 0 หมายถึงให้ไมโครคอนโทรลเลอร์รับคำสั่งจากหน่วยความจำภายนอก นอกจากนี้ขานี้ยังทำหน้าที่รับแรงดันไปสำหรับการโปรแกรม (Vpp)ขนาด 21 โวลท์ เพื่อใช้ในการโปรแกรม EPROM

- ขา XTAL1 และ ขา XTAL2 เป็นขาอินพุทและเอาท์พุทของวงจรอินเวอร์ติ้งออสซิลเลเตอร์แอมปลิไฟเออร์ (Inverting Ossilator Amplifier Circuit)สำหรับใช้ต่อร่วมกับคริสตอลภายนอก

บทที่ 3
ภาษาของ PLC

คำสั่งที่ใช้เขียนโปรแกรม PLC มี 4 ภาษาคือ ภาษาแลดเดอร์ ภาษาบูลลีน ภาษาบล็อก และคำสั่งข้อความภาษาอังกฤษ (English statement language) ซึ่งแต่ละภาษามีวิธีการใช้ที่แตกต่างกัน ภาษาแลดเดอร์และภาษาบูลลีนเป็นภาษาพื้นฐานที่ใช้กับ PLC ขนาดเล็กแทนอุปกรณ์รีเลย์ อุปกรณ์หน่วงเวลาและนับจำนวนในการควบคุมแบบ ON/OFF ภาษาบล็อกและคำสั่งข้อความภาษาอังกฤษเป็นภาษาระดับสูงมักใช้กับการควบคุมที่ซับซ้อนหรือมีการคำนวณทางคณิตศาสตร์มาเกี่ยวข้อง เช่นการควบคุมแบบอนาล็อกและการควบคุมตำแหน่ง โดยใช้ร่วมกับภาษาแลดเดอร์และภาษาบูลลีนใน PLC ขนาดกลางและขนาดใหญ่

3.1 ภาษาที่ใช้สำหรับ PLC มีหลายภาษาด้วยกัน คือ

1 ภาษาแลดเดอร์

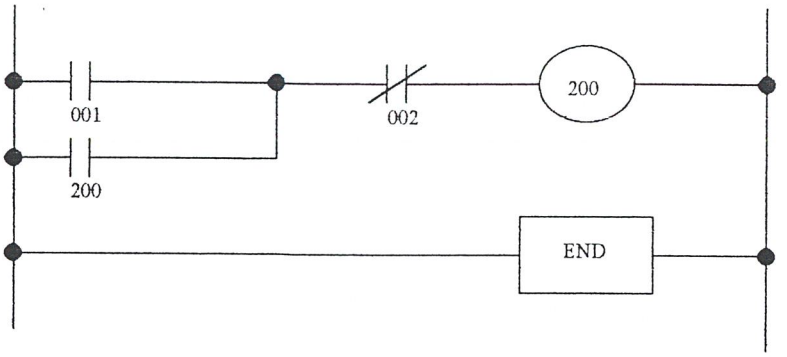
2 ภาษาบูลลีน

3 ภาษาบล็อค

4 ภาษาข้อความภาษาอังกฤษ

3.1.1 ภาษาแลดเดอร์

ภาษาแลดเดอร์ (Ladder language) ประกอบด้วยสัญลักษณ์หน้าสัมผัส มีลักษณะคล้ายวงจรรีเลย์ การเขียนโปรแกรมภาษาแลดเดอร์จากวงจรรีเลย์จึงทำได้ง่าย PLC ที่ใช้ภาษาแลดเดอร์ คือ PLC ขนาดเล็กและขนาดกลาง ปัจจุบันคำสั่งภาษาแลดเดอร์มีครบทั้ง 6 กลุ่มคำสั่ง คือ วงจรรีเลย์และปฏิบัติการตรรก การหน่วงเวลา และนับจำนวน การคำนวณทางคณิตศาสตร์ การจัดการข้อมูล การเคลื่อนย้ายข้อมูล และคำสั่งควบคุมโปรแกรม โปรแกรมภาษาแลดเดอร์ที่ประกอบด้วยหน้าสัมผัสต่างๆจะทำงานร่วมกันเพื่อส่งสภาวะการควบคุมไปยังอุปกรณ์ output 1 จุด หรือที่เรียกว่า*รังก์* บางครั้งโปรแกรมภาษาแลดเดอร์ 1 รังก์ อาจมีอุปกรณ์ output มากกว่า 1 จุด แต่อุปกรณ์ output เหล่านี้ต้องได้รับสภาวะการควบคุมจากจุดเดียวกันเสมอ



รูปที่ 3.1 ตัวอย่างการเขียนโปรแกรมภาษาแลดเดอร์

3.1.2 ภาษาบูลลีน

PLC ขนาดกลางในปัจจุบันนิยมใช้คำสั่งภาษาบูลลีนภาษาบูลลีนเป็นภาษาพื้นฐานของ PLC มีลักษณะคล้ายกับพีชคณิตบูลลีน

โครงสร้างของภาษาบูลลีนประกอบด้วยส่วนหลัก ที่สำคัญ 3 ส่วน ที่สำคัญคือ

1. หมายเลขกำหนดบรรทัด ของโปรแกรม (ADDRESS OR STEP)
2. คำสั่ง (INSTRUETION WORD)
3. หมายเลขกำกับอุปกรณ์ และหน้าสัมผัสต่าง ๆ (DATA)

ADDRESS	INSTRUCTION	DATA
00000	LD	00000
00001	OR	10000
00002	AND NOT	00001
00003	OUT	10000
00004	END	-

รูปที่ 3.2 ตัวอย่างการเขียนโปรแกรมคำสั่งบูลลีน

3.1.3 ภาษาบล็อค

เป็นการเขียนโปรแกรมคำสั่งของ PC โดยใช้ชื่อสัญลักษณ์ต่าง ๆ คล้ายกับภาษาแลดเดอร์ แต่จัดไว้ในบล็อครูปสี่เหลี่ยมภาษาบล็อคนี้จะใช้คำสั่ง หรือควบคุมงานที่ซับซ้อน หรือมีข้อมูลที่

เป็นตัวเลขเกี่ยวข้อง เช่น การควบคุมที่คำนวณทางคณิตศาสตร์

คำสั่งภาษาบล็อคแบ่งออกเป็น 4 กลุ่มคำสั่งคือ

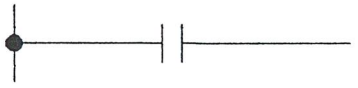
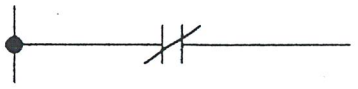
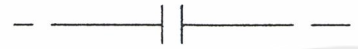
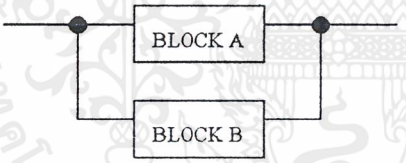
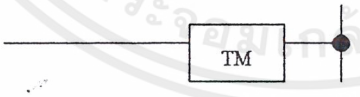
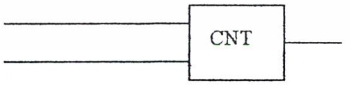
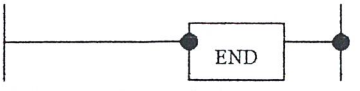
1. คำสั่งหน่วงเวลาและนับจำนวน
2. คำสั่งคำนวณทางคณิตศาสตร์
3. คำสั่งการจัดการข้อมูล
4. คำสั่งการเคลื่อนย้ายข้อมูล

รูปที่ 3.3 แสดงลักษณะตัวอย่างคำสั่งบล็อค

3.1.4 คำสั่งภาษาอังกฤษ

คำสั่งข้อความภาษาอังกฤษที่ใช้กับ PLC ถูกดัดแปลงจากภาษาระดับสูงของคอมพิวเตอร์ เช่น ภาษา BASIC ภาษา PASCAL PLC ที่ใช้คำสั่งข้อความภาษาอังกฤษส่วนใหญ่ใช้กับPLCขนาดใหญ่ มีการคำนวณที่ซับซ้อนและการจัดการข้อมูลจำนวนมาก

3.2 สรุปคำสั่งพื้นฐานของ PLC

INSTRUCTION	SYMBOL	MNEMONIC	DATA
LOAD		LD	POINT NO.
LOAD NOT		LD NOT	POINT NO.
AND		AND	POINT NO.
AND NOT		AND NOT	POINT NO.
OR		OR	POINT NO.
OR NOT		OR NOT	POINT NO.
AND LOAD		AND LD	-
OR LOAD		OR LD	-
OUT		OUT	POINT NO.
TIMER		TM	POINT NO. SET VALUE
COUNTER		CNT	POINT NO. SET VALUE
END		END	-

* POINT NO. = หมายเลขกำหนดอุปกรณ์ต่าง ๆ ของ PLC

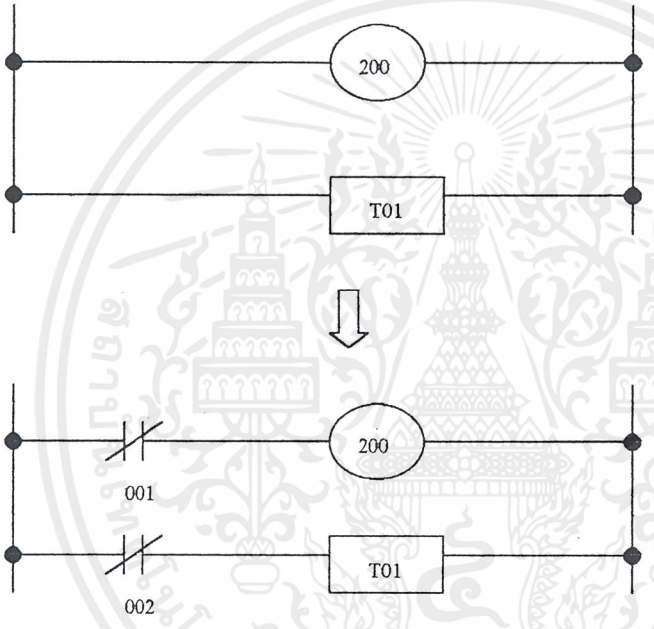
รูปที่ 3.4 สรุปคำสั่งพื้นฐานของ PLC

3.3 หลักการเขียนแลดเดอร์ไดอะแกรม(Ladder Diagram)

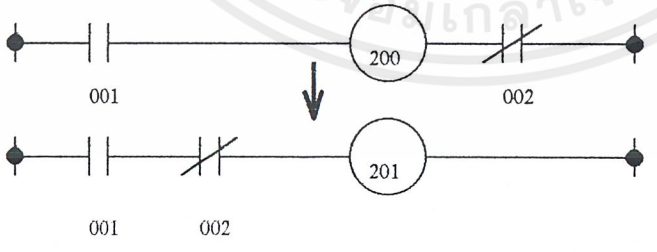
แลดเดอร์ไดอะแกรมเป็นโปรแกรมเป็นภาษาที่นิยมเขียนกันมากเพราะคล้ายกับวงจรรีเลย์เพียงแต่ต้องรู้

หลักการแลดเดอร์ไดอะแกรมเพิ่มเติมโดยมีหลักการเขียนดังต่อไปนี้

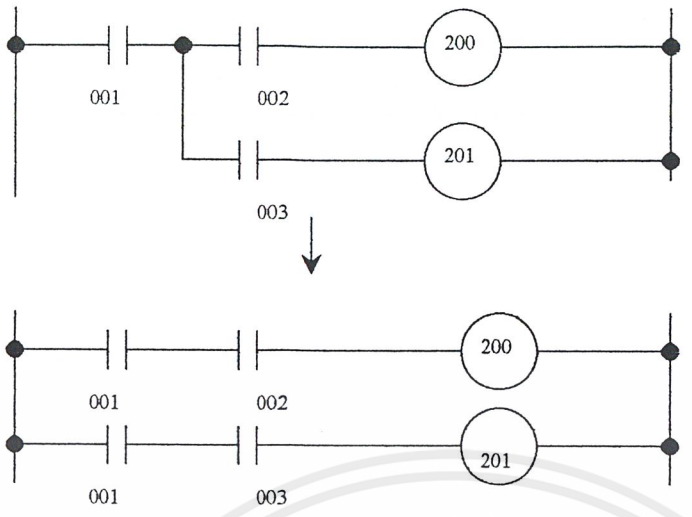
1. การกำหนดอินพุท เอาท์พุท รีเลย์ภายใน ตัวนับตัวตั้งเวลาหรือหมายเลขกำกับหน้าสัมผัสขึ้นอยู่กับพีแอลซีขึ้นอยู่แต่ละยี่ห้อ และคุณสมบัติของแต่ละรุ่นและยี่ห้อ
2. ในการเขียนโปรแกรมไม่สามารถเขียนโปแกรมเชื่อมต่อระหว่างบัส(Bus)กับคอยล์โดยตรงได้ในกรณีที่ต้องการต่อโดยตรงต้องใช้คำสั่งพิเศษหรือรีเลย์พิเศษช่วย



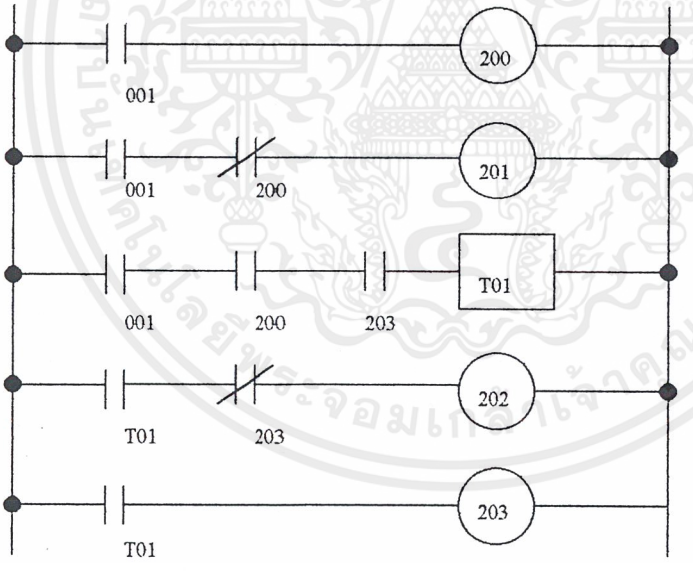
3. ตำแหน่งหน้าสัมผัสจะวางหลังรีเลย์ไม่ได้



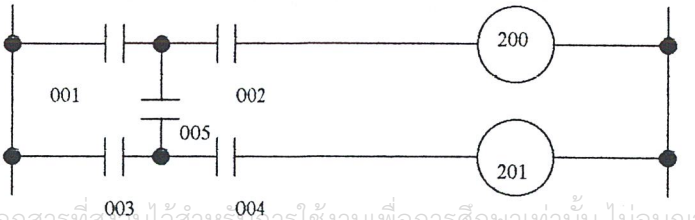
4. การเขียนโปรแกรมไม่ควรเขียนโปรแกรมให้สั้นแต่ซับซ้อนเพียงแค่ประหยัดหน่วยความจำเท่านั้นในการเขียนเขียนโปรแกรมนั้นต้องให้เข้าใจง่ายเพื่อสะดวกต่อการตรวจสอบ

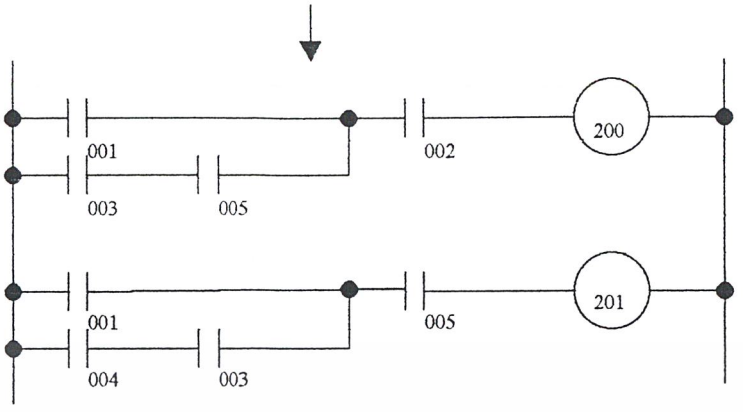


5. หน้าสัมผัสของอินพุท เอาตท์พุท รีเลย์ภายในตัวตั้งเวลา ตัวนับ หมายเลขเดียวกันนั้น สามารถใช้ในโปรแกรม ได้หลายครั้ง โดยไม่จำกัดจำนวนแต่ใช่หมายเลขเดียวกันในบรรทัดเดียวกันไม่ได้

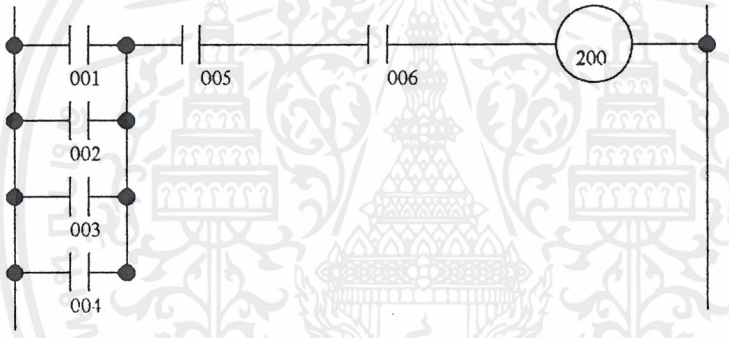


6. สัญญาณควบคุมจะไหลจากว้ายไปขวาเท่านั้นไม่สามารถไหลย้อนกลับได้

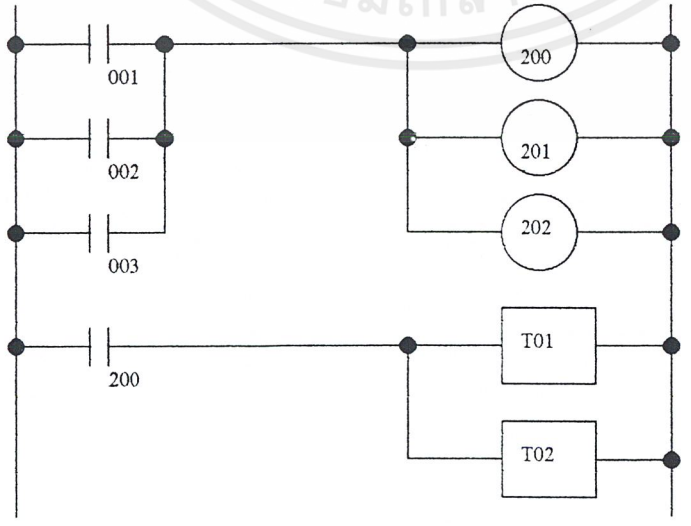




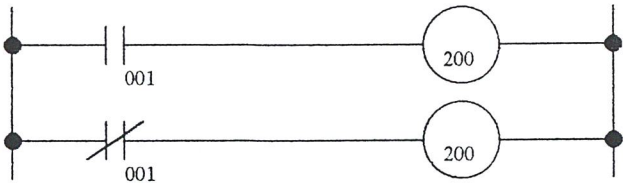
7. หน้าสัมผัสอินพุทเอาท์พุท ตัวตั้งเวลา ตัวนับ รีเลย์ภายใน จะนำมาขนานหรืออนุกรม จำนวนมากเท่าใดก็ได้ไม่จำกัด แต่อย่าใช้เบอร์เดียวกันขนานหรืออนุกรมกัน



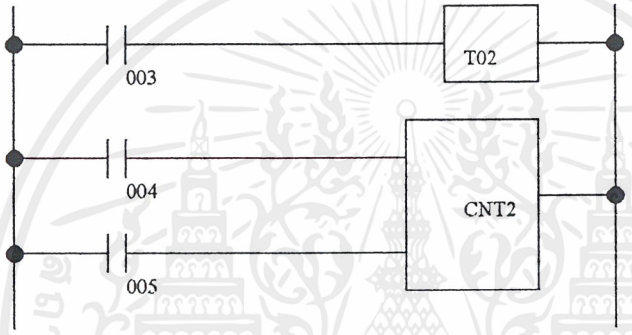
8. สามารถที่จะนำขดลวดเอาท์พุทรีเลย์ภายใน ตัวนับตัวตั้งเวลาขนานกันได้ แต่อย่าใช้เบอร์เดียวกันมาต่อขนานกัน



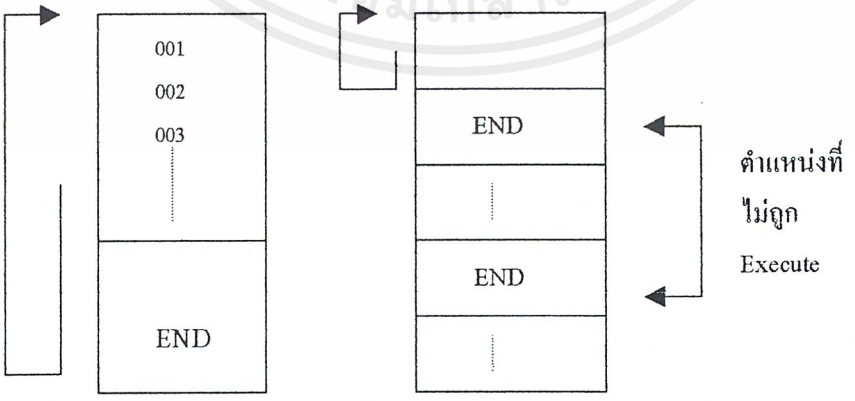
9. การต่อสัญญาณควบคุมมากกว่าหนึ่งครั้งไปยังเอาพุทที่เป็นหมายเลขเดียวกันไม่ได้ ถ้าต้องใช้ต้องนำฟังก์ชั่นพิเศษมาช่วยด้วย



10. ตัวตั้งเวลาและตัวนับเวลาใช้หมายเลขเดียวกันไม่ได้และไม่สามารถใช้หมายเลขเดียวกันมากกว่าหนึ่งครั้งต้องเปลี่ยนหมายเลขใหม่ที่ไม่ซ้ำกัน



11. โปรแกรมที่ได้เขียนขึ้น CPU จะทำการ Executed จากบรรทัดแรกถึงบรรทัดคำสั่ง END อาจมีหลายตำแหน่งก็เป็นไปได้ซึ่งต้องใช้คำสั่งหรือฟังก์ชั่นพิเศษเข้ามาช่วยที่เป็นเช่นนี้เพื่อจุดประสงค์สำหรับการทดลองโปรแกรม



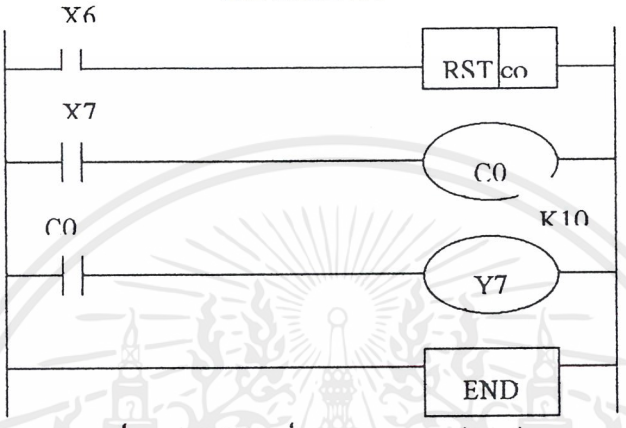
3.4 ตัวอย่างการเขียนโปรแกรมด้วยภาษาแลดเดอร์และบูลีน

- ตัวอย่างการใช้คำสั่ง RST กับ เคาท์เตอร์

ภาษาบูลีน

```
LD    X6
RST   C0
LD    X7
OUT   CO
SP    K10
LD    CO
OUT   Y7
```

ภาษาแลดเดอร์



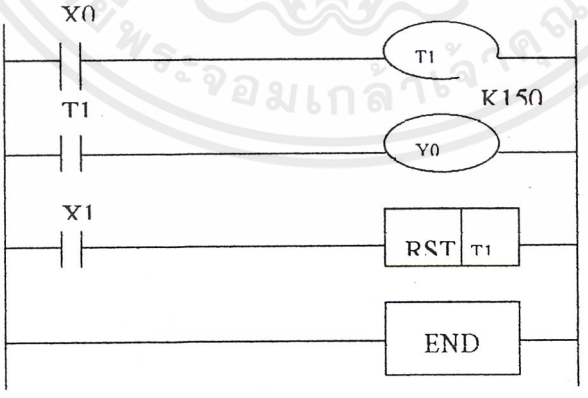
รูปที่ 3.5 การใช้คำสั่ง RST กับเคาท์เตอร์

- ตัวอย่างการใช้ RST กับ ไทม์เมอร์

ภาษาบูลลีน

```
LD    X0
OUT   T1
SP    K150
LD    T1
OUT   YO
LD    X1
RST   T1
END
```

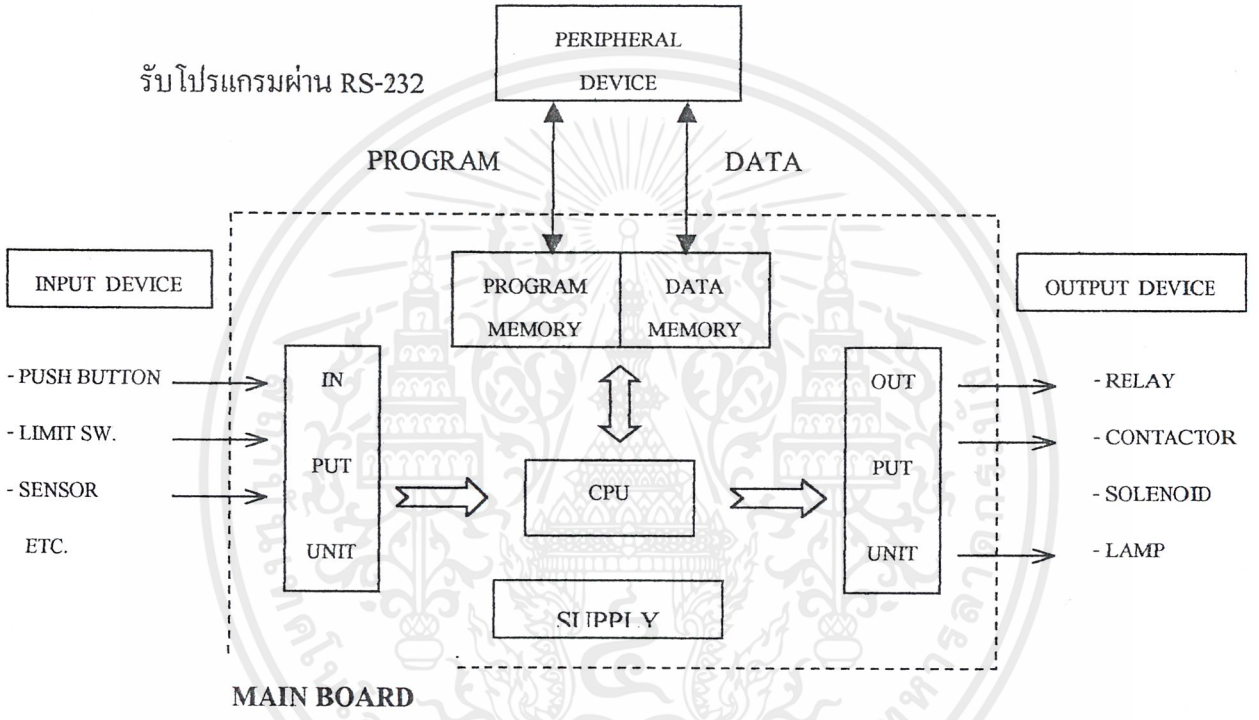
ภาษาแลดเดอร์ไดอะแกรม



รูปที่ 3.6 การใช้คำสั่ง RST กับ ไทม์เมอร์

บทที่ 4
ฮาร์ดแวร์แวร์ของโครงงานพีแอลซี

สำหรับโครงงานพีแอลซีที่ได้ทำการสร้างขึ้น โดยมีส่วนของประกอบหลักของโครงสร้างที่จะอธิบายแบ่งออกเป็น 2 ส่วน คือ ส่วนวงจรของหน่วยประมวลผลและหน่วยความจำ และส่วนของวงอินพุทและเอาท์พุท ซึ่งในการจัดทำได้รวมทั้งสองส่วนไว้บนบอร์ดหลัก (Main board) อธิบายแต่ละส่วนดังต่อไปนี้



รูปที่ 4.1 โครงสร้างของพีแอลซีที่สร้างขึ้น

หมายเหตุ - อุปกรณ์เอาท์พุท จะเป็นอุปกรณ์ที่ต่อได้ตามต้องการในการใช้งาน ซึ่งในวงจรส่วนเอาท์พุทจะเป็นหน้าสัมผัสของรีเลย์

- สำหรับโครงงานนี้จะใช้แหล่งจ่ายไฟเพียงชุดเดียว + 5 V ทั้งส่วนที่เป็นวงจรเอาท์พุทรีเลย์

4.1 บอร์ดหลักของวงจรหน่วยประมวลผลและหน่วยความจำ ตามวงจรรูปที่ 4.3

ประกอบไปด้วยส่วนของหน่วยประมวล(CPU) ที่ใช้ไมโครคอนโทรลเลอร์ตระกูล MCS –51 เบอร์ AT 89S8252 ของบริษัท Atmel ที่มี Flash Memory 8 Kbytes , EEPROM 2 Kbytes และมี RAM 256 Bytes

เครื่องพีแอลซี (PLC) ที่สร้างขึ้นอาศัยคุณสมบัติของชิพเบอร์ AT 89S8252 ที่มี PORTSPI ในตัว ซึ่งในการสร้างบอร์ดก็ได้ทำขั้วต่อลักษณะ 10 PIN (SPI PORT) เพื่อใช้ในการพัฒนาโปรแกรมที่สามารถทำได้ในรูปแบบที่ไม่ต้องถอดชิบเข้าออกเพื่อทำการโปรแกรมภายนอก ซึ่งจะมีความสะดวกกว่า หรือที่เรียกว่า " IN SYSTEM PROGRAMMING " คือ จะทำการโหลดข้อมูลจากเครื่องคอมพิวเตอร์ลงสู่ตัวชิบภายในบอร์ดพีแอลซีโดยตรง ทำให้การพัฒนาง่ายขึ้นและสะดวกยิ่งขึ้น โดยเครื่องโปรแกรมที่ใช้ในการพัฒนาคือ SPI – LOAD ของบริษัทศิลารีเสร์ช จำกัด

หน่วยความจำที่ใช้ในการเก็บโปรแกรม ขนาด 32 Kbytes ที่มีให้เลือก 2 แบบ คือ

- โวลาไทล์ (Volatile)หรือแรมธรรมดา เบอร์ 62256

ข้อดี คือ ราคาถูก

ข้อเสีย คือ ข้อมูลจะสูญหายทั้งหมดเมื่อไม่มีแหล่งจ่ายไฟเลี้ยง

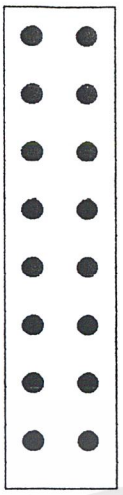
- นอนโวลาไทล์ (NonVolatile)หรือ NVRAM เบอร์ DS1230Y/AB

ข้อดี คือ ข้อมูลจะไม่สูญหายเมื่อไม่มีแหล่งจ่ายไฟเลี้ยงและอายุการใช้งานยาวนาน

ข้อเสีย คือ ราคาแพง

ไอซีเบอร์ 74LS373 ทำหน้าที่สำหรับพักข้อมูล (Latch) เนื่องจากระบบบัสแอดเดรสและบัสข้อมูลของไมโครคอนโทรลเลอร์ที่ใช้เป็นบัสเดียวกัน ดังนั้นจึงต้องใช้ลักษณะการมัลติเพล็กซ์สัญญาณจากพอร์ตเดียวกัน กล่าวคือ ในระยะเวลาเริ่มต้นจากสัญญาณจากพอร์ตจะใช้ในการส่งค่าแอดเดรสของตำแหน่งที่ต้องการติดต่อด้วยในช่วงเวลาต่อมาจึงเปลี่ยนเป็นสภาวะอิมพีแดนซ์สูงเพื่อใช้งานในฐานะของบัสข้อมูล แต่เนื่องจากหน่วยความจำที่ใช้กันทั่วไปนั้นไม่ใช้การมัลติเพล็กซ์และมีขาสัญญาณบัสแอดเดรสและบัสข้อมูลแยกจากกันโดยชัดเจน ดังนั้นในการเชื่อมต่อหน่วยความจำที่ต้องมีวงจรประเภทแลตซ์ (Latch)ประกอบเพิ่มเติมขึ้น เพื่อค้างค่าแอดเดรสที่ส่งจากไมโครคอนโทรลเลอร์ ในช่วงเวลาแรกให้กับขาสัญญาณแอดเดรสของหน่วยความจำต่อไป

ไอซีเบอร์ 74LS156 เป็นไอซีถอดรหัสเพื่อเลือกตำแหน่งเริ่มต้นและขนาดของหน่วยความจำแต่ละจุดของ Jumper ที่ใส่ลงไปจะมีขนาดของหน่วยความจำ 8 Kbytesเพิ่มจากจุดเริ่มต้น



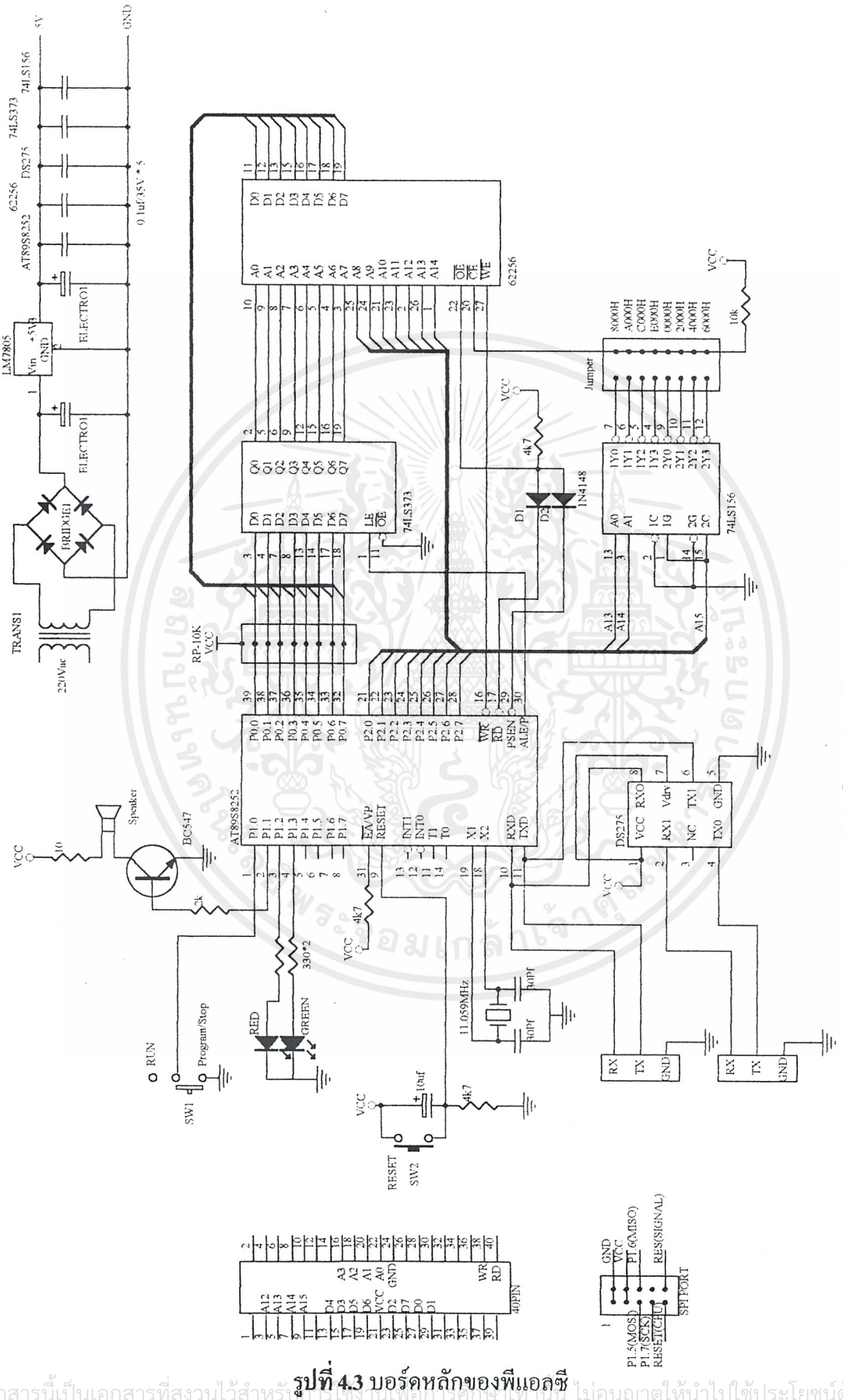
รูปที่ 4. 2 ตำแหน่งการเลือกคาเริ่มต้นและขนาดของหน่วยความจำ

ไดโอด D1 , D2 และความต้านทาน R- 4.7 K ถูกต่อในลักษณะของวงจร AND Gate แบบ 2 อินพุท คือขา RD และ PSEN ของไมโครคอนโทรลเลอร์ ส่งเอาท์พุทไปที่ขา OE ของหน่วยความจำแรม

ไอซีเบอร์ DS275 เป็นตัวเปลี่ยนระดับแรงดันของการสื่อสารแบบ RS – 232 เป็นแรงดันระดับ Digital Logic ที่ " 0 " คือ 0 โวลท์และ " 1 " คือแรงดันบวก 5 โวลท์ มีความสามารถเดียวกับแบบไอซี MAX 232 CPE แต่สามารถรับช่องการสื่อสารได้หนึ่งคู่คือ การส่งและการรับอย่างละชุดและการใช้งานง่ายกว่าที่ไม่ต้องเพิ่มส่วนของวงจรเสริมขึ้นมา

ภายในบอร์ดได้เพิ่มวงจรเสียง เพื่อตอบรับการโปรแกรม และมีสวิทซ์เลื่อน SW1 ทำการออกแบบไว้เพื่อทำงานสัมพันธ์กับ LED สีเขียวและสีแดง

ส่วนของพอร์ตแบบ 40 PIN เป็นส่วนในการติดต่อกับบอร์ดอื่น ๆในเครื่องพีแอลซีได้จัดวางตำแหน่งขาสัญญาณ เพื่อสามารถนำบอร์ดนี้ไปประยุกต์ใช้งานกับบอร์ดอื่นที่มีขายตามท้องตลาดหรือนำไปประยุกต์ใช้งานในเรื่องอื่น ๆ ได้อีก



รูปที่ 4.3 บอร์ดหลักของพีแอลซี

4.2 ส่วนวงจรอินพุทและเอาท์พุท ตามรูปที่ 4.4

ตามวงจรรูปที่ 4.4 สำหรับโครงงานนี้ได้ออกแบบให้ส่วนของอินพุทและเอาท์พุทอยู่บนบอร์ดเดียวกันโดยการทำงานของวงจรจะมีไอซี 74LS138*1 และ 74LS138*2 ทำหน้าที่เป็นไอซีถอดรหัส ซึ่งไอซี 74LS138*1 จะทำการตั้งค่าของตำแหน่งการถอดรหัสของสองไบท์แรก และไอซี 74LS138*2 จะทำการตั้งค่าของตำแหน่งการถอดรหัสสองไบท์หลัง รวมเป็น 4 ไบท์ หรือ 16 บิต โดยมีรายละเอียดการตั้งค่าตามที่เขียนไว้ในวงจรแล้ว ซึ่งตำแหน่งที่เลือกนั้นจะประกอบด้วย 8 อินพุท และ 8 เอาท์พุท

สัญญาณที่ส่งจากขาของไมโครคอนโทรลเลอร์ WR (ขา 16) และ RD (ขา 17) จะเป็นตัวแยกว่าจะรับค่าสัญญาณอินพุท หรือจะส่งสัญญาณเอาท์พุท

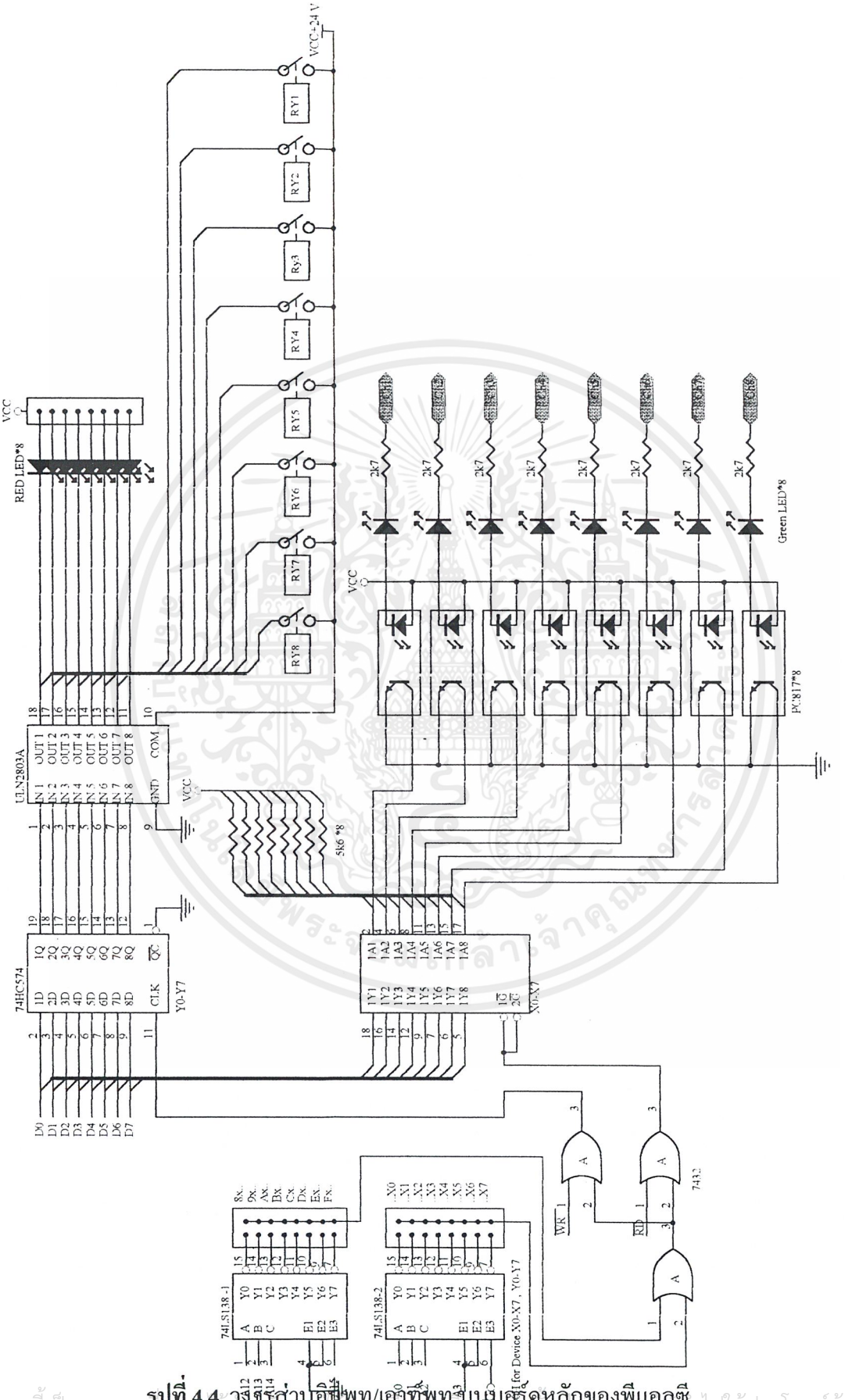
ส่วนของวงจรอินพุท คือ วงจรออปโต้คอบเปเลอร์ 8 วงจร ออปโต้คอปเปเลอร์เป็นอุปกรณ์ที่มีการแบ่งส่วนอินพุทและเอาท์พุทออกจากการทำงานทางไฟฟ้าโดยสิ้นเชิงการถ่ายทอดสัญญาณระหว่างส่วนอินพุทและเอาท์พุทจะใช้การเชื่อมโยงทางแสงเท่านั้น ทำให้กราวด์ของอินพุทและเอาท์พุทไม่เชื่อมต่อกัน สำหรับในการใช้งานจริงจะใช้วงจรแหล่งจ่ายไฟแยกต่างหากกัน แต่ในโครงงานนี้ได้ใช้แหล่งจ่ายไฟเพียงชุดเดียวคือ + 5 โวลท์ เพื่อความสะดวกในการทดลอง และการจัดสร้างก็ง่ายขึ้น ซึ่งสามารถใช้งานได้จริง

ส่วนวงจรเอาท์พุท คือ วงจรรีเลย์ 8 ชุด วงจรรีเลย์จะใช้ไอซีเบอร์ ULN 2803 เป็นไอซีไดรเวอร์ ซึ่งคุณสมบัติของไอซีเบอร์นี้มีความสามารถสูงกว่า 74LS06 และ 74LS07 โดยภายในบรรจุอินเวอร์เตอร์เกตแบบคอลเล็กเตอร์เปิด 8 ตัว สามารถใช้กับแรงดันสูงสุด 50 โวลท์ และทนกระแสสูงสุด 500 มิลลิแอมป์ สำหรับในวงจรนี้ได้ออกแบบมาเพื่อรับอินพุทขนาด 5 โวลท์แบบ TTL หรือ CMOS นอกจากนั้นภายในไอซีเองยังต่อไดโอดป้องกันไว้ทุกขาเอาท์พุท สามารถต่อโหลดที่เป็นขดลวดได้ทันที อีกทั้งยังลดความยุ่งยากของการต่อวงจรขับที่เป็นแบบทรานซิสเตอร์ลงได้

จกวงจรอินพุทและวงจรเอาท์พุท ของพีแอลซีได้ออกแบบอุปกรณ์เชื่อมต่อกับอุปกรณ์ภายนอกตัวพีแอลซี การรับค่าอินพุทเป็นการเชื่อมโยงด้วยแสงโดยใช้ออปโต้คอปเปเลอร์ และการส่งค่าอาท์พุทจะใช้รีเลย์ที่ตัดต่อหน้าสัมผัส ซึ่งจะทำให้กราวด์อินพุทและเอาท์พุทแยกออกจากกันและจะเป็นการป้องกันการเสียหายแก่ตัวพีแอลซีได้เป็นอย่างดี

วงจรทั้งอินพุทและเอาท์พุท มีการแสดงผลด้วย LED

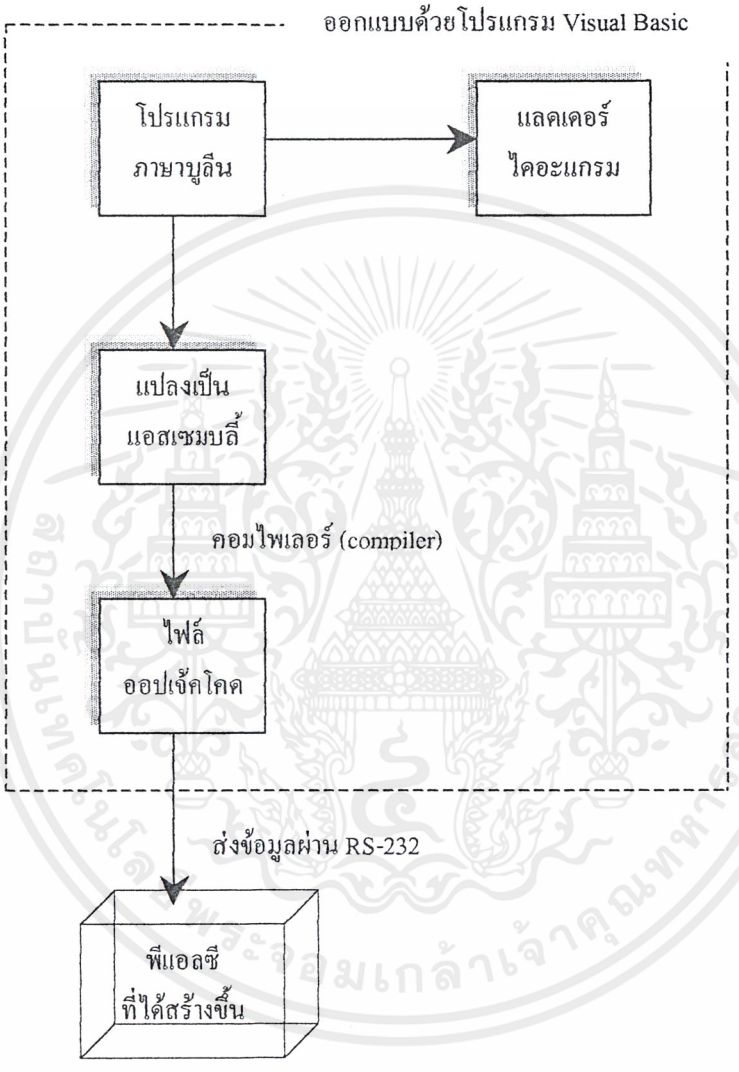
สำหรับการขยายบอร์ดอินพุทและเอาท์พุทจะออกแบบในลักษณะเดียวกับวงจรในส่วนของอินพุทและเอาท์พุท โดยใช้ไอซี 74LS138 เป็นตัวถอดรหัสและใช้ Jumper เลือกตำแหน่งแอดเดรสของอินพุทและเอาท์พุท



รูปที่ 4.4 วงจรส่วนอินพุท/เอาท์พุทบนบอร์ดหลักของพีแอลซี

บทที่ 5
ซอฟต์แวร์ที่ใช้ในการควบคุมพีแอลซีของโครงงาน

5.1 ขั้นตอนการออกแบบซอฟแวร์



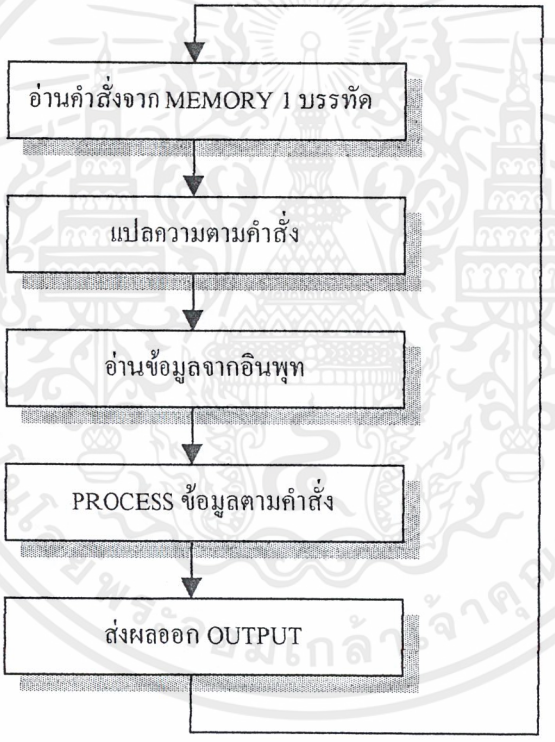
รูปที่ 5.1 แสดงขั้นตอนการออกแบบซอฟแวร์ที่ควบคุมพีแอลซี

จากรูปที่ 5.1 ในส่วนของเส้นประเป็นส่วนของซอฟแวร์ที่สร้างขึ้นบนคอมพิวเตอร์ โดยใช้โปรแกรม Visual Basic ในการออกแบบวินโดว์ที่ติดต่อกับผู้ใช้ และออกแบบการใช้คำสั่งภาษาบูลีน หลังจากนั้นต้องนำรูปแบบคำสั่งของภาษาบูลีน มาแปลงเป็นภาษาแอสเซมบลี้ (ASM) และนำโปรแกรมแอสเซมบลี้ที่ได้ไปทำการคอมไพล์ (Compiler) เพื่อให้ได้เป็นไฟล์ออปเจ้คโค้ด เมื่อเสร็จจากการคอมไพล์เป็นออปเจ้คโค้ดซึ่งเป็นภาษาที่เครื่องพีแอลซีเข้าใจ แล้วซอฟแวร์ที่ออกแบบขึ้นมา

สามารถที่จะจัดส่งข้อมูลที่เป็นไฟล์ออปเจ้คโค้ดหรือตัวโปรแกรมที่เป็นคำสั่งควบคุมพีแอลซีลงไปบนหน่วยความจำบนเครื่องพีแอลซีได้โดยมาตราฐานการสื่อสารของ RS-232

ในโปรแกรมที่ออกแบบขึ้นในโครงงานนี้ ผู้ใช้สามารถเขียนคำสั่งควบคุมพีแอลซีได้ด้วยภาษาบูลีน โดยเขียนลงบนโปรแกรม Notepad ทำให้ง่ายและสะดวกต่อการใช้งานซึ่งในการจัดเก็บข้อมูลนี้จะจัดเก็บในรูปแบบของเท็กซ์ไฟล์ คือ จัดเก็บเป็นนามสกุล ".txt" และระหว่างทำการเขียนโปรแกรมของพีแอลซีด้วยภาษาบูลีนนั้นสามารถทำการเรียกดูลักษณะการต่อวงจรในรูปแบบของแลดเดอร์ไดอะแกรม (Ladder Diagram) เพื่อเปรียบเทียบกับคำสั่งที่เขียนขึ้นโดยภาษบูลีน (Boolean language) และง่ายแก่การตรวจสอบหรือทำความเข้าใจ

5.2 กระบวนการของไมโครคอนโทรเลอร์ที่ดำเนินการตามภาษาพีแอลซี

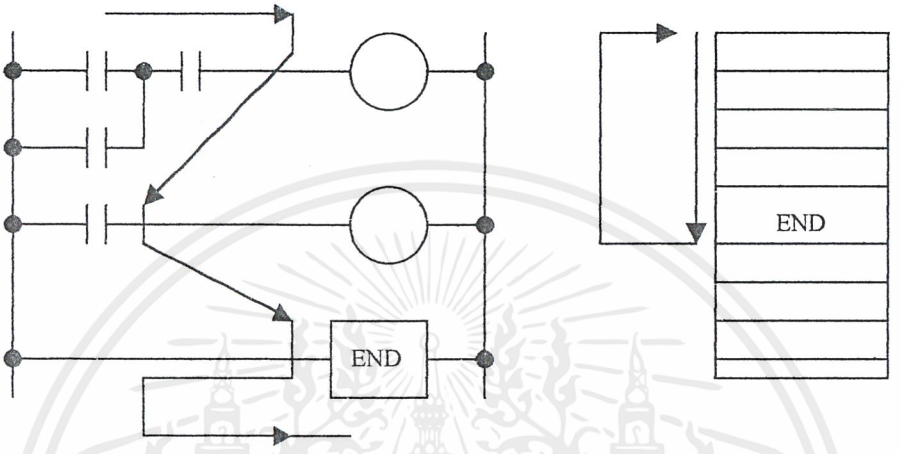


รูปที่ 5.2 FLOW CHART แสดงการทำงานของ CPU

กระบวนการทำคำสั่งของไมโครคอนโทรเลอร์ที่ดำเนินการภาษาพีแอลซี ซึงเป็นไปตามขั้นตอนในรูปที่ 5.2 หลังจากที่โหลดโปรแกรมคำสั่งลงบนเครื่องพีแอลซีแล้ว หน่วยประมวลผลกลางหรือซีพียู (CPU) จะทำหน้าที่ควบคุมกาทำงานของพีแอลซี โดยปกติหน้าที่ของซีพียู คือ รับข้อมูลอินพุทเข้ามาประมวลผล แล้วส่งผลที่ได้ออกไปยังหน่วยเอาท์พุท จากนั้นก็จะวนกลับไปรับข้อมูล

อินพุทเข้ามาอีกแล้วทำซ้ำๆ ในลักษณะนี้ไปเรื่อยๆ ซึ่งเรียกว่า การสแกน หรือการทำงานหนึ่งวงรอบเรียกว่า เวลาสแกน (scan time) บางครั้งเรียกว่า เวลาหนึ่งไซเกิล (cycle time)

การสแกนของซีพียูประกอบด้วยการรับค่าของสภาวะอุปกรณ์ภายนอกจากอินพุท/เอาท์พุทมาเก็บไว้ในหน่วยความจำ หลังจากนั้นจะนำโปรแกรมที่ผู้ใช้เขียนขึ้นมาปฏิบัติทีละคำสั่ง โดยเริ่มต้นจากคำสั่งแรกจนสิ้นสุดโปรแกรมในหน่วยความจำ



รูปที่ 5.3 การสแกนโปรแกรมคำสั่งในพีแอลซี

แผนผังกระบวนการของโปรแกรมพีแอลซีจะเริ่มอ่านคำสั่งในโปรแกรมเริ่มจากบรรทัดที่ 1 ของภาษามูลลีนจนถึงบรรทัดที่เป็นคำสั่ง END ก็จะออกไปที่กระบวนการของการส่งค่าเอาท์พุท ซึ่งในแต่ละบรรทัดของคำสั่งจะมีการอ่านค่าอินพุทหรือการส่งค่าเอาท์พุท จะใช้การพักไว้ในหน่วยความจำแบบชั่วคราว (RAM) โดยจะกำหนดไว้ตามที่ได้แบ่งส่วนของหน่วยความจำหรือ “Map Memory”

5.3 รูปแบบของไฟล์ในการสื่อสาร

รายละเอียดของไฟล์มีดังนี้ในการส่งโปรแกรมผ่านการสื่อสารตามมาตราฐาน RS-232 นั้น โปรแกรมที่เขียนไว้ในตัวไมโครคอนโทรเลอร์ จะจัดการเฉพาะรูปแบบของ INTEL-HEX FILE ซึ่งเป็นไฟล์มาตราฐานอันหนึ่ง ทั้งนี้เนื่องจากไฟล์นี้มีรูปแบบเหมาะสมหลายประการ กล่าวคือมีการกำหนดตำแหน่งแอดเดรสของข้อมูลได้ และมีระบบการตรวจสอบผลรวม ที่สามารถตรวจสอบความถูกต้องของข้อมูลได้ และที่สำคัญเป็นไฟล์แบบแอสกี้ (ASCII) คือสามารถใช้ในการส่งออกทางพอร์ตสื่อสารต่างๆ ได้ (ไฟล์ที่จะส่งออกทางพอร์ตสื่อสารจะต้องเป็นไฟล์แบบแอสกี้เท่านั้นทั้งนี้เนื่องจากการสื่อสาร จำเป็นต้องมีรหัสในการสื่อสารเองอยู่แล้ว จึงไม่สามารถใช้ไฟล์ของชิ้นงาน (Object) ได้ เพราะข้อมูลอาจจะไปซ้ำกับรหัสการสื่อสารได้)

:BCAAAATTHH.....HHCC

: คือ ตัวอักษรของการเริ่มบรรทัด เป็นรหัสแอสกี้

BC คือ จำนวนไบท์ของข้อมูลในบรรทัด มีค่าเป็นเลขฐาน 16 (HEX)
ถ้า BC = 00 จะเป็นบรรทัดสุดท้ายของการบันทึกไฟล์

AAAA คือ ตำแหน่งแอดเดรสของข้อมูลในไบท์แรก

TT คือ ชนิดของข้อมูลในบรรทัดนั้นๆ
ถ้า TT = 00 เป็นการบันทึกข้อมูล
ถ้า TT = 01 เป็นบรรทัดสุดท้ายของการบันทึกไฟล์

HH คือ ข้อมูลในแต่ละไบท์

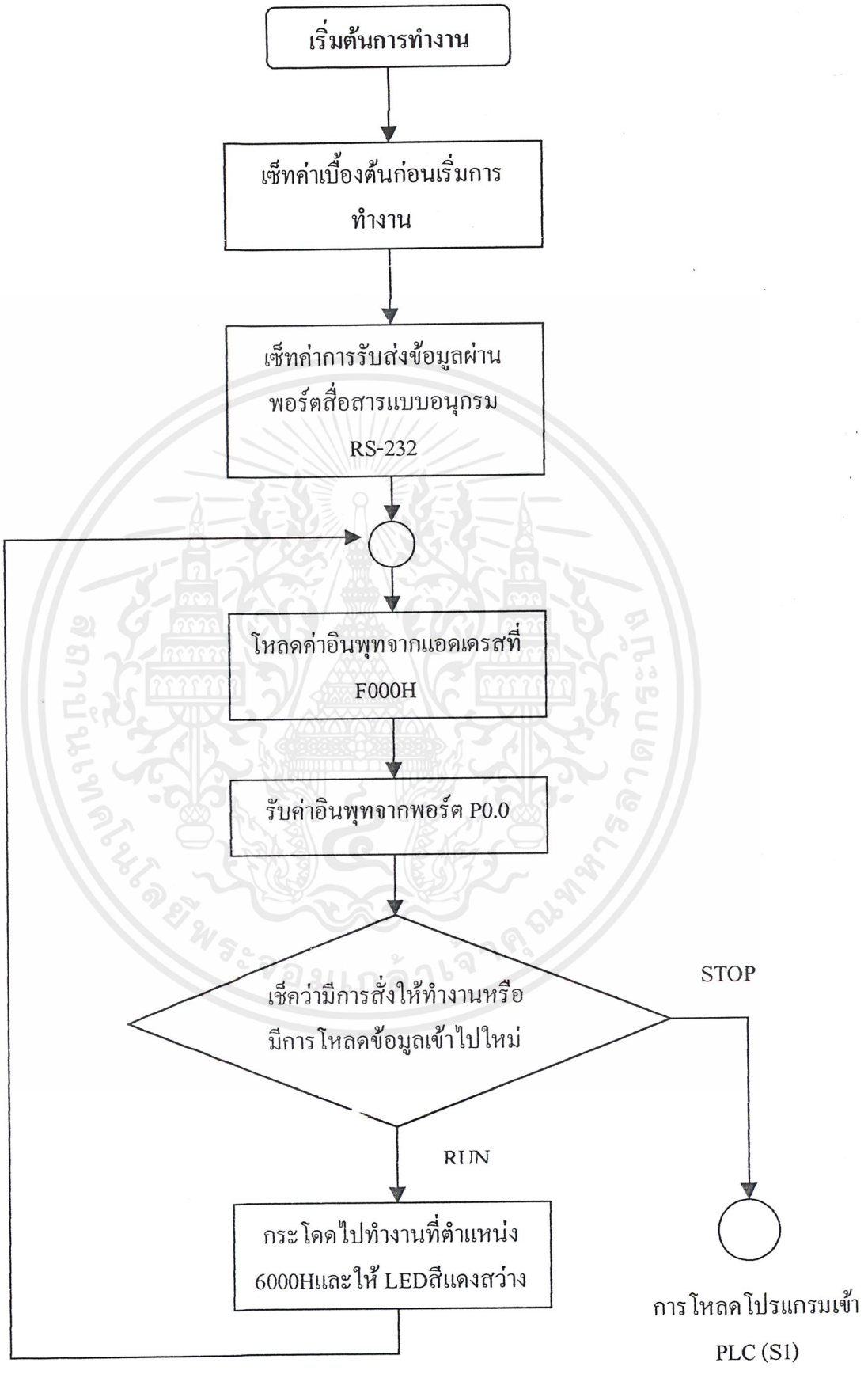
CC คือ ค่าในการตรวจสอบผลรวมของบรรทัดนั้นๆ โดยจะเป็นค่า 2'S COMPLEMENT ของผลบวกของข้อมูลทุกๆ ไบท์ในบรรทัด ซึ่งรวม BC, AAAA, TT และ HH

สำหรับซอฟแวร์ที่ได้สร้างขึ้นเมื่อทำการคอมไพล์เป็นไฟล์ออปเจ้คโค้ดแล้วจะทำการจัดส่งข้อมูลโดยกำหนดรูปแบบการส่งเป็นลักษณะตามที่กล่าวมา ยกเว้น CC ซึ่งไม่จำเป็น สามารถตัดออกได้เพื่อความสะดวกและลดความยุ่งยากในการจัดรูปแบบของไฟล์ และในการออกแบบโปรแกรม

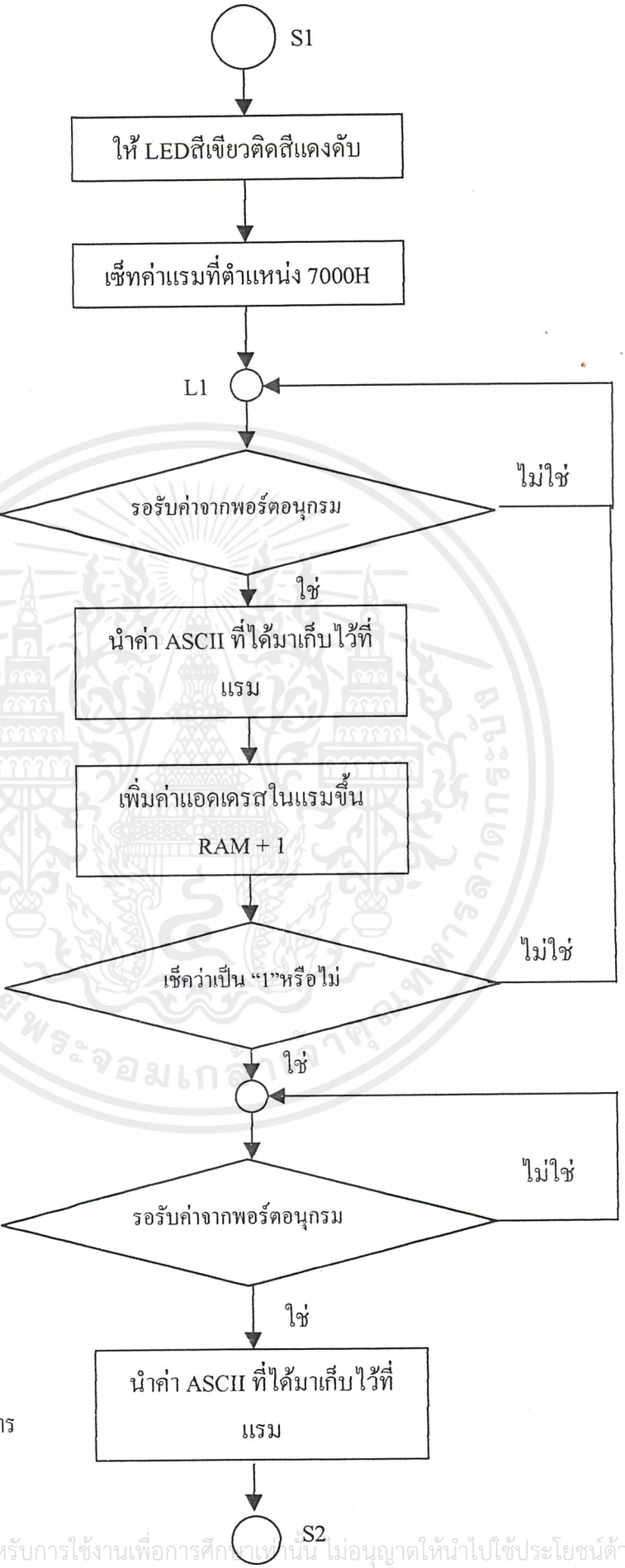
ในส่วนของโปรแกรมหลักจะทำการตรวจสอบข้อมูลที่รับเข้ามาเป็นรหัสแอสกี้ " : " และแปลงเป็น Hex Code ซึ่งจะได้ค่าเท่ากับ 3A ในฐาน16 ก่อนเท่านั้น จากนั้นจึงจะเริ่มตรวจสอบตำแหน่งของหน่วยความจำที่โปรแกรมต้องการจะจัดเก็บที่ไบท์ของ AAAA เป็นอันดับต่อมา จาก

นั้นจะนำข้อมูลซึ่งเป็นตัวโปรแกรมที่ใช้ควบคุมพีแอลซีที่ผู้ใช้เขียนขึ้น HH ลงบนหน่วยความจำแรม (RAM)

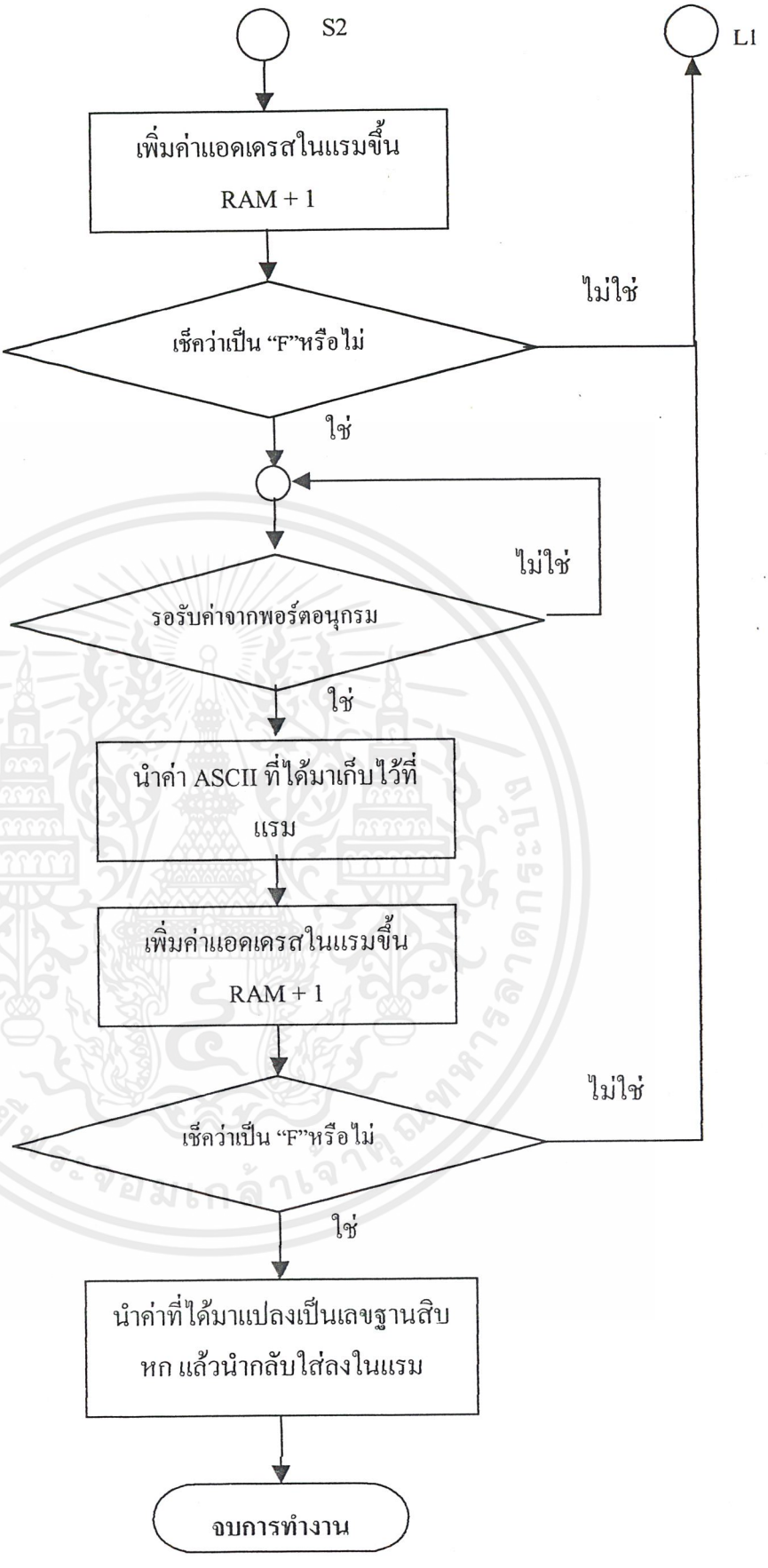
5.4 โปรแกรมมอนิเตอร์ บนหน่วยความจำของไมโครคอนโทรเลอร์



รูปที่ 5.4 Flow Chart แสดงการทำงานโปรแกรมมอนิเตอร์



รูปที่ 5.4 Flow Chart แสดงการโปรแกรมมอนิเตอร์ (ต่อ)



รูปที่ 5.4 Flow Chart แสดงการทำงานโปรแกรมมอนิเตอร์ (ต่อ)

ในบอร์ดทดลองพีแอลซีนี้ใช้ไมโครคอนโทรเลอร์เป็นตัวควบคุม โดยแยกส่วนโปรแกรมออกเป็นสองส่วน คือ โปรแกรมมอนิเตอร์และโปรแกรมของผู้ใช้ ซึ่งโปรแกรมมอนิเตอร์นี้เป็นโปรแกรมที่จะกำหนดว่า ให้โปรแกรมทำงานตามคำสั่งของผู้ใช้ หรือรับค่าโปรแกรมของพีแอลซีจากคอมพิวเตอร์ เพื่อใส่ข้อมูลลงในแรม ซึ่งจะถูกกำหนดโดย SW1 โดยมีบอร์ดเรทในการรับส่งเป็น 9600 bps โดยค่าที่ทำการรับส่งนั้นเป็น ascii ซึ่งไม่สามารถนำมาใช้ได้โดยตรง ต้องนำค่าข้อมูลที่ได้ทำการเป็นเลขฐานสิบหกแล้วนำไปเก็บไว้ที่ตำแหน่งแอดเดรส 6000H ขึ้นไป

การใช้สวิทช์ควบคุมการทำงานของเครื่องพีแอลซี SW.1 เป็นแบบโยก คือเมื่อ สับสวิทช์ลง แอลอีดี (LED) สีแดงจะติด คือ ตำแหน่งการ “RUN” คือการดำเนินการตามภาษาพีแอลซี และเมื่อสับสวิทช์ขึ้น แอลอีดี (LED) สีเขียวจะติด คือ ตำแหน่ง “STOP” คือ หยุดการทำโปรแกรมตามภาษาพีแอลซีโดยไม่ลบข้อมูลใด และเมื่อเลื่อนกลับมาที่ตำแหน่ง “RUN” พีแอลซีก็จะทำงานต่อไป

เมื่อต้องการโปรแกรมการทำงานของพีแอลซีใหม่ จะต้องทำการสับสวิทช์ SW.1 มาที่ตำแหน่ง “STOP” ก็เท่ากับว่าสามารถส่งข้อมูลจากคอมพิวเตอร์มาที่พีแอลซีได้ และเมื่อเสร็จสิ้นการส่งข้อมูล ก็จะมีเสียงตอบรับจากลำโพง 2 ครั้ง เป็นอันเสร็จสมบูรณ์ และสามารถใช้โปรแกรมที่พึ่งโหลดมาได้ให้พีแอลซีทำงาน โดยสับสวิทช์ให้อยู่ตำแหน่ง “RUN” และกด SW.2 หนึ่งครั้ง แอลอีดี (LED) สีแดงจะติด

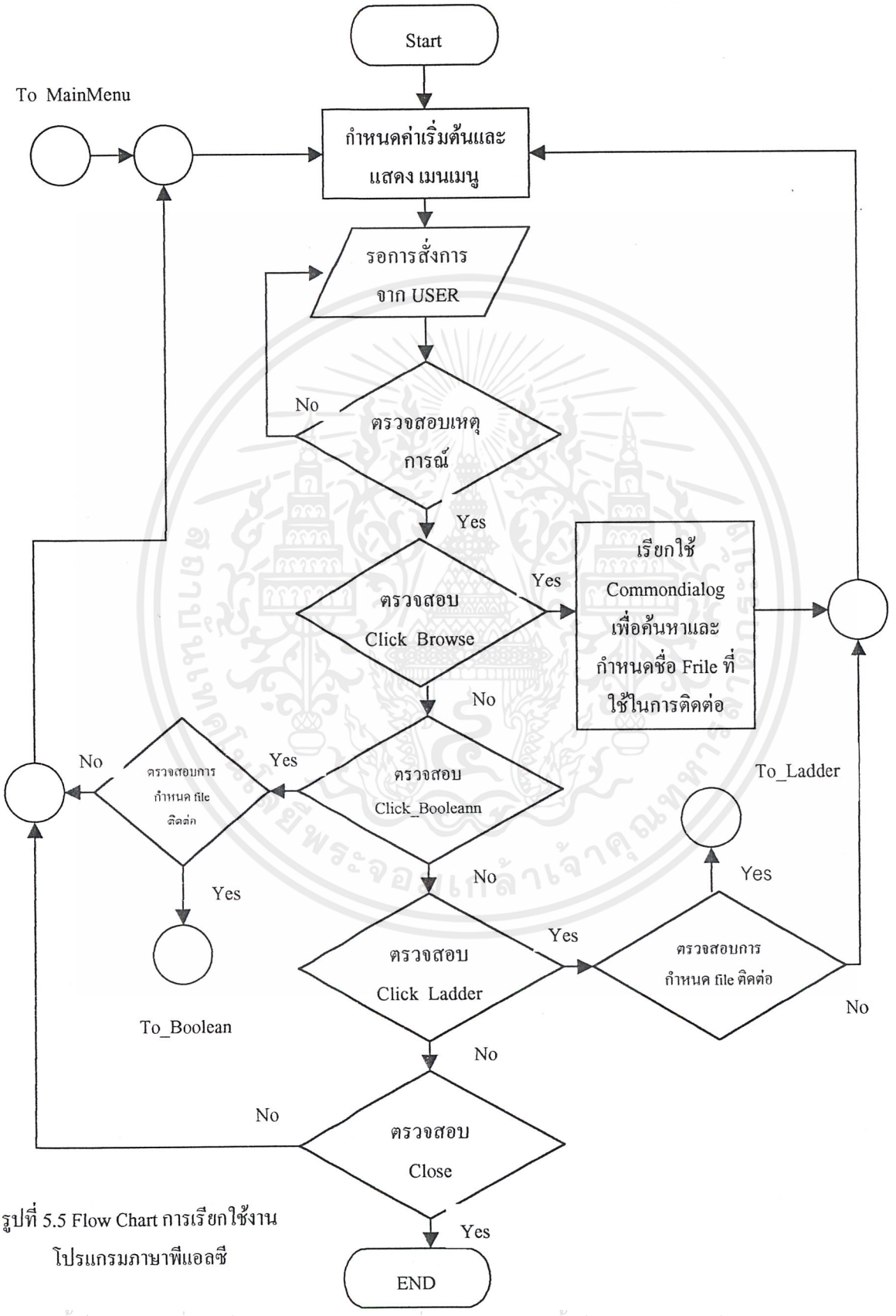
5.5 การจัดแบ่งพื้นที่ของหน่วยความจำของเครื่องพีแอลซี (Map Memory)

ค่าโครงสร้างรีจีสเตอร์ภายในของPLC

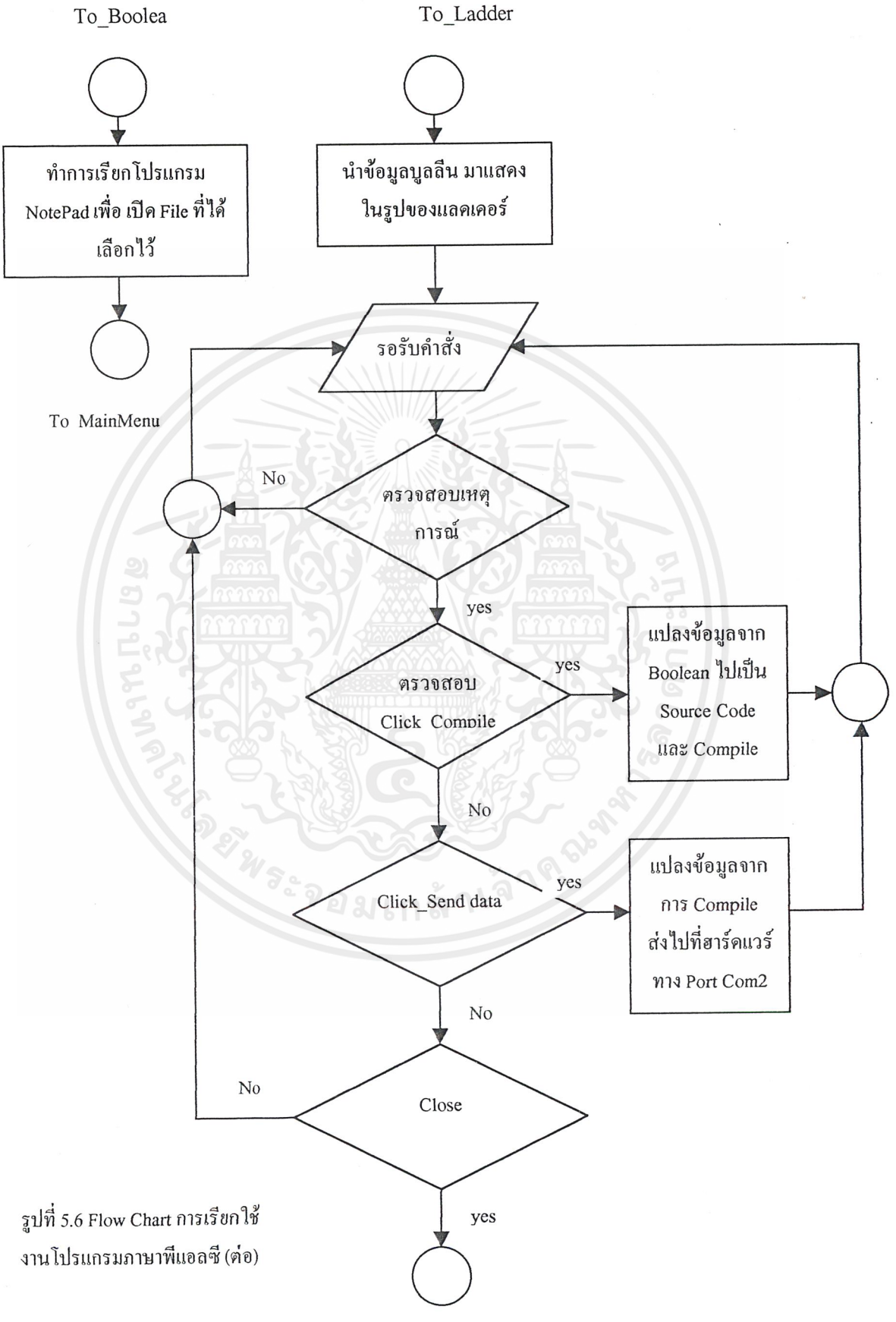
แอดเดรส	สถานะ
0000 – 1fff	ค่าแอดเดรสโปรแกรมคุมฮาร์ดแวร์
6000 – 6fff	ค่าแอดเดรสส่วนของตัวโปรแกรม
7000 – 77ff	ค่าแอดเดรสของรีจิสเตอร์ต่าง ๆ ของ PLC
7800	ค่าแอดเดรสของ อินพุท (x0 - 7)
7801	ค่าแอดเดรสของเอาพุท (y0 – 7)
7802	ค่าแอดเดรสของรีเลย์ช่วย(m0 – m7)
7803 – 7806	ค่าแอดเดรสของรีเลย์ช่วย(m8 – m40)
7807	เก็บสถานะ constact ของ counter (c0 – 7)
7808	เก็บสถานะการกดสวิตท์
7809 – 7810	เก็บสถานะค่าอ้างอิงในการนับ (K)ของ counter (PLC
7811	เก็บสถานะ constact ของ counter (c8 – 15)
7812	เก็บสถานะการกดสวิตท์
7813 – 781a	เก็บสถานะค่าอ้างอิงในการนับ (K)ของ counter (c8 – c15)
781b	เก็บสถานะ constact ของ timer (t0 – 7)
781c – 7823	เก็บสถานะค่าเวลาของไทเมอร์ ของค่า th (t0 – 7)
7823 – 782a	เก็บสถานะค่าเวลาของไทเมอร์ ของค่า tl (t0 – 7)
782b	เก็บสถานะ constact ของ timer (t8 – 15)
782c – 7833	เก็บสถานะค่าเวลาของไทเมอร์ ของค่า th (t8 – 15)
7833 – 784a	เก็บสถานะค่าเวลาของไทเมอร์ ของค่า tl (t8 – 15)
*f000	ค่าแอดเดรสของอินพุทและเอาท์พุท

ตารางที่ 5.1 การจัดพื้นที่หน่วยความจำแมโมรี่แม๊พ

5.6 แผนผังแสดงซอร์แวร์ที่สร้างขึ้นบนเครื่องคอมพิวเตอร์



รูปที่ 5.5 Flow Chart การเรียกใช้งาน
โปรแกรมภาษาพีแอลซี



รูปที่ 5.6 Flow Chart การเรียกใช้งานโปรแกรมภาษาพีแอลซี (ต่อ)

5.7 อุปกรณ์และเบอร์อุปกรณ์ ในพีแอลซีที่ทำจะมีอุปกรณ์ต่างๆดังนี้

X : อินพุทออปโต้คอปเปลอร์ (Input Optocoupled)

เป็นอุปกรณ์ออปโต้สำหรับใช้รับสัญญาณจากสวิทช์ตรวจจับ ซึ่งต่อเข้ามาที่ขั้วเข้าของพีแอลซี

เบอร์ของอินพุทออปโต้คอปเปลอร์นี้จะนำหน้าด้วย X เสมอ

Y : เอาท์พุทรีเลย์ (Output Relay)

เป็นรีเลย์ขาออกที่ใช้ในการขับโหลดที่ต่อกับขั้วของพีแอลซี เอาท์พุทรีเลย์จะมีเบอร์นำหน้าด้วย Y เสมอ

M : รีเลย์ช่วย (Auxiliary Relay)

เป็นรีเลย์ที่ใช้ในการสร้างวงจรควบคุมซีเคว้นซ์ ภายในพีแอลซีจะจัดเตรียมรเลย์ช่วยไว้มาก เบอร์ของรีเลย์ช่วยนี้จะนำหน้าด้วย M เสมอ

อุปกรณ์	เบอร์	จำนวน
X	0-7	8
Y	0-7	8
M	0-7	8

ตารางที่ 5.2 อุปกรณ์หลักของเครื่องพีแอลซี

บทที่ 6
การทดลองการทำงานของเครื่องพีแอลซี

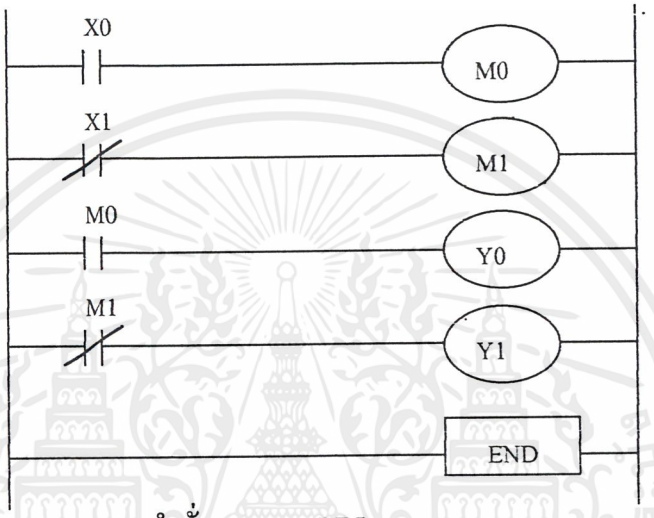
6.1 การทดลองการทำงานของเครื่องพีแอลซีตามคำสั่งต่างๆ ดังนี้

6.1.1 การทดสอบการใช้คำสั่ง LD และ LDI

ภาษาบูลีน

```
LD    X0
OUT   MO
LDI   X1
OUT   M1
LD    M0
OUT   Y0
LDI   M1
OUT   Y1
END
```

ภาษาแลดเดอร์

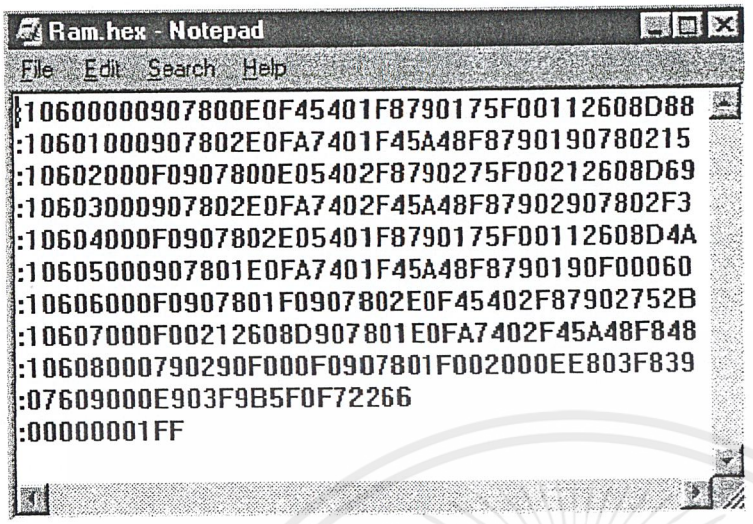


รูปที่ 6.1 วงจรทดสอบคำสั่ง LD และ LDI

การทำงาน

- LD และ LDI เป็นคำสั่งอ่านหน้าสัมผัส ใช้อ่านหน้าสัมผัสอินพุทออปโต้ X, เอาท์พุทรีเลย์ Y และรีเลย์ช่วย M
- คำสั่งเอาท์ (OUT) เป็นคำสั่งขับคอยล์ ใช้ขับคอยล์ของอุปกรณ์ได้ทุกชนิด ยกเว้นอินพุท X
- เมื่อ X0 ON จะทำให้ M0 ทำงาน เป็นผลให้ Y0 ทำงานด้วย
- เมื่อ X1 ON จะทำให้ M1 หยุดทำงาน เป็นผลให้ Y1 หยุดทำงานด้วย

ไฟล์ออปเจ้คโค้ด

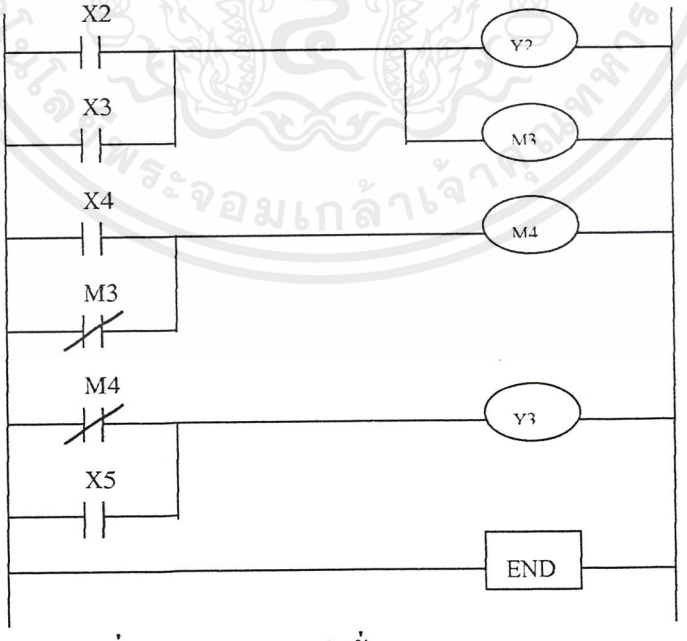


6.1.2 การทดสอบการใช้คำสั่ง OR และ ORI

ภาษาบูลลีน

```
LD    X2
OR    X3
OUT   Y2
OUT   M3
LD    X4
ORI   M3
OUT   M4
LDI   M4
OR    X5
OUT   Y3
END
```

ภาษาแลดเดอร์

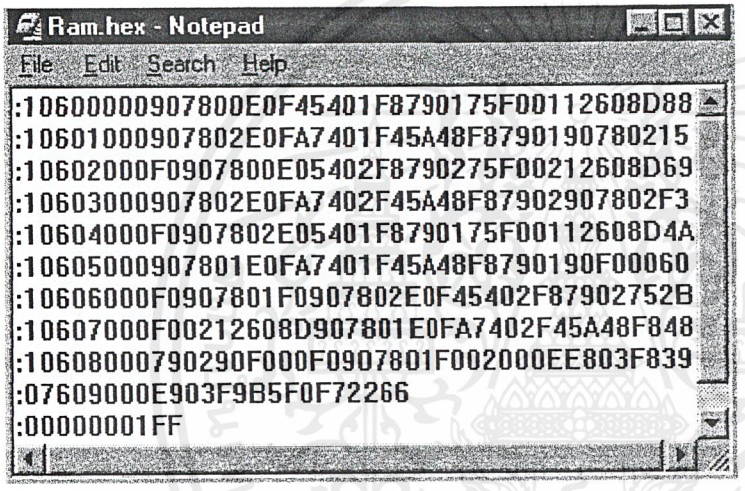


รูปที่ 6.2 วงจรทดสอบคำสั่ง OR และ ORI

การทำงาน

- เมื่อ X2 หรือ X3 ON จะทำให้ Y2 กับ M3 ทำงาน เป็นผลทำให้ M4 หยุดทำงาน และ Y3 ก็จะทำงาน
- เมื่อ X4 ON จะทำให้รีเลย์ช่วย M4 ทำงาน เป็นผลทำให้ Y3 หยุดทำงาน
- เมื่อ X5 ON จะทำให้ Y3 ทำงาน โดยไม่ขึ้นอยู่กับรีเลย์ช่วย M4

ไฟล์ออปเจ้คโค้ด

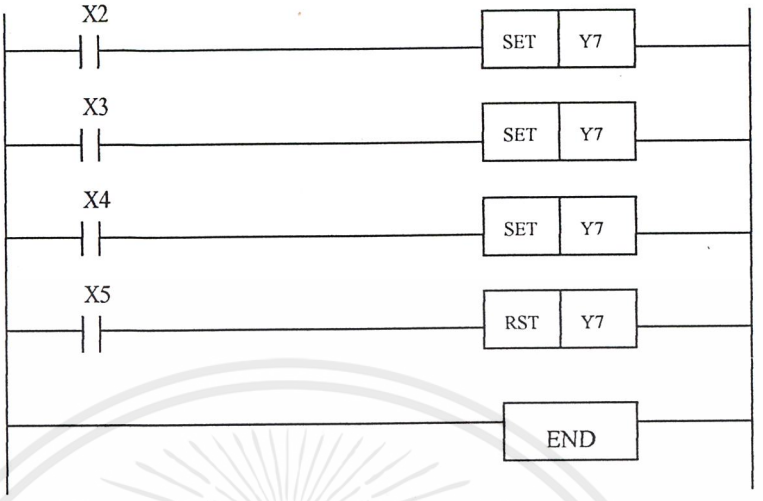


6.1.3 การทดสอบการใช้คำสั่ง SET และ RESET

ภาษาบูลลีน

```
LD    X2
SET   Y7
LD    X3
SET   Y7
LD    X4
SET   Y7
LD    X5
RST   Y7
END
```

ภาษาแลดเดอร์

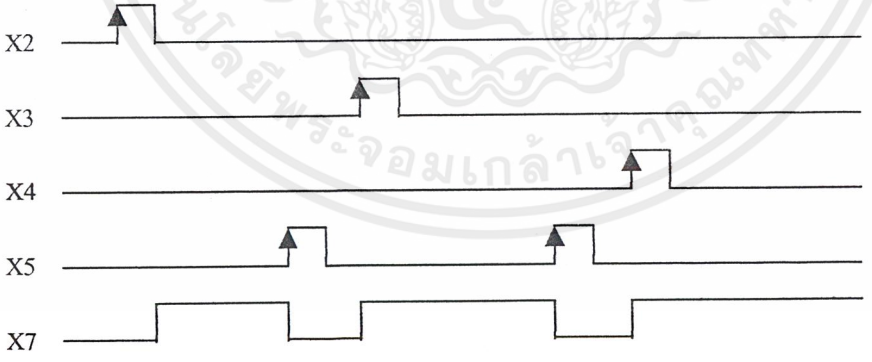


รูปที่ 6.3 วงจรทดสอบการทำงาน SET และ RESET

การทำงาน

- คำสั่ง SET และ RST สามารถใช้กับคอยล์ของรีเลย์เบอร์เดียวซ้ำกันหลายๆครั้งได้
- เมื่อ X2 หรือ X3 หรือ X4 ON จะทำให้ เอาท์พุท Y7 ถูกเซทและทำงาน
- เมื่อ X5 ON จะทำให้เอาท์พุท Y7 ถูกรีเซทและหยุดทำงาน

Timing Diagram



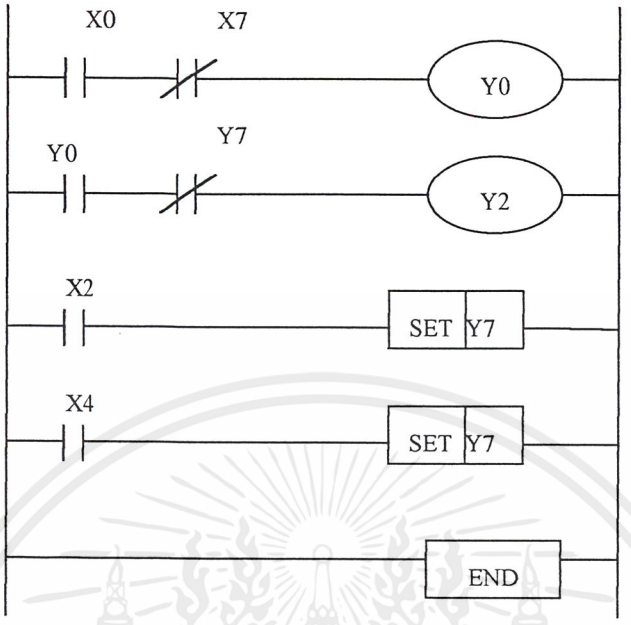
รูปที่ 6.4 Timing Diagram ของวงจรทดสอบ SET และ RST

6.1.4 การทดสอบใช้คำสั่ง LD กับ ANI

ภาษาบูลีน

```
LD    X0
ANI   Y7
OUT   Y0
LD    Y0
ANI   Y7
OUT   Y2
LD    X2
SET   Y7
LD    X4
RST   Y7
END
```

ภาษาแลดเดอร์

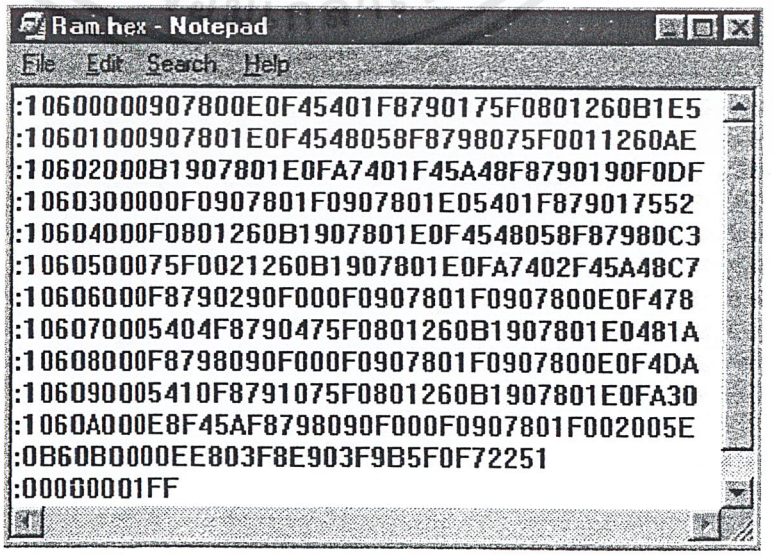


รูปที่ 6.5 วงจรทดสอบการใช้คำสั่ง LD กับ ANI

การทำงาน

- เมื่อ X0 ON ทำให้ Y0 ทำงานเป็นผลให้ Y2 ทำงาน
- เมื่อ X2 ON ทำให้ Y7 ถูกเซ็ท Y7 ทำงาน เป็นผลทำให้ Y0 และ Y2 หยุดทำงาน
- เมื่อ X4 ON ทำให้ Y7 ถูกรีเซ็ท Y7 หยุดทำงาน ดังนั้น Y0 กับ Y2 จะทำงานหรือไม่ขึ้นกับ X0

ไฟล์ออปเจ็คโค้ด

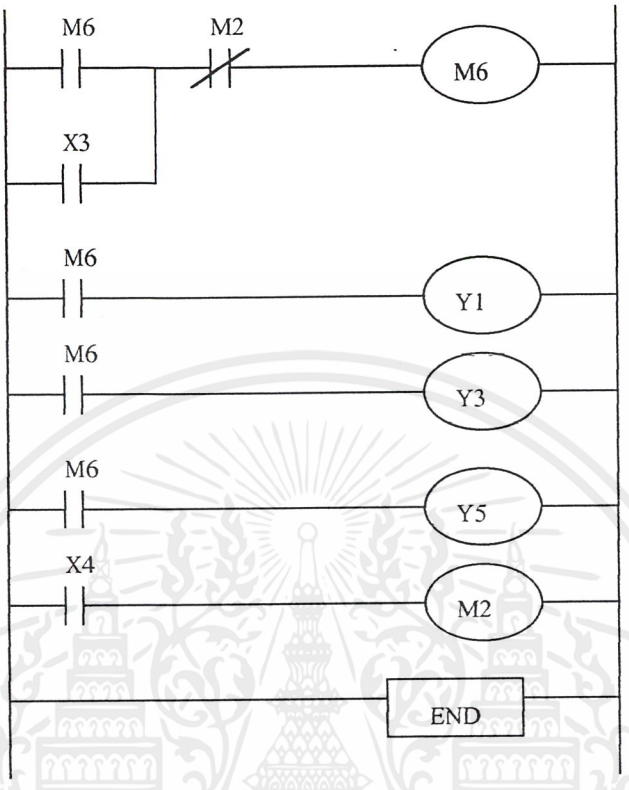


6.1.5 วงจรรักษาสภาพ

ภาษาบูลลีน

```
LD    M6
OR    X3
ANI   M2
OUT   M6
LD    M6
OUT   Y1
LD    M6
OUT   Y3
LD    M6
OUT   Y5
LD    X4
OUT   M2
END
```

ภาษาแลดเดอร์

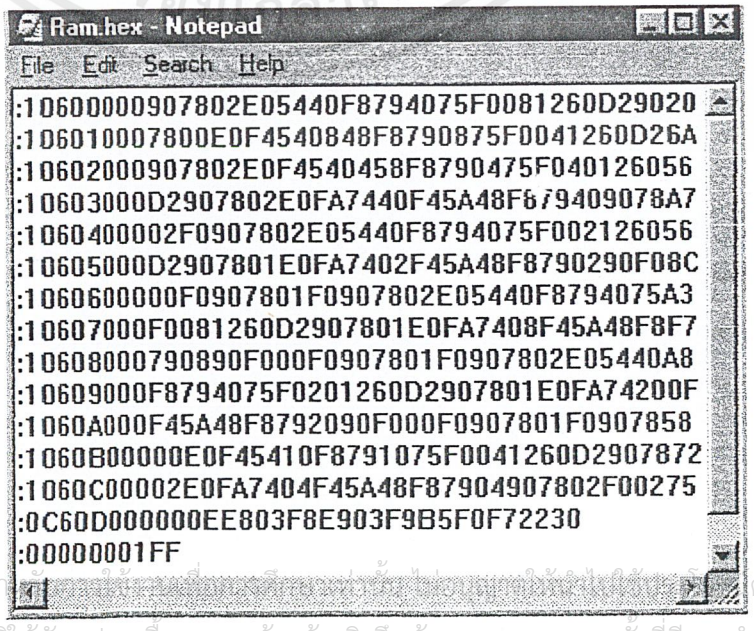


รูปที่ 6.6 วงจรรักษาสภาพ

การทำงาน

- เมื่อ X3 ON ทำให้ M6 ทำงาน เป็นผลให้ Y1, Y3, Y5 ทำงาน ถึงแม้ X3 จะ OFF ก็ตาม
- เมื่อ X4 ON ทำให้ M2 ทำงาน เป็นผลให้ M6 หยุดทำงาน Y1, Y2, Y5 หยุดทำงานด้วย

ออปเจ้คโค้ด



6.2 การทดลองการทำงานส่วนฮาร์ดแวร์ของพีแอลซี

แบ่งการทดลองออกเป็น 2 ส่วน ดังนี้

6.2.1 การทดลองการเชื่อมต่อข้อมูลระหว่างไมโครคอนโทรเลอร์กับเครื่องคอมพิวเตอร์

ในการใช้งานจำเป็นต้องนำคำสั่งของเครื่องพีแอลซีที่ผู้ใช้เขียนขึ้นบนคอมพิวเตอร์ ซึ่งแปลงเป็นไฟล์ออปเจ้คโค้ดแล้ว นำมาโหลดโปรแกรมลงเครื่องพีแอลซีไปเก็บไว้ที่แรม โดยไอซีที่เป็นตัวกลางระหว่างไมโครคอนโทรเลอร์และเครื่องคอมพิวเตอร์ คือ เบอร์ DS275 สัญญาณที่ออกจากไอซีจะอยู่ที่ระดับ 0-5 โวลท์

ในการทำการทดลอง โดยใช้ไอซี AT89S8252 ที่โปรแกรมแล้ว ตั้ง SW1 ให้อยู่ในโหมด “STOP” หรือโหมด “PROGRAM” เพื่อบันทึกข้อมูลที่ส่งจากคอมพิวเตอร์ลงบนแรม โดยส่งผ่านการสื่อสารข้อมูล RS-232 โดยโปรแกรมจะทำการเปลี่ยนรหัสแอสกี้เป็น HEX Code (ฐาน 16) และทำการส่งข้อมูลจากเครื่องคอมพิวเตอร์ลงบนหน่วยความจำแรมของพีแอลซี

ทดสอบโดยการเขียนโปรแกรมเป็นภาษาแอสเซมบลี้ของ MCS-51 ในอีดิเตอร์ของดอสโหมด โดยเลือกโปรแกรมง่ายๆ เพื่อเหมาะในการทดสอบ และนำตัวคอมไพล์แอสเซมบลี้ ชื่อ CROSS 32 นำมาทำการคอมไพล์เลอร์ (Compiler) เป็นไฟล์ออปเจ้คโค้ด โดยที่หัวของโปรแกรมแอสเซมบลี้ ต้องกำหนดแอดเดรสเริ่มต้นที่ 6000H โดยเขียนเป็น “ORG 6000H” หลังจากที่คอมไพล์ได้เป็นไฟล์ออปเจ้คโค้ด จะใช้การส่งโปรแกรมบนดอส โดยการกำหนดค่าการสื่อสารดังนี้

C:\MODE COM1:96,N,8,1,P

โดย COM1 : เป็นการกำหนดพอร์ตที่ใช้ในการสื่อสาร

96 : เป็นการตั้งค่า Baud Rate เท่ากับ 9600 บิทต่อวินาที

N : พาริตี้บิท เท่ากับ None

8 : เป็นการกำหนดดาต้าบิทเท่ากับ 8 บิท

1 : เป็นการกำหนดบิทหยุด (Stop bit) เท่ากับ 1 บิท

จากนั้นทำการส่งไฟล์ที่ต้องการส่ง โดยมีรูปแบบดังนี้

C:\COPY (ชื่อไฟล์).HEX COM1

หลังจากนั้นทำการส่งไฟล์จากคอมพิวเตอร์ลงเครื่องพีแอลซี หากมีการตอบรับจากเครื่องพีแอลซีด้วยสัญญาณเสียง หมายความว่าการส่งข้อมูลครบสมบูรณ์แล้ว สามารถที่จะทดสอบใช้งานโปรแกรม โดยการโยกตำแหน่งสวิทช์ SW1 ไว้ในโหมดการ "RUN" จากนั้นตรวจสอบการทำงานของพีแอลซีว่าถูกต้องหรือไม่

6.2.2 การทดสอบการติดต่อกับพอร์ทอินพุท/เอาท์พุท

สามารถทดสอบโดยการเขียนโปรแกรมไฟวิ่ง การทำงานของโปรแกรมนี้คือ ไฟวิ่งลักษณะของการนับเลขไบนารี่ หากสวิทช์อินพุททั้งหมดปิดจะเป็นการนับขึ้น และหากสวิทช์อินพุทตัวใดเปิดการนับเลขก็จะตรงข้าม

```
;Program test port Input/output
PORTOUT     EQU   0F00H        ;OUT PORT
PORTIN      EQU   0F00H        ;IN PORT
            ORG   4000H

            MOV   R7,#OH
            MOV   DPTR,#PORTOUT
            MOV   A,#0H
START_A     MOV   C,P1.0
            JNC   LOOPA
            CLR   P1.3
            SETB  P1.2
            JMP   LOOPB
LOOPA       CLR   P1.2
            SET   P1.3
            LJMP  START_A      ;0040H
LOOPB       MOVX  @DPTR,A
            MOV   DPTR,#PORTIN
            MOVX  A,@DPTR
            JNZ   GO
            ;
            ;
```

```
                MOV   DPTR,#PORTOUT
                MOV   A,R7
                MOVX @DPTR,A       ;OUT PORTA
                INC    R7
                LCALL DELAY
                LJMP   START_A
GO              MOV   DPTR,#PORTOUT
                MOV   A,R7
                MOVX @DPTR,A       ;OUT PORTA
                DEC    R7
                LCALL DELAY
                LJMP   START_A

*****************DELAY*******************
;
DELAY           PUSH   ACC
                PUSH   PSW
                MOV   R1,#0FFH
DELAY2          MOV   R2,#0FFH
                DJNZ   R2,$
                DJNZ   R1,DELAY2
                POP    PSW
                POP    ACC
                RET
                ;
                END
```

6.3 สรุปผลการทดลอง

จากการทดลองทั้งหมดที่ได้ทำการออกแบบและทดสอบ ซึ่งแบ่งออกได้เป็นสามส่วน คือ ในส่วนฮาร์ดแวร์ ในส่วนซอฟแวร์ที่เป็นยูสเซอร์อินเตอร์เฟสและในส่วนโปรแกรมมอนิเตอร์

ในส่วนของฮาร์ดแวร์นั้นไม่ค่อยพบปัญหาในการทำงาน สำหรับซอร์ฟแวร์ที่ออกแบบมาให้ใช้กับโปรแกรมภาษาบูลลีน สามารถใช้ได้กับคำสั่งพื้นฐานเช่น คำสั่ง LD, LDI, AND, ANI, OR, ORI, SET, RST, และ OUT กับอุปกรณ์ อินพุท (X) เอาท์พุท (Y) และรีเลย์ช่วย (M) เท่านั้น ยังไม่สามารถใช้กับคำสั่งซับซ้อนอย่าง เคาท์เตอร์ (C) หรือ ไทม์เมอร์ได้ (T) เนื่องจากผู้ทำได้ติดปัญหาหลายอย่าง อาทิ ตัวคอมไพล์เลอร์ที่นำมาใช้ สำหรับการทดลองใช้งาน สามารถใช้ได้ดีโดยไม่มีปัญหาเกิดขึ้น

ในส่วนของโปรแกรมยูสเซอร์อินเตอร์เฟสสำหรับใช้บนวินโดส์จะใช้วิชวลเบสิกในการเขียนซึ่งมีความซับซ้อนพอสมควรโดยจะใช้ในการเขียนเป็นภาษาบูลลีนในรูปของเท็กไฟแล้วมาแปลงเป็น (.asm) แอสเซมบลี้ ซึ่งจะมีความซับซ้อนในการตัดคำของภาษาแอสเซมบลี้เพื่อนำมาคอมไพเลอร์เป็นเลขฐานสิบหกเพื่อนำส่งออกไปให้บอร์ด PLCทำงานต่อไป

ในส่วนของโปรแกรมมอนิเตอร์เป็นโปรแกรมที่ใช้สำหรับควบคุมบอร์ดไมโครคอนโทรลเลอร์ โดยจะเป็นตัวสั่งให้ไมโครคอนโทรลเลอร์กระโดดไปตามตำแหน่งที่เราต้องการ และเป็นโปรแกรมที่ใช้ในการนำค่าโปรแกรมที่ได้เขียนในวิชวลนำมาโหลดลงสู่แรมซึ่งจะมีความยุ่งยากในการรับส่งค่าเพราะเราต้องนำรหัส ASCII มาแปลงเป็นเลขฐานสิบหกซึ่งจะมีความยุ่งยากในระยะเริ่มแรกในการออกแบบ

ปัญหาที่พบเจอในด้านการเขียนโปรแกรมซึ่งจะแบ่งออกเป็น 2 ส่วนคือส่วนของที่ใช้โปรแกรม Visual Basic(VB) และส่วนที่ใช้โปรแกรมแอสแซมบลี้ (Assembly) ซึ่งแต่ละส่วนมีความซับซ้อนในตัวเองเมื่อนำมา อินเตอร์เฟส (Interface)กัน ทำให้โปรแกรมมีความผิดพลาดเกิดขึ้น

บทที่ 7
บทสรุปและวิจารณ์

สำหรับการทำโครงงานพีแอลซีนี้แบ่งขั้นตอนการสร้างออกเป็น 2 ส่วนใหญ่ๆ คือ 1.ส่วนของฮาร์ดแวร์ 2. ส่วนของซอฟแวร์ สำหรับในส่วนของฮาร์ดแวร์นั้นไม่ค่อยยุ่งยากและปัญหาที่พบน้อยมาก และสามารถแก้ไขได้ด้วยซอฟแวร์ เป็นต้น และในการออกแบบสร้างส่วนของฮาร์ดแวร์ ซึ่งประกอบไปด้วย 3 ส่วนหลัก คือ วงจรส่วนประมวลผล วงจรส่วนอินพุทและเอาท์พุท ทางผู้จัดทำได้ออกแบบแผ่นปริ้นลายวงจรโดยให้วงจรทั้ง 3 ส่วนอยู่บนบอร์ดเดียวกัน ทำให้ขนาดของพีแอลซีมีขนาดเล็กกระทัดรัด เหมาะแก่การนำไปใช้งานได้จริง

สำหรับส่วนที่สำคัญของโครงงานนี้อยู่ที่ซอฟแวร์ ที่ให้ผู้ใช้เขียนโปรแกรมควบคุมการทำงานของพีแอลซี ซึ่งในการจัดสร้างโปรแกรมซอฟแวร์นั้นแบ่งออกเป็น 2 ส่วน คือ 1. ส่วนที่เป็นซอฟแวร์บนเครื่องคอมพิวเตอร์โดยใช้โปรแกรม Visual Basic ออกแบบ 2. ส่วนที่เป็นซอฟแวร์หลักที่เก็บบนหน่วยความจำบนไมโครคอนโทรเลอร์ สำหรับการที่จะจัดสร้างซอฟแวร์ควบคุมพีแอลซี ในโครงงานนี้ ผู้จัดทำจำเป็นจะต้องศึกษาโปรแกรมต่างๆ มากมาย อาทิเช่น

- โปรแกรมภาษาที่ควบคุมพีแอลซี แบ่งออกเป็น ภาษาบูลีน และภาษาแลดเดอร์
- โปรแกรมภาษาแอสเซมบลี้ ที่ใช้กับไมโครคอนโทรเลอร์ MCS-51
- โปรแกรม Visual Basic

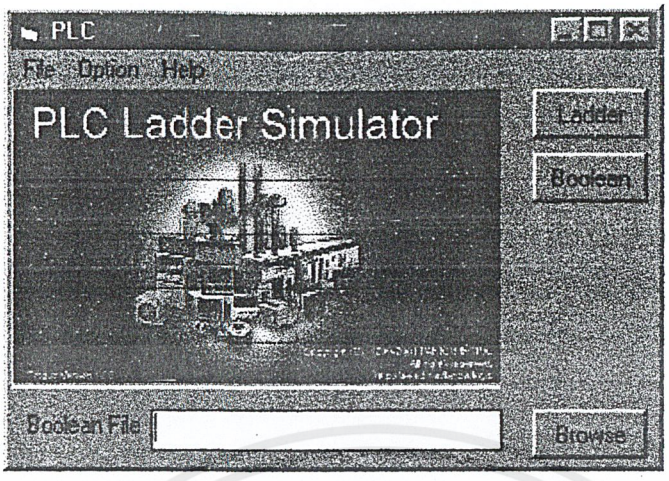
รวมไปถึงต้องศึกษาโครงสร้างของฮาร์ดแวร์ ในการจัดแบ่งพื้นที่การใช้งานในหน่วยความจำให้มีความสัมพันธ์กับซอฟแวร์ที่ออกแบบขึ้นมา ซึ่งผู้จัดทำต้องใช้เวลาในการศึกษาโปรแกรมต่างๆ กว่าจะจัดทำขึ้นเป็นซอฟแวร์พีแอลซีได้ และยังพบปัญหาที่เกิดขึ้น ในเรื่องของการแปลง (Compile) จากภาษาบูลีนมาเป็นภาษาแอสเซมบลี้ ซึ่งมีความซับซ้อนมาก

ในเบื้องต้นผู้จัดทำได้ออกแบบสร้างส่วนซอฟแวร์บนเครื่องคอมพิวเตอร์โดยใช้โปรแกรมภาษาเดลไฟล์ซึ่งต้องพบปัญหาต่างๆ มากมายทั้งในเรื่องของซอร์ทโค้ด และหนังสือที่มีอธิบายการใช้โปรแกรมเดลไฟล์น้อยรวมไปถึงต้องศึกษาคำสั่งของภาษาปาสคาลซึ่งมีความยุ่งยากทำให้ต้องเสียเวลาไปช่วงหนึ่ง จึงทำให้ผู้จัดทำหันไปศึกษาโปรแกรม Visual Basic แทนและนำมาทำเป็นซอฟแวร์บนเครื่องคอมพิวเตอร์ซึ่งมีความง่ายดายมากกว่าเพราะปัจจุบันนี้โปรแกรม Visual Basic เป็นที่นิยมใช้กันมาก และเหมาะแก่ผู้ที่จะมาศึกษาเพื่อทำความเข้าใจหรือพัฒนาในโครงงานนี้ต่อไปในอนาคต ให้ดียิ่งขึ้น

ภาคผนวก

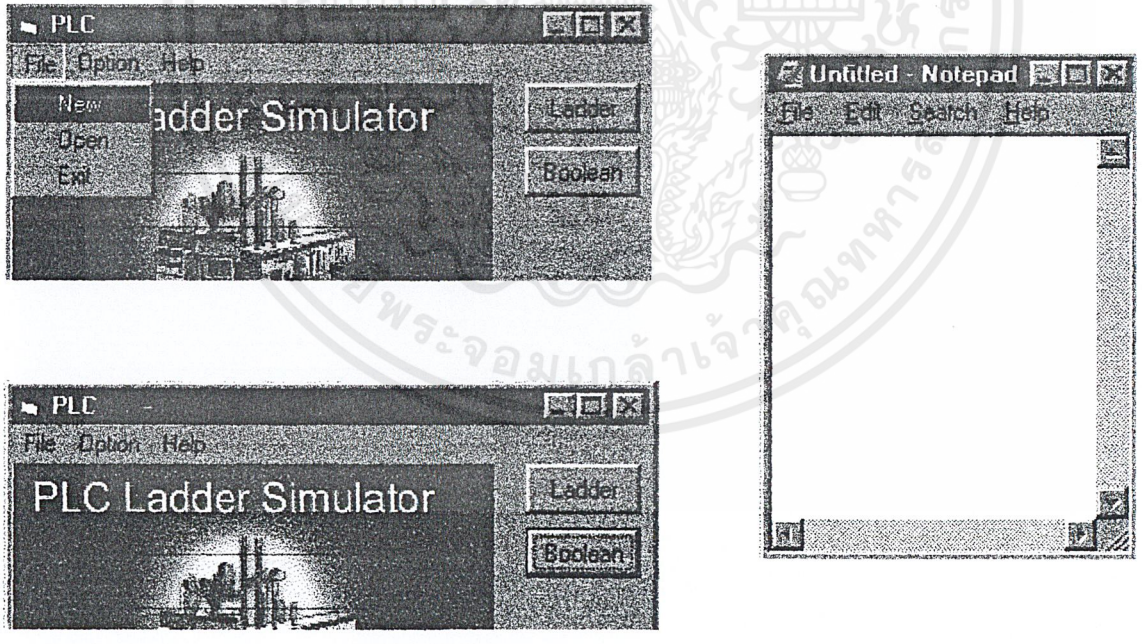
ก
ซอร์ฟแวร์และโปรแกรมหลัก

ซอร์ฟแวร์ (SOFT WARE)

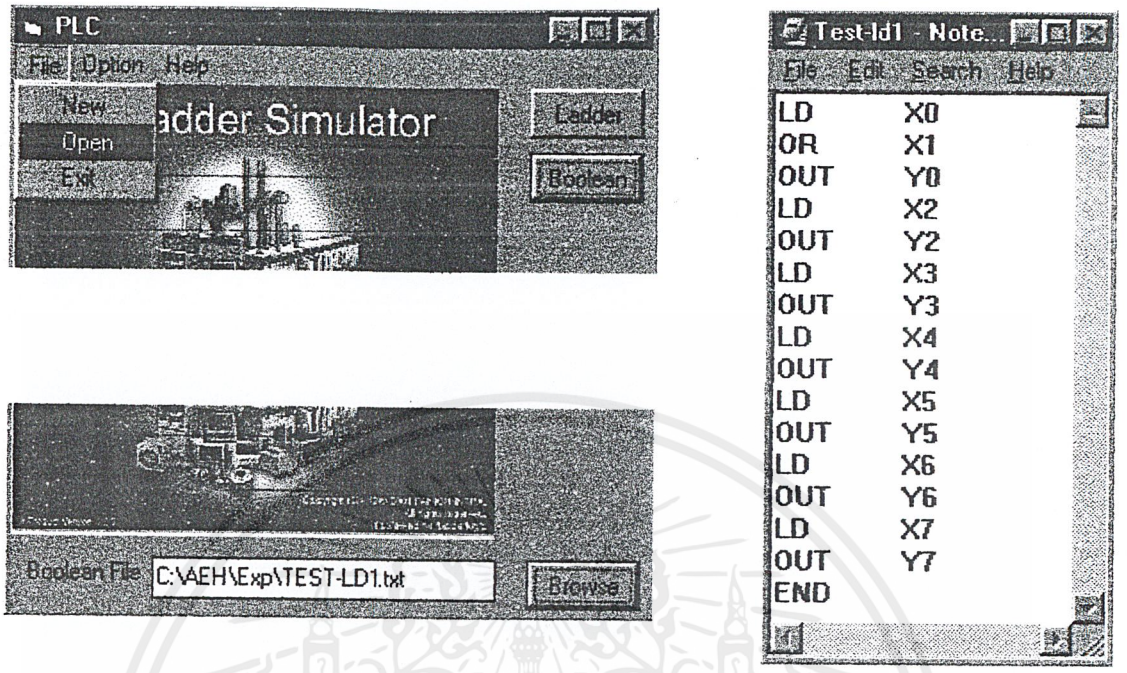


รูปที่ 1 หน้าต่างของโปรแกรมพีแอลซี

การใช้งาน

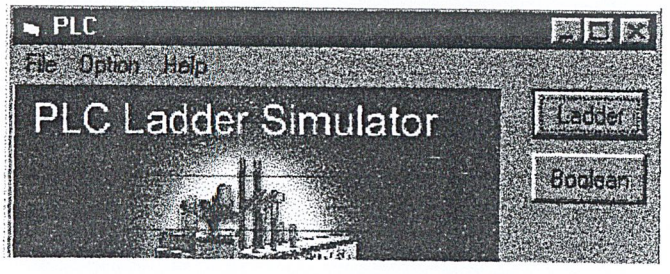


รูปที่ 2 การเรียกใช้โปรแกรม Note Pad เพื่อเขียนคำสั่งภาษาบูลลีน

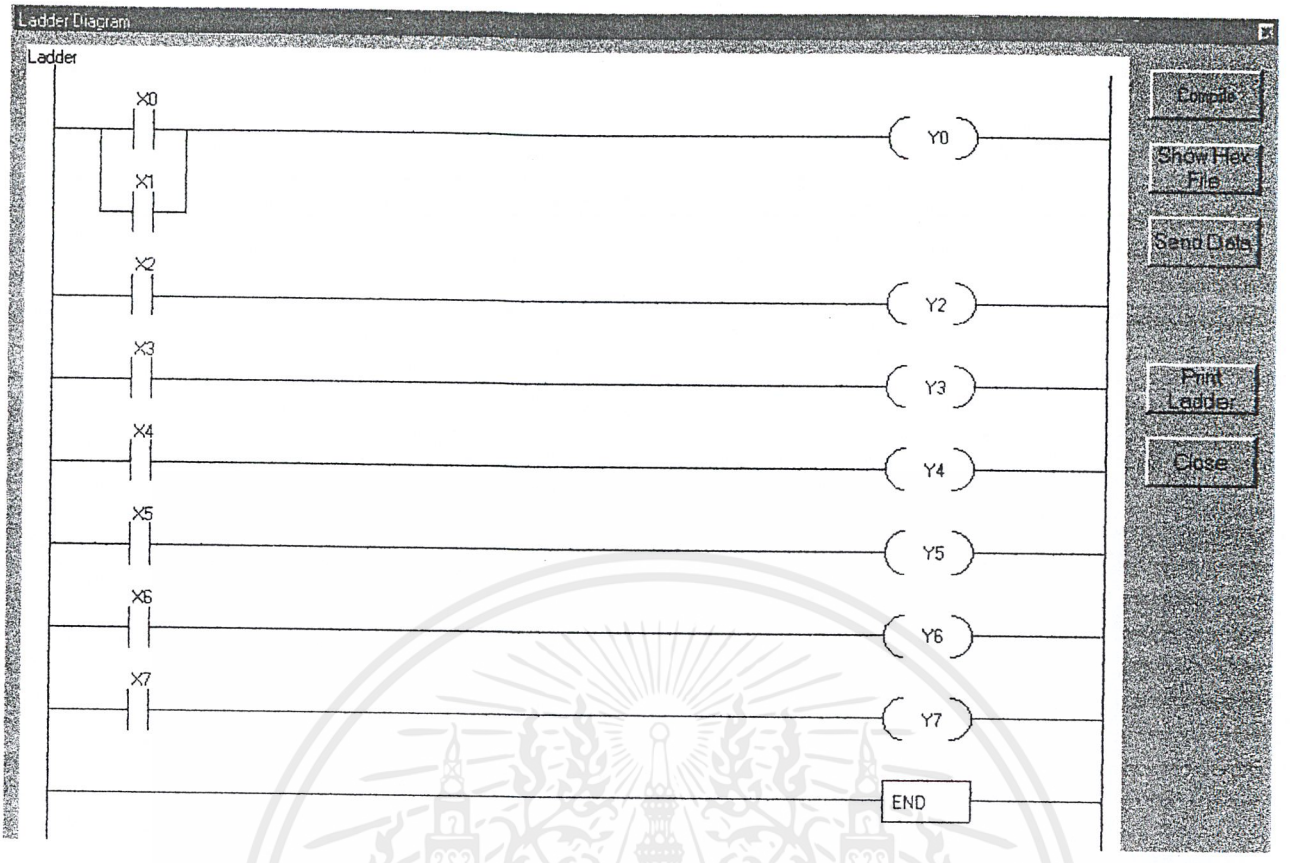


รูปที่ 3 เปิดโปรแกรมภาษาบูลลีนที่ถูกเขียนขึ้นบน Note Pad

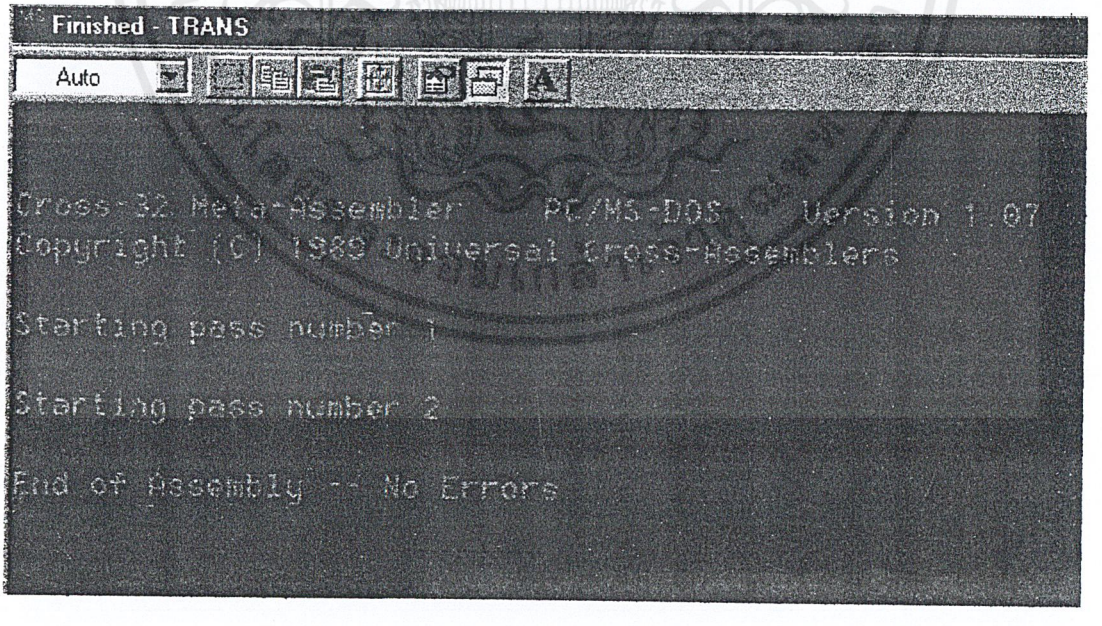
- สามารถเริ่มจากการเลือกที่จะสร้างโปรแกรมใหม่ หรือ เปิดโปรแกรมที่สร้างไว้แล้ว โดยเป็นไฟล์สกุล .txt ตามรูปที่ 2 และรูปที่ 3
- ในการสร้างโปรแกรมใหม่ โดยเขียนเป็นโปรแกรมภาษาบูลลีนลงบนโปรแกรม Note Pad โดยใช้อักษรตัวพิมพ์ใหญ่ทั้งหมด และใช้ TAB ในการเว้นวรรค
- เมื่อสร้างโปรแกรมเสร็จแล้ว สามารถทำการเซฟเป็นไฟล์นามสกุล .txt
- จากนั้นเมื่อต้องการที่จะเรียกดูแลดเดอร์ไดอะแกรม โดยการเปิดไฟล์โปรแกรมที่สร้างไว้แล้วตามรูปที่ 3 และทำการเรียกดูแลดเดอร์ไดอะแกรม ตามรูปที่ 4



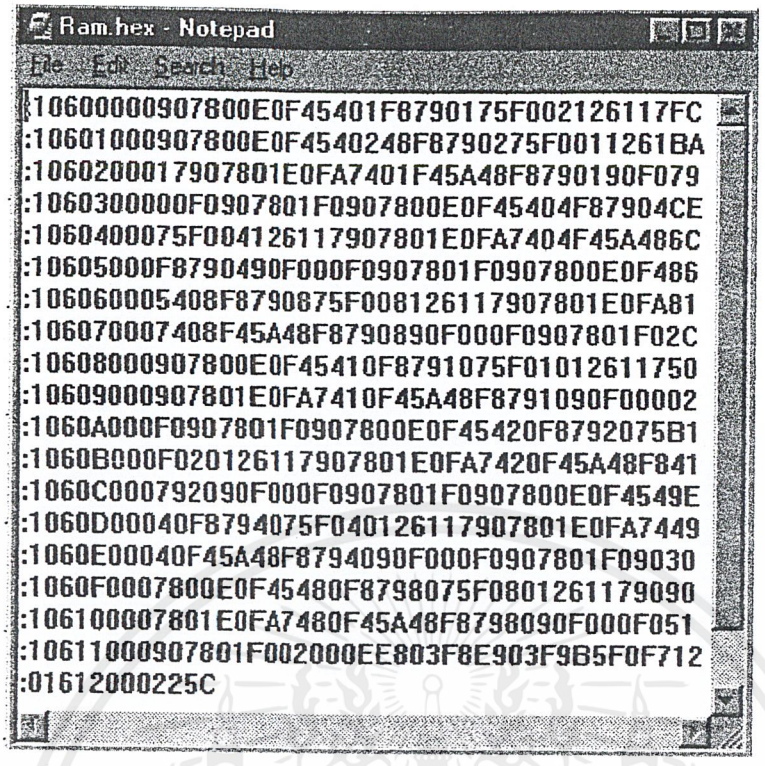
รูปที่ 4 แสดงการเรียกเลดเดอร์ไดอะแกรม



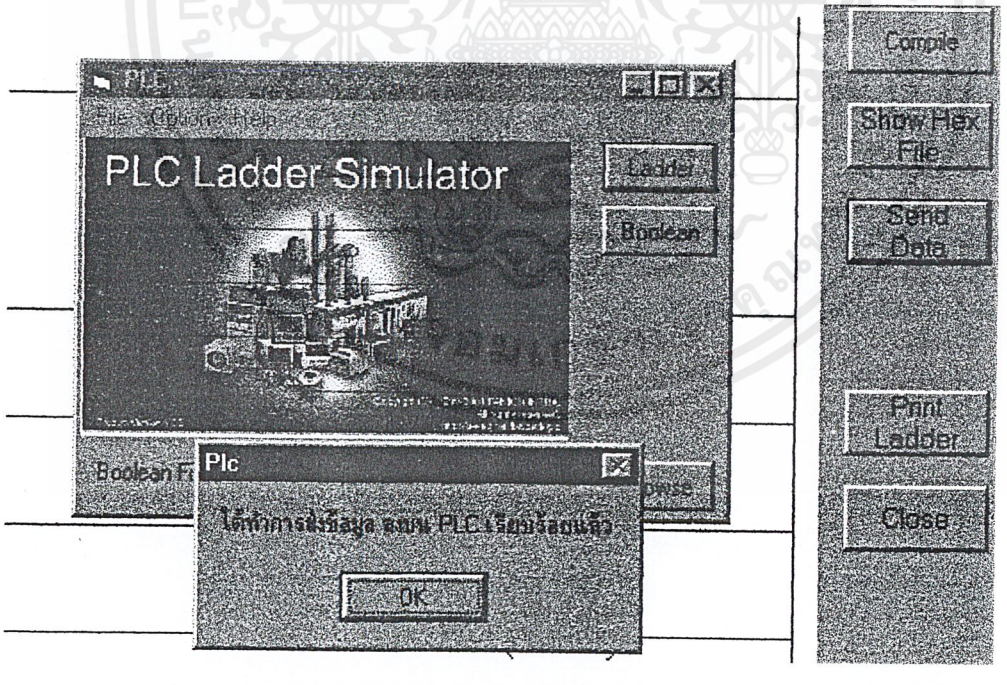
รูปที่ 5 แลดเดอร์ไดอะแกรม



รูปที่ 6 แสดงการ Compile ไม่มีการ errors



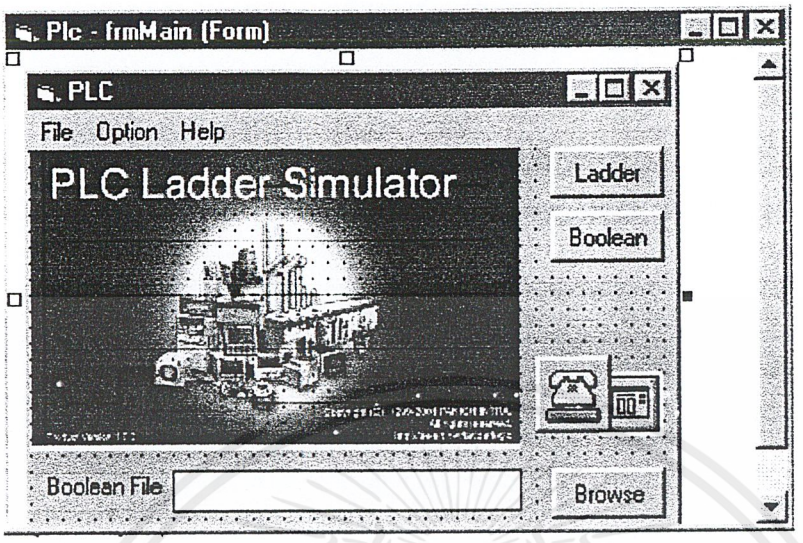
รูปที่ 7 แสดง Hex File



รูปที่ 8 แสดงการส่งข้อมูลจากคอมพิวเตอร์ลงบน พีแอลซี

- เมื่อทำการเขียนโปรแกรมเสร็จแล้ว ทำการ Compile โดยการเปลี่ยนเป็นไฟล์ออปเจ้คโค้ด โดยเราสามารถเรียกดู Hex File ได้ก่อนที่จะทำการส่ง
- เมื่อโปรแกรมที่เขียนถูกต้อง การ Compile จะไม่มี errors เกิดขึ้น
- สามารถที่จะส่งข้อมูลลงบน PLC ได้ โดยทำการ Sent Data เมื่อทำการส่งข้อมูลเสร็จสมบูรณ์แล้วจะขึ้นหน้าต่างดังรูปที่ 8 เมื่อข้อมูลส่งลงเครื่อง พีแอลซี เรียบร้อยแล้ว ฮาร์ดแวร์จะทำการตอบรับข้อมูลที่สมบูรณ์ โดยการส่งเสียงตอบรับ 2 ครั้ง เป็นอันสมบูรณ์

โปรแกรมหลัก



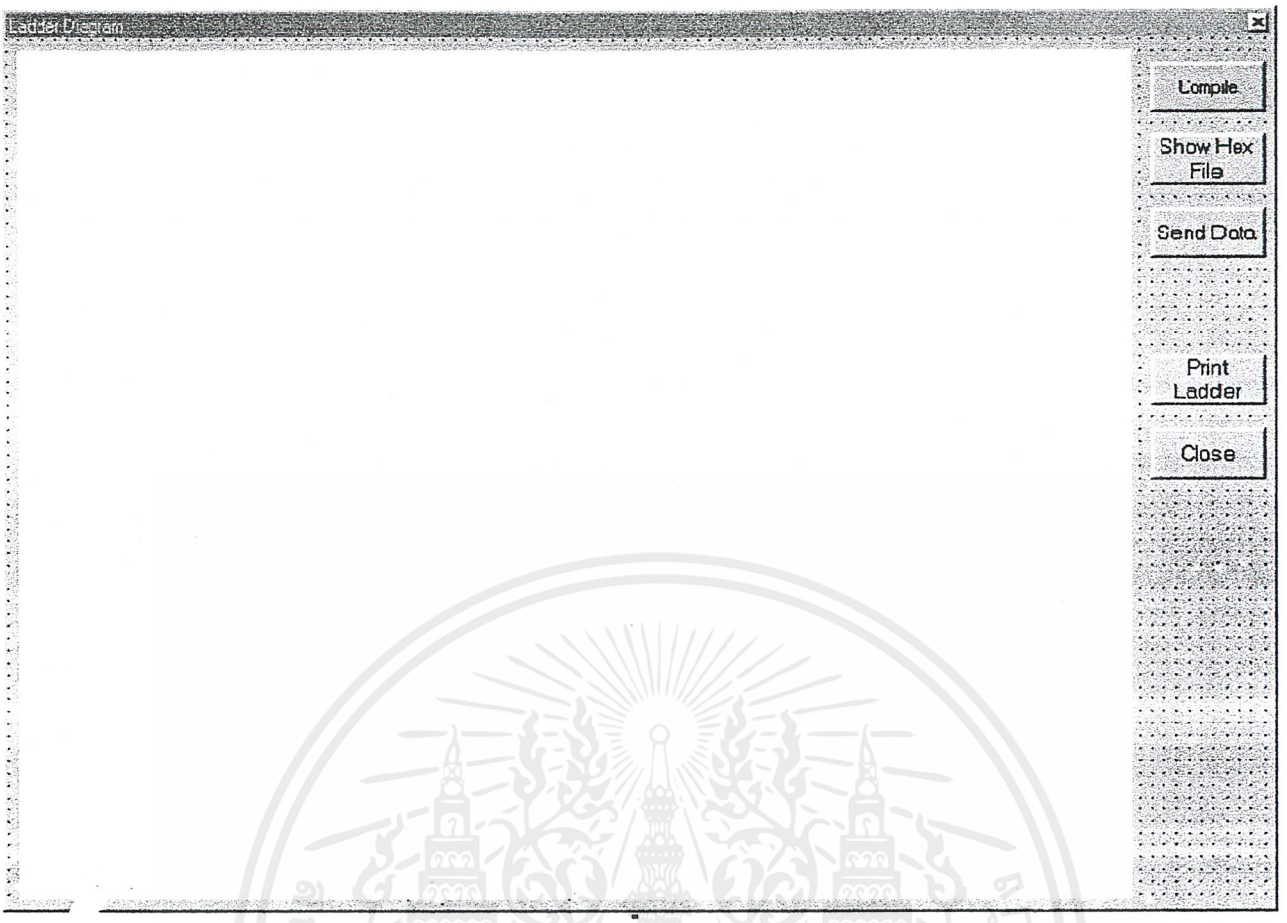

```
Sub browse()
   CommonDialog1.ShowOpen
   Text1 = CommonDialog1.FileName
End Sub
Sub opennet()
   fn = Text1
   fnn = "c:\windows\notepad " + fn
   Shell fnn, vbNormalFocus
End Sub

Private Sub Command1_Click()
   browse
End Sub

Private Sub Command4_Click()
   opennet
End Sub

Private Sub Command5_Click()
   If Text1 <> "" Then
      readvalue
      graphfrm.Show
      opengraph
   End If
End Sub
```

```
Private Sub mmexit_Click()
   Unload Me
End Sub
Private Sub mnuFileneslist_Click()
   browse
End Sub
Private Sub mnuFilenewfile_Click()
   opennet
End Sub
```




```
Private Sub Command1_Click()
   convest
End Sub

Private Sub Command2_Click()
    send
End Sub

Private Sub Command3_Click()
   Unload Me
End Sub

Private Sub Command4_Click()
   Shell "notepad ram.hex", vbNormalFocus
End Sub

Private Sub Command5_Click()
   PrintForm
End Sub

Dim datas(800) As String
```

```
Dim chk(800)
Dim Comd(800) As String
Dim Value(800) As String
Dim fn

Public fMainForm As frmMain
Sub Main()
  Set fMainForm = New frmMain
  fMainForm.Show
End Sub

Sub readvalue() 'open file and read value
  Sum = 0
  fn = FreeFile
  Open fMainForm.Text1 For Input As #fn
  While Not EOF(fn)
    Sum = Sum + 1
    Input #fn, chk(Sum)
  Wend
  Close #fn
  For n = 1 To 800
    Comd(n) = ""
    Value(n) = ""
  Next
  For n = 1 To Sum - 1
    For nn = 1 To 800
      datas(nn) = 0
    Next
    nnn = 0
    For nn = Len(chk(n)) To 1 Step -1
      nnn = nnn + 1
      datas(nnn) = Asc(Right$(chk(n), nn))
      If datas(nnn) = 9 Then
        Comd(n) = datanet
        datanet = ""
      End If
      If datas(nnn) <> 9 Then
        datanet = datanet + Chr$(datas(nnn))
      End If
      If Len(chk(n)) = nnn Then
        Value(n) = datanet
        datanet = ""
      End If
    Next
  Next
End Sub
```

```
Sub opengraph()
  n = 0
  X0 = 13.5
  X1 = graphfrm.Picture1.Width - 10
  Y0 = 10
  Y1 = graphfrm.Picture1.Height - 10
  Ys = Y1 / 10
  Xs = X1 / 10
  Xo = X1 / 50
  Yo = Y1 / 40
  graphfrm.Picture1.Cls
  graphfrm.Picture1.Print ("Ladder")
  graphfrm.Picture1.Line (X0, Y0)-(X0, Y1)
  graphfrm.Picture1.Line (X1, Y0)-(X1, Y1)
  For i = 1 To 800
    If Comd(i) = "LD" Then
      checkor = 0
      nnn = 2
      nn = 1
      n = n + 1
      graphfrm.Picture1.Line (X0, Ys * n)-(Xs, Ys * n)
      graphfrm.Picture1.Line (Xs, Ys * n - Yo)-(Xs, Ys * n + Yo), 0
      graphfrm.Picture1.Line (Xs + Xo, Ys * n - Yo)-(Xs + Xo, Ys * n + Yo), 0
      graphfrm.Picture1.Circle (Xs + Xo / 5, Ys * n - 2 * Yo), 0.01
      graphfrm.Picture1.Print Value(i)
      nn = nn + 1
    End If
    If Comd(i) = "LDI" Then
      checkor = 0
      nnn = 2
      nn = 1
      n = n + 1
      graphfrm.Picture1.Line (X0, Ys * n)-(Xs, Ys * n)
      graphfrm.Picture1.Line (Xs, Ys * n - Yo)-(Xs, Ys * n + Yo), 0
      graphfrm.Picture1.Line (Xs + Xo, Ys * n - Yo)-(Xs + Xo, Ys * n + Yo), 0
      graphfrm.Picture1.Line (Xs, Ys * n + Yo)-(Xs + Xo, Ys * n - Yo), 0
      graphfrm.Picture1.Circle (Xs + Xo / 5, Ys * n - 2 * Yo), 0.01
      graphfrm.Picture1.Print Value(i)
      nn = nn + 1
    End If
    If Comd(i) = "OR" Then
      n = n + 1
      ny = n
      If nnn > 4 Then nnn = nnn + nnn - 4
      graphfrm.Picture1.Line (Xs * (nnn - 1) / 2 + Xo, Ys * ny)-(Xs * (nnn - 1) / 2 + Xo, Ys * (ny - 1))
      graphfrm.Picture1.Line (Xs * nnn / 2 + Xo, Ys * ny)-(Xs * (nnn + 1) / 2, Ys * ny)
      graphfrm.Picture1.Line (Xs * (nnn + 1) / 2, Ys * ny)-(Xs * (nnn + 1) / 2, Ys * (ny - 1))
```

```
    graphfrm.Picture1.Line (Xs * (nnn - 1) / 2 + Xo, Ys * ny)-(Xs * nnn / 2, Ys * ny)
    graphfrm.Picture1.Line (Xs * nnn / 2, Ys * ny - Yo)-(Xs * nnn / 2, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * nnn / 2 + Xo, Ys * ny - Yo)-(Xs * nnn / 2 + Xo, Ys * ny + Yo), 0
    graphfrm.Picture1.Circle (Xs * nnn / 2 + Xo / 5, Ys * ny - 2 * Yo), 0.01
    graphfrm.Picture1.Print Value(i)
    checkor = checkor + 1
  End If
  If Comd(i) = "ORI" Then
    n = n + 1
    ny = n
    If nnn > 4 Then nnn = nnn + nnn - 4
    graphfrm.Picture1.Line (Xs * (nnn - 1) / 2 + Xo, Ys * ny)-(Xs * (nnn - 1) / 2 + Xo, Ys * (ny - 1))
    graphfrm.Picture1.Line (Xs * nnn / 2 + Xo, Ys * ny)-(Xs * (nnn + 1) / 2, Ys * ny)
    graphfrm.Picture1.Line (Xs * (nnn + 1) / 2, Ys * ny)-(Xs * (nnn + 1) / 2, Ys * (ny - 1))
    graphfrm.Picture1.Line (Xs * (nnn - 1) / 2 + Xo, Ys * ny)-(Xs * nnn / 2, Ys * ny)
    graphfrm.Picture1.Line (Xs * nnn / 2, Ys * ny - Yo)-(Xs * nnn / 2, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * nnn / 2 + Xo, Ys * ny - Yo)-(Xs * nnn / 2 + Xo, Ys * ny + Yo), 0
    graphfrm.Picture1.Line (Xs * nnn / 2, Ys * ny + Yo)-(Xs * nnn / 2 + Xo, Ys * ny - Yo), 0
    graphfrm.Picture1.Circle (Xs * nnn / 2 + Xo / 5, Ys * ny - 2 * Yo), 0.01
    graphfrm.Picture1.Print Value(i)
    checkor = checkor + 1
  End If
  If Comd(i) = "AND" And checkor > 0 Then
    graphfrm.Picture1.Line (Xs * (nn - 1) + Xo, Ys * (n - checkor))-(Xs * nn, Ys * (n - checkor))
    graphfrm.Picture1.Line (Xs * nn, Ys * (n - checkor) - Yo)-(Xs * nn, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * nn + Xo, Ys * (n - checkor) - Yo)-(Xs * nn + Xo, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Circle (Xs * nn + Xo / 5, Ys * (n - checkor) - 2 * Yo), 0.01
    graphfrm.Picture1.Print Value(i)
    nn = nn + 1
    nnn = nn + 1
  End If
  If Comd(i) = "AND" And checkor = 0 Then
    nnn = 1
    graphfrm.Picture1.Line (Xs * (nn - 1) + Xo, Ys * n)-(Xs * nn, Ys * n)
    graphfrm.Picture1.Line (Xs * nn, Ys * n - Yo)-(Xs * nn, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * nn + Xo, Ys * n - Yo)-(Xs * nn + Xo, Ys * n + Yo), 0
    graphfrm.Picture1.Circle (Xs * nn + Xo / 5, Ys * n - 2 * Yo), 0.01
    graphfrm.Picture1.Print Value(i)
    nn = nn + 1
    nnn = nn + 1
  End If
  If Comd(i) = "ANI" And checkor > 0 Then
    graphfrm.Picture1.Line (Xs * (nn - 1) + Xo, Ys * (n - checkor))-(Xs * nn, Ys * (n - checkor))
    graphfrm.Picture1.Line (Xs * nn, Ys * (n - checkor) - Yo)-(Xs * nn, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * nn + Xo, Ys * (n - checkor) - Yo)-(Xs * nn + Xo, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * nn, Ys * (n - checkor) + Yo)-(Xs * nn + Xo, Ys * (n - checkor) - Yo), 0
    graphfrm.Picture1.Circle (Xs * nn + Xo / 5, Ys * (n - checkor) - 2 * Yo), 0.01
```

```
    graphfrm.Picture1.Print Value(i)
    nn = nn + 1
    nnn = nn + 1
  End If
  If Comd(i) = "ANI" And checkor = 0 Then
    nnn = 1
    graphfrm.Picture1.Line (Xs * (nn - 1) + Xo, Ys * n)-(Xs * nn, Ys * n)
    graphfrm.Picture1.Line (Xs * nn, Ys * n - Yo)-(Xs * nn, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * nn + Xo, Ys * n - Yo)-(Xs * nn + Xo, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * nn, Ys * n + Yo)-(Xs * nn + Xo, Ys * n - Yo), 0
    graphfrm.Picture1.Circle (Xs * nn + Xo / 5, Ys * n - 2 * Yo), 0.01
    graphfrm.Picture1.Print Value(i)
    nn = nn + 1
    nnn = nn + 1
  End If
  If Comd(i) = "RST" And checkor = 0 Then
    graphfrm.Picture1.Line (Xs * (nn - 1) + Xo, Ys * n)-(Xs * 8, Ys * n)
    graphfrm.Picture1.Line (Xs * 8, Ys * n - Yo)-(Xs * 8, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 2, Ys * n - Yo)-(Xs * 8 + Xo * 2, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * n - Yo)-(Xs * 8 + Xo * 4, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * n - Yo)-(Xs * 8 + Xo * 4, Ys * n - Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * n + Yo)-(Xs * 8 + Xo * 4, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * n + Yo)-(Xs * 8 + Xo * 4, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * n)-(X1, Ys * n)
    graphfrm.Picture1.Circle (Xs * 8 + Xo / 2, Ys * n - Yo / 2), 0.01
    graphfrm.Picture1.Print ("RST")
    graphfrm.Picture1.Circle (Xs * 8 + Xo * 5 / 2, Ys * n - Yo / 2), 0.01
    graphfrm.Picture1.Print Value(i)
  End If
  If Comd(i) = "RST" And checkor > 0 Then
    graphfrm.Picture1.Line (Xs * (nn - 1) + Xo, Ys * (n - checkor))-(Xs * 8, Ys * (n - checkor))
    graphfrm.Picture1.Line (Xs * 8, Ys * (n - checkor) - Yo)-(Xs * 8, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 2, Ys * (n - checkor) - Yo)-(Xs * 8 + Xo * 2, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * (n - checkor) - Yo)-(Xs * 8 + Xo * 4, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * (n - checkor) - Yo)-(Xs * 8 + Xo * 4, Ys * (n - checkor) - Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * (n - checkor) + Yo)-(Xs * 8 + Xo * 4, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * (n - checkor) + Yo)-(Xs * 8 + Xo * 4, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * (n - checkor))-(X1, Ys * (n - checkor))
    graphfrm.Picture1.Circle (Xs * 8 + Xo / 2, Ys * (n - checkor) - Yo / 2), 0.01
    graphfrm.Picture1.Print ("RTS")
    graphfrm.Picture1.Circle (Xs * 8 + Xo * 5 / 2, Ys * (n - checkor) - Yo / 2), 0.01
    graphfrm.Picture1.Print Value(i)
  End If
  If Comd(i) = "SP" And checkor = 0 Then
    graphfrm.Picture1.Circle (Xs * 8 + Xo * 5 / 2, Ys * n + Yo / 2), 0.01
    graphfrm.Picture1.Print Value(i)
  End If
```

```
If Comd(i) = "SP" And checkor > 0 Then
    graphfrm.Picture1.Circle (Xs * 8 + Xo * 5 / 2, Ys * (n - checkor) + Yo / 2), 0.01
    graphfrm.Picture1.Print Value(i)
End If
If Comd(i) = "SET" And checkor = 0 Then
    graphfrm.Picture1.Line (Xs * (nn - 1) + Xo, Ys * n)-(Xs * 8, Ys * n)
    graphfrm.Picture1.Line (Xs * 8, Ys * n - Yo)-(Xs * 8, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 2, Ys * n - Yo)-(Xs * 8 + Xo * 2, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * n - Yo)-(Xs * 8 + Xo * 4, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * n - Yo)-(Xs * 8 + Xo * 4, Ys * n - Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * n + Yo)-(Xs * 8 + Xo * 4, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * n + Yo)-(Xs * 8 + Xo * 4, Ys * n + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * n)-(X1, Ys * n)
    graphfrm.Picture1.Circle (Xs * 8 + Xo / 2, Ys * n - Yo / 2), 0.01
    graphfrm.Picture1.Print ("SET")
    graphfrm.Picture1.Circle (Xs * 8 + Xo * 5 / 2, Ys * n - Yo / 2), 0.01
    graphfrm.Picture1.Print Value(i)
End If
If Comd(i) = "SET" And checkor > 0 Then
    graphfrm.Picture1.Line (Xs * (nn - 1) + Xo, Ys * (n - checkor))-(Xs * 8, Ys * (n - checkor))
    graphfrm.Picture1.Line (Xs * 8, Ys * (n - checkor) - Yo)-(Xs * 8, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 2, Ys * (n - checkor) - Yo)-(Xs * 8 + Xo * 2, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * (n - checkor) - Yo)-(Xs * 8 + Xo * 4, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * (n - checkor) - Yo)-(Xs * 8 + Xo * 4, Ys * (n - checkor) - Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * (n - checkor) + Yo)-(Xs * 8 + Xo * 4, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * 8, Ys * (n - checkor) + Yo)-(Xs * 8 + Xo * 4, Ys * (n - checkor) + Yo), 0
    graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * (n - checkor))-(X1, Ys * (n - checkor))
    graphfrm.Picture1.Circle (Xs * 8 + Xo / 2, Ys * (n - checkor) - Yo / 2), 0.01
    graphfrm.Picture1.Print ("SET")
    graphfrm.Picture1.Circle (Xs * 8 + Xo * 5 / 2, Ys * (n - checkor) - Yo / 2), 0.01
    graphfrm.Picture1.Print Value(i)
End If
If Comd(i) = "OUT" And checkor = 0 Then
    graphfrm.Picture1.Line (Xs * (nn - 1) + Xo, Ys * n)-(Xs * 8, Ys * n)
    graphfrm.Picture1.Circle (Xs * 8 + Yo, Ys * n), Yo, 0, 3.14 / 2, 3 * 3.14 / 2
    graphfrm.Picture1.Circle (Xs * 8 + Xo * 4 - Yo, Ys * n), Yo, 0, 3 * 3.14 / 2, 3.14 / 2
    graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * n)-(X1, Ys * n)
    graphfrm.Picture1.Circle (Xs * 8 + Xo * 1.8, Ys * n - Yo / 2), 0.01
    graphfrm.Picture1.Print Value(i)
End If
If Comd(i) = "OUT" And checkor > 0 Then
    graphfrm.Picture1.Line (Xs * (nn - 1) + Xo, Ys * (n - checkor))-(Xs * 8, Ys * (n - checkor))
    graphfrm.Picture1.Circle (Xs * 8 + Yo, Ys * (n - checkor)), Yo, 0, 3.14 / 2, 3 * 3.14 / 2
    graphfrm.Picture1.Circle (Xs * 8 + Xo * 4 - Yo, Ys * (n - checkor)), Yo, 0, 3 * 3.14 / 2, 3.14 / 2
    graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * (n - checkor))-(X1, Ys * (n - checkor))
    graphfrm.Picture1.Circle (Xs * 8 + Xo * 1.8, Ys * (n - checkor) - Yo / 2), 0.01
    graphfrm.Picture1.Print Value(i)
```

```
    End If
  Next
      graphfrm.Picture1.Line (X0, Ys * (n + 1))-(Xs * 8, Ys * (n + 1))
      graphfrm.Picture1.Line (Xs * 8, Ys * (n + 1) - Yo)-(Xs * 8, Ys * (n + 1) + Yo), 0
      graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * (n + 1) - Yo)-(Xs * 8 + Xo * 4, Ys * (n + 1) + Yo), 0
      graphfrm.Picture1.Line (Xs * 8, Ys * (n + 1) - Yo)-(Xs * 8 + Xo * 4, Ys * (n + 1) - Yo), 0
      graphfrm.Picture1.Line (Xs * 8, Ys * (n + 1) + Yo)-(Xs * 8 + Xo * 4, Ys * (n + 1) + Yo), 0
      graphfrm.Picture1.Line (Xs * 8, Ys * (n + 1) + Yo)-(Xs * 8 + Xo * 4, Ys * (n + 1) + Yo), 0
      graphfrm.Picture1.Line (Xs * 8 + Xo * 4, Ys * (n + 1))-(X1, Ys * (n + 1))
      graphfrm.Picture1.Circle (Xs * 8 + Xo / 2, Ys * (n + 1) - Yo / 2), 0.01
      graphfrm.Picture1.Print ("END")
End Sub


Sub send()
  fMainForm.MSComm1.CommPort = 2
  fMainForm.MSComm1.Settings = "9600,n,8,1"
  fMainForm.MSComm1.PortOpen = True
  fn = FreeFile
  Open "Ram.hex" For Input As #fn
  While Not EOF(fn)
    Sum = Sum + 1
    fMainForm.MSComm1.Output = Input(1, #fn)
  Wend
  MsgBox ("ได้ทำการส่งข้อมูล ลงบน PLC เรียบร้อยแล้ว")
  fMainForm.MSComm1.PortOpen = False
  Close #fn
End Sub

############################################################
#########################Compile#############################
```

```
Sub convest()
  readvalue
  fn = FreeFile
  Open "ram.asm" For Output As #fn
  Print #fn, "CPU " + Chr(34) + "8051.TBL" + Chr(34)
  Print #fn, "HOF " + Chr(34) + "INT8" + Chr(34)
  Print #fn, "ORG 6000H"
  Print #fn, "B:  EQU  0F0H"
  Print #fn, "TMOD:  EQU  89H"
  Print #fn, "TCON:  EQU  88H"
  Print #fn, "TH0:  EQU  8CH"
  Print #fn, "TL0:  EQU  8AH"
  Print #fn, "DPH: EQU  83H"
  Print #fn, "DPL: EQU  82H"
  For i = 1 To 800
```

```
If Comd(i) = "LD" Then
   If Value(i) = "X0" Then
      Print #fn, "MOV    DPTR,#7800h"
      Print #fn, "MOVX   A,@DPTR"
      Print #fn, "CPL    A"
      Print #fn, "ANL    A,#00000001b"
      Print #fn, "MOV    R0,A"
      Print #fn, "MOV    R1,#00000001b"
   End If
   If Value(i) = "X1" Then
      Print #fn, "MOV    DPTR,#7800h"
      Print #fn, "MOVX   A,@DPTR"
      Print #fn, "CPL    A"
      Print #fn, "ANL    A,#00000010b"
      Print #fn, "MOV    R0,A"
      Print #fn, "MOV    R1,#00000010b"
   End If
   If Value(i) = "X2" Then
      Print #fn, "MOV    DPTR,#7800h"
      Print #fn, "MOVX   A,@DPTR"
      Print #fn, "CPL    A"
      Print #fn, "ANL    A,#00000100b"
      Print #fn, "MOV    R0,A"
      Print #fn, "MOV    R1,#00000100b"
   End If
   If Value(i) = "X3" Then
      Print #fn, "MOV    DPTR,#7800h"
      Print #fn, "MOVX   A,@DPTR"
      Print #fn, "CPL    A"
      Print #fn, "ANL    A,#00001000b"
      Print #fn, "MOV    R0,A"
      Print #fn, "MOV    R1,#00001000b"
   End If
   If Value(i) = "X4" Then
      Print #fn, "MOV    DPTR,#7800h"
      Print #fn, "MOVX   A,@DPTR"
      Print #fn, "CPL    A"
      Print #fn, "ANL    A,#00010000b"
      Print #fn, "MOV    R0,A"
      Print #fn, "MOV    R1,#00010000b"
   End If
   If Value(i) = "X5" Then
      Print #fn, "MOV    DPTR,#7800h"
      Print #fn, "MOVX   A,@DPTR"
      Print #fn, "CPL    A"
      Print #fn, "ANL    A,#00100000b"
      Print #fn, "MOV    R0,A"
```

```
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "X6" Then
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "X7" Then
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
  End If                                    '.....LD...X
  If Value(i) = "Y0" Then
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000001b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "Y1" Then
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "Y2" Then
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "Y3" Then
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "Y4" Then
```

```
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "Y5" Then
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00100000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "Y6" Then
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "Y7" Then
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
  End If                                    '.....LD.... Y...
  If Value(i) = "M0" Then
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000001b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "M1" Then
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "M2" Then
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
```

```
        End If
        If Value(i) = "M3" Then
           Print #fn, "MOV    DPTR,#7802h"
           Print #fn, "MOVX   A,@DPTR"
           Print #fn, "ANL    A,#00001000b"
           Print #fn, "MOV    R0,A"
           Print #fn, "MOV    R1,#00001000b"
        End If
        If Value(i) = "M4" Then
           Print #fn, "MOV    DPTR,#7802h"
           Print #fn, "MOVX   A,@DPTR"
           Print #fn, "ANL    A,#00010000b"
           Print #fn, "MOV    R0,A"
           Print #fn, "MOV    R1,#00010000b"
        End If
        If Value(i) = "M5" Then
           Print #fn, "MOV    DPTR,#7802h"
           Print #fn, "MOVX   A,@DPTR"
           Print #fn, "ANL    A,#00100000b"
           Print #fn, "MOV    R0,A"
           Print #fn, "MOV    R1,#00100000b"
        End If
        If Value(i) = "M6" Then
           Print #fn, "MOV    DPTR,#7802h"
           Print #fn, "MOVX   A,@DPTR"
           Print #fn, "ANL    A,#01000000b"
           Print #fn, "MOV    R0,A"
           Print #fn, "MOV    R1,#01000000b"
        End If
        If Value(i) = "M7" Then
           Print #fn, "MOV    DPTR,#7802h"
           Print #fn, "MOVX   A,@DPTR"
           Print #fn, "ANL    A,#10000000b"
           Print #fn, "MOV    R0,A"
           Print #fn, "MOV    R1,#10000000b"
        End If                                  '.....LD....M0-7...
End If
'.....LDI.....
     If Comd(i) = "LDI" Then
        If Value(i) = "X0" Then
           Print #fn, "MOV    DPTR,#7800h"
           Print #fn, "MOVX   A,@DPTR"
           Print #fn, "ANL    A,#00000001b"
           Print #fn, "MOV    R0,A"
           Print #fn, "MOV    R1,#00000001b"
        End If
        If Value(i) = "X1" Then
```

```
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "X2" Then
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "X3" Then
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "X4" Then
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "X5" Then
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00100000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "X6" Then
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "X7" Then
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
```

```
End If                                  '....LDI.....X0-7...
If Value(i) = "Y0" Then
   Print #fn, "MOV    DPTR,#7801h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "CPL    A"
   Print #fn, "ANL    A,#00000001b"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#00000001b"
End If
If Value(i) = "Y1" Then
   Print #fn, "MOV    DPTR,#7801h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "CPL    A"
   Print #fn, "ANL    A,#00000010b"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#00000010b"
End If
If Value(i) = "Y2" Then
   Print #fn, "MOV    DPTR,#7801h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "CPL    A"
   Print #fn, "ANL    A,#00000100b"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#00000100b"
End If
If Value(i) = "Y3" Then
   Print #fn, "MOV    DPTR,#7801h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "CPL    A"
   Print #fn, "ANL    A,#00001000b"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#00001000b"
End If
If Value(i) = "Y4" Then
   Print #fn, "MOV    DPTR,#7801h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "CPL    A"
   Print #fn, "ANL    A,#00010000b"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#00010000b"
End If
If Value(i) = "Y5" Then
   Print #fn, "MOV    DPTR,#7801h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "CPL    A"
   Print #fn, "ANL    A,#00100000b"
   Print #fn, "MOV    R0,A"
```

```
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "Y6" Then
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "Y7" Then
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
  End If                                  '.....LDI....Y...
  If Value(i) = "M0" Then
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000001b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "M1" Then
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "M2" Then
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "M3" Then
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00001000b"
```

```
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#00001000b"
      End If
      If Value(i) = "M4" Then
        Print #fn, "MOV    DPTR,#7802h"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "CPL    A"
        Print #fn, "ANL    A,#00010000b"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#00010000b"
      End If
      If Value(i) = "M5" Then
        Print #fn, "MOV    DPTR,#7802h"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "CPL    A"
        Print #fn, "ANL    A,#00100000b"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#00100000b"
      End If
      If Value(i) = "M6" Then
        Print #fn, "MOV    DPTR,#7802h"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "CPL    A"
        Print #fn, "ANL    A,#01000000b"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#01000000b"
      End If
      If Value(i) = "M7" Then
        Print #fn, "MOV    DPTR,#7802h"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "CPL    A"
        Print #fn, "ANL    A,#10000000b"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#10000000b"
      End If                                  '.....LDI....M0-7...
    End If
'.....0R.....
    If Comd(i) = "OR" Then
      If Value(i) = "X0" Then
        Print #fn, "MOV    B,#00000001b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7800h"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "CPL    A"
        Print #fn, "ANL    A,#00000001b"
        Print #fn, "ORL    A,R0"
        Print #fn, "MOV    R0,A"
```

```
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "X1" Then
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "X2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "X3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "X4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "X5" Then
```

```
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00100000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "X6" Then
    Print #fn, "MOV    B,#01000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "X7" Then
    Print #fn, "MOV    B,#10000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
  End If                              '...OR...X0-7...
  If Value(i) = "Y0" Then
    Print #fn, "MOV    B,#00000001b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000001b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "Y1" Then
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
```

```
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "Y2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "Y3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "Y4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "Y5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00100000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "Y6" Then
    Print #fn, "MOV    B,#01000000b"
```

```
    Print #fn, "LCALL   Change"
    Print #fn, "MOV     DPTR,#7801h"
    Print #fn, "MOVX    A,@DPTR"
    Print #fn, "ANL     A,#01000000b"
    Print #fn, "ORL     A,R0"
    Print #fn, "MOV     R0,A"
    Print #fn, "MOV     R1,#01000000b"
  End If
  If Value(i) = "Y7" Then
    Print #fn, "MOV     B,#10000000b"
    Print #fn, "LCALL   Change"
    Print #fn, "MOV     DPTR,#7801h"
    Print #fn, "MOVX    A,@DPTR"
    Print #fn, "ANL     A,#10000000b"
    Print #fn, "ORL     A,R0"
    Print #fn, "MOV     R0,A"
    Print #fn, "MOV     R1,#10000000b"
  End If                                    '...OR...Y0-7...
  If Value(i) = "M0" Then
    Print #fn, "MOV     B,#00000001b"
    Print #fn, "LCALL   Change"
    Print #fn, "MOV     DPTR,#7802h"
    Print #fn, "MOVX    A,@DPTR"
    Print #fn, "ANL     A,#00000001b"
    Print #fn, "ORL     A,R0"
    Print #fn, "MOV     R0,A"
    Print #fn, "MOV     R1,#00000001b"
  End If
  If Value(i) = "M1" Then
    Print #fn, "MOV     B,#00000010b"
    Print #fn, "LCALL   Change"
    Print #fn, "MOV     DPTR,#7802h"
    Print #fn, "MOVX    A,@DPTR"
    Print #fn, "ANL     A,#00000010b"
    Print #fn, "ORL     A,R0"
    Print #fn, "MOV     R0,A"
    Print #fn, "MOV     R1,#00000010b"
  End If
  If Value(i) = "M2" Then
    Print #fn, "MOV     B,#00000100b"
    Print #fn, "LCALL   Change"
    Print #fn, "MOV     DPTR,#7802h"
    Print #fn, "MOVX    A,@DPTR"
    Print #fn, "ANL     A,#00000100b"
    Print #fn, "ORL     A,R0"
    Print #fn, "MOV     R0,A"
    Print #fn, "MOV     R1,#00000100b"
```

```
End If
If Value(i) = "M3" Then
   Print #fn, "MOV    B,#00001000b"
   Print #fn, "LCALL  Change"
   Print #fn, "MOV    DPTR,#7802h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "ANL    A,#00001000b"
   Print #fn, "ORL    A,R0"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#00001000b"
End If
If Value(i) = "M4" Then
   Print #fn, "MOV    B,#00010000b"
   Print #fn, "LCALL  Change"
   Print #fn, "MOV    DPTR,#7802h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "ANL    A,#00010000b"
   Print #fn, "ORL    A,R0"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#00010000b"
End If
If Value(i) = "M5" Then
   Print #fn, "MOV    B,#00100000b"
   Print #fn, "LCALL  Change"
   Print #fn, "MOV    DPTR,#7802h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "ANL    A,#00100000b"
   Print #fn, "ORL    A,R0"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#00100000b"
End If
If Value(i) = "M6" Then
   Print #fn, "MOV    B,#01000000b"
   Print #fn, "LCALL  Change"
   Print #fn, "MOV    DPTR,#7802h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "ANL    A,#01000000b"
   Print #fn, "ORL    A,R0"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#01000000b"
End If
If Value(i) = "M7" Then
   Print #fn, "MOV    B,#10000000b"
   Print #fn, "LCALL  Change"
   Print #fn, "MOV    DPTR,#7802h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "ANL    A,#10000000b"
```

```
            Print #fn, "ORL    A,R0"
            Print #fn, "MOV    R0,A"
            Print #fn, "MOV    R1,#10000000b"
        End If                                  '...OR...M0-7...
End If


'.....0RI.....
    If Comd(i) = "ORI" Then
        If Value(i) = "X0" Then
            Print #fn, "MOV    B,#00000001b"
            Print #fn, "LCALL  Change"
            Print #fn, "MOV    DPTR,#7800h"
            Print #fn, "MOVX   A,@DPTR"
            Print #fn, "ANL    A,#00000001b"
            Print #fn, "ORL    A,R0"
            Print #fn, "MOV    R0,A"
            Print #fn, "MOV    R1,#00000001b"
        End If
        If Value(i) = "X1" Then
            Print #fn, "MOV    B,#00000010b"
            Print #fn, "LCALL  Change"
            Print #fn, "MOV    DPTR,#7800h"
            Print #fn, "MOVX   A,@DPTR"
            Print #fn, "ANL    A,#00000010b"
            Print #fn, "ORL    A,R0"
            Print #fn, "MOV    R0,A"
            Print #fn, "MOV    R1,#00000010b"
        End If
        If Value(i) = "X2" Then
            Print #fn, "MOV    B,#00000100b"
            Print #fn, "LCALL  Change"
            Print #fn, "MOV    DPTR,#7800h"
            Print #fn, "MOVX   A,@DPTR"
            Print #fn, "ANL    A,#00000100b"
            Print #fn, "ORL    A,R0"
            Print #fn, "MOV    R0,A"
            Print #fn, "MOV    R1,#00000100b"
        End If
        If Value(i) = "X3" Then
            Print #fn, "MOV    B,#00001000b"
            Print #fn, "LCALL  Change"
            Print #fn, "MOV    DPTR,#7800h"
            Print #fn, "MOVX   A,@DPTR"
            Print #fn, "ANL    A,#00001000b"
            Print #fn, "ORL    A,R0"
            Print #fn, "MOV    R0,A"
            Print #fn, "MOV    R1,#00001000b"
```

```
End If
If Value(i) = "X4" Then
   Print #fn, "MOV    B,#00010000b"
   Print #fn, "LCALL  Change"
   Print #fn, "MOV    DPTR,#7800h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "ANL    A,#00010000b"
   Print #fn, "ORL    A,R0"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#00010000b"
End If
If Value(i) = "X5" Then
   Print #fn, "MOV    B,#00100000b"
   Print #fn, "LCALL  Change"
   Print #fn, "MOV    DPTR,#7800h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "ANL    A,#00100000b"
   Print #fn, "ORL    A,R0"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#00100000b"
End If
If Value(i) = "X6" Then
   Print #fn, "MOV    B,#01000000b"
   Print #fn, "LCALL  Change"
   Print #fn, "MOV    DPTR,#7800h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "ANL    A,#01000000b"
   Print #fn, "ORL    A,R0"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#01000000b"
End If
If Value(i) = "X7" Then
   Print #fn, "MOV    B,#10000000b"
   Print #fn, "LCALL  Change"
   Print #fn, "MOV    DPTR,#7800h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "ANL    A,#10000000b"
   Print #fn, "ORL    A,R0"
   Print #fn, "MOV    R0,A"
   Print #fn, "MOV    R1,#10000000b"
End If                                  '...ORI...X0-7...
If Value(i) = "Y0" Then
   Print #fn, "MOV    B,#00000001b"
   Print #fn, "LCALL  Change"
   Print #fn, "MOV    DPTR,#7801h"
   Print #fn, "MOVX   A,@DPTR"
   Print #fn, "CPL    A"
```

```
    Print #fn, "ANL    A,#00000001b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "Y1" Then
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "Y2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "Y3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "Y4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
```

```
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "Y5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00100000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "Y6" Then
    Print #fn, "MOV    B,#01000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "Y7" Then
    Print #fn, "MOV    B,#10000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
  End If                                '...ORI...Y0-7...
  If Value(i) = "M0" Then
    Print #fn, "MOV    B,#00000001b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000001b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "M1" Then
```

```
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "M2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "M3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "M4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "M5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
```

```
        Print #fn, "MOVX    A,@DPTR"
        Print #fn, "CPL     A"
        Print #fn, "ANL     A,#00100000b"
        Print #fn, "ORL     A,R0"
        Print #fn, "MOV     R0,A"
        Print #fn, "MOV     R1,#00100000b"
      End If
      If Value(i) = "M6" Then
        Print #fn, "MOV     B,#01000000b"
        Print #fn, "LCALL   Change"
        Print #fn, "MOV     DPTR,#7802h"
        Print #fn, "MOVX    A,@DPTR"
        Print #fn, "CPL     A"
        Print #fn, "ANL     A,#01000000b"
        Print #fn, "ORL     A,R0"
        Print #fn, "MOV     R0,A"
        Print #fn, "MOV     R1,#01000000b"
      End If
      If Value(i) = "M7" Then
        Print #fn, "MOV     B,#10000000b"
        Print #fn, "LCALL   Change"
        Print #fn, "MOV     DPTR,#7802h"
        Print #fn, "MOVX    A,@DPTR"
        Print #fn, "CPL     A"
        Print #fn, "ANL     A,#10000000b"
        Print #fn, "ORL     A,R0"
        Print #fn, "MOV     R0,A"
        Print #fn, "MOV     R1,#10000000b"
      End If                                    '...ORL...M0-7...
    End If
'.....................AND.....................
    If Comd(i) = "AND" Then
      If Value(i) = "X0" Then
        Print #fn, "MOV     B,#00000001b"
        Print #fn, "LCALL   Change"
        Print #fn, "MOV     DPTR,#7800h"
        Print #fn, "MOVX    A,@DPTR"
        Print #fn, "CPL     A"
        Print #fn, "ANL     A,#00000001b"
        Print #fn, "ANL     A,R0"
        Print #fn, "MOV     R0,A"
        Print #fn, "MOV     R1,#00000001b"
      End If
      If Value(i) = "X1" Then
        Print #fn, "MOV     B,#00000010b"
        Print #fn, "LCALL   Change"
        Print #fn, "MOV     DPTR,#7800h"
```

```
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "X2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "X3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "X4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "X5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00100000b"
```

```
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "X6" Then
    Print #fn, "MOV    B,#01000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "X7" Then
    Print #fn, "MOV    B,#10000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
  End If                                    '...AND...X0-7...
  If Value(i) = "Y0" Then
    Print #fn, "MOV    B,#00000001b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000001b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "Y1" Then
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "Y2" Then
```

```
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "Y3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "Y4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "Y5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00100000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "Y6" Then
    Print #fn, "MOV    B,#01000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
```

```
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "Y7" Then
    Print #fn, "MOV    B,#10000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
  End If                                '...AND...Y0-7...
  If Value(i) = "M0" Then
    Print #fn, "MOV    B,#00000001b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000001b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "M1" Then
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "M2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "M3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
```

```
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "M4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "M5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00100000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "M6" Then
    Print #fn, "MOV    B,#01000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "M7" Then
    Print #fn, "MOV    B,#10000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
  End If                              '...AND...M0-7...

End If
```

```
'.....ANI.....'
    If Comd(i) = "ANI" Then
      If Value(i) = "X0" Then
        Print #fn, "MOV    B,#00000001b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7800h"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "ANL    A,#00000001b"
        Print #fn, "ANL    A,R0"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#00000001b"
      End If
      If Value(i) = "X1" Then
        Print #fn, "MOV    B,#00000010b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7800h"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "ANL    A,#00000010b"
        Print #fn, "ANL    A,R0"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#00000010b"
      End If
      If Value(i) = "X2" Then
        Print #fn, "MOV    B,#00000100b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7800h"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "ANL    A,#00000100b"
        Print #fn, "ANL    A,R0"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#00000100b"
      End If
      If Value(i) = "X3" Then
        Print #fn, "MOV    B,#00001000b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7800h"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "ANL    A,#00001000b"
        Print #fn, "ANL    A,R0"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#00001000b"
      End If
      If Value(i) = "X4" Then
        Print #fn, "MOV    B,#00010000b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7800h"
        Print #fn, "MOVX   A,@DPTR"
```

```
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "X5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#00100000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "X6" Then
    Print #fn, "MOV    B,#01000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "X7" Then
    Print #fn, "MOV    B,#10000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7800h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
  End If                    '...ANDI...X0-7...
  If Value(i) = "Y0" Then
    Print #fn, "MOV    B,#00000001b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000001b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "Y1" Then
```

```
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000010b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "Y2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "Y3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "Y4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "Y5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
```

```
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00100000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
  End If
  If Value(i) = "Y6" Then
    Print #fn, "MOV    B,#01000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#01000000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
  End If
  If Value(i) = "Y7" Then
    Print #fn, "MOV    B,#10000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#10000000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
  End If                                    '...ANDI...Y0-7...
  If Value(i) = "M0" Then
    Print #fn, "MOV    B,#00000001b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000001b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
  End If
  If Value(i) = "M1" Then
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000010b"
```

```
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
  End If
  If Value(i) = "M2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00000100b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
  End If
  If Value(i) = "M3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00001000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
  End If
  If Value(i) = "M4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00010000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
  End If
  If Value(i) = "M5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802h"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,#00100000b"
    Print #fn, "ANL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
```

```
    End If
    If Value(i) = "M6" Then
      Print #fn, "MOV    B,#01000000b"
      Print #fn, "LCALL  Change"
      Print #fn, "MOV    DPTR,#7802h"
      Print #fn, "MOVX   A,@DPTR"
      Print #fn, "CPL    A"
      Print #fn, "ANL    A,#01000000b"
      Print #fn, "ANL    A,R0"
      Print #fn, "MOV    R0,A"
      Print #fn, "MOV    R1,#01000000b"
    End If
    If Value(i) = "M7" Then
      Print #fn, "MOV    B,#10000000b"
      Print #fn, "LCALL  Change"
      Print #fn, "MOV    DPTR,#7802h"
      Print #fn, "MOVX   A,@DPTR"
      Print #fn, "CPL    A"
      Print #fn, "ANL    A,#10000000b"
      Print #fn, "ANL    A,R0"
      Print #fn, "MOV    R0,A"
      Print #fn, "MOV    R1,#10000000b"
    End If                                  '...ANDI...M0-7...
  End If
'.....SET.....
  If Comd(i) = "SET" Then
    If Value(i) = "Y0" Then
      Print #fn, "MOV    B,#00000001b"
      Print #fn, "LCALL  Change"
      Print #fn, "MOV    DPTR,#7801H"
      Print #fn, "MOVX   A,@DPTR"
      Print #fn, "ORL    A,R0"
      Print #fn, "MOV    R0,A"
      Print #fn, "MOV    R1,#00000001b"
      Print #fn, "MOV    DPTR,#0F000H"
      Print #fn, "MOVX   @DPTR,A"
      Print #fn, "MOV    DPTR,#7801H"
      Print #fn, "MOVX   @DPTR,A"
    End If
    If Value(i) = "Y1" Then
      Print #fn, "MOV    B,#00000010b"
      Print #fn, "LCALL  Change"
      Print #fn, "MOV    DPTR,#7801H"
      Print #fn, "MOVX   A,@DPTR"
      Print #fn, "ORL    A,R0"
      Print #fn, "MOV    R0,A"
      Print #fn, "MOV    R1,#00000010b"
```

```
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
```

```
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "ORL    A,R0"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#00100000b"
        Print #fn, "MOV    DPTR,#0F000H"
        Print #fn, "MOVX   @DPTR,A"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   @DPTR,A"
      End If
      If Value(i) = "Y6" Then
        Print #fn, "MOV    B,#01000000b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "ORL    A,R0"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#01000000b"
        Print #fn, "MOV    DPTR,#0F000H"
        Print #fn, "MOVX   @DPTR,A"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   @DPTR,A"
      End If
      If Value(i) = "Y7" Then
        Print #fn, "MOV    B,#10000000b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "ORL    A,R0"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#10000000b"
        Print #fn, "MOV    DPTR,#0F000H"
        Print #fn, "MOVX   @DPTR,A"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   @DPTR,A"
      End If                                    '...SET...Y0-7...
    End If
'.....RST.....
    If Comd(i) = "RST" Then
      If Value(i) = "Y0" Then
        Print #fn, "MOV    B,#00000001b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "MOV    R2,A"
        Print #fn, "MOV    A,R0"
        Print #fn, "CPL    A"
```

```
    Print #fn, "ANL    A,R2"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y1" Then
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,R0"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,R0"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,R0"
```

```
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,R0"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,R0"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y6" Then
    Print #fn, "MOV    B,#01000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
```

```
        Print #fn, "MOV    A,R0"
        Print #fn, "CPL    A"
        Print #fn, "ANL    A,R2"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#01000000b"
        Print #fn, "MOV    DPTR,#0F000H"
        Print #fn, "MOVX   @DPTR,A"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   @DPTR,A"
      End If
      If Value(i) = "Y7" Then
        Print #fn, "MOV    B,#10000000b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "MOV    R2,A"
        Print #fn, "MOV    A,R0"
        Print #fn, "CPL    A"
        Print #fn, "ANL    A,R2"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#10000000b"
        Print #fn, "MOV    DPTR,#0F000H"
        Print #fn, "MOVX   @DPTR,A"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   @DPTR,A"
      End If                                    '...RST...Y0-7...
    End If
'.....OUT.....'
    If Comd(i) = "OUT" Then
      If Value(i) = "Y0" Then
        Print #fn, "MOV    B,#00000001b"
        Print #fn, "LCALL  Change"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   A,@DPTR"
        Print #fn, "MOV    R2,A"
        Print #fn, "MOV    A,#00000001b"
        Print #fn, "CPL    A"
        Print #fn, "ANL    A,R2"
        Print #fn, "ORL    A,R0"
        Print #fn, "MOV    R0,A"
        Print #fn, "MOV    R1,#00000001b"
        Print #fn, "MOV    DPTR,#0F000H"
        Print #fn, "MOVX   @DPTR,A"
        Print #fn, "MOV    DPTR,#7801H"
        Print #fn, "MOVX   @DPTR,A"
      End If
      If Value(i) = "Y1" Then
```

```
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00000010b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00000100b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00001000b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
```

```
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00010000b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00100000b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y6" Then
    Print #fn, "MOV    B,#01000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#01000000b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
```

```
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "Y7" Then
    Print #fn, "MOV    B,#10000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#10000000b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
    Print #fn, "MOV    DPTR,#0F000H"
    Print #fn, "MOVX   @DPTR,A"
    Print #fn, "MOV    DPTR,#7801H"
    Print #fn, "MOVX   @DPTR,A"
  End If                    '...OUT..Y0-7...
  If Value(i) = "M0" Then
    Print #fn, "MOV    B,#00000001b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00000001b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000001b"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "M1" Then
    Print #fn, "MOV    B,#00000010b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00000010b"
    Print #fn, "CPL    A"
```

```
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000010b"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "M2" Then
    Print #fn, "MOV    B,#00000100b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00000100b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00000100b"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "M3" Then
    Print #fn, "MOV    B,#00001000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00001000b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00001000b"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "M4" Then
    Print #fn, "MOV    B,#00010000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00010000b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
```

```
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00010000b"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "M5" Then
    Print #fn, "MOV    B,#00100000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#00100000b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#00100000b"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "M6" Then
    Print #fn, "MOV    B,#01000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#01000000b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#01000000b"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   @DPTR,A"
  End If
  If Value(i) = "M7" Then
    Print #fn, "MOV    B,#10000000b"
    Print #fn, "LCALL  Change"
    Print #fn, "MOV    DPTR,#7802H"
    Print #fn, "MOVX   A,@DPTR"
    Print #fn, "MOV    R2,A"
    Print #fn, "MOV    A,#10000000b"
    Print #fn, "CPL    A"
    Print #fn, "ANL    A,R2"
    Print #fn, "ORL    A,R0"
    Print #fn, "MOV    R0,A"
    Print #fn, "MOV    R1,#10000000b"
```

```
        Print #fn, "MOV    DPTR,#7802H"
        Print #fn, "MOVX   @DPTR,A"
      End If                    '...OUT..M0-7...
    End If
  Next
  Print #fn, "LJMP   000EH"
  Print #fn, "Change:"
  Print #fn, "LOOPA1: MOV    A,R0"
  Print #fn, "RR     A"
  Print #fn, "MOV    R0,A"
  Print #fn, "MOV    A,R1"
  Print #fn, "RR     A"
  Print #fn, "MOV    R1,A"
  Print #fn, "CJNE   A,B,LOOPA1"
  Print #fn, "RET"
  Print #fn, "END"
  Close #fn
  Shell "trans.bat", vbNormalFocus
End Sub
```

ข

โปรแกรมมอนิเตอร์

```
;************************************HEAD CODE 8051*****************
        CPU     "8051.TBL"
        HOF     "INT8"
;*******************************FIX BOAUDRATE 9600*****************
        RAM     EQU     6000H
;*********************************START PROGRAM*****************
        ORG     0000H
;*********************************SET INITIAL VALUE*****************
START:lcall     initial_value
;*********************************SET       SERIAL PORT***************
        mov     scon,#52h
        mov     tmod,#20h
        mov     th1,#0fdh
        setb    tr1
;*********************************RECEIVER INPUT******************
BACK: lcall     input
;*********************************CHECK STATUS RUN/STOP***********
        mov     a,p1
        anl     a,#00000001b
        cjne    a,#00000001b,stop
;*******************************RUN PROGRAM PLC AT 6000H***********
RUN:    setb    p1.2
        clr     p1.3
        ljmp    6000h
;*********************************STOP FOR BURN PROGRAM***********
STOP:   clr     p1.2
        setb    p1.3
```

```
;**********************RECEIVER PROGRAMPLC FROM COMPUTER*****
        mov     dptr,#7000h
STOP_RECEIVER:jnb   ri,$
        clr     ri
        mov     a,sbuf
        movx    @dptr,a
        inc     dptr
        cjne    a,#31h, stop_receiver
        jnb     ri,$
        clr     ri
        mov     a,sbuf
        movx    @dptr,a
        inc     dptr
        cjne    a,#46h, stop_receiver
        jnb     ri,$
        clr     ri
        mov     a,sbuf
        movx    @dptr,a
        inc     dptr
        cjne    a,#46h, stop_receiver
;**************************CHANG ASCII TO HEX*******************
        mov     dptr,#7000h
STOP_AGIAN: movx    a,@dptr
        inc     dptr
        cjne    a,#3ah,stop_agian
        lcall   hex
        mov     r7,a
        lcall   hex
        mov     r6,a
        lcall   hex
        mov     r5,a
        lcall   hex
```

```
        cjne    a,#00h,stop_end
STOP_AGIAN1:lcall   hex
        push    dph
        push    dpl
        mov     dph,r6
        mov     dpl,r5
        movx    @dptr,a
        inc     dptr
        mov     r6,dph
        mov     r5,dpl
        mov     r4,dph
        mov     r3,dpl
        pop     dpl
        pop     dph
        djnz    r7,stop_agian1
        sjmp    stop_agian
STOP_END: cjne      a,#01h,stop_error
        push    dph
        push    dpl
        mov     dph,r4
        mov     dpl,r3
        movx    @dptr,a
        inc     dptr
        mov     r4,dph
        mov     r3,dpl
        pop     dpl
        pop     dph
        lcall   hex
        cjne    a,#0ffh,stop_error
        push    dph
        push    dpl
        mov     dph,r4
```

```
        mov     dpl,r3
        movx    @dptr,a
        inc     dptr
        mov     r4,dph
        mov     r3,dpl
        pop     dpl
        pop     dph
        lcall   hex
        lcall   beep
        lcall   delay
        lcall   delay
        lcall   beep
        jmp     $
STOP_ERROR: setb    p1.1
        jmp     $
;*********************************FUNTION INITIAL VALUE*************
INITIAL_VALUE:clr   p1.1
        clr     p1.2
        clr     p1.3
        mov     dptr,#0f000h
        mov     a,#00h
        movx    @dptr,a
        mov     dptr,#7801h
        movx    @dptr,a
        RET
;*****************************FUNCTION INPUT**********************
INPUT:dptr,#0f000h
        movx    a,@dptr
        mov     dptr,#7800h
        movx    @dptr,a
        RET
```

```
;*******************************FUNTION DELAY TIME****************
DELAY:mov   r1,#0ffh
DELAY2:mov  r2,#0f0h
       djnz  r2,$
       djnz  r1,delay2
       RET
;*******************************FUNTION BEEP VOICE*******************
BEEP:  setb  p1.1
       mov   r0,#0ffh
       mov   tmod,#01h
BEEP_2:mov   th0,#0ffh
       mov   tl0,#01ah
       setb  tr0
       jnb   tf0,$
       clr   tr0
       clr   tf0
       djnz  r0,beep_2
       clr   p1.1
       RET
;*****************************FUNCTION ASCII TO HEX*****************
HEX:   mov   r1,#01h
AGIAN:movx   a,@dptr
       inc   dptr
       clr   c
       subb  a,#40h
       jc    na_f
       add   a,#09h
NA_F:  anl   a,#0fh
       cjne  r1,#00h,swapp
       orl   a,r2
       sjmp  exit
SWAPP:swap   a
```

```
        mov     r2,a
        dec     r1
        ajmp    agian
EXIT:   RET
;**********************************END PROGRAM*******************
        END
```

ค

DATA SHEET

Features

- Compatible with MCS-51™ Products
- 8K Bytes of In-System Reprogrammable Downloadable Flash Memory
  - SPI Serial Interface for Program Downloading
  - Endurance: 1,000 Write/Erase Cycles
- 2K Bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
- 4V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Nine Interrupt Sources
- Programmable UART Serial Channel
- SPI Serial Interface
- Low-power Idle and Power-down Modes
- Interrupt Recovery From Power-down
- Programmable Watchdog Timer
- Dual Data Pointer
- Power-off Flag

Description

The AT89S8252 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of downloadable Flash programmable and erasable read only memory and 2K bytes of EEPROM. The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 instruction set and pinout. The on-chip downloadable Flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with downloadable Flash on a monolithic chip, the Atmel AT89S8252 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89S8252 provides the following standard features: 8K bytes of downloadable Flash, 2K bytes of EEPROM, 256 bytes of RAM, 32 I/O lines, programmable watchdog timer, two data pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89S8252 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

The downloadable Flash can be changed a single byte at a time and is accessible through the SPI serial interface. Holding RESET active forces the SPI bus into a serial programming interface and allows the program memory to be written to or read from unless Lock Bit 2 has been activated.

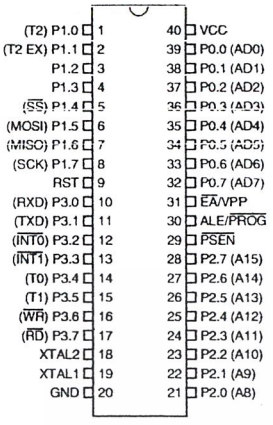
8-bit Microcontroller with 8K Bytes Flash

AT89S8252

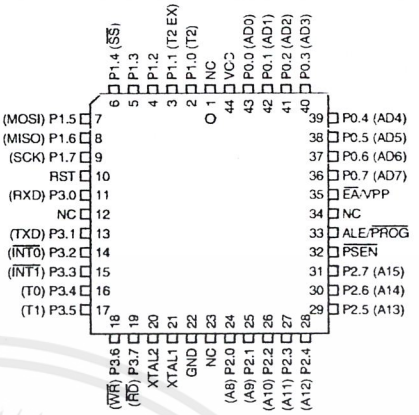
Rev. 0401E–02/00

Pin Configurations

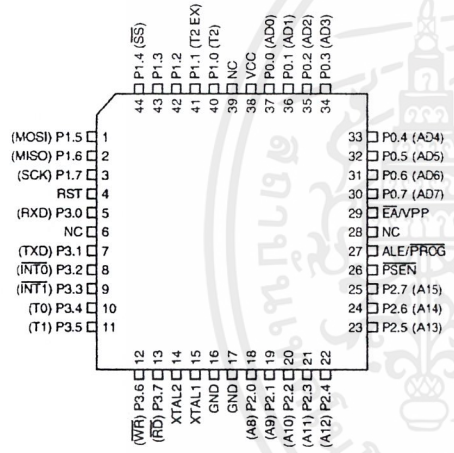
PDIP



PLCC



PQFP/TQFP



Pin Description

VCC

Supply voltage.

GND

Ground.

Port 0

Port 0 is an 8-bit open drain bbi-didirectional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

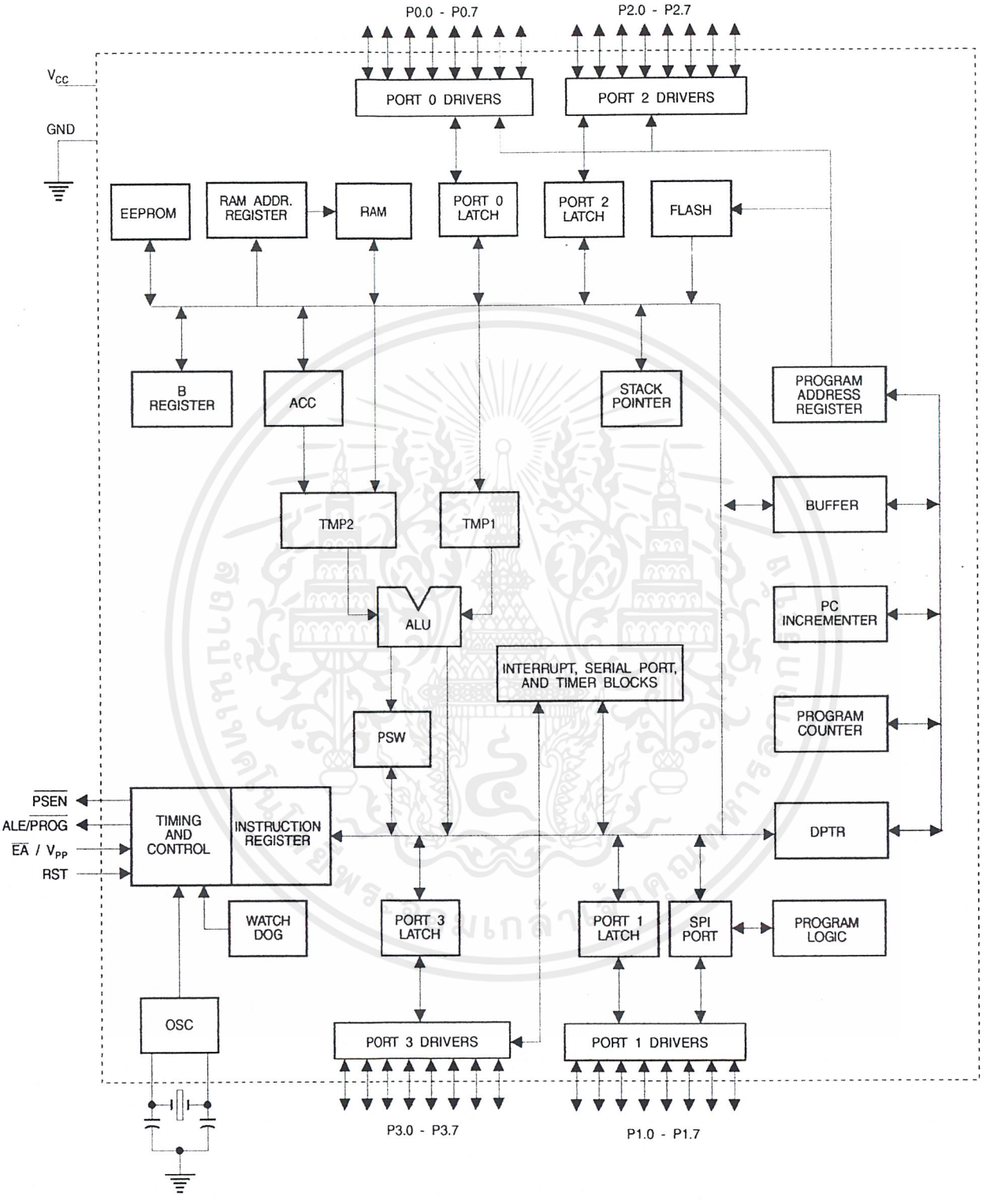
Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1

Port 1 is an 8-bit bi-directional I/O port with internal pullups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Block Diagram



Some Port 1 pins provide additional functions. P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively.

Pin Description

Furthermore, P1.4, P1.5, P1.6, and P1.7 can be configured as the SPI slave port select, data input/output and shift clock input/output pins as shown in the following table.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.4	$\overline{SS}$ (Slave port select input)
P1.5	MOSI (Master data output, slave data input pin for SPI channel)
P1.6	MISO (Master data input, slave data output pin for SPI channel)
P1.7	SCK (Master clock output, slave clock input pin for SPI channel)

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port 2

Port 2 is an 8-bit bi-directional I/O port with internal pullups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pullups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control

Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pullups.

Port 3 also serves the functions of various special features of the AT89S8252, as shown in the following table.

Port 3 also receives some control signals for Flash programming and verification.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$

Address Latch Enable is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

$\overline{PSEN}$

Program Store Enable is the read strobe to external pro-

HM62256B Series

32,768-word × 8-bit High Speed CMOS Static RAM

HITACHI

ADE-203-135D (Z)
Rev. 4.0
Nov. 29, 1995

Description

The Hitachi HM62256B is a CMOS static RAM organized 32-kword × 8-bit. It realizes higher performance and low power consumption by employing 0.8 µm Hi-CMOS process technology. The device, packaged in 8 × 14 mm TSOP, 8 × 13.4 mm TSOP with thickness of 1.2 mm, 450-mil SOP (foot print pitch width), 600-mil plastic DIP, or 300-mil plastic DIP, is available for high density mounting. It offers low power standby power dissipation; therefore, it is suitable for battery back-up systems.

Features

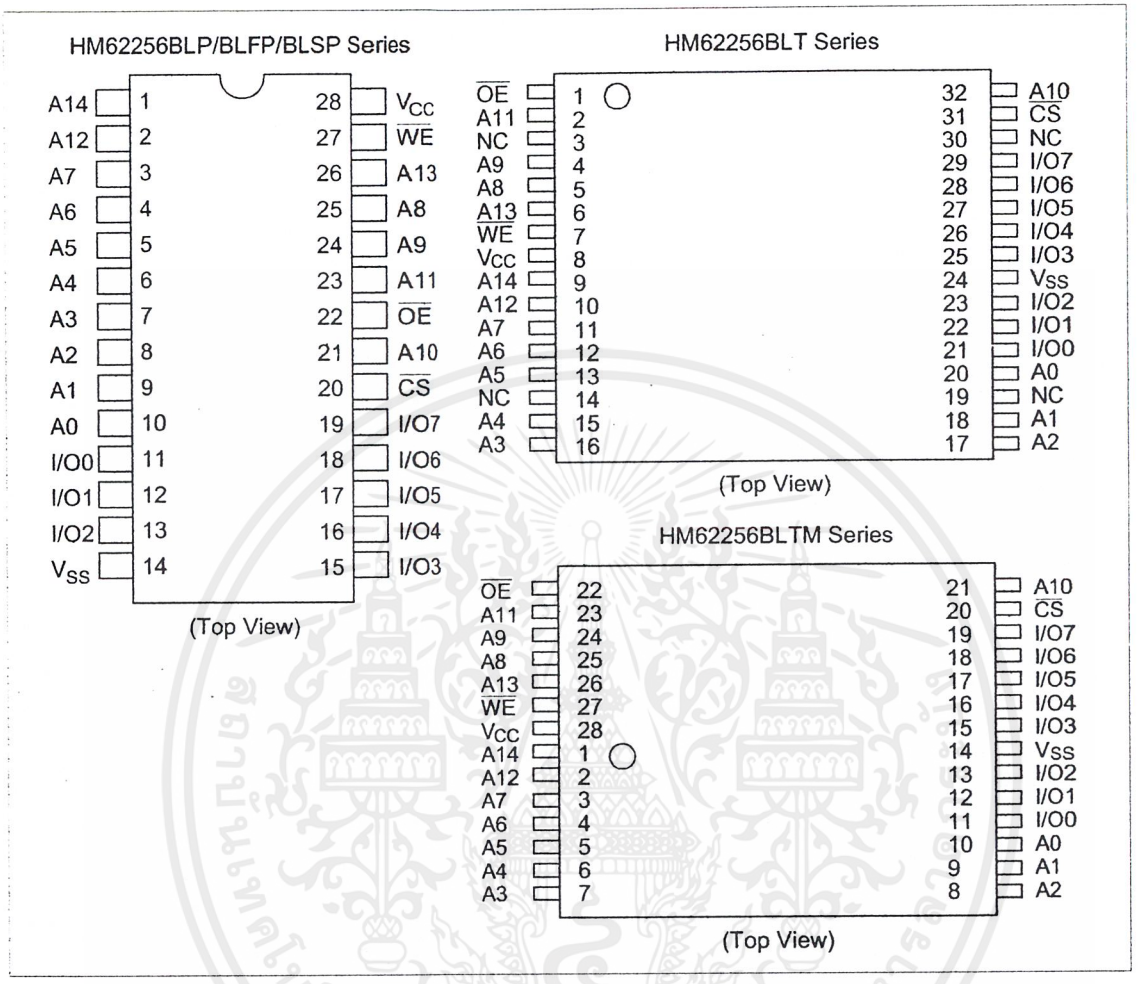
- High speed
  Fast access time: 45/55/70/85 ns (max)
- Low power
  Standby: 1.0 µW (typ)
  Operation: 25 mW (typ) (f = 1 MHz)
- Single 5 V supply
- Completely static memory
  No clock or timing strobe required
- Equal access and cycle times
- Common data input and output
  Three state output
- Directly TTL compatible
  All inputs and outputs
- Capability of battery back up operation

Ordering Information

Type No.	Access Time	Package
HM62256BLP-7	70 ns	600-mil 28-pin plastic DIP (DP-28)
HM62256BLP-7SL	70 ns	
HM62256BLSP-7	70 ns	300-mil 28-pin plastic DIP (DP-28NA)
HM62256BLSP-7SL	70 ns	
HM62256BLFP-7T	70 ns	450-mil 28-pin plastic SOP (FP-28DA)
HM62256BLFP-4SLT[*1]	45 ns	
HM62256BLFP-5SLT	55 ns	
HM62256BLFP-7SLT	70 ns	
HM62256BLFP-7ULT	70 ns	
HM62256BLT-8	85 ns	8 mm × 14 mm 32-pin TSOP (TFP-32DA)
HM62256BLT-7SL	70 ns	
HM62256BLTM-8	85 ns	8 mm × 13.4 mm 28-pin TSOP (TFP-28DA)
HM62256BLTM-4SL[*1]	45 ns	
HM62256BLTM-5SL	55 ns	
HM62256BLTM-7SL	70 ns	
HM62256BLTM-7UL	70 ns	

Note: 1. Under development

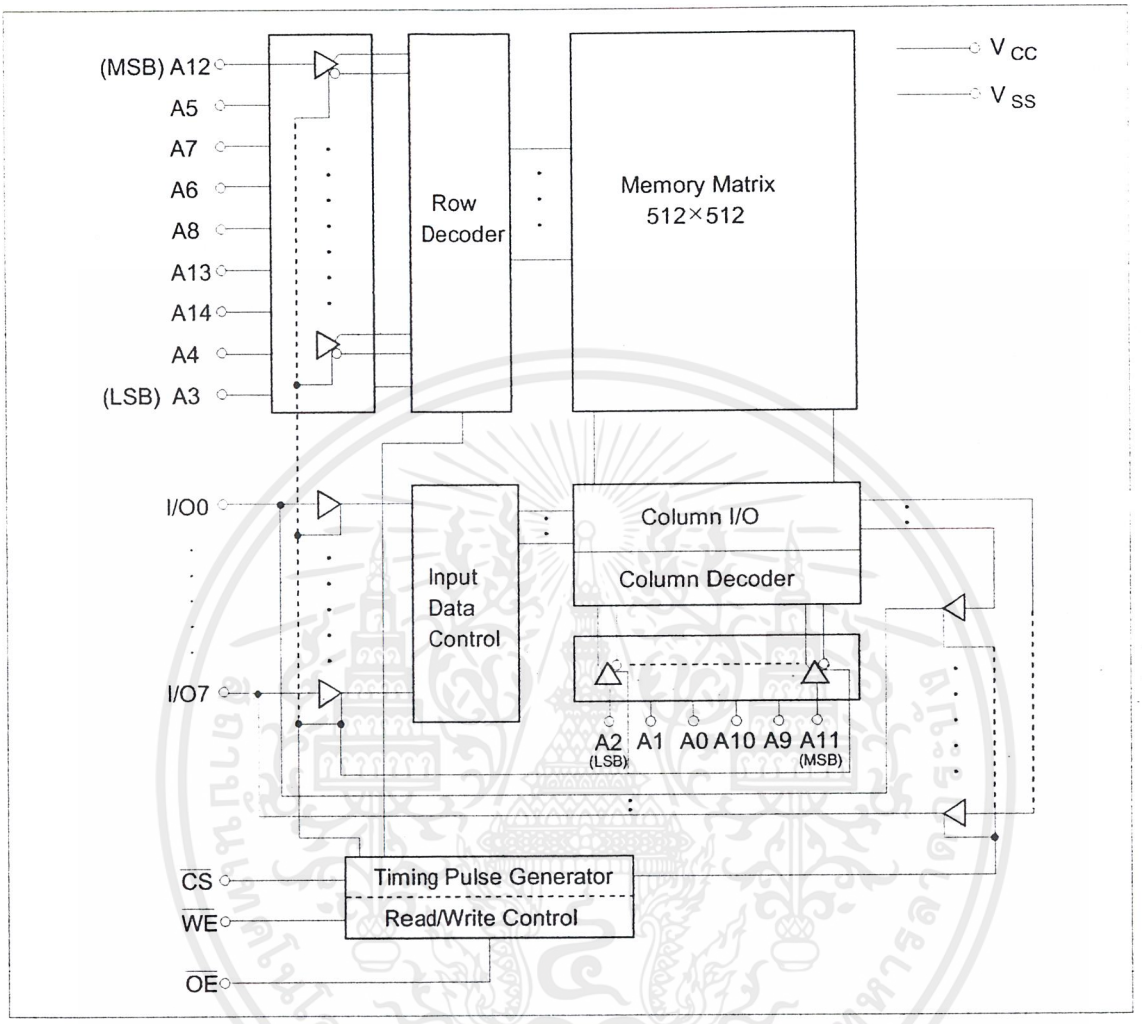
Pin Arrangement



Pin Description

Symbol	Function
A0 – A14	Address
I/O0 – I/O7	Input/output
$\overline{CS}$	Chip select
$\overline{WE}$	Write enable
$\overline{OE}$	Output enable
NC	No connection
V_{cc}	Power supply
V_{ss}	Ground

Block Diagram



Function Table

$\overline{WE}$	$\overline{CS}$	$\overline{OE}$	Mode	V_{cc} Current	I/O Pin	Ref. Cycle
X	H	X	Not selected	I_{SB}, I_{SB1}	High-Z	—
H	L	H	Output disable	I_{CC}	High-Z	—
H	L	L	Read	I_{CC}	Dout	Read cycle (1)–(3)
L	L	H	Write	I_{CC}	Din	Write cycle (1)
L	L	L	Write	I_{CC}	Din	Write cycle (2)

Note: X: H or L

Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Power supply voltage*1	V_{CC}	−0.5 to +7.0	V
Terminal voltage*1	V_T	−0.5*2 to V_{CC} + 0.3*3	V
Power dissipation	P_T	1.0	W
Operating temperature	Topr	0 to + 70	°C
Storage temperature	Tstg	−55 to +125	°C
Storage temperature under bias	Tbias	−10 to +85	°C

Notes: 1. Relative to V_{SS}
2. V_T min: −3.0 V for pulse half-width ≤ 50 ns
3. Maximum voltage is 7.0 V

Recommended DC Operating Conditions (Ta = 0 to +70°C)

Parameter	Symbol	Min	Typ	Max	Unit
Supply voltage	V_{CC}	4.5	5.0	5.5	V
	V_{SS}	0	0	0	V
Input high (logic 1) voltage	V_{IH}	2.2	—	V_{CC}+0.3	V
Input low (logic 0) voltage	V_{IL}	−0.5 *1	—	0.8	V

Note: 1. V_{IL} min: −3.0 V for pulse half-width ≤ 50 ns

DC Characteristics (Ta = 0 to +70°C, V_{CC} = 5 V ±10%, V_{SS} = 0 V)

Parameter		Symbol	Min	Typ*1	Max	Unit	Test Conditions
Input leakage current		$\|I_{LI}\|$	—	—	1	µA	Vin = V_{SS} to V_{CC}
Output leakage current		$\|I_{LO}\|$	—	—	1	µA	$\overline{CS}$ = V_{IH} or $\overline{OE}$ = V_{IH} or $\overline{WE}$ = V_{IL}, $V_{SS} \le V_{I/O} \le V_{CC}$
Operating power supply current		I_{CC}	—	6	15	mA	$\overline{CS}$ = V_{IL}, others = V_{IH}/V_{IL}, $I_{I/O}$ = 0 mA
Average operating power supply current	HM62256B-4	I_{CC1}	—	—	70	mA	min cycle, duty = 100 %, $I_{I/O}$ = 0 mA, $\overline{CS}$ = V_{IL}, others = V_{IH}/V_{IL}
	HM62256B-5	I_{CC1}	—	—	60		
	HM62256B-7	I_{CC1}	—	33	60		
	HM62256B-8	I_{CC1}	—	29	50		
		I_{CC2}	—	5	15	mA	Cycle time = 1 µs, $I_{I/O}$ = 0 mA, $\overline{CS}$ = V_{IL}, V_{IH} = V_{CC}, V_{IL} = 0
Standby power supply current		I_{SB}	—	0.3	2	mA	$\overline{CS}$ = V_{IH}
		I_{SB1}	—	0.2	100	µA	Vin ≥ 0 V, $\overline{CS}$ ≥ V_{CC} – 0.2 V,
			—	0.2*2	50*2		
				0.2*3	10*3		
Output low voltage		V_{OL}	—	—	0.4	V	I_{OL} = 2.1 mA
Output high voltage		V_{OH}	2.4	—	—	V	I_{OH} = –1.0 mA

Notes: 1. Typical values are at V_{CC} = 5.0 V, Ta = +25°C and not guaranteed.
2. This characteristics is guaranteed only for L-SL version.
3. This characteristics is guaranteed only for L-UL version.

Capacitance (Ta = 25°C, f = 1.0 MHz)*1

Parameter	Symbol	Min	Typ	Max	Unit	Test Conditions
Input capacitance*1	Cin	—	—	8	pF	Vin = 0 V
Input/output capacitance*1	$C_{I/O}$	—	—	10	pF	$V_{I/O}$ = 0 V

Note: 1. This parameter is sampled and not 100% tested.

FAIRCHILD
SEMICONDUCTOR™

August 1986
Revised March 2000

DM74LS138 • DM74LS139
Decoder/Demultiplexer

General Description

These Schottky-clamped circuits are designed to be used in high-performance memory-decoding or data-routing applications, requiring very short propagation delay times. In high-performance memory systems these decoders can be used to minimize the effects of system decoding. When used with high-speed memories, the delay times of these decoders are usually less than the typical access time of the memory. This means that the effective system delay introduced by the decoder is negligible.

The DM74LS138 decodes one-of-eight lines, based upon the conditions at the three binary select inputs and the three enable inputs. Two active-low and one active-high enable inputs reduce the need for external gates or inverters when expanding. A 24-line decoder can be implemented with no external inverters, and a 32-line decoder requires only one inverter. An enable input can be used as a data input for demultiplexing applications.

The DM74LS139 comprises two separate two-line-to-four-line decoders in a single package. The active-low enable input can be used as a data line in demultiplexing applications.

All of these decoders/demultiplexers feature fully buffered inputs, presenting only one normalized load to its driving circuit. All inputs are clamped with high-performance Schottky diodes to suppress line-ringing and simplify system design.

Features

- Designed specifically for high speed:
  - Memory decoders
  - Data transmission systems
- DM74LS138 3-to-8-line decoders incorporates 3 enable inputs to simplify cascading and/or data reception
- DM74LS139 contains two fully independent 2-to-4-line decoders/demultiplexers
- Schottky clamped for high performance
- Typical propagation delay (3 levels of logic)
  - DM74LS138 21 ns
  - DM74LS139 21 ns
- Typical power dissipation
  - DM74LS138 32 mW
  - DM74LS139 34 mW

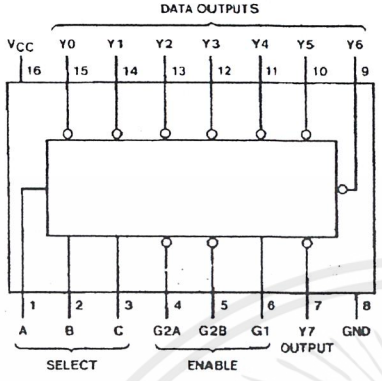
Ordering Code:

Order Number	Package Number	Package Description
DM74LS138M	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150 Narrow
DM74LS138SJ	M16D	16-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS138N	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide
DM74LS139M	M16A	16-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-012, 0.150 Narrow
DM74LS139SJ	M16D	16-Lead Small Outline Package (SOP), EIAJ TYPE II, 5.3mm Wide
DM74LS139N	N16E	16-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300 Wide

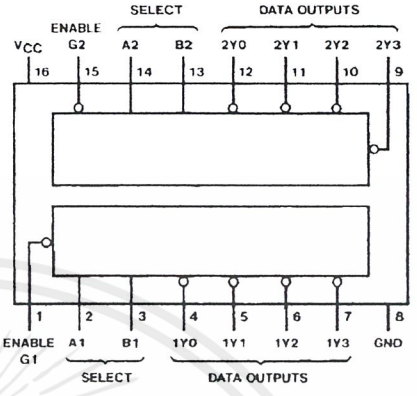
Devices also available in Tape and Reel. Specify by appending the suffix letter "X" to the ordering code.

Connection Diagrams

DM74LS138



DM74LS139



Function Tables

DM74LS138

Inputs					Outputs							
Enable		Select										
G1	G2 (Note 1)	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

DM74LS139

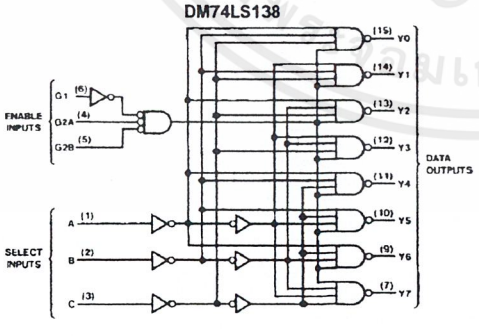
Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Level
L = LOW Level
X = Don't Care

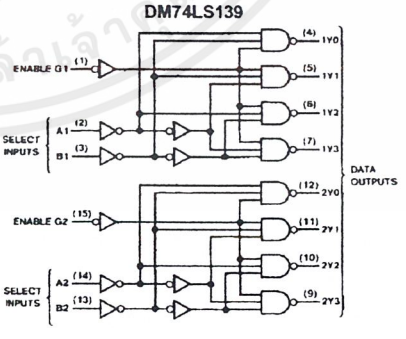
Note 1: G2 = G2A + G2B

Logic Diagrams

DM74LS138



DM74LS139



Absolute Maximum Ratings(Note 2)

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note 2: The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the Electrical Characteristics tables are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.

DM74LS138 Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage	2			V
V_{IL}	LOW Level Input Voltage			0.8	V
I_{OH}	HIGH Level Output Current			−0.4	mA
I_{OL}	LOW Level Output Current			8	mA
T_A	Free Air Operating Temperature	0		70	°C

DM74LS138 Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 3)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	2.7	3.4		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min		0.35	0.5	V
		I_{OL} = 4 mA, V_{CC} = Min		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.36	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 4)	−20		−100	mA
I_{CC}	Supply Current	V_{CC} = Max (Note 5)		6.3	10	mA

Note 3: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 4: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Note 5: I_{CC} is measured with all outputs enabled and OPEN.

DM74LS138 Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

Symbol	Parameter	From (Input) To (Output)	Levels of Delay	R_L = 2 kΩ				Units
				C_L = 15 pF		C_L = 50 pF		
				Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Select to Output	2		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Select to Output	2		27		40	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Select to Output	3		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Select to Output	3		27		40	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Enable to Output	2		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Enable to Output	2		24		40	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Enable to Output	3		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Enable to Output	3		28		40	ns

DM74LS139 Recommended Operating Conditions

Symbol	Parameter	Min	Nom	Max	Units
V_{CC}	Supply Voltage	4.75	5	5.25	V
V_{IH}	HIGH Level Input Voltage	2			V
V_{IL}	LOW Level Input Voltage			0.8	V
I_{OH}	HIGH Level Output Current			−0.4	mA
I_{OL}	LOW Level Output Current			8	mA
T_A	Free Air Operating Temperature	0		70	°C

DM74LS139 Electrical Characteristics

over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ (Note 6)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	HIGH Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	2.7	3.4		V
V_{OL}	LOW Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min		0.35	0.5	V
		I_{OL} = 4 mA, V_{CC} = Min		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V			0.1	mA
I_{IH}	HIGH Level Input Current	V_{CC} = Max, V_I = 2.7V			20	µA
I_{IL}	LOW Level Input Current	V_{CC} = Max, V_I = 0.4V			−0.36	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 7)	−20		−100	mA
I_{CC}	Supply Current	V_{CC} = Max (Note 8)		6.8	11	mA

Note 6: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 7: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Note 8: I_{CC} is measured with all outputs enabled and OPEN.

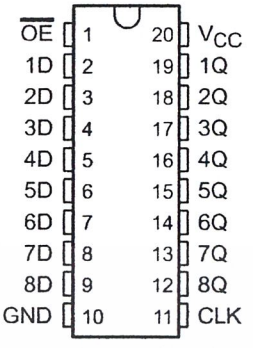
DM74LS139 Switching Characteristics

at V_{CC} = 5V and T_A = 25°C

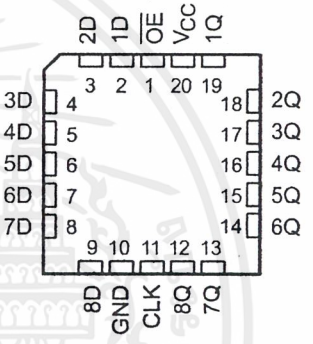
Symbol	Parameter	From (Input) To (Output)	R_L = 2 kΩ				Units
			C_L = 15 pF		C_L = 50 pF		
			Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Select to Output		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Select to Output		27		40	ns
t_{PLH}	Propagation Delay Time LOW-to-HIGH Level Output	Enable to Output		18		27	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Level Output	Enable to Output		24		40	ns

- **High-Current 3-State Noninverting Outputs Drive Bus Lines Directly or up to 15 LSTTL Loads**
- **Bus-Structured Pinout**
- **Package Options Include Plastic Small-Outline (DW) and Ceramic Flat (W) Packages, Ceramic Chip Carriers (FK), and Standard Plastic (N) and Ceramic (J) 300-mil DIPs**

SN54HC574 . . . J OR W PACKAGE
SN74HC574 . . . DW OR N PACKAGE
(TOP VIEW)



SN54HC574 . . . FK PACKAGE
(TOP VIEW)



description

These octal edge-triggered D-type flip-flops feature 3-state outputs designed specifically for bus driving. They are particularly suitable for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

The eight flip-flops enter data on the low-to-high transition of the clock (CLK) input.

A buffered output-enable ($\overline{OE}$) input can be used to place the eight outputs in either a normal logic state (high or low logic levels) or the high-impedance state. In the high-impedance state, the outputs neither load nor drive the bus lines significantly. The high-impedance state and increased drive provide the capability to drive bus lines without interface or pullup components.

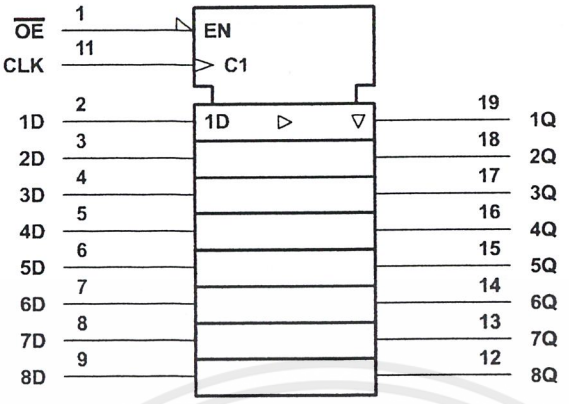
$\overline{OE}$ does not affect the internal operations of the flip-flops. Old data can be retained or new data can be entered while the outputs are in the high-impedance state.

The SN54HC574 is characterized for operation over the full military temperature range of –55°C to 125°C. The SN74HC574 is characterized for operation from –40°C to 85°C.

FUNCTION TABLE
(each flip-flop)

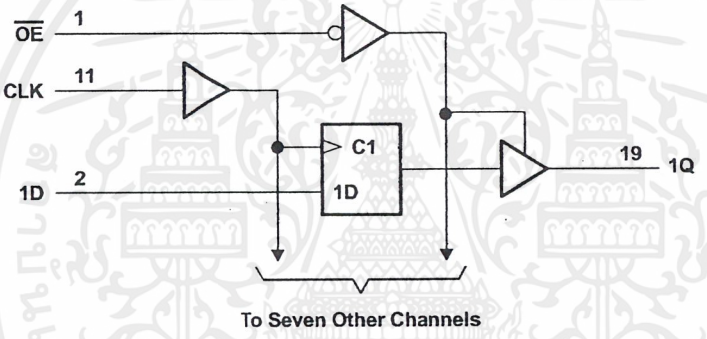
INPUTS			OUTPUT
OE	CLK	D	Q
L	↑	H	H
L	↑	L	L
L	H or L	X	Q_0
H	X	X	Z

logic symbol†



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range‡

Supply voltage range, V_{CC} ..... –0.5 V to 7 V
Input clamp current, I_{IK} ($V_I < 0$ or $V_I > V_{CC}$) (see Note 1) ..... ±20 mA
Output clamp current, I_{OK} ($V_O < 0$ or $V_O > V_{CC}$) (see Note 1) ..... ±20 mA
Continuous output current, I_O ($V_O = 0$ to V_{CC}) ..... ±35 mA
Continuous current through V_{CC} or GND ..... ±70 mA
Package thermal impedance, θ_{JA} (see Note 2): DW package ..... 97°C/W
N package ..... 67°C/W
Storage temperature range, T_{stg} ..... –65°C to 150°C

‡ Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. The input and output voltage ratings may be exceeded if the input and output current ratings are observed.
2. The package thermal impedance is calculated in accordance with JESD 51, except for through-hole packages, which use a trace length of zero.

recommended operating conditions

			SN54HC574			SN74HC574			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage		2	5	6	2	5	6	V
V_{IH}	High-level input voltage	V_{CC} = 2 V	1.5			1.5			V
		V_{CC} = 4.5 V	3.15			3.15			
		V_{CC} = 6 V	4.2			4.2			
V_{IL}	Low-level input voltage	V_{CC} = 2 V	0		0.5	0		0.5	V
		V_{CC} = 4.5 V	0		1.35	0		1.35	
		V_{CC} = 6 V	0		1.8	0		1.8	
V_I	Input voltage		0		V_{CC}	0		V_{CC}	V
V_O	Output voltage		0		V_{CC}	0		V_{CC}	V
t_t	Input transition (rise and fall) time	V_{CC} = 2 V	0		1000	0		1000	ns
		V_{CC} = 4.5 V	0		500	0		500	
		V_{CC} = 6 V	0		400	0		400	
T_A	Operating free-air temperature		–55		125	–40		85	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS		V_{CC}	T_A = 25°C			SN54HC574		SN74HC574		UNIT
				MIN	TYP	MAX	MIN	MAX	MIN	MAX	
V_{OH}	V_I = V_{IH} or V_{IL}	I_{OH} = –20 μA	2 V	1.9	1.998		1.9		1.9		V
			4.5 V	4.4	4.499		4.4		4.4		
			6 V	5.9	5.999		5.9		5.9		
		I_{OH} = –6 mA	4.5 V	3.98	4.3		3.7		3.84		
		I_{OH} = –7.8 mA	6 V	5.48	5.8		5.2		5.34		
V_{OL}	V_I = V_{IH} or V_{IL}	I_{OL} = 20 μA	2 V		0.002	0.1		0.1		0.1	V
			4.5 V		0.001	0.1		0.1		0.1	
			6 V		0.001	0.1		0.1		0.1	
		I_{OL} = 6 mA	4.5 V		0.17	0.26		0.4		0.33	
		I_{OL} = 7.8 mA	6 V		0.15	0.26		0.4		0.33	
I_I	V_I = V_{CC} or 0		6 V		±0.1	±100		±1000		±1000	nA
I_{OZ}	V_O = V_{CC} or 0		6 V		±0.01	±0.5		±10		±5	μA
I_{CC}	V_I = V_{CC} or 0,	I_O = 0	6 V			8		160		80	μA
C_i			2 V to 6 V		3	10		10		10	pF

DALLAS SEMICONDUCTOR

DS275
Line-Powered RS-232 Transceiver Chip

www.dalsemi.com

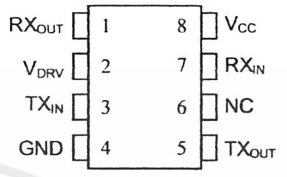
FEATURES

- Low-power serial transmitter/receiver for battery-backed systems
- Transmitter steals power from receive signal line to save power
- Ultra-low static current, even when connected to RS-232-E port
- Variable transmitter level from +5 to +12 volts
- Compatible with RS-232-E signals
- Available in 8-pin, 150 mil wide SOIC package (DS275S)
- Low-power CMOS

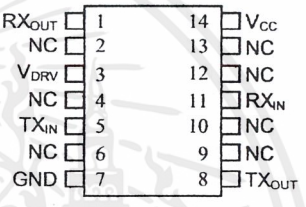
ORDERING INFORMATION

DS275	8-pin DIP
DS275S	8-pin SOIC
DS275E	14-pin TSSOP

PIN ASSIGNMENT



DS275 8-Pin DIP (300-mil)
DS275 8-Pin SOIC (150-mil)



DS275E 14-Pin TSSOP

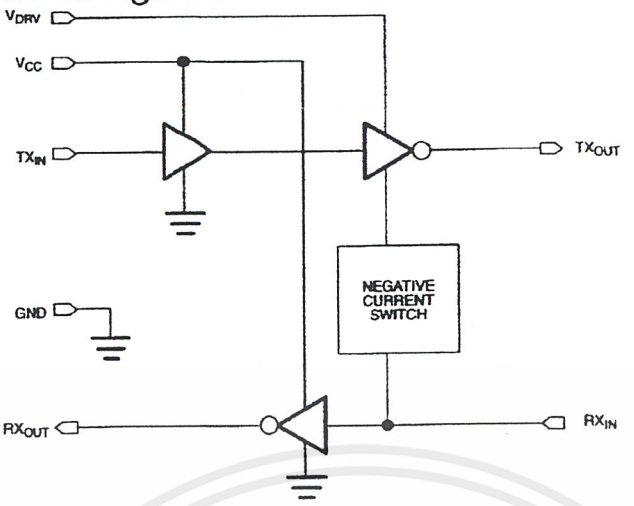
PIN DESCRIPTION

RX_{OUT}	- RS-232 Receiver Output
V_{DRV}	- Transmit driver +V
TX_{IN}	- RS-232 Driver Input
GND	- System Ground (0V)
TX_{OUT}	- RS-232 Driver Output
NC	- No Connection
RX_{IN}	- RS-232 Receive Input
V_{CC}	-System Logic Supply (+5V)

DESCRIPTION

The DS275 Line-Powered RS-232 Transceiver Chip is a CMOS device that provides a low-cost, very low-power interface to RS-232 serial ports. The receiver input translates RS-232 signal levels to common CMOS/TTL levels. The transmitter employs a unique circuit which steals current from the receive RS-232 signal when that signal is in a negative state (marking). Since most serial communication ports remain in a negative state statically, using the receive signal for negative power greatly reduces the DS275's static power consumption. This feature is especially important for battery-powered systems such as laptop computers, remote sensors, and portable medical instruments. During an actual communication session, the DS275's transmitter will use system power (5-12 volts) for positive transitions while still employing the receive signal for negative transitions.

DS275 BLOCK DIAGRAM Figure 1



OPERATION

Designed for the unique requirements of battery-backed systems, the DS275 provides a low-power half-duplex interface to an RS-232 serial port. Typically, a designer must use an RS-232 device which uses system power during both negative and positive transitions of the transmit signal to the RS-232 port. If the connector to the RS-232 port is left connected for an appreciable time after the communication session has ended, power will statically flow into that port, draining the battery capacity. The DS275 eliminates this static current drain by stealing current from the receive line (RXIN) of the RS-232 port when that line is at a negative level (marking). Since most asynchronous communication over an RS-232 connection typically remains in a marking state when data is not being sent, the DS275 will not consume system power in this condition. System power would only be used when positive-going transitions are needed on the transmit RS-232 output (TXOUT) when data is sent. However, since synchronous communication sessions typically exhibit a very low duty-cycle, overall system power consumption remains low.

RECEIVER SECTION

The RXIN pin is the receive input for an RS-232 signal whose levels can range from ±3 to ±15 volts. A negative data signal is called a mark while a positive data signal is called a space. These signals are inverted and then level-shifted to normal +5-volt CMOS/TTL logic levels. The logic output associated with RXIN is RXOUT which swings from +VCC to ground. Therefore, a mark on RXIN produces a logic 1 at RXOUT; a space produces a logic 0.

The input threshold of RXIN is typically around 1.8 volts with 500 millivolts of hysteresis to improve noise rejection. Therefore, an input positive-going signal must exceed 1.8 volts to cause RXOUT to switch states. A negative-going signal must now be lower than 1.3 volts (typically) to cause RXOUT to switch again. An open on RXIN is interpreted as a mark, producing a logic 1 at RXOUT.

TRANSMITTER SECTION

TXIN is the CMOS/TTL-compatible input for digital data from the user system. A logic 1 at TXIN produces a mark (negative data signal) at TXOUT while a logic 0 produces a space (positive data signal). As mentioned earlier, the transmitter section employs a unique driver design that uses the RXIN line for swinging to negative levels. The RXIN line must be in a marking or idle state to take advantage of this design; if RXIN is in a spacing state, TXOUT will only swing to ground. When TXOUT needs to transition to a positive level, it uses the VDRV power pin for this level. VDRV can be a voltage supply between 5 to 12

volts, and in many situations it can be tied directly to the +5 volt VCC supply. *It is important to note that VDRV must be greater than or equal to VCC at all times.*

The voltage range on VDRV permits the use of a 9-volt battery in order to provide a higher voltage level when TXOUT is in a space state. When VCC is shut off to the DS275 and VDRV is still powered (as might happen in a battery-backed condition), only a small leakage current (about 50-100 nA) will be drawn. If TXOUT is loaded during such a condition, VDRV will draw current only if RXIN is not in a negative state. During normal operation (VCC=5 volts), VDRV will draw less than 2 uA when TXOUT is marking. Of course, when TXOUT is spacing, VDRV will draw substantially more current— about 3 mA, depending upon its voltage and the impedance that TXOUT sees.

The TXOUT output is slew rate-limited to less than 30 volts/us in accordance with RS-232 specifications. In the event TXOUT should be inadvertently shorted to ground, internal current-limiting circuitry prevents damage, even if continuously shorted.

RS-232 COMPATIBILITY

The intent of the DS275 is not so much to meet all the requirements of the RS-232 specification as to offer a low-power solution that will work with most RS-232 ports with a connector length of less than 10 feet. As a prime example, the DS275 will not meet the RS-232 requirement that the signal levels be at least ±5 volts minimum when terminated by a 3 kΩ load and VDRV = +5 volts. Typically a voltage of 4 volts will be present at TXOUT when spacing. However, since most RS-232 receivers will correctly interpret any voltage over 2 volts as a space, there will be no problem transmitting data.

APPLICATIONS INFORMATION

The DS275 is designed as a low-cost, RS-232-E interface expressly tailored for the unique requirements of battery-operated handheld products. As shown in the electrical specifications, the DS275 draws exceptionally low operating and static current. During normal operation when data from the handheld system is sent from the TXOUT output, the DS275 only draws significant VDRV current when TXOUT transitions positively (spacing). This current flows primarily into the RS-232 receiver's 3-7 kΩ load at the other end of the attaching cable. When TXOUT is marking (a negative data signal), the VDRV current falls dramatically since the negative voltage is provided by the transmit signal from the other end of the cable. This represents a large reduction in overall operating current, since typical RS-232 interface chips use charge-pump circuits to establish both positive and negative levels at the transmit driver output.

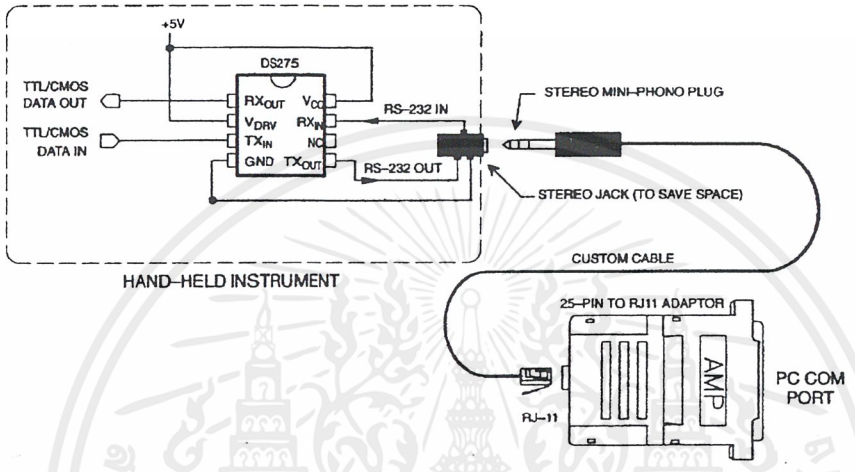
To obtain the lowest power consumption from the DS275, observe the following guidelines. First, to minimize VDRV current when connected to an RS-232 port, always maintain TXIN at a logic 1 when data is not being transmitted (idle state). This will force TXOUT into the marking state, minimizing VDRV current. Second, VDRV current will drop to less than 100 nA when VCC is grounded. Therefore, if VDRV is tied directly to the system battery, the logic +5 volts can be turned off to achieve the lowest possible power state.

FULL-DUPLEX OPERATION

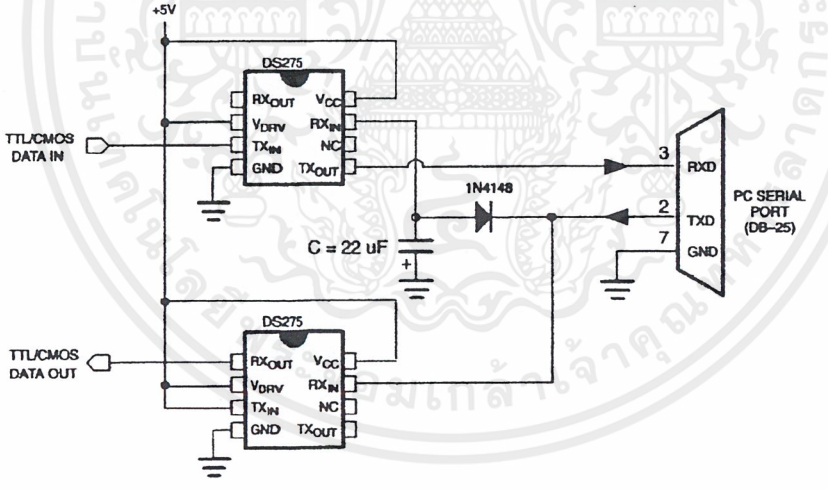
The DS275 is intended primarily for half-duplex operation; that is, RXIN should remain idle in the marking state when transmitting data out TXOUT and visa versa. However, the part can be operated full-duplex with most RS-232-E serial ports since signals swinging between 0 and +5V will usually be correctly interpreted by an RS-232-E receiver device. The 5-volt swing occurs when TXOUT attempts to swing negative while RXIN is at a positive voltage, which turns on an internal weak pulldown to ground for the TXOUT driver's negative reference. So, transmit mark signals at TXOUT may have voltage jumps from some negative value (corresponding to RXIN marking) to approximately ground. One possible

problem that may occur in this case is if the receiver at the other end requires a negative voltage for recognizing a mark. In this situation, the full-duplex circuit shown in Figure 3 can be used as analternative. The 22 µF capacitor forms a negative-charge reservoir; consequently, when the TXD line is spacing (positive), TXOUT still has a negative source available for a time period determined by the capacitor and the load resistance at the other end (3-7 kΩ). This circuit was tested from 150-19,200 bps with error-free operation using a SN75154 Quad Line Receiver as the receiver for the TXOUT signal. Note that the SN75154 can have a marking input threshold below ground; hence there is the need for TXOUT to swing both positive and negative in full-duplex operation with this device.

HANDHELD RS-232-C APPLICATION USING A STEREO MINI-JACK Figure 2



FULL-DUPLEX CIRCUIT USING NEGATIVE-CHARGE STORAGE Figure 3



NOTE:

The capacitor stores negative charge whenever the TXD signal from the PC serial port is in a marking data state (a negative voltage that is typically -10 volts). The top DS275's TXOUT uses this negative charge reservoir when it is in a marking state. The capacitor will discharge to 0 volts when the TXD line is spacing (and TXOUT is still marking) at a time constant determined by its value and the value of the load resistance reflected back to TXOUT. However, when TXD is marking the capacitor will quickly charge back to -10 volts. Note that TXD remains in a marking state when idle, which improves the performance of this circuit.

ABSOLUTE MAXIMUM RATINGS*

VCC	-0.3 to +7.0 volts
VDRV	-0.3 to +13.0 volts

RX_{IN}	±15 volts
TX_{IN}	-0.3 to V_{CC} + 0.3 volts
TX_{OUT}	±15 volts
RX_{OUT}	-0.3 to V_{CC} + 0.3 volts
Storage Temperature	-55°C to +125°C
Operating Temperature	0°C to 70°C

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

RECOMMENDED DC OPERATING CONDITIONS

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Logic Supply	V_{CC}	4.5	5.0	5.5	V	1
Transmit Driver Supply	V_{DRV}	4.5	5-12	13.0	V	1
Logic 1 Input	V_{IH}	2.0		V_{CC}+0.3	V	2
Logic 0 Input	V_{IL}	-0.3		+0.8	V	
RS-232 Input Range (RX_{IN})	V_{RS}	-15		+15	V	
Dynamic Supply Current						
$TX_{IN} = V_{CC}$	I_{DRV1}		400	800	µA	
	I_{CC1}		40	100	µA	3
TX_{IN} = GND	I_{DRV1}		3.8	5.0	µA	
	I_{CC1}		40	100	µA	
Static Supply Current						
$TX_{IN} = V_{CC}$	I_{DRV2}		1.5	10.0	µA	
	I_{CC2}		10.0	15.0	µA	4
TX_{IN} = GND	I_{DRV2}		3.8	5.0	mA	
	I_{CC2}		10.0	20.0	µA	
Driver Leakage Current (V_{CC}=0V)	I_{DRV3}		0.05	1.0	µA	5

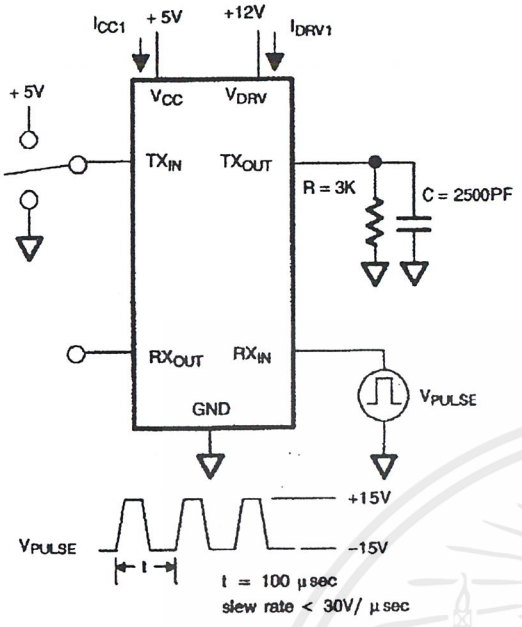
DC ELECTRICAL CHARACTERISTICS (0°C to 70°C; V_{CC} = V_{DRV} = 5V ± 10%)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
TX_{OUT} Level High	V_{OTXH}	3.5	4.0	5.0	V	6
TX_{OUT} Level Low	V_{OTXL}	-8.5	-9.0		V	7
TX_{OUT} Short Circuit Current	I_{SC}		+60	+85	mA	
TX_{OUT} Output Slew Rate	t_{SR}			30	V/µs	
Propagation Delay	t_{PD}		5		µs	8
RX_{IN} Input Threshold Low	V_{TL}	0.8	1.2	1.6	V	
RX_{IN} Input Threshold High	V_{TH}	1.6	2.0	2.4	V	
RX_{IN} Threshold Hysteresis	V_{HYS}	0.5	0.8		V	9
RX_{OUT} Output Current @ 2.4V	I_{OH}	-1.0			mA	
RX_{OUT} Output Current @ 0.4V	I_{OL}			3.2	mA	

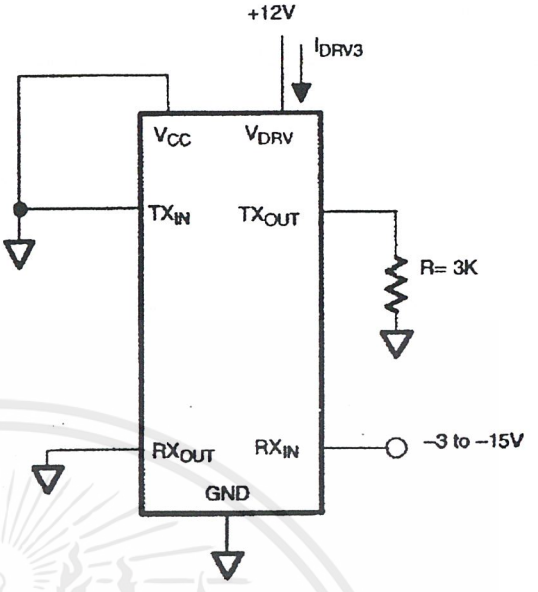
NOTES:

1. V_{DRV} must be greater than or equal to V_{CC}.
2. V_{CC} = V_{DRV} = 5V ± 10%.
3. See test circuit in Figure 4.
4. See test circuit in Figure 5.
5. See test circuit in Figure 6.
6. TX_{IN} = V_{IL} and TX_{OUT} loaded by 3 kΩ to ground.
7. TX_{IN} = V_{IH}, RX_{IN} = -10 volts and TX_{OUT} loaded by 3 kΩ to ground.
8. TX_{IN} to TX_{OUT} - see Figure 7.
9. V_{HYS} = V_{TH} - V_{TL}.

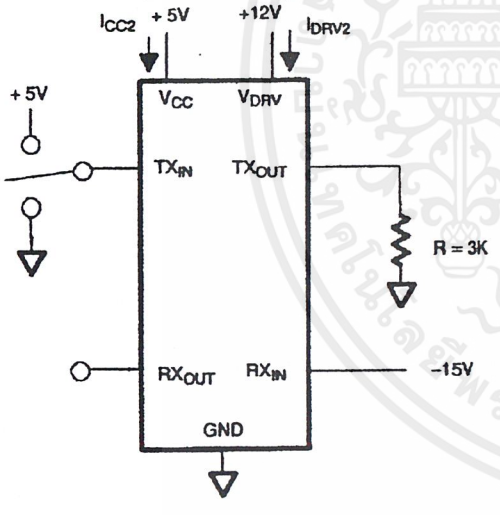
DYNAMIC OPERATING CURRENT TEST CIRCUIT Figure 4



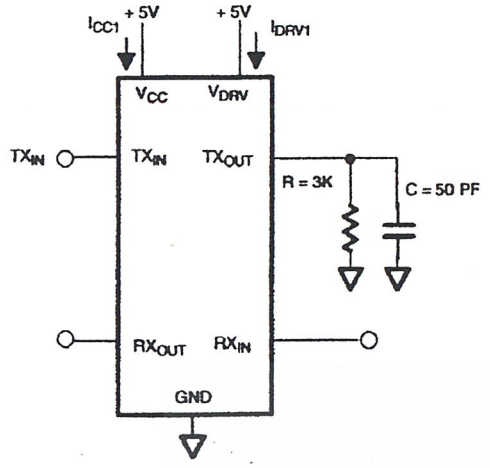
DRIVER LEAKAGE TEST CIRCUIT Figure 6

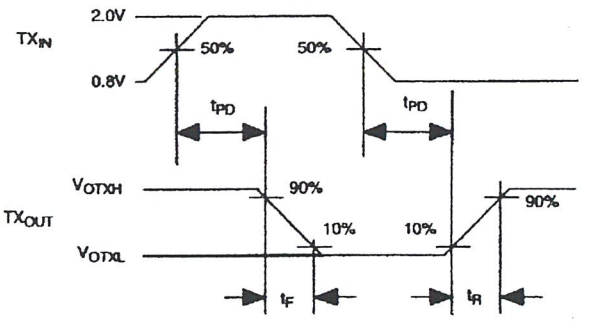


STATIC OPERATING CURRENT TEST CIRCUIT Figure 5



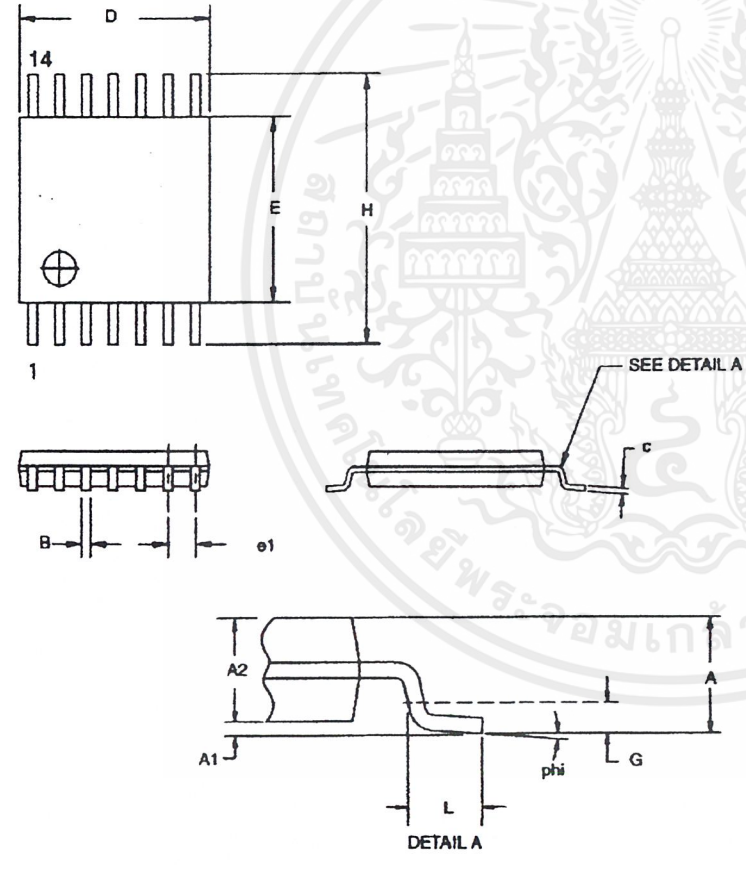
PROPAGATION DELAY TEST CIRCUIT Figure 7





$$t_{SR} = \frac{0.8(V_{OTXH} - V_{OTXL})}{t_F \text{ or } t_R}$$

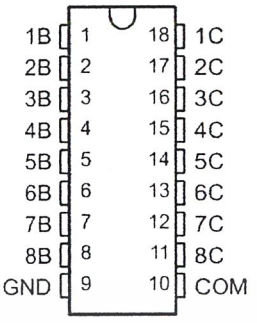
DS275E 14-PIN TSSOP



DIM	14-PIN	
	MIN	MAX
A MM	-	1.10
A1 MM	0.05	-
A2 MM	0.75	1.05
B MM	0.18	0.30
C MM	0.09	0.18
D MM	4.90	5.10
E MM	4.40 NOM	
e1 MM	0.65 BSC	
G MM	0.25 REF	
H MM	6.25	6.55
L MM	0.50	0.70
phi	0°	8°

- **500 mA Rated Collector Current (Single Output)**
- **High-Voltage Outputs . . . 50 V**
- **Output Clamp Diodes**
- **Inputs Compatible With Various Types of Logic**
- **Relay Driver Applications**
- **Compatible with ULN2800A Series**

N PACKAGE
(TOP VIEW)



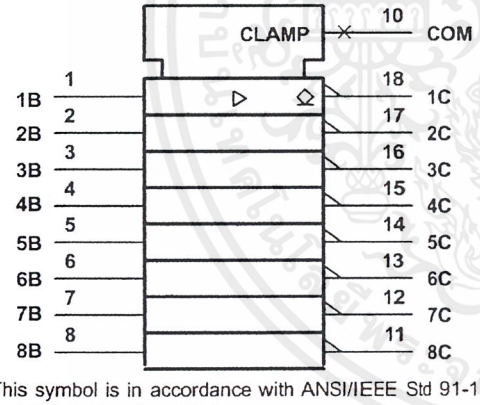
description

The ULN2803A is a monolithic high-voltage, high-current Darlington transistor array. The device consists of eight npn Darlington pairs that feature high-voltage outputs with common-cathode clamp diodes for switching inductive loads. The collector-current rating of each Darlington pair is 500 mA. The Darlington pairs may be paralleled for higher current capability.

Applications include relay drivers, hammer drivers, lamp drivers, display drivers (LED and gas discharge), line drivers, and logic buffers. The ULN2803A has a 2.7-kΩ series base resistor for each Darlington pair for operation directly with TTL or 5-V CMOS devices.

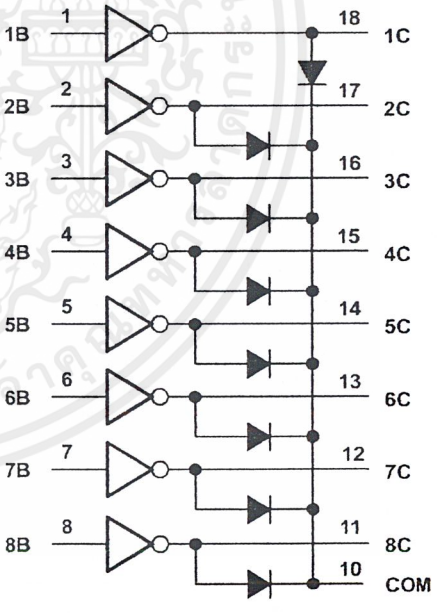
The ULN2803A is offered in a standard 18-pin dual in-line (N) package. The device is characterized for operation over the temperature range of –20°C to 85°C.

logic symbol†

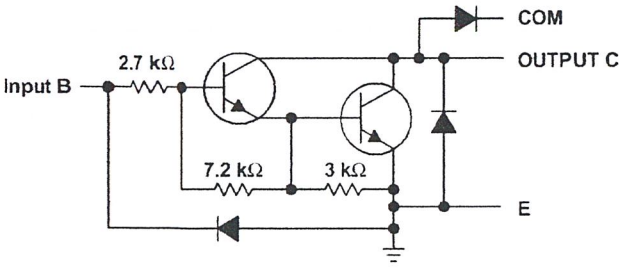


† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

logic diagram (positive logic)



schematic (each Darlington pair)



absolute maximum ratings at 25°C free-air temperature (unless otherwise noted)†

Collector-emitter voltage	50 V
Input voltage (see Note 1)	30 V
Continuous collector current	500 mA
Output clamp diode current	500 mA
Total substrate-terminal current	–2.5 A
Continuous dissipation at (or below) 25°C free-air temperature	1150 mW
Operating free-air temperature range, T_A	–20°C to 85°C
Storage temperature range, T_{stg}	–65°C to 150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds:	260°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltages values, unless otherwise noted, are with respect to the emitter/substrate terminal GND.

electrical characteristics at 25°C free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
I_{CEX}	Collector cutoff current	V_{CE} = 50 V, See Figure 1	I_I = 0,			50	µA
$I_{I(off)}$	Off-state input current	V_{CE} = 50 V, T_A = 70°C,	I_C = 500 µA, See Figure 2	50	65		µA
$I_{I(on)}$	Input current	V_I = 3.85 V,	See Figure 3		0.93	1.35	mA
$V_{I(on)}$	On-state input voltage	V_{CE} = 2 V, See Figure 4	I_C = 200 mA			2.4	V
			I_C = 250 mA			2.7	
			I_C = 300 mA			3	
$V_{CE(sat)}$	Collector emitter saturation voltage	I_I = 250 µA, See Figure 5	I_C = 100 mA,		0.9	1.1	V
		I_I = 350 µA, See Figure 5	I_C = 200 mA,		1	1.3	
		I_I = 500 µA, See Figure 5	I_C = 350 mA,		1.3	1.6	
I_R	Clamp diode reverse current	V_R = 50 V,	See Figure 6			50	µA
V_F	Clamp diode forward voltage	I_F = 350 mA,	See Figure 7		1.7	2	V
C_i	Input capacitance	V_I = 0 V,	f = 1 MHz		15	25	pF

switching characteristics at 25°C free-air temperature

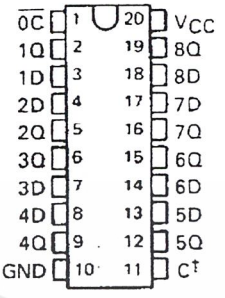
PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output	V_S = 50 V, C_L = 15 pF,	R_L = 163 Ω, See Figure 8		130		ns
t_{PHL}	Propagation delay time, high-to-low level output				20		
V_{OH}	High-level output voltage after switching	V_S = 50 V, See Figure 9	I_O = 300 mA,	V_S–20			mV

SN54LS373, SN54LS374, SN54S373, SN54S374, SN74LS373, SN74LS374, SN74S373, SN74S374
OCTAL D-TYPE TRANSPARENT LATCHES AND EDGE-TRIGGERED FLIP-FLOPS

OCTOBER 1975—REVISED MARCH 1988

- Choice of 8 Latches or 8 D-Type Flip-Flops In a Single Package
- 3-State Bus-Driving Outputs
- Full Parallel-Access for Loading
- Buffered Control Inputs
- Clock/Enable Input Has Hysteresis to Improve Noise Rejection ('S373 and 'S374)
- P-N-P Inputs Reduce D-C Loading on Data Lines ('S373 and 'S374)

SN54LS373, SN54LS374, SN54S373, SN54S374 . . . J OR W PACKAGE
SN74LS373, SN74LS374, SN74S373, SN74S374 . . . DW OR N PACKAGE
(TOP VIEW)



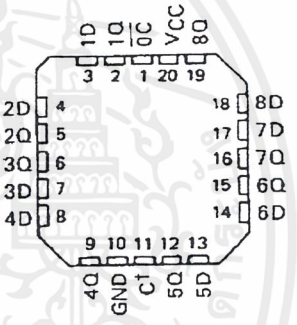
'LS373, 'S373
FUNCTION TABLE

OUTPUT ENABLE	ENABLE LATCH	D	OUTPUT
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

'LS374, 'S374
FUNCTION TABLE

OUTPUT ENABLE	CLOCK	D	OUTPUT
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

SN54LS373, SN54LS374, SN54S373, SN54S374 . . . FK PACKAGE
(TOP VIEW)



†C for 'LS373 and 'S373; CLK for 'LS374 and 'S374.

description

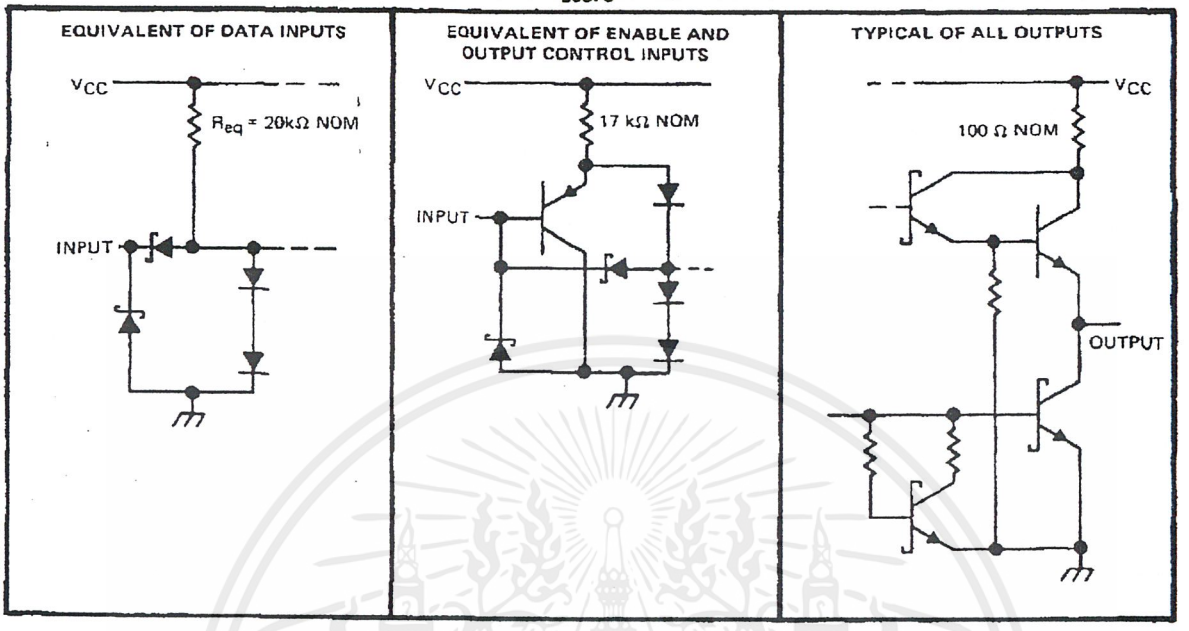
These 8-bit registers feature three-state outputs designed specifically for driving highly-capacitive or relatively low-impedance loads. The high-impedance third state and increased high-logic-level drive provide these registers with the capability of being connected directly to and driving the bus lines in a bus-organized system without need for interface or pull-up components. They are particularly attractive for implementing buffer registers, I/O ports, bidirectional bus drivers, and working registers.

The eight latches of the 'LS373 and 'S373 are transparent D-type latches meaning that while the enable (C) is high the Q outputs will follow the data (D) inputs. When the enable is taken low the output will be latched at the level of the data that was set up.

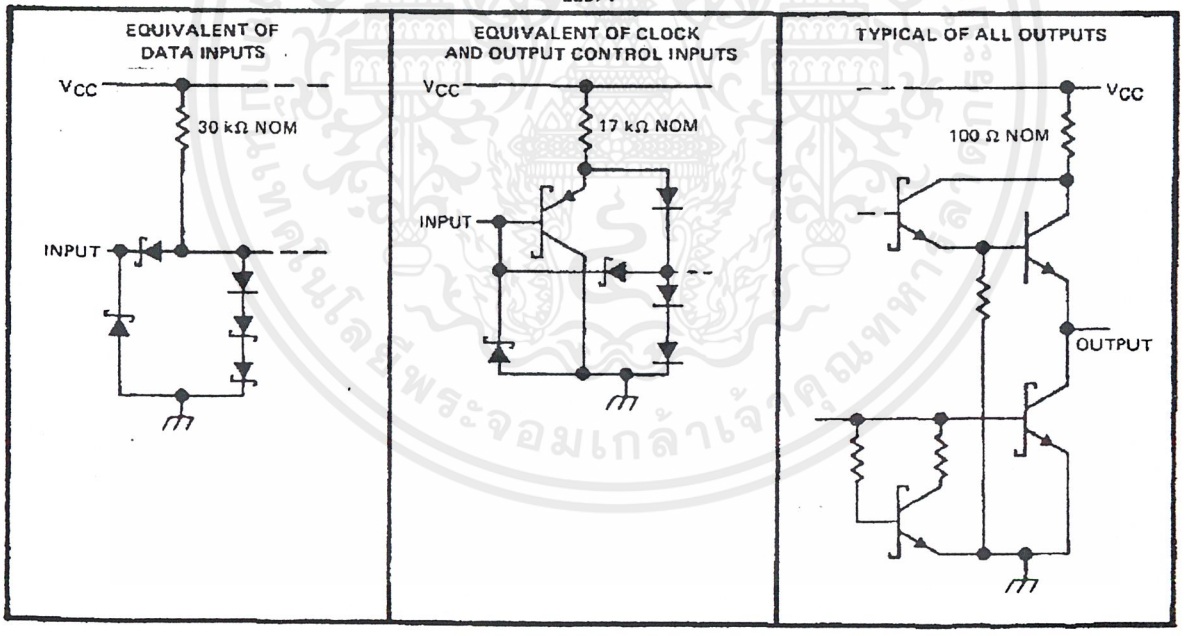
TEXAS INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

schematic of inputs and outputs

'LS373



'LS374



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage, V_{CC} (see Note 1) . . . . . 7 V
Input voltage . . . . . 7 V
Off-state output voltage . . . . . 5.5 V
Operating free-air temperature range: SN54LS' . . . . . −55°C to 125°C
SN74LS' . . . . . 0°C to 70°C
Storage temperature range . . . . . −65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

recommended operating conditions

			SN54LS'			SN74LS'			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
V_{OH}	High-level output voltage				5.5			5.5	V
I_{OH}	High-level output current				−1†			−2.6	mA
I_{OL}	Low-level output current				12			24	mA
t_w	Pulse duration	CLK high	15			15			ns
		CLK low	15			15			
t_{su}	Data setup time	'LS373	5↓			5↓			ns
		'LS374	20↑			20↑			
t_h	Data hold time	'LS373	20↓			20↓			ns
		'LS374†	5↑			0↑			
T_A	Operating free-air temperature		−55		125	0		70	°C

†The t_h specification applies only for data frequency below 10 MHz. Designs above 10 MHz should use a minimum of 5 ns. (Commercial only)

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS†		SN54LS'			SN74LS'			UNIT
				MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage			2			2			V
V_{IL}	Low-level input voltage					0.7			0.8	V
V_{IK}	Input clamp voltage	V_{CC} = MIN, I_I = −18 mA				−1.5			−1.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL}max, I_{OH} = MAX		2.4	3.4		2.4	3.1		V
V_{OL}	Low-level output voltage	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL}max	I_{OL} = 12 mA		0.25	0.4		0.25	0.4	V
			I_{OL} = 24 mA					0.35	0.5	
I_{OZH}	Off-state output current, high-level voltage applied	V_{CC} = MAX, V_{IH} = 2 V, V_O = 2.7 V				20			20	μA
I_{OZL}	Off-state output current, low-level voltage applied	V_{CC} = MAX, V_{IH} = 2 V, V_O = 0.4 V				−20			−20	μA
I_I	Input current at maximum input voltage	V_{CC} = MAX, V_I = 7 V				0.1			0.1	mA
I_{IH}	High-level input current	V_{CC} = MAX, V_I = 2.7 V				20			20	μA
I_{IL}	Low-level input current	V_{CC} = MAX, V_I = 0.4 V				−0.4			−0.4	mA
I_{OS}	Short-circuit output current§	V_{CC} = MAX		−30		−130	−30		−130	mA
I_{CC}	Supply current	V_{CC} = MAX, Output control at 4.5 V	'LS373		24	40		24	40	mA
			'LS374		27	40		27	40	

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.
§ Not more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	'LS373			'LS374			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
f_{max}			C_L = 45 pF, R_L = 667 Ω, See Notes 2 and 3				35	50		MHz
t_{PLH}	Data	Any Q			12	18				ns
t_{PHL}					12	18				
t_{PLH}	Clock or enable	Any Q			20	30		15	28	ns
t_{PHL}					18	30		19	28	
t_{PZH}	Output Control	Any Q			15	28		20	26	ns
t_{PZL}					25	36		21	28	
t_{PHZ}	Output Control	Any Q	C_L = 5 pF, R_L = 667 Ω, See Note 3		15	25		15	28	ns
t_{PLZ}	Output Control	Any Q			12	20		12	20	ns

NOTES: 2. Maximum clock frequency is tested with all outputs loaded.
3. Load circuits and voltage waveforms are shown in Section 1.

f_{max} = maximum clock frequency
t_{PLH} = propagation delay time, low-to-high-level output
t_{PHL} = propagation delay time, high-to-low-level output
t_{PZH} = output enable time to high level
t_{PZL} = output enable time to low level
t_{PHZ} = output disable time from high level
t_{PLZ} = output disable time from low level

MOTOROLA

DUAL 1-OF-4 DECODER/ DEMULTIPLEXER

SN54/74LS155
SN54/74LS156

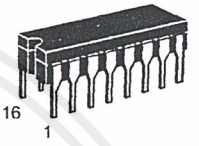
DUAL 1-OF-4 DECODER/ DEMULTIPLEXER

LS156-OPEN-COLLECTOR LOW POWER SCHOTTKY

The SN54/74LS155 and SN54/74LS156 are high speed Dual 1-of-4 Decoder/Demultiplexers. These devices have two decoders with common 2-bit Address inputs and separate gated Enable inputs. Decoder "a" has an Enable gate with one active HIGH and one active LOW input. Decoder "b" has two active LOW Enable inputs. If the Enable functions are satisfied, one output of each decoder will be LOW as selected by the address inputs. The LS156 has open collector outputs for wired-OR (DOT-AND) decoding and function generator applications.

The LS155 and LS156 are fabricated with the Schottky barrier diode process for high speed and are completely compatible with all Motorola TTL families.

- Schottky Process for High Speed
- Multifunction Capability
- Common Address Inputs
- True or Complement Data Demultiplexing
- Input Clamp Diodes Limit High Speed Termination Effects
- ESD > 3500 Volts



J SUFFIX
CERAMIC
CASE 620-09

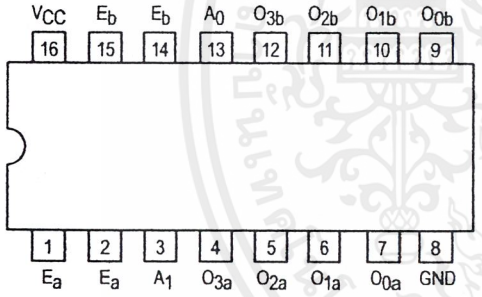
N SUFFIX
PLASTIC
CASE 648-08

D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

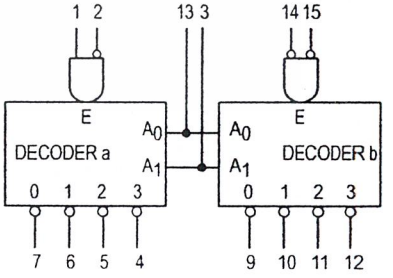
SN54LSXXXJ	Ceramic
SN74LSXXXN	Plastic
SN74LSXXXD	SOIC

CONNECTION DIAGRAM DIP (TOP VIEW)



NOTE:
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

LOGIC SYMBOL



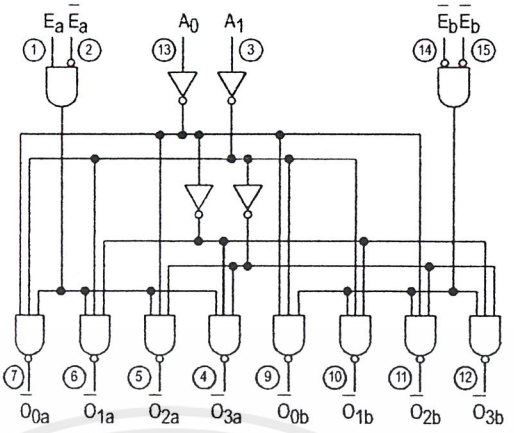
VCC = PIN 16
GND = PIN 8

PIN NAMES

		LOADING (Note a) HIGH	LOW
$\underline{A}_0$, $\underline{A}_1$	Address Inputs	0.5 U.L.	0.25 U.L.
E_a, E_b	Enable (Active LOW) Inputs	0.5 U.L.	0.25 U.L.
$\underline{E}_a$	Enable (Active HIGH) Input	0.5 U.L.	0.25 U.L.
$O_0 - O_3$	Active LOW Outputs (Note b)	10 U.L.	5 (2.5) U.L.

NOTES:
a) 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b) The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges. The HIGH level drive for the LS156 must be established by an external resistor.

LOGIC DIAGRAM



V_{CC} = PIN 16
GND = PIN 8
◯ = PIN NUMBERS

FUNCTIONAL DESCRIPTION

The LS155 and LS156 are Dual 1-of-4 Decoder/Demultiplexers with common Address inputs and separate gated Enable inputs. When enabled, each decoder section accepts the binary weighted Address inputs (A_0,A_1) and provides four mutually exclusive active LOW outputs (O_0–O_3). If the Enable requirements of each decoder are not met, all outputs of that decoder are HIGH.

Each decoder section has a 2-input enable gate. The enable gate for Decoder "a" requires one active HIGH input and one active LOW input (E_a•$\overline{E}_a$). In demultiplexing applications, Decoder "a" can accept either true or complemented data by using the E_a or $\overline{E}_a$ inputs respectively. The enable gate for Decoder "b" requires two active LOW inputs ($\overline{E}_b$•$\overline{E}_b$). The LS155 or LS156 can be used as a 1-of-8 Decoder/Demultiplexer by tying E_a to $\overline{E}_b$ and relabeling the common connection as (A_2). The other $\overline{E}_b$ and $\overline{E}_a$ are connected together to form the common enable.

The LS155 and LS156 can be used to generate all four minterms of two variables. These four minterms are useful in some applications replacing multiple gate functions as shown in Fig. a. The LS156 has the further advantage of being able to AND the minterm functions by tying outputs together. Any number of terms can be wired-AND as shown below.

$$f = (E + A_0 + A_1) \cdot (E + \overline{A}_0 + A_1) \cdot (E + A_0 + \overline{A}_1) \cdot (E + A_0 + A_1)$$

where $E = E_a + \overline{E}_a$; $E = E_b + E_b$

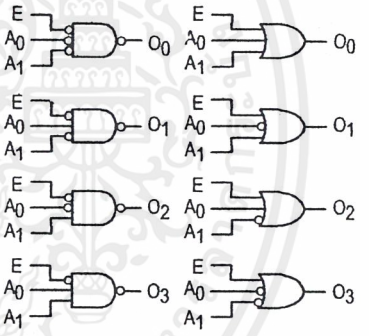


Figure a

TRUTH TABLE

ADDRESS		ENABLE "a"		OUTPUT "a"				ENABLE "b"		OUTPUT "b"			
A_0	A_1	E_a	$\overline{E}_a$	O_0	O_1	O_2	O_3	E_b	$\overline{E}_b$	O_0	O_1	O_2	O_3
X	X	L	X	H	H	H	H	H	X	H	H	H	H
X	X	X	H	H	H	H	H	X	H	H	H	H	H
L	L	H	L	L	H	H	H	L	L	L	H	H	H
H	L	H	L	H	L	H	H	L	L	H	L	H	H
L	H	H	L	H	H	L	H	L	L	H	H	L	H
H	H	H	L	H	H	H	L	L	L	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

PC817 Series

High Density Mounting Type Photocoupler

* Lead forming type (I type) and taping reel type (P type) are also available. (**PC817I/PC817P**)
** TÜV (VDE0884) approved type is also available as an option.

■ Features

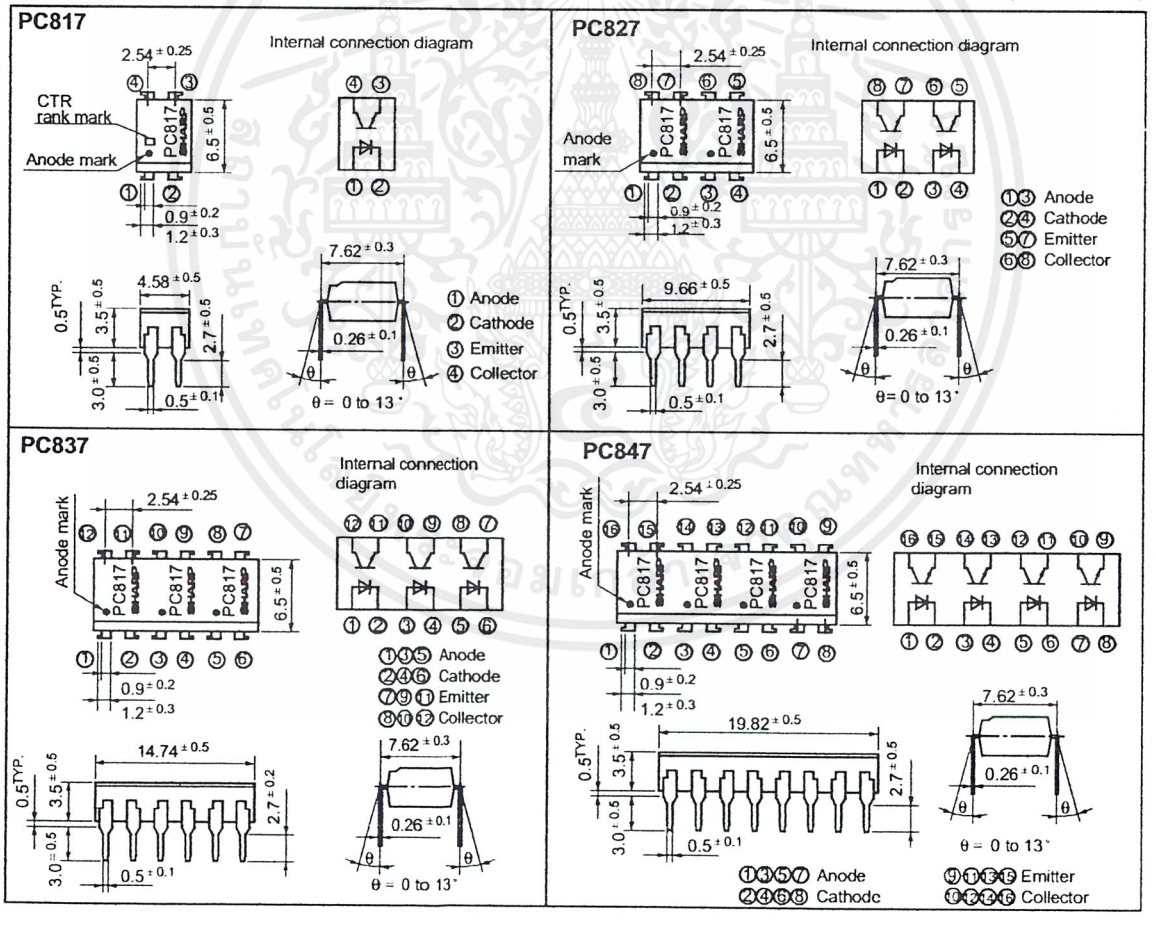
1. Current transfer ratio
   (CTR: MIN. 50% at I_F = 5mA , V_{CE}=5V)
2. High isolation voltage between input and output (V_{iso} : 5 000V $_{rms}$)
3. Compact dual-in-line package
   PC817 : 1-channel type
   PC827 : 2-channel type
   PC837 : 3-channel type
   PC847 : 4-channel type
4. Recognized by UL, file No. E64380

■ Applications

1. Computer terminals
2. System appliances, measuring instruments
3. Registers, copiers, automatic vending machines
4. Electric home appliances, such as fan heaters, etc.
5. Signal transmission between circuits of different potentials and impedances

■ Outline Dimensions

(Unit : mm)



■ Absolute Maximum Ratings (Ta= 25°C)

Parameter		Symbol	Rating	Unit
Input	Forward current	I_F	50	mA
	*1Peak forward current	I_{FM}	1	A
	Reverse voltage	V_R	6	V
	Power dissipation	P	70	mW
Output	Collector-emitter voltage	V_{CEO}	35	V
	Emitter-collector voltage	V_{ECO}	6	V
	Collector current	I_C	50	mA
	Collector power dissipation	P_C	150	mW
Total power dissipation		P_{tot}	200	mW
*2Isolation voltage		V_{iso}	5 000	V_{rms}
Operating temperature		T_{opr}	- 30 to + 100	°C
Storage temperature		T_{stg}	- 55 to + 125	°C
*3Soldering temperature		T_{sol}	260	°C

*1 Pulse width <=100μs, Duty ratio : 0.001
*2 40 to 60% RH, AC for 1 minute
*3 For 10 seconds

■ Electro-optical Characteristics (Ta= 25°C)

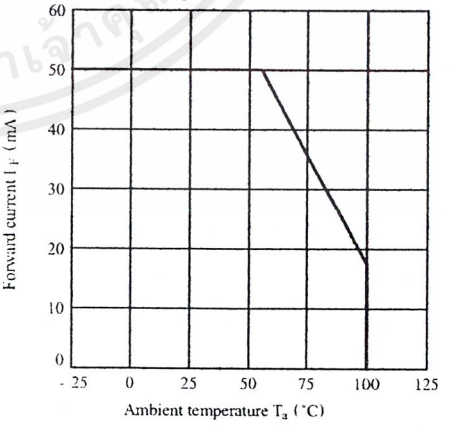
Parameter			Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input	Forward voltage		V_F	I_F = 20mA	-	1.2	1.4	V
	Peak forward voltage		V_{FM}	I_{FM} = 0.5A	-	-	3.0	V
	Reverse current		I_R	V_R = 4V	-	-	10	μA
	Terminal capacitance		C_t	V = 0, f = 1kHz	-	30	250	pF
Output	Collector dark current		I_{CEO}	V_{CE} = 20V	-	-	10^{-7}	A
Transfer characteristics	*4Current transfer ratio		CTR	I_F = 5mA, V_{CE} = 5V	50	-	600	%
	Collector-emitter saturation voltage		$V_{CE(sat)}$	I_F = 20mA, I_C = 1mA	-	0.1	0.2	V
	Isolation resistance		R_{ISO}	DC500V, 40 to 60% RH	5×10^{10}	10^{11}	-	Ω
	Floating capacitance		C_f	V = 0, f = 1MHz	-	0.6	1.0	pF
	Cut-off frequency		f_c	V_{CE} = 5V, I_C = 2mA, R_L = 100Ω, - 3dB	-	80	-	kHz
	Response time	Rise time	t_r	V_{CE} = 2V, I_C = 2mA, R_L = 100Ω	-	4	18	μs
		Fall time	t_f		-	3	18	μs

*4 Classification table of current transfer ratio is shown below.

Model No.	Rank mark	CTR (%)
PC817A	A	80 to 160
PC817B	B	130 to 260
PC817C	C	200 to 400
PC817D	D	300 to 600
PC8＊7AB	A or B	80 to 260
PC8＊7BC	B or C	130 to 400
PC8＊7CD	C or D	200 to 600
PC8＊7AC	A, B or C	80 to 400
PC8＊7BD	B, C or D	130 to 600
PC8＊7AD	A, B, C or D	80 to 600
PC8＊7	A, B, C, D or No mark	50 to 600

＊: 1 or 2 or 3 or 4

Fig. 1 Forward Current vs. Ambient Temperature



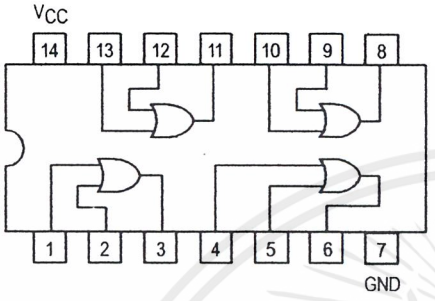
MOTOROLA

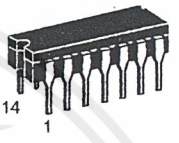
QUAD 2-INPUT OR GATE

SN54/74LS32

QUAD 2-INPUT OR GATE

LOW POWER SCHOTTKY





J SUFFIX
CERAMIC
CASE 632-08

N SUFFIX
PLASTIC
CASE 646-06

D SUFFIX
SOIC
CASE 751A-02

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T_A	Operating Ambient Temperature Range	54 74	−55 0	25 25	125 70	°C
I_{OH}	Output Current — High	54, 74			−0.4	mA
I_{OL}	Output Current — Low	54 74			4.0 8.0	mA

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter		Limits Min	Limits Typ	Limits Max	Unit	Test Conditions	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table	
		74	2.7	3.5		V		
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA	
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Short Circuit Current (Note 1)		−20		−100	mA	V_{CC} = MAX	
I_{CC}	Power Supply Current Total, Output HIGH				6.2	mA	V_{CC} = MAX	
	Total, Output LOW				9.8			

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25°C)

Symbol	Parameter	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
t_{PLH}	Turn-Off Delay, Input to Output		14	22	ns	V_{CC} = 5.0 V, C_L = 15 pF
t_{PHL}	Turn-On Delay, Input to Output		14	22	ns	

กิตติกรรมประกาศ

แด่พ่อและแม่ที่คอยเป็นห่วงทั้งเรื่องการเรียนและอนาคตของลูก และเป็นผู้ให้โอกาสในการเลือกเส้นทางเดินชีวิตเอง และขอขอบคุณพี่อิ๊ดที่ท่านคอยแนะแนวทางให้

ขอขอบคุณอาจารย์ชินภัทร นันทจิวากรชัยที่ท่านคอยเป็นที่ปรึกษา คอยแนะนำและช่วยเหลือในการทำโครงงานนี้

ขอขอบใจเพื่อนๆนายวิสันต์ คัมภีร์บวรที่ให้คำปรึกษาด้านไมโครคอนโทรเลอร์ และนายวิน .... ที่ให้คำปรึกษาด้าน Visual Basic เป็นอย่างดี และเพื่อนๆทุกคนที่ไม่ได้เอ่ยนามมา ณ. ที่นี้ด้วยเป็นอย่างดี

บรรณานุกรม

1. กฤษดา วิศวธีรานนท์, สมบูรณ์ จงชัยกิจ, สุเธียร เกียรติสุนทร, " เอกสารการอบรม PROGRAMMABLE SEQUENCE CONTROLLER ", สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
2. พรจิต ประทุมสุวรรณ และคณะ, " ทฤษฎีและการใช้งาน (PC/PLC) PROGRAMMABLE CONTROLLER ", พ.ศ. 2536
3. ธาริน สิทธิธรรมชารี, " คู่มือการเขียนโปรแกรม Microsoft Visual Basic Version 6.0 ", พิมพ์ครั้งที่ 7, บริษัท ซัคเซส มีเดีย จำกัด
4. ไกรวุฒิ โรจน์ประเสริฐสุด, " เข้าใจ/สร้าง/เล่น ไมโครโปรเซสเซอร์ 2 ", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)