เครื่องควบคุมที่โปรแกรมได้

PROGRAMMABLE LOGIC CONTROLLER : PLC

โดย

นายพิชญ์ ศรีมหาวงษ์

นายสมโชค จารุรัตนเกื้อ

นายสุทิน ตันรัตนากร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2534

ปริญญานิพนธ์ปีการศึกษา 2534

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องควบคุมที่โปรแกรมได้
(PROGRAMMABLE LOGIC CONTROLLER : PLC)

ผู้จัดทำ

นายพิชญ์	ศรีมหาวงษ์	31.1176
นายสมโชค	จารุรัตนเกื้อ	31.1300
นายสุทิน	ตันรัตนากร	31.1344

.....อาจารย์ที่ปรึกษา
(อาจารย์สุพรรณ กุลพาณิชย์)

เครื่องควบคุมที่โปรแกรมได้

พิชญ์ ศรีมหาวงษ์
สมโชค จารุรัตนเกื้อ
สุทิน ตันรัตนากร

สุพรรณ กุลพาณิชย์
... อาจารย์ที่ปรึกษา

2534

บทคัดย่อ

"เครื่องควบคุมที่โปรแกรมได้" หรือที่รู้จักกันดีว่า "พีแอลซี" (PLC:programmable logic controller) เป็นเครื่องมือที่มีบทบาทและความสำคัญอย่างมากในกระบวนการต่าง ๆ ทางอุตสาหกรรม และเป็นที่แพร่หลายมากในวงการอุตสาหกรรมต่างประเทศ โดยลักษณะของเครื่องควบคุมในปัจจุบัน เน้นเรื่องการใช้งานให้ง่ายแต่มีความสามารถในการทำงานสูงขึ้น คือทำงานที่ซับซ้อนมากขึ้นได้ และง่ายต่อการเปลี่ยนแปลงแก้ไขเงื่อนไขการทำงานของกระบวนการเพียงแต่เปลี่ยนแปลงโปรแกรมเท่านั้นปริญญานิพนธ์ ฉบับนี้ ขอเสนอการออกแบบ พีแอลซี โดยใช้ไมโครโปรเซสเซอร์ ขนาด 8 บิท เบอร์ 8052 AH ในตระกูล Mcs-51 จำนวน 1 ตัวเป็นหน่วยประมวลผลกลาง ทำหน้าที่ควบคุมเงื่อนไขการทำงานทั้งหมดของกระบวนการรวมทั้งควบคุมหน่วยป้อนโปรแกรมในเครื่องเดียวกัน ข้อดีที่สำคัญคือ สามารถจะใช้ติดต่อกับอุปกรณ์ภายนอก จำนวนมาก เปลี่ยนแปลงลักษณะและเงื่อนไขการควบคุมได้โดยง่ายนอกจากนี้การประมวลผลยังกระทำได้อย่างถูกต้องและรวดเร็ว

PROGRAMMABLE LOGIC CONTROLLER

MR. PHIT SIMAHAWONG

MR. SOMCHOK JARURATTANAKUA

MR. SUTIN TANRATTANAKORN

MR. SUPARN KULAPANIT

... ADVISOR

1991

ABSTRACT

PROGRAMMABLE LOGIC CONTROLLER, WELL KNOWN AS "PLC", IS A SEQUENCE CONTROLLER, WHICH IS VERY IMPORTANT TO MANY INDUSTRIAL PROCESSING AND MUCH USED IN DEVELOPED COUNTRIES THE CHARACTERISTICS OF PRESENT-DAY SEQUENCE CONTROLLER , STRESS THE IMPORTANCE OF USING THIS DEVICE EASILY , AND ENHANCE THE EFFICIENCY OF ITS PERFORMANECES AT THE SAME TIME, IT CAN DO MORE COMPLICATED JOBS AND CAN BE PROGRAMMED. THIS THESIS PRESENTS A DESIGN OF PLC, USE A 8052 AH, 8 BIT MICROPROCESSOR AS THE CPU UNIT AND ALSO CONTROL THE PROGRAMMING UNIT TOO. THE ADVENTAGE OF THIS DESIGNED PLC LIES IN ITS ABILITY TO INTERFACE WITH A LARGE NUMBER OF I/O DEVICES, CHANGE THE CONTROL SEQUENCE EASILY , PROCESS TIME SWIFTLY.

สารบัญ

บทที่ 1
บทนำ

PLC (Programmable logic Controllers)

หรือเรียกอีกอย่างหนึ่งว่า เครื่องควบคุมที่สามารถโปรแกรมได้ เรียกทั่ว ๆ ไปว่า PLC เป็นอุปกรณ์ที่ถูกผลิตขึ้นปลายปี ค.ศ. 1960 เพื่อนำมาใช้งานแทนระบบควบคุมแบบเก่า โดยในตอนแรกเริ่มแรกนั้น PLC ถูกประดิษฐ์เพื่อใช้งานอย่างง่าย ๆ โดยการโปรแกรม ส่วนอินพุตและเอาต์พุตของ PLC จะมีลักษณะเป็นแบบดิจิตอล คือ มี ON กับ OFF สัญญาณ ON และ OFF เหล่านี้จะถูกนำไปใช้ควบคุมอุปกรณ์ภายนอก เช่น รีเลย์ หรือหน่วยแสดงผลอื่น ๆ ดังนั้น คำว่า "Programmable logic Controllers" จึงเป็นที่รู้จักอย่างกว้างขวาง

ปัจจุบันนี้ PLC ได้ถูกพัฒนาขึ้นมาอย่างรวดเร็ว โดยมีความสามารถสูงขึ้น สามารถทำให้งานควบคุมทั้งทางด้านอนาล็อก และดิจิตอล โดยที่ขนาดของ PLC มิได้ใหญ่โตขึ้น อุปกรณ์ส่วนใหญ่ ของ PLC เป็นอุปกรณ์พวกไมโครอิเล็กทรอนิกส์ต่าง ๆ โดยมีส่วนประมวลผลกลาง (CPU) เป็นไมโครโปรเซสเซอร์ควบคุมการทำงานภายในอาจจะเป็น IC ตระกูล 51 หรือของ Motorola ตระกูล 68 เป็นต้น

นอกจากนี้ยังมีการนำเอา PLC มาเชื่อมต่อกับเครื่องคอมพิวเตอร์ตลอดจนอุปกรณ์ควบคุมอื่น ๆ ทำให้ PLC มีความสามารถในการใช้งาน มากขึ้น (โดยสามารถทำ Self Diagnestic Self test ตลอดจน Communication Link ฯลฯ) ดังนั้น PLC จึงกลายเป็นหัวใจสำคัญสำหรับงานอุตสาหกรรมสมัยใหม่ ด้วยเหตุผลหลัก ๆ คือ

ประการแรก - การใช้งานสามารถทำได้โดยการเขียนโปรแกรมเป็นภาษาง่าย ๆ โดยที่ตัวโปรแกรมสามารถดัดแปลงแก้ไขได้ และถูกเก็บเอาไว้ใน PLC

ประการที่สอง - รูปร่างลักษณะภายนอก ได้ถูกออกแบบมาให้ใช้งานได้อย่างสะดวกเหมาะสมกับสภาพของโรงงานอุตสาหกรรมในปัจจุบัน

อย่างไรก็ตามแม้ว่าในแถบยุโรปและอเมริกา PLC จะเป็นที่นิยมอย่างสูงสุด แต่ประเทศไทย ค่อนข้างจะเป็นของใหม่ (ที่มีขายอยู่ก็ราคาแพง) การใช้งานก็ยังไม่กว้างขวางนัก การควบคุมกระบวนการทางอุตสาหกรรมส่วนใหญ่จะเป็นการนำอุปกรณ์ไฟฟ้าเชิงกล เช่น รีเลย์ (relay) ตัวนับ (counter) ตัวตั้งเวลา (timer) มาต่อร่วมกันเป็นระบบควบคุมแต่การใช้วิธีการดังกล่าวนี้มีข้อเสียมากเนื่องจากอุปกรณ์ไฟฟ้าเชิงกลจะสิ้นเปลืองกำลังงานสูงและมีอายุการใช้งานค่อนข้างสั้น นอกจากนั้นเมื่อต้องการเปลี่ยนแปลงลำดับการทำงานของกระบวนการก็จะต้องแก้ไขระบบควบคุมใหม่ทำให้ยุ่งยากและเสียเวลามาก หรือในกรณีที่มีการเปลี่ยนแปลงมาก ๆ อาจจะต้องออกแบบระบบควบคุมใหม่ทั้งหมด

การประยุกต์ใช้ไมโครโปรเซสเซอร์มาใช้ในการควบคุมกระบวนการเป็นวิธีหนึ่งที่มีประสิทธิภาพสูง เนื่องจากสามารถเปลี่ยนแปลงลำดับขั้นการทำงานหรือการควบคุมได้โดยการเปลี่ยนแปลงหรือแก้ไขโปรแกรม ดังนั้นในปัจจุบันจึงมีการใช้ไมโครโปรเซสเซอร์ หรือไมโครคอมพิวเตอร์ ในการควบคุมกระบวนการกันอย่างแพร่หลายอย่างไรก็ดีการแก้ไขหรือเปลี่ยนแปลงโปรแกรมนั้น จะต้องกระทำโดยผู้ที่เข้าใจโครงสร้างทางฮาร์ดแวร์ (Hardware) ของระบบควบคุมหรือกระบวนการเป็นอย่างดี ดังนั้นผู้ปฏิบัติงาน หรือผู้ใช้มักจะไม่สามารถแก้ไขหรือเปลี่ยนแปลงโปรแกรมดังกล่าวได้เอง ดังนั้นจึงได้มีการพัฒนาเครื่องควบคุมที่โปรแกรมได้ (Programmable Logic Controller) ขึ้น ซึ่งผู้ปฏิบัติงานหรือผู้ใช้สามารถที่จะเปลี่ยนแปลง หรือแก้ไขโปรแกรมการควบคุมได้เองโดยใช้ภาษาที่กำหนด เครื่องควบคุมที่โปรแกรมได้ส่วนใหญ่จะใช้ภาษาสำหรับทำการโปรแกรม เช่น ภาษาโปรแกรมแบบเลดเดอร์ไดอะแกรม โปรแกรมแบบบูลลีนหรือภาษาชั้นสูงต่าง ๆ

โครงงานฉบับนี้ ได้นำเสนอเครื่องควบคุมที่โปรแกรมได้ ซึ่งออกแบบเทคโนโลยีขึ้น จากความสามารถทางวิชาการไมโครโปรเซสเซอร์ โดยวางแนวทางให้มีความสามารถและคุณสมบัติใกล้เคียงกับ PLC จากต่างประเทศ สามารถป้อนโปรแกรมแบบบูลลีน โดยแก้ไข เปลี่ยนแปลงโปรแกรมทางคีย์บอร์ด ทั้งยังแสดงผลของกระบวนการทาง LCD และ LED ให้ทราบได้

บทที่ 2
ทฤษฎีและโครงสร้างของเครื่องควบคุมที่โปรแกรมได้

เครื่องควบคุมที่โปรแกรมได้ เป็นการประยุกต์ใช้เทคโนโลยีทางด้านไมโครโปรเซสเซอร์ทั้งทางด้านฮาร์ดแวร์ (Hardware) และ ซอฟท์แวร์ (Software) มาใช้ในการควบคุมเครื่องจักรอัตโนมัติ หรือกระบวนการทางอุตสาหกรรม ทำให้การควบคุมกระบวนการ หรือเครื่องจักรต่าง ๆ เหล่านั้นเป็นไปอย่างมีประสิทธิภาพ นอกจากนั้นยังสามารถเปลี่ยนแปลงรูปแบบของการควบคุม และสามารถพัฒนาขีดความสามารถให้สูงขึ้นได้

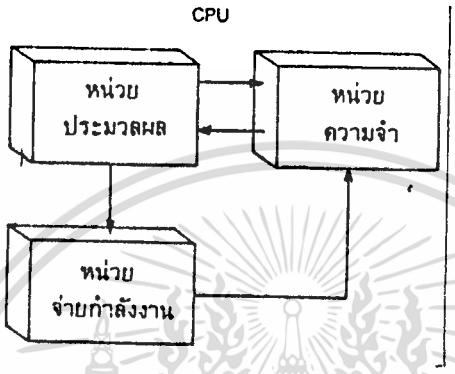
2.1 หลักการทำงานของเครื่องควบคุมที่โปรแกรมได้

เครื่องควบคุมที่โปรแกรมได้ เป็นระบบคอมพิวเตอร์ที่อินพุทและเอาท์พุทได้ถูกออกแบบให้มีสภาวะเหมาะสมกับกระบวนการที่ต้องการควบคุม และมีหน่วยประมวลผลกลางที่ทำหน้าที่ควบคุมการทำงานของระบบทั้งหมด หน่วยประมวลผลกลางหมายถึงระบบไมโครโปรเซสเซอร์ที่ทำหน้าที่ควบคุมการทำงานทั้งหมด โปรแกรมที่ใช้ควบคุมการทำงานของหน่วยประมวลกลางดังกล่าวจะถูกแบ่งออกเป็น 2 ส่วน ส่วนหนึ่งคือโปรแกรมที่ใช้ในการกำหนดลำดับการทำงานของการควบคุม ซึ่งผู้ใช้สามารถที่จะเปลี่ยนแปลงหรือแก้ไขโปรแกรมในส่วนนี้ได้ ซึ่งโปรแกรมในส่วนนี้ก็คือโปรแกรมที่บ่งบอกเงื่อนไขต่าง ๆ ที่ใช้สำหรับการควบคุม โปรแกรมอีกส่วนหนึ่งคือโปรแกรมที่ใช้ในการควบคุมระบบ โปรแกรมส่วนนี้จะทำหน้าที่ควบคุมการรับค่าสภาวะอินพุท ซึ่งได้มาจากกระบวนการ แล้วนำมาประมวลผลทางเอาท์พุท ตามลำดับขั้นตอนของโปรแกรมในส่วนแรก โครงสร้างของเครื่องควบคุมแสดงดังรูปที่ 2.1

รูปที่ 2.1 โครงสร้างของเครื่องควบคุมที่โปรแกรมได้

หน่วยประมวลผลกลาง (Central Processing Unit : CPU)

หน่วยประมวลผลกลางประกอบด้วย ระบบไมโครโปรเซสเซอร์ซึ่งทำหน้าที่ในการประมวลผลข้อมูล หน่วยความจำสำหรับแก้โปรแกรมของผู้ใช้และหน่วยจ่ายกำลังไฟฟ้าดังรูปที่ 2.2



รูปที่ 2.2 โครงสร้างของหน่วยประมวลผลกลาง

ในเครื่องควบคุมบางแบบอาจจะรวมหน่วยประมวลผลกลาง หน่วยความจำและหน่วยจ่ายกำลังไฟฟ้าเข้าไว้ด้วยกันเพื่อให้มีขนาดเล็กกระทัดรัด แต่บางแบบก็อาจจะแยกกันออกเป็นส่วน ๆ เพื่อสะดวกในการบำรุงรักษา ระบบไมโครโปรเซสเซอร์ ซึ่งทำหน้าที่ในการประมวลผลเป็นส่วนประกอบที่สำคัญของเครื่องควบคุม มีหน้าที่ในการประมวลผลข้อมูลการคำนวณทางคณิตศาสตร์และทางลอจิก รวมทั้งควบคุมการทำงานของส่วนต่าง ๆ ทั้งหมด

ในปัจจุบันเทคโนโลยีงางด้านไมโครโปรเซสเซอร์ได้พัฒนาขึ้นมาก จึงได้มีการนำเอาไมโครโปรเซสเซอร์หลาย ๆ ตัวมาทำงานร่วมกัน (Multi-Processor) เพื่อให้มีการประมวลผลได้รวดเร็วและมีประสิทธิภาพยิ่งขึ้น โดยแยกการทำงานของไมโครโปรเซสเซอร์แต่ละตัวเป็นอิสระต่อกันแต่มีการแลกเปลี่ยนข้อมูลกันอยู่ตลอดเวลา นอกจากนี้ในหน่วยอินพุทหรือเอาพ์พุทบางชนิดก็จะมีไมโครโปรเซสเซอร์แยกอิสระจากเครื่องควบคุมอีกที่หนึ่ง เช่น หน่วยอินพุทเอาพ์พุทแบบ PID ซึ่งสามาทำงานได้ด้วยตัวเอง

ค่าสภาวะต่าง ๆ ถูกนำมาจากกระบวนการโดยผ่านทางหน่วยอินพุท ผ่านการประมวลผลจากนั้นผลที่ได้จะถูกส่งออกทางเอาพ์พุท เพื่อไปควบคุมการทำงานของกระบวนการ ตามต้องการและ จะย้อนกลับมารับค่าสภาวะจากกระบวน

การเพื่อนำไปประมวลผล และส่งผลออกทางเอาพ์พุทอีก จะมีการทำอย่างนี้ตลอดไป การรับค่าสภาวะจากภายนอกมาทำการประมวลผลและส่งค่าออกทางเอาพ์พุทในแต่ละครั้งเรียกว่าการสแกน (Scanning) หรือหนึ่งรอบของการทำงาน ระยะเวลาในการสแกนหนึ่งรอบ (1 Scan Time) จะมีค่าเท่ากับระยะเวลาของการอ่านค่าสภาวะอิทพุทครั้งแรกจนกระทั่งมีการอ่านค่าสภาวะอินพุทครั้งต่อไปดังรูปที่ 2.3 ส่วนรายละเอียดของแต่ละหน่วยได้อธิบายตามลำดับในเนื้อหาต่อไป

รูปที่ 2.3 ระยะเวลาการสแกนหนึ่งรอบ

ระยะเวลาในการสแกนจะขึ้นอยู่กับความยาวของโปรแกรมที่ผู้ใช้งานโปรแกรมเงื่อนไขการทำงานเข้าไปเพื่อควบคุม ซึ่งผู้ผลิตมักจะกำหนดไว้ว่าเครื่องควบคุมนั้น ๆ สามารถทำโปรแกรมได้สูงสุดเพียงใด และการใช้ชุดคำสั่งในการโปรแกรมนั้นแต่ละคำสั่งใช้เวลาในการประมวลผลเท่าใดนอกจากนั้นก็จะขึ้นอยู่กับสมรรถนะของเครื่องควบคุมเอง รวมทั้งอุปกรณ์ภายนอกที่ติดต่ออยู่กับเครื่องควบคุม เนื่องาจากการควบคุมกระบวนการ จะต้องกระทำให้อย่างรวดเร็ว และแม่นยำอย่างไรก็ตามถ้าเครื่องควบคุมมีค่าระยะเวลาการสแกนที่มากเกินไป อาจจะทำให้การควบคุมเกิดการผิดพลาดขึ้นได้เพราะว่าค่าสภาวะบางค่าอาจจะมีการเปลี่ยนแปลงที่เร็วมาก ๆ ซึ่งถ้าการเปลี่ยนแปลนี้มีค่าเวลาที่สั้นกว่าค่าระยะเวลากของการสแกนแล้วจะทำให้ข้อมูลที่ได้รับ เพื่อใช้ในการประมวลผลผิดพลาดได้

แต่ในทางปฏิบัติแล้ว การเปลี่ยนแปลงค่าสภาวะอินพุทของกระบวนการมักจะมีระยะเวลาที่ยาวกว่าเวลาการสแกนของเครื่องควบคุม เนื่องจากอุปกรณ์ที่เป็นอินพุทส่วนใหญ่จะเป็นอุปกรณ์ไฟฟ้าเชิงกล หรือสิทช์ชนิดต่าง ๆ ซึ่งมีค่าระยะเวลาการเปลี่ยนแปลงที่ช้ามาก นอกจากจะเป็นค่าสภาวะที่ได้รับมาจาก

อุปกรณ์อิเลคทรอนิคส์ ซึ่งมีความเร็วในการสวิทช์สูง ในการใช้งานลักษณะนี้เครื่องควบคุมจำเป็นจะต้องมีอุปกรณ์พิเศษ เพื่อช่วยในการรับค่าสภาวะมาทำการประมวลผลขั้นหนึ่งก่อน บางแบบอาจจะใช้วิธีอินเตอร์รัพท์ (Interrupt) เพื่อให้เครื่องควบคุมมาอ่านข้อมูลไปประมวลผลทันที หรือนำข้อมูลส่งออกไปเอาพ์พุทในทันที เป็นต้น

หน่วยความจำในหน่วยประมวลผลกลางจะถูกแบ่งออกเป็น 2 ส่วนใหญ่ ๆ คือ ส่วนแรกใช้ในการเก็บโปรแกรมและข้อมูลสำหรับการควบคุมระบบ และอีกส่วนหนึ่งจะใช้ในการเก็บโปรแกรมของผู้ใช้ และข้อมูลสภาวะอินพุทเอาพ์พุท ในเครื่องควบคุมจะมีหน่วยความจำอยู่จำกัดและไม่สามารถขยายได้เหมือนกับคอมพิวเตอร์ เครื่องควบคุมขนาดใหญ่จะมีหน่วยความจุสูงแต่จะมีราคาแพง และเครื่องควบคุมขนาดเล็ก ก็จะมีความจุของหน่วยความจำต่ำ ซึ่งอาจจะไม่เพียงพอต่อการใช้งาน ดังนั้นการเลือกใช้เครื่องควบคุมจะต้องคำนึงถึงสิ่งเหล่านี้ด้วย

2.2 โครงสร้างของ PLC

PLC มีลักษณะเหมือนกับไมโครคอมพิวเตอร์ ซึ่งอุปกรณ์ส่วนใหญ่ได้แก่ CPU memory I/P,O/P และหน่วยจ่ายไฟตรง ข้อแตกต่างหลักระหว่าง PLC กับไมโครคอมพิวเตอร์ คือ

- เทคนิคการโปรแกรม

ดังนั้นการผลิต และการค้นคว้า PLC จึงได้ถูกพัฒนาให้ใช้ควบคู่ไปกับไมโครคอมพิวเตอร์ เป็นการขยายขีดความสามารถของไมโครคอมพิวเตอร์ให้สูงขึ้นอีก ส่วน CPU คือ หัวใจของ PLC จะทำการ Execute คำสั่งต่าง ๆ จากหน่วยความจำ (memory) หน่วยความจำที่ใช้อยู่ในปัจจุบันมีทั้งแบบ EPROM, EEPROM และ RAM ซึ่งแบบ RAM ต้องการใช้แบตเตอรี่ที่เป็นแหล่งจ่ายไฟสำรอง สำหรับสัญญาณที่ใช้อยู่ก็มีระดับตั้งแต่ 5VAC-230VAC ในการแสดงผลของ I/P และ O/P จะถูกเปลี่ยนแปลงหรือไม่ ขึ้นอยู่กับโปรแกรมควบคุมระบบนั้น ๆ โดยทั่วไปแล้ว I/P จะถูกอ่านเข้ามาเมื่อเริ่ม Scanning ส่วน O/P จะถูกกำหนดในช่วงท้ายของการ Scanning ในส่วนที่ซับซ้อนของ PLC คือ เป็นการทำงานที่ต้องมีการคำนวณเลขคณิต ส่วนการควบคุมระบบอุปกรณ์ I/O ของ PLC นั้นจะ

มีการติดต่อกับอุปกรณ์ในการใช้งานทั่วไป เช่น Proximity Switch, Limit Switch ,LED สเตรนเกจ เทอร์โมคับเปิล และอุปกรณ์ interface ต่าง ๆ เช่น Printer Barcode Recorder

ใน I/O modules ที่ผลิตขึ้นมาใหม่ ๆ จะมี Microprocessor program ซึ่งผู้ผลิตเครื่อง เป็นผู้ทำขึ้นโดยสามารถ Process ข้อมูลก่อนและหลังการทำงานของ PLCs Processor ซึ่งทำให้เกิดการทำงานที่มีประสิทธิภาพมากขึ้น ตัวอย่างที่เห็นกันง่าย ๆ เช่นการควบคุม Stepping motor ที่มีความเร็วสูงเท่ากับ 25,000 steps/second ในส่วนของ I/O ที่มี Processor จัดการควบคุมทางด้านนี้แทนโดยที่ PLC ก็ควบคุมการทำงานทั้งหมดของระบบโดยเฉพาะอย่างยิ่งงาน PID Controller นั้น กาว Process Signal จะทำที่ Micro process ของ PLC แทนที่จะใช้การสุ่มตัวอย่าง (Sampling-Speed) ซึ่งทำให้การทำงานมีความเที่ยงตรงและแน่นอน

2.3 หน่วยประมวลผลกลาง (CPU UNIT)

เป็นหน่วยที่ทำหน้าที่ประมวลผลโดยจะตรวจรับข้อมูลอินพุทจากอุปกรณ์ตรวจวัด (Sensor Device) แล้วทำประมวลผลตามโปรแกรมเงื่อนไขที่ได้กำหนดไว้ในหน่วยความจำ จากนั้นก็ส่งสัญญาณเอาท์พุทออกไปเพื่อทำการควบคุมกระบวนการ

หน่วยประมวลผลกลางนี้ประกอบด้วย

1. หน่วยประมวลผล (Processor)
2. หน่วยความจำ (Memory)

รูปที่ 2.2 แสดงโครงสร้างของ หน่วยประมวลผลกลาง โดยหน่วยประมวลผลทำหน้าที่ควบคุมการทำงานทั้งหมดของเครื่องนำข้อมูลมาจากหน่วยความจำ มาปฏิบัติเพื่อควบคุมอุปกรณ์ภายนอกผ่านหน่วยอินพุท-เอาท์พุท ในขณะที่หน่วยจ่ายพลังงานทำหน้าที่จ่ายแรงดันไฟฟ้าให้กับหน่วยประมวลผล และหน่วยความจำ

2.3.1 หน่วยประมวลผล (Processor)

ทำหน้าที่ตรวจสอบสภาวะต่าง ๆ ในหน่วยความจำและประมวลผลตามโปรแกรมต่าง ๆ ที่สร้างขึ้น โดยจะทำซ้ำ ๆ วนรอบเช่นนี้ไปเรื่อย ๆ ในการทำงานแต่ละรอบเรียกว่าการสแกน (Scanning) และระยะเวลาที่ใช้ในการสแกน 1 รอบ เรียกว่า ScanTime ช่วงเวลาของการสแกนขึ้นกับขนาดของหน่วยความจำ (ขนาดของโปรแกรมใช้งานและจำนวนอินพุท-เอาท์พุท)และความเร็วของหน่วยประมวลผล ช่วยเวลาสแกนนี้จะทำให้เราทราบถึงความสามารถในการตอบสนองต่อการเปลี่ยนแปลงของอินพุท-เอาท์พุทของพีแอลซีว่ามีความรวดเร็วเพียงใด เช่น พีแอลซีที่มีช่วงการสแกน 10 msec. ย่อมไม่สามารถรับค่าสภาวะที่แท้จริงของอุปกรณ์ที่มีการเปลี่ยนแปลงทุก 7 msec. ถ้าใช้พีแอลซีควบคุมอุปกรณ์ดังกล่าว จะทำให้ผลการควบคุมผิดพลาดหมด โดยปกติแล้วช่วงเวลาสแกน จะใช้เวลาประมาณ 1-100 msec.

2.3.2 หน่วยความจำ (Memory)

ส่วนนี้ทำหน้าที่เก็บโปรแกรมและข้อมูลต่าง ๆ ที่พีแอลซี ใช้ในการประมวลผลแบ่งหน่วยความจำออกเป็น 2 ส่วนคือ

1. หน่วยความจำระบบ (System Memory) เก็บโปรแกรมบริหารระบบและข้อมูลของระบบ

2. หน่วยความจำผู้ใช้ (User Memory) เป็นโปรแกรมผู้ใช้ ข้อมูลของหน่วยอินพุท-เอาท์พุทและอุปกรณ์ภายใน

หน่วยความจำที่นำมาใช้กับ พีแอลซี มี 2 ชนิด เช่นเดียวกับที่ใช้กับคอมพิวเตอร์ทั่ว ๆ ไป คือ

1. วอลอะทิล (volatile)

2. นอนวอลอะทิล (nonvolatile)

ข้อมูลภายในหน่วยความจำแบบวอลอะทิลจะสูญหายหมด ถ้าไม่มีกระแสไฟฟ้า (สามารถใช้ battery backup ได้) ส่วนข้อมูลภายหน่วยความจำแบบนอนวอลอะทิบจะยังคงอยู่ถึงแม้ว่าจะไม่มีกระแสไฟฟ้า

โปรแกรมบริหารระบบ (Supervisory Program หรือ Operating System) และข้อมูลของระบบ จะเก็บอยู่ในหน่วยความจำแบบนอนวอลอะทิล เช่น PROM EPROM EEPROM

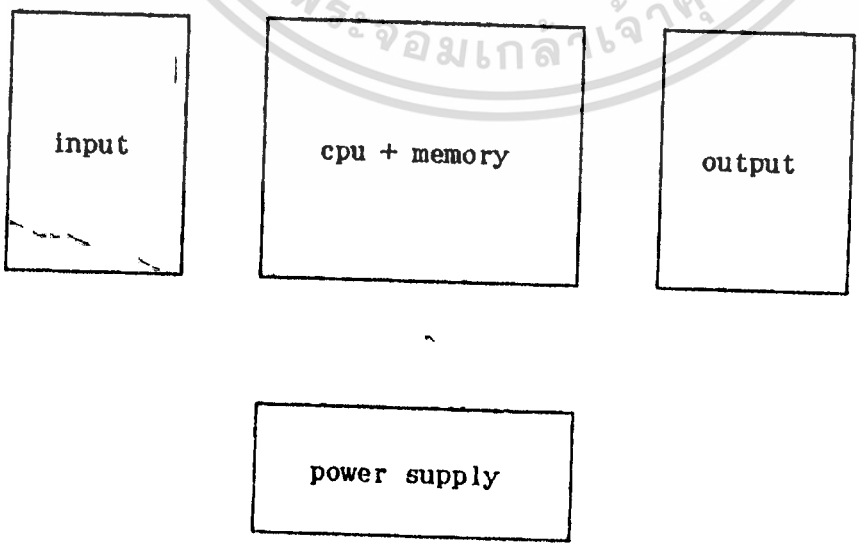
ในขณะที่ โปรแกรมผู้ใช้ (User Program) อาจเก็บอยู่ในหน่วยความจำแบบใดก็ได้ ถ้าเป็นโปรแกรมที่ไม่ต้องการเปลี่ยนแปลงแก้ไขอีกก็จะเก็บในลักษณะเดียวกับโปรแกรมบริหารระบบ แต่ถ้ายังจำเป็นต้องมีการตรวจสอบแก้ไขโปรแกรมนั้นอยู่ก็จะเก็บไว้ในหน่วยความจำประเภท RAM โดยอาจมีหน่วยจ่ายพลังงานสำรองจ่ายให้ชั่วคราว ขณะไฟฟ้าดับ

ส่วนข้อมูลที่เป็นข้อมูลของหน่วยอินพุท-เอาท์พุทและอุปกรณ์ภายใน (data memory) จะถูกเก็บอยู่ในหน่วยความจำแบบ RAM เท่านั้นเพราะเป็นข้อมูลที่มีการประมวลผลเปลี่ยนแปลงตลอดเวลา โดยอาจมีหน่วยพลังงานสำรองหรือไม่มีก็ได้

2.4 หน่วยป้อนโปรแกรม (Programming Unit)

ทำหน้าที่ติดต่อระหว่าง PC กับผู้ใช้ รับโปรแกรมที่เขียนขึ้นเก็บไว้ในหน่วยความจำปกติหน่วยป้อนโปรแกรม จะต่อเชื่อมกับ PLC เมื่อผู้ใช้ต้องการป้อน ตรวจสอบ หรือแก้ไขโปรแกรมเท่านั้น

และ PLC สามารถทำงานได้โดยไม่ต้องพึ่งหน่วยป้อนโปรแกรม ดังนั้นหมายป้อนโปรแกรม จึงไม่ได้ถูกจัดเป็นส่วนประกอบของ PLC



รูปที่ 2.4 ส่วนประกอบหลักของ PLC

แบ่งได้ดังนี้

- cpu+memory module
- input module (ส่วนคีย์บอร์และแสดงผลอยู่ในส่วนนี้ด้ว
- output module

2.5 หน่วยจ่ายพลังงาน (Power Supply)

หน่วยจ่ายหลังงาน ทำหน้าที่จ่ายกระแสไฟฟ้าให้ส่วนต่าง ๆ ของ พีแอลซี คือหน่วยประมวลผล หน่วยความจำ หน่วยอินพุท-เอาท์พุท โดยรักษาระดับแรงดันไฟฟ้าให้คงที่ "ความน่าเชื่อถือของพีแอลซี ขึ้นกับการทำงานของหน่วยจ่ายหลังงาน"

2.6 การใช้งานจอภาพแสดงผลแบบ Dot Matrix LCD Module

จอภาพแสดงผลแบบ Dot Matrix LCD Module เป็นส่วนแสดงผลที่มีประสิทธิภาพสูง สามารถแสดงตัวอักษร ตัวเลขต่าง ๆ ได้ครบถ้วน อีกทั้งสามารถสร้างตัวอักษรขึ้นใช้เองได้อีกด้วย โดยเขียนข้อมูลเก็บไว้ใน Character Genertor RAM ของ LCD Module

ภายใน Dot Matrix LCD Module จะมีหน่วยควบคุมการทำงาน (controller) อยู่ในตัวเอง นั่นคือ HD44780 Dot-Matrix LCD Controller & Driver ดังนั้นเพียงแต่เราค่าสั่งควบคุมต่าง ๆ ตามลำดับขั้นตอนที่ถูกต้อง ก็สามารถควบคุมและใช้งาน LCD นี้ได้ตามความต้องการ

2.6.1 Initialization

ก่อนเริ่มใช้งาน LCD Module ต้องมีการ initialization ให้แก่ HD44780 (controller) เสียก่อน ซึ่งมีอยู่ 2 วิธี คือ

1) Initialization by internal reset circuit :

ตามปกติแล้วทุกครั้งที่เริ่มจ่ายไฟเลี้ยงให้กับ Module HD44780 จะทำการ reset (initialization) โดยอัตโนมัติ ด้วยวงจรภายในตัวเอง ซึ่งมีลำดับขั้นตอนตามค่าสั่งข้างล่างนี้ และ Busy Flag จะมีค่า = 1 จนกระทั่งสิ้นสุด initialization ซึ่งใช้เวลาประมาณ 10 ms. หลังจาก V_{cc} มีค่า = 4.5 volt

- Display Clear
- Function set DL = 1 : รับส่งข้อมูลขนาด 8 บิท

  N = 0 : แสดงผล 1 บรรทัด

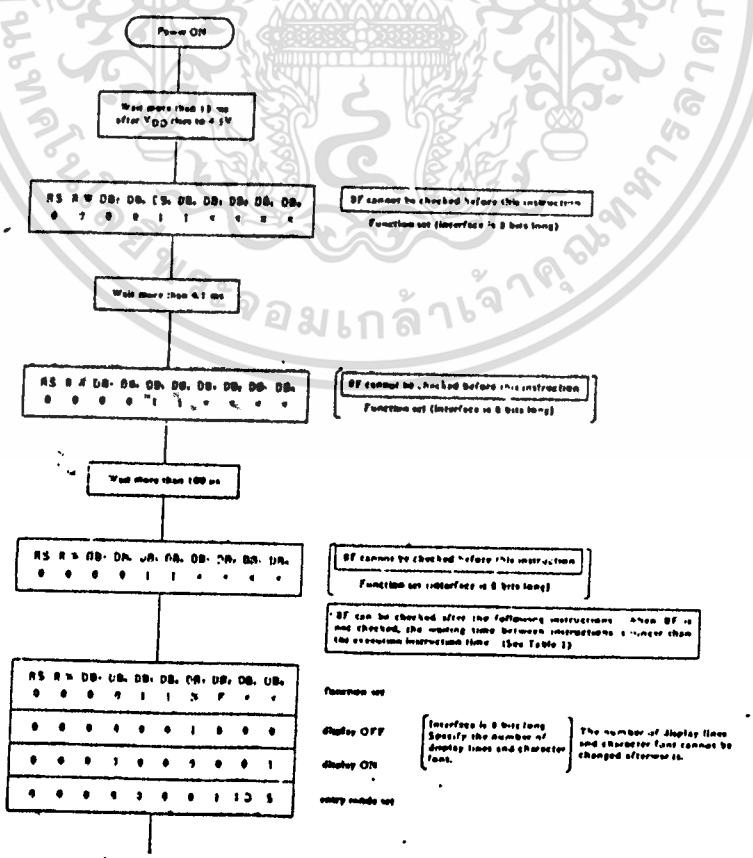
  F = 0 : 5*7 dot character font
- Display ON/OFF control D = 0 : display OFF

  C = 0 : cursor OFF

  B = : blink OFF
- Bntry mode set I/D = : +1 (increment)
- Write DD RAM

2) Initialization ด้วย คำสั่ง :

ในกรณีที่ internal reset circuit ไม่สามารถทำงานได้ตามปกติ (ซึ่งมักจะมีผลมาจากไฟเลี้ยง) เป็นผลให้ Initialization ไม่ได้ด้วยตัวเอง จึงมีความจำเป็นต้องดำเนินการ ด้วยคำสั่งดัง Flow Chart ในรูปที่ 2.5



2.6.2 โครงสร้างภายนอกและการต่อใช้งาน LCD Module

การนำ LCD Module มาต่อเพื่อใช้งานนั้น สิ่งที่สำคัญที่สุดคือ ต่อขาสัญญาณให้ถูกต้อง โดยเฉพาะอย่างยิ่ง ไฟเลี้ยง เพราะเป็นส่วนสำคัญที่จะทำให้ LCD Module เสียหายได้

LCD Module ที่นำมาใช้งานมีขาใช้งานอยู่ 14 ขา แบ่งออกได้ 3 กลุ่ม ดังนี้คือ

1. กลุ่มขาไฟเลี้ยง มี 3 ขา คือ V_{dd} (ขา 1), V_{ss} (ขา 2), V_o (ขา 3) โดยมีหน้าที่ต่างกัน คือ V_{dd} เป็นขาไฟเลี้ยงวงจรขนาด 5 volt, V_{ss} เป็นขา Ground ของวงจร V_{dd}-V_o เป็นระดับโวลท์เตจปรับความเข้มของจอภาพ

2. กลุ่มขาข้อมูลขนาด 8 บิท คือ DB_0-DB_7 มี 8 ขา คือ ขา 7-14 ทำหน้าที่รับ-ส่งข้อมูลกับภายนอก เพื่อเป็นคำสั่งแก่ LCD Module

3. กลุ่มขาควบคุม จำนวน 3 ขา คือ RS (ขา 4), R/W (ขา 5), E (ขา 6) ทำหน้าที่ควบคุมคำสั่งที่ป้อนเข้ามา

2.6.3 คำสั่งควบคุมการทำงานของ LCD Module

CLEAR DISPLAY

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	0	0	0	0	0	0	0	0	0	1

คำสั่งนี้เป็นการเขียนช่องว่าง หรือ Space (ASCII 20H) เข้าไปใน DD RAM (display date RAM) ทั้งหมด และทำการเซ็ต DD RAM แอดเดรสให้เป็นศูนย์ตัวเคอดร์เซอร์จะกลับไปอยู่ตำแหน่งมุมบนซ้ายของจอภาพ เซ็ต I/D=1, S ไม่มีการเปลี่ยน

RETURN HOME

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	0	0	0	0	0	0	0	0	1	*

คำสั่งนี้จะทำการเซ็ต DD RAM แอดเดรสให้เป็นศูนย์ ตัวเดอร์เซอร์จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจดภาพ ข้อมูลในจอภาพ (DD RAM) ไม่เปลี่ยน

ENTRY MODE SET

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	0	0	0	0	0	0	0	1	I/D	S

BIT I/D : โดยจะเป็นตัวกำหนดให้ว่าเมื่อเขียน หรืออ่านข้อมูลแล้ว จะทำให้ DD RAM แอดเดรสเพิ่มขึ้น หรือลดลงหนึ่ง โดย 1=เพิ่ม, 0=ลดลงหนึ่ง

BIT S : เป็นตัวกำหนดการแสดงผล ถ้า S=1 จะเป็นการใส่ข้อมูลแล้วตัวเดอร์เซอร์จะถูกดันไปทางซ้าย ถ้า S=0 ข้อมูลจะอยู่กับที่ตัวเดอร์เซอร์จะถูกดันไปทางขวามือ

DISPLAY ON/OFF CONTROL

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	0	0	0	0	0	0	1	D	C	B

BIT D : เป็นบิตที่ใช้ควบคุมการปิดเปิดหน้าจอ โดยถ้า D = 1 จะ ON และ D = 0 จะ OFF

BIT C : ใช้แสดง curror เพื่อให้บิต C = 1 และถ้าไม่ต้องการ cursor ก็ใช้บิต C = 0 โดยที่ตัว cursor จะอยู่ที่แถวที่ 8

BIT B : เป็นบิตที่ใช้เซ็ตการกระพริบของ cursor โดย B = 1 มีการกระพริบ ถ้า B = 0 ไม่มีการกระพริบ ระยะเวลาการกระพริบประมาณ 379.2 ms.

CURSOR AND DISPLAY SHIFT

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	0	0	0	0	0	1	S/C	R/L	*	*

เป็นคำสั่งกำหนดให้ตำแหน่ง cursor หรือข้อมูลไปปรากฏทางซ้าย หรือขวาโดยไม่ต้องใช้คำสั่งเขียน หรืออ่านโดย

S/C	R/L	
0	0	ทำการย้าย cursor จากตำแหน่งเดิมไปทางซ้ายหนึ่งตำแหน่ง
0	1	ทำการย้าน cursor จากตำแหน่งเดิมไปทางขวาหนึ่งตำแหน่ง
1	0	เป็นการดันตัวอักษรที่ปรากฏไปทางซ้ายมือ
1	1	เป็นการดันตัวอักษรที่ปรากฏไปทางขวามือ

FUNCTION SET

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	0	0	0	0	1	DL	N	F	*	*

DL : เป็นการเซ็ตการติดต่อว่าจะให้เป็นแบบ 8 หรือ 4 บิต โดยถ้าต้องการติดต่อ 4 บิต DL = 0 และ 8 บิต DL = 1

N : เป็นการเซตบรรทัดการแสดงผล N = 0 แสดง 1 บรรทัด

N = 1 แสดง 2 บรรทัด (หรือมากกว่า)

F : เป็นการเซตขนาดจุดของการแสดงผล F = 0 เป็นแบบ 5*7

F = 1 เป็นแบบ 5*10

SET CG RAM ADDRESS

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	0	0	0	1	A	A	A	A	A	A

(01AAAAAA : CG RAM address)

คำสั่งนี้จะเป็นการเซตแอดเดรสใน CG RAM โดยต้องทำการเซตแอดเดรสก่อนเขียนหรืออ่านข้อมูลจาก CG RAM

SET DD RAM ADDRESS

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	0	0	1	A	A	A	A	A	A	A

(1AAAAAAA: DD RAM address)

เป็นค่าสั่งเซตค่าแอดเดรสใน DD RAM ในการเขียนหรืออ่านค่าจาก DD RAM โดยจำนวนแอดเดรสที่จะเกิดขึ้นบนจอ LCD ขึ้นอยู่กับการเซตค่า N ด้วย

ถ้า N = 0 (1 บรรทัด) แอดเดรสอยู่ที่ 00h-4fh

ถ้า N = 1 (2 บรรทัด) แอดเดรสจะอยู่ที่ 00h-27h สำหรับบรรทัด 1

ที่ 40h-67h สำหรับบรรทัด 2

READ BUSY FLAG & ADDRESS

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	0	1	BF	A	A	A	A	A	A	A

(DF : Busy Flag

AAAAAAA : CG cr DD RAM address)

เป็นค่าสั่งอ่านค่า busy flag ซึ่งเป็นตัวบอกว่า HD4780 อยู่ในขบวนการทำงานภายใน หรืออยู่ในสภาพพร้อมรับข้อมูล โดย

BF = 1 ไม่พร้อมรับข้อมูลหรือค่าสั่ง

BF = 0 พร้อมรับข้อมูลหรือคำสั่ง

นอกจากนี้ยังเป็นคำสั่งอ่านค่าข้อมูลแอดเดรสของ CG RAM หรือ DD RAM

|

WRITE DATA TO CG or DD RAM

	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	1	0	D	D	D	D	D	D	D	D

(DDDDDDDD : Data to CG or DD RAM)

เป็นคำสั่งเขียนข้อมูลเข้าไปใน CG หรือ DD RAM โดยเมื่อเขียนข้อมูลเข้าไปในแอดเดรสที่กำหนดไว้ก่อนหน้านี้ และจากนั้นแอดเดรสจะเพิ่มหรือลดโดยอัตโนมัติตามคำสั่งเซ็นใน entry mode

READ DATA FROM CG or DD RAM

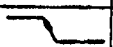
	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0
Code	1	1	D	D	D	D	D	D	D	D

(DDDDDDDD : Data from CG or DD RAM)

เป็นคำสั่งอ่านค่าข้อมูลจาก CG RAM หรือ DD RAM โดยก่อนอ่านค่าจาก CG RAM หรือ DD RAM ควรจะใช้คำสั่งเซตแอดเดรสก่อนเพื่อให้รู้ว่าข้อมูลที่อ่านได้นั้นเป็นข้อมูลจาก CG RAM หรือ DD RAM

2.6.4 สัญญาณจากขา E

สำหรับขา E จะทำหน้าที่ในการรับคำสั่งต่าง ๆ ที่ได้กล่าวมาทั้งหมดข้างต้นตรวจ โดยจะมีการทำงานแตกต่างกันตามชุดคำสั่ง ดังตารางข้างล่างนี้

RS	R/W	E	Operation
0	0		IR write as internal operation (Display clear, etc.)
0	1		Read busy flag (DB7) and address counter (DB0 ~ DB6)
1	0		DR write as internal operation (DR to DD or CG RAM)
1	1		DR read as internal operation (DD or CG RAM to DR)

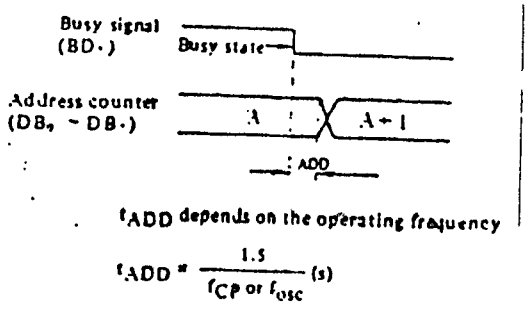
รูปที่ 2.6 แสดงสภาวะของสัญญาณขา E ขณะส่งค่าคำสั่งต่าง ๆ

2.6.5 Busy Flag

เป็น Flag ของ LCD Module ที่ใช้สำหรับตรวจสอบว่า LCD Module กำลังอยู่ในสภาวะ excute หรือไม่ ซึ่งถ้าใช่ Busy Flag จะมีค่า = 1 ดังนั้นก่อนที่จะเขียนคำสั่งใหม่ให้กับ LCD Module จะต้องมั่นใจว่า Busy Flag มีค่า = 0 ซึ่งสามารถมั่นใจได้ดังนี้ คือ

1. คำสั่งแต่ละคำสั่งจะใช้เวลาในการ exeeute ไม่เท่ากัน (ดังรายละเอียดในตาราง) ซึ่งเป็นช่วงเวลาที่ Busy Flag = 1 (ไม่สามารถเก็บรับคำสั่งใหม่ได้) ดังนั้นก่อนที่จะส่งคำสั่งใหม่แก่ HD44780 (controller) จะต้องหน่วงเวลาไว้ไม่น้อยกว่า execute time ของคำสั่งก่อนหน้า (รายละเอียดข้อมูลในรูปที่ 2.4)

2. ภายหลังจากการ exccute คำสั่งอ่านหรือเขียน CG/DD RAM ระบบภายใน LCD Module จะเพิ่มหรือลด RAM counter โดยอัตโนมัติ ซึ่งจะมีเวลาหน่วงหลังจากที่ falling dege ของ Busy Flag = t_{ADD} = $1.5/f_{osc}$ (sec.) (f_{osc} : operating frequency)

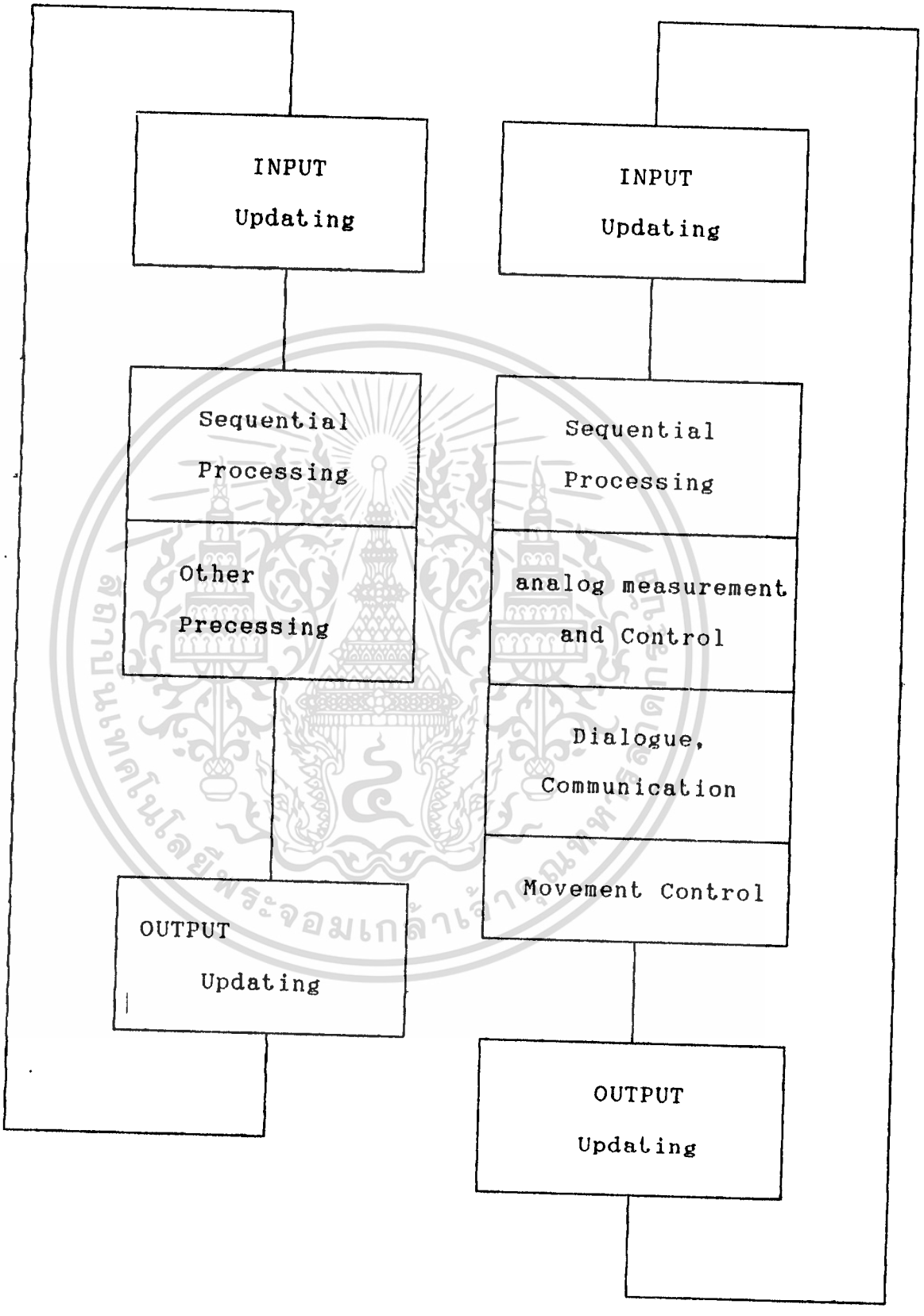


รูปที่ 2.7 แสดง Busy Flag

สรุปข้อควรระวังในการเขียนคำสั่งได้ว่า

1. ในการเขียนคำสั่งแต่ละสั่งเข้าสู่ LCD Module จะต้องกำหนดการทำงานของขา E ที่เหมะสม

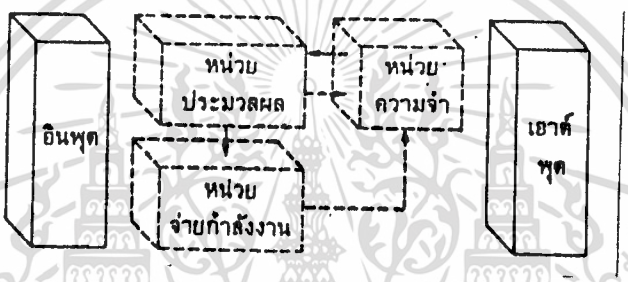
2. ระหว่างคำสั่งแต่ละคำสั่งที่เขียนเข้าสู่ LCD Module จะต้องมีระยะเวลาห่างกันมากกว่า execute time ของแต่ละคำสั่ง



รูปที่ 2.8 ขั้นตอนการทำงานของ PLC

2.7 หน่วยอินพุท/หน่วยเอาต์พุต

หน่วยอินพุต/เอาต์พุต ทำหน้าที่ติดต่อระหว่าง PC กับอุปกรณ์ภายนอก หน่วยอินพุตทำหน้าที่รับสภาวะและค่าวัดจากอุปกวณ์ภายนอก เช่น การ ON/OFF ของสวิตท์ตำแหน่งเครื่องจักร ระดับของเหลว อุณหภูมิ ความดัน ระดับแรงดัน และกระแสไฟฟ้าส่งต่อให้ PC CPU จะใช้ค่าหรือสภาวะจากหน่วยอินพุต/เอาต์พุต เป็นข้อมูลในการประมวลผลตามโปรแกรมผู้ใช้ และส่งผลที่ได้ไปที่หน่วยเอาต์พุต เพื่อควบคุมอุปกรณ์ภายนอก เช่น รีเลย์ มอเตอร์ไฟฟ้า ปั๊ม และ วาล์ว



รูปที่ 2.9 หน่วนอินพุต/เอาต์พุต

ระยะแรก PC มีหน่วยอินพุต/เอาต์พุตแบบสภาวะลอจิก (discrete input/output) เพียงชนิดเดียว ทำให้ PC ถูกใช้ในการควบคุมแบบ ON/OFF ซึ่งเป็นส่วนหนึ่งของระบบควบคุมทั้งหมดเท่านั้น ปัจจุบันหน่วยอินพุต/เอาต์พุตแบบตัวเลข (numerical data input/output) ทำให้ขอบเขตการใช้ PC ในการควบคุมกว่างขึ้น ทั้งการควบคุมเครื่องจักรและกระบวนการอุตสาหกรรม ดังรายละเอียดต่อไปนี้

2.7.1 หน่วยอินพุต/เอาพุตแบบสภาวะลอจิก

หน่วยอินพุต/เอาท์พุตชนิดนี้ ทำหน้าที่ติดต่อกับอุปกรณ์ภายนอกที่มีการเปลี่ยนแปลงเพียง 2 สภาวะ เช่น การ ON/OFF ของสวิตช์ไฟฟ้า หรือหน้าสัมผัสของรีเลย์

2.7.1.1 สัญญาณมาตรฐานของอุปกรณ์อินพุต/เอาต์พุตแบบสภาวะลอจิก

สัญญาณไฟฟ้าที่อุปกรณ์อินพุต/เอาต์พุตแบบสภาวะลอจิกใช้มีหลายค่า ทั้ง

ไฟฟ้ากระแสตรงและกระแสสลับ สัญญาณมาตรฐานที่หน่วยอินพุต/เอาต์พุตของ PC ใช้เชื่อมต่อกับอุปกรณ์ภายนอกได้แสดงในตารางที่ 2.1

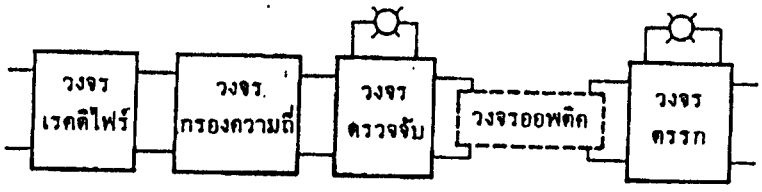
อุปกรณ์อินพุต	อุปกรณ์เอาต์พุต
24 VAC/DC	12-48 VAC/DC
48 VAC/DC	120 VAC/DC
120 VAC/DC	220 VAC/DC
220 VAC/DC	หน้าสัมผัสรีเลย์
ระดับ TTL	ระดับ TTL

ตารางที่ 2.1 สัญญาณมาตาฐานของอุปกรณ์พุต/เอาต์พุตแบบสภาวะลอจิก

หน่วยอินพุตและสภาวะลอจิก ทำหน้าที่รับค่าทางไฟฟ้าของอุปกรณ์ภายนอกและเปลี่ยนเป็นสภาวะทางลอจิกเพื่อเก็บไว้ในหน่วยความจำส่วนตารางอินพุตของ CPU บิตที่มีค่า "1" หมายถึง การ ON หรือปิดวงจรไฟฟ้า และบิตที่มีค่า "0" หมายถึง การ OFF หรือเปิดวงจรไฟฟ้าของอุปกรณ์อินพุต หน่วยเอาต์พุตจะรับสภาวะลอจิกจากตารางเอาต์พุตของ CPU และเปลี่ยนเป็นค่าทางไฟฟ้าเพื่อควบคุมอุปกรณ์ภายนอกอีกที่หนึ่ง บิตที่มีค่า "1" หมายถึงการ ON หรือการต่อวงจรไฟฟ้า และบิตที่มีค่า "0" หมายถึง OFF หรือการตัดวงจรไฟฟ้า การเลือกใช้หน่วยินพุต/เอาต์พุตชนิดที่ถูกต้องและเหมาะสม จำเป็นต้องเข้าใจคุณลักษณะและการทำงานของหน่วยอินพุต/เอาต์พุตแต่ละชนิด

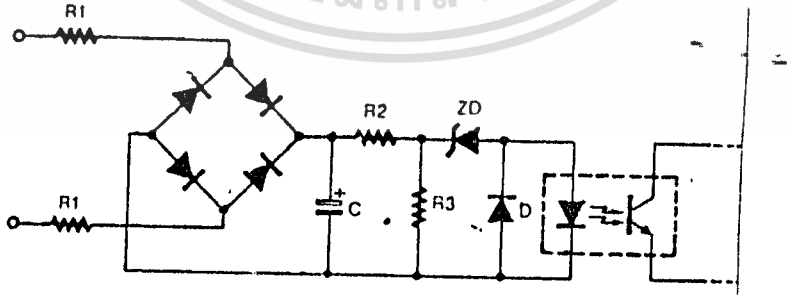
2.7.1.2 หน่วยอินพุตแบบ AC/DC

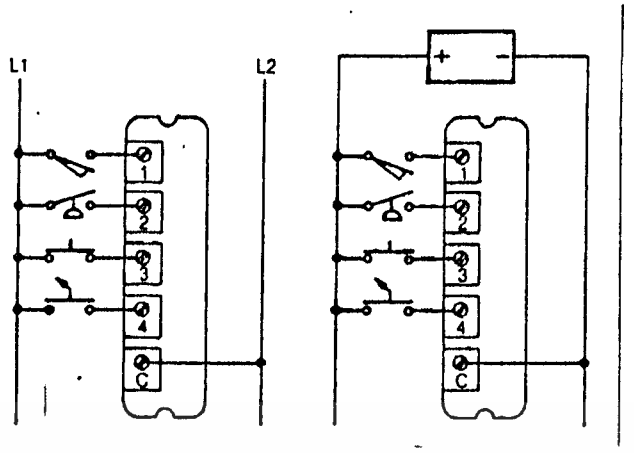
หน่วยอินพุตแบบ AC/DC ทำหน้าที่ รับสัญญาณไฟฟ้าจากอุปกรณ์อินพุต เปรียบเทียบระดับแรงดันไฟฟ้าและเปลี่ยนเป็นสภาวะทางลอจิกส่งไปยัง CPU หน่วยอินพุตแบบ AC/DC ประกอบด้วยวงจรเรคติไฟร์ (rectifier) วงจรกรองความถี่ (filter) วงจรตรวจจับ (detector) และวงจรตรรกที่ทำหน้าที่ส่งข้อมูล



รูปที่ 2.10 หน่วยอินพุตแบบ AC/DC

วงจรเรคติไฟร์ ทำหน้าที่เปลี่ยนสัญญาณ AC หรือ DC เป็นระดับสัญญาณ DC วงจรกรองความถี่ ทำหน้าที่กำจัดสัญญาณรบกวน วงจรตรวจจับ ทำหน้าที่เปรียบเทียบระดับสัญญาณเพื่อตรวจสอบสภาวะของอุปกรณ์อินพุตและวงจรตรรกทำหน้าที่ติดต่อกับ CPU ระหว่างวงจรตรรกกับวงจรส่วนอื่นจะใช้การเชื่อมต่อแบบออพติค (optical coupling) เพื่อป้องกัน CPU จากการรบกวนของอุปกรร์ภายนอก ดังในรูป และแสดงโครงสร้าง และวงจรของหน่วยอินพุตแบบ AC/DC หน่วยอินพุตชนิดนี้ ต้องใช้ เวลาตรวจสอบสภาวะของอุปกรณ์อินพุตประมาณ 9 ถึง 5 ms เนื่องจากการทำงานของวงจรเรคติไฟร์ และวงจรกรองความถี่ หน่วยอินพุตชนิดนี้จะมีสัญญาณไฟแจ้งสภาวะของอุปกรณ์อินพุตให้ผู้ใช้ทราบ รูปที่ 2.11 ได้แสดงการเชื่อมต่อหน่วยอินพุตแบบ AC/DC กับอุปกรณ์ภายนอก

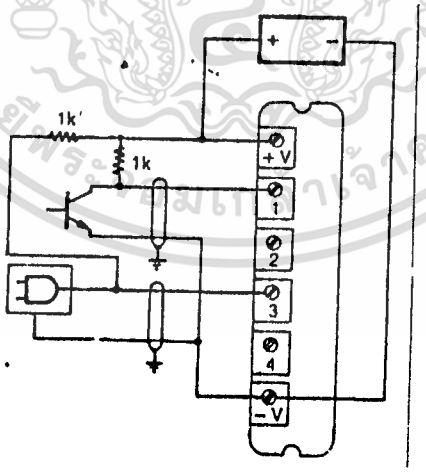




รูปที่ 2.11 การเชื่อมต่อหน่วยอินพุตแบบ AC/DC กับอุปกรณ์ภายนอก

2.7.1.3 หน่วยอินพุตแบบ TTL

หน่วยอินพุตแบบ TTL ทำหน้าที่รับสัญญาณอินพุตจากอุปกรณ์อิเล็กทรอนิกส์ชนิด TTL (Transistor Transistor Logic) หรือ อุปกรณ์ที่ใช้สัญญาณไฟฟ้า 5-VDC หน่วยอินพุตชนิดนี้ไม่จำเป็นต้องมีวงจรเรคติไฟร์และวงจรกรองความถี่ จึงทำงานได้กว่าหน่วยอินพุตแบบ AC/DC คือ ใช้เวลาเพียง 1 ถึง 3 ms. รูปที่ 2.12 แสดงการเชื่อมต่อหน่วยอินพุตแบบ TTL



รูปที่ 2.12 แสดงการเชื่อมต่อหน่วยอินพุตแบบ TTL

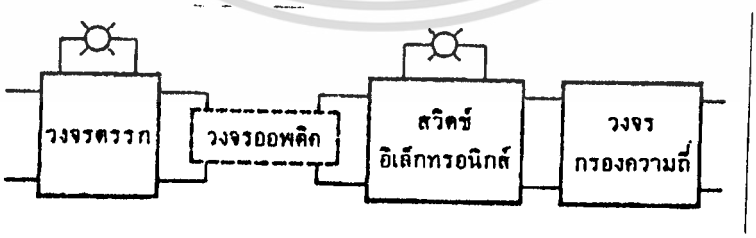
2.7.1.4 หน่วยอินพุตแบบหน้าสัมผัส (Contact input)

หน่วยอินพุตแบบหน้าสัมผัส ทำหน้าที่ ตรวจสอบสภาวะของอุปกรณ์อินพุตที่ไม่มีแรงดันไฟฟ้าในตัว เช่นสวิตช์ไฟฟ้า และหน้าสัมผัสของรีเลย์ โดยหน่วยอินพุตชนิดนี้จะใช้แรงดันไฟฟ้า 12 หรือ 24 VDC จ่ายให้อุปกรณ์ภายนอก

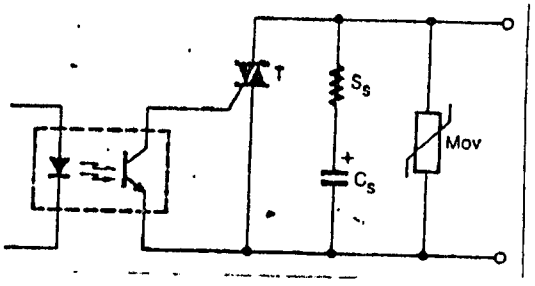
2.7.1.5 หน่วยเอาต์พุตแบบ AC

หน่วยเอาต์พุตแบบ AC ทำหน้าที่รับสภาวะการควบคุมจาก CPU เปลี่ยนเป็นระดับแรงดันไฟฟ้า กระแสสลับเพื่อควบคุมการทำงานของอุปกรณ์เอาต์พุต · หน่วยเอาต์พุตแบบ AC ประกอบด้วยวงจรตรรกทำหน้าที่ติดต่อกับ CPU วงจรเชื่อมต่อแบบออพติค สวิตช์อินเล็กทรอนิกส์ (electronics switch) และวงจรกรองความถี่ รูปที่ 2.13 แสดงส่วนประกอบ และวงจรของหน่วยเอาต์พุตแบบ AC

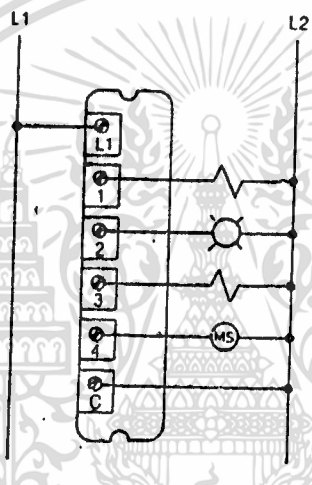
วงจรตรรกรับสัญญาณควบคุมจาก CPU และควบคุมการทำงานของสวิตช์อิเล็กทรอนิกส์โดยใช้การเชื่อมต่อแบบออพติค และวงจรกรองถี่ทำหน้าที่ปรับแรงดันเอาต์พุตให้สม่ำเสมอสวิตช์อิเล็กทรอนิกส์ในหน่วยเอาต์พุตชนิดนี้มักใช้ไทร์แอค (triac) หรือ SCR ซึ่งเป็นและปิดวงจรไฟฟ้าโดยไม่เกิดประกายไฟ เช่น หน้าสัมผัสของรีเลย์ จึงไม่รบกวนการทำงานของอุปกรณ์ภายนอกและยืดอายุการใช้งาน หน่วยเอาต์พุตชนิดนี้มักมีสัญญาณไฟแจ้งสภาวะการ ON หรือ OFF ให้ผู้ใช้ทราบ รูปที่ 2.14 แสดงการเชื่อมต่อหน่วยเอาต์พุตแบบ AC กับอุปกรณ์ภายนอก



รูปที่ 2.13 แสดงส่วนประกอบ



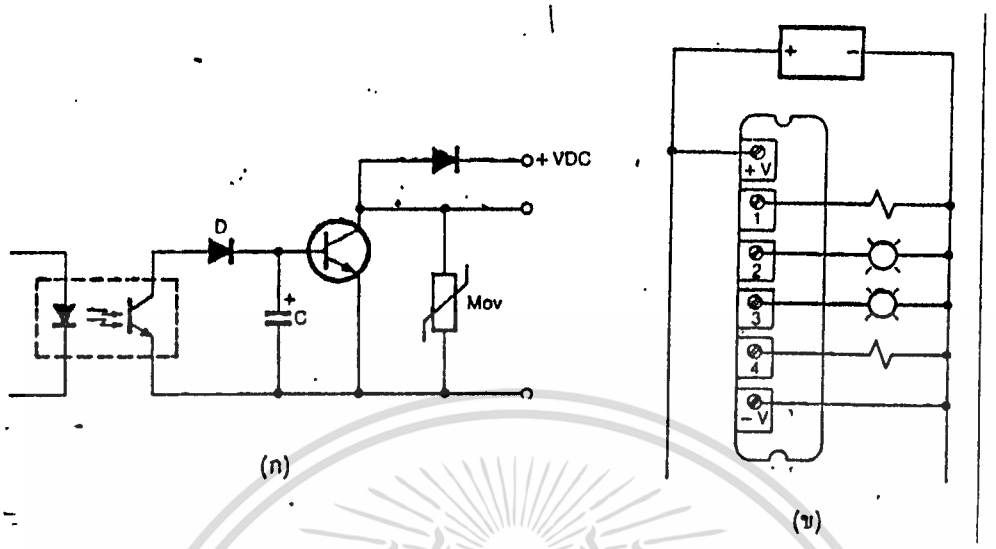
วงจรของ หน่วยเอาต์พุตแบบ AC



รูปที่ 2.14 แสดงการเชื่อมต่อหน่วยเอาต์พุตแบบ AC กับอุปกรณ์ภายนอก

2.7.1.6 หน่วยเอาต์พุตแบบ DC

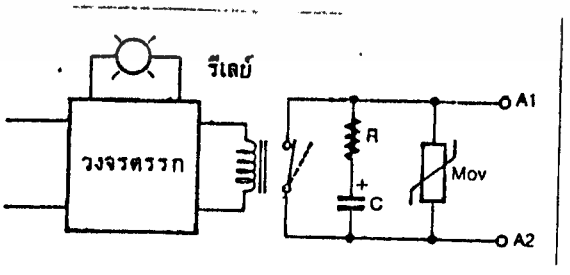
หน่วยเอาต์แบบ DC ทำหน้าที่ ควบคุมอุปกรณ์เอาต์พุตทำงานด้วยสัญญาณไฟฟ้ากระแสตรงลักษณะและการทำงานของวงจรคล้ายกับหน่วยเอาต์แบบ AC แต่ใช้ทรานส์ซิสเตอร์เปิดและปิดวงจรไฟฟ้าแทนไทร์แอคและ SCR



รูปที่ 2.15 วงจรหน่วงเอาต์พุตแบบ DC และ การเชื่อมต่อกับอุปกรณ์ภายนอก

2.7.1.7 หน่วยเอาต์พุตแบบหน้าสัมผัส (Contact output)

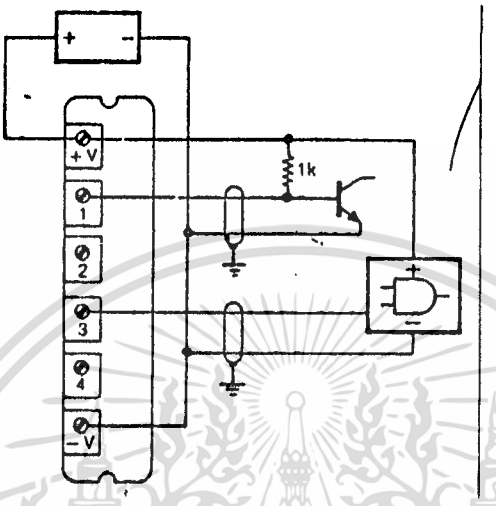
หน่วยเอาต์พุตแบบหน้าสัมผัส ทำหน้าที่แทนหน้าสัมผัสของรีเลย์ในการควบคุมอุปกรณ์ ที่มีแรงดันไฟฟ้าของตนเอง ลักษณะและการทำงานของวงจรคล้ายกับหน่วยเอาต์พุตแบบ AC และ DC แต่ใช้หน้าสัมผัสและรีเลย์เปิดและปิดวงจรไฟฟ้าแทนอุปกรณ์อิเล็กทรอนิกส์ หน่วยเอาต์พุตชนิดนี้ใช้กับอุปกรณ์ไฟฟ้ากระแสตรงหรือกระแสสลับกับได้



รูปที่ 2.16 วงจรหน่วยเอาต์พุตแบบหน้าสัมผัส

2.7.1.8 หน่วยเอาต์พุตแบบ TTL

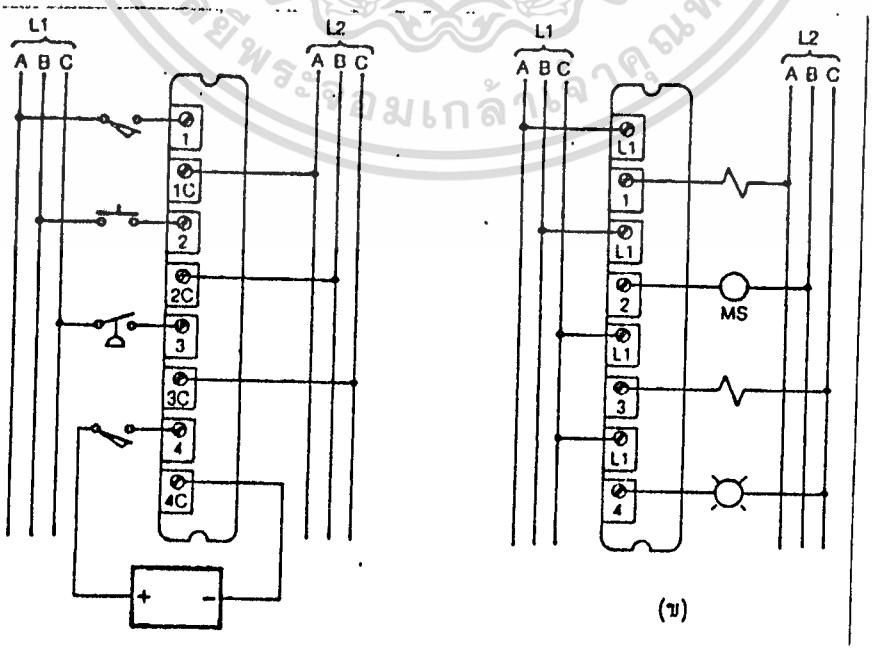
หน่วยอินพุตแบบ TTL ทำหน้าที่ควบคุมการทำงานของอุปกรณ์อิเล็กทรอนิกส์ที่เป็น TTL หรืออุปกรณ์ที่ใช้สัญญาณไฟฟ้า 5 VDC หน่วยเอาต์พุตชนิดนี้มักต้องการหน่วยจ่ายกำลังงานขนาด 5 VDC จากภายนอก



รูปที่ 2.17 หน่วยเอาต์พุตแบบ TTL

2.7.1.9 หน่วยอินพุต/เอาต์พุตแบบอิสระ (Isolated input/output)

หน่วยอินพุต/เอาต์พุตแบบอิสระ ทำหน้าที่คล้ายกับหน่วยอินพุต/เอาต์พุตแบบหน้าสัมผัส แต่แยกวงจรไฟฟ้าของอุปกรณ์ทั้งหมดออกจากกัน ทำให้อุปกรณ์ทุกชิ้นมีวงจรไฟฟ้าและหน่วยจ่ายกำลังงานของตนเอง



รูปที่ 2.18 หน่วยอินพุต/เอาต์พุตแบบอิสระ

2.7.2 หน่วยอินพุต/เอาต์พุตแบบตัวเลข

ไมโครโปรเซสเซอร์ทำให้ PC สามารถ คำนวณทางคณิตศาสตร์ ตรวจสอบและเคลื่อนย้ายข้อมูลที่เป็นตัวเลข หน่วยอินพุต/เอาต์พุตแบบจำนวนเลข ที่ให้ PC สามารถรับข้อมูลจากอุปกรณ์วัด และควบคุมเครื่องจักรหรือกระบวนการอุตสหากรรม ระบบควบคุมที่ใช้ PC จึงครบวงจรทั้งการควบคุมแบบ ON/OFF และอนาลอก

หน่วยอินพุต/เอาต์พุตแบบตัวเลขมี 2 ชนิด คือ แบบรหัสเลขฐานสอง และ แบบอนาลอกหน่วยอินพุต/เอาต์พุตแบบรหัสเลขฐานสองทำงานคล้ายกับแบบสภาวะลอจิก แต่การรับและส่งข้อมูลทำได้พร้อมกันครั้งละหลายบิต ข้อมูลอาจอยู่ในรูปรหัส ASCII รหัส BCD หรือรหัส GRAY หน่วยอินพุต/เอาต์พุตแบบอนาลอกทำหน้าที่รับ และส่งข้อมูลที่เป็นสัญญาณไฟฟ้า ทำให้ PC สามารถติดต่อกับอุปกรณ์วัดและควบคุมในกระบวนการอุตสาหกรรม โดยตารางที่ 87050 ได้แสดงตัวอย่างอุปกรณ์อินพุต/เอาต์พุตแบบจำนวนเลขทั้ง 2 ชนิด

2.7.2.1 สัญญาณมาตรฐานของอุปกรณ์อินพุต/เอาต์พุตแบบตัวเลข

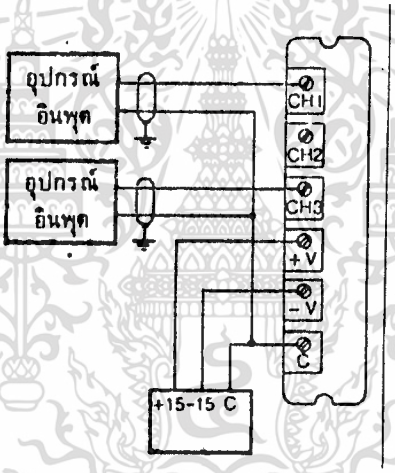
สัญญาณไฟฟ้าของอุปกรณ์อินพุต/เอาต์พุตแบบอะนาลอกอาจมีทั้งค่าบวกและลบหรือค่าบวกหรือลบอย่างใดอย่างหนึ่ง ซึ่งผู้ใช้สามารถปรับเลือกขนาดและชนิดของสัญญาณได้ โดยใช้วิธีทางซอฟต์แวร์หรือ ฮาร์แวร์ตามที่บริษัทผู้ผลิตกำหนด ตารางที่ 2.2 สำหรับหน่วยอินพุต/เอาต์พุตแบบรหัสเลขฐานสองมักจะใช้สัญญาณชนิดเดียวกับอุปกรณ์อินพุต/เอาต์พุตแบบสภาวะลิจิก

อุปกรณ์อินพุต	อุปกรณ์เอาต์พุต
4 – 20 mADC	4 – 20 mADC
0 – 1 VDC	10 – 50 mADC
0 – 5 VDC	0 – 5 VDC
0 – 10 VDC	0 – 10 VDC
1 – 5 VDC	±25 VDC
±5 VDC	±5 VDC
±10 VDC	±10 VDC

ตารางที่ 2.2 สัญญาณมาตรฐานของอุปกรณ์อินพุต/เอาต์พุตแบบอะนาลอก

2.7.2.2 หน่วยอินพุตแบบอะนาลอก

หน่วยอินพุตแบบอะนาลอก ทำหน้าที่รับสัญญาณอินอุปกรณ์ภายนอก ปรับระดับให้เหมาะสม และใช้วงจรแปลงสัญญาณนาลอก/ดิจิตอล หรือ ADC (Analog to Digital Converter) เปลี่ยนสัญญาณอะนาลอกเป็นค่าทางดิจิตอลส่งให้ CPU วงจรส่วนแรกของหน่วยอินพุตคือ วงจรกันชน และวงจรกรองความถี่เพื่อลดสัญญาณรับกวนจากภายนอก วงจรกันชนทำหน้าที่เพิ่มค่าอิมพิแดนซ์ด้านอินพุต (input impedance) ให้สูงขึ้นเพื่อเชื่อมต่อกับอุปกรณ์อินพุตที่มีอิมพิแดนซ์สูง สายส่งที่เชื่อมต่อระหว่างหน่วยอินพุตและอุปกรณ์ภายนอกควรเป็นสายโคแอ๊คเชียลเพื่อป้องกันการรบกวนจากสนามไฟฟ้าและรักษาค่าอิมพิแดนซ์ให้คงที่ การเชื่อมต่อหน่วยอินพุตแบบอะนาลอกับอุปกรณ์ภายนอกได้แสดงในรูปที่ 2.19



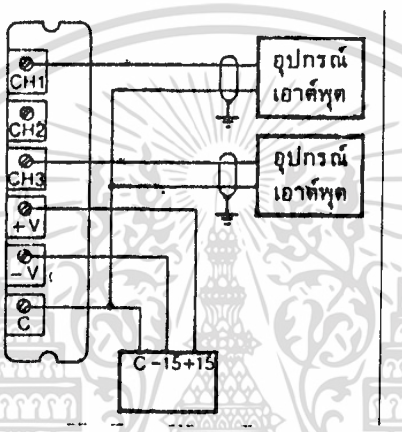
รูปที่ 2.19 การเชื่อมต่อหน่วยอินพุตแบบอะนาลอกกับอุปกรณ์ภายนอก

2.7.2.3 หน่วยเอาต์พุตแบบอะนาลอก

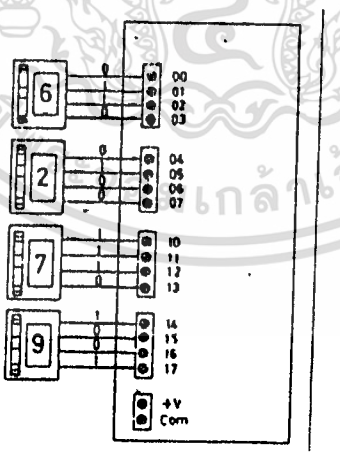
หน่วยเอาต์พุตแบบอะนาลอกทำหน้าที่รับข้อจาก CPU เปลี่ยนเป็นระดับสัญญาณไฟฟ้า โดยใช้วงจรแปล่งสัญญาณดิจิตอล/อะนาลอก หรือ DAC (Digital to Analog Coverter) เพื่อควบคุมอุปกรณ์ภายนอกหน่วยเอาต์พุตชนิดนี้ใช้การเชื่อมต่อแบบออฟติค เพื่อป้องกันการรบกวน CPU จากอุปกรณ์ภายนอก การเชื่อมต่ออุปกรณ์เอาต์พุตแบบอะนาลอกกับ PC ในรูปที่ 2.20 ต้องให้หน่วยจ่ายกำลังงานภายนอกเพื่อให้วงจร DAC ทำงาน

2.7.2.4 หน่วยอินพุตแบบรีจิสเตอร์ (Register input)

หน่วยอินพุตแบบรีจิสเตอร์ ทำหน้าที่รับข้อมูลรหัสเลขฐานสอง เช่น รหัส BCD และรหัส GRAY จากอุปกรณ์ภายนอก เช่น สวิตช์รหัส ส่งให้ CPU อุปกรณ์อินพุตมักใช้สัญญาณไฟฟ้าขนาด 5 VDC หรือ ระดับ TTL โดยสัญญาณไฟฟ้า 0 VDC หมายถึงสภาวะลอจิก "0" และ 5 VDC หมายถึงสภาวะลอจิก "1" การรับข้อมูลของหน่วยอินพุตมีจำนวนครั้งละ 16 หรือ 32 บิต รูปที่ 2.21 แสดงการเชื่อมต่อหน่วยอินพุตแบบรีจิสเตอร์



รูปที่ 2.20 การเชื่อมต่ออุปกรณ์เอาต์พุตแบบอะนาลอก



รูปที่ 2.21 การเชื่อมต่อหน่วยอินพุตแบบรีจิสเตอร์

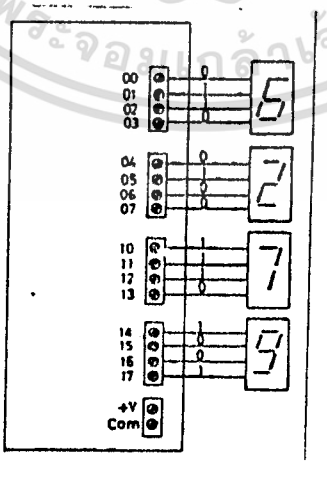
2.7.2.5 หน่วยเอาต์พุตแบบรีจิสเตอร์ (Register output)

หน่วยเอาต์พุตแบบรีจิสเตอร์ ทำหน้าที่ส่งข้อมูลรหัสเลขฐานสอง เช่น รหัส BCD รหัส CRAY จาก CPU ส่งให้อุปกรณ์เอาต์พุต เช่น ตัวแสดงผลเจ็ด

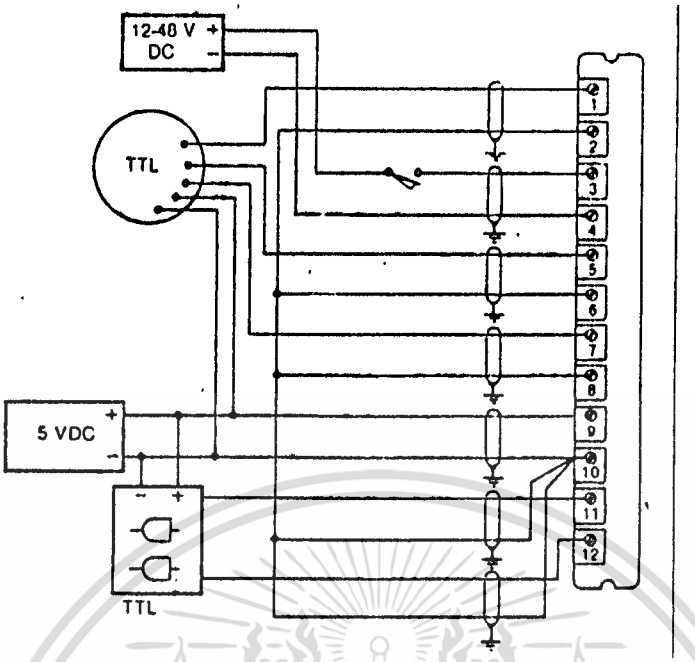
ส่วน และจอภาพ สัญญาณไฟฟ้าของหน่วยเอาต์พุต มีขนาด 5 ถึง VDC และจ่ายกระแสไฟฟ้า ได้ประมาณ 0.5 แอมแปร์ รูป ที่ 2.22 แสดงการเชื่อมต่อหน่วยเอาต์พุตแบบรีจิสเตอร์

2.7.2.6 หน่วยเชื่อมต่อหน่วยจรนับและเข้ารหัส

หน่วยเชื่อมต่อแบบวงจรนับและเข้ารัหสทำหน้าที่เป็นวงจรนับความเร็วสูงการทำงานอิสระจาก CPU วงจรนับจะทำการนับทันทีที่ได้รับสัญญาณจากภายนอก หน่วยเชื่อมต่อชนิดนี้ใช้ตรวจนับจำนวนสินค้าและตรวจสอบตำแหน่งเครื่องจัด วงจรนับจะติดต่อรับคำสั่งเริ่มต้น หยุด และยกเลิกการทำงาน ค่าเริ่มต้นและค่าสุดท้ายในการนับจาก CPU ขณะที่วงจรนับทำงานจะส่งผลการนับให้วงจรเข้ารหัสเปลี่ยนค่าเป็น รหัส BCD หรือ รหัส GRAY ส่งไปยัง CPU ทุกช่วงการสแกนเมื่อนับได้จำนวนที่ต้องการ หน่วยเชื่อมต่อจะส่งสัญญาณควบคุมไปยังอุปกรณ์ภายนอก และแจ้งให้ CPU ทราบ ความเร็วในการนับตั้งแต่ 100 Hz ถึง 50 khz รูปที่ 2.23 แสดงการเชื่อมต่อวงจรนับและเข้ารหัส หน่วยเชื่อมต่อ ต้องใช้หน่วยจ่ายกำลังงานจากภายนอก หน่วยเชื่อมต่อรับสัญญาณอินพุตขนาด 48 VDC และ ใช้สัญญาณเอาต์พุตขนาด 24 VDC หรือ ระดับ TTL ถ้าการเชื่อมต่อมีระยะเกินกว่า 50 ฟุต ต้องใช้สายส่งแบบ โคแอคเชียลเพื่อป้องกันการรวบกวนจากสนามไฟฟ้า



รูปที่ 2.22 การเชื่อมต่อหน่วยเอาต์พุตแบบบรีจิสเตอร์



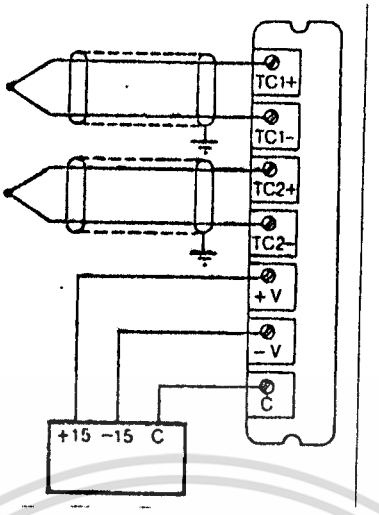
รูปที่ 2.23 แสดงการเชื่อมต่อวงจรนับและเข้ารหัส

2.7.3 หน่วยเชื่อมต่อแบบพิเศษ

PC ส่วนใหญ่ใช้หน่วยอินพุต/เอาต์พุตแบบสภาวะลอจิกและแบบตัวเลข ในการตรวจสอบและควบคุมอุปกรณ์ภายนอก ทั้งเครื่องจักรและกระบวนการอุตสาหกรรม นอกจากหน่วยอินพุต/เอาต์พุต ทั้ง 2 ชนิด การควบคุมบางอย่างอาจต้องใช้หน่วยเชื่อมต่อพิเศษ เช่น หน่วยอินพุตที่ใช้กับสัญญาณระดับต่ำ หรือมีความเร็วสูงมาก ๆ หน่วยอินพุต/เอาต์พุตที่ใช้ไมโครโปรเซสเซอร์สำหรับงานควบคุมที่ต้องการตัดสินใจอิสระจาก PC และทำงานร่วมกับ CPU ในลักษณะของระบบมัลติโปรเซสเซอร์ หน่วยเชื่อมต่อแบบพิเศษนี้มีใช้ใน PC ขนาดกลาง และขนาดใหญ่เท่านั้น

2.7.3.1 หน่วยอินพุตแบบเทอร์โมคัปเปิล (Thermocouple input)

หน่วยอินพุตแบบเทอร์โมคัปเปิลทำหน้าที่ติดต่อกับ CPU เพื่อรับสัญญาณอินพุตจาก เทอร์โมคัปเปิล ที่มีระดับสัญญาณต่ำมาก เพียงประมาณ 50 mVDC เท่านั้น หน่วยอินพุตประกอบด้วยวงจรชดเชยอุณหภูมิ วงจรปรับและขยายสัญญาณให้เหมาะสม และวงจร ADC การใช้หน่วยอินพุตชนิดนี้ต้องทราบชนิดของเทอร์โมคัปเปิลที่ใช้ก่อนเสมอ เพราะเทอร์โมคัปเปิลแต่ละชนิดมีคุณลักษณะต่างกัน



รูปที่ 2.24 การเชื่อมต่ออินพุตแบบเทอร์โมคัปเปิล

2.7.3.2 หน่วยอินพุตความเร็วสูง (High speed input)

หน่วยอินพุตความเร็วสูง ทำหน้าที่ตรวจจับสัญญาณพัลส์ที่มีความเร็วสูงและช่วงกว้างสั้นมากจนหน่วยอินพุตแบบสภาวะลอจิกตรวจสอบไม่ได้ หน่วยอินพุตชนิดนี้ จะสอดแทรกการทำงานของ CPU ทันทีที่ตรวจพบสัญญาณพัลส์จากภายนอกสัญญาณพัลส์ที่มีขนาด 10 ถึง 24 VDC และมีความกว้างประมาณ 50 ถึง 100 us. ซึ่งเร็วกว่าการสแกนตามปกติของ PC มาก จะสามารถกระตุ้นให้หน่วยอินพุตชนิดนี้ทำงาน

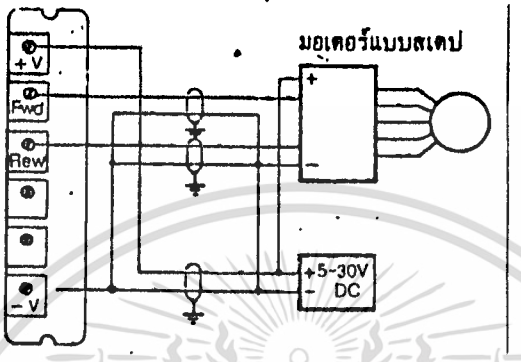
2.7.3.3 หน่วย ASCII

หน่วย ASCII ทำหน้าที่ ติดต่อ รับส่งข้อมูลที่เป็นรหัส ASCII ระหว่าง PC กับอุปกรณ์ร่วมภายนอก เช่น จอภาพ เครื่องพิมพ์ การรับและส่งข้อมูลมีหลายวิธี เช่น RS-232 RS-422 และลูปกระแส (current loop) 20 mADC

2.7.3.4 หน่วยอินพุตแบบสเตรนเกจ (Strain gage input)

หน่วยอินพุตแบบสเตรนเกจ ทำหน้าที่วัดค่าความเครียด (strain) จากสเตรนเกจ

2.7.3.5 หน่วยเชื่อมต่อมอเตอร์แบบสเตป (Stepper motor interface) ทำหน้าที่ควบคุมเตอร์แบบสเตป ให้จำนวนสัญญาณพัลส์ กำหนดระยะการหมุน ความถี่กำหนดความเร็ว และการเปลี่ยนแปลงความถี่กำหนดความเร่งและความหน่วง

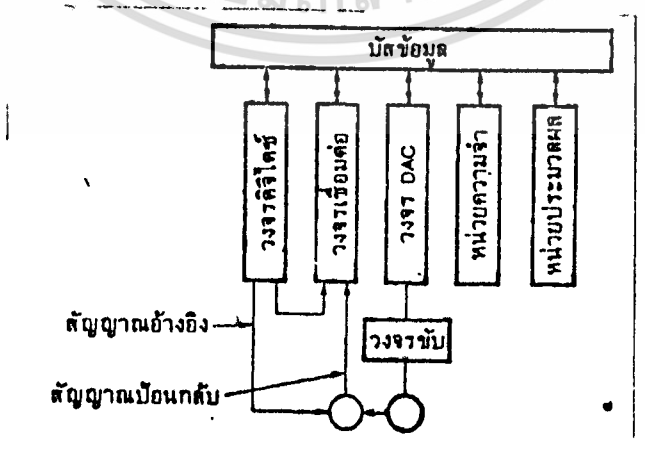


รูปที่ 2.25 หน่วยเชื่อมต่อมอเตอร์แบบสเตป

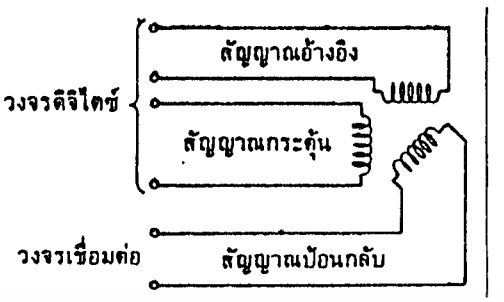
2.7.3.6 หน่วยเชื่อมต่อเซอร์โว (Servo interface) ใช้ในการควบคุมระบบคลัตช์และเกียร์ หรือตำแหน่งเครื่องจักร แทนคอมพิวเตอร์ควบคุมเครื่องจักรเชิงเลข (CNC : Computer Numerical Control)

ประกอบด้วยการทำงาน 2 ส่วน คือ

- สัญญาณอ้างอิงที่ถูกส่งออกจากหน่วยเชื่อมต่อ
- สัญญาณที่ถูกป้อนกลับจากอุปกรณ์ภายนอก



รูปที่ 2.26 ตัวอย่างของการเชื่อมต่อเซอร์โว



รูปที่ 2.27 วงจรดิจิไตซ์

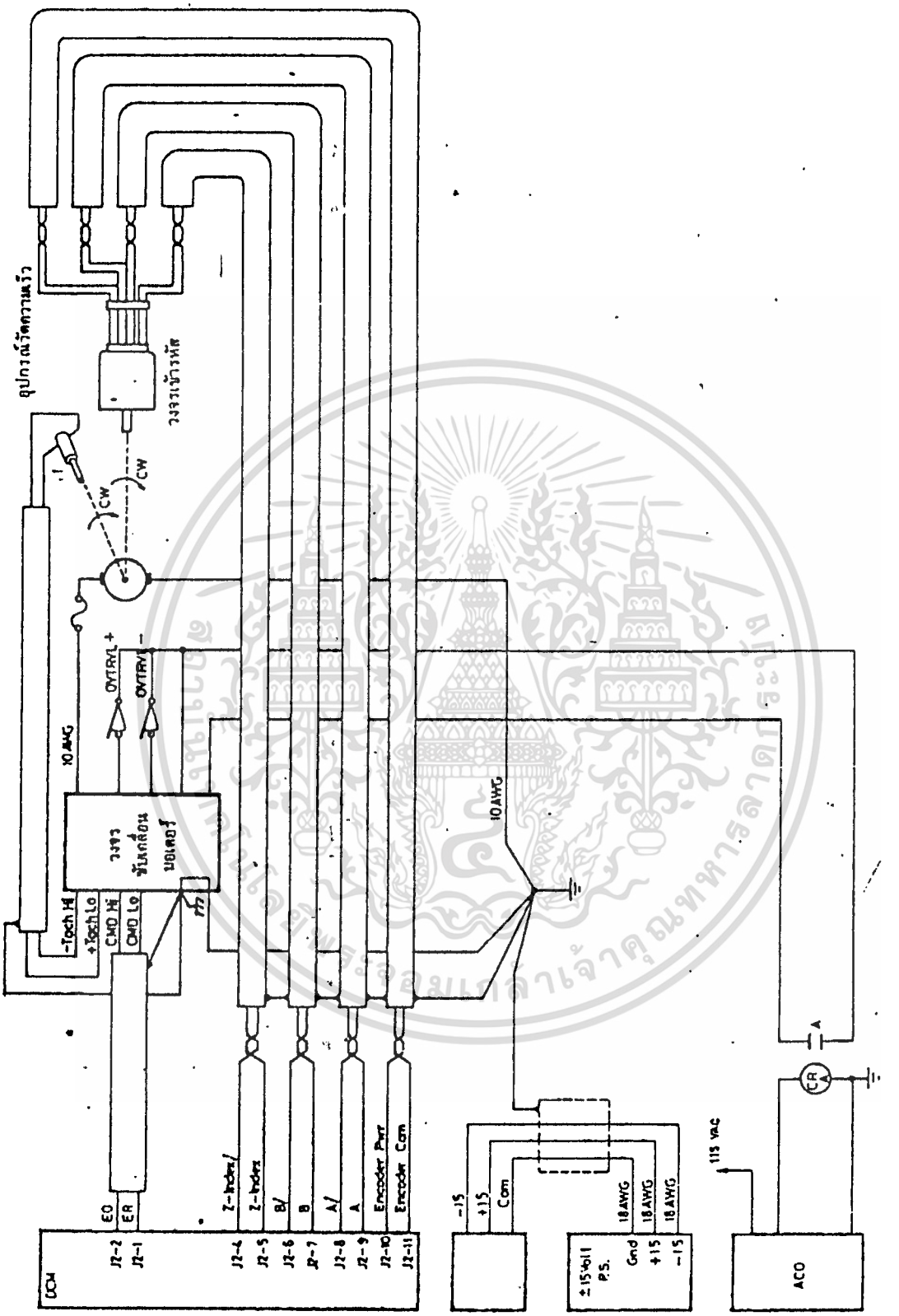
รูปที่ 2.28 สัญญาณที่สร้างขึ้นโดยวงจงดิจิไตซ์

2.7.3.7 หน่วยเชื่อมต่อ FDM (Feedrate Motor) และ DCM (DC Motor)

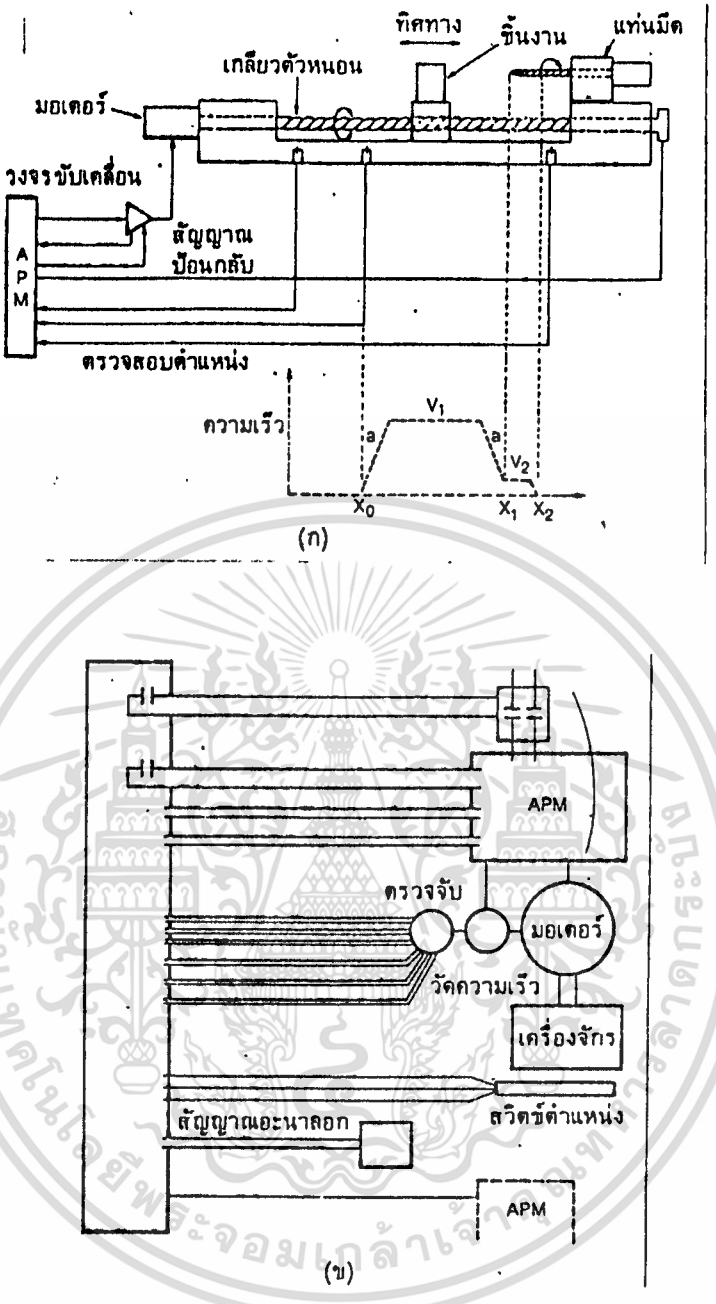
ควบคุมระบบเซอร์โว หน่วย FDM ส่งสัญญาณพัลส์แจ้งตำแหน่งเซอร์โวให้หน่วย DCM หน่วย DCM จะเปรียบเทียบตำแหน่งของเซอร์โวกับกับจุดที่ต้องการ และส่งสัญญาณควบคุมผ่านวงจร DAC แสดงในรูปที่ 2.29

2.7.3.8 หน่วย APM (Axis Positioning Modle)

หน่วย APM ทำหน้าที่ ควบคุมระบบเซอร์โวให้ทำงานเป็นลำดับขั้นตอนที่ต้องการโดยใช้ไมโครโปรเซสควบคุม หน่วย APM ทำงานอิสระจาก CPU แสดงในรูปที่ 2.30



รูปที่ 2.29 การเชื่อมต่อหน่วย FDM และ DCM



รูปที่ 2.30 ตัวอย่างการใช้หน่วย APM (ก) และการเชื่อมต่อ (ข)

2.7.3.9 หน่วย PID

หน่วย PID ทำหน้าที่เป็นเครื่องควบคุมแบบ PID ในกระบวนการอุตสาหกรรม เช่น การควบคุมเตาเผา ระดับและความดันของหม้อน้ำ อัตราการไหลของก๊าซ โดยหน่วย PID จะตรวจสอบค่าตัวแปรกระบวนการ หรือ PV (Process Variable) ที่ต้องการควบคุม ผ่านวงจร ADC เปรียบเทียบกับค่าเป้าหมาย หรือ SP (Set Point) นำผลการเปรียบเทียบมาประมวลผลหา

สัญญาณควบคุม คือ CV (Control Variable) เพื่อควบคุมกระบวนการ โดยใช้วงจร DAC ผลการทำงานของหน่วย PID จะทำให้ตัวแปรกระบวนการมีค่าเท่ากับ เป้าหมายที่ต้องการหน่วย PID ทำงานโดยใช้หลักการ แสดงในรูปที่ 2.31

3.7.3.10 หน่วยประมวลข้อมูล (Data processing module)

คำนวณทางคณิตศาสตร์ ตรวจสอบและ เครื่อนย้ายข้อมูล

2.7.3.11 หน่วยเชื่อมต่อโครงข่าย (Network interface module)

เชื่อมต่อโครงข่ายของ PC เข้ากับ ระบบโครงข่าย ทำให้ PC สามารถติดต่อแลกเปลี่ยนข้อมูลกับ PC ต่างระบบ คอมพิวเตอร์ ระบบควบคุมอื่น และอุปกรณ์ร่วมภายนอก เช่น จอภาพและเครื่องพิมพ์ การรับส่งข้อมูลระหว่าง PC โดยใช้หน่วยเชื่อมต่อ โครงข่ายจะมีความเร็วสูงมาก

2.7.4 หน่วยอินพุต/เอาต์พุตแบบรีโมต

มีหน่วยประมวลผลและหน่วยจ่ายกำลังงานของตนเอง การทำงานอิสระจาก CPU การเชื่อมต่อระหว่าง CPU และหน่วยอินพุต/เอาต์พุต แบบรีโมตใช้สายส่งคู่เดียว ทำให้ค่าใช้จ่ายในการเดินสายระหว่าง PC กับหน่วยอินพุต/เอาต์พุตลดลง

การเชื่อมต่อมี 2 วิธี คือ

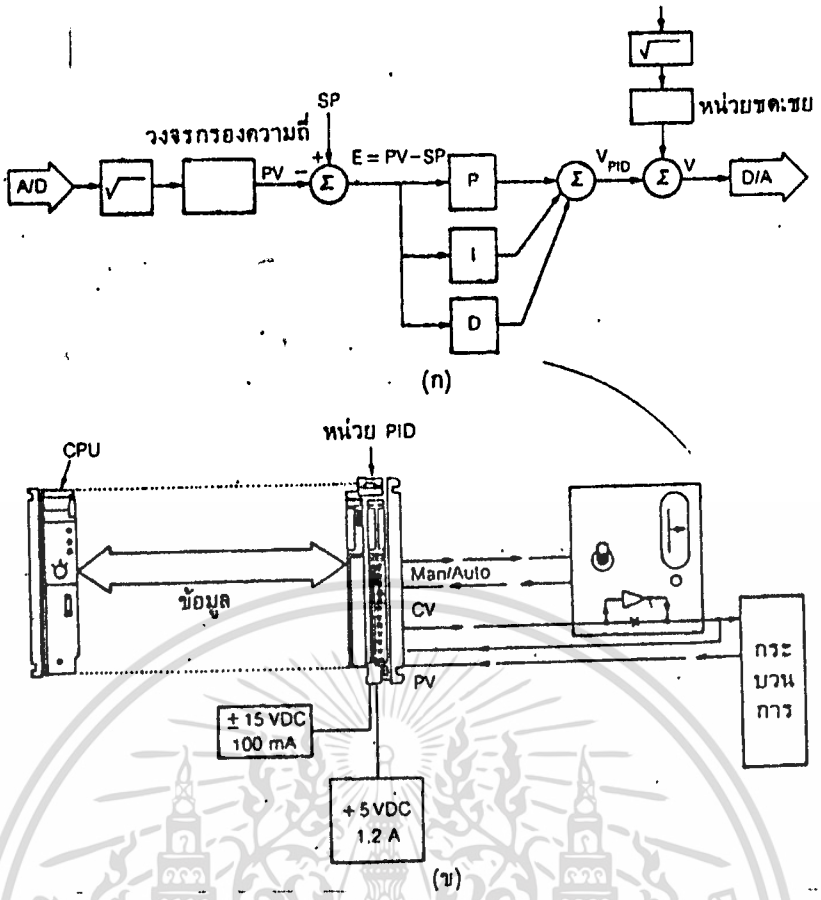
- การเชื่อมต่อระบบลูกโซ่ (daisy chain)
- การเชื่อมต่อระบบแบบกระจาย 9star)

$CV = K_p E + K_i \int E.dt + K_d . dE/dt$

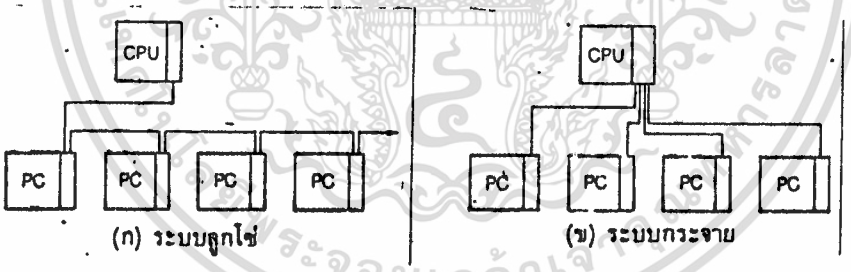
$E = PV - SP$ โดย K_p คือ อัตราขยายของเครื่องควบคุม (proportional gain)

K_i คือ ค่าคงที่ในการอินทีเกรต (integral gain) และ

K_d คือ ค่าคงที่ในการหาอนุพันธ์ (derivative gain)



รูปที่ 2.31 โครงสร้างของหน่วย PID (ก) และการเชื่อมต่อ (ข)

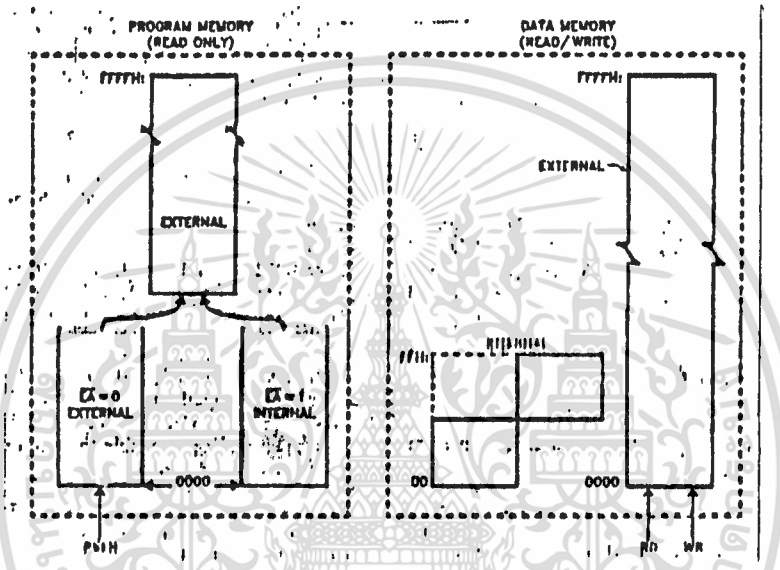


รูปที่ 2.32 การเชื่อมต่อหน่วยอินพุต เอาต์พุตแบบรีโมต

บทที่ 3 โครงสร้างของไมโครโปรเซสเซอร์ MCS-51

ในส่วนนี้จะกล่าวถึง รายละเอียดบางประการของไมโครโปรเซสเซอร์ตระกูล MCS-51 แนวทางในการออกแบบ และพัฒนา ดังจะเสนอต่อไปนี้

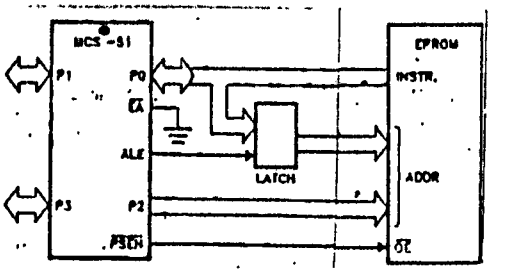
ส่วนฮาร์ดแวร์ของ MCS-51



รูปที่ 1

Reset address 0000H

External Interrupt0	0003H	Time0	000BH
External Interrupt1	0013H	Time1	B001BH



รูปที่ 2

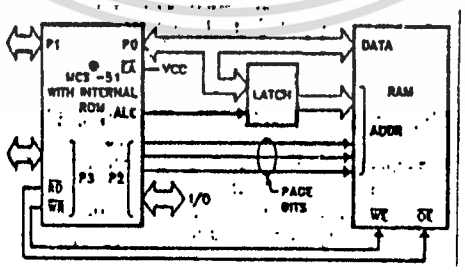
จากรูปที่ 2 แสดงถึงการต่อทางด้าน hardware โดยใช้ external program execution

จะเห็นได้ว่า 16I/O line (Port0 และ Port2) ใช้สำหรับทำการเกี่ยวกับบัส (Bus) ระหว่างการ fetch คำสั่ง จาก external program menory โดยที่ Porto จะ multiplex เอาสัญญาณ address และ data bus ออกมา ถ้ากระทำตัวแบบ address bus ก็จะปล่อยส่วนของ program Counter (PCL) ที่ไบต์ต่ำออกมา แล้วก็จะเกิดสถานะลอย (float state) เพื่อรอโค๊ตคำสั่ง [Code byte] จากส่วนของ program menory

และในระหว่างที่ไบต์ต่ำของส่วนโปรแกรมเคาร์เตอร์ (PC) คำสั่งมีสถานะชัดเจน (Valid) จะมีสัญญาณ ALE (Address latch enable) ไป demultipex เอา address latch เอาไว้ ในขณะที่พอร์ต 3 (P2) จะปล่อย address ไบต์สูงของโปรแกรมเคาน์เตอร์ออกมา (PCH) และเมื่อมีสัญญาจาก PSEN (program strobe enable) ไปกระตุ้น EPROM ให้ปล่อยโค๊ตคำสั่งออกมาให้ไมโครคอนโทรลเลอร์

แอดเดรส ของ program memory จะเป็นแบบ 16 บิตเสมอถึงแม้จะถูกตัวต่ำกว่า 16 Kbyte ก็ตาม

โดยที่การกระทำคำสั่งกับ external program พอร์ต 8 บิต 2 พอร์ต จะถูกนำมาใช้เสมอในการอ้างแอดเดรส คือ พอร์ต 0 และ พอร์ต 1

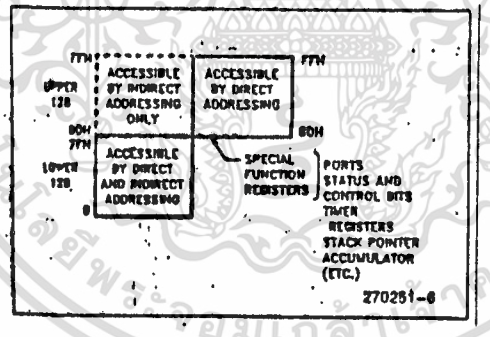


รูปที่ 3

รูปที่ 3 ภายใน 8051 นั้น มีหน่วยความจำถาวร ROM อยู่ภายใน

แล้ว ฉะนั้นจะ fetch คำสั่ง ภายในตัวมันเองเลย โดยจะเริ่มจาก 0000H ภายใน Rom ที่อยู่ในตัวมันเองสัญญาที่จะบอกว่า fetch ภายในหรือภายนอกก็คือ EA (External Access) ถ้า EA-0 fetch ให้ external คำสั่งทั้งหมด program memory ถ้า EA=1 ให้ fetch คำสั่งเริ่มแรกที่ ภายในตัวมันเอง ส่วนภายนอกจะต่อกับหน่วยความจำชั่วคราว (RAM) ขนาด 2 Kbyte โดยใช้แอดเดรสไบต์สูงเลือก Page เลือกได้ 8 page page ละ 256 byte ได้ขนาด 256 x 8 = 2048 bytes = 2 Kbyte นั่นเอง ส่วนบิตที่เหลือของ P2 ก็สามารถใช้เป็นอินพุต และเอาท์พุตได้อีกด้วย สัญญาที่ใช้ควบคุมในการอ่านหน่วยความจำชั่วคราว คือสัญญา RD และ WR อันเป็น บิตหนึ่งของพอร์ต P3 ส่วนพอร์ตที่เหลือก็สามารถนำไปใช้เป็น I/O ได้อีก

เราสามารถขยายหน่วยความจำ ได้สูงเป็น 64 Kbyte ก็อ้างแอดเดรสของ external data memory สามารถแยกเป็นแบบ 1 byte และ 2 byte ได้ กรณี 1 ไบต์ ใช้ กรณีแบ่งเป็นเพจ (page) ให้ RAM memory กรณี 2 ไบต์ คือ ต้องการปล่อย address byte สูงออกไปยัง port P2



รูปที่ 4 internal data memory

จากรูปที่ 4 ภายในของ Internal date memory จะถูกแบ่งออกเป็น 3 บล๊อก (3 block)

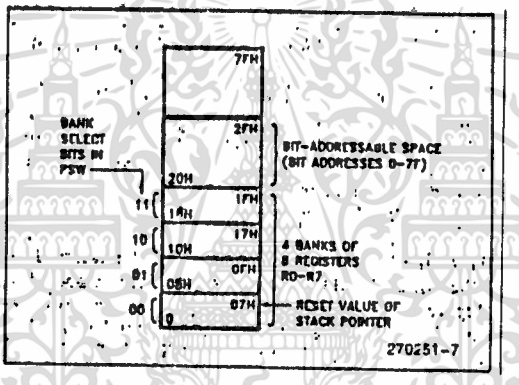
1. Lower 128 (บล๊อกต่ำ)
2. Upper 128 (บล๊อกสูง)
3. SFR Space

จำนวนไบต์ที่อ้าง Internal data memory หมายถึง แอดเดรสใช้เพียง 1 ไบต์ ก็เพียงพอ ซึ่งอ้างได้ 256 ไบต์เท่านั้น แต่อย่างไรก็ตามในความเป็นจริงสามารถอ้างได้ถึง 256 + 128 = 384 ไบต์ โดยใช้การทริค

(Simple trick)

จากรูปที่ 4 จะเห็นว่า 80-FFH จะ 2 blocks ส่วนแรกอ้างแอดเดรสไม่โดยตรงกับอีกส่วนอ้างแบบโดยตรง ถึงแม้ว่าบอกว่า Upper128 และ SFR จำอยู่ที่แอดเดรสเดียวกันก็ตาม แต่จริง ๆ ทางกายภาพแยกกันโดยเด็ดขาด (physically separate entitics)

ในรูปที่ 5 32 ไบต์ต่ำสุดจะถูกแบ่งเป็นกลุ่มได้ 4 กลุ่ม (4 banks) แต่ละกลุ่มจะมี 8 ไบต์ แต่ละไบต์เราจะมองเหมือน Register 1 ตัว ฉนั้น แต่ละกลุ่มหรือแบงค์ จะมี Register 8 ตัว (RO-R7) โดยที่โปรแกรมจะนำรีจีสเตอร์เหล่านี้ไปใช้



รูปที่ 5 Lower 128 bytes of internal RAM

เราสามารถเลือกใช้รีจีสเตอร์แต่ละแบงค์ได้ โดยไป Set บิตของ PSW (program status word) บิตที่ใช้เลือกแบ๊งค์ คือ บิตที่ 3 และบิตที่ 4 โดยที่ PSW จะเป็น Registor ที่อยู่ในพื้นที่ของ SFR space แอดเดรสที่ ODOH

CY	AC	FO	RSI	RSO	OV		P

bit1 bit0

Select bank

สำหรับ 16 ไบต์ เหนือขึ้นไปจากรีจีสเตอร์แบ๊งค์ จะเป็นรูปแบบของการอ้างถึงแบบบิตได้ (bit-addressable memory space) ในเซ็ตคำสั่งของ MCS-51 จะรวมคำสั่งที่กระทำแบบบิตได้ เอาไว้ด้วย 16 ไบต์ จะมีทั้งหมด 16x8 = 128 บิต สามารถอ้างได้โดยตรงแบบบิต โดยที่ตำแหน่งแอดเดรสของบิตเริ่มที่ OOH จนถึง 7FH

ในรูปที่ 5 Lower 128 สามารถอ้างทั้งที่เป็นแบบ direct และ indirect ได้

การอธิบายส่วน hardware ในช่วงนี้จะอธิบายถึง

- Port drivers การทำงานของ Port, สำหรับ Port0 และ Port2 ในการกระทำแบบ bus operation
- ส่วนของ Timer/Counters
- Serial Interface
- Interrupt System
- Reset
- The Reduced Power Modes in the CHMOS devices
- The EPROM versions of the 8051 AH, 8052A#, และ 80C51BH

จากตารางที่ 1 บอกว่า เบอร์ของไมโครคอนโทรลเวอร์ นั้นที่แสดงให้ตารางอยู่ในตระกูล MCS-51 ทั้งหมด แต่ละเบอร์จะมีข้อปลีกย่อยต่างกันไบต์เบอร์ 8051 โดยทั่วไปก็เช่น 8051, 8051AH, 80C51BH คำสั่งที่ใช้กับไมโครคอนโทรลเลอร์เหมือนกันแต่จะแตกต่างกันบ้างเล็กน้อยที่สถาปัตยกรรมภายใน เช่น 8051 มี ROM ภายใน 4K RAM 12K ไบต์ มี Timer แบบ 16 bit ให้ใช้ได้ 2 ตัว แต่ถ้าเป็น 8031 จะไม่มี Rom ภายในตัวมัน ต้องต่อออกมาใช้ที่ภายนอกเท่านั้น

ACC คือ Accumulator register เป็น nmemonics สำหรับ Accumulator คำสั่งเฉพาะ เขียนง่าย ๆ ว่า "A"

B register

รีจีสเตอร์ B จะใช้สำหรับกระทำการคูณ และหาร และใช้ได้ในการกรณีทั่วไปเหมือนรีจีสเตอร์ตัวอื่น ๆ

Program Status Word

เขียนย่อ ๆ เอาไว้ PSW PSW รีจีสเตอร์จะเก็บข้อมูลสถานะของโปรแกรม

STACK POINTER

สแต็ค พอยเตอร์ รีจีสเตอร์ (Stack Pointer Register) มีความกว้าง 8 บิต สแตคพอยเตอร์จะเพิ่มค่าก่อนที่จะเก็บข้อมูลลงไป ในเมื่อใช้คำสั่ง PUSH แล้วถึงค่อย CALL executions ซึ่ง Stack เราสามารถกำหนดให้อยู่บนส่วนใด ๆ ของ RAM ก็ได้ แต่ค่าเริ่มต้นของ Stack Printer จะอยู่ที่ 07H หลักจากที่เกิดการ Reset เป็นสาเหตุให้ Stack จะเริ่มที่แอดเดรส 08H

DATA POINTER

เขียนย่อ ๆ DTPR จะประกอบด้วย ไบต์สูง (DPH) และไบต์ต่ำ (DPL) ดาต้าพอยเตอร์นี้จะเก็บแอดเดรส 16 บิต เอาไว้ สามารถกำหนดเป็นได้ทั้งรีจีสเตอร์แบบ 16 บิต และรีจีตเตอร์ 8 บิต แยกออกจากกันก็ได้

Ports 0 ถึง 3

พอร์ต 0,1,2,3 จะอยู่ในส่วนของ SFR แบบ latch ของพอร์ต P0, P1, P2, P3 ตามลำดับ

Serial data Buffer (SBUF)

ซีเรียส ดาต้า บัฟเฟอร์ จะประกอบด้วยรีจีสเตอร์ 2 ตัวแยกจากกัน ส่วนหนึ่งจะทำหน้าที่เป็น transmit buffer Register และอีกส่วนหนึ่งทำหน้าที่เป็น receive buffer register

เมื่อเรา move ข้อมูลไปที่ SBUF ข้อมูลจะไปอยู่ที่

transmit-buffer ซึ่งจะเป็นส่วน Serial transmission (Moving a byte to SBUF is What initiatcs the transmission)

เมื่อ data ถูก mov มาจาก SBUF ข้อมูลจะมาเก็บไว้ที่ receive butter

Timer Registers

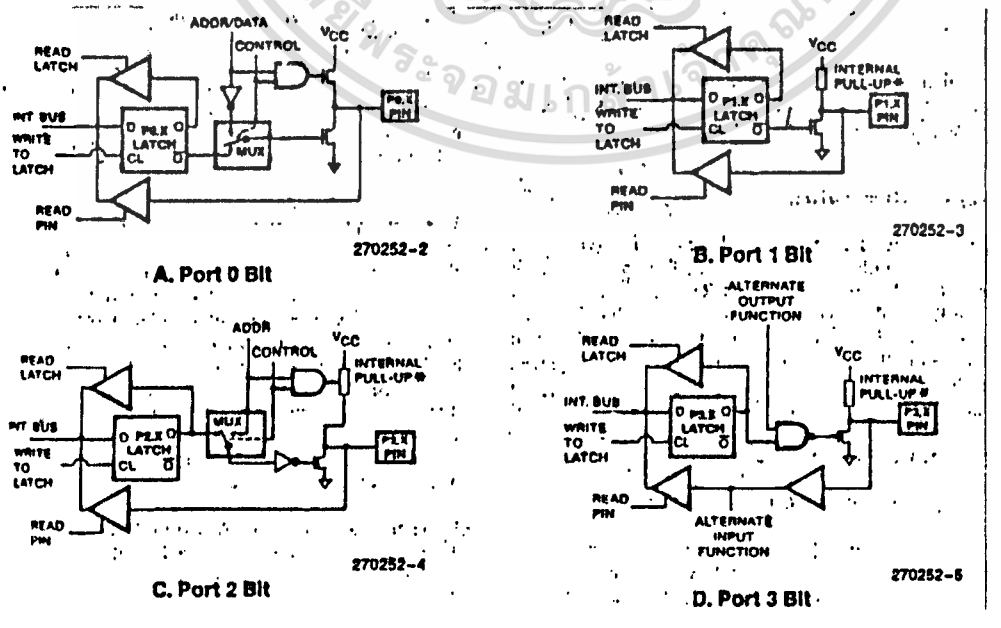
เป็นรีจีสเตอร์คู่ (register pairs) (TH0, TL0), (TH1, TL1) และ (TH2, TL2) เป็นรีจีสเตอร์นับ 16 บิต สำหรับ Timer/Counter 0, 1 และ 2 ตามลำดับ

CAPTURE Registers

เป็นรีจีสเตอร์คู่ (RCAP2H, RCAP2L) เป็น Capture รีจีสเตอร์สำหรับ Time2 ใน "Capture Mode" ใน Mode นี้ จะมีผลใน 8052 ที่ขา T2EX, โดยที่ TH2 และ TL2 จะถูกก๊อปปี้ไว้ใน RCAP2H และ RCAP2L ซึ่ง Timer2 จะมี 16 บิต auto-reload mode โดยที่ RCAP2H และ RCAP2L จะเก็บค่า reload เอาไว้ จะอธิบายละเอียดหัวข้อถัดไป

CONTROL REGISTERS

เป็น Special Function Registers IP, IE, TMOD, TOON T2CON, SCON และ PCON ประกอบด้วยคอนโทลบิตและสถานะบิตเอาไว้สำหรับ interrupt System, Time/Counters, และ serial port



(8051 Port bit latches and I/O Buffers)

Port Structure and Operation

ทั้ง 4 พอร์ต ใน 8051 เป็นแบบ bidirectional คือ ทิศทางของข้อมูลไหลได้ ทั้ง 2 ทิศทาง ส่งและรับได้ และสามารถเก็บเก็บข้อมูล (latch) ไว้ได้ (Special Function Registers PO-P3), มี Output driver, และมี input buffer

Output drive ของ พอร์ต PO และ P2 และ input buffer ของพอร์ต PO ใช้สำหรับอ้างถึงส่วน external menmory ในการนำไปประยุกต์ใช้งาน PO Output เป็นแอดเดรสไบต์ต่ำ ของ external memory และใช้ Time-multi-plexed กับข้อมูลที่อาจอ่านหรือเขียนด้วย

P2 Output เป็นแอดเดรสไบต์สูงของส่วน external memory เมื่อมีการอ้างแอดเดรสแบบ 16 บิต นอกเหนือจากติดต่อกับ external memory แล้ว P2 จะทำหน้าที่เป็นพอร์ตธรรมดา นั่นคือจะเก็บข้อมูลส่งออกมา ของ Port2, (ที่เก็บอยู่ใน SFR content)

(Port 3) P3 ทุกขา เป็นแบบ multifuntional คือ นอกจากจะทำหน้าที่เป็นแบบพอร์ตธรรมดาแล้ว ยังมีหน้าที่พิเศษอย่างอื่น ๆ อีก (ซึ่งรวมทั้ง Port 1,0 และ Port 1.1 ด้วย) ดังนี้ (สำหรับ P1.0, P1. 18052 เท่านั้น)

Port Pin	Alternate Function
P1.0	T2 (Timer/Counter 2 external input)
P1.1	T2EX (Timer/Counter2 CAPTURE/Reload-triggle

(P.1.0, P1.1 สำหรับ 8052 เท่านั้น)

Port Pin	Alternate Function
P3,0	RXD (serial input Port)
P3.1	TXD (serial output Port)
P3.2	INTO (external interrupt)
P3.3	INTI (exteranl interrupt)
P3.4	TO (Timer/Counter0 external input)
P3.5	TI (Timer/Counter1 external input)
P3.6	WR (external data Memory Write strobe)
P3.7	RD (external data Memory Read Strobe)

I/O configuration

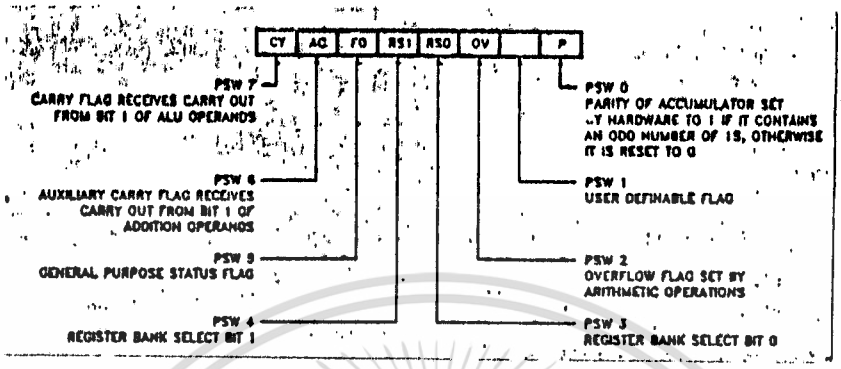
จากรูปที่ 1 แสดง functional diagram ของ bit latch และ I/O buffer ใน แต่ละพอร์ต, บิตแลนท์ (บิตหนึ่งในหนึ่งพอร์ตของ SFR) แสดงในรูปแบบของ D-FLIP FLOP

3.2 THE MCS-51 INSTRUCTION SET

ไมโครโปรเซสเซอร์ตระกูล MCS-51 ทุกเบอร์จะใช้ชุดคำสั่งเหมือนกันหมด เนื่องจากชุดคำสั่งได้ถูกจัดให้มีความเหมาะสม (optimize) สำหรับ 8 บิต คอนโทรล (8-bit control application) มีการอ้างข้อมูลได้อย่างรวดเร็ว เนื่องด้วนมีหน่วยความจำชั่วคราวอยู่ภายใน (interanl RAM) สำหรับ 8051 มี 25 byte สำหรับกรณี 8031 (Romless Version) มีเพียง 128 ไบต์ แต่ก็เพียงพอสำหรับข้อมูลขนาดเล็ก ๆ ได้ นอกจากนี้ยังมีคำสั่งสนับสนุนการอ้างพอร์ตภายนอกแบบ 1 บิตด้วย ซึ่งถูกออกแบบมาสำหรับงานควบคุม ซึ่งใช้หลักการทางลอจิก หรือเรียกอีกแบบว่า Boolean processing ก็ได้

Program Status Word

Program Status word (PSW) จะเก็บสถานะบิตที่ตอบสนองกับความเป็นไปในขณะนั้นของ CPU ดังรูปที่ 8



รูปที่ 8

ซึ่ง PSW จะอยู่ในส่วนของ SFR (Special Function Reg.) RSW ซึ่งประกอบไปด้วย Carry Bit, Auxilary Carry (for BCD operation and two user definable status flags)

Carry bit นอกจากจะใช้กระทำเกี่ยวกับการกระทำทางคณิตศาสตร์แล้ว (Carithematic function) ยังทำหน้าที่เป็น เสมือน Accumulator" สำหรับการกระทำ ทางลอจิก อีกด้วย (Boolean operations)

RS0 และ RS1 เป็นตัวเลือก bank ของ register (R0 R7) เลือกได้ทั้งหมด 4 bank นั่นคือ มี R0-R7 อยู่ 4 ชุด แยกออกจากกันได้ 4 bank โดยการ Set RS0 และ RSI ดูรูปที่ 6

Parity bit จะมีผลต่อลอจิก 1 (15) ใน Accumulator ถ้าใน Accumulator มีลอจิก 1 เป็นคี่ P=1, มีลอจิก 1 เป็นจำนวนครบคู่ที่ให้ P=0 อาจกล่าวว่า จำนวนลอจิกใน Accumalator ที่มีลอจิก 1 เมื่อรวมกับ Parity bit แล้วจะต้องเป็นเลขคู่เสมอ และอีก 2 bit ของ PSW ถูกใช้สำหรับกรณีทั่วไป

Addressing Modes

ลักษณะการอ้างแอดเดรส โดยการใช้ชุดคำสั่งของ MCS-51 มีดังนี้

- แบบ Direct Addressing (แบบอ้างโดยตรง)

การอ้างแบบโดยตรง ถูกกำหนดแอดเดรสในช่วง 8 bit เท่านั้น คือ สามารถอ้างได้ภายใน interanl RAM และในส่วน FSR เท่านั้น โดยการอ้างแบบตรงเท่านั้น

- แบบ Indirect-Addressing

การออกแบบโดยอ้อม โดยเราจะใช้ Register กับแอดเดรสของข้อมูลอีกทีหนึ่ง ก่อนที่เราจะอ้างถึงซึ่งส่วนที่เราสามารถอ้างแบบอ้อมได้ ก็มีส่วน Internal RAM และ External (memory) ทั้งหมดโดยเราจะใช้ R0 หรือ R1 ที่มาจากการเลือกแบงค์แล้ว สำหรับ การรีอ้างแอดเดรส 8 บิต, หรือ Stack Pointer ด้วย

แต่กรณีจะอ้างแอดเดรส แบบ 16 bit สามารถใช้ Register 16 บิต data pointer", DPTR ได้เท่านั้น

คำสั่ง Register (Register Instructions)

ใน Register bank, จะประกอบด้วย R0-R7 การอ้างถึงคำสั่งโดยผ่านทางรีจิสเตอร์ นั้น ทำให้การทำงานมีประสิทธิภาพขึ้น ตัวปัญหา เรื่อง addressbyte เมื่อมีการ execute คำสั่ง 1 ใน 8 ของรีจีสเตอร์ในแบงค์ที่เราเลือกใช้ จะถูกอ้างถึง เราสามารถเลือกแบงค์ได้โดย Set bank Setect ของ bank ที่ PSW

Register-Specific instructions

ในบางคำสั่งจะถูกกำหนดให้กระทำทางรีจีสเตอร์ที่แน่นอนไปเลย เช่น ในบางคำสั่งจะให้กระทำเฉพาะใน Accumulator เท่านั้น ในบางคำสั่งใช้เฉพาะ data pointer เป็นต้นในคำสั่งจะอ้างถึง Accumulation ด้วยสัญญลักษณ์ "A" ใน assembler ซึ่งเป็น accumulator-specific opcodes

IMMEDIATE CONTANTS

จัดการข้อมูลที่เป็นค่าคงที่ MOV A, #100 คือ ให้ค่าในรีจีสเตอร์ A ด้วยค่าคงที่ 100 (ฐาน 10) หรือเท่ากับ 64H ในฐาน 16

INDEXED Addressing

เฉพาะในส่วนของ Program memory เท่านั้น ที่สามารถอ้างแอดเดรสแบบ indexed ได้ (ตัวแบบตัวชี้) และอ่านได้อย่างเดียวเท่านั้น

โดยการใช้อ่านค่าจากตารางข้อมูล โดยจะกำหนดให้ 16-bit base register

มีอยู่ 2 ตัว คือ

- DPTR
- Program Counter

จะถูกกำหนดให้เป็นฐานของตาราง (base table) แล้วจะกำหนดรีจีสเตอร์ A ถูก Set ให้เลื่อนไปเลื่อนมา เพื่อข้อมูลในตาราง

เพื่อข้อมูลในตาราง

หลักการ : การอ่านข้อมูล (อ่าน) ใน Program memory โดยกำหนด

DDTR, Program Counter เป็นฐานของตาราง และบอกค่ากับรีจีสเตอร์เข้าไปกับ base table ก็จะได้ address ของตารางใน table ในส่วนของ Program memory ออกมา

ยังมีอีกกรณีในการตัวแบบ ตัวชี้ คือ กรณี "case Jump" ในชุดคำสั่ง โดยที่ค่าที่ปลายทางจะถูกต้องโดยการบอกค่าในรีจีสเตอร์ A กับ base pointer (จะได้ address นั้น)

ชุดคำสั่ง คณิตศาสตร์ (Arithmetic Intructions)

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ADD A,<byte>	A = A + <byte>	X	X	X	X	1
ADDC A,<byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A,<byte>	A = A − <byte> − C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A − 1	Accumulator only				1
DEC <byte>	<byte> = <byte> − 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

ชุดในคำสั่งดังในตารางที่ 1 จะบอกแอดเดรส mode ต่าง ๆ ดังในตัวอย่าง ตัวอย่าง

```
ADD A,7FH   (A)  (A) + (7H)     (direct addressing)
ADD A,@RO   (A)  (A) + ((RO))  (indirect addressing)
ADD A,R7    (A)  (A) + (R7)    (register addressing)
ADD A,#127  (A)  (A) + 127     (immediate constant)
```

ในตารางที่ 1 สมมติให้ในวงจร MCS-51 ใช้ clock ความถี่ 12 MHz แล้ว จะได้ว่าทุก ๆ คำสั่ง ในตาราง จะใช้เวลาในการกระทำคำสั่งนั้น ๆ เป็นเวลา 1 MS ยกเว้นคำสั่ง INC DPTR ใช้เวลา 2 usec

Note : ที่ไบต์ใด ๆ ในส่วนของ Internal data memory space เราสามารถที่จะเพิ่มหรือลดค่าภายในตัวมันเองโดยไม่ต้องผ่าน Accumulator

คำสั่ง INC บน 16 bit data pointer, DPTR ใช้สำหรับ การอ้างแอดเดรสแบบ 16 bit ใช้กรณี external memory ฉนั้นการอ้างเพิ่มแบบเป็นการอ้างที่ให้ความสะดวก

คำสั่ง MUL AB เป็นการหาผลคูณระหว่าง ข้อมูลใน Accumulator กับ รีจีสเตอร์ B จะได้ผลลัพธ์เป็นแบบ 16 บิต ไว้ที่รีจีสเตอร์ B และ A (Accumulator)

คำสั่ง DIV AB จะหาร Accumulator ด้วยข้อมูลในรีจีสเตอร์ B จะเก็บผลที่ได้ (QUOTIENT) ใน Accumulator ส่วนที่เหลือจากผลหาร (remainder) เก็บในรีจีสเตอรี B การหาอาจทำได้อีกแบบ โดย อาศัยหลักการ Strift ไปทางขวาจะหารด้วย 2 ไปเรื่อย n ครั้ง ก็เท่ากับตัวหาร 2n คำสั่ง Dir AB จะใช้เวลา 4 โดยที่ bit ที่ถูก stift ไปทางขวาจะเก็บอยู่ที่ registor B (remainder นั่นเอง)

คำสั่ง DAA สำหรับ BCD arithmetic operation

ในการกระทำทางคณิตศาสตร์แบบ BCD นั้น ถ้ามีการใช้ ADD หรือคำสั่ง ADDC มักมีคำสั่ง DAA ตามมาด้วยเสมอ

NOTE : DAA ไม่ได้ convert จาก binary number ไปเป็น BCD แต่ DAA มีประโยชน์ หลักการที่เกิดการบวกกันของ BCD 2 จำนวนก่อน (First Step)

* คำสั่งทางลอจิก LOGICAL instructions

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
ANL A,<byte>	A = A .AND. <byte>	X	X	X	X	1
ANL <byte>,A	<byte> = <byte> .AND. A	X				1
ANL <byte>,#data	<byte> = <byte> ,AND. #data	X				2
ORL A,<byte>	A = A .OR. <byte>	X	X	X	X	1
ORL <byte>,A	<byte> = <byte> .OR. A	X				1
ORL <byte>,#data	<byte> = <byte> .OR. #data	X				2
XRL A,<byte>	A = A .XOR. <byte>	X	X	X	X	1
XRL <byte>,A	<byte> = <byte> .XOR. A	X				1
XRL <byte>,#data	<byte> = <byte> .XOR. #data	X				2
CRL A	A = 00H	Accumulator only				1
CPL A	A = .NOT. A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

จากตารางที่ 2 แสดงคำสั่งทางลอจิกของ MCS-51 เป็นการกระทำบูลลีน (Boolean Operations) AND, OR, Exceccusive OR, NOT

ตัวอย่างถ้า (A) = (00110101 B)

<byte> = (01010011 B)

ANL A, <byte>

ผลที่ได้ เมื่อดูผลในรีจีสเตอร์ A ได้ผลลัพทธ์ดังนี้ (A) = (00010001 B)

ANK A, <byte>

การใช้คำสั่งนั้น สามารถเข้าออก <byte> ได้หลายแบบเช่น

```
ANL A, 7FH      (direct addressing)
ANL A, @R1      (indirect addressing)
ANL A, R6       (Registor addressing)
ANL A, #53H     (immediate constant)
```

NOTE : การกระทำแบบบูลลีน (Boolean operations) สามารถกระทำบนไบต์ใด ๆ ใน Internal Data memory โดยไม่ต้องผ่านรีจีสเตอร์ A ก็ได้ WRL <byte>, #data ทำให้รวมเร็ว

XRL P1, #0FFH

ถ้า opertion นั้น ตอบสนองตาม interupt, อย่าใช้ Accumulator เก็บ (Save time) ควรเก็บค่าต่าง ๆ ลงใน Stack สำหรับการบริหารโปรแกรมย่อย

คำสั่ง Rotate RLA, RLC A เป็นการเลื่อน Accumulator ไป 1 บิต ทางซ้าย (หรือขวา) กรณี left บิตสูงจะเลื่อนวนไปทางบิตต่ำ ส่วนบิตต่ำเลื่อนไปแทนบิตสูง

คำสั่ง SWAP A เป็นคำสั่งทำหน้าที่สลับ 4 บิตสูง กับ 4 บิตต่ำ ของ Accumulator มีประโยชน์กรณีกระทำแบบ BCD

ตัวอย่าง

ถ้า Accumlator เก็บค่า binary number ซึ่งมีค่าน้อยกว่า 100 เราสามารถแปลง เป็น BCD ได้ดังนี้

MOV B, #19

DIV AB

SWAP A

ADD A,B

การหารด้วย 10 จะให้ผลอยู่ที่ 4 บิตล่างของ Accumulator ขณะที่เศษที่เหลืออยู่ที่ regster B หลังจากนั้นก็สลับ 4 bit สูงกับ 4 บิตต่ำของ Accumulator แล้วก็นำผลจาก Register B มาบวก จะได้ รูปแบบของ (binary) HEX BCD

เช่น ใน register A (A) = (20H) Hex

(A) = (32) decimale

การโอนการย้ายข้อมูล Data Transfers

การโอนย้ายข้อมูล ภายใน Internal Ram

ตารางที่ 3

A list of the MCS-5i data transfer Instructions that Access internal data memory Space

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
MOV A,<src>	A = <src>	X	X	X	X	1
MOV <dest>,A	<dest> = A	X	X	X		1
MOV <dest>, <src>	<dest> = <src>	X	X	X	X	2
MOV DPTR, #data16	DPTR = 16-bit immediate constant.				X	2
PUSH <src>	INC SP : MOV "@SP", <src>	X				2
POP <dest>	MOV <dest>, "@SP" : DEC SP	X				2
XCH A,<byte>	ACC and <byte> exchange data	X	X	X		1
XCHD A,@Ri	ACC and @Ri exchange low nibbles		X			1

จากตารางที่ 3 แสดงชุดคำสั่งที่ทำหน้าที่โอนย้ายข้อมูลภายใน internal memory

คำสั่ง MOV <dest>, <src> อนุญาติให้มีการถอนย้ายข้อมูลระหว่าง internal RAM หรือ SFR โดยไม่ผ่านรีจีสเตอร์ A ได้

ข้อควรคำนึงไว้เสมอว่า 128 ไบต์บนของ data Ram อ้างถึงเฉพาะ

แบบ indirect เท่านั้น และในส่วน SFR on ได้แบบ direct เท่านั้น

NOTE : ในทุก ๆ เบอร์ตระกูล MCS-51 ไมโครโปรเซอเซอร์นั้น สแตค (stack) จะอยู่ภายในชิพของหน่วยความจำภายในอยู่แล้ว RAM ทุกครั้งที่ใช้จะมีการเพิ่มค่า (grows upwards)

- คำสั่ง PUSH ทุกครั้งที่มีการใช้คำสั่งนี้ มีการกระทำดังนี้เกิดขึ้น

1. เพิ่มค่า Stack Pointer (SP)
2. เอาข้อมูลจาก (Src) ไปใช้ใน Stack

- คำสั่ง POP

1. เอาข้อมูลจาก Stack ไปเก็บไว้ที่ <lest>
2. ลดค่าใน Stack Pointer (SP)

คำสั่ง PUSH, POP ใช้กรณีแบบ direct เท่านั้น เพื่อแสดงความแน่ใจว่าเป็นข้อมูลที่เรา save หรือ restord เอาไว้ แต่ตัว SP เอง สามารถอ้างแอดเดรสแบบ indirect ได้โดยใช้ SP register เราสามารถกำหนดให้ใช้ 128 ไบต์บนได้ ด้วยการเขียนคำสั่ง แต่ต้องไม่ใช่ ในส่วนของ SFR space

128 ไบต์บนไม่สามารถถูกใช้ได้บน 8051, 8051AH, หรือ 80C51BH รวมทั้ง Romless หรือ EPROM Version

ถ้าใช้ SP ชี้ไปที่ 128 ไบต์บน ถ้าใช้ PUSH ข้อมูลจะหาย ถ้า POP ก็จะได้ข้อมูลที่ไม่ต้องการออกมา

การโอนย้ายข้อมูลรวม 16 บิต addr. Mov เอาไว้ด้วย โดยเริ่มต้น DPTR ให้ชี้ไปที่ส่วนของ TABLE ของ Program Memory หรือ 16 บิต ในส่วน external Data Memory

คำสั่ง XCH A, <byte> ทำให้เกิดการเปลี่ยนข้อมูลระหว่างกัน คือ ระหว่างรีจีสเตอร์ A กับ <byte> ส่วน XCHD A, CRi ก็คล้าย ๆ กัน แต่จะเปลี่ยนข้อมูลระหว่างกัน แค่ 4 บิตล่าง เท่านั้น

อย่างแรก ปัญหาการชีพ (เลื่อน) 8 bit BCD (ตัวเลข 2 ตัว) ไปทางขวาตามรูปที่ 9 ซึ่งจะเปรียบเทียบกับการใช้คำสั่ง MOV_ และ XCH_ ซึ่งผลสุดท้ายจะได้บรรทัดสุดท้าย ในการเลื่อนเหมือนกัน แต่การใช้ XCH จะประหยัดจำนวนไบต์คำสั่ง และใช้เวลาน้อยกว่า ถ้าใช้คำสั่ง MOV จะเสียเวลา 9 uSec

ถ้า XCH เสียเวลา 5uSec เมื่อเทียบกับสัญญาณนาฬิกา 12 MH_

ประโยชน์ของ XCH กับ XCHD ดังในรูปที่ 9

	2A	2B	2C	2D	2E	ACC
MOV A,2EH	00	12	34	56	78	78
MOV 2EH,2DH	00	12	34	56	56	78
MOV 2DH,2CH	00	12	34	34	56	78
MOV 2CH,2BH	00	12	12	34	56	78
MOV 2BH,#0	00	00	12	34	56	78

(a) Using direct MOVs: 14 bytes, 9 μs

	2A	2B	2C	2D	2E	ACC
CLR A	00	12	34	56	78	00
XCH A,2BH	00	00	34	56	78	12
XCH A,2CH	00	00	12	56	78	34
XCH A,2DH	00	00	12	34	78	56
XCH A,2EH	00	00	12	34	56	78

(b) Using XCHs: 9 bytes, 5 μs

รูปที่ 9 Shifting = BCD Number Two Ligits to the right

ตัวอย่าง การใช้คำสั่งเลื่อนตัวเลย BCD nember ไปทางขวา 1 ตัวเลข โดยใช้คำสั่ง XCHD ดังรูปที่ 10

		2A	2B	2C	2D	2E	ACC
	MOV R1,#2EH	00	12	34	56	78	XX
	MOV R0,#2DH	00	12	34	56	78	XX
	loop for R1 = 2EH:						
LOOP:	MOV A,@R1	00	12	34	56	78	78
	XCHD A,@R0	00	12	34	56	78	76
	SWAP A	00	12	34	56	78	67
	MOV @R1,A	00	12	34	56	67	67
	DEC R1	00	12	34	56	67	67
	DEC R0	00	12	34	56	67	67
	CJNE R1,#2AH,LOOP						
	loop for R1 = 2DH:	00	12	38	45	67	45
	loop for R1 = 2CH:	00	18	23	45	67	23
	loop for R1 = 2BH:	08	01	23	45	67	01
	CLR A	08	01	23	45	67	00
	XCH A,2AH	00	01	23	45	67	08

จากรูปที่ 10 พ้อยเตอร์ (pointer) R1, และ R0 จะถูกเซ็ตให้ชี้ไปที่ 2 ไบต์ที่เก็บตัวเลขแบบ BCD แล้ว รูปแบบการเลื่อน จะให้ข้อมูลตัวเลขเลื่อนไปทางขวา 1 ตัว และให้ทางซ้ายสุดมีศูนย์มาแทนที่

การโอนย้ายของข้อมูลทาง External RAM

Address Width	Mnemonic	Operation	Execution Time (μs)
8 bits	MOVX A,@Ri	Read external RAM @Ri	2
8 bits	MOVX @Ri,A	Write external RAM @Ri	2
16 bits	MOVX A,@DPTR	Read external RAM @DPTR	2
16 bits	MOVX @DPTR,A	Write external RAM @DPTR	2

ตารางที่ 4

[A List of the MCS-51 Data Transfer Instructions that]

Access External Data Memory Space

External RAM

จากตารางที่ 4 แสดงชุดคำสั่งการโอนย้ายข้อมูลเฉพาะ external data memory RAM

เป็นการอ้างแอดเดรสแบบ indirect เท่านั้น ต้องการอ้างแบบ 8 บิตก็ใช้ Ri อาจจะเป็น R0 หรือไม่ก็ R1 ตัวใดตัวหนึ่ง สามารถเลือก Bank ที่เราต้องการได้

ถ้าเป็นการอ้างแบบ 16 (2 ไบต์), @DPTR

ข้อเสียของการอ้าง address แบบ 16 บิต คือต้องใช้ Port 2 ทั้งหมด

ถ้าใช้ Ram ภายนอกเพียงไม่กี่กิโลไบต์ (K byte) แต่ถ้าให้อ้างแบบ 8 บิต เหมือน

กรณีรูปที่ 3 ท่าให้ P2 เหลืออยู่ท่าให้นำไปใช้งานอื่น ๆ ได้อีก

NOTE : ในการอ้าง external DATA RAM access, Accumulalor จะถูกกำหนดให้ปลายทาง (destination) หรือไม่ก็ต้นทาง (source) ของข้อมูลเสมอ

สัญญาณ strobe ทั้งขา RD และขา WR จะตอบสนองและส่งสัญญาณ RD หรือ WR ออก ไปยัง external RAM จะมีต่อเมื่อใช้คำสั่ง MOVX

ซึ่งในกรณีที่ไม่มีการใช้สัญญาณทั้งสัญญาณนี้เลย ขาเหล่านี้จะถูกกำหนดเป็น

I/O line ได้ (หน้า 5-11)

LOOKUP TABLES

Mnemonic	Operation	Execution Time (μs)
MOVC A,@A+DPTR	Read Pgm Memory at (A+DPTR)	2
MOVC A,@A+PC	Read Pgm Memory at (A+PC)	2

ตารางที่ 5 Lookup table Read Instructions

จากตารางที่ 5 จะเห็นว่า มีคำสั่ง 2 คำสั่งที่จะสามารถอ่านค่าจากตารางในส่วน Program Memory ได้ เพียงถูกออกแบบมาให้อ้างถึงได้เฉพาะส่วน Program Memory ตารางสามารถอ่านได้เพียงอย่างเดียวและไม่สามารถแก้ไขได้ สัญญาลักษณ์ "MOVC" ก็คือ "MOVC CONSTNAT"

ถ้ามีการอ้างถึง external program memory สัญญาณ Read strede คือ PSEN จะ active

คำสั่ง MOVE คำสั่งแรกจากตาราง สามารถอ้างข้อมูลอยู่ในช่วง 0-255 ได้ทั้งหมด 256 ตำแหน่ง DPTR จะเก็บค่าของ data pointer เก็บค่าชี้ที่จุดเริ่มต้นของตาราง จำนวนของข้อมูลที่จะอ้างในตารางก็เก็บไว้ในรีจีสเตอร์ A ดังนี้

MOVE A. @A+DPTR

ข้อมูลที่ได้มาจะเก็บค่าไว้ในรีจีสเตอร์ A

ส่วนคำสั่ง MOVE คำสั่งที่สอง การทำงานเหมือนคำสั่งแรก ยกเว้น

ยกเว้น Program Counter (PC) จะถูกใช้เป็นฐานของตาราง และตารางจะถูกอ้าง ผ่านทางโปรแกรมย่อม (Subroutine)

ค่าแรกที่ต้องการ (ข้อมูล) จะถูกนำมาเก็บไว้ใน Accumulator ดังนี้

```
MOV A, ENTRY NUMBER
CALL TABLE
```

โปรแกรมย่อย TABLE คือ

```
TABLE : MOVC A, @A+PC
        RET
```

ซึ่งเห็นว่า TABLE จะจบด้วยคำสั่ง RET (return) ใน Program memory

การอ้างข้อมูลแบบนี้อ้างข้อมูลทั้งหมด (entrics) ได้ 255 entrics 1-255 entory ที่ 0 ใช้ไม่ได้ เพราะว่า เวลาที่คำสั่ง MOVE กำลังทำงาน PC จะเก็บตำแหน่งของคำสั่ง RET ฉนั้น entry numbers จะเป็นคำสั่ง RET จากตัวมันเอง

คำสั่ง บูลลีน Boolean Operations

MCS-51 ไมโครโปรเซอเซอร์ จำมีส่วนทำหน้าที่ Boolean (Single-bit) processor โดยที่ ภายใน Internal Ram จะอ้างข้อมูลได้ 128 address-bit และในส่วน SFR สามารถทำได้อีก 128 แอดเดรสบิตและทุกพอร์ต (0-3) สามารถอ้างข้อมูลได้แบบบิตแอดเดรส แยกออกจากกัน คำสั่งการอ้างแอดเดรสไม่ใช่สำหรับการกระโดดแบบเงื่อนไขเท่านั้น แต่ยังมีคำสั่งต่าง ๆ อีกเช่นคำสั่ง move, set, clear, complement, OR, และ AND

ชุดคำสั่งสำหรับบูลลีนโปรเซสเซอร์ แสดงในตารางที่ 6 ทุกบิตมีการอ้างถึงแบบ direct แอดเดรส บิตแอดเดรสตั้งแต่ 00H ถึง 7FH ในส่วนของ

128 ไบต์แรกของ Internal RAM และบิตแอดเดรสตั้งแต่ 80H-FFH ในส่วนของ SFR Space จากตัวอย่างเราสามารถนำค่า flag ภายในย้ายออกไปที่พอร์ต pin ได้แก่

```
MOV  C, FLAG

MOV  P1.0,C
```

อธิบายตัวอย่าง : FLAG เป็นชื่อของบิตแอดเดรสใด ๆ ในส่วน 128 ไบต์แรกของ RAM ภายใน หรือ SFR space, ในส่วน I/O (LSB ของ พอร์ต 1 ในตัวอย่าง) จะถูก set หรือ cleared ขึ้นอยู่กับบิตของแฟล็ค ว่าจะเป็นลอจิก 1 หรือ 0

ตารางที่ 6 A List of the MCS-51 Bookean Instructions

Mnemonic	Operation	Execution Time (μs)
ANL C;bit	C = C .AND. bit	2
ANL C,/bit	C = C .AND. .NOT. bit	2
ORL C,bit	C = C .OR. bit	2
ORL C,/bit	C = C .OR. .NOT. bit	2
MOV C,bit	C = bit	1
MOV bit,C	bit = C	2
CLR C	C = 0	1
CLR bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = .NOT. C	1
CPL bit	bit = .NOT. bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit,rel	Jump if bit = 1	2
JNB bit,rel	Jump if bit = 0	2
JBC bit,rel	Jump if bit = 1; CLR bit	2

ส่วน Carry bit ใน PSW จะถูกใช้เป็น Accumulator 1 บิต สำหรับการกระทำ แบบบูลลีน (Boolean) ของส่วน Boolean processor คำสั่งเกี่ยวกับบิตนั้น จะอ้างถึง Carry bit ด้วยตัว C เป็น Carry-Specific instructions (CLRC, etc), Carry bit ตัวแบบ

โดยตรงได้ด้วย เนื่องด้วยมันอยู่ภายใน PSW register ซึ่งมีการตัวแบบ bit addressable

NOTE : คำสั่งบูลลีน จะรวมเอา ANL และ ORL แต่ไม่น่า XRL มาใช้ซึ่ง XRL สามารถเขียนได้ด้วยโปรแกรม

ตัวอย่างถ้าเราต้องการรูปแบบการ Exclusive OR สำหรับ บิต 2 บิต

C = bit1. XRL. bit2

สามารถแทนด้วย ซอฟท์แวร์ดังนี้

```
        MOV     C, bit1
        JNB     bit2, OVER
        CPL     C
OVER : (Continued)
```

อธิบายตัวอย่าง : ขั้นแรก เอาบิต 1 ไปเก็บไว้ใน Carry ถ้าบิต 2=0 แล้ว C จะได้ค่าที่ถูกต้อง นั่นคือ bit1 XRL. XRL. bit2=bit1 ถ้า bit2=0 หรืออีกกรณีหนึ่งคือ ถ้า bit2-1, แล้ว C จะต้องคอมพลีเมนต์ก่อน จึงจะได้ที่ถูกต้อง

RELATIVE OFFSET

การกำหนดปลายทางในการกระโดดไปนั้น ตัวแอสเซมเบลอร์ จะกำหนดได้เป็น 2 แบบ ลาเบล (LABEL) หรือไม่ก็เป็น address จริง ๆ กันเลย อย่างไรก็ตาม การแปลงแอดเดรสปลายทางจะถูกแปลงเป็น Relative offset byte This is a signed (two's complement) offset byte which is added to the PC in two's complement arithemetic if the jump is executed

ช่วงของการกระโดดอยู่ในช่วง -128 ถึง +127 ใน Program memory โดยจะสัมพันธ์กับแอดเดรสของคำสั่งนั้น

(-128 to +127 program memory bytes relative to the

first byte following the instruction

Jump Instructions คำสั่งการอ้างกระโดด

Mnemonic	Operation	Execution Time (μs)
JMP addr	Jump to addr	2
JMP @A+DPTR	Jump to A+DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

ตารางที่ 7 Unconditional Jumps in MCS-51 Devices

ในตารางจะแสดง IMP คำสั่งเดียวเท่านั้น แต่ความจริงแล้วมีความสั่งอยู่ 3 คำสั่งคือ SJMP, LJMP, AJMP มีรูปแบบ (Format) ในการใช้กระโดดไปที่ปลายทางต่างกัน

คำสั่ง JMP เป็น Memonic ทั่วไป (เป็นตัวแทน) กลุ่มคำสั่ง JMP ซึ่งสามารถใช้ได้ถ้าโปรแกรมเมอร์ ไม่สนใจว่าจะกระโดดไปทางทิศทางไหน ช่วงไหน

คำสั่ง SJMP กำหนดช่วงปลายทางของแอดเดรสในช่วง relative offset ความยาวของ คำสั่ง 2 ไบต์ ประกอบด้วย opcode และ relative offset byte ช่วงการกระโดด -128 to +127 bytes Relative ที่ตามหลังคำสั่ง SJMP คำสั่ง LJMP กำหนดช่วงแอดเดรส 16 bit ความยาวของคำสั่ง 3 ไบต์ คือ opcode 1 ไบต์ และ 2 ไบต์ สำหรับแอดเดรส กระโดดได้ในช่วง 64 K ที่ใดก็ได้ในช่วง 64 Kbytes นี้ ใน Program memory Spece

คำสั่ง AJMP กำหนดช่วงปลายทางด้วย 11 บิต แอดเดรส ความยาวคำสั่ง 2 ไบต์ ประกอบไปด้วย opcode ซึ่งจะเป็น 3 บิต บนของแอดเดรสไว้ด้วย และอีก 1 ไบต์จะเก็บ 8 บิตไบต์ต่ำของแอดเดรสที่จะก้าวกระโดดไป เมื่อคำสั่งถูก executed 11 บิต แอดเดรส ที่เก็บเอาไว้นี้จะเอาไปแทนที่ใน 11 บิต ใน PC 5 บิต สูงของ PC ยังคงเหมือนเดิม ซึ่งคำสั่ง AJMP จะใช้ก้าวกระโดดภายในช่วง 2K เท่านั้น

ในทุกกรณี (The all case) โปรแกรมเมอร์กำหนดแอดเดรสปลายทางในแอสเซมเบอร์วิธีที่เหมือนกันได้เลย คืออาจจะเป็น Lable หรือไม่ก็ 16-bit constant แอสเซมเบอร์ก็จะใส่ค่าแอดเดรสปลายทางให้ในรูปแบบ (Format) ที่ถูกต้องให้เลย แต่ถ้าเรากำหนด Format ไม่ถูกต้องในการกระโดยไปที่แอดเดรสปลายทางให้ในรูปแบบ (Format) ที่ถูกต้องให้เลย แต่ถ้าเรากำหนด Format ไม่ถูกต้องในการกระโดดไปที่แอดเดรสปลายทาง

ข้อความ "Destination out of range" จะถูกแต่เอาไว้ที่ List file

คำสั่ง JMP @A+DPTR มีไว้สนับสนุนการกระโดดแบบมีเงื่อนไข หรือกรณี (case) แอดเดรสปลายทางจะถูกคำนวณเมื่อมีการ excuted คำสั่ง โดยแอดเดรสปลายทางมาจากผลบวกของ 16 บิต DPTR รีจิสเตอร์ และแอคคิวมูเลเลอร์ โดยทั่วไป DPTR จะถูกใช้เป็นตัวบอกแอดเดรสของ TABLE ส่วนแอดคิวมูเลเตอร์ จะกำหนดเป็นตัวชี้ตาราง

ตัวอย่าง

```
MOC   DPTR, # JUMP_TABLE
MOV   A, IMDEX_NUMBER
RL    A
JMP   @A+DPTR
JUMP_TABLE
AJMP  CASE_0
AJMP  CASE_1
AJMP  CASE-2
AJMP  CASE-3
AJMP  CASE-4
```

จากตัวอย่างเป็นการกระโดด 5ฑ (5-way branch) ค่า 0-4 ซึ่งจะถูก load เข้าไปเก็บไว้ในแอดคิวมูเลเตอร์

คำสั่ง RLA จะ Convert ตัว index-number (0-4) โดยจะ Rotate จำนวน คู่ 0-ถึง 8 เพราะว่า แต่ละตำแหน่งในตารางห่างกัน 2 ไบต์ พอดี

จากตารางที่ 7 คำสั่ง CALL addr หมายถึง LCALL และ ACALL CALL in a generic mnemonic which can be used if the programmer does not care which way the address in encoded

คำสั่ง LCALL ใช้ 16 บิตแอดเดรส สามารถเรียกโปรแกรมย่อยได้ในช่วง 2K ไบต์ ที่ใด ๆ ใน Program memory Space คำสั่ง ACALL ใช้ 11 บิต แอดเดรส สามารถเรียกโปรแกรมย่อยได้ในชวง 2K ไบต์ (block)

เราสามารถกำหนดแอดเดรสของโปรแกรมย่อย ได้ด้วย LABEL หรือ ค่าคงที่ 16 บิตแอสเซมเบลอร์ จะใสแดเดรสในรูปแบบที่ถูกต้องให้เลย การใช้โปรแกรมย่อยจะต้องลงด้วย คำสั่ง RET ซึ่งจะ Return คำสั่งมายังคำสั่งต่อไปที่ถัพมาจากคำสั่ง CALL

คำสั่ง RETI จะถูกใช้กรณีการกระโดดกลับจาก interrupt service routine

ข้อแตกต่างระหว่าง RET และ RETI คือ

RETI tells the interrupt control System that the interrupt in progress is done. if there is no tnterrupt in progress at the time RETI is executed, THEN the RETI is functionally identical to RET

Mnemonic	Operation	Addressing Modes				Execution Time (μs)
		Dir	Ind	Reg	Imm	
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNE A,<byte>,rel	Jump if A ≠ <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

ตารางที่ 8 Conditional Jumps in MCS-51 Devices

จากตารางที่ 8 แสดงการกระโดดแบบมีเงื่อนไข ทุกคำสั่งการกระโดดแบบมีเงื่อนไขแอดเดรสปลายทางเป็นแบบ relative offset ช่องการก้าวกระโดด -128 to+127 ไบต์ จากคำสั่งที่ตามหลังคำสั่ง Jump Condition Important to note : however, the user specifics to the actual destiration address the same way as the other Jumps : as a label or a 16-bit constant

คำสั่ง DJNZ ลดค่า และก้าวกระโดดกรณีที่ยังมีค่าไม่เท่ากับศูนย์เป็นคำสั่งสำหรับ loop control

กรณีที่จะส่งให้กระทำลูป N ครั้ง โดยการโหลดค่าที่ต้องการลงไป เช่น

ตัวอย่าง

N = 10 จะได้ว่า

```
        MOV   COUNTER, #10
LOOP :      (begin loop)
            .
            .
            .
          (end loop)
        DJNZ COUNTER, LOOP
          (continue)
```

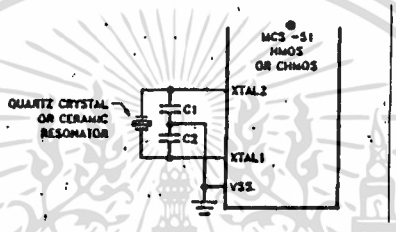
คำสั่ง CJNE เป็นคำสั่งเปรียบเทียบค่า และจะก้างกระโดดเมื่อทั้ง 2 ค่า ที่มาเปรียบเทียบมีค่าไม่เท่ากัน มีประโยชน์สำหรับ loop control เช่นกัน ตัวอย่างในรูปที่ 10 ต้องมี operand 2 ตัว และจะก้างกระโดดเมื่อ operand ทั้งสองตัวมีไม่เท่ากัน operand ตัวแรกคือ ข้อมูลที่เก็บไว้ใน R1 ค่าเริ่มต่นคือ 2EH และ operand ตัวที่ 2 คือ 2AH ทุกครั้งที่กระทำ loop R1 จะถูกลดค่า ลูปจะถูกกระทำจนกระทั่งข้อมูลใน R1 มีค่าเท่ากับ 2AH

การนำไปใช้อย่างอื่น เช่น เปรียบเทียบมากกว่าหรือน้อยกว่า โดยเปรียบจำนวน 2 จำนวน เป็น จำนวนเต็มที่ไม่คิดเครื่องหมาย (unsinged integer) ถ้าจำนวนแรกน้อยกว่าจำนวนที่ 2 ฉฟพพี บิต จะถูกเซ็ตให้เป็น "1"

ถ้าจำนวนแรกมีค่ามากกว่าหรือเท่ากับ Carry bit จะถูกเคลียร์ Clear ให้เป็นศูนย์

CPU Timing

MCS-51 ทุกตัว จะมีวงจรกำเนิดสัญญาณ (oscillator) ภายในตัว IC อยู่แล้วเราเพียงต่อตัว Resonater เช่น Crytal หรือไม่ก็ Ceramic resanater ระหว่างขา XTAL1 และ ขา XTAL2 เท่านั้น และก็ต่อตัวเก็บประจุดังรูปที่ 11



รูปที่ 11

กรณีที่เราต้องการใช้ oscillator จากภายนอก เราสามารถต่อไปดังรูปที่ 12 ตัวกำเนิดสัญญาณนาฬิกา ภายใน MCS-51 ทำให้เกิดขบวนของ Machin cycle ดังนี้

Machine Cycle

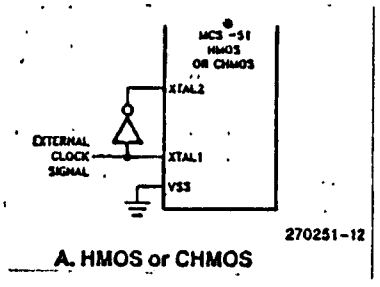
ใน 1 machine Cycle ประกอบลไปด้วย sequence 6 states
คือ สเตรท S1 ถึง S6 แต่ละเสตรทประกอบด้วย 2 คาบเวลา (2 period)

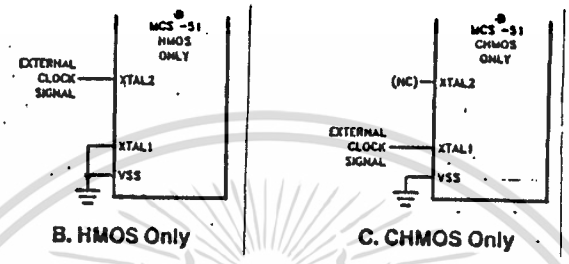
1 machin cycle = 6 state (S1-S6)

1 State = 2 period

จะได้ว่า 1 machine Cycle จะมี 12 period กรณีที่ใช้ความถี่ 12 MH_z 1 machine Cycle นี่ก็คือ 1uSec นั่นเอง ดังรูปที่ 13 (5-15)

โดยทั่วไปจะมีการ fetch คำสั่งเสตรทที่ 1 และ 4 (S1 และ S4) และเมื่อ State ถึง S6 การ execute จะเสร็จสิ้นด้วย ใน Machine Cycle เดียวกัน





รูปที่ 12 Using an External clock

คู่มือนำทางสำหรับนักโปรแกรมและชุดคำสั่งของ MCS-51

(MCS-51 Programmer's Guide and Instruction Set)

Memory organization

Program memory

8051 (รวมทั้ง 8031 ด้วย) ได้แยกส่วนแอดเดรนสเป็นสองส่วนคือ

-ส่วน Program memory

-ส่วน datd memory

ส่วนของ Program memory อ้างได้ถึง 64K

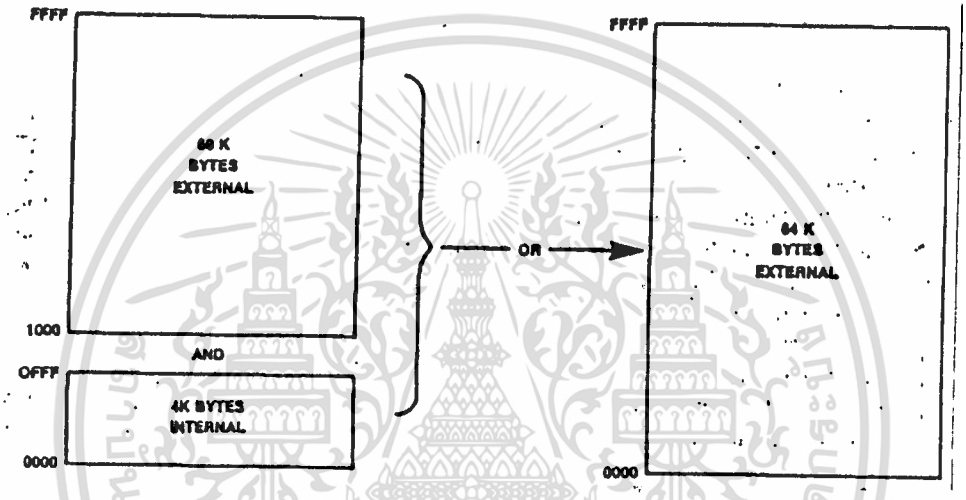
ในเบอร์ 8051 มีอยู่ภายในตัวมันแล้ว 4K (8051 มี 8K) ส่วน 8031 ไม่มี Program memory ในตัว ต้องต่อใช้ด้านนอก

Data memory

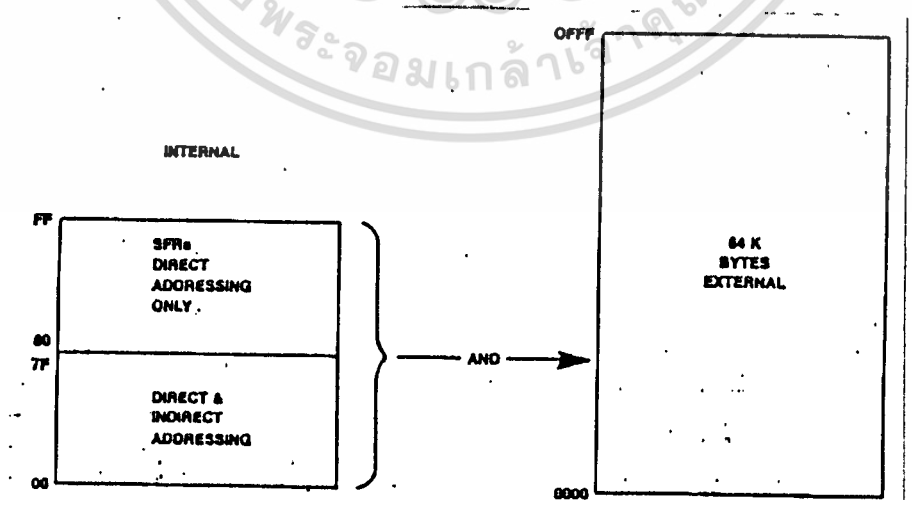
ใน 8051 ส่วน data memory สามารถ อ้างได้ถึง 64K bytes ใน 8051 มี data memory ภายในตัว 128 bytes (RAM) (ส่วน 8052 มี

256 bytes) เบอร์ 8031 มี data memory ภายในตัว 128 byte เช่นกัน และยังมีส่วน Special Function Registers (SFR_)

โดยที่ 128 byte ค่าเป็นส่วนของ RAM และ 128 ไบต์ บนใช้กรณี SFR_ การใช้คำสั่ง "MOVX" เป็นการอ้างถึงแอดเดรสนี่เป็นส่วน external data memory 128 byte ต่ำ (RAM) สามารถอ้างได้ทั้งแบบ direct (MOV data addv) หรือไม่ก็ indirect (MOV @Ri) ก็ได้ ส่วน SFR ได้ตัวแบบ direct เท่านั้น



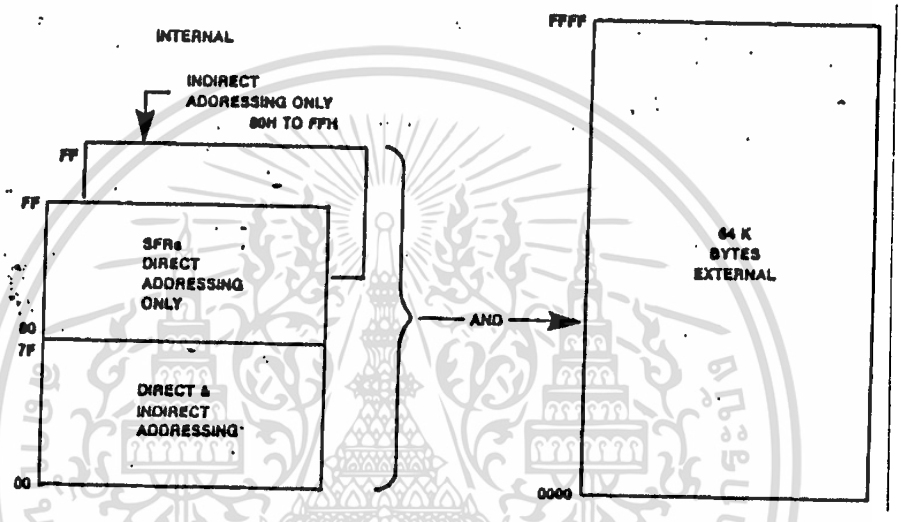
รูปที่ 13 The 8051 Program Memory



รูปที่ 14 แสดงส่วน data memory ของ 8051 (8031)

Indirect ADDRESS AREA

จะเห็นได้ว่า ส่วนที่เป็น SFR และส่วน indirect addressing จะอยู่บนแอดเดรสเดียวกัน 80-FFN มองเหมือนว่าซ้อนทับกันอยู่ แต่อย่างไรก็ตาม เราสามารถแยกพื้นที่ออกจากกันได้โดยเด็ดขาด ขึ้นอยู่กับการใช้คำสั่งสำหรับกรณี กรณี 8052 ส่วนของ data memory ดังในรูปที่ 15



รูปที่ 15 8052 data memory

จากตัวอย่าง

MOV 80H. # OAAH

จะเห็นว่า นั่นคือการเอาค่า OAAH ไปใช้ที่ address 80H (หรือจากตารางแอดเดรส ของส่วน SFR ที่ต้องแบบ direct เท่านั้น คือ Port0 นั่นเอง) นั่นคือตอนนี้ที่ Port0 มีข้อมูล OAAH อยู่

และจากตัวอย่าง

MOV RO, #80H

MOV @RO, #OBBH

นั่นคือใส่ข้อมูลไปที่ address 80H เช่นกัน ในส่วนของ data RAM ถ้าเรา execute คำสั่ง ทั้งสองชุด คือ 1 และ 2 เราจะได้ Port0 มี OAAH ส่วนแอดเดรส 80H ของ RAM มี OBBH (ทั้งที่ Port0 ก็คือ แอดเดรส 80H เช่นกัน) เราจะมองสองส่วนนี้แยกออกจากกัน สำหรับ 128 bytes บน กรณีนี้มีปัญหาเฉพาะของ 2 เท่านั้น กรณีของ 8051 (รวมทั้ง 8031) จะมีเฉพาะส่วน SFR เท่านั้น นั่นคือ 128 bytes บน สามารถอ้างได้แบบ direct เท่านั้น

ส่วนของ DIRECT AND INDIRECT ADDRESS AREA

128 bytes ล่าง สามารถอ้างได้ทั้ง direct และ indirect แบ่งเป็น 3 segments คือ

1. Register Bank 0-3 :

อยู่ตั้งแต่แอดเดรส 0-ถึง IFH (32 ไบต์) หลังที่ผ่านการ Reset จะมีค่า default ไว้ที่ bank0 การที่จะเลือก bank ของ Register ผู้ใช้สามารถเลือกได้จากซอฟท์แวร์และแต่ละรีจีสเตอร์แบงค์ จะมีรีจีสเตอร์ 8 ตัว (ตัวละไบต์) Register 0 ถึง 7 พอ Reset เริ่มต้น Stack Pointer จะอยู่ที่ 07H และจะเริ่มเพิ่มค่าที่ address 08H ซึ่งตรงกับตำแหน่ง R0 ของRegister bank1 ฉนั้น กรณีที่เราต้องการใช้ Register bank มากกว่า 1 bank เราต้องไปกำหนด SP register ให้ชี้ไปยัง address RAM ที่ไม่ใช้เก็บข้อมูล

2. Bit addressable AREA :

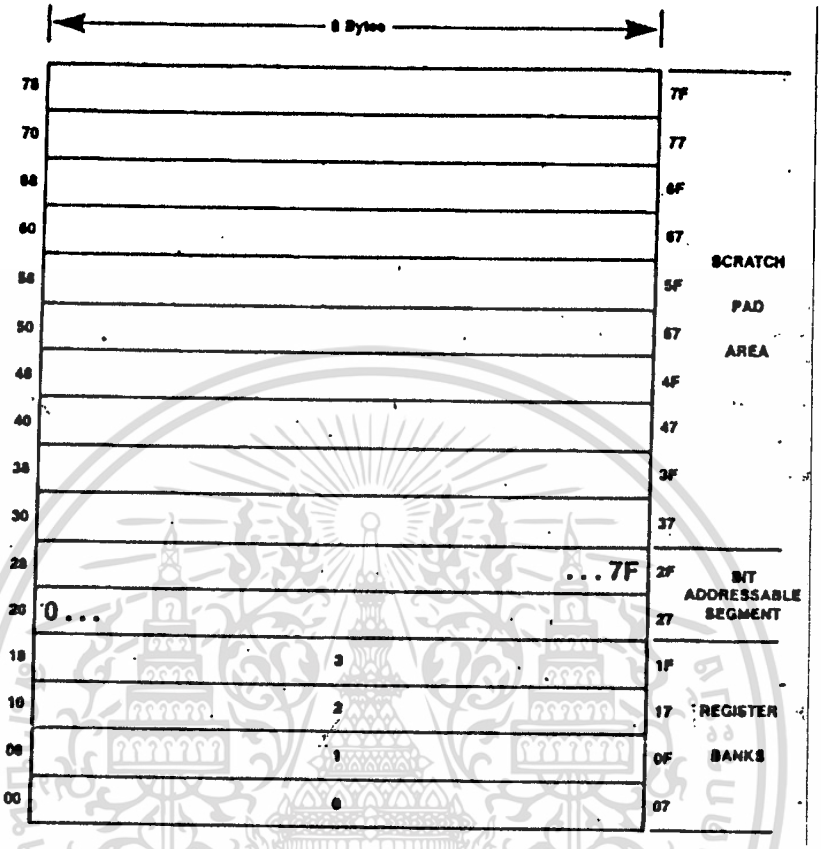
มีทั้งหมด 16 byte ตั้งแต่ แอดเดรส 20H-2FH หรือ 128 บิต โดยตัวบิตแอดเดรสได้ตั้งแต่ 0-7FH ได้โดยตรงเลย

การเข้าถึงแบบแบิตอาจทำได้ 2 วิธี วิธีแรกอ้างถึงแอดเดรสบิต 0-7FH อีกวิธีหนึ่งอ้างแอดเดรส 20H-2FH, บิต 0-7 โดยอ้างดังนี้

20.0 -20.7, บิตที่ 8-FH ก็คือ 21.0-21.7 ใน 16 byte นี้ (หมายถึงบิตแอดเดรสเอเบิลแอเรีย่) สามารถอ้างแบบไบต์ก็ได้

3. Scrath Pad Area

คือ บริเวณแอดเดรสไบต์ที่ 30H ถึง 7FH เป็นส่วนที่สามารถนำมาใช้งานได้



รูปที่ 16 128 bytes OF RAM direct and indirect and indirect addressable

Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
*TCON	Timer/Counter Control	88H
*+T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
+TH2	Timer/Counter 2 High Byte	0CDH
+TL2	Timer/Counter 2 Low Byte	0CCH
+RCAP2H	T/C 2 Capture Reg. High Byte	0CBH
+RCAP2L	T/C 2 Capture Reg. Low Byte	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

ตารางที่ 1 แสดงส่วนของ SFR และแอดเดรสทั้งหมดของ 8051

Register	Value In Binary
*ACC	00000000
*B	00000000
*PSW	00000000
SP	00000111
DPTR	
DPH	00000000
DPL	00000000
*P0	11111111
*P1	11111111
*P2	11111111
*P3	11111111
*IP	8051 XXX00000, 8052 XX000000
*IE	8051 0XX00000, 8052 0X000000
TMOD	00000000
*TCON	00000000
*+T2CON	00000000
TH0	00000000
TL0	00000000
TH1	00000000
TL1	00000000
+TH2	00000000
+TL2	00000000
+RCAP2H	00000000
+RCAP2L	00000000
*SCON	00000000
SBUF	Indeterminate
PCON	HMOS 0XXXXXXX CHMOS 0XXX0000

X = Undefined
* = Bit Addressable
+ = 8052 only

SFR หลังจาก Power-On หรือ hardware Reset

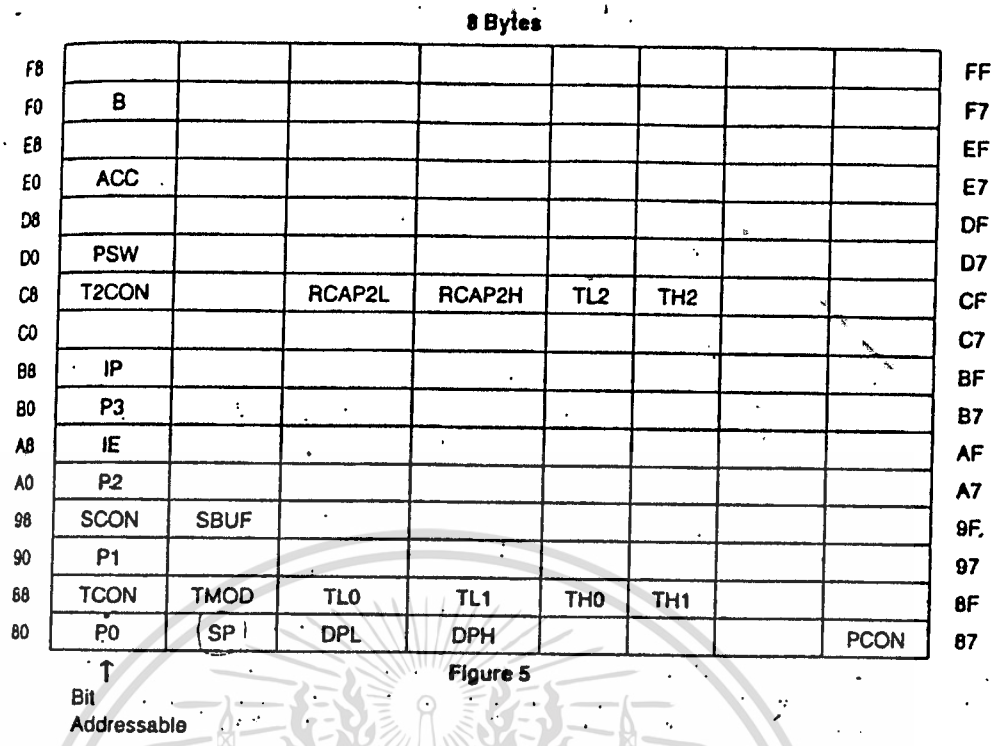


Figure 5

SFR MEMORY MAP

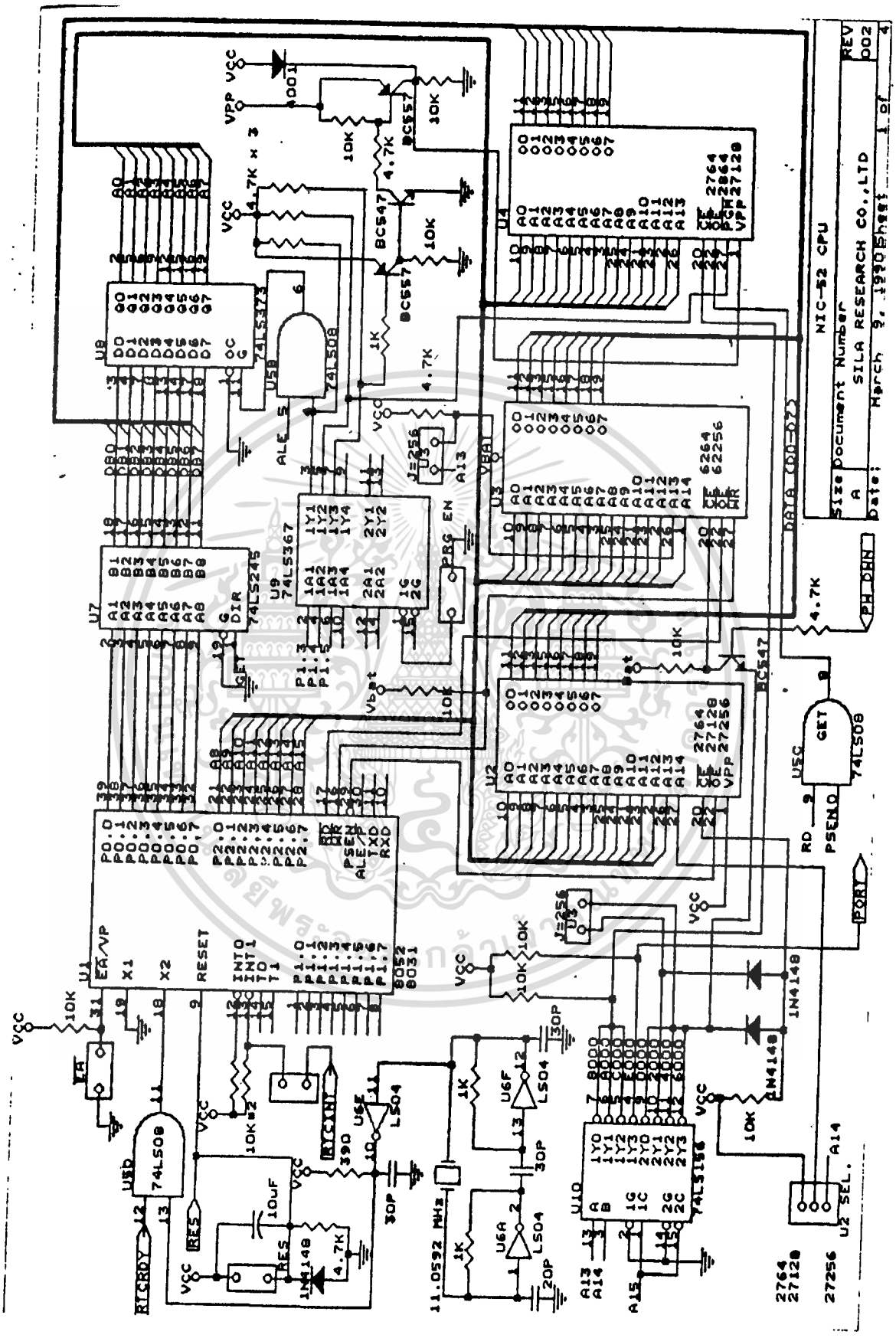
บทที่ 4

การออกแบบและสร้าง PLC

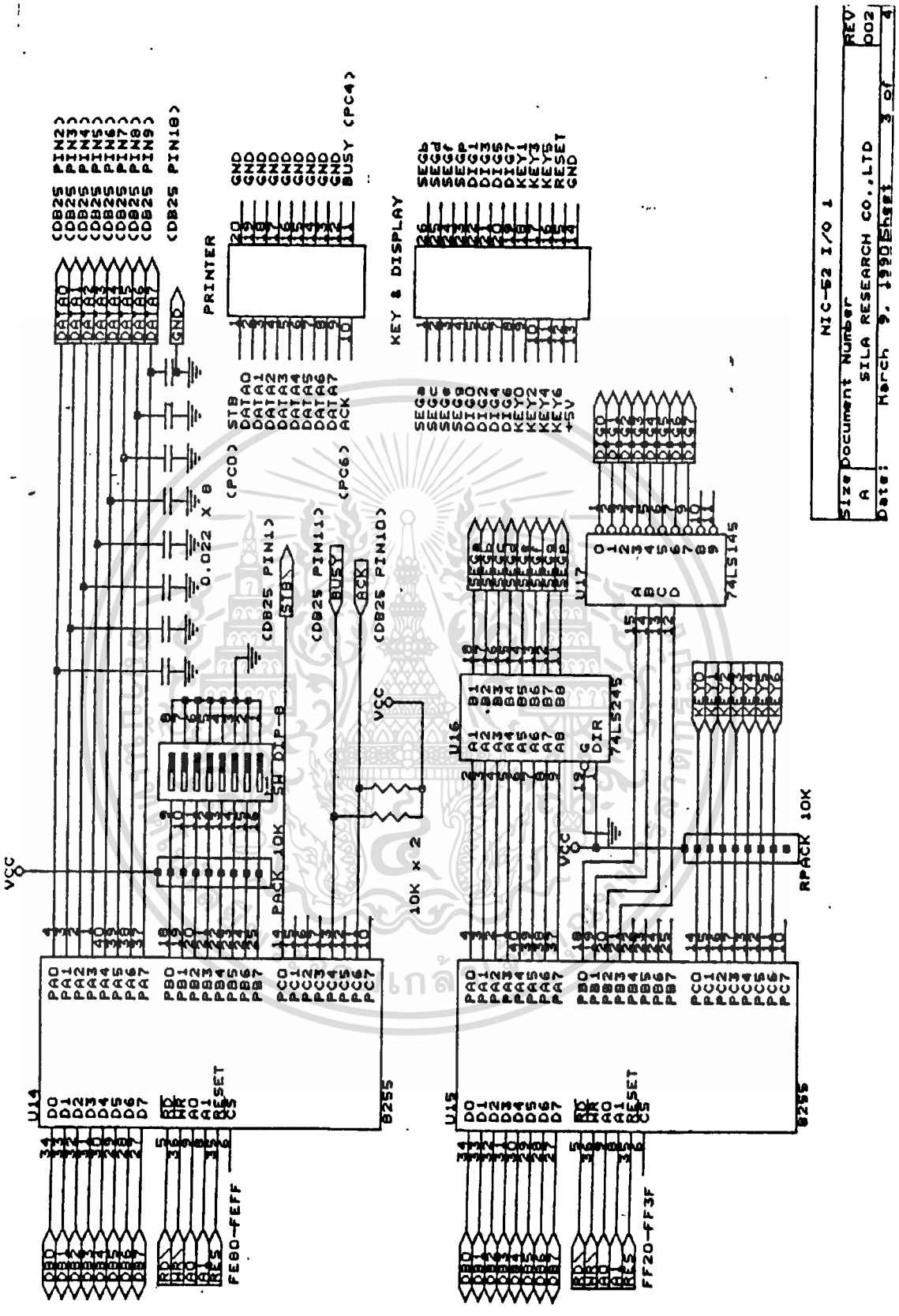
ในส่วนของฮาร์ดแวร์ของ PLC พื้นฐานนั้นมาจากการนำเอาวงจรไมโครโปรเซสเซอร์มาประยุกต์ใช้งาน ดังนั้นเพื่อความรวดเร็วและประหยัดเวลาทางกลุ่มและอาจารย์ที่ปรึกษาโครงงานจึงได้เลือกใช้วงจรสำเร็จรูป NIC-52 ซึ่งเป็นผลิตภัณฑ์ของบริษัทศิลา รีเสริช ที่เหมาะแก่การพัฒนางานทางด้าน PLC และ Microcontroller มาก นอกจากจะสะดวกเพราะไม่ต้องเสียเวลากับการสร้างอุปกรณ์ที่จำเป็นสำหรับ CPU 8052 แล้ว ภายในบอร์ด NIC-52 ยังมี PORT ต่างๆให้เลือกใช้ตามแต่ความต้องการ ทั้ง USER PORT ซึ่งใช้สื่อสารข้อมูลแบบขนานกับอุปกรณ์ภายนอก SERIAL PORT ซึ่งใช้สื่อสารแบบอนุกรมกับอุปกรณ์ภายนอกโดยผ่านทางเครื่องคอมพิวเตอร์ PC เพื่อช่วยการแก้ไขโปรแกรมที่พัฒนา รวมทั้งตัว NIC-52 เองยังมีหน่วยความจำที่เพียงพอสำหรับโปรแกรมบริหาร PLC อีกด้วย

นอกจากทางกลุ่มได้เลือกใช้ NIC-52 เป็นหน่วยประมวลผลสำหรับ PLC แล้ว ทางกลุ่มยังได้เลือกใช้ผลิตภัณฑ์ในตระกูล NIC-52 อีก คือ NIC-POWER ซึ่งจะใช้เป็นหน่วยจ่ายพลังงานให้แก่บอร์ดทุกๆบอร์ด NIC-OPTO ซึ่งจะใช้เป็นหน่วยอินพุทของ PLC NIC-SSR ซึ่งจะใช้เป็นหน่วยเอาท์พุทของ PLC ซึ่งรายละเอียดในส่วนประกอบต่างๆของ PLC จะได้กล่าวต่อไปดังนี้

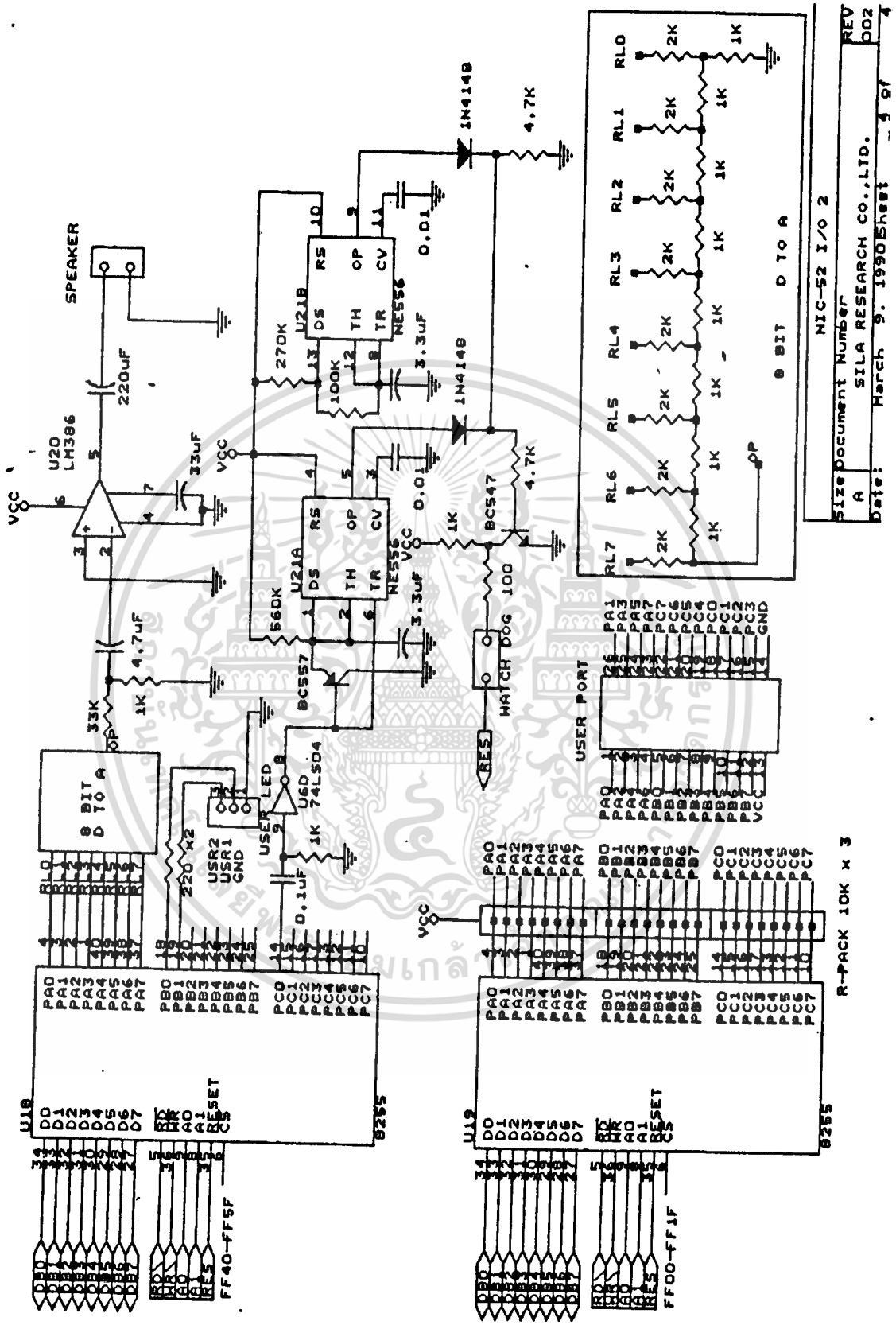
1.หน่วยประมวลผลกลาง ดังที่ได้กล่าวแล้วข้างต้นเกี่ยวกับความสำคัญของหน่วยประมวลผลกลางซึ่งเปรียบเสมือนหัวใจของ PLC CPU ของ PLC บนบอร์ดของ NIC-52 นั้นจึงใช้ CPU เบอร์ 8052 AH ซึ่งบนบอร์ดนี้ได้ถูกออกแบบสร้างวงจรที่มีอุปกรณ์ที่จำเป็นต่อการใช้งานไอซี 8052 ไว้พร้อม ไม่ว่าจะเป็นหน่วยความจำทั้งแบบ ROM และ RAM PORT สื่อสารทั้ง PRINTER PORT, USER PORT (8255), PORT 1 CPU รวมทั้ง KEYBOARD & DISPLAY PORT และยังมี SYSTEM BUS ซึ่งเป็นZ-80 COMPATIBLE อีกด้วย ซึ่งฮาร์ดแวร์ของบอร์ด NIC-52 ดังแสดงไว้ในรูป 3.1 - 3.5



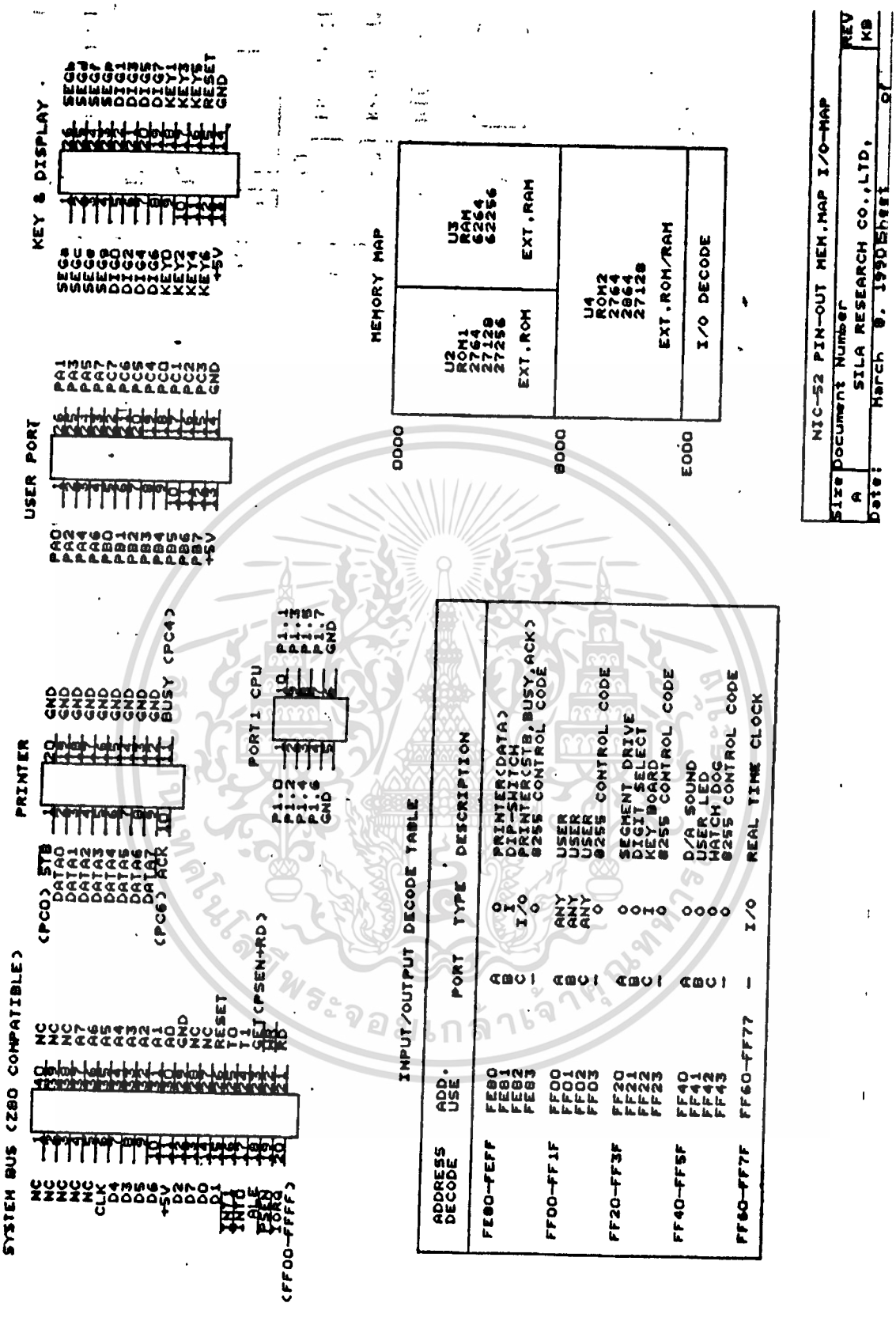
รูปที่ 3.1 แสดงวงจรฮาร์ดแวร์ของ NIC-52



รูปที่ 3.2 แสดงวงจร I/O ของ NIC-52



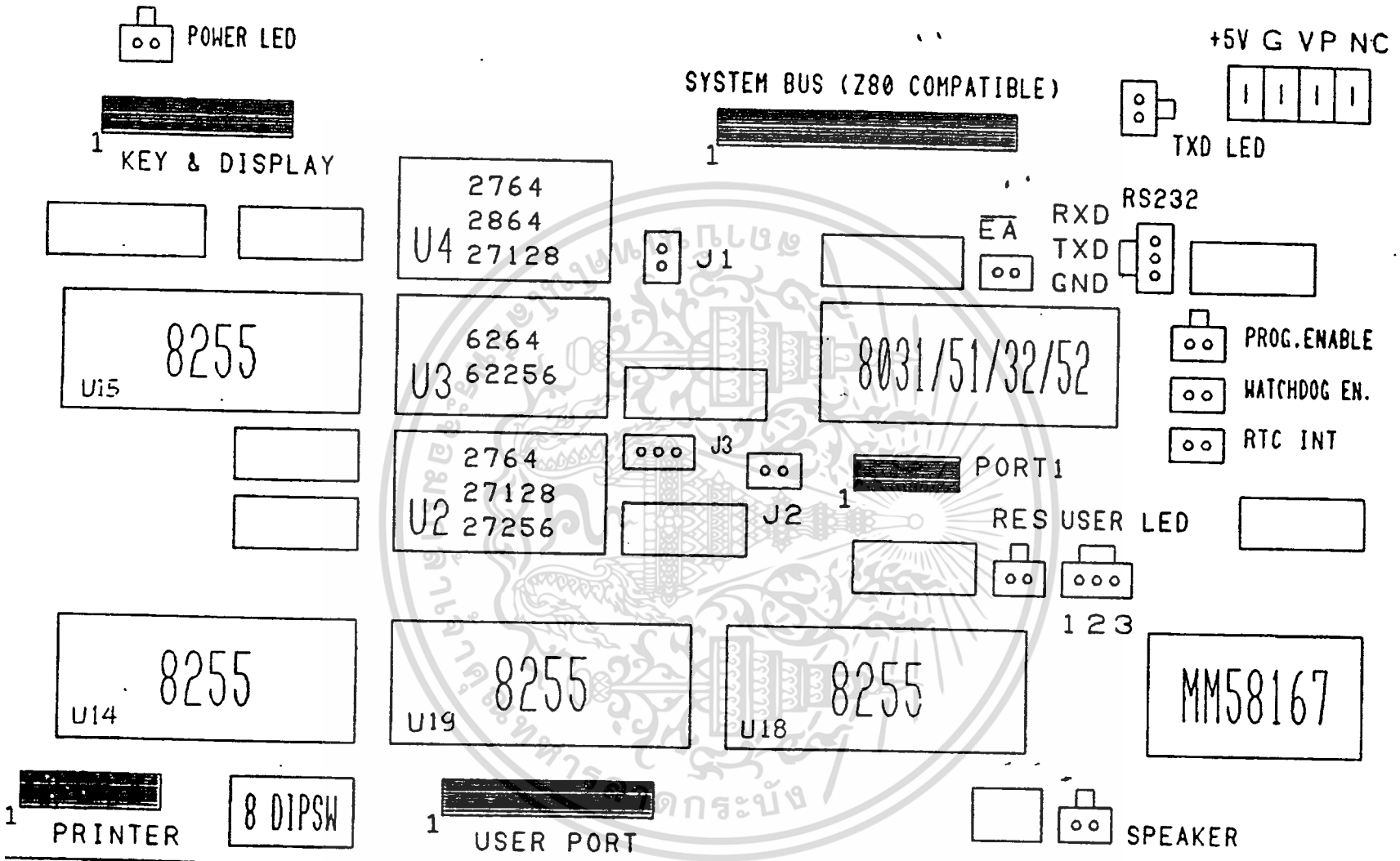
รูปที่ 3.3 แสดง I/O ของ NIC-52



รูปที่ 3.4 แสดงลักษณะของขา CONNECTOR ต่างๆที่อยู่บนบอร์ด NIC-52

ตารางการ DECODE I/O ต่างๆของ NIC-52

และตารางการ DECODE หน่วยความจำของ NIC-52



SELECT U2 2764/128 J3:LEFT
J3:RIGHT

SELECT U3 6264 J1:J2:OPEN
62256 J1:J2:CLOSE

รูปที่ 3.5 แสดง LAYOUT ของบอร์ด NIC-52

สำหรับการประยุกต์เอาบอร์ด NIC-52 เพื่อใช้เป็น CPU นั้น หน่วยความจำของ PLC จะแยกออกเป็น 2 ส่วนด้วยกัน คือ

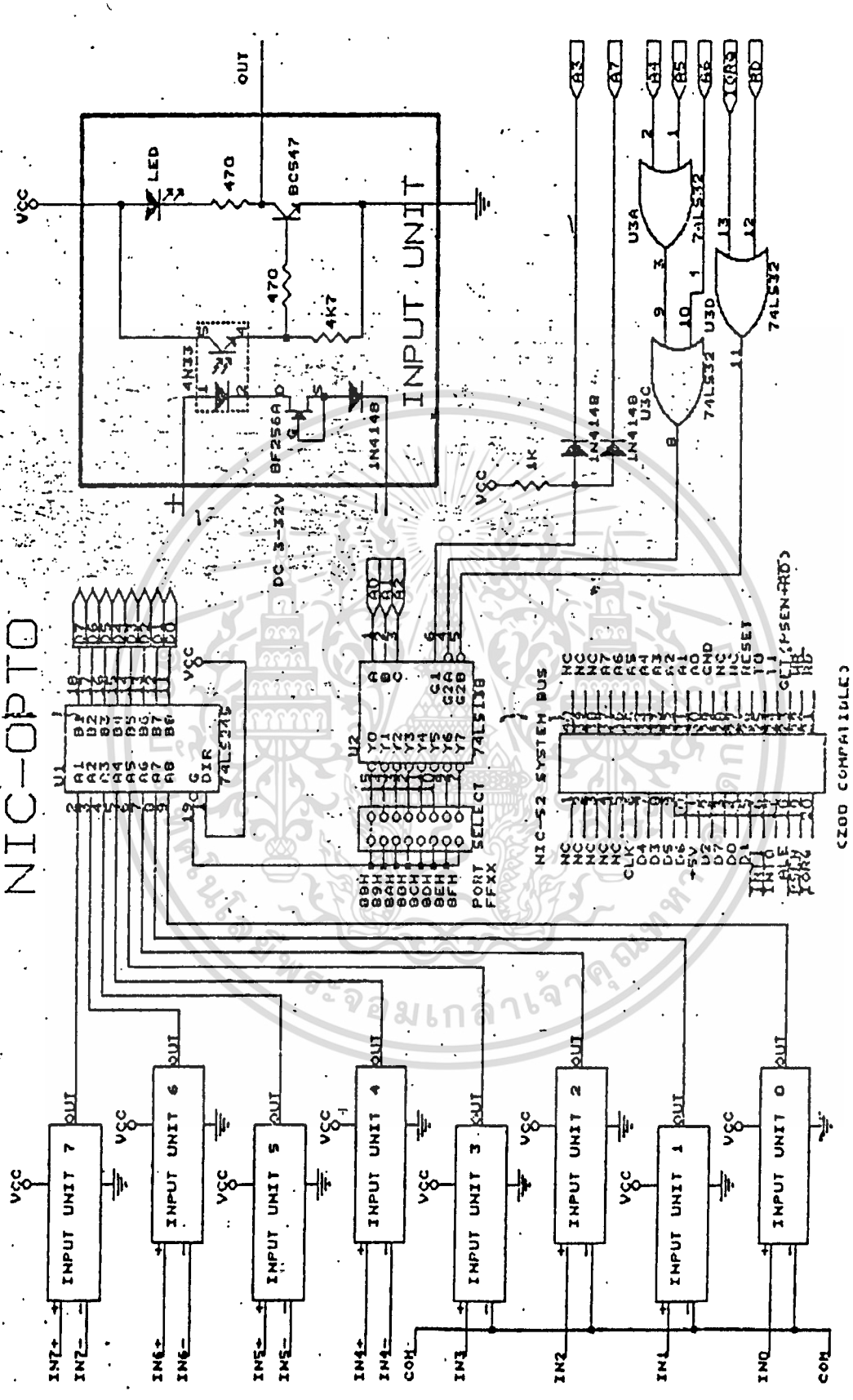
- ROM ซึ่งจะใช้ ROM ขนาด 32 กิโลไบต์ ทำหน้าที่เก็บโปรแกรมบริหารของระบบและข้อมูลของระบบ เป็นโปรแกรมที่ถูกพัฒนาขึ้นเพื่อควบคุมการทำงานของ PLC มีตำแหน่งอยู่ที่ U2 บนบอร์ด
- RAM ซึ่งจะใช้ RAM ขนาด 32 กิโลไบต์เช่นกัน ทำหน้าที่เก็บโปรแกรมใช้งาน ซึ่งผู้ใช้จะเป็นผู้เขียน ข้อมูลสภาวะของหน่วยอินพุท-เอาท์พุท ค่าเวลา และค่านับต่างๆ ตลอดจนค่าตั้งนับ (set value) ของตัวตั้งเวลา (timer) และตัวนับ (counter) รวมทั้งค่าในบัฟเฟอร์ต่างๆ ที่โปรแกรมบริหารระบบ ใช้ในการประมวลผลก็จะอยู่ใน RAM ตัวนี้เช่นกัน

2.หน่วยอินพุท-เอาท์พุท ดังที่ได้กล่าวไว้ในทฤษฎีของ PLC แล้วว่า นอกจาก CPU ของ PLC แล้ว PLC จะต้องประกอบด้วยหน่วยอินพุท-เอาท์พุท เพื่อที่จะรับสภาวะที่ต้องการเข้ามาทางหน่วยอินพุทเพื่อประมวลผล แล้วจึงส่งผลที่ได้จากการประมวลออกไปทางหน่วยเอาท์พุท

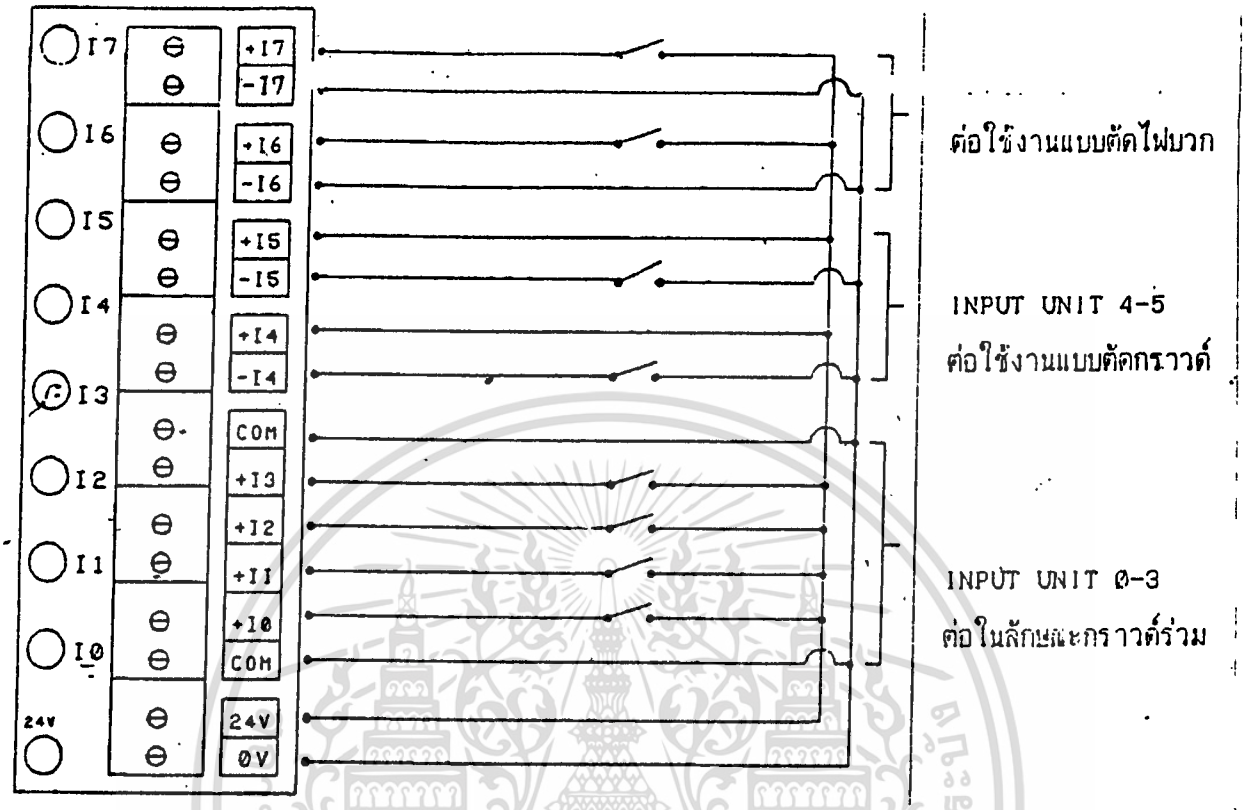
เช่นกันกับ CPU ทางกลุ่มได้เลือกใช้ผลิตภัณฑ์ตระกูล NIC-52 เพื่อใช้เป็นหน่วยอินพุท-เอาท์พุทดังนี้ คือ

1. <u>หน่วยอินพุท</u> ได้เลือกใช้ NIC-OPTO ซึ่งเป็นแบบ OPTO-ISOLATE LNPUT BOARD มีจำนวนอินพุท 8 แชนแนล และรับสัญญาณอินพุทแบบ DC ในช่วงแรงดันระหว่าง 3-32 V โดยเชื่อมต่อกับบอร์ด NIC-52 ทาง SYSTEM BUS ซึ่งเป็น Z-80 COMPATIBLE NIC-OPTO มีวงจรฮาร์ดแวร์ดังรูปที่ 3.6

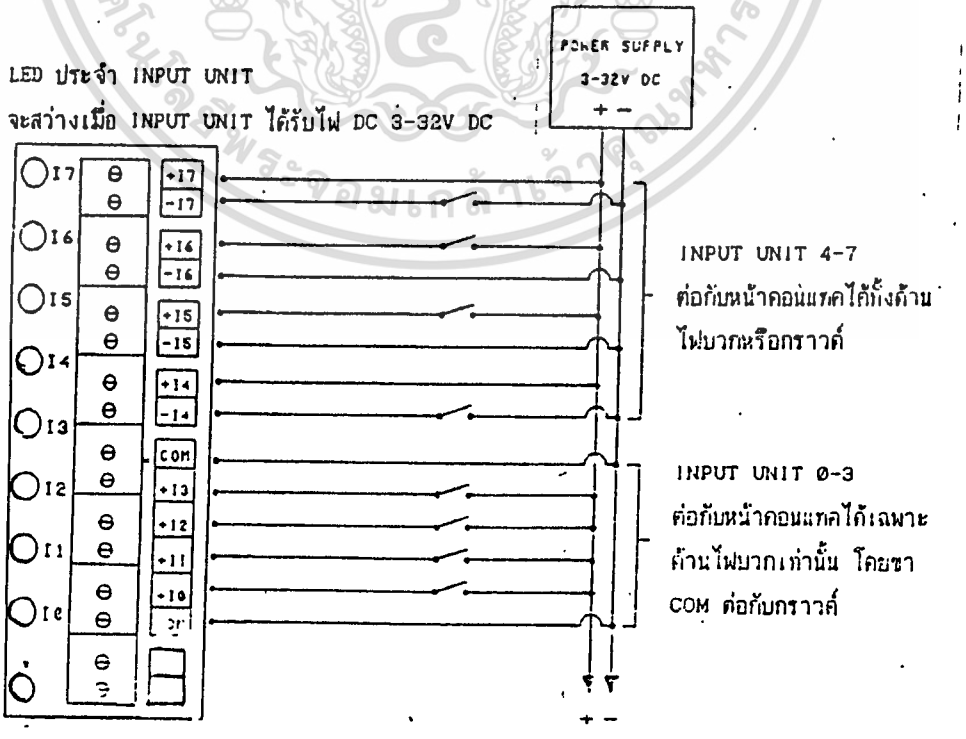
สำหรับการใช้ขั้นตอนการใช้ NIC-OPTO สามารถทำได้ 2 แบบ คือ ใช้ไฟ 24 V DC จากบอร์ด NIC-OPTO เอง หรือใช้ไฟ 3-32 V DC จากแหล่งจ่ายไฟภายนอก



รูปที่ 3.6 แสดงวงจรฮาร์ดแวร์ของ NIC-OPTO



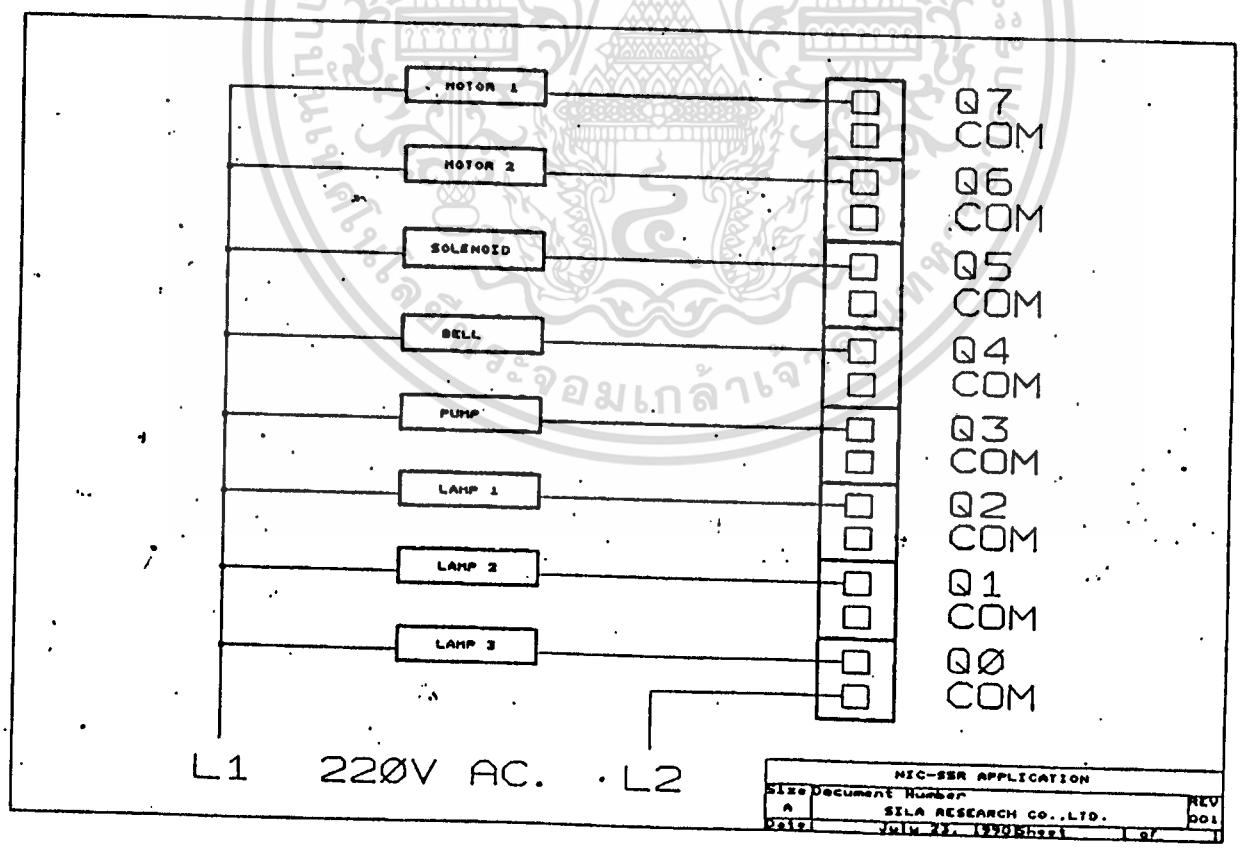
รูปที่ 3.7 แสดงตัวอย่างการใช้ NIC-OPTO แบบใช้ไฟ 24 V DC จากบอร์ด



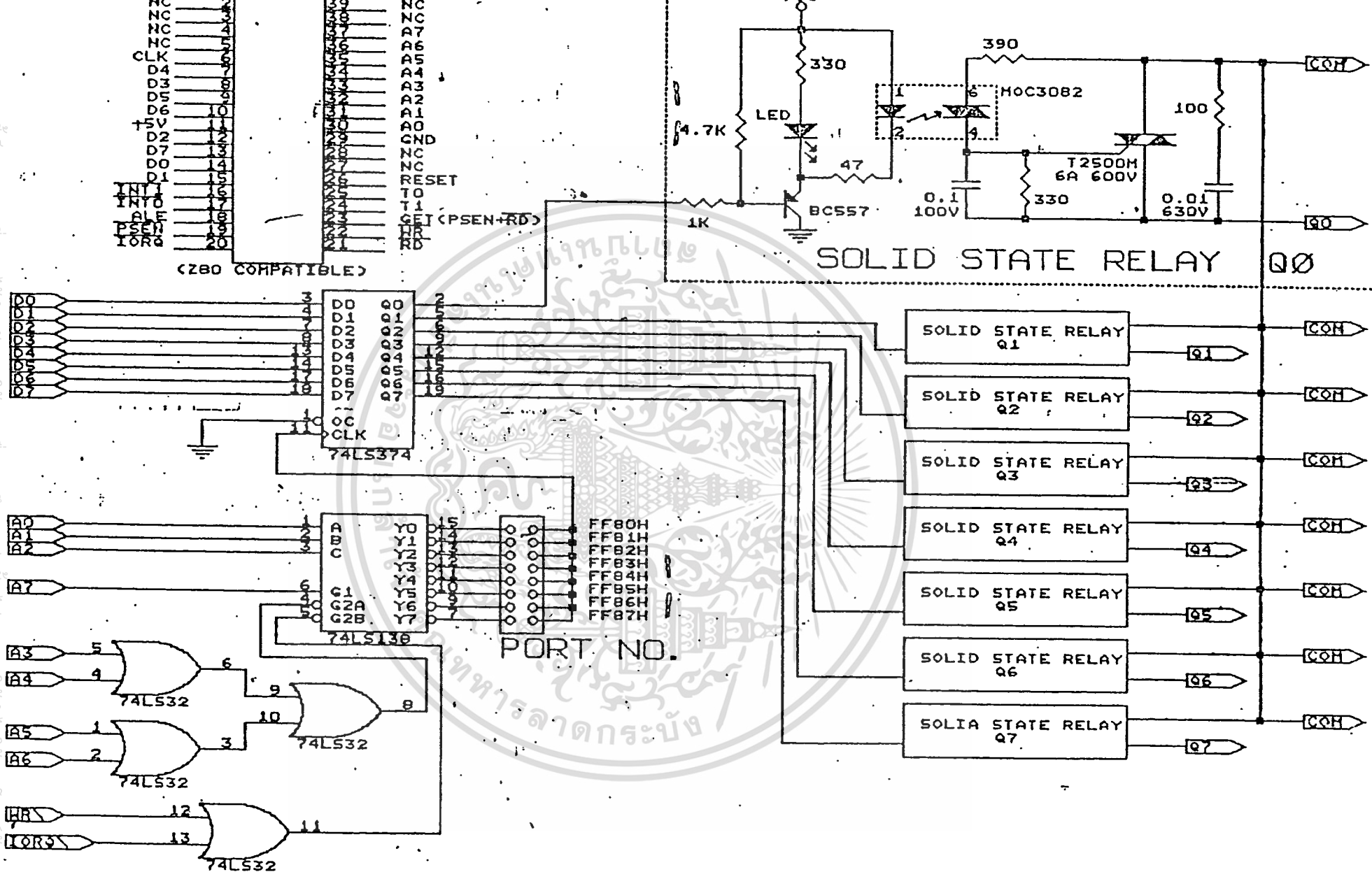
รูปที่ 3.8 แสดงตัวอย่างการใช้ NIC-OPTO แบบใช้แหล่งจ่ายไฟภายนอก

โดยที่ INPUT0 - INPUT3 จะต่อกับหน้าคอนแทคได้เฉพาะไฟบวกเท่านั้น และต้องต่อร่วมขา COM ทั้งสองกับ GROUND ของแหล่งจ่ายไฟ ดังแสดงในตัวอย่างรูปที่ 3.7 และ 3.8 ส่วน INPUT4 - INPUT7 นั้น สามารถต่อกับหน้าคอนแทคได้ทั้งด้านไฟบวกหรือ GROUND

2. หน่วยเอาท์พุท ได้เลือกใช้ NIC-SSR ซึ่งเป็นแบบ SOLID STATE RELAY จำนวน 8 แชนแนล โดยเอาท์พุทสามารถควบคุมไฟ AC 220 V กระแส 6 A และมีการเชื่อมกับบอร์ด NIC-52เช่นกันกับ NIC-OPTO NIC-SSR มีวงจรฮาร์ดแวร์ดังรูปที่ 3.10 และตัวอย่างการใช้ NIC-SSR ก็จะแสดงในรูปที่ 3.9



รูปที่ 3.9 ตัวอย่างการใช้ NIC-SSR กับโหลดภายนอกแบบต่างๆ



รูปที่ 3.10 แสดงวงจรภายในของ NIC-SSR

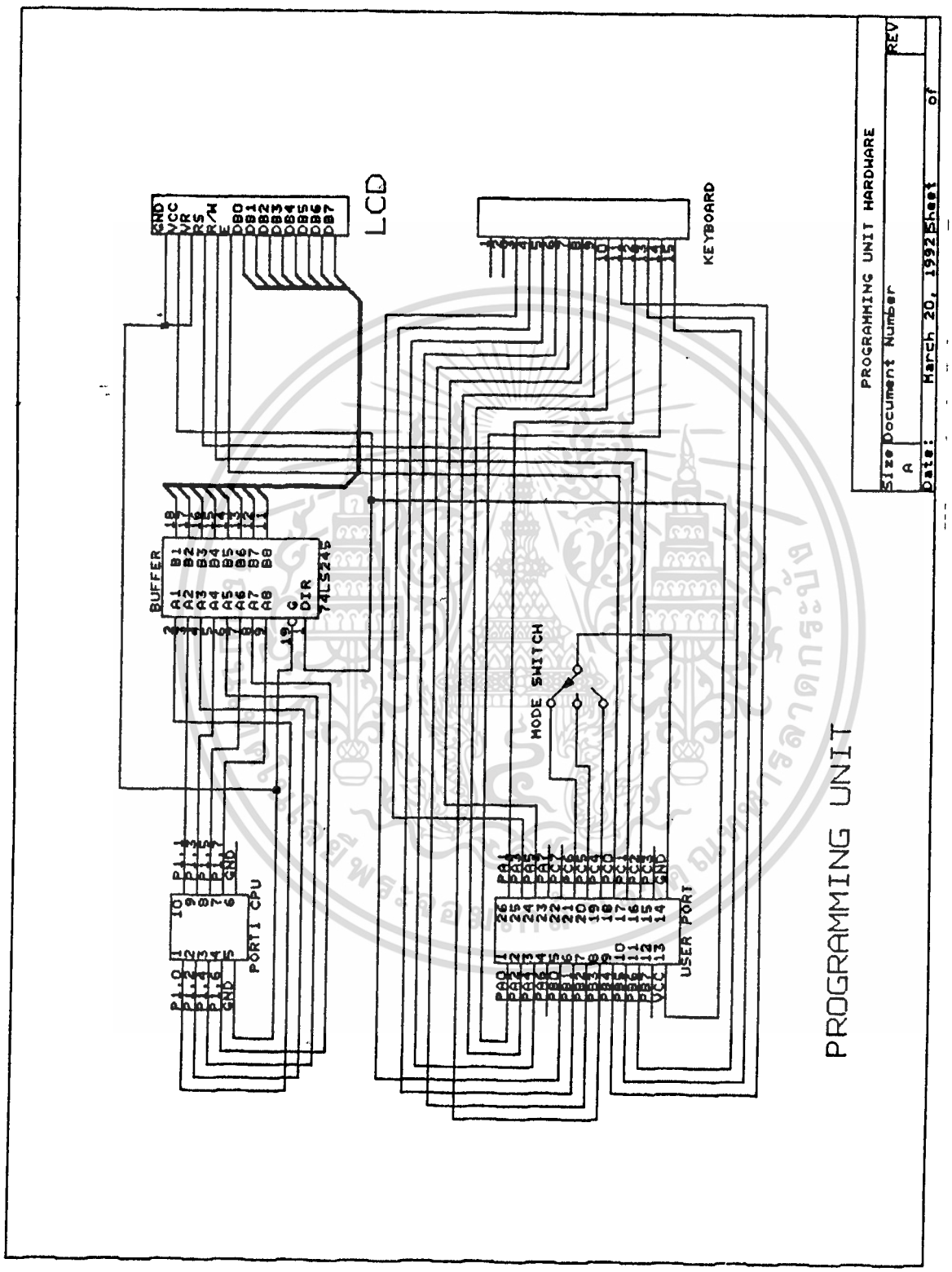
3.หน่วยป้อนโปรแกรม เป็นหน่วยที่มีหน้าที่ติดต่อสื่อสารผู้ใช้กับเครื่อง PLC เพื่อป้อนชุดคำสั่งต่างๆที่ประกอบกันเป็นโปรแกรมในการควบคุม รวมทั้งแก้ไขโปรแกรมที่เก็บไว้ใน RAM ของเครื่อง และตรวจสอบสภาวะ ลอจิกต่างๆของเครื่องในระหว่างที่เครื่องทำงาน นอกจากนั้นแล้วยังเป็นตัวสั่งให้เครื่อง PLC เริ่มทำงานอีกด้วย ซึ่งหน่วยป้อนโปรแกรมนั้นจะประกอบด้วยส่วนประกอบ 3 ส่วนด้วยกัน คือ

1. สวิทช์เลือกโหมดการทำงาน
2. คีย์บอร์ดสำหรับป้อน ตรวจสอบ และแก้ไขโปรแกรม
3. จอภาพแสดงผลแบบ LCD

สำหรับการทำงานของหน่วยป้อนโปรแกรมนั้น จะทำงานโดยการต่อโดยตรงกับ PORT 1 CPU และ USER PORT (ซึ่ง USER PORT นั้นตามวงจรฮาร์ดแวร์ในรูปที่ 3.3 นั้น จะเห็นว่าก็คือ PORT ทั้ง 3 PORT ของ 8255 (U19) นั้นเอง) ของ NIC-52 ถึงแม้ว่าในตัว NIC-52 เองจะมีทั้ง KEYBOARD & DISPLAY PORT แล้วก็ตาม การที่ไม่ใช้ PORT ดังกล่าวเนื่องมาจากว่าคีย์บอร์ดของหน่วยป้อนโปรแกรมนั้นมีจำนวนคีย์มากกว่าที่มีอยู่บน PORT (คีย์ของหน่วยป้อนโปรแกรมมี 42 คีย์ แต่ PORT สำเร็จรูปบน NIC-52 มีเพียง 12 คีย์ (4x3)) และจอแสดงผลของหน่วยป้อนโปรแกรมเป็นจอแบบ LCD ซึ่งบน PORT ของ NIC-52 จะเป็นแบบ 7-SEGMENT LED จำนวน 8 ตัว จึงไม่สามารถใช้ร่วมกันได้ วงจรของหน่วยป้อนโปรแกรมแสดงไว้ดังรูปที่ 3.11

1.<u>สวิทช์เลือกโหมดการทำงาน</u> เป็นสวิทช์ที่ทำหน้าที่เลือกโหมดการทำงานของ PLC ซึ่งจะต้องแบ่งเป็น 3 โหมด คือ โหมดโปรแกรม (PROGRAM MODE), โหมดมอนิเตอร์ (MONITOR MODE), และโหมดการทำงาน (RUN MODE)

สวิทช์เลือกโหมดการทำงานนั้นจะต่อกับ USER PORT บนบอร์ด NIC-52 ที่ขา PC_4,PC_5 และ PC_6 ของ 8255 (U19) ซึ่งจากรูปวงจรของ USER



รูปที่ 3.11 แสดงวงจรหน่วยป้อนโปรแกรม

PORT (รูปที่ 3.3) จะเห็นว่าทุก PORT ของ 8255 ได้มีการ PULL UP ค่าทุกค่าไว้แล้ว ดังนั้นถ้าไม่มีการต่อกับวงจรใดๆค่าที่ PORT C ของ 8255 ได้รับ คือ 11111111_B และเมื่อดูวงจรของหน่วยป้อนโปรแกรมแล้ว (รูปที่ 3.11) ค่าที่ PORTC ล่าง จะได้รับเมื่อสวิทช์เลือกโหมดต่างๆ ดังตารางที่ 3.1 แสดงไว้

โหมด	PC_7	PC_6	PC_5	PC_4
RUN	1	1	1	0
MONITOR	1	1	0	1
PROGRAM	1	0	1	1

ตารางที่ 3.1 แสดงข้อมูลของ PORT C ล่างเมื่อสวิทช์เลือกที่โหมดใดๆ

2.<u>คีย์บอร์ด</u> คีย์บอร์ดที่ใช้เป็นคีย์บอร์ดสำเร็จรูป จาก PROGRAMMING CONSOLE ของ PLC OMRON ซึ่งอาจารย์ที่ปรึกษากลุ่มจัดหามาให้ เป็นคีย์บอร์ดชนิด เมตริก เมมเบรนสวิทช์ จำนวน 6x7 คือ 42 คีย์ สำหรับคีย์บอร์ดนั้นจะต่อโดยตรงกับบอร์ด NIC-52 ผ่านทาง USER PORT ที่ขา PA_0 -PA_5 และ PB_0 - PB_6 ของ 8255 (U19) ดังแสดงในรูปที่ 3.11 ซึ่งการทำงานของคีย์บอร์ดก็จะทำงานโดยอาศัยหลักการสแกนคีย์ทั่วๆไป โดยที่ PB_0 - PB_6 จะส่งข้อมูลสแกนเอาท์ออกมา และPA_0 - PA_5 จะรับค่าสแกนอิน

นอกจากคีย์ต่างๆที่แสดงบนหน้าปัดอยู่แล้ว ยังมีฟังก์ชั่นการใช้งานบางฟังก์ชั่นที่จะต้องกดคีย์หลังจากกดคีย์ FUN อันได้แก่

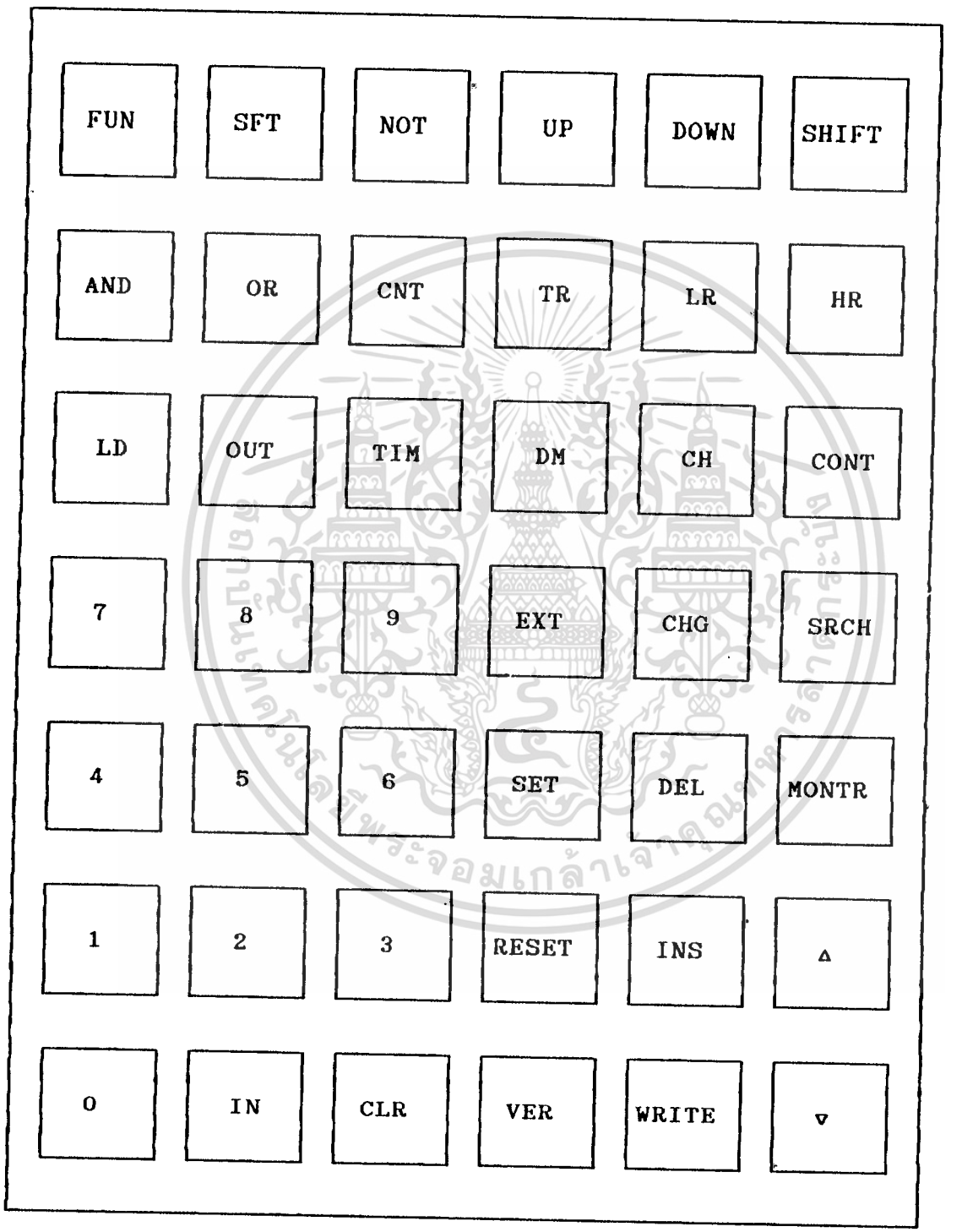
ลำดับการกดคีย์	INSTRCTION
FUN 1	END
FUN 2	T SHOT
FUN 3	T OFF
FUN 4	RTM
FUN 5	RCNT

ตารางที่ 3.2 แสดงฟังก์ชั่นพิเศษที่ไม่ได้แสดงบนหน้าปัดคีย์บอร์ด

3. <u>จอภาพแสดงผลแบบ LCD</u> เป็นส่วนแสดงผลเพื่อติดต่อสื่อสารกับผู้ใช้ จอภาพนี้เป็นแบบ DOT MATRIX LCD ขนาด 4 บรรทัด บรรทัดละ 16 ตัวอักษร ซึ่งสามารถแสดงผลเป็นตัวอักษรทำให้ง่ายแก่การเข้าใจ จอภาพนี้จะแสดงผลต่างกันในแต่ละโหมดการทำงานของ PLC คือ ในโหมดโปรแกรมจอภาพจะแสดงลำดับคำสั่งและค่าสั่งที่ผู้ใช้ได้โปรแกรมผ่านคีย์บอร์ดเข้าไป หรือที่เก็บไว้ในหน่วยความจำในโหมดมอนิเตอร์ จอภาพจะแสดงสภาวะของอินพุท เอาท์พุท ตัวนับเวลา ตัวนับ และ internal relay ในขณะที่กำลังทำงานตามโปรแกรมอยู่ หรือที่ถูกแก้ไขระหว่างการทำงาน และในโหมดการทำงาน จอภาพก็จะแสดงผลเช่นเดียวกันกับในโหมดมอนิเตอร์ เพียงแต่ไม่สามารถแก้ไขสภาวะได้

จอภาพ LCD นั้นจะต่อกับ NIC-52 โดยผ่านทาง USER PORT และ PORT 1 CPU โดยสัญญาณควบคุม (E,RS,R/W) จะถูกส่งผ่าน USER PORT ทางขา PC_0 - PC_3 ของ 8255 (U19) และสัญญาณข้อมูล (DB_0 - DB_7) บนบอร์ดของหน่วยป้อนโปรแกรม ดังแสดงในรูปที่ 3.11

ส่วนทฤษฎีเกี่ยวกับขาสัญญาณของจอ LCD และการเซ็ตจอภาพ ตลอดจนการใช้งานจอภาพได้กล่าวรวมในบทที่ 2 แล้ว

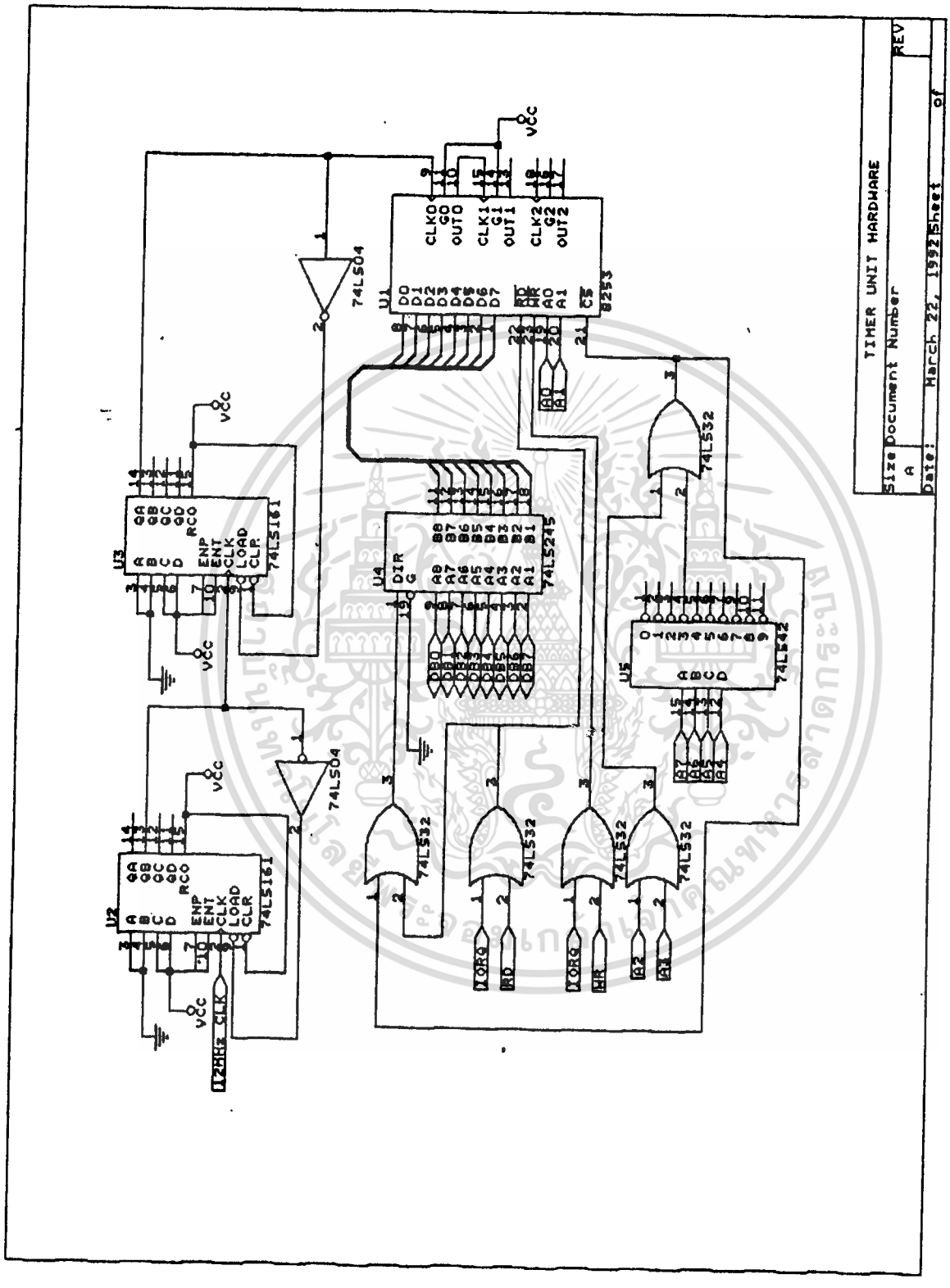


รูปที่ 3.12 แสดงตำแหน่งของคีย์ต่างๆ

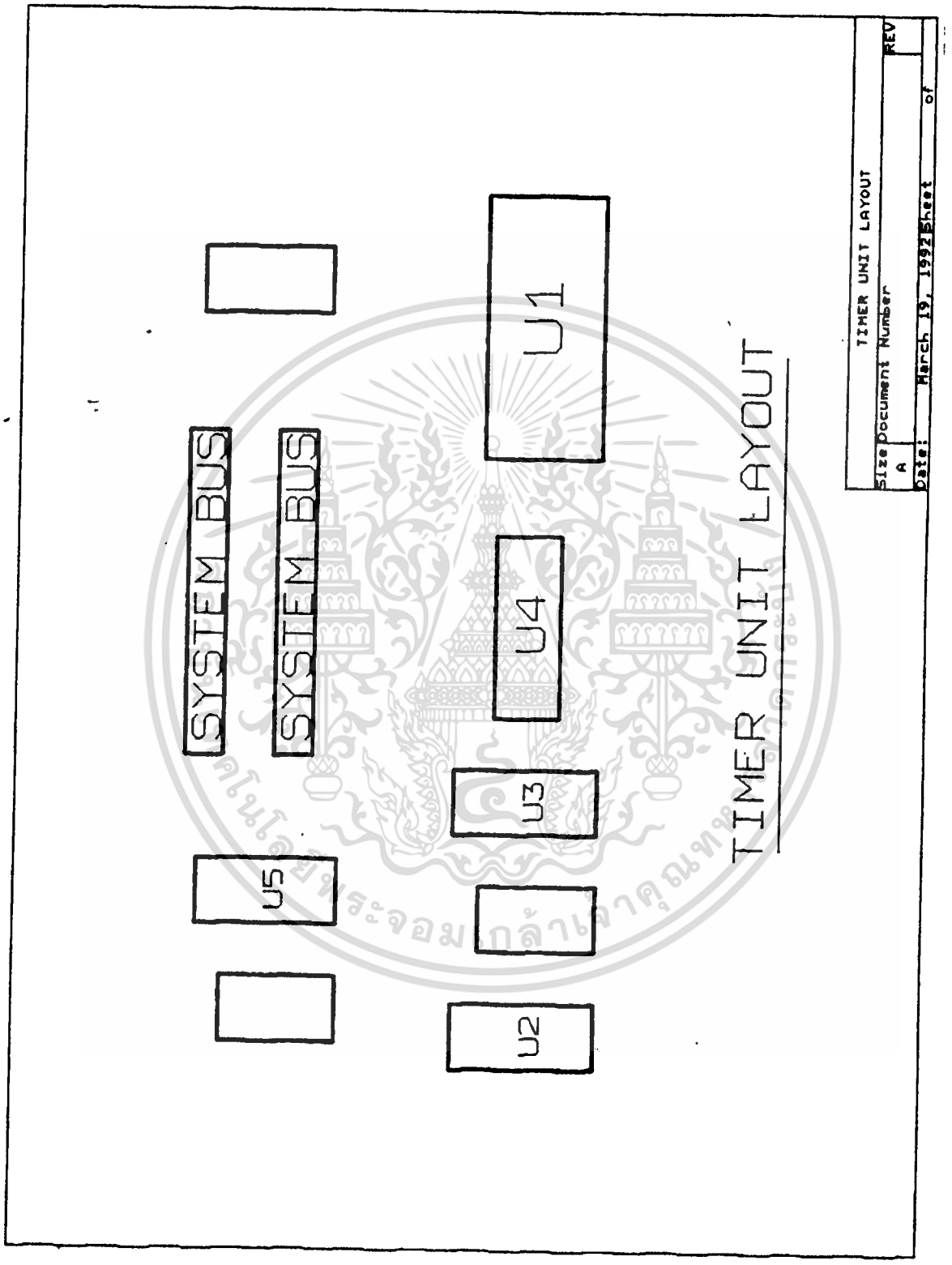
4.หน่วยนับเวลามาตราฐาน เป็นหน่วยพิเศษที่ถูกสร้างขึ้น เพื่อใช้ประโยชน์ในการสร้างคาบเวลามาตราฐาน สำหรับคำสั่งที่เกี่ยวกับตัวนับเวลา (TIMER) และฟังก์ชั่นที่เกี่ยวข้องกับชุดคำสั่งนี้ บอร์ดของหน่วยนับเวลามาตราฐานจะเชื่อมต่อกับบอร์ด CPU (NIC-52) โดยทาง SYSTEM BUS 40 เส้น แต่เนื่องจาก SYSTEM BUS ของ NIC-52 นั้น ผู้ออกแบบได้ออกแบบให้เป็น Z-80 COMPATIBLE (ดูได้จากรูปที่ 3.4) จึงต้องทำให้ต้องใช้ไอซีในตระกูลเดียวกันกับ Z-80 นั้นคือใช้ไอซีเบอร์ 8253 ดังแสดงรูปวงจรของหน่วยนับเวลามาตราฐานในรูปที่ 3.13

จากวงจรจะสังเกตุได้ว่ามีการใช้ไอซีเบอร์ 74LS161 ซึ่งเป็นวงจรหารความถี่ ที่ใช้เนื่องจากสัญญาณ CLOCK ที่จะให้แก่ 8253 นั้นจะต้องไม่เกิน 3.5 MHz ตามลักษณะของไอซีในตระกูลนี้ แต่ไอซีในตระกูล MCS-51 นั้นใช้สัญญาณ CLOCK 12 MHz ดังนั้นจึงต้องวงจรหารความถี่ของสัญญาณ CLOCK ขึ้นมา จากวงจรค่าสัญญาณ CLOCK ที่ออกมาจะได้ 1.59 MHz ซึ่งเพียงพอที่จะทำให้ 8253 ทำงาน

การทำงานของบอร์ดนี้ จะเริ่มรับคำสั่งให้เริ่มนับจาก CPU ตั้งแต่ $FFFF_H$ ลงไปถึง 0000_H เช่นนี้ไปเรื่อยๆ ในการนับ 1 รอบ ($FFFF_H$ - 0000_H) จะใช้เวลา 1 วินาที ดังนั้นในการนับย่อยๆ (เช่น $FFFF_H$ - $FFFE_H$) จะใช้เวลาเท่ากับ 0.1 ms. CPU จะเริ่มดึงเอาข้อมูลที่นับจากบอร์ดนี้ เมื่อโปรแกรมนั้นมีคำสั่ง TIM (หรืออื่นๆในชุดนี้) จากนั้น CPU จะค่อยๆดึงเอาข้อมูลต่อๆมาเข้าสู่ CPU เรื่อยๆ และค่อยๆนำข้อมูลเหล่านั้นมาเปรียบเทียบกับข้อมูลตัวแรกที่ CPU ได้เรียกเข้าไป ถ้าผลลัพธ์ได้เท่ากับค่าที่ต้องการนับแล้ว CPU ก็จะหยุดเรียกข้อมูลจากบอร์ดนั้นเข้ามา ซึ่งหมายความว่า คาบเวลาน้อยที่สุดที่ CPU จะนับได้จะเท่ากับ ผลต่างที่น้อยที่สุดของข้อมูลที่ CPU จะเรียกเข้าไปเปรียบเทียบกัน นั้นคือ 1 บิทของเลขฐาน 16 ซึ่งจะเท่ากับ 0.1 ms. ดังนั้นในชุดคำสั่ง TIM นั้นจะมีฐานเวลาคือ 0.1 ms.

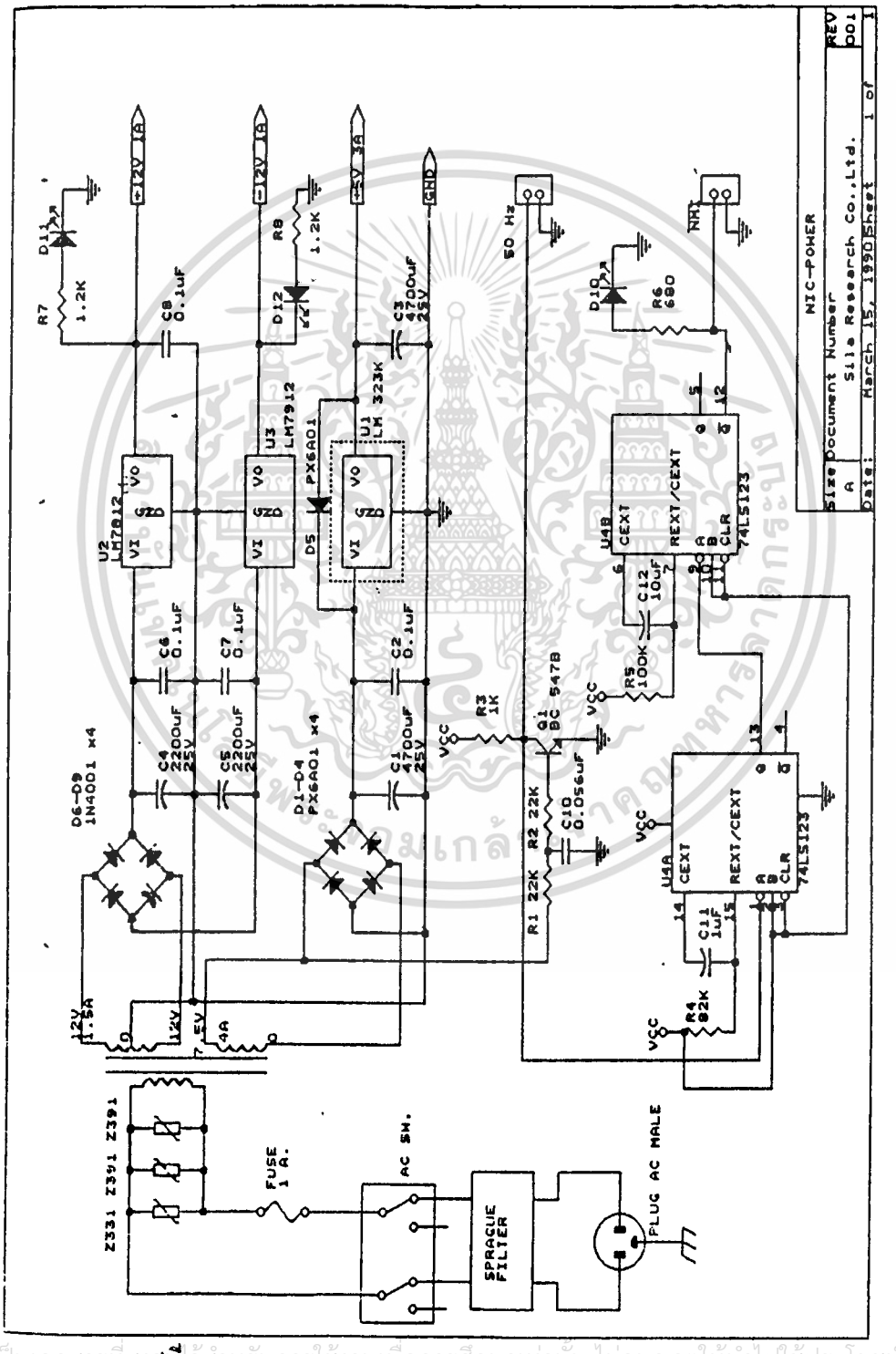


รูปที่ 3.13 แสดงวงจรของหน่วยนับเวลามาตราฐาน



รูปที่ 3.14 แสดง LAYOUT ของบอร์ดหน่วยนับเวลามาตราฐาน

5. หน่วยจ่ายพลังงาน ดังที่ได้กล่าวมาแล้วถึงหน้าที่ของหน่วยจ่ายพลังงาน ในการแปลงแรงดันไฟฟ้าจาก 220 V AC ให้เป็นแรงดัน 5 V DC เพื่อช่วยในการเลี้ยงวงจรและไอซีทุกตัว NIC-POWER ก็เป็นอุปกรณ์สำเร็จรูปอีกอุปกรณ์ที่ทางกลุ่มได้นำมาร่วมใช้กับบอร์ดทุกตัวในตระกูลนี้ วงจรของ NIC-POWER ได้แสดงไว้ในรูปที่ 3.15 แล้ว



รูปที่ 3.15 แสดงวงจรของ NIC-POWER

บทที่ 5 การออกแบบโปรแกรมบริหารระบบพีแอลซี

พีแอลซีที่ออกแบบเป็นเครื่องมือที่สามารถเปลี่ยนแปลงแก้ไข เงื่อนไขในการควบคุมได้ โดยผู้ใช้สามารถเขียนโปรแกรมผู้ใช้ดัง รายละเอียดที่ได้กล่าวมาแล้ว นอกจากนี้พีแอลซีที่มีโปรแกรมผู้ใช้อยู่แล้วจะต้องสามารถ ควบคุมการทำงานกับอุปกรณ์ภายนอกให้ได้ตามโปรแกรมที่เขียนมา อีกทั้งในระหว่างที่กำลังควบคุมการทำงานของกระบวนการอยู่นั้น ผู้ใช้ยังสามารถมอนิเตอร์ดูค่าสภาวะต่าง ๆ ได้

ทั้งหมดที่กล่าวมาแล้วข้างต้นนั้น พีแอลซี ทำได้อย่างไร คำถามนี้เองนำไปสู่แนวความคิดในการพัฒนา "โปรแกรมบริหารระบบพีแอลซี" ดังจะกล่าวถึงต่อไปนี้

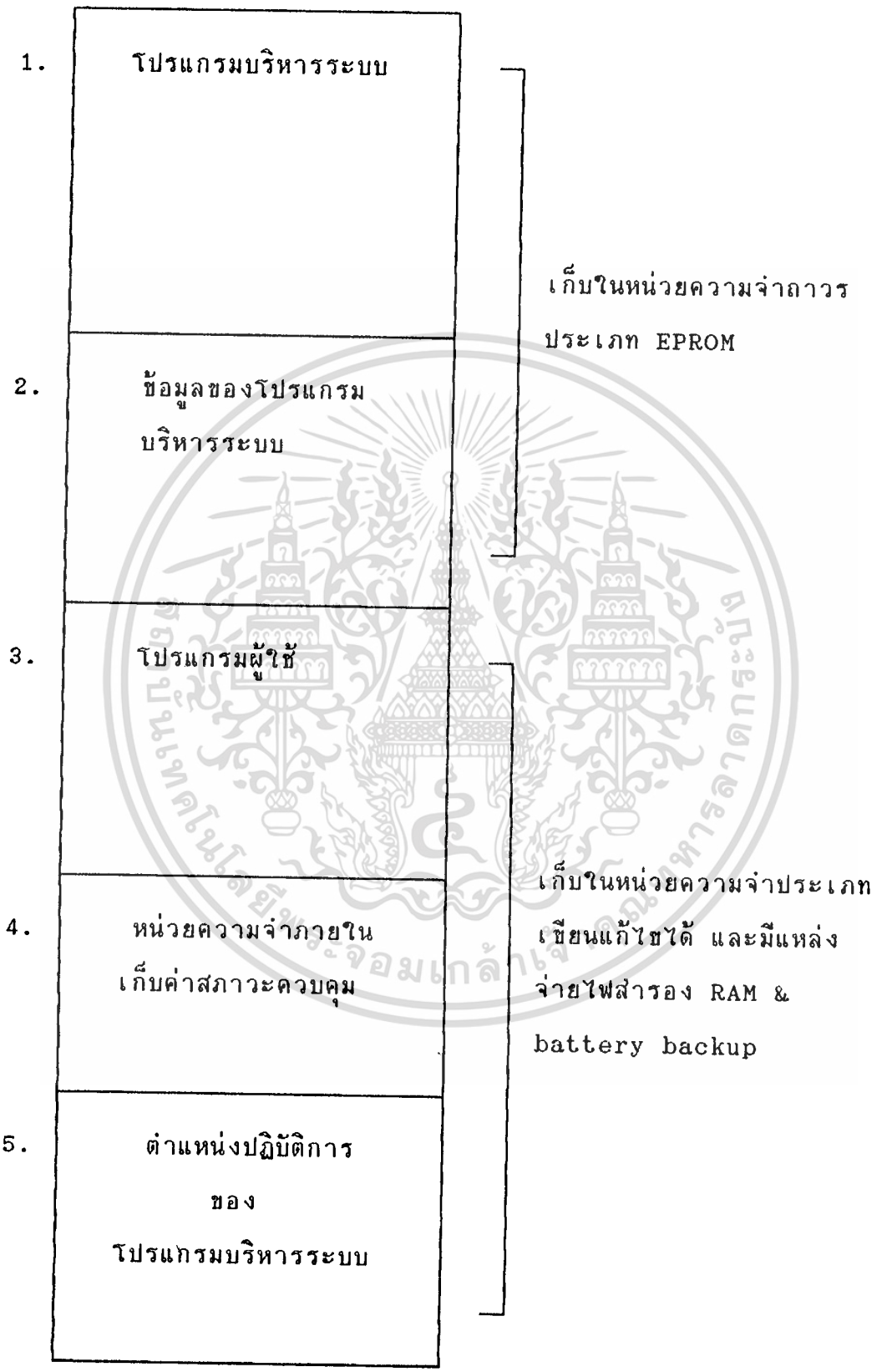
5.1 โปรแกรมบริหารระบบทำหน้าที่อะไรบ้าง

เราสามารถแบ่งแยกการทำงานของโปรแกรมบริหารระบบ ออกเป็นส่วน ๆ เพื่อสร้างแนวความคิดในการพัฒนาต่อไป ได้ดังนี้

1. จัดการเกี่ยวกับการป้อนโปรแกนของผู้ใช้
2. ตรวจสอบโปรแกรมผู้ใช้
3. จัดาการให้โปรแกรมผู้ใช้สามารถทงานได้ตามฟังก์ชั่นที่ออกแบบไว้
4. อ่านค่าสภาวะและเปลี่ยนแปลงค่าสภาวะนั้นได้

ก่อนที่จะกล่าวถึงรายละเอียดในการออกแบบโปรแกรม ขอทำความเข้าใจเกี่ยวกับการจัดการหน่วยความจำของพีแอลซี ดังต่อไปนี้

5.2 การจัดหน่วยความจำของพีแอลซี



รูปที่ 5.1 การจำหน่ายความจำของพีแอลซี (PLC MEMORY MAP)

จัดแบ่งหน่วยความจำออกเป็น 5 ส่วนคือ

1. โปรแกรมบริหารระบบ
2. ข้อมูลของโปรแกรมบริหารระบบ
3. โปรแกรมผู้ใช้
4. หน่วยความ่จำภายในเก็บค่าสภาวะควบคุม
5. ตำแหน่งปฏิบัติการของโปรแกรมบริหารระบบ

5.2.1 โปรแกรมบริหารระบบ

หน่วยความจำส่วนนนี้อยู่ที่ตำแหน่งแรกของหน่วยประมวลผลกลาง เป็นหัวใจหลักของการทำงานของเครื่อง เป็นส่วนทีเราพัฒนาขึ้นมา ทำหน้าที่บริหารควบคุมการทำงาน โปรแกรมผู้ใช้ อีกทีหนึ่ง 4.2.2 ข้อมูลของโปรแกรมบริหารระบบ เป็นข้อมูลที่โปรแกรมบริหารระบบอ่านไปเพื่อปฏิบัติการณ์ต่างๆ ส่วนใหญ่แล้วเป็นข้อมูลสำหรับการแสดงผลออกที่จอภาพแบบ แอลซีดี

5.2.3 โปรแกรมผู้ใช้

หน่วยความจำส่วนนี้จัดสรรไว้ให้เก็ยโปรแกรมผู้ใช้ที่เขียนขึ้น การจัดเก็บการปฏิบัติการณ์ตามโปรแกรมเป็นไปตามการควบคุมของโปรแกรมบริหารระบบ

5.2.4 หน่วยความจำภายในเก็บค่าสภาวะควบคุม

เก็บค่าสภาวะของ โอเปอร์แรนด์ ต่าง ๆ ได้แก่

-สภาวะ เปิด-ปิด ของ อินพุท เอาท์พุท ตัวตั้งเวลา ตัวนับ

-ค่าตั้งเวลา (Timert Set Value) ค่าตั้งตัวนับ (Counter Set Value) ค่าปัจจุบันของตัวตั้งเวลา (Running Time) , ค่าปัจจุบันของตัวนับ (Running Count)

5.2.5 ตำแหน่งปฏิบัติการของโปรแกรมบริหารระบบ

เป็นกลุ่มหน่วยความจำที่เป็นที่ปฏิบัติการของโปรแกรม เป็นสแตกของโปรแกรม เป็ฯบัฟเฟอร์ต่าง ๆ ทำหน้าที่ตั้งเงื่อนไขต่าง ๆ แก่โปรแกรม โปรแกรมบริหารระบบจะมีการอ่านและเขียนกับหน่วยความจำส่วนนี้ตลอดเวลา

5.3 การจัดการเกี่ยวกับการป้อนโปรแกรมของผู้ใช้

ในโหมดโปรแกรมการทำงานหลักของโปรแกรมบริหารระบบคือ การจัดการเกี่ยวกับการจัดเก็บโปรแกรมผู้ใช้ ตลอดจนการแก้ไขตรวจสอบโปรแกรมที่ผู้ใช้เขียนขึ้นแล้ว

แนวคิดในการออกแบบโปรแกรมบริหารระบบในส่วนที่ทำหน้าที่นี้ คือ

1. การตรวจสอบคีย์ที่กด (SCAN KEY AND EXECUTE)

2. วิธีการตรวจสอบกลุ่มคีย์ที่กดและจัดเก็บ โค๊ดคำสั่ง ลงในหน่วยความจำ ส่วนที่เป็นโปรแกรมผู้ใช้

5.4 การตรวจสอบโปรแกรมผู้ใช้

ระหว่างที่เรากำลังเขียนโปรแกรมผู้ใช้อยู่ และทันทีที่เลื่อนสวิตช์เลือกโหมดการทำงานไปที่โหมดมิเตอร์หรือโหมดรัน โปรแกรมผู้ใช้ที่เราเขียน จะถูกตรวจสอบโดย โปรแกรม บริหารระบบ เพื่อป้องกันโปรแกรมที่ผู้ใช้ที่ผิดพลาด

5.5 การอ่านค่าสภาวะและเปลี่ยนแปลงค่าสภาวะนั้นได้

ในโหมดมอนิเตอร์ นอกจากการจัดการให้โปรแกรมผู้ใช้ให้สามารถทำงานตามที่ออกแบบไว้ด้วย โปรแกรมบริการระบบ ยังคอยควบคุมการอ่านค่าสภาวะของกระบวนการ ตลอดจนการเปลี่ยนแปลงค่าสภาวะด้วย

5.6 การจัดการให้โปรแกรมผู้ใช้สามารถทำงานได้ตามฟังก์ชันที่ออกแบบไว้

ในโหมดมอนิเตอร์และโหมดรัน โปรแกรมบริหารระบบ จะอ่านโค๊ดคำสั่งจากโปรแกรมผู้ใช้ และให้ทำงานตามที่ได้ออกแบบไว้

บทที่ 6
วิธีการใช้งานเครื่อง PLC เบื้องต้น

ในการนำเอา PLC ไปใช้งานนั้น จะประกอบด้วยงานใหญ่ๆ 3 งาน คือ งานการป้อนโปรแกรม งานการตรวจสอบ แก้ไข และทดลองการทำงานตามโปรแกรมนั้นๆ และการทำงานจริง จึงเป็นเหตุให้การออกแบบ PLC เพื่อการนำไปใช้งาน จึ้งประกอบด้วย โหมดการทำงาน 3 โหมด คือ

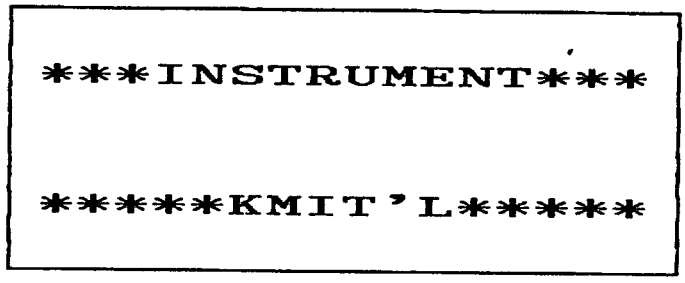
1. โหมดโปรแกรม : ในโหมดการทำงานนี้ ผู้ใช้งานสามารถป้อนชุดคำสั่ง กำหนดเงื่อนไขการทำงานของกระบวนการต่างๆ ทั้งยังสามารถตรวจสอบและแก้ไขคำสั่งได้

2. โหมดมอนิเตอร์ : ในโหมดนี้ เป็นโหมดที่สามารถตรวจสอบสภาวะการทำงานต่างๆของระบบที่กำลังทำงานตามชุดคำสั่งที่ป้อนเข้าไปแล้วได้ นอกจากนั้น ยังสามารถเปลี่ยนแปลงสถานะของอินพุท, เอาท์พุท, ค่าของตัวตั้งเวลาและตัวนับได้อีกด้วย

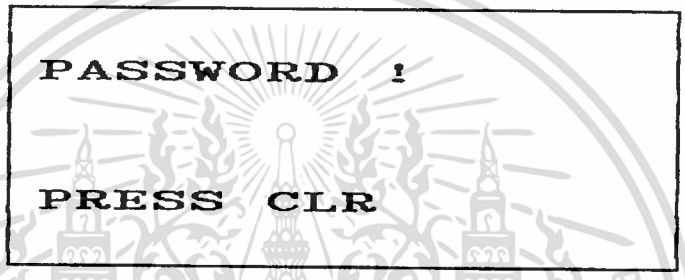
3. โหมดรัน : เป็นโหมดที่ใช้เมื่อไม่ต้องการตรวจสอบ หรือ แก้ไขโปรแกรมอีกแล้ว เครื่อง PLC ก็จะทำงานตามชุดคำสั่งที่ป้อนเข้าไป

6.1 การเริ่มต้นการใช้

หลังจากเสียบสายหน่วยป้อนโปรแกรมเข้ากับเครื่องแล้ว ทันทีที่กดปุ่ม RESET เพื่อให้ PLC เริ่มต้นทำโปรแกรมบริหารระบบใหม่อีกครั้ง จอ LCD จะแสดงภาพดังนี้



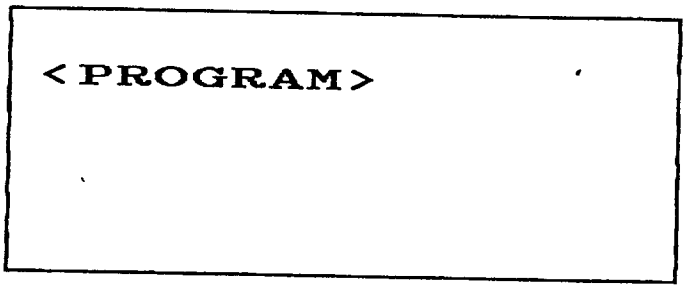
กดปุ่มใดๆ เพื่อทำการใส่ PASSWORD จอภาพจะแสดงภาพดังนี้



จากนั้นจึงกดปุ่ม CLR เพื่อทำงาน CLEAR จอ LCD จะแสดงให้ทราบว่าขณะนี้เครื่อง PLC ทำงานในโหมดใด เช่น สวิทช์เลือกโหมดโปรแกรมจอ LCD จะแสดง

6.2 การป้อนโปรแกรม ในโหมดโปรแกรม

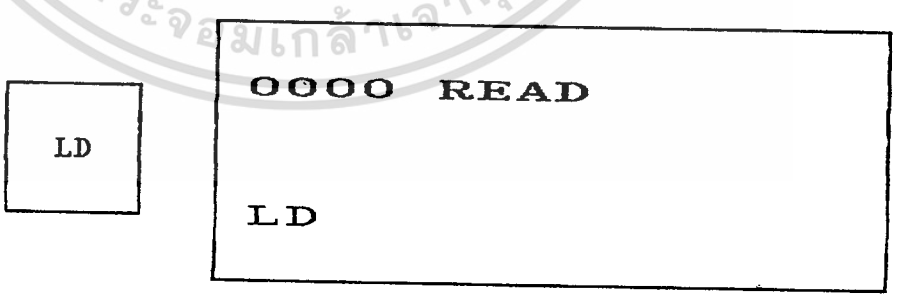
หลังจากกดคีย์ CLR เพื่อเข้าสู่โหมดใดๆแล้ว เมื่อต้องการป้อนโปรแกรมก็ให้เลื่อนสวิทช์ไปที่ P จอภาพก็จะแสดง

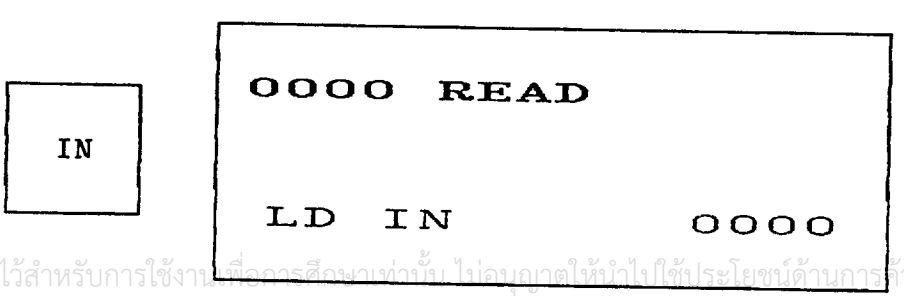


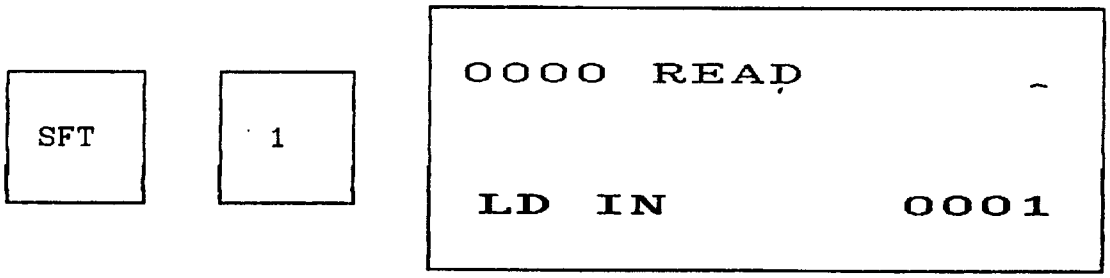
กดปุ่ม CLR เพื่อเริ่มการโปรแกรม จอภาพจะแสดงโปรแกรมที่แอดเดรส 0000 ถ้าไม่มีอะไรเลยแสดงว่า ไม่มีโปรแกรมที่แอดเดรสนี้

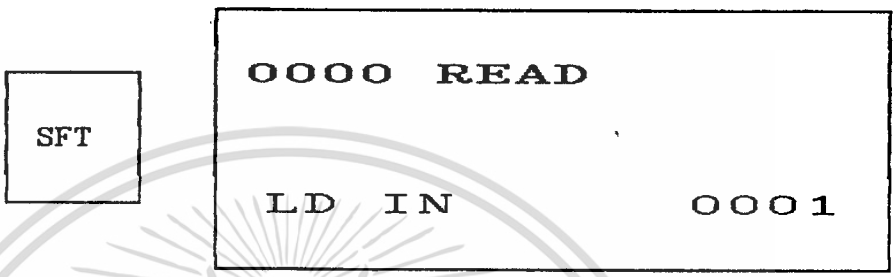
จากนั้นก็สามารถเริ่มต้นทำการป้อนโปรแกรมได้

ตัวอย่างที่ 6.1 ต้องการป้อนคำสั่งต่อไปนี้ LD IN 0001 มีขั้นตอนดังนี้









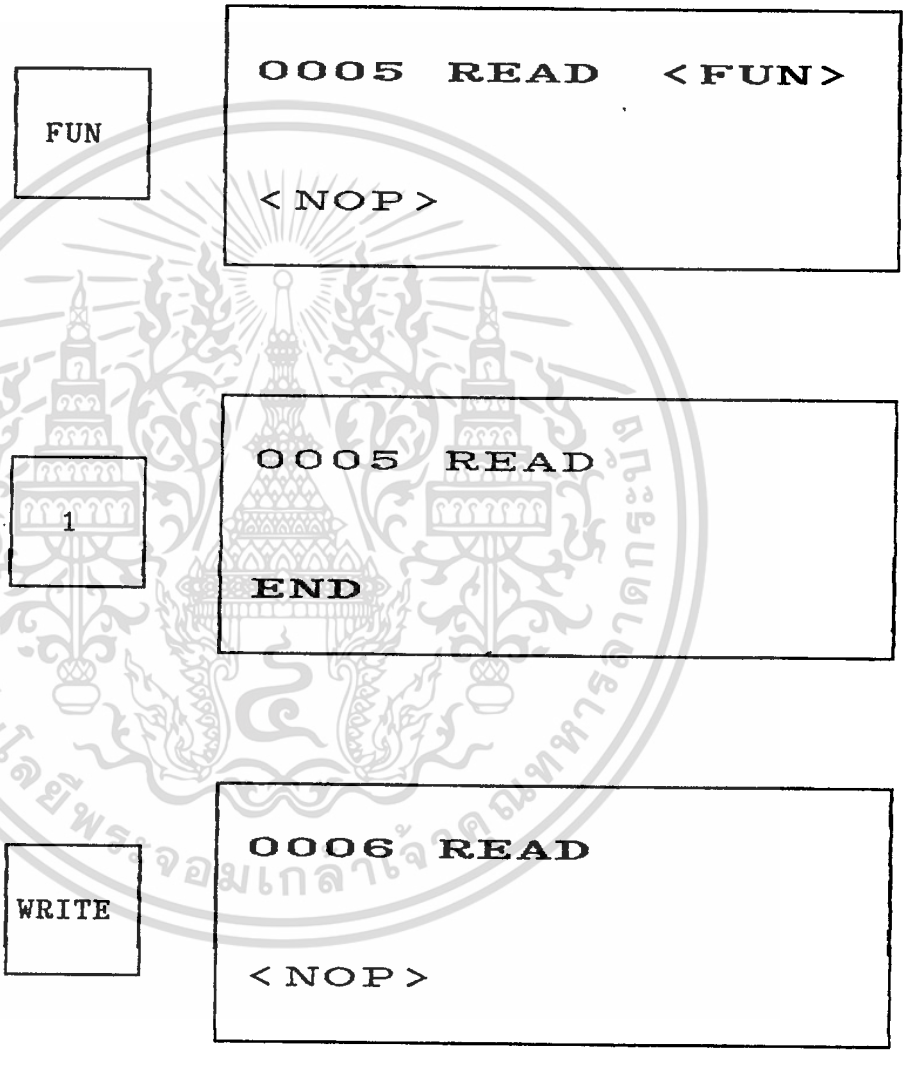
หมายเหตุ 1. ก่อนการป้อนตัวเลขทุกครั้ง ไม่ว่าในกรณีใดก็ตาม จะต้องมีการกด SFT ทุกครั้ง และเมื่อมีการกด SFT แล้วจะมีสัญลักษณ์ _ ปรากฏขึ้นมา (ถ้าต้องการป้อนตัวเลขตัวถัดไป ต้องกดคีย์ SFT แม้จะมีสัญลักษณ์ _ ปรากฏขึ้นแล้วก็ตาม) และเมื่อมีการป้อนตัวเลขเรียบร้อยแล้ว ต้องมียกเลิกการป้อนตัวเลขก่อน โดยการกด SFT อีกครั้ง สัญลักษณ์ _ ก็จะหายไป

2. เมื่อป้อนโปรแกรมในแอดเดรสนั้น จะต้องมีการกดคีย์ WRITE ทุกครั้ง เพื่อให้เครื่องนำโปรแกรมที่แสดงเก็บเข้าไปในหน่วยความจำ

3. เมื่อกดคีย์ WRITE แล้ว เครื่องก็จะแสดงโปรแกรมในแอดเดรสถัดไปเองโดยอัตโนมัติ (สำหรับแอดเดรสอื่น ยกเว้นการเรียกแอดเดรส 0000 ครั้งแรก ถ้าไม่มีโปรแกรม จอภาพจะแสดงคำว่า

<NOP> ทุกครั้ง)

4. มีคำสั่งบางคำสั่งที่ไม่แสดงบนหน้าปัด เป็นคำสั่งที่อยู่ในรูป FUN คือ กดคีย์ FUN แล้วตามด้วยตัวเลข ไม่ต้องกดคีย์ SFT ก่อนการป้อนตัวเลข เช่น ถ้าต้องการจบโปรแกรม ต้องป้อนคำสั่ง END (FUN1) ทำได้โดย



5. เมื่อสิ้นสุดโปรแกรมแล้ว จะต้องมีคำสั่ง END ต่อท้ายโปรแกรมด้วยทุกครั้ง มิฉะนั้น เมื่อเปลี่ยนโหมดการทำงาน เครื่อง PLC จะไม่ทำงานและจอภาพจะแสดง

NO END
INSTR ERROR

6.ในการใช้ TIMER และ COUNTER รวมทั้ง REVERSABLE COUNTER ทุกโหมดจะใช้ช่องเดียวกัน ในโปรแกรมเดียวกัน สามารถเรียกช่องดังกล่าวได้เพียง 1 ครั้งเท่านั้น และห้ามเขียนค่าตั้งเวลาหรือค่าการนับ 2 ครั้ง ในช่องเดียวกัน

6.3 การตรวจโปรแกรม

สามารถทำได้โดยการใช้คำสั่ง ▵ และ ▿ เพิ่มและลดทีละแอดเดรส จอภาพก็จะแสดงโปรแกรมที่เขียนในแต่ละแอดเดรสออกมาให้ตรวจสอบ

6.4 การแก้ไขโปรแกรม

สามารถทำได้ 3 แบบ คือ

1.การเขียนทับโปรแกรมเดิม โดยใช้คำสั่ง WRITE เมื่อต้องการป้อนโปรแกรมทับที่แอดเดรสใด ก็ให้เรียกแอดเดรสนั้นออกมาตรวจสอบ จากนั้นจึงป้อนโปรแกรมใหม่ลงไป แล้วกดคีย์ WRITE โปรแกรมใหม่ก็จะเขียนโปรแกรมเก่า

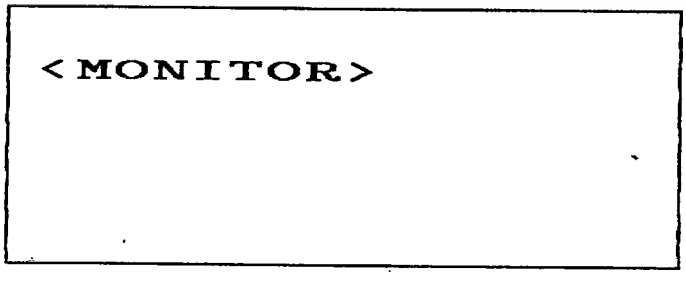
2.การลบโปรแกรมในแอดเดรสนั้น ทำโดยการเรียกโปรแกรมที่แอดเดรสนั้นออกมา แล้วกดคีย์ DEL โปรแกรมในแอดเดรสนั้นก็จะถูกลบทิ้งและโปรแกรมในแอดเดรสถัดมาก็จะเลื่อนขึ้นมาตามลำดับเรื่อยๆ

3. การเขียนโปรแกรมแทรกก่อนโปรแกรมในแอดเดรสนั้น ทำโดยการเรียกโปรแกรมที่ต้องการเขียนโปรแกรมใหม่แทนโปรแกรมในแอดเดรสนี้ แล้วเขียนโปรแกรมใหม่ทับลงไป แล้วจึงกดคีย์ INS โปรแกรมใหม่ที่เขียนลงไปก็จะถูกเขียนลงไปยังหน่วยความจำ ณ แอดเดรสนั้น ส่วนโปรแกรมที่อยู่ในแอดเดรสเดิม ก็จะถูกเลื่อนลงไปอีก 1 แอดเดรส

6.5 การเริ่มต้นการทำงาน

หลังจากที่ได้ทำการป้อนโปรแกรม ในโหมดโปรแกรมเรียบร้อยแล้ว เมื่อต้องการให้เครื่อง PLC ทำงานตามโปรแกรมก็สามารถทำได้โดย เลื่อนสวิทช์เลือกโหมดการทำงานไปยังที่ M จอ LCD ก็จะแสดงดังรูป

จากนั้นให้กดคีย์ CLR เพื่อทำการเคลียร์หน้าจอและเข้าสู่โหมดมอนิเตอร์ จอ LCD ก็จะแสดงดังรูป



จากนั้นให้ทำการกดคีย์ OUT ติดต่อการประมาณ 2-3 ครั้ง และให้สังเกตุไฟ LED

บนบอร์ด CPU ที่แสดง USER 1 และ USER 2 ว่าเริ่มต้นกระพริบหรือไม่ ถ้าเริ่มต้นแสดงว่า เครื่องเริ่มต้นโหลดโปรแกรมจากหน่วยความจำเพื่อนำไปประมวลผลแล้ว ไฟ LED ทั้งสองดวงจะกระพริบตลอดการทำงาน ถ้าไฟทั้งสองเกิดการติดค้างไว้ตลอดหรือเกิดดับขึ้น แสดงว่าเกิดการแฮงค์ของเครื่อง PLC แล้ว ให้ทำการรีเซ็ตโปรแกรมใหม่ทั้งหมด

บทที่ 7
คำสั่งและการใช้

คำสั่งเบื้องต้น

1. คำสั่ง LD

คำสั่งนี้มักจะใช้เป็นคำสั่งแรกของชุดคำสั่งแต่ละชุด และตามด้วยโอเปอร์แรนด์ตัวเลขที่ใช้ในการประมวลผลของชุดคำสั่งนั้น

2. คำสั่ง AND

คำสั่งนั้หมายถึงการต่ออนุกรมกันระหว่างผลลัพธ์ทางลอจิก ของชุดคำสั่งหรือโอเปอร์แรนด์ก่อนหน้านี้ กับโอเปอร์แรนด์ของคำสั่งนี้

3. คำสั่ง OR

คำสั่งนี้แสดงถึงการตค่อขนานกันระหว่างผลลัพธ์ทางลอจิก ของชุดคำสั่งหรือโอเปอร์แรนด์ ก่อนหน้านี้กับโอเปอร์แรนด์ของคำสั่งนี้

4. คำสั่ง OUT

คำสั่งนี้จะเป็นการแสดงผลลัพธ์ของการปฏิบัติการทางลอจิกทั้งหมดที่เกิดขึ้น

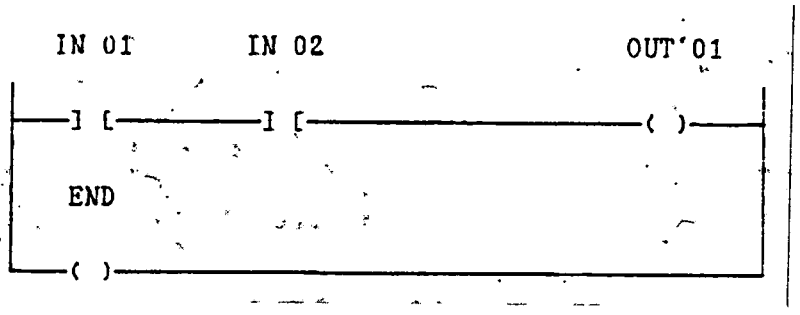
5. คำสั่ง NOT

ตามปกติคำสั่งนี้เป็นการแสดงความเป็นนิเสธของ โอเปอร์แรนด์ แต่ถ้าอยู่หลัง คีย์ [OUT] (OUT NOT) จะการแสดงผลลัพธ์ที่เป็นนิเสธกับผลลัพธ์ที่เกิดขึ้นจริง

6. คำสั่ง END

เป็นคำสั่งแสดงถึง การสิ้นสุดโปรแกรม จะต้องป้อนเข้าเป็นคำสั่งสุดท้ายในทุกครั้งที่เขียน โปรแกรมผู้ใช้ ถ้าไม่มีคำสั่งนี้ พีแอลซีจะไม่สามารถทำงานได้

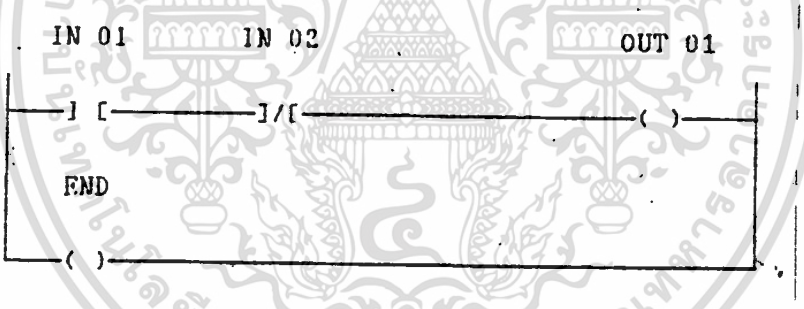
ตัวอย่างที่ 7.1



โปรแกรมคำสั่ง
```
LD   IN   0001
AND IN   0002
OUT      0001
END
```

ความหมาย เอาท์พุท 0001 จะทำงาน(Active) เมื่ออินพุท 0001 และอินพุท 0002 ทำงานพร้อมกัน

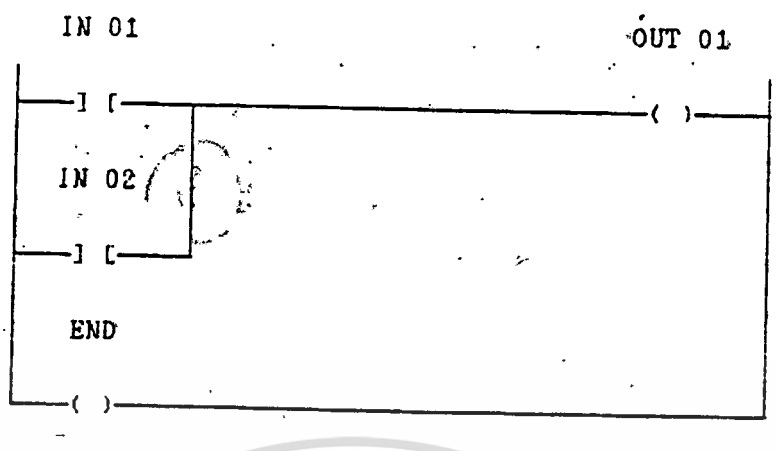
ตัวอย่างที่ 7.2



โปรแกรมคำสั่ง
```
LD   IN      0001
AND IN NOT 0002
OUT          0001
END
```

ความหมาย เอาท์พุท 0001 จะทำงาน(Active) เมื่ออินพุท 0001 ทำงานและอินพุท 0002 ไม่ทำงาน พร้อม ๆ กัน

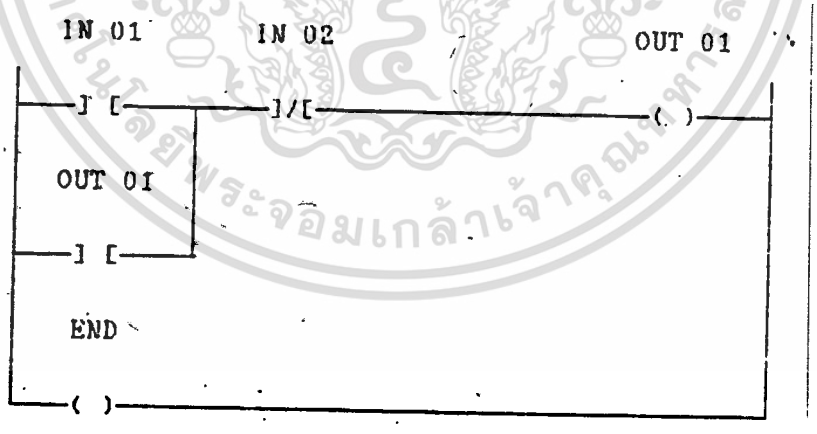
ตัวอย่างที่ 7.3



โปรแกรมคำสั่ง
```
LD   IN   0001
OR   IN   0002
OUT       0001
END
```

ความหมาย เอาท์พุท 0001 จะทำงาน(Active) เมื่ออินพุท 0001 หรือ อินพุท 0002 ตัวใดตัวหนึ่งทำงาน

ตัวอย่างที่ 7.4

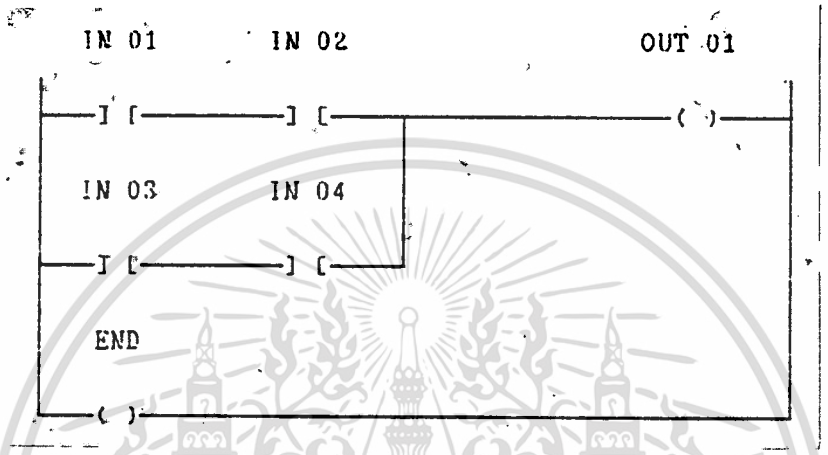


โปรแกรมคำสั่ง
```
LD   IN       0001
OR   OUT      0001
AND  IN NOT   0002
OUT           0001
END
```

ความหมาย เอาท์พุท 0001 จะทำงาน(Active) เมื่ออินพุท

0001 ทำงานนานเพียง 1 คาบเวลา สแกนและอินพุท
0002 ยังไม่ทำงาน OUT 0001 ยังคงทำงานต่อไป
เพราะมีสัญญาณ OUT 0001 หรืออยู่กับ IN 0001
และจัดตัดการทำงานเมื่อ IN 0002 ทำงาน

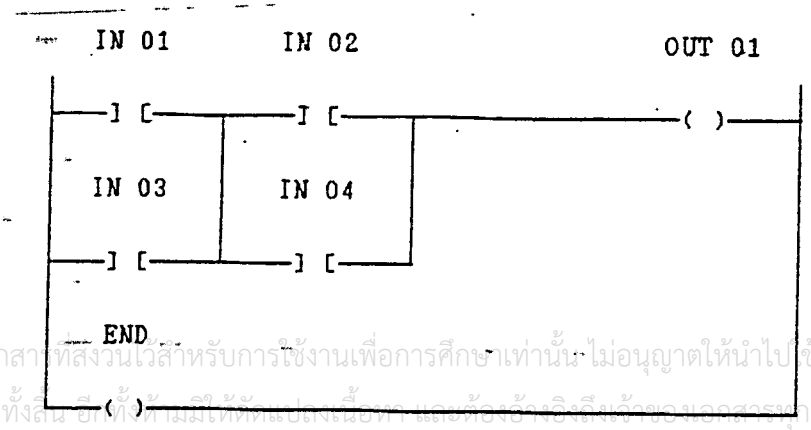
ตัวอย่างที่ 7.5



โปรแกรมคำสั่ง
```
LD   IN  0001
AND IN  0002
LD   IN  0003
AND IN  0004
OR   LD
OUT      0001
END
```

ความหมาย OUT 01 = (IN01*IN02)+(IN03*IN04)

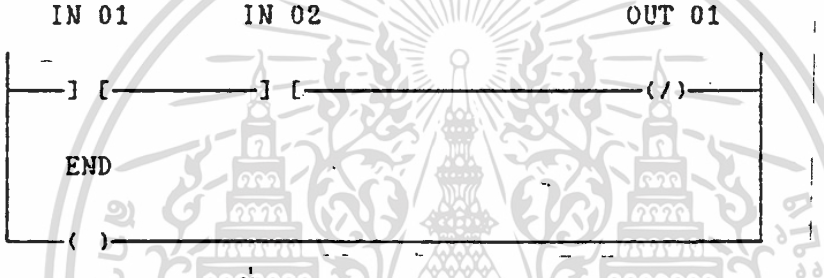
ตัวอย่างที่ 7.6



โปรแกรมคำสั่ง LD IN 0001
OR IN 0002
LD IN 0003
OR IN 0004
AND LD
OUT 0001
END

ความหมาย OUT 01 = (IN01D+IN02)*(IN03+IN04)

ตัวอย่างที่ 7.7



โปรแกรมคำสั่ง LD IN 0001
AND IN 0002
OUT NOT 0001
END

ความหมาย เอาท์พุท 0001 จะไม่ทำงาน (cut-off)
เมื่ออินพุท 0001 และอินพุท 0002 ทำงานพร้อมกัน

บทที่ 8

บทสรุปและปัญหาที่พบ

8.1. สรุปคุณลักษณะของเครื่องพีแอลซีตามโครงงาน

1.แหล่งจ่ายไฟ ใช้ไฟฟ้ากระแสสลับ 220 V ความถี่ 50 Hz สามารถจ่ายกระแสได้สูงสุด 3 A โดยใช้บอร์ดแหล่งจ่ายไฟสำเร็จรูป NIC-POWER ของบริษัทศิลา รีเสริช จำกัด

2.อินพุท แบ่งเป็นอินพุทภายนอก 8 ตัวและอินพุทภายในอีก 24 ตัว ไฟเลี้ยงสำหรับอินพุทเป็นไฟฟ้ากระแสตรง 24 V 1 A บอร์ดอินพุทภายนอกเป็นบอร์ดสำเร็จรูป NIC-OPTO ของบริษัทศิลา รีเสริช จำกัด

3.เอาท์พุท จำนวนเอาท์พุทใช้งาน เป็นเอาท์พุทภายนอกชนิด SSR (SOLID STATE RELAY) จำนวน 8 เอาท์พุท นอกจากนั้นยังมีเอาท์พุทภายในอีก 24 ตัวไว้ใช้งานตามความต้องการของผู้ใช้ โดยเอาท์พุทภายนอกจะใช้บอร์ดเอาท์พุทสำเร็จรูป NIC-SSR ของบริษัทศิลา รีเสริช จำกัด ซึ่งบอร์ดนี้สามารถขับสัญญาณเอาท์พุทได้สูงสุด 1 A

4.หน่วยประมวลผลกลาง หน่วยประมวลผลกลางของพีแอลซีตามโครงงานนี้ใช้ไมโครโปรเซสเซอร์ตระกูล MCS-51 เบอร์ 8052AH เป็นคอนโทรลเลอร์ โดยทางกลุ่มใช้บอร์ดประมวลผลกลางสำเร็จรูป NIC-52 ของบริษัทศิลา รีเสริช จำกัด ซึ่งบนบอร์ด NIC-52 นี้มีหน่วยความจำรมขนาด 32 K และพอร์ทใช้งาน สำหรับผู้ใช้จะขยายระบบพีแอลซีต่อไป

5.ตัวตั้งเวลา จำนวนตัวตั้งเวลามีทั้งหมด 32 ตัว โดยตัวตั้งเวลาของเครื่องพีแอลซีแบ่งเป็น 4 ชนิด คือ

- หน่วงเวลาเปิด
- หน่วงเวลาปิด
- เปิดเวลาคงที่ 1 จังหวะ
- เปิดเวลาคงที่ 1 จังหวะแบบทริกใหม่ได้

ค่าตั้งเวลาต่ำสุด - สูงสุด คือ 0.1 - 999.9 วินาที

6.ตัวนับ จำนวนตัวนับมีทั้งหมด 32 ตัว โดยแบ่งเป็น 2 ชนิด คือ ตัวนับขึ้นและตัวนับลง สำหรับค่าตัวนับต่ำสุด - สูงสุด 1-9999

7.โปรแกรมผู้ใช้ จำนวนบรรทัด (แอดเดรส) ใช้งานสูงสุด 250 บรรทัดสูงสุด

8.2.ปัญหาที่พบในโครงงานนี้

ในการทำโครงงานเครื่อง PLC โดยใช้ไมโครโปรเซสเซอร์ตระกูล MCS-51 เบอร์ 8052AH เป็นตัวคอนโทรลเลอร์นี้ ประสบปัญหาอย่างมากในการจัดการโปรแกรมบริหารระบบ ทั้งนี้เพราะไมโครโปรเซสเซอร์ 8052AH มีขีดความสามารถจำกัดในการจัดการเกี่ยวกับคู่รีจิสเตอร์ ทั้งนี้เพราะในไมโครโปรเซสเซอร์ 8052AH ประกอบด้วยคู่รีจิสเตอร์ใช้งานได้เพียงคู่เดียว คือ DATA POINTER (DPTR) แต่เนื่องจากการเขียนโปรแกรมบริหารระบบเพื่อให้เครื่อง PLC ทำงานตามลำดับขั้นตอน และสามารถแก้ไขปรับปรุงโปรแกรมผู้ใช้ (USER PROGRAM) ได้ในขณะเวลาใดๆจำเป็นต้องมีการจัดระบบหน่วยความจำของเครื่องที่ดี คือ ในขณะที่เครื่อง PLC กำลังทำงานจะมีการติดต่อใช้งานหน่วยความจำตลอดเวลา และเนื่องจากต้องมีการติดต่อใช้งานหน่วยความจำตลอดเวลานี้เองที่ต้องทำให้โปรแกรมบริหารระบบมีการใช้งาน DPTR ตลอดเวลา ในการการหาค่า OFFSET เพิ่ม ลด หรือโอนย้ายข้อมูลหรือพักข้อมูลตลอดเวลา ซึ่งไมโครโปรเซสเซอร์ 8052AH และไมโครโปรเซสเซอร์ในตระกูล MCS-51 มีขีดความสามารถจำกัดในด้านนี้เพราะมีคู่รีจิสเตอร์ไว้ใช้งานเพียงคู่เดียว คือ DPTR นอกจากจุดอ่อนในด้านคู่รีจิสเตอร์ใช้งานแล้ว ไมโครโปรเซสเซอร์ 8052AH ยังมีแฟลก (FLAG) ไม่เพียงพอต่อการใช้งาน ในโครงงานเครื่อง PLC ชิ้นนี้ได้ใช้รีจิสเตอร์ PSW (PROGRAM STATUS WORD) เป็นแฟลกในการใช้งาน แต่เนื่องจากจำนวนบิทที่สามารถใช้งานได้ในรีจิสเตอร์ PSW นั้นมีเพียง 2 บิท คือ PSW.7 และ PSW.5 ซึ่งทั้ง 2 บิทนี้ยังแบ่งการทำงานออกเป็น 2 ระบบ คือ เซ็ทและรีเซ็ทได้ด้วยตัวมันเองเมื่อทำงานตามอ๊อปโค้ด (OPCODE) และอีกแบบหนึ่ง คือ เซ็ทและรีเซ็ทได้โดยการบังคับของผู้ใช้ ดังนั้นจะเห็นได้ว่าเพื่อที่จะรักษาสถานะของ

แฟลกใดๆนั้น จะต้องมีการเก็บและเรียกใช้ค่าจาก STACK POINTER ตลอดเวลา สำหรับข้อเสียที่สำคัญอีกประการหนึ่งของ ไมโครโปรเซสเซอร์ 8052AH คือ การทำงานในคำสั่งใดๆก็ตามในการบวกลบพีชคณิต การกระทำทางลอจิก การโอนย้ายข้อมูลเข้าออกจากหน่วยความจำ การโอนย้ายข้อมูลระหว่างรีจิสเตอร์ จะต้องกระทำโดยใช้รีจีสเตอร์ A (ACCUMULATOR) เป็นทางผ่านหรือตัวกลางเสมอ การเขียนโปรแกรมบริหารเครื่อง PLC โดยใช้ภาษาแอสเซมบลี้ของไมโครโปรเซสเซอร์ 8052AH จึงไม่มีความคล่องตัวเลย นอกจากนั้นฟังก์ชั่นการทำงานบางตัวที่มีในไมโครโปรเซสเซอร์เบอร์อื่นๆในตระกูลคอนโทรลเลอร์เบอร์อื่นๆ ก็ไม่มีคำสั่งของไมโครโปรเซสเซอร์ รีจีสเตอร์ ทางผู้จัดทำจึงต้องสร้างคำสั่งใช้งานเทียม (PSEUDO INSTRUTION SET) ที่สร้างขึ้นเป็นรูทีน (ROUTINE) การทำงานที่ต้องเรียกใช้งานในคำสั่ง LCALL เมื่อต้องใช้งานในแต่ละครั้ง เวลาของโปรแกรมบริหารระบบจึงเสียไปกับการเรียกใช้รูทีนเหล่านี้เป็นส่วนมาก นอกจากนี้ยังมีขีดจำกัดทางด้านฮาร์ดแวร์ในการใช้งานของพอร์ตไมโครโปรเซสเซอร์ 8052 AH ทั้งนี้พอร์ตใช้งานจริงๆคือ PORT 1 เพียงพอร์ตเดียวและพอร์ตของไมโครโปรเซสเซอร์ 8052AH ยังไม่สามารถกำหนดเป็นอินพุทพอร์ตหรือเอาท์พุทพอร์ตได้ด้วยซอฟท์แวร์ เพราะพอร์ตสามารถเป็นได้ทั้ง SINK และ SOURCE โดยขึ้นอยู่กับกระแสไฟฟ้าภายนอกว่ามีปริมาณมากกว่าหรือน้อยกว่ากระแสไฟฟ้าของพอร์ต ถ้าหากกระแสไฟฟ้าของอุปกรณ์ภายนอกที่ต่อร่วมกับพอร์ตมีปริมาณมากกว่ากระแสไฟฟ้าที่พอร์ตสามารถจ่ายได้ พอร์ตก็จะทำตัวเป็น SINK ให้กระแสไฟภายนอกไหลเข้ามาในพอร์ต ในลักษณะนี้ พอร์ตจะกลายเป็นอินพุทพอร์ตโดยไม่สามารถบังคับได้โดยซอฟท์แวร์ แต่ในทางตรงกันข้าม ถ้ากระแสไฟของอุปกรณ์ภายนอกที่ต่อร่วมกับพอร์ตมีปริมาณน้อยกว่ากระแสไฟที่พอร์ตสามารถจ่ายได้ พอร์ตก็จะทำตัวเป็น SOURCE จ่ายกระแสไฟออกไปยังภายนอก พอร์ตก็จะกลายเป็นเอาท์พุทพอร์ต

จากการทำโครงงานเครื่อง PLC โดยใช้ไมโครโปรเซสเซอร์ 8052 AH ซึ่งเป็นไมโครคอนโทรลเลอร์ที่ได้รับความนิยมอยู่ในขณะนี้เป็นตัวควบคุมการทำงานของระบบ PLC นั้น จึงไม่เหมาะเพราะไม่มีความคล่องตัวในการใช้งานในระบบใหญ่ๆอย่างเครื่อง PLC เลย ทั้งนี้เนื่องจากขีดจำกัดทางซอฟท์แวร์และ

ฮาร์ดแวร์ เช่น คำสั่งปฏิบัติการต่างๆและรีจีสเตอร์ใช้งานต่างๆดังที่ได้กล่าวมาแล้ว ทางกลุ่มจึงหวังว่า ทางคณะกรรมการจะรับไว้พิจารณาเพื่อใช้เป็นแนวทางในการกำหนดหัวข้อโปรเจคสำหรับนักศึกษารุ่นต่อๆไป

ผู้จัดทำ

กิตติกรรมประกาศ

โครงงานเครื่อง PLC ชิ้นนี้ คงจะไม่สามารถดำเนินไปได้อย่างราบรื่นอย่างแน่นอน ถ้าปราศจากความช่วยเหลือและคำแนะนำอันทรงคุณค่าจากบุคคลเหล่านี้อันได้แก่ **อาจารย์วิริยะ กองรัตน์** **อาจารย์พิพัฒน์ เลาหสงคราม** **อาจารย์ประภาส อุคคกิมาพันธุ์** และที่สำคัญที่สุด ท่านอาจารย์ที่ปรึกษาของเรา **อาจารย์สุพรรณ กุลพาณิชย์** ซึ่งทางกลุ่มขอขอบพระคุณท่านเหล่านี้ไว้ ณ ที่นี้ด้วย

สุทินขอบคุณ : พี่กาและพี่เล็ก ที่ดาต้าแมทที่ให้เบอร์โทรศัทพ์พี่ธวัช * **น้องวรพล** ที่ให้ยืมพริ้นเตอร์เป็นประจำ * **พี่ตือ รุ่น 24** ที่ให้ยืมพริ้นเตอร์ P6300 พิมพ์ SOURCE FILE * **คุณโอภาส หนูเพ็ง** ผู้ช่วยยกเครื่อง EPROM EMULATOR

พิชญ์ขอบคุณ : พี่หมี พี่เหมียว พี่มิกซ์ พี่ลี้ พี่ธวัช และ **พี่โส่ย รุ่น 26** สำหรับความช่วยเหลือและคำแนะนำต่างๆมากมาย * **นัทและส่ง** ผู้มีส่วนช่วยเหลือผลักดัน THESIS ให้ออกมาเป็นรูปร่างได้อย่างสวยงาม * **เพื่อน 4 J** สำหรับกำลังใจและคำชู่ตลอดการทำ PROJECT

สมโชคขอบคุณ : อาจารย์มนัส สังวรศิลป์ เทอดศักดิ์ สมนึก สำหรับ PLOTTER * **พี่เอ๋ (ธันวา)** สำหรับคำแนะนำทางด้านฮาร์ดแวร์ * **อาจารย์และพี่สต๊าฟภาคเครื่องกล** สำหรับ CASE PLC * **โก้ บางมด** สำหรับคำแนะนำและหนังสืออ้างอิง * **สยาม** สำหรับ CASE DESIGN

รวมทั้งผู้ที่ไม่ได้กล่าวถึงอีกมาก ขอบคุณจริงๆครับ

สุดท้ายขอขอบพระคุณคุณพ่อและคุณแม่สำหรับทุกสิ่งทุกอย่าง

หนังสืออ้างอิง

1. สุพรรณ กุลพาณิชย์, "การใช้งาน PROGRAMMABLE CONTROL 1", สำนักพิมพ์ดวงกมล, 258 หน้า, 2533
2. สุพรรณ กุลพาณิชย์, "การใช้งาน PROGRAMMABLE CONTROL 2"
3. สุเธียร เกียรติสุนทร, "หลักการทำงานและเทคนิคการประยุกต์ใช้งาน PC/PLC", บริษัท ซีเอ็ดยูเคชั่น จำกัด, 200 หน้า, 2531
4. วิชัย ตันติจริยางกูร, "การออกแบบพีแอลซี", วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2528, 174 หน้า, 2528
5. บุญชู งามไพโรจน์พิบูลย์, พจน์ เจริญผลดี, พรเดช หวังวิวัฒน์สิน, อัชฌฏิล เจียระไนรุ่งโรจน์, "เครื่องควบคุมแบบโปรแกรมได้", บริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์บัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2533, 100 หน้า, 2533
6. "LINEAR DATA BOOK",NATIONNAL SEMICONDUCTOR CORPORATION
7. OMRON," C 200H PROGRAMMABLE CONTROLLER SYSTEM C-SERIES: USER MANUAL", OMRON TATEISI ELECTRONICS
8. "MANUAL MICROPROCESSOR MCS-51", INTEL
9. FRANK D.PETRUZELLA, "PROGRAMMABLE LOGIC CONTROLLERS" MCGRAW-HILL BOOK COMPANY TJ 223. P96 P44

PC TJ 223. P96 W3

เครื่องควบคุมโปรแกรม T 35912

Progr Mable A/D → D/A

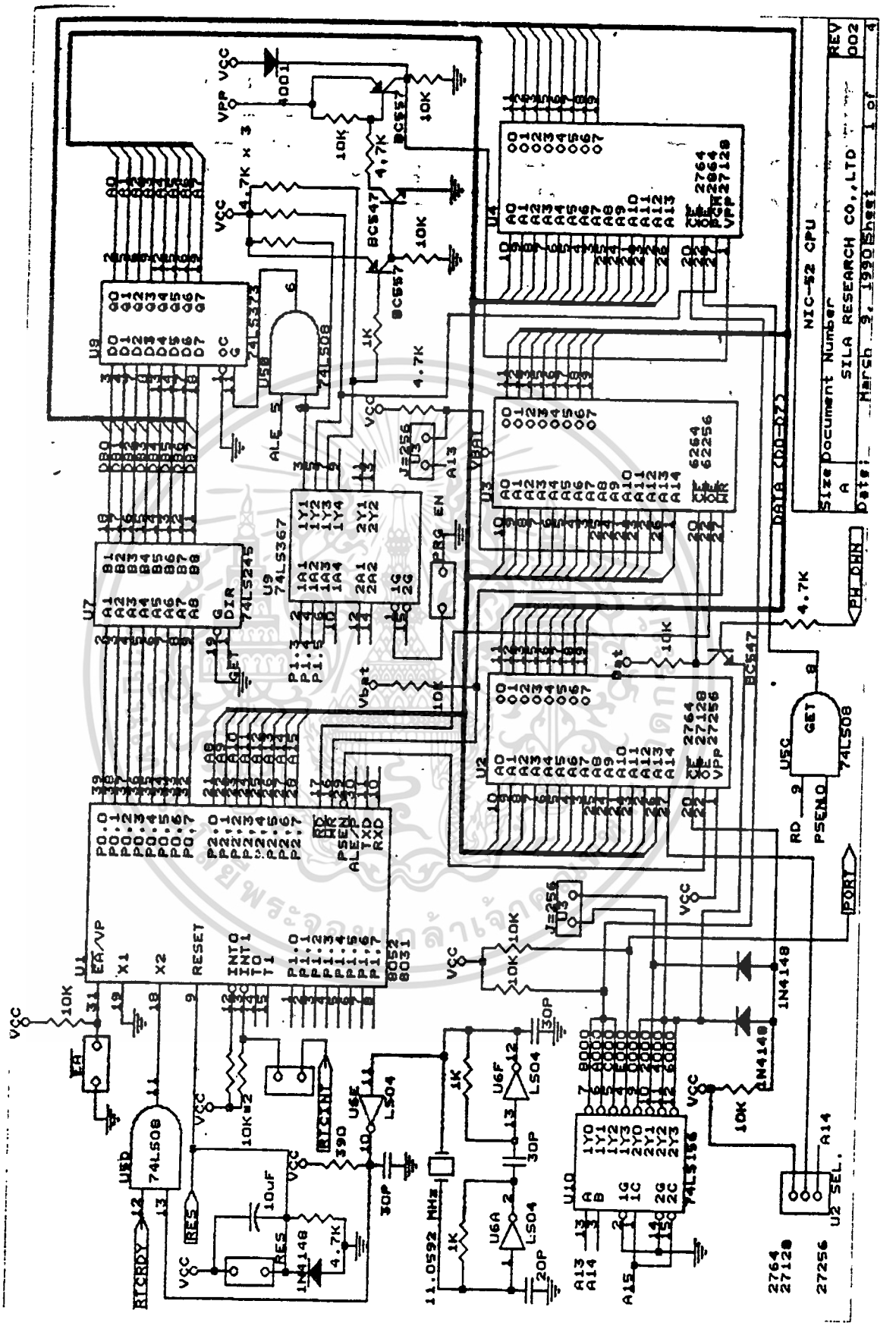
โปรแกรม T35298 PLCs

PLC ภาษา TJ 223. P76 S72

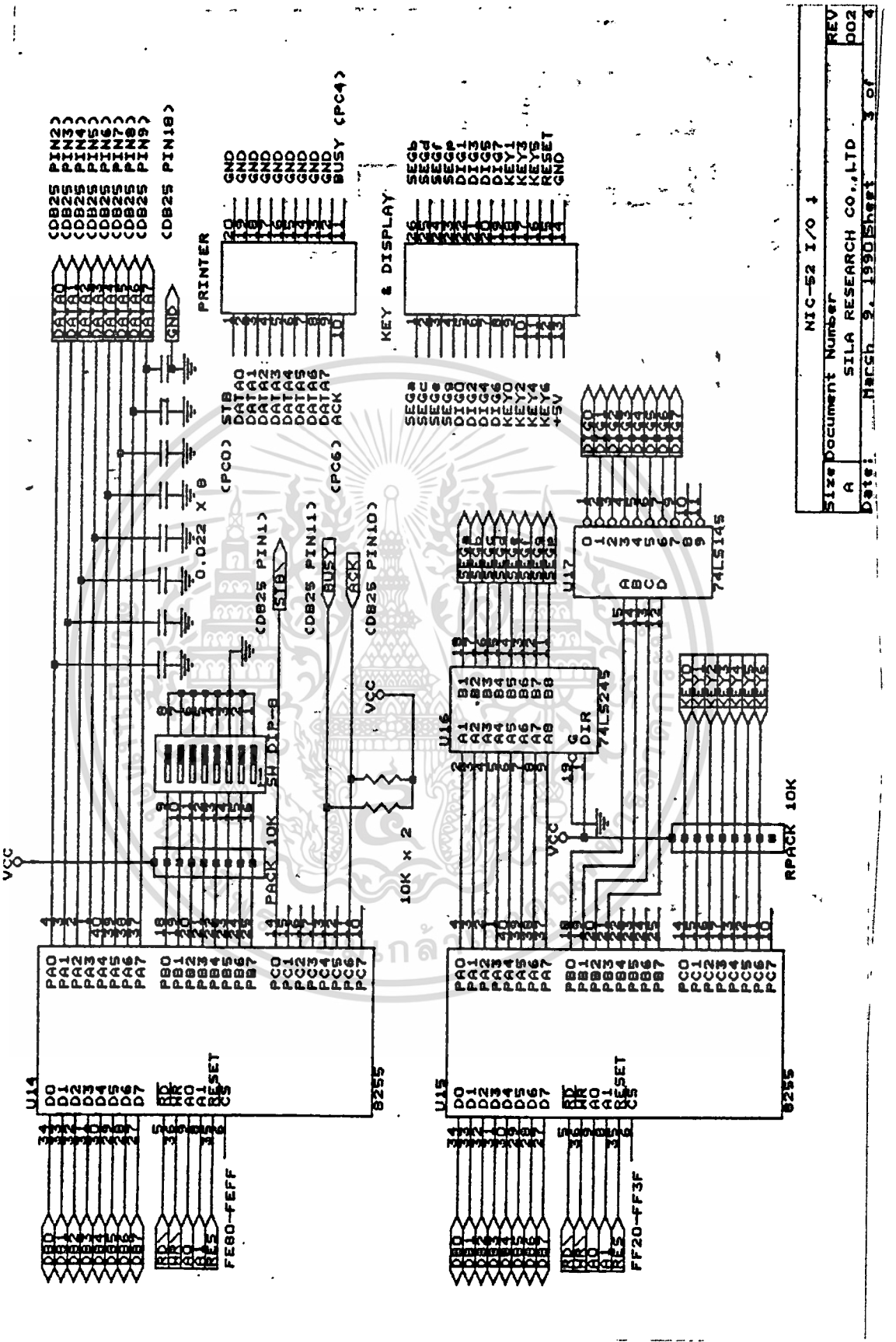
ภาคผนวก

ภาคผนวก ก

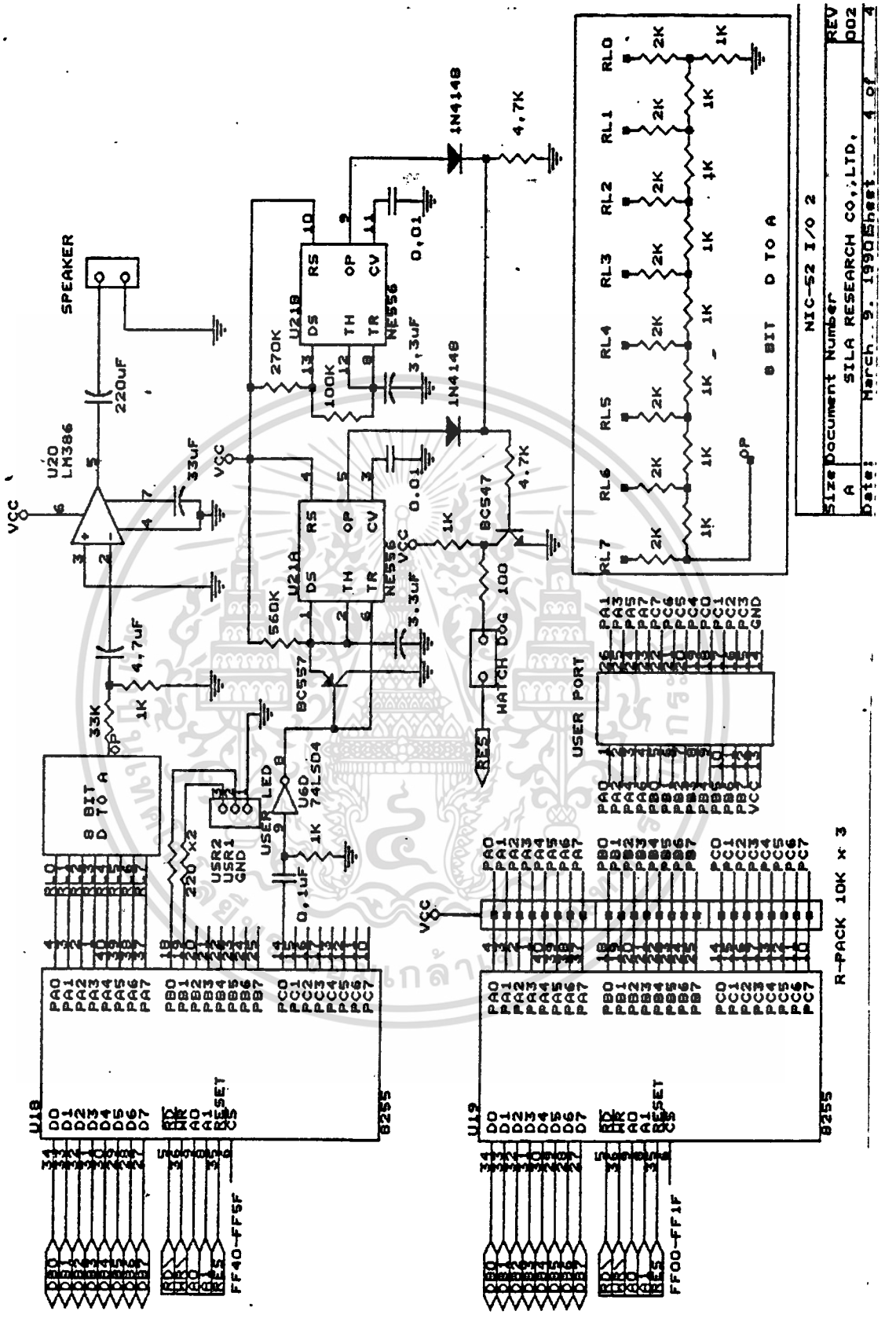
วงจรฮาร์ดแวร์ของเครื่อง PLC



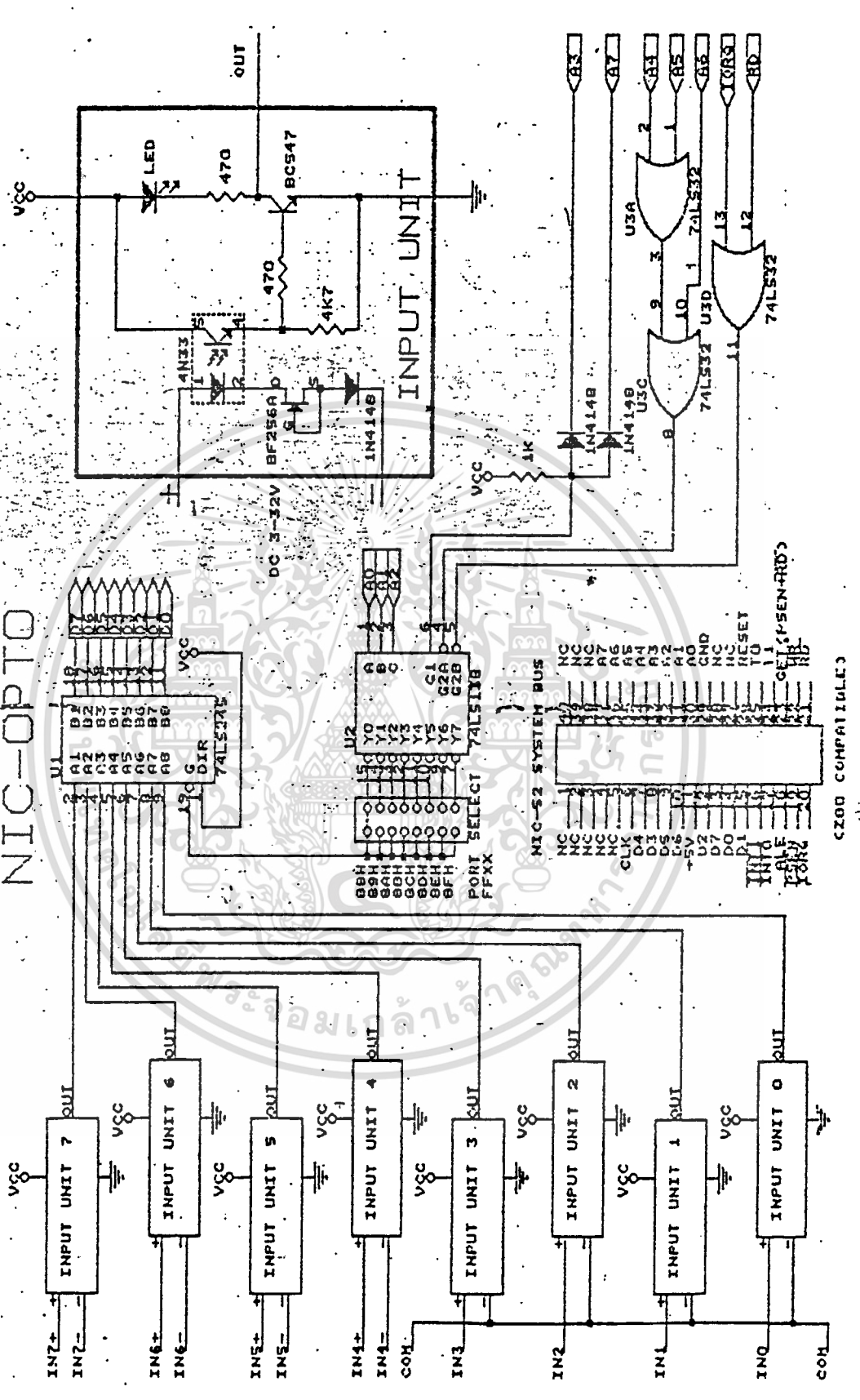
รูปที่ 1 แสดงวงจรฮาร์ดแวร์ของ NIC-52



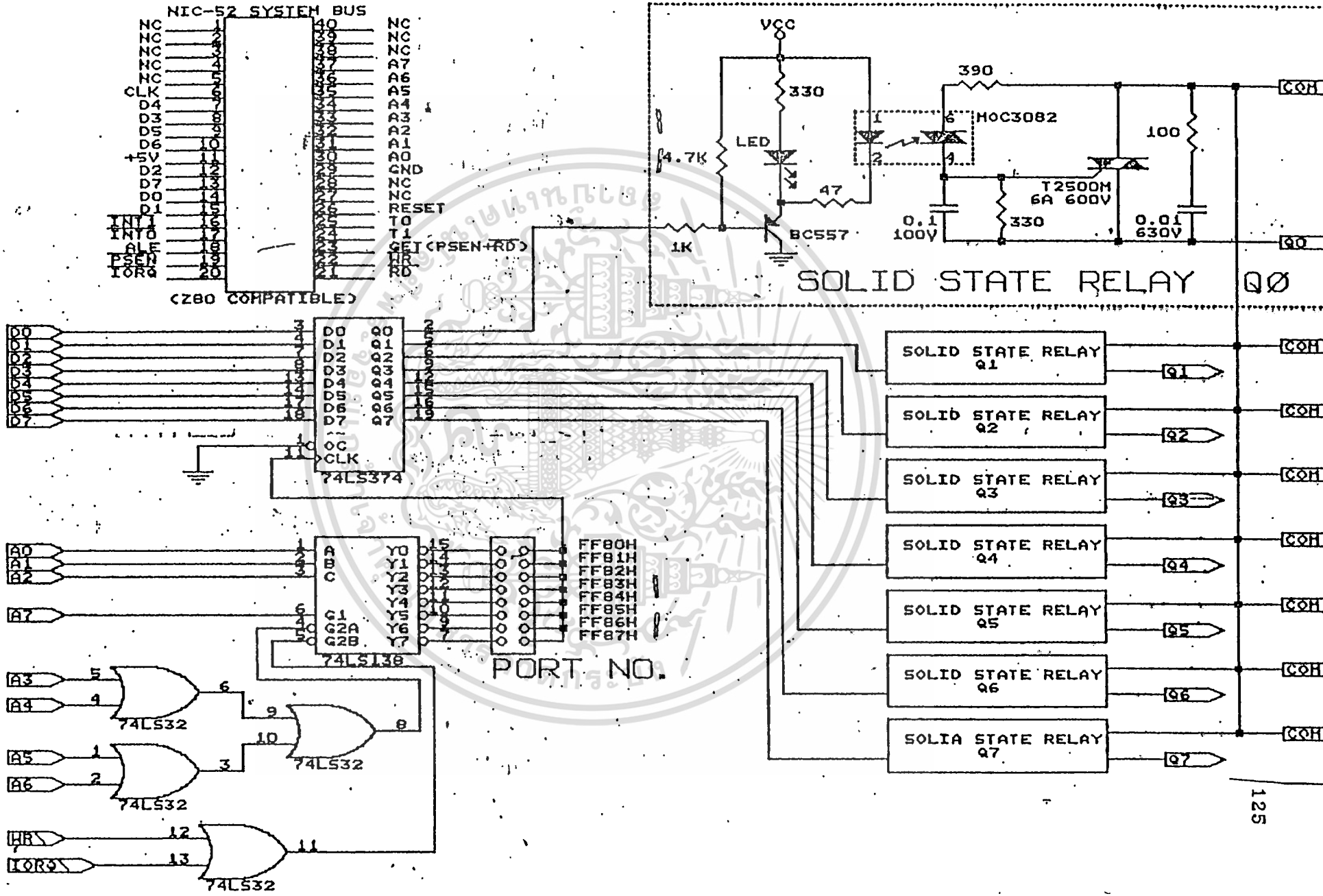
รูปที่ 2 แสดงวงจร I/O ของ NIC-52

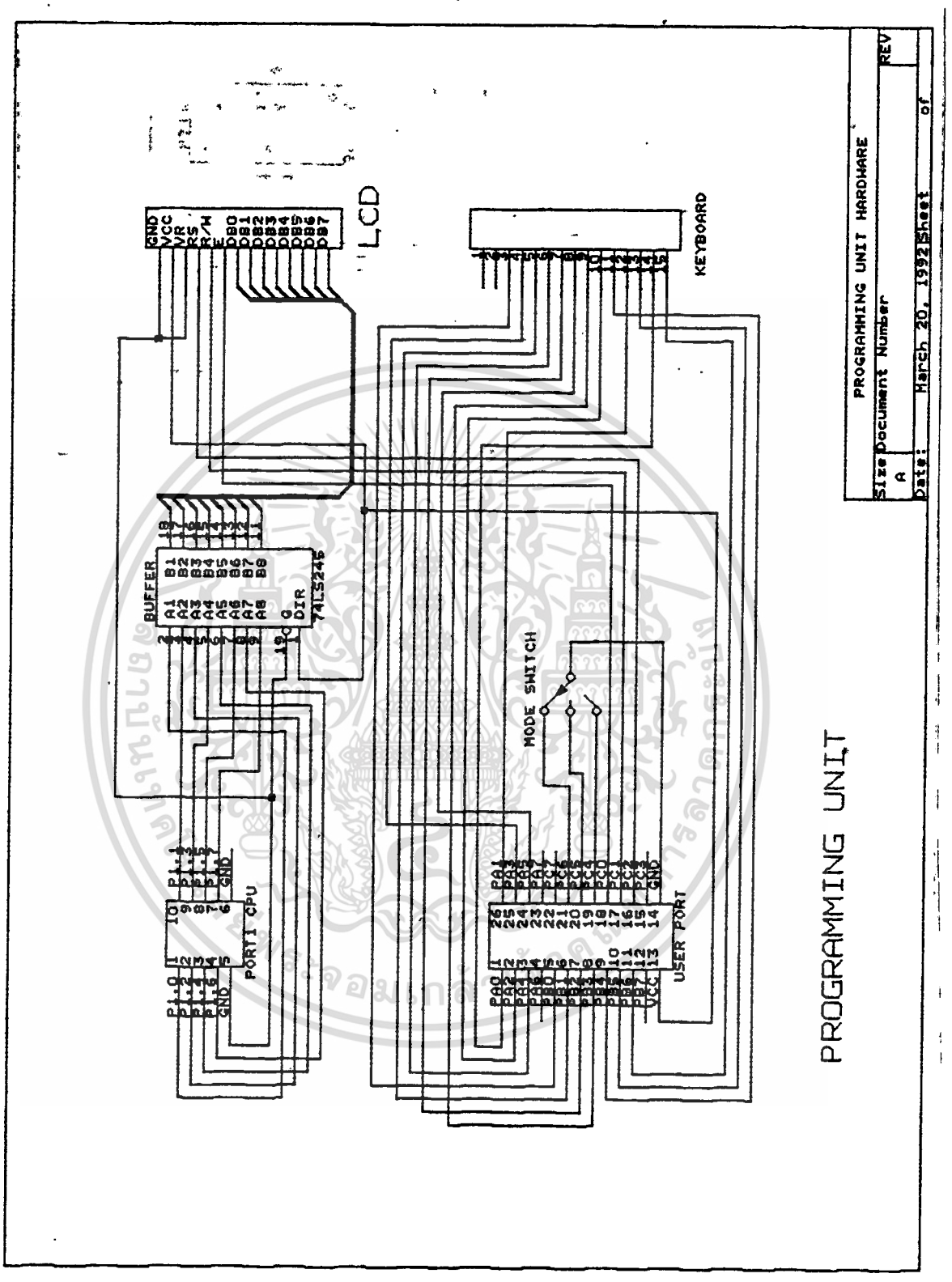


รูปที่ 3 แสดงวงจร I/O ของ NIC-52

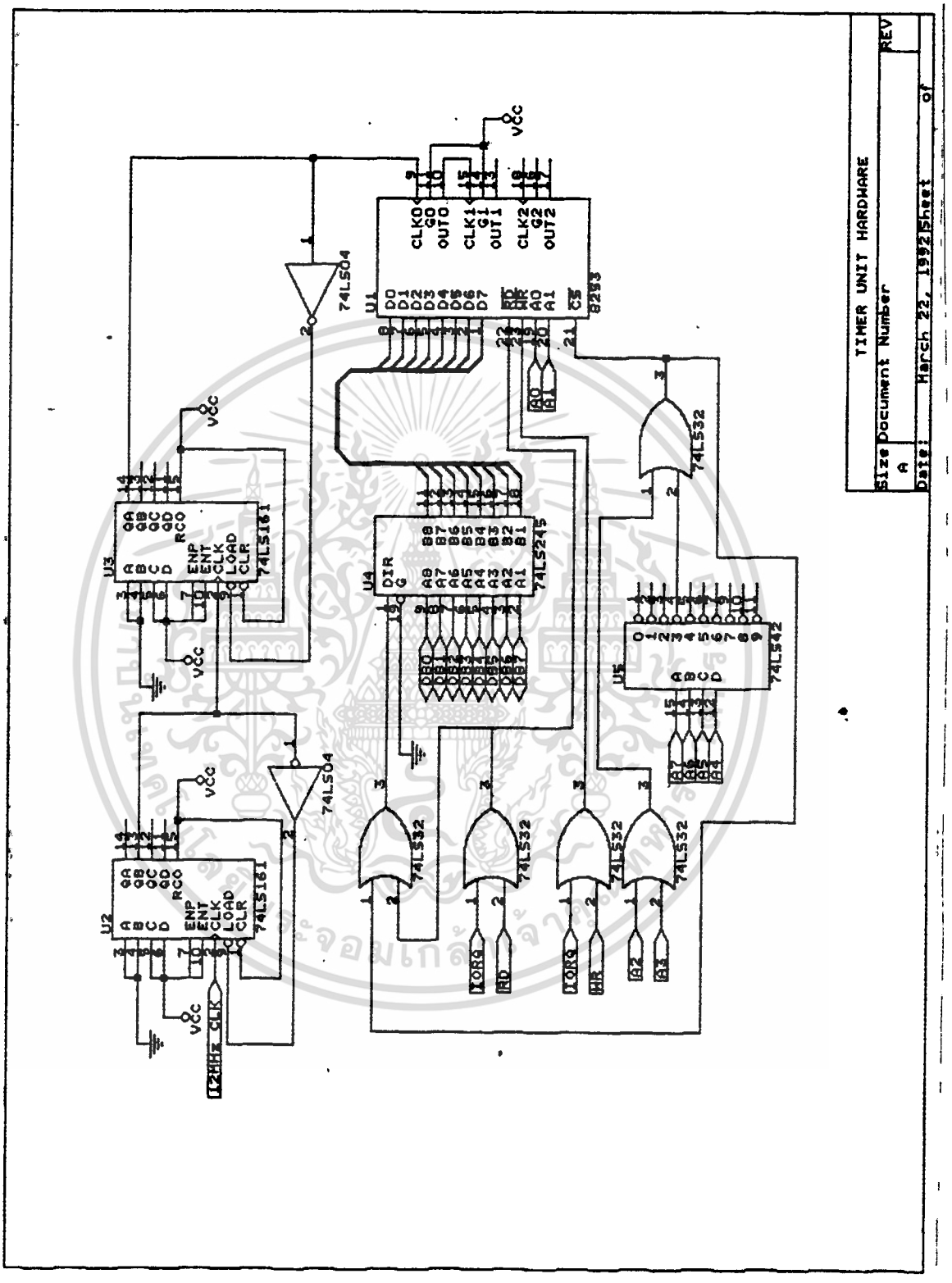


รูปที่ 4 แสดงวงจรฮาร์ดแวร์ของ NIC-OPTO

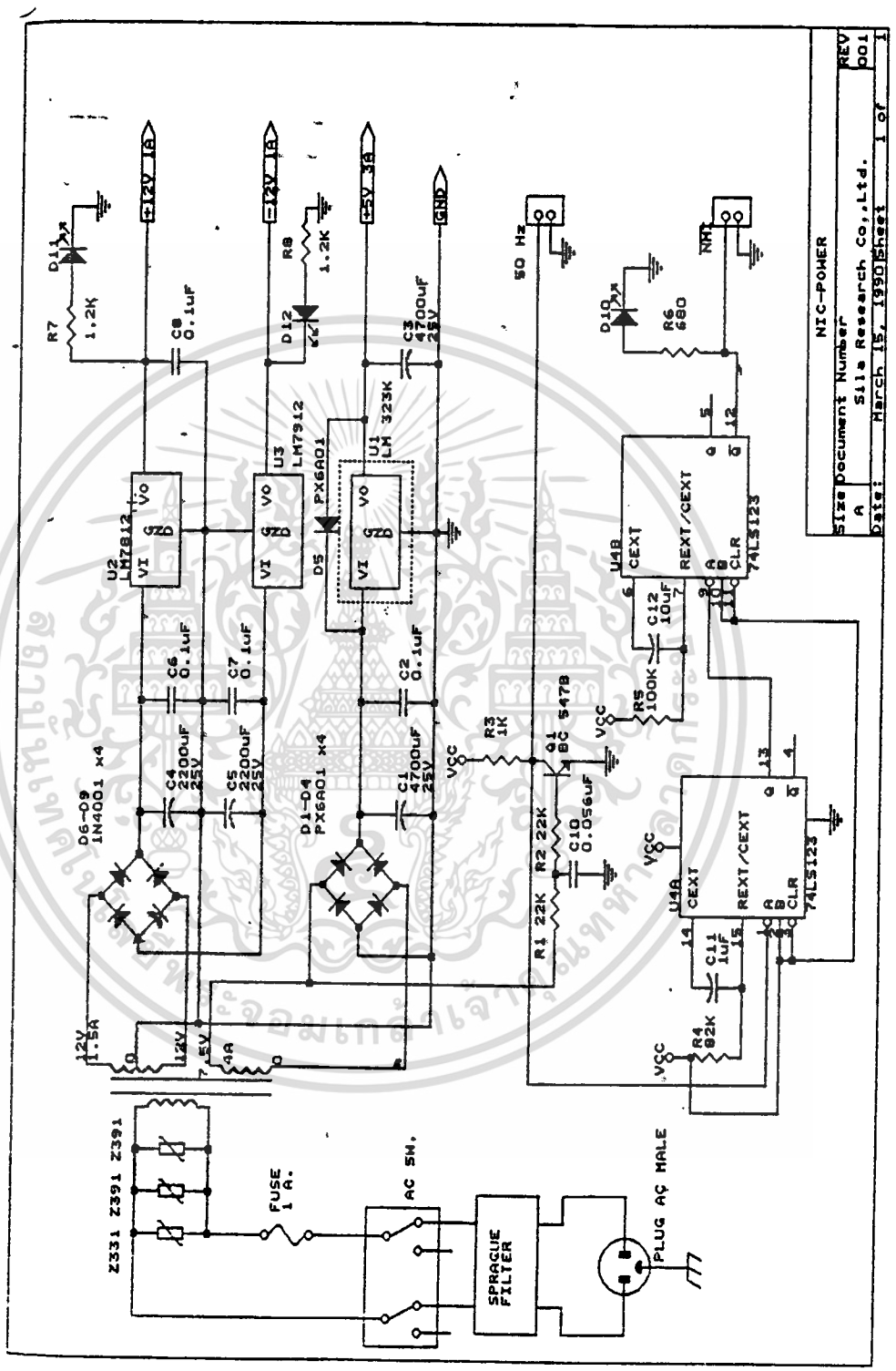

NIC-52 SYSTEM BUS
(Z80 COMPATIBLE)
VCC
390
330
LED
4.7K
47
1K
BC557
HOC3082
100
T2500M
6A 600V
0.1
100V
330
0.01
630V
SOLID STATE RELAY Q0
74LS374
74LS138
74LS32
SOLID STATE RELAY Q1
SOLID STATE RELAY Q2
SOLID STATE RELAY Q3
SOLID STATE RELAY Q4
SOLID STATE RELAY Q5
SOLID STATE RELAY Q6
SOLIA STATE RELAY Q7
PORT NO.
FF80H
FF81H
FF82H
FF83H
FF84H
FF85H
FF86H
FF87H



รูปที่ 6 แสดงวงจรหน่วยป้อนโปรแกรม



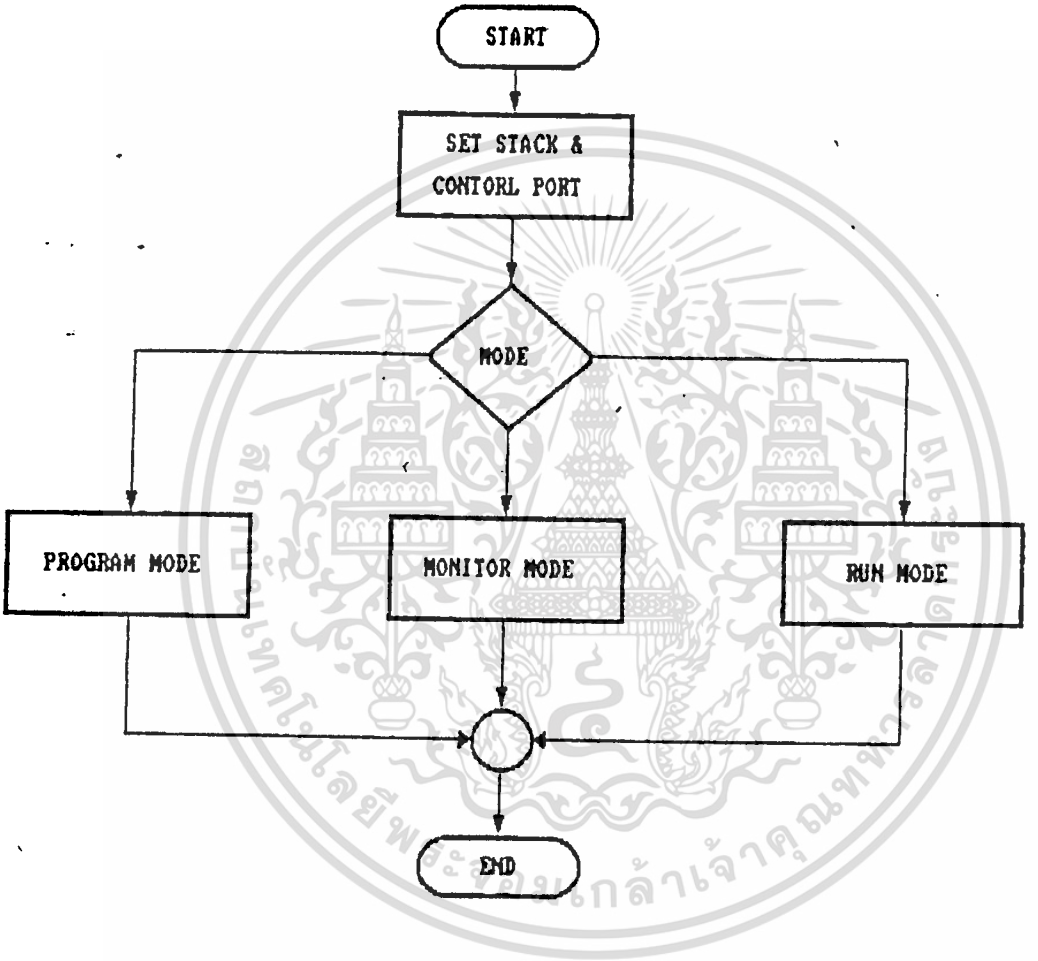
รูปที่ 7 แสดงวงจรของหน่วยนับเวลามาตราฐาน



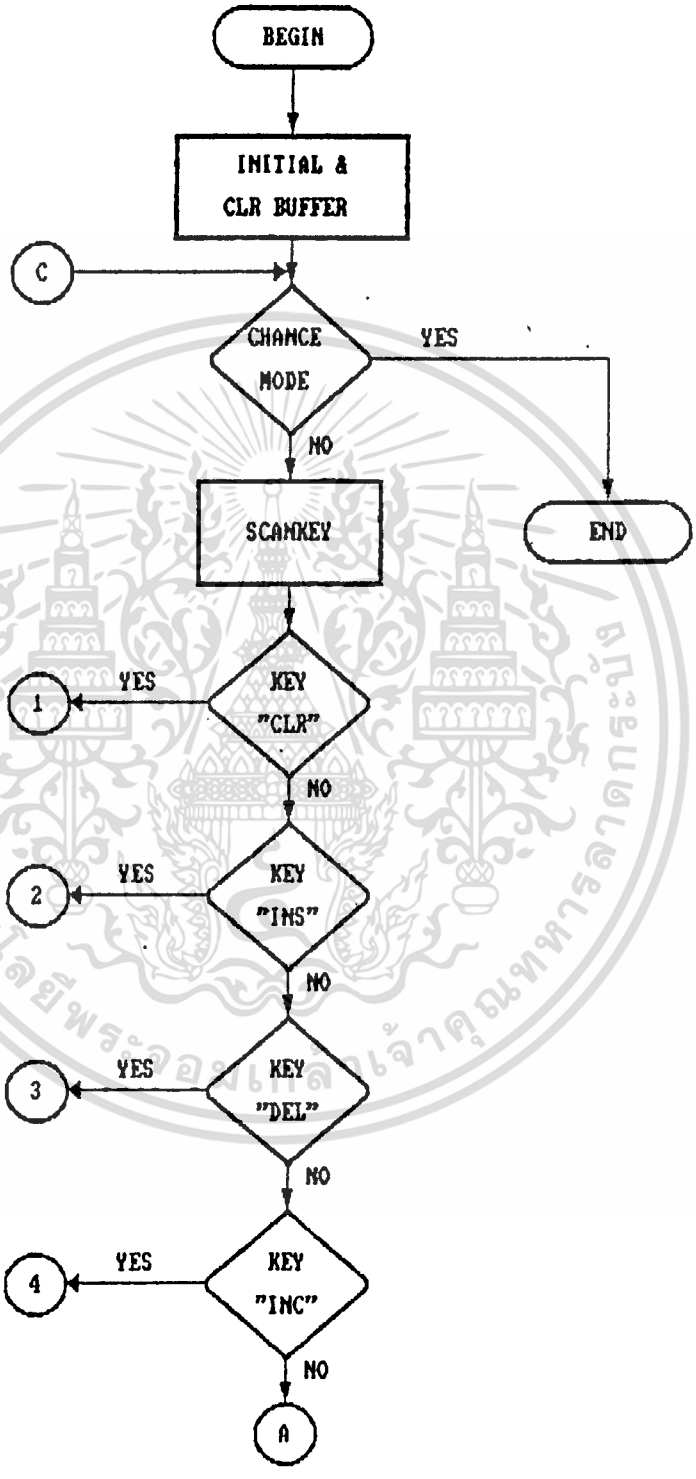
รูปที่ 8 แสดงวงจรของ NIC-POWER

ภาคผนวก ข

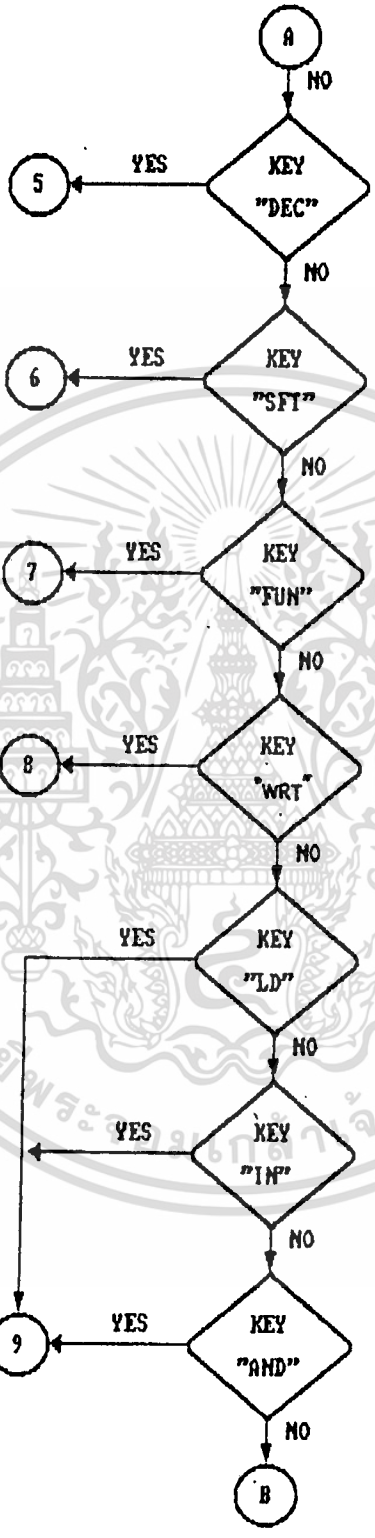
FLOW CHART โปรแกรมบริหารระบบ



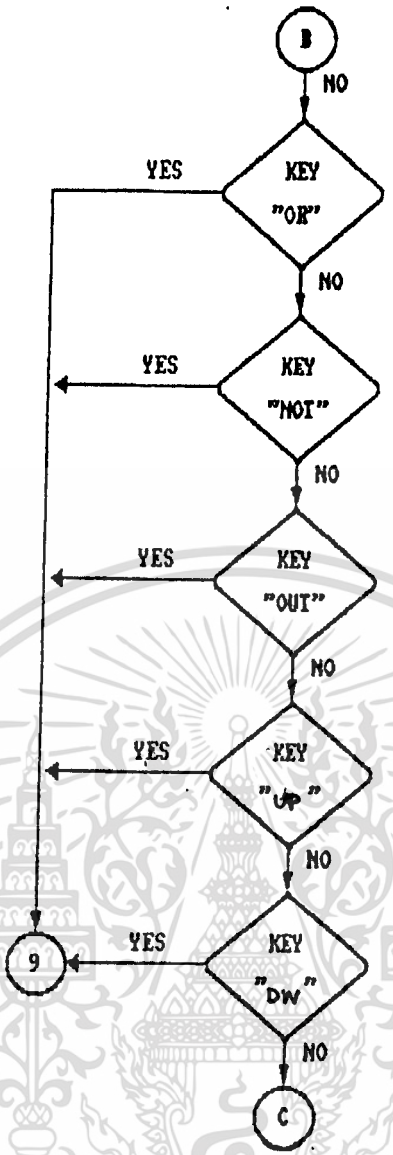
รูปที่ 2 แสดงการทำงานในการเลือกโหมดของระบบ PLC



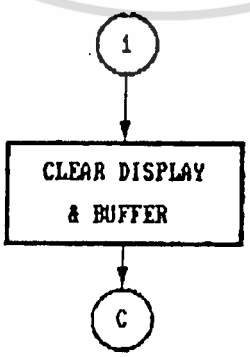
รูปที่ 3.3 แสดงการทำงานของ PLC ในโหมดโปรแกรม



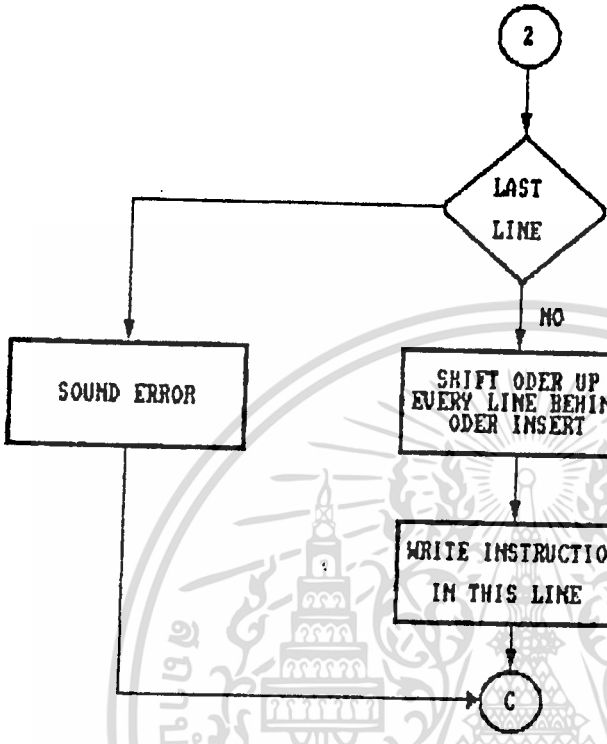
รูปที่ 3 แสดงการทำงานของ PLC ในโหมดโปรแกรม (ต่อ)



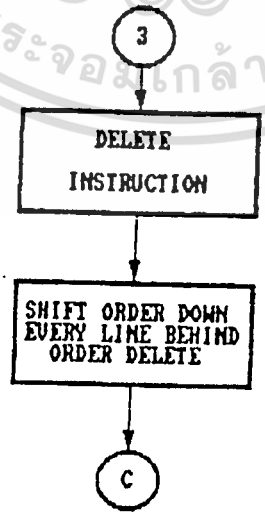
รูปที่ 3 แสดงการทำงานของ PLC ในโหมดโปรแกรม (ต่อ)



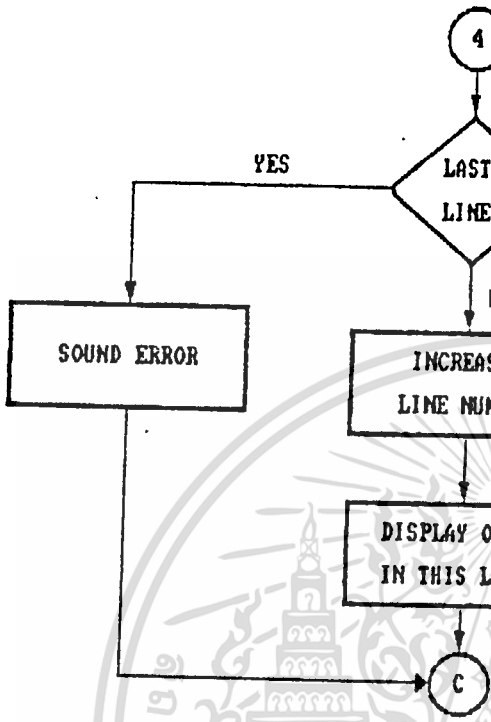
รูปที่ 4 แสดงการทำงานของ CLEAR ในโหมดโปรแกรม



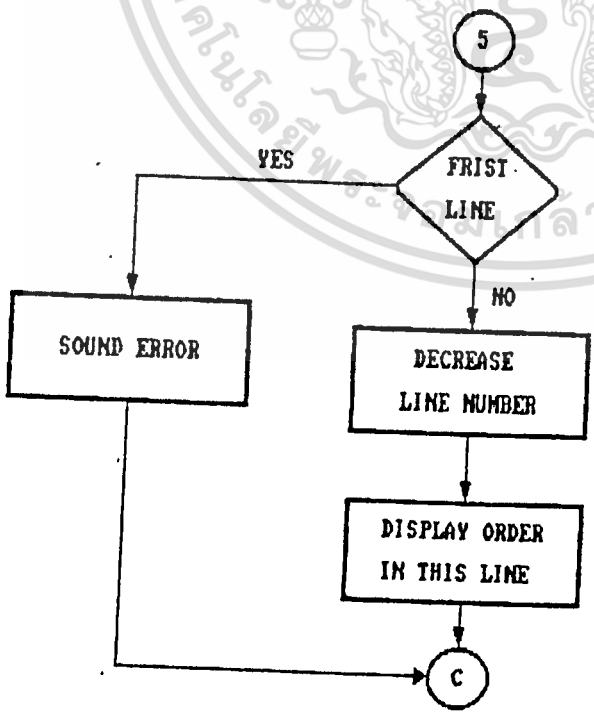
รูปที่ 5 แสดงการทำงานของการ INSERT ในโหมดโปรแกรม



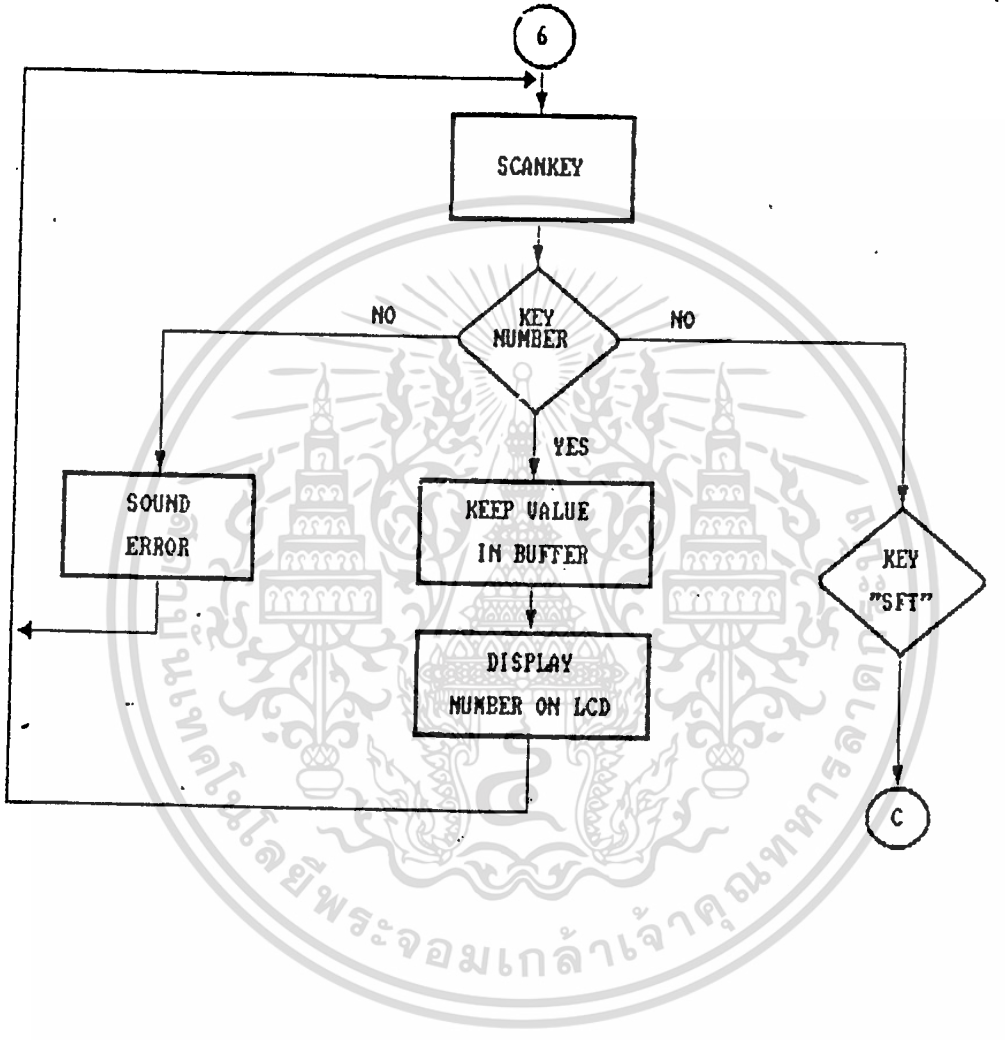
รูปที่ 6 แสดงการทำงานของการ DELETE ในโหมดโปรแกรม



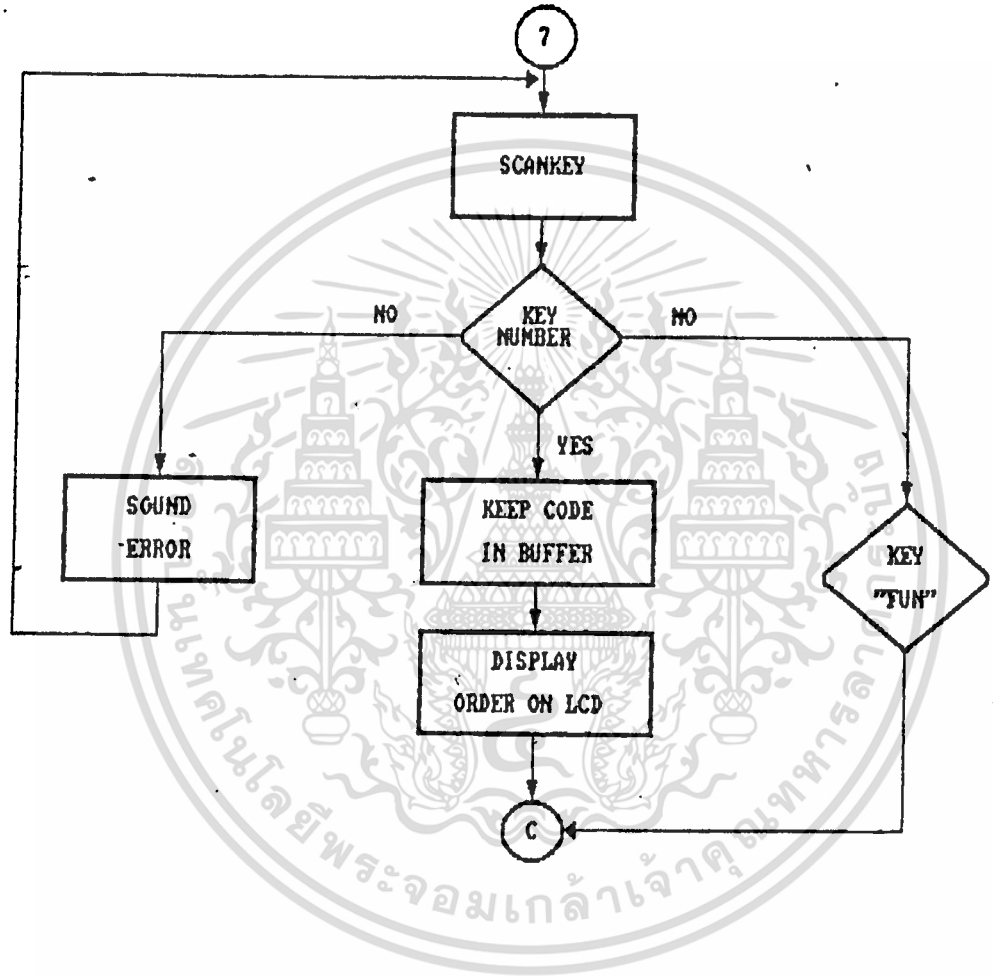
รูปที่ 7 แสดงการทำงานของการ INCRESE ในโหมดโปรแกรม



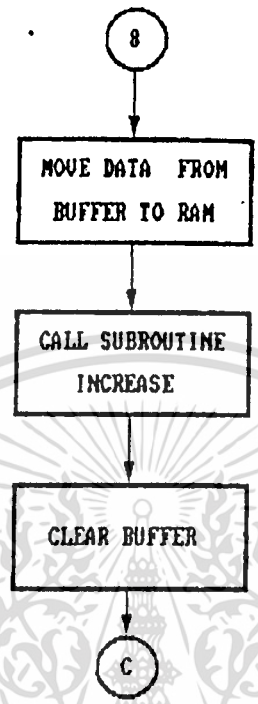
รูปที่ 8 แสดงการทำงานของการ DECRESE ในโหมดโปรแกรม



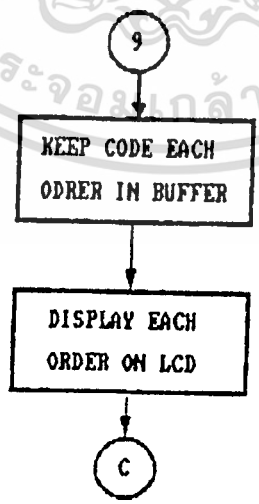
รูปที่ 19 แสดงการทำงานของการ SHIFT ในโหมดโปรแกรม



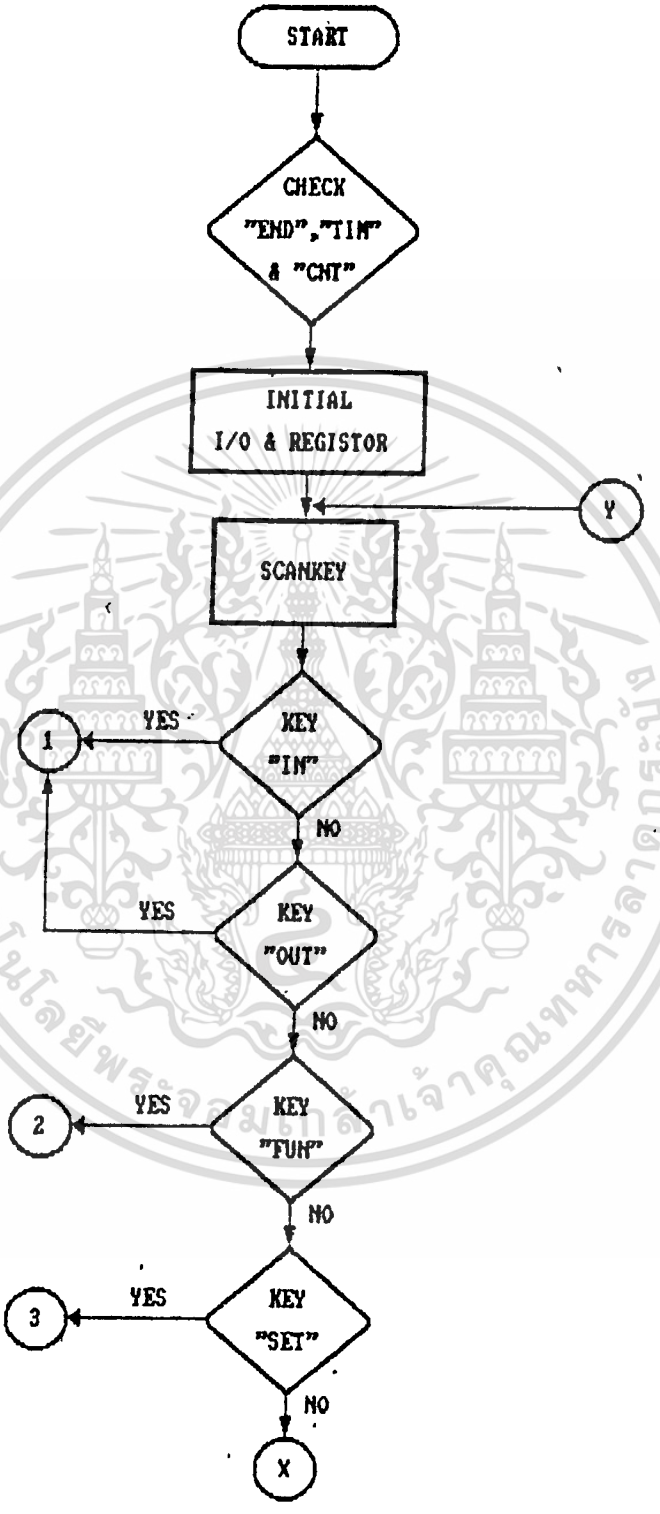
รูปที่ 4.10 แสดงการทำงานของการ FUNCTION ในโหมดโปรแกรม



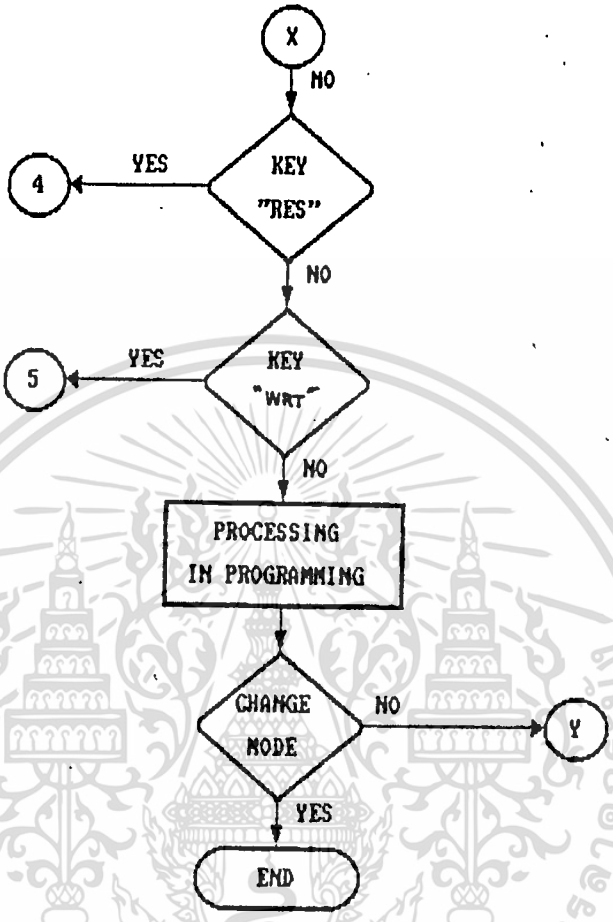
รูปที่ 11 แสดงการทำงานของการ WRITE ในโหมดโปรแกรม



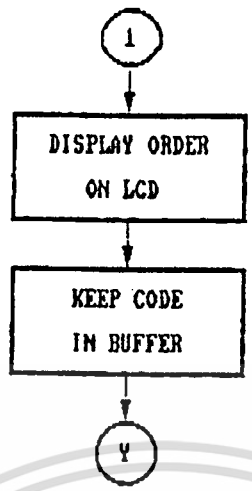
รูปที่ 12 แสดงการทำงานของการ "คีย์คำสั่ง" ในโหมดโปรแกรม



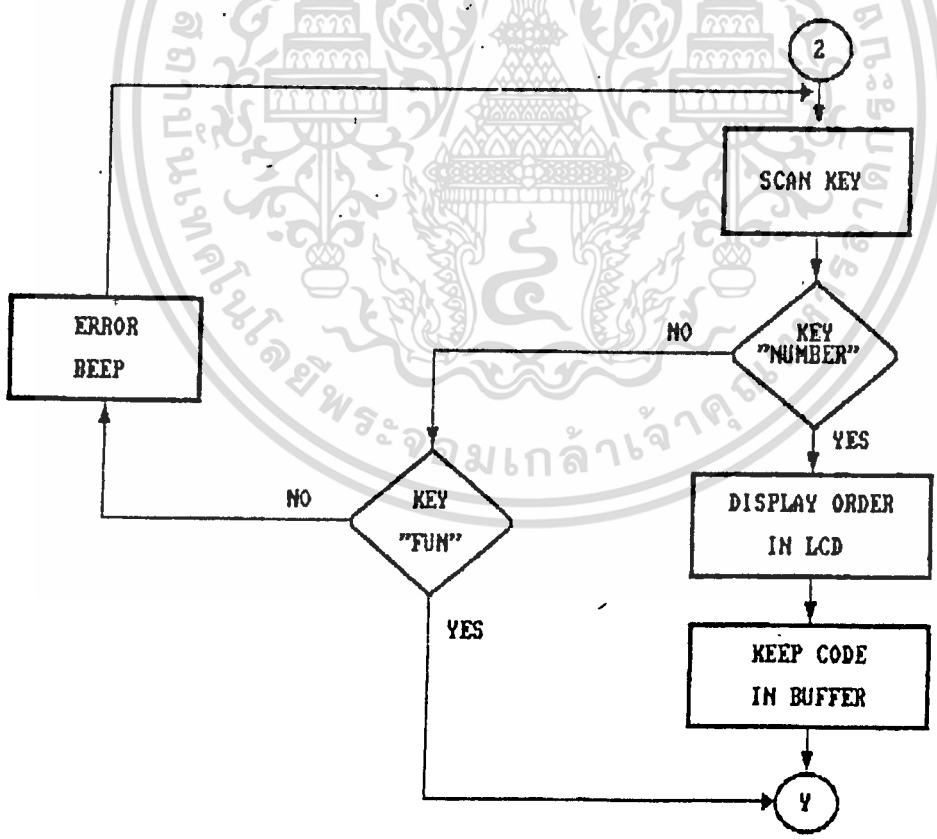
รูปที่ 13 แสดงการทำงานในโหมดมอนิเตอร์



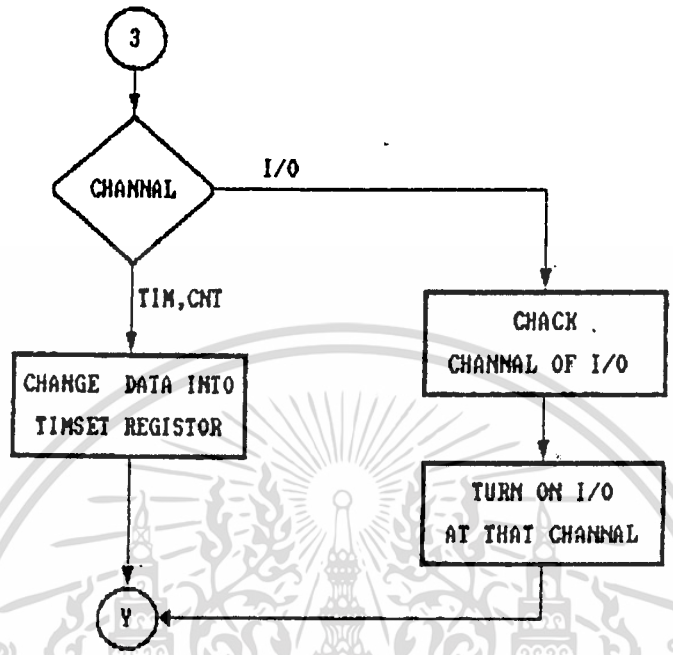
รูปที่ 3.13 แสดงการทำงานในโหมดมอนิเตอร์ (ต่อ)



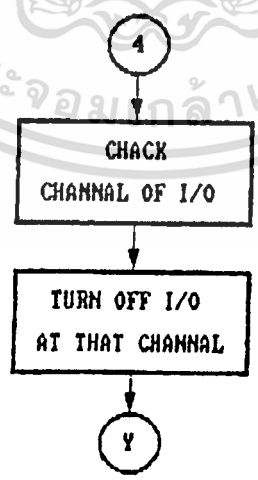
รูปที่ 14 แสดงการทำงานในการมอนิเตอร์ค่าอินพุท-เอาท์พุท



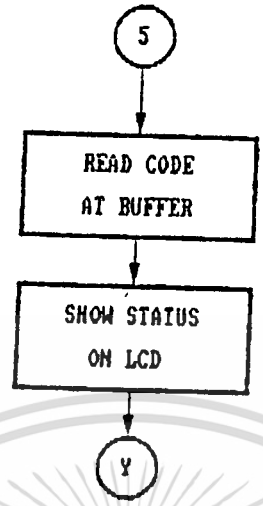
รูปที่ 15 แสดงการทำงานในการมอนิเตอร์ค่าตั้งเวลา ตัวนับ



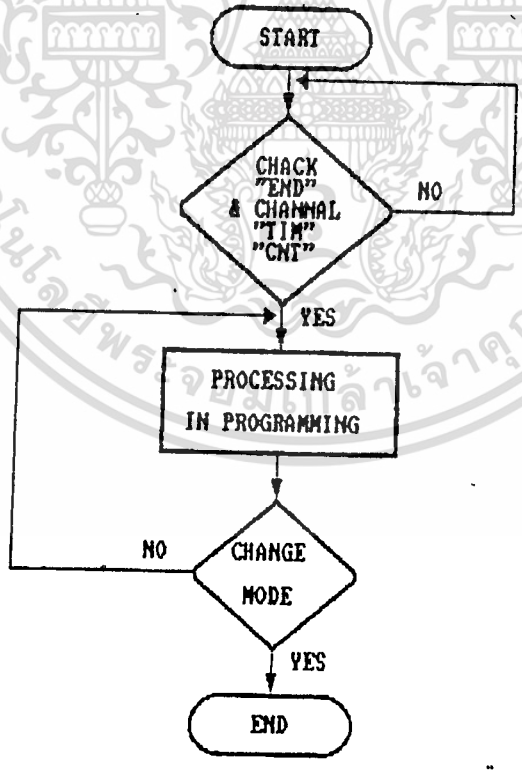
รูปที่ 16 แสดงการทำงานในการ "เซ็ต" ค่าโอเปอร์แรนด์ต่างๆ



รูปที่ 17 แสดงการทำงานในการ "รีเซ็ต" ค่าโอเปอร์แรนด์ต่างๆ

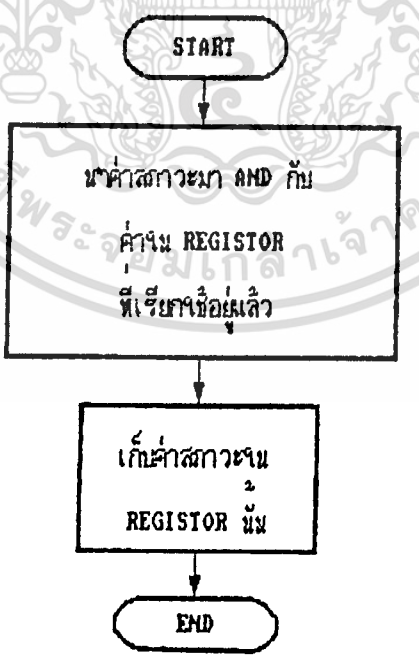


รูปที่ 18 แสดงการทำงานเมื่อกดคีย์ "WRITE" เพื่อมอนิเตอร์ค่าต่างๆ

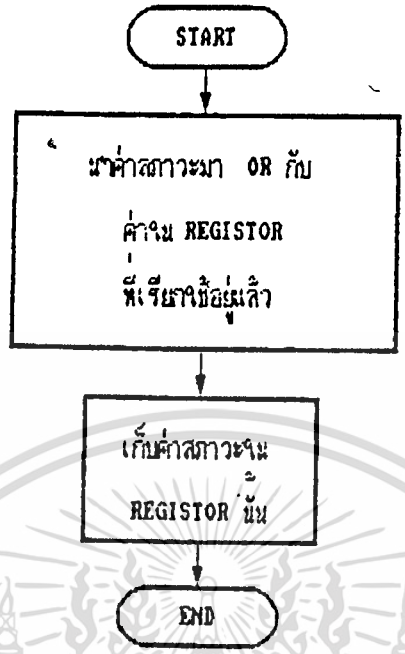


รูปที่ 19 แสดงการทำงานในโหมดรัน

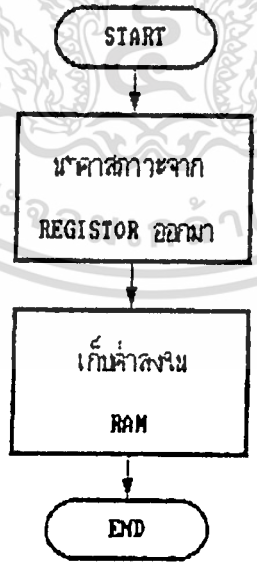
รูปที่ 20 แสดงการทำงานตามคำสั่ง LD



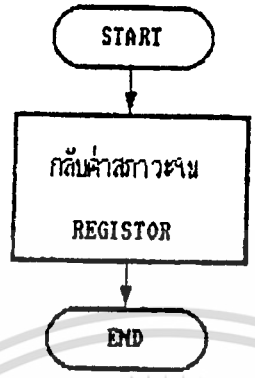
รูปที่ 21 แสดงการทำงานตามคำสั่ง AND



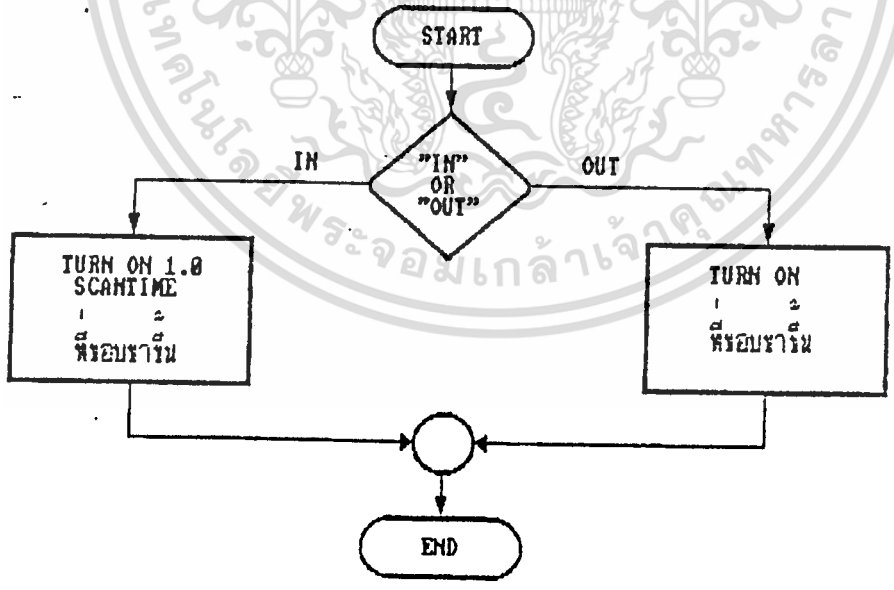
รูปที่ 22 แสดงการทำงานตามคำสั่ง OR



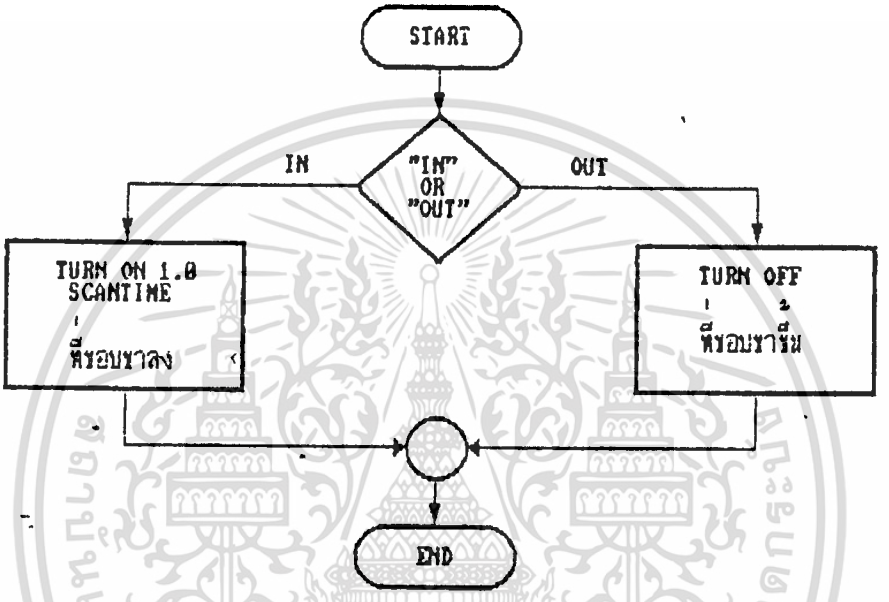
รูปที่ 23 แสดงการทำงานตามคำสั่ง OUT



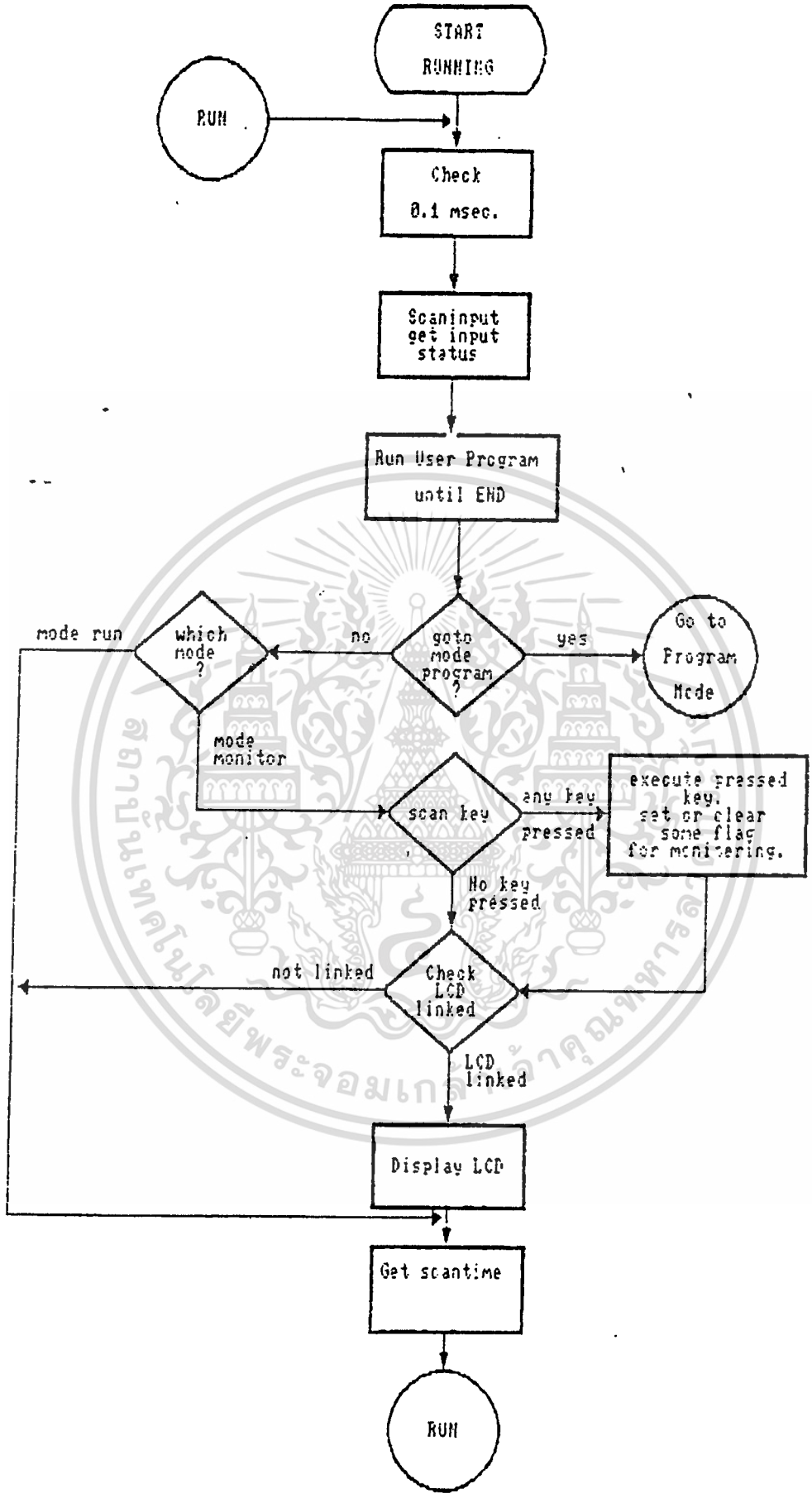
รูปที่ 24 แสดงการทำงานตามคำสั่ง NOT



รูปที่ 25 แสดงการทำงานตามคำสั่ง TON



รูปที่ 26 แสดงการทำงานตามคำสั่ง TOFF



รูปที่ 27 แสดงลำดับการทำงานในโหมดมอนิเตอร์และโหมดรัน

ภาคผนวก ค

SOURCE FILE โปรแกรมบริหารระบบ