รายงานการวิจัยและพัฒนา
ในโครงการหลัก
วงจรอิเลคทรอนิกส์เพื่ออุตสาหกรรม

เครื่องควบคุมที่โปรแกรมได้ชนิดที่มีความสามารถด้านจัดการข้อมูล

PROGRAMMABLE CONTROLLER WITH DATA HANDLING CAPABILITY

ดำเนินการโดย

รองศาสตราจารย์ กฤษดา วิศวธีรานนท์
อาจารย์ ดร. สมบูรณ์ จงชัยกิจ
อาจารย์ สิทธิพร ประวัติรุ่งเรือง

ห้องวิจัยวัดคุมทางอุตสาหกรรม ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ได้รับทุนสนับสนุนการวิจัยและพัฒนาจาก
ศูนย์เทคโนโลยีอิเลคทรอนิกส์และคอมพิวเตอร์แห่งชาติ
กระทรวงวิทยาศาสตร์เทคโนโลยีและการพลังงาน

ปีงบประมาณ 2533

รายงานผลการวิจัยและพัฒนา

โครงการหลัก การพัฒนาวงจรอิเล็กทรอนิกส์เพื่ออุตสาหกรรม
(Industrial Electronics)
โครงการย่อย เครื่องควบคุมที่โปรแกรมได้ชนิดที่มีความสามารถ
ด้านจัดการข้อมูล
(Programmable Controller with Data Handling Capability)

โดย
รองศาสตราจารย์ กฤษดา วิศวธีรนานนท์
อาจารย์ ดร. สมบูรณ์ จงชัยกิจ
อาจารย์ สิทธิพร ประวัติรุ่งเรือง

ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย

มกราคม 2535

ทุนอุดหนุนการวิจัยและพัฒนา
ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ
ปีงบประมาณ 2533

โครงการวิจัยหลัก การพัฒนาวงจรอิเล็กทรอนิกส์เพื่ออุตสาหกรรม
โครงการย่อย เครื่องควบคุมที่โปรแกรมได้ชนิดที่มีความสามารถทางด้านจัดการข้อมูล
ผู้วิจัย รองศาสตราจารย์ กฤษดา วิศวธีรานนท์
อาจารย์ ดร.สมบูรณ์ จงชัยกิจ
อาจารย์ สิทธิพร ประวัติรุ่งเรือง
ทุนวิจัย ทุนอุดหนุนการวิจัยและพัฒนา
ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์
ปี 2533

บทคัดย่อ

รายงานนี้กล่าวถึง การออกแบบเครื่องควบคุมชนิดโปรแกรมได้ซึ่งมีความสามารถทางด้านจัดการข้อมูล ผู้ใช้สามารถกำหนดขนาดและชนิดของอินพุท/เอาท์พุทตามความเหมาะสมของงานได้ มีโครงสร้างของระบบเป็นแบบโมดูล สามารถต่ออินพุท/เอาท์พุทได้สูงสุด 512 จุด สามารถรับสัญญาณอินพุทแบบอนาลอกได้ ตัวป้อนโปรแกรมเป็นแบบมือถือแสดงผลโดย LCD การเขียนโปรแกรมควบคุมเป็นแบบโปรแกรมขั้นบันได (Ladder) และฟังก์ชั่นบล๊อก (Function block) มีคำสั่งทั้งสิ้น 64 คำสั่ง และมีคำสั่งในการจัดข้อมูลต่างๆ ด้วย ความเร็วในการทำงานของคำสั่งเบื้องต้นประมาณ 5.7 µsec ผู้ใช้สามารถเขียนโปรแกรมควบคุมได้สูงสุดประมาณ 4000 คำสั่ง การแปลงคำสั่งขั้นบันไดเป็นแบบผสมระหว่างวิธีคอมพายล์ (Compile) และวิธีคอล (Call) และมีความเร็วในการทำงานแต่ละรอบ (Scan time) น้อยกว่า 100 mesec ผลการทดสอบเป็นที่น่าพอใจสามารถใช้เป็นต้นแบบผลิตภัณฑ์ได้

Project Title	Industrial Electronics
Sub Title	PROGRAMMABLE CONTROLLER WITH DATA HANDLING CAPABILITY
Researcher	Assoc. Prof. Krisada Visavateeranon
	Dr. Somboon Chongchaikit
	Mr. Sittiporn Prawatrungruang
	Dept. of Electrical Engineering
	Chulalongkorn University
Research Fund	NECTEC 1990

ABSTRACT

This report presents a development of programmable controller with data handling capability. The number and type of input/output can be easily chosen by the user due to its modular structure. The controller can handle up to 512 input/output points which include analog input. The programming console is hand-held one with LCD display. The programming language is based on ladder diagram and function block. There are 64 instructions which include data handling ones. The average execution time of basic instruction is about 5.7 μsec. The user can program up to 4000 steps with scan time less than 100 msec. The user program interpretation is based on compiler and call technique. The tests are quite satisfactory and can be adapted as industrial prototype.

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญภาพ

ภาพ หน้า

สารบัญภาพ(ต่อ)

ภาพ หน้า

สารบัญภาพ(ต่อ)

ภาพ หน้า

บทที่ 1

บทนำ

1.1 ความเบื้องต้น

เครื่องควบคุมที่โปรแกรมได้ (Programmable Controller) เรียกย่อ ๆ ว่า PC บางครั้งก็เรียกย่อ ๆ ว่า PLC (Programmable Logic Controller) เป็นอุปกรณ์ควบคุมที่ใช้ในการควบคุมลำดับการทำงานของเครื่องจักร (Sequence Control) เช่น ระบบสายพานลำเลียง เครื่องผสมอาหารสัตว์ เครื่องฉีดพลาสติก และเครื่องจักรต่าง ๆ ที่ใช้ในการผลิตในโรงงานอุตสาหกรรม[8] ก่อนที่จะมีการคิดค้นและพัฒนา PC มาใช้ในอุตสาหกรรมนั้น การควบคุมลำดับจะใช้วงจรรีเลย์แม่เหล็กไฟฟ้าเป็นส่วนใหญ่ เนื่องจากวงจรรีเลย์มีขนาดใหญ่ ใช้องค์ประกอบวงจรจะต้องเดินสายเชื่อมโยงอุปกรณ์ต่าง ๆ ทำให้สิ้นเปลืองเวลา และทำการแก้ไขหรือปรับปรุงวงจรยุ่งยาก ในปี 1960 บริษัทเยนเนรัล มอเตอร์ได้สนับสนุนให้มีการพัฒนาเครื่องควบคุมที่โปรแกรมได้มาใช้แทนวงจรรีเลย์นี้ ในระยะแรกของ PC จะใช้เครื่องคอมพิวเตอร์ขนาดใหญ่เป็นหน่วยควบคุมกลาง มีหน้าที่ในการประมวลผลลอจิกต่าง ๆ ของวงจร ในระยะต่อมามีการผลิตไมโครโปรเซสเซอร์และชิพหน่วยความจำที่มีราคาถูกลง และขนาดเล็กลงมาก และสามารถใช้งานในโรงงานอุตสาหกรรมได้อย่างกว้างขวาง[8]

PC เริ่มเข้ามาในประเทศไทยเมื่อประมาณปี 2525 โดยเข้ามาพร้อมกับเครื่องจักรกลที่นำเข้ามาจากต่างประเทศ ต่อมาก็มีบริษัทสั่งเข้ามาจำหน่ายในประเทศ PC จะผลิตในประเทศอเมริกา,ยุโรป และญี่ปุ่นมากที่สุด ยี่ห้อที่รู้จักกันดีในวงการได้แก่ Allan Bladley, Gould Modicon, Omron, Mitsubishi, Siemens เป็นต้น ในปัจจุบันมี PC อยู่ถึง 50 ยี่ห้อ เนื่องจาก PC มีวิธีการใช้แตกต่างจากวงจรรีเลย์ทั่วไป เพราะมีคุณสมบัติเป็นคอมพิวเตอร์ จึงต้องมีการเขียนโปรแกรมซึ่งมีลักษณะเหมือนวงจรควบคุมเพื่อป้อนเข้า PC ก่อนที่จะเริ่มให้ทำงาน วิศวกรและช่างเทคนิคในประเทศที่รู้จักวิธีใช้งาน PC ยังมีน้อย จึงได้มีการจัดอบรมสัมมนาเพื่อแนะนำให้รู้จักวิธีใช้ PC หน่วยงานที่เคยจัดสัมมนาได้แก่สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)[1] เทคโนโลยีพระจอมเกล้าลาดกระบัง และบริษัทตัวแทนจำหน่าย เป็นต้น

เนื่องจาก PC ยังเป็นสิ่งใหม่สำหรับประเทศไทย จึงยังมีการวิจัยและพัฒนากันทางด้านนี้ไม่มากนัก การวิจัยที่เกี่ยวข้องกับ PC ได้แก่ การต่อ PC เข้ากับไมโครคอมพิวเตอร์ เพื่อให้สามารถส่งถ่ายโปรแกรมและข้อมูลระหว่างกัน[26] การจำลองการทำงานของ PC บนเครื่องไมโครคอมพิวเตอร์[27] เป็นต้น นอกจากนั้นยังมีการวิจัยและพัฒนาที่เกี่ยวข้องกับ PC โดยเป็นการใช้ไมโครโปรเซสเซอร์ในการควบคุม เช่น การพัฒนาเครื่องควบคุมสิฟท์โดยสารโดยใช้ไมโครคอมพิวเตอร์[22,23,24] การพัฒนาเครื่องควบคุมกระบวนการผลิตในโรงงานอาหารสัตว์[18,19,20] การพัฒนาเครื่องควบคุมการบรรจุถุง(4) งานวิจัยเหล่านี้เป็นงานวิจัยพัฒนาที่ใช้แนวความคิดและโครงสร้างระบบเช่นเดียวกับ PC นี้ คณะผู้วิจัยได้ทำการวิจัยการสร้าง PC ขึ้นเป็นครั้งแรกในประเทศ โดยได้รับทุนจากศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ ในโครงการหลักชื่อว่าโครงการพัฒนาอิเล็กทรอนิกส์เพื่ออุตสาหกรรม ในปี 2531 โครงการวิจัยแล้วเสร็จในปี 2532 และได้ส่งรายงานวิจัยเป็นที่เรียบร้อยแล้ว[28]

1.2 สถานภาพของเรื่องที่จะทำการวิจัยและพัฒนา

ในทุกกระบวนการผลิตในโรงงานอุตสาหกรรมที่ต้องใช้เครื่องจักร จะต้องมีการควบคุมลำดับการทำงานของอุปกรณ์ต่าง ๆ ให้ทำงานสัมพันธ์กัน ไม่ว่าจะเป็นจังหวะการทำงาน และเงื่อนไขที่อุปกรณ์จะทำงาน เครื่องจักรที่ต้องการมีการควบคุม ได้แก่ สายพานลำเลียง ในอุตสาหกรรมทุกชนิด แขนกล สำหรับจับและโยกย้ายชิ้นงาน การสตาร์ทเครื่องจักร เช่น หม้อไอน้ำ กลุ่มของมอเตอร์ เครื่องจักรจำพวกเครื่องกลึง เครื่องกัด เครื่องไส การลำเลียงและการเลือกวัตถุดิบในโรงงานอาหารสัตว์ แม้แต่เครื่องจักรขนาดเล็ก เช่น เครื่องฉีดพลาสติค เครื่องสกรีนผ้า เครื่องปั่นด้าย หรือเครื่องทอผ้า ก็ต้องมีการควบคุมลำดับทั้งสิ้น

PC เป็นเครื่องควบคุมเอนกประสงค์สามารถนำไปใช้ควบคุมลำดับการทำงานของเครื่องจักรได้ทุกชนิด เพียงแต่ป้อนโปรแกรมควบคุมที่เข้ากับการทำงานของเครื่องจักรแต่ละเครื่องให้เท่านั้น ในปัจจุบันทีแนวโน้มที่เห็นได้ชัดเจนว่า PC จะเข้ามาแทนวงจรรีเลย์แม่เหล็กไฟฟ้าเดิมที่มีการใช้งานอยู่ เพราะมีข้อดีกว่าหลายประการ เช่นประกอบวงจรได้รวดเร็วกว่า พัฒนากละแก้ไขวงจรได้รวดเร็วกว่ามีความซับซ้อนในการควบคุมได้มากกว่า และจะมีราคาถูกกว่าในบางกรณี ความนิยมของ PC นี้จะเห็นได้จากตัวเลขยอดขาย PC ในประเทศ จากการสอบถาม บริษัท OMRON-TRISAK ที่มีการจำหน่าย PC ยี่ห้อ OMRON นั้น ยอดขาย PC ขนาดเล็ก ในปี 2528 จำนวน 50 ตัว เพิ่มเป็น 100 ตัว ในปี 2530 และเพิ่มเป็นประมาณ 300 ตัว ในปี 2532 PC ที่ติดมากับเครื่องจักรที่นำเข้ามาใช้ในการผลิตในประเทศ ในโรงงานที่ได้

รับการส่งเสริมจาก BOI จำนวนประมาณ 1000 โรงงานจะมีมากถึง 3000 ตัว ซึ่งเป็นการเพิ่มอย่างมากในระยะที่มีการลงทุนจากต่างประเทศนี้ ปัจจุบันมีตัวแทนจำหน่าย PC ของต่างประเทศกว่า 10 ราย ประมาณกันว่า PC จะมีตลาดในประเทศในปี 2533 กว่า 1000 เครื่อง และจะเพิ่มขึ้นเรื่อย ๆ ด้วย อัตราเพิ่มที่สูงมากเนื่องจากมีตลาดซึ่งเป็นโรงงานใหม่และมีตลาดที่เป็นโรงงานเก่า นำ PC มาทดแทนวงจรีเลย์ เมื่อวิศวกรและช่างเทคนิคในประเทศรู้จักวิธีใช้ PC อย่างดีแล้ว จะมีตลาดเพิ่มมากขึ้นอย่างมากมาย

PC ที่ขายได้ดีที่สุดในปัจจุบันเป็น PC ขนาดเล็กและขนาดกลาง มีจำนวนอินพุท และเอาท์พุทประมาณ 100 จุด เนื่องจากใช้ในการควบคุมเครื่องจักรแบบตัวเดียวโดดๆ PC ขนาดเล็กซึ่งมีความสามารถการรับคำสั่ง ซีเคว็นซ์ พื้นฐานจะใช้ในงานควบคุมเครื่องจักรง่าย ๆ แต่ PC ขนาดกลางที่มีความสามารถในการับคำสั่งทางด้านการจัดการข้อมูล จะใช้ในการควบคุมเครื่องจักรที่ใหญ่ขึ้น เช่น เครื่องฉีดพลาสติค เครื่องบรรจุกล่อง เครื่องนับเม็ดยา เครื่องผสมอาหารสัตว์ เป็นต้น

การวิจัยครั้งนี้ จะเป็นการวิจัยต่อเนื่องจากเรื่องการพัฒนาเครื่องควบคุมที่โปรแกรม ซึ่งเป็นเครื่อง PC ขนาดเล็ก มีอินพุท/เอาท์พุท 40 จุด สามารถโปรแกรมได้ 1000 Step และควบคุมได้เฉพาะการทำลอจิกของสัญญาณเปิด-ปิด การวิจัยจะเป็นการพัฒนาเครื่องควบคุมชนิดโปรแกรมได้ขึ้นใหม่ ให้มีลักษณะโครงสร้างของระบบเป็นแบบโมดูล ผู้ใช้สามารถเลือกจำนวนและชนิดของอินพุท/เอาท์พุทได้เอง มีโมดูลสำหรับติดต่อกับสัญญาณพิเศษต่าง ๆ และพัฒนาให้เครื่องมีความสามารถในการจัดการข้อมูลได้ ลักษณะของเครื่องควบคุมชนิดต่าง ๆ และพัฒนาให้เครื่องมีความสามารถในการจัดการข้อมูลได้ ลักษณะของเครื่องควบคุมที่ได้พัฒนานี้ จะสามารถติดต่ออินพุท/เอาท์พุทได้สูงสุด 512 จุด สามารถเขียนโปรแกรมได้สูงสุด 4000 Step และภาษาที่ใช้เขียนโปรแกรม จะเป็นภาษาแลดเดอร์ โดยการพัฒนาเพิ่มฟังก์ชั่นบล็อก (Function Block) สำหรับคำสั่งฟังก์ชั่นต่าง ๆ เช่น คำสั่งคำนวณ คำสั่งเปรียบเทียบคำสั่งเคลื่อนย้ายข้อมูล คำสั่งลอจิก คำสั่งควบคุมพิเศษต่าง ๆ เปลี่ยนวิธีแปลงคำสั่งแลดเดอร์ไปเป็นภาษาเครื่องใหม่ เพื่อให้เวลาในการทำงานของเครื่องเร็วขึ้น แก้ไขปัญหาทางด้านสัญญาณรบกวนที่เข้ามาทางสายอินพุท และรบกวนการทำงานของเครื่อง ในระบบบัส อันเป็นจุดอ่อนของเครื่อง PC ขนาดเล็กที่ได้วิจัยไปในปี 2531 โดยการออกแบบทางด้านชั้วเข้ารับสัญญาณใหม่ ออกแบบระบบบัสใหม่และออกแบบกล่องโลหะที่บรรจุวงจร ให้สามารถลดการรบกวนจากสัญญาณภายนอก

การพัฒนา PC ในประเทศจะมีความสำคัญ ไม่เพียงแต่จะเป็นการลดการนำเข้าแต่จะเป็นการสนับสนุนให้อุตสาหกรรมให้อุตสาหกรรมขนาดเล็กและกลางในประเทศมีโอกาสปรับปรุงเครื่องจักรและประสิทธิภาพการผลิตด้วย

1.3 วัตถุประสงค์ของโครงการ

พัฒนาเครื่องควบคุมที่โปรแกรมได้ขนาดกลางที่มีความสามารถทางด้านการประมูลข้อมูล จนสามารถนำไปใช้ในอุตสาหกรรมได้

1.4 เป้าหมายของโครงการที่จะได้รับเมื่อจบโครงการ

สร้างต้นแบบของ PC ชนิดที่มีความสามารถทางด้านการจัดการข้อมูลที่มีอินพุท/เอาท์พุท 64 จุด ขยายได้สูงสุด 512 จุด มีชุดคำสั่ง สำหรับการโปรแกรม Ladder daigram มีความจำสำหรับเก็บโปรแกรม 4000 คำสั่ง มีอุปกรณ์สำหรับป้อนโปรแกรมเข้าเครื่องและตรวจสอบดูสภาวะการทำงานของเครื่องได้ PC ต้นแบบ จะมีคุณสมบัติใกล้เคียงกับผลิตภัณฑ์ที่มาจากต่างประเทศ แต่มีราคาถูกกว่าและจะทำการทดสอบจนสามารถใช้งานในโรงงานอุตสาหกรรมได้จริง

1.5 ผลงานที่ได้รับเมื่อจบโครงการ

ได้ต้นแบบเครื่องควบคุมที่โปรแกรมได้ ขนาดกลางชนิดที่ความสามารถในการจัดการข้อมูล สามารถใช้งานได้จริง โดยมี ฮาร์ดแวร์แบ่งเป็นโมดุลต่างๆ ดังต่อไปนี้

1. CPU โมดุล (CPU Module)
2. อินพุทโมดุล (Input Module)
3. เอาท์พุทโมดุล (Output Module)
4. อานาลอกอินพุทโมดุล (Analog Input Module)
5. โมดุลสำหรับตั้งค่า BCD (BCD Set Module)
6. โมดุลตัวนับความเร็วสูง (High Speed Counter Module)
7. โมดุลเชื่อมต่อกับคอมพิวเตอร์ (Microcomputer Link Module)
8. ส่วนโปรแกรม (Programming Panel)

1.6 ขอบเขตการวิจัยและพัฒนา

1. ออกแบบเครื่องควบคุมชนิดโปรแกรมได้ที่มีลักษณะเป็นโมดูล สามารถต่ออินพุท/เอาท์พุต ได้สูงสุด 512 จุด
2. พัฒนาซอฟท์แวร์ของเครื่องควบคุมชนิดโปรแกรมได้ ให้มีความสามารถในการจัดการข้อมูลได้
3. ออกแบบสร้างโมดูลต่างๆ ได้แก่ โมดูลหน่วยประมวลผล, โมดูลย์คีย์บอร์ดและแสดงผล, โมดูลอินพุท, โมดูลเอาท์พุท และโมดูลอานาลอกอินพุท
4. ทดลองเครื่องต้นแบบกับเครื่องจักรจริงในโรงงานอุตสาหกรรม

ไม่มีหน้า

ตารางเวลาการวิจัยและพัฒนาในแต่ละขั้นตอน

ขั้นตอน / ระยะเวลา	เดือนที่ 1	2	3	4	5	6	7	8	9	10
1. ศึกษาผลิตภัณฑ์	x	x	x	x	x					
2. ออกแบบฮาร์ดแวร์		x	x	x	x	x	x			
3. พัฒนาซอฟท์แวร์				x	x	x	x	x	x	x	x	x	x	x
4. ออกแบบผลิตภัณฑ์						x	x	x	x	x	x	x	x	x
5. ทดสอบ										x	x	x	x	x

1.7.2 เครื่องมือและครุภัณฑ์ที่ใช้ในการวิจัย

1. เครื่องมือวัดอิเล็กทรอนิกส์พื้นฐาน เช่น มัลติมิเตอร์ ออสซิลสโคป พัลส์เยนเนอเรเตอร์

2. ชุดพัฒนาระบบไมโครโปรเซสเซอร์ (Microprocessor development system) ซึ่งมีชุดไมโครคอมพิวเตอร์ Emulator และซอฟท์แวร์

3. ชุดจำลองการควบคุมลำดับ (Sequence control model plant)

ภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย ได้จัดเตรียมเครื่องเหล่านี้ไว้แล้วที่ห้องปฏิบัติการวัดคุมทางอุตสาหกรรมและห้องปฏิบัติการ EDL (Electronic design Laboratory)

1.7.3 ระยะเวลาทำการวิจัยรวม 10 เดือน

เริ่ม 1 ต.ค.2533 สิ้นสุด 30 ก.ย.2534

1.7.4 สถานที่ทำการวิจัยและพัฒนา

ใช้ห้องปฏิบัติการวัดคุมทางอุตสาหกรรมและห้องปฏิบัติการ EDL ในภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

1.8 ผลกระทบทางเศรษฐกิจและสังคม

6.9.1 ลดการนำเข้า PC จากต่างประเทศ มูลค่าปีหนึ่ง ไม่ต่ำกว่า 100 ล้านบาทและจะเพิ่มมูลค่ามากขึ้นตามการขยายตัวของอุตสาหกรรม

6.9.2 PC ที่พัฒนาขึ้นมาจะมีราคาถูก และเหมาะกับการใช้งานในประเทศ เปิดโอกาสให้อุตสาหกรรมขนาดเล็กและกลางนำไปใช้งาน สามารถปรับปรุงการทำงานของเครื่องจักรและเพิ่มประสิทธิภาพการผลิตได้ มีผลต่อเศรษฐกิจส่วนรวมของประเทศ

บทที่ 2

การออกแบบเครื่องควบคุมชนิดโปรแกรมได้

การออกแบบเครื่องควบคุมชนิดโปรแกรมได้ (PC) นี้เราควรคำนึงถึงปัจจัยหลายอย่าง ได้แก่ ความต้องการของผู้ใช้งาน ความเหมาะสมของระบบ ประสิทธิภาพและความสามารถของเครื่อง ความมีเสถียรภาพของเครื่อง ซึ่งอาจแบ่งออกเป็นหัวข้อต่าง ๆ คือ การออกแบบระบบ การออกแบบฮาร์ดแวร์ การออกแบบซอฟท์แวร์ และการออกแบบการจัดเก็บข้อมูล ซึ่งการออกแบบแต่ละหัวข้อนั้นอาจจะสัมพันธ์และเกี่ยวข้องกับหัวข้ออื่นด้วย ในบทนี้จะกล่าวถึงแนวคิดและหลักการกว้างๆ เท่านั้น สำหรับรายละเอียดของบางหัวข้อจะได้กล่าวถึงในบทต่อไป

2.1 ลักษณะของเครื่องควบคุมชนิดโปรแกรมได้

เครื่องควบคุมชนิดโปรแกรมได้ที่มีอยู่ในปัจจุบันแบ่งออกได้เป็นหลายขนาด และมีโครงสร้างของระบบแตกต่างกัน การแบ่งขนาดของ PC อาจพอแบ่งได้เป็น 3 ขนาด[11] โดยการพิจารณาจากจำนวนอินพุท/เอาท์พุทและขนาดของหน่วยความจำที่สามารถเขียนคำสั่งได้ดังนี้

1. เครื่อง PC ขนาดเล็ก
2. เครื่อง PC ขนาดกลาง
3. เครื่อง PC ขนาดใหญ่

เครื่อง PC ขนาดเล็กโดยทั่วไปจะมีจำนวนอินพุท/เอาท์พุท สูงสุดประมาณ 40/40 จุด และมีหน่วยความจำในการเก็บโปรแกรมได้ประมาณ 1 K Step ลักษณะของเครื่องจะมีลักษณะที่กระทัดรัด มีอุปกรณ์อินพุท/เอาท์พุทต่าง ๆ อยู่รวมกัน การเพิ่มขยายทำได้จำกัดและมักรับสัญญาณได้เฉพาะสัญญาณจำพวกเปิด-ปิดเท่านั้น เหมาะสำหรับใช้กับเครื่องจักรขนาดเล็กที่มีการควบคุมแบบลำดับที่เป็นลอจิก(Logic) เท่านั้น

เครื่อง PC ขนาดกลาง จะมีจำนวนอินพุท/เอาท์พุท สูงสุดประมาณ 128/128 จุด มีหน่วยความจำในการเก็บโปรแกรมได้ประมาณ 4 K ขึ้น ส่วนเครื่อง PC ขนาดใหญ่จะมีจำนวน

* ต่อไปนี้จะใช้คำว่า PC แทนเครื่องควบคุมชนิดโปรแกรมได้

อินพุท/เอาท์พุท และหน่วยความจำสูงกว่านี้ เครื่อง PC ขนาดกลางและใหญ่นี้การออกแบบอินพุท เอาท์พุท และอุปกรณ์ต่าง ๆ มักเป็นแบบโมดูล สามารถเพิ่มขยายได้ง่ายมีโมดูลสำหรับรับสัญญาณพิเศษต่าง ๆ มีคำสั่งและฟังก์ชั่นในการคำนวณต่าง ๆ สามารถติดต่อสื่อสารกันเองและกับคอมพิวเตอร์ได้

ขนาด PC	จำนวนอินพุท/เอาท์พุทสูงสุด	ขนาดหน่วยความจำ
ขนาดเล็ก	40/40	1 K
ขนาดกลาง	128/128	4 K
ขนาดใหญ่	>128/>128	>4 K

ตารางที่ 2.1 แสดงการแบ่งขนาดของ PC

	PC ขนาดเล็ก	PC ขนาดกลาง
รุ่น	Mitsubishi F 240	Allon-Bradley Mini PLC 2
ซีพียู	Intel 8031	Zilog Z 80 A
เวลาหนึ่งรอบ	7 ms ต่อ 1 K	20 ms ต่อ 1 K
จำนวน I/O สูงสุด	40 จุด	256 จุด
อนาลอกอินพุท	ไม่มี	มี
ภาษาที่ใช้	แลดเดอร์	แลดเดอร์

ตาราง 2.2 แสดงตัวอย่างลักษณะของ PC ขนาดเล็ก และขนาดกลาง[11]

สำหรับเครื่อง PC ที่ออกแบบต้องคำนึงถึงความสามารถและความสะดวกในการใช้งาน โดยทั่วไปหลักเกณฑ์ในการพิจารณาออกแบบเครื่อง PC ได้แก่

- ควรมีลักษณะที่ง่ายในการเพิ่มขยายระบบ หรือการซ่อมบำรุง
- สามารถติดต่อกับสัญญาณอินพุท - เอาท์พุทมาตรฐานต่าง ๆ ได้
- ภาษาที่โปรแกรมควรจะง่าย

- สามารถแก้ไขเปลี่ยนแปลงโปรแกรมที่หน้างาน(On-site)ได้ง่าย
- ป้องกันการรบกวนจากสัญญาณภายนอกได้ดี

2.2 แนวความคิดในการออกแบบระบบ

การออกแบบเครื่องควบคุมชนิดโปรแกรมได้ ของการวิจัยนี้มีแนวความคิดในการออกแบบส่วนต่าง ๆ ดังต่อไปนี้

1. **ควรมีลักษณะเป็นโมดูล** เพื่อให้ความสะดวกในการใช้งานแก่ผู้ใช้ โดยสามารถถอดเปลี่อน และซ่อมบำรุงโมดูลต่าง ๆ ของเครื่อง PC ได้อย่างสะดวก โมดูลเหล่านี้จะประกอบด้วยโมดูลประมวลผล โมดูลแสดงผลและคีย์บอร์ด โมดูลอินพุท/เอาท์พุท ชนิดต่างๆ การที่ออกแบบเป็นโมดูลนี้จะทำให้การออกแบบสร้างโมดูลอินพุท/เอาท์พุท เพิ่มเติมในอนาคตทำได้สะดวก

2. **ตำแหน่งในการต่อโมดูลต่างๆ ไม่ถูกกำหนดโดยผู้ออกแบบ** ผู้ใช้สามารถเสียบโมดูลอินพุท/เอาท์พุทที่ตำแหน่งใดๆ ก็ได้ โดยที่ระบบจะต้องทำการตรวจสอบเองโดยอัตโนมัติขณะเริ่มต้นทำงานว่ามีโมดูลชนิดใดบ้างต่ออยู่ ณ ตำแหน่งใดบ้าง การออกแบบลักษณะนี้จะให้ความสะดวกแก่ผู้ใช้ขณะติดตั้ง

3. **ผู้ใช้สามารถเลือกจำนวนอินพุทและเอาท์พุทได้เอง** ระบบที่ออกแบบนี้ไม่ควรกำหนดจำนวนของอินพุท/เอาท์พุทแต่ละชนิด ควรให้ผู้ใช้เลือกจำนวนของอินพุท และจำนวนของเอาท์พุทแต่ละชนิดได้เอง เพื่อให้เหมาะสมกับงานควบคุมแต่ละชนิด โดยเครื่องจะกำหนดหมายเลขตำแหน่งต่างๆ ของอินพุท/เอาท์พุทให้เรียงลำดับตามการต่อโมดูลต่างๆ โดยอัตโนมัติ ระบบที่ออกแบบควรที่จะสามารถต่อกับอินพุท และเอาท์พุท รวมกันได้สูงสุด 256 จุด ถ้าต้องการจำนวนอินพุท และเอาท์พุทมากขึ้นก็สามารถทำได้โดยไม่ยากนัก

4. **คำสั่งในการโปรแกรม** เนื่องจากเครื่อง PC ที่ออกแบบต้องมีความสามารถในการจัดการข้อมูล ดังนั้นการใช้คำสั่งแลดเดอร์เบื้องต้น (เช่น การอ่านอินพุท การทำตรรกศาสตร์ การเขียนเอาท์พุท) เพียงอย่างเดียวจึงไม่เพียงพอ จำเป็นต้องเพิ่มคำสั่งอื่น ๆ เข้ามาอีกอันได้แก่ คำสั่งเกี่ยวกับการเคลื่อนย้ายข้อมูล คำสั่งคำนวณ คำสั่งเปรียบเทียบข้อมูล คำสั่งในการแปลงรหัส คำสั่งทางตรรกศาสตร์(หรือคำสั่งลอจิก) และคำสั่งควบคุมการทำงานพิเศษอื่นๆ ดังนั้นภาษาที่ใช้จะเป็นภาษา LADDER DIAGRAM WITH FUNCTION BLOCK ซึ่งภาษานี้จะเหมาะในการพัฒนากับระบบที่มีการแสดงผลที่ไม่ใช่แบบกราฟฟิค

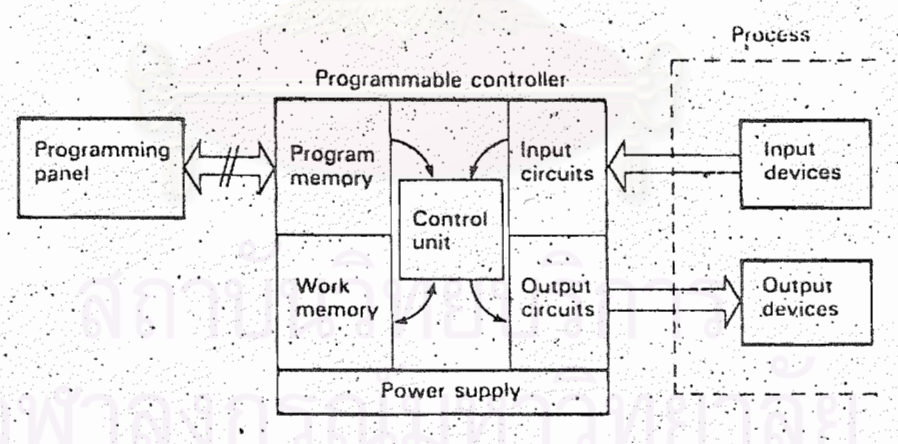
5. **การป้อนโปรแกรมและแสดงผล** ระบบที่ออกแบบควรจะมีความสะดวกในการใช้การป้อนโปรแกรม และต้องสามารถเปลี่ยนแปลงแก้ไขโปรแกรมที่หน้างาน(On site)ได้ สำหรับระบบที่ออกแบบจะมีตัวป้อนโปรแกรม (ซึ่งมีส่วนแสดงผลอยู่ในตัวเดียวกัน) โดยมีลักษณะเป็นแบบมือถือ สามารถถอดเข้าออกกับตัวเครื่อง PC ได้ การวิจัยในอนาคตอาจมีการพัฒนาปรับปรุง

ระบบให้ใหญ่ขึ้น ออกแบบให้มีพอร์ทสื่อสารกับคอมพิวเตอร์

6. จำนวนคำสั่งที่สามารถโปรแกรมได้ จำนวนคำสั่งนี้ควรจะสัมพันธ์กับจำนวนอินพุทและเอาท์พุทสูงสุดของระบบด้วย เครื่องที่ออกแบบมีจำนวนอินพุท/เอาท์พุทสูงสุด 256 จุด จำนวนคำสั่งที่สามารถโปรแกรมได้มีจำนวนสูงสุดได้ถึง 4000 ชั้น หน่วยความจำที่ใช้ในการเก็บข้อมูลอื่น ๆ ควรมีอย่างละไม่น้อยกว่า 256 ตำแหน่ง หน่วยความจำเหล่านี้ได้แก่ รีเลย์ช่วยทั้งแบบ Rententive Relay และแบบ Non-Retentive Relay และหน่วยความจำในการเก็บข้อมูลสำหรับการจัดการข้อมูล

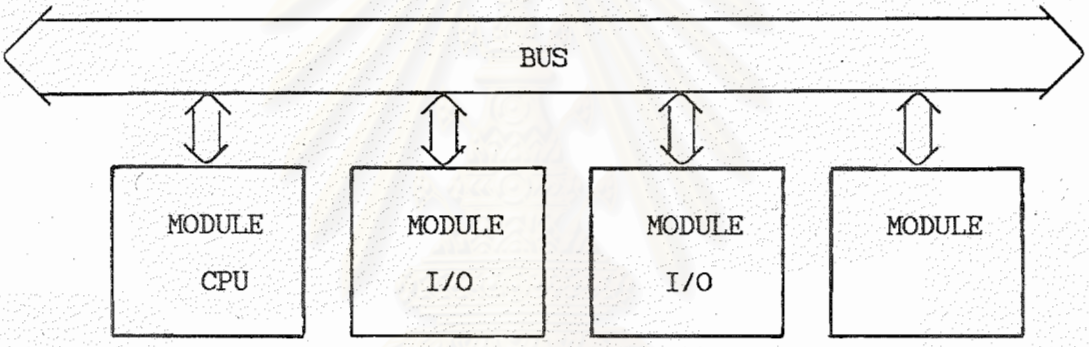
2.3 แนวความคิดในการออกแบบฮาร์ดแวร์

โดยหลักการแล้วเครื่อง PC เป็นไมโครคอมพิวเตอร์(Special-purpose micro computer) ที่ออกแบบเฉพาะสำหรับงานควบคุมแบบลำดับ ส่วนประกอบหลัก ๆ ได้แก่ตัวประมวลผล(CPU) หน่วยความจำ และอุปกรณ์อินพุท/เอาท์พุท โดยมีลักษณะโครงสร้างดังรูปที่ 2.1



รูปที่ 2.1 แสดงลักษณะโครงสร้างของ PC

การออกแบบเครื่อง PC ในส่วนฮาร์ดแวร์นั้นก็ยังคงมีส่วนประกอบหลักๆเหมือนเดิม อันได้แก่ ตัวประมวลผล หน่วยความจำ อุปกรณ์อินพุท/เอาท์พุท การวิจัยจะเน้นการปรับปรุงลักษณะการเชื่อมโยงของส่วนต่าง ๆ ใหม่เพื่อให้ระบบมีคุณสมบัติตามหัวข้อ 2.2.1 - 2.2.3 ประเด็นที่สนใจได้แก่ การออกแบบให้มีลักษณะเป็นโมดูลเพื่อความสะดวกในการขยายระบบ และความสะดวกในการซ่อมบำรุง การออกแบบในลักษณะโมดูลจำเป็นต้องมีบัสในการติดต่อระหว่างโมดูลต่าง ๆ และการแบ่งส่วนประกอบต่าง ๆ ออกเป็นโมดูล ซึ่งควรพยายามออกแบบโมดูลต่าง ๆ ให้เป็นอิสสระคือ ไม่จำเป็นต้องเสียบทุกโมดูลในระบบจึงทำงานได้ ระบบควรสามารถทำงานได้ขณะที่เสียบโมดูลพื้นฐาน(Minimum feature) เท่านั้น การออกแบบในลักษณะโมดูลนี้อาจมีลักษณะโครงสร้างดังรูปคือ



รูปที่ 2.2 แสดงลักษณะโครงสร้างของ PC แบบโมดูล

ข้อพิจารณาอื่นในการออกแบบฮาร์ดแวร์ของเครื่อง PC ได้แก่ หลีกเลี่ยงการออกแบบฮาร์ดแวร์ในลักษณะที่ทำให้เสียเวลาซีพียูในการทำงาน เนื่องจากว่าข้อสำคัญอันหนึ่งของเครื่อง PC คือ จะต้องพยายามทำงานเร็วที่สุด เพื่อให้เวลาในการทำงาน 1 รอบ (Scantime) มีค่าน้อยที่สุด ซึ่งปกติควรจะน้อยกว่า 1ØØ mSEC ถ้าเวลานี้ช้าเกินไปอาจทำให้ซีพียูไม่สามารถรับรู้การเปลี่ยนแปลงของสัญญาณอินพุทต่าง ๆ ได้ทัน จะทำให้การควบคุมการทำงานผิดพลาดได้ ฮาร์ดแวร์ในลักษณะที่เสียเวลาการทำงานได้แก่ การออกแบบวงจรคีย์บอร์ดในลักษณะที่ใช้ซีพียูสแกนอ่านค่าต่าง ๆ หรือการแสดงผลในลักษณะใช้ซีพียูสแกนข้อมูลเอง รวมทั้งวงจรต่าง ๆ ที่ต้องใช้ซีพียูมาวนอ่านค่าสถานะของวงจร สำหรับเครื่อง PC ที่ออกระบบก็พยายามหลีกเลี่ยงปัญหาเหล่านี้ โดยฮาร์ดแวร์สำหรับอ่านค่าคีย์บอร์ดจะใช้ไอซีสำหรับอ่านค่าคีย์บอร์ดโดยเฉพาะทำให้ลดเวลาการทำงานซีพียูในการสแกนคีย์บอร์ด ส่วนการแสดงผลเลือกใช้ LCD ขนาด 16 x 2 ตัวอักษรซึ่งมีลักษณะเป็นโมดูลสำเร็จ โดยเพียงซีพียูส่งรหัสคำสั่งหรือข้อมูลที่ต้องการแสดงผลมาให้เท่านั้น โมดูล LCD จะนำคำสั่งและข้อมูลที่ได้ไปแสดงผลให้โดยอัตโนมัติ

การป้องกันการรบกวนจากสัญญาณภายนอก เป็นปัญหาที่สำคัญอันหนึ่งของเครื่อง PC เนื่องจาก PC ประกอบด้วยไมโครโปรเซสเซอร์เป็นอุปกรณ์หลัก ซึ่งทำงานที่ความเร็วสูงและระดับแรงดันไฟต่ำและมีความไวต่อการรบกวนจากสัญญาณแม่เหล็กและสัญญาณไฟฟ้า การออกแบบเครื่อง PC จึงต้องคำนึงถึงปัญหาเหล่านี้ด้วย

เครื่อง PC ที่ออกแบบนี้จะประกอบด้วยโมดูลต่าง ๆ หลายโมดูลคือ

โมดูลประมวลผล ประกอบด้วยซีพียู หน่วยความจำสำหรับเก็บโปรแกรมควบคุมการทำงานของเครื่อง และหน่วยความจำที่ใช้เก็บโปรแกรมของผู้ใช้ สำหรับหน่วยความจำที่เก็บโปรแกรมของผู้ใช้จะต้องมีระบบจ่ายไฟสำรอง เพื่อไม่ให้โปรแกรมของผู้ใช้ที่เก็บไว้สูญหายเมื่อปิดเครื่อง นอกจากนั้นยังประกอบด้วยวงจรฐานเวลา เพื่อสร้างฐานเวลาสำหรับการทำงานของคำสั่งไทเมอร์ต่าง ๆ และวงจรวอทช์ดอกไทเมอร์ โมดูลนี้เป็นโมดูลที่มีความสำคัญมาก และจะต้องเสียบอยู่เสมอในเครื่อง PC จึงจะสามารถทำงานได้

โมดูลแสดงผล และคีย์บอร์ด โมดูลนี้ควรออกระบบในลักษณะที่ถ้าไม่เสียบโมดูลนี้เครื่อง PC จะต้องทำงานได้ตามปกติ โมดูลนี้มีหน้าที่ในการแสดงผลข้อมูลต่าง ๆ ในขณะที่มีการป้อนและแก้ไขโปรแกรม หรือใช้แสดงค่าของข้อมูลต่าง ๆ ที่ผู้ใช้ต้องการดู และมีหน้าที่รับค่าคีย์บอร์ดจากผู้ใช้ในการป้อนแก้ไขโปรแกรมต่าง ๆ รวมทั้งการป้อนคำสั่งต่าง ๆ โมดูลแสดงผลและคีย์บอร์ดที่ออกแบบมีลักษณะเป็น Handle Programming Console ซึ่งเพิ่มความสะดวกในการใช้งานของผู้ใช้ นอกจากนั้นโมดูลนี้ยังออกแบบให้มีพอร์ทสำหรับติดต่อสื่อสารกับคอมพิวเตอร์ไว้ด้วย เพื่อใช้ในการพัฒนาระบบในอนาคต

โมดูลอินพุท ทำหน้าที่รับสัญญาณจากภายนอกส่งให้โมดูลประมวลผล สัญญาณที่รับมีลักษณะเป็นแบบ On-off เท่านั้น โดยจะมีวงจรฟิลเตอร์ (Filter) และวงจรไอโซเลเตอร์ (Isolater)

โมดูลเอาท์พุท ทำหน้าที่รับสัญญาณจากโมดูลประมวลผล และคงสถานะของข้อมูลไว้แล้วเพื่อควบคุมการทำงานของอุปกรณ์ภายนอก สำหรับเอาท์พุทของโมดูลจะใช้เป็นหน้าสัมผัสของรีเลย์

โมดูลอนาลอกอินพุท เป็นโมดูลสำหรับรับสัญญาณแบบอนาลอกจากภายนอก ซึ่งจะออกแบบให้รับสัญญาณมาตรฐาน 4-20 mA_{dc} หรือ 1-5 V_{dc} โดยจะนำสัญญาณที่ได้รับส่งไปยังวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล(A/D) และผ่านวงจรอ๊อฟโต้คัปเปอร์ไปยังหน่วยประมวลผล ขนาดความแม่นยำของวงจร A/D ที่ออกแบบมีขนาด 12 บิท ซึ่งจะมีค่าเป็นเลขฐาน 16 อยู่ระหว่าง ØØØØ ถึง ØFFF

โมดูลกำหนดค่าตัวเลข จุดประสงค์ของโมดูลนี้ออกแบบมาเพื่อเพิ่มความสะดวกในการใช้งานของผู้ใช้ โดยผู้ใช้งานสามารถเปลี่ยนค่าพารามิเตอร์ของฟังก์ชั่นต่าง ๆ โดยการเปลี่ยนตัวเลขที่หน้าบัดของโมดูลนี้ ซึ่งจะใช้งานได้ง่ายกว่าการไปเปลี่ยนค่าที่โมดูลแสดงผลและคีย์บอร์ด

ซึ่งมีขั้นตอนที่ยุ่งยากกว่า

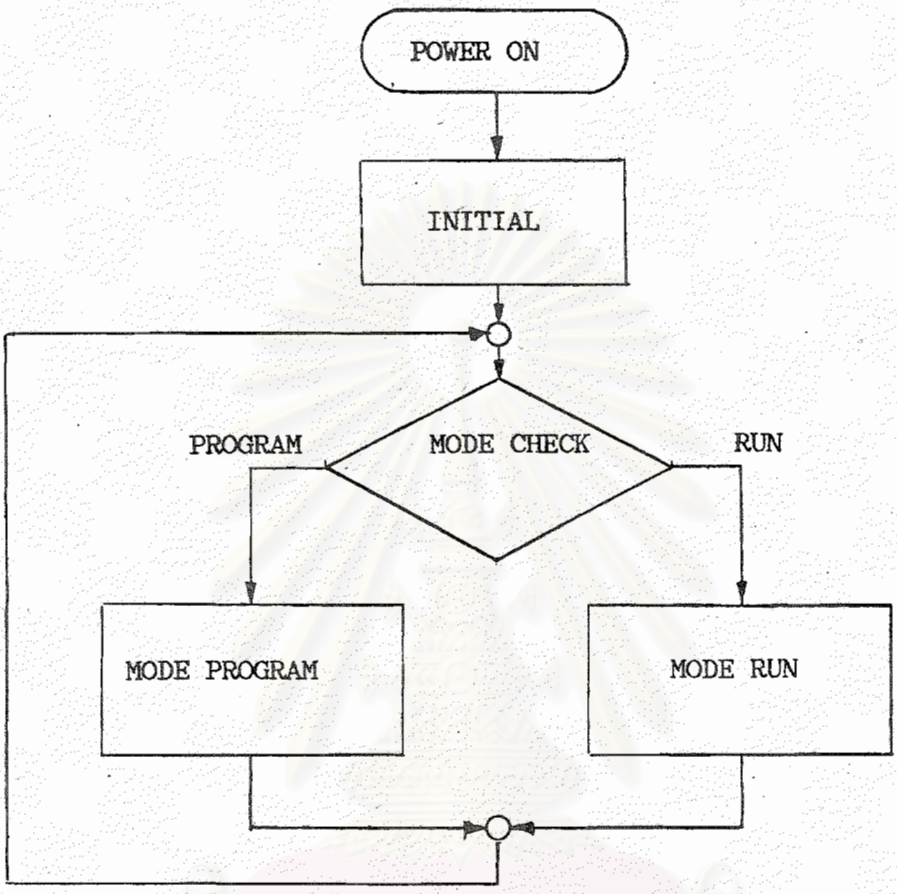
เครื่อง PC นี้ สามารถออกแบบโมดูลอินพุท และเอาท์พุทแบบอื่น ๆ มาเสียบต่อในเครื่องได้ง่าย โดยเครื่องที่ออกแบบไว้นี้สามารถเสียบโมดูลต่าง ๆ ได้สูงสุดถึง 16 โมดูล

2.4 แนวความคิดในการออกแบบซอฟท์แวร์

โปรแกรมควบคุมการทำงานของเครื่อง PC ควรออกแบบให้มีประสิทธิภาพที่สุด โดยพิจารณาจากความต้องการในการทำงานของแต่ละโหมด สำหรับเครื่อง PC ที่ออกแบบนี้จะแบ่งโหมดในการทำงานได้ 2 โหมดหลักคือ

1. โหมดโปรแกรม (Program mode)
2. โหมดทำงาน (Run mode)

โหมดโปรแกรมมีหน้าที่สำคัญคือ ทำหน้าที่รับการป้อนโปรแกรมจากผู้ใช้ โดยโปรแกรมจะต้องรับ และโต้ตอบการกดคีย์ต่าง ๆ ของผู้ใช้ ซึ่งอาจเป็นการกดคำสั่งแลดเดอร์ต่างๆ การกดชนิดของข้อมูล และการใส่ค่าของข้อมูล โดยที่โปรแกรมควบคุมในโหมดนี้จะต้องแยกแยะการกดคีย์ชนิดต่าง ๆ และนำมารวมกันเป็นคำสั่งแลดเดอร์ จากนั้นก็แปลงเป็นภาษาเครื่องที่ซีพียูสามารถทำงานได้ แล้วนำลงไปเก็บยังหน่วยความจำสำหรับผู้ใช้ ซึ่งการทำงานนี้ยังรวมถึงการตรวจสอบแก้ไขโปรแกรมต่าง ๆ ด้วย เช่นการแก้ไขฟังก์ชั่นหรือค่าข้อมูลของคำสั่งนั้น การลบคำสั่งออก(Delete) การแทรกคำสั่งใหม่เพิ่มลงไป(Insert) ซึ่งการทำงานของโปรแกรมควบคุมในส่วนที่กล่าวมานี้ค่อนข้างจะยุ่งยากและมีความซับซ้อน เนื่องจากคำสั่งต่างๆ มีถึง 64 คำสั่ง ซึ่งแต่ละคำสั่งจะมีลักษณะที่แตกต่างกัน เช่น วิธีการกดคำสั่งต่าง ๆ ความยาวของแต่ละคำสั่งที่แตกต่างกัน และรูปแบบของคำสั่งที่แตกต่างกัน เป็นต้น หน้าที่อีกอย่างของโปรแกรมควบคุมในโหมดนี้คือ การรับคำสั่งของเครื่อง PC ได้แก่ การลบโปรแกรมของผู้ใช้ทั้งหมด การลบค่าของหน่วยความจำที่เก็บข้อมูล การแสดงสถานะหรือการแก้ไขข้อมูลของเครื่อง การออกแบบโปรแกรมควบคุมการทำงานของเครื่องในโหมดนี้จะใช้หลักการของ KEYBOARD PARSING[10] ซึ่งเป็นอัลกอริธึมที่นิยมใช้ในการรับคีย์บอร์ดของอินสตรูเมนต์ต่าง ๆ มีหลักการคล้ายกับการทำงานของวงจรซีเควันซ์ โดยการเทียบคีย์บอร์ดที่กดเข้ามากลับสถานะที่เป็นอยู่ แล้วดูว่าควรจะไปยังสถานะใด และทำงานตามรูทีนใด จากนั้นก็คอยรับการกดคีย์บอร์ดครั้งต่อไป การใช้ KEYBOARD PARSING โดยตรงกับทุกคำสั่งของเครื่อง PC จะมีปัญหาคือจะทำให้ตารางที่ใช้เก็บจังหวะการทำงานมีขนาดโตมาก เนื่องจากมีจำนวนของคำสั่งมาก ซึ่งในการออกแบบโปรแกรมในโหมดนี้จะใช้เทคนิคการจับกลุ่มที่คล้ายกัน และการใช้ตารางซ้อนลงไปอีกครั้ง ซึ่งรายละเอียดจะกล่าวถึงในบทถัดไป



รูปที่ 2.3 แสดงลักษณะการทำงานของโปรแกรมควบคุม

โหมดทำงานโปรแกรมควบคุมการทำงานในโหมดนี้ จะมีจุดประสงค์ต่างจากโหมดโปรแกรมคือโปรแกรมควบคุมการทำงาน ในโหมดทำงานต้องการความเร็วในการทำงานเป็นสิ่งสำคัญ ทั้งนี้เพื่อให้เวลาหนึ่งรอบการทำงาน (1 SCANTIME) มีค่าน้อยที่สุด ส่วนในโหมดโปรแกรมนั้นเราเน้นความสะดวกของผู้ใช้งานเป็นสำคัญ การทำงานของโหมดทำงานนี้ส่วนที่สำคัญคือการทำตามคำสั่งของโปรแกรมแลดเดอร์ที่ผู้ใช้เขียนเก็บไว้ในหน่วยความจำ เทคนิคในการจัดเก็บ และการประมวลผลคำสั่งโปรแกรมแลดเดอร์ เทคนิคในการประมวลผลคำสั่งอาจแบ่งได้เป็น 4 วิธี [8] ดังตารางที่ 2.3

	TABLE LOOK UP	วิธี CALL	วิธี AUTO INCREMENT	วิธี COMPILER
BYTE/STEP	3	5	4	4-15
EXECUTION SPEED (APPROX) /US/STEP	2ØØ	115	55	2Ø
กระบวนการแปลงรหัสและถอดรหัส	ง่าย	ปานกลาง	ปานกลาง	ยาก
การจัด DATA STRUCTURE	ง่าย	ปานกลาง	ปานกลาง	ยาก

หมายเหตุ EXECUTED SPEED เปรียบเทียบที่ CLOCK 33Ø nSEC

ตารางที่ 2.3 แสดงการเปรียบเทียบโปรแกรมควบคุมการแปลคำสั่ง (Ladder Interpreter)

จากตารางจะพบว่าแต่ละวิธีจะมีข้อดีข้อเสียที่แตกต่างกัน วิธี COMPILER จะเป็นวิธีที่ดีที่สุด และให้ความเร็วในการทำงานดีที่สุด แต่จะมีความยุ่งยากในการจัดเก็บข้อมูลและการแปลงและถอดรหัส สำหรับเครื่อง PC ที่ออกแบบนี้จะเลือกใช้วิธี COMPILER เป็นหลักในการทำงานเพื่อให้ได้ความเร็วในการทำงานดีที่สุด อย่างไรก็ตามการใช้วิธี COMPILER นี้จะมีความไม่เหมาะสมกับคำสั่งบางคำสั่งที่ความยาวมาก ๆ เพราะทำให้เสียหน่วยความจำในการเก็บคำสั่งมาก และความเร็วที่เร็วขึ้นก็ไม่มากนัก การวิจัยครั้งนี้จะใช้เทคนิคแบบผสมประสานระหว่างวิธี COMPILER และวิธี CALL โดยมีข้อพิจารณาในการใช้ดังนี้

คำสั่งต่าง ๆ ในเครื่อง PC แบ่งออกได้เป็น 2 กลุ่ม หลัก ๆ คือ

1. คำสั่งเบื้องต้น
2. คำสั่งฟังก์ชั่นพิเศษต่าง ๆ

คำสั่งเบื้องต้น หมายถึงคำสั่งในการอ่านค่าอินพุท คำสั่งในการทำลอจิก และคำสั่งเอาท์พุท ได้แก่ คำสั่ง LD LDNOT AND ANDNOT OR ORNOT ANDLD ORLD OUT OUTNOT คำสั่งเหล่านี้จะมีความยาวประมาณ 4-8 ไบท์ต่อคำสั่ง คำสั่งเหล่านี้ปกติจะมีความถี่ในการใช้ในการเขียนโปรแกรมบ่อยมาก

คำสั่งฟังก์ชั่นพิเศษต่าง ๆ หมายถึง คำสั่งอื่น ๆ ที่นอกเหนือจากคำสั่งเบื้องต้นคำสั่งเหล่านี้ได้แก่คำสั่ง MOV CMP ADD SUB MUL DIV เป็นต้น คำสั่งฟังก์ชั่นต่าง ๆ เหล่านี้จะมีความยาวของตัวโปรแกรมที่ทำงานจริงนั้นยาวมาก บางคำสั่งอาจยาวหลายร้อยไบท์ เช่น คำสั่งการคูณ (MUL) การหาร (DIV)

เนื่องจากคำสั่งของการเขียนโปรแกรมบางคำสั่งมีขนาดยาวมาก ทำให้วิธีการประมวลผลคำสั่งโปรแกรมแลดเดอร์นั้น จะต้องใช้ร่วมกันระหว่างวิธี COMPILER กับวิธี CALL เพื่อให้ใช้งานได้อย่างมีประสิทธิภาพ สำหรับการเปรียบเทียบการใช้วิธี CALL กับวิธี AUTO INCREMENT นั้น เนื่องจากวิธี AUTO INCREMENT ไม่เหมาะกับระบบที่มีการ Interrupt และความเร็วในการทำงานของวิธี AUTO INCREMENT ก็ไม่ได้เร็วกว่าวิธี CALL เท่าใดนัก เนื่องจากคำสั่งที่ต้องทำงานมีความยาวมาก ดังนั้นจึงเลือกใช้วิธีการ CALL ร่วมกับวิธี COMPILER

การพิจารณาว่าคำสั่งใดควรใช้วิธีประมวลผลแบบวิธี COMPLILER หรือวิธี CALL อาจพิจารณาจากความเร็วที่เพิ่มขึ้น เมื่อใช้วิธี COMPILER เทียบกับวิธี CALL การพิจารณานี้อาจรวมถึงจำนวนหน่วยความจำที่ใช้ในการเก็บคำสั่งแบบต่าง ๆ ด้วย

คำสั่ง	จำนวนไบท์แบบ COMPILER	จำนวนไบท์แบบ CALL	ความเร็วที่เพิ่ม	จำนวนไบท์ที่เพิ่ม
OUT	4	5	388%	-25%
DIV	316	1Ø	<5%	+316Ø%

ตารางที่ 2.4 เปรียบเทียบการประมวลวิธี COMPILER กับวิธี CALL

จากตารางจะเห็นบางคำสั่งเมื่อเปลี่ยนจากวิธี CALL เป็นวิธี COMPILER ความเร็วในการทำงานเพิ่มขึ้นไม่มากนัก แต่จะทำให้ใช้พื้นที่หน่วยความจำในการเก็บคำสั่งเพิ่มขึ้นอีกมาก ดังนั้นเพื่อความเหมาะสมในเครื่อง PC ที่ออกแบบจะใช้วิธี COMPILE ร่วมกับวิธี CALL

2.5 แนวความคิดในการจัดเก็บข้อมูล

เมื่อพิจารณาการทำงานของคำสั่งแลดเดอร์ของเครื่อง PC จะเห็นว่ามีการเกี่ยวข้องกับข้อมูล 2 จำพวกคือ ข้อมูลที่เป็นสถานะของอินพุทหรือเอาท์พุท ซึ่งมีค่าสถานะเป็น ON หรือ OFF กับข้อมูลที่เป็นค่าจำนวน ซึ่งเกี่ยวข้องกับคำสั่งด้านการจัดการข้อมูล ดังนั้นในการออกแบบการจัดเก็บข้อมูลของเครื่อง PC นี้ จะจำแนกลักษณะการจัดเก็บข้อมูลได้เป็น 2 จำพวก

การเก็บข้อมูลสำหรับสถานะของอินพุท และเอาท์พุทอาจมีวิธีการจัดเก็บได้หลายวิธี ซึ่งวิธีการต่าง ๆ นั้น จะมีผลต่อการออกแบบฮาร์ดแวร์ของเครื่อง รวมถึงการออกแบบโปรแกรมควบคุมการทำงานของเครื่อง PC ด้วย สำหรับการวิจัยนี้จะออกแบบในแนวทางที่ให้เครื่อง PC ทำงานได้สะดวกรวดเร็ว และไม่ยุ่งยากซับซ้อนเกินไป เนื่องจากสถานะของอินพุทและเอาท์พุทมีเพียงสองสถานะคือ สถานะ ON และสถานะ OFF ดังนั้นในการเก็บข้อมูลนั้นเองอาจใช้หน่วยความจำเพียงหนึ่งบิทในการเก็บสถานะอินพุท/เอาท์พุทหนึ่งจุด หรืออาจจะใช้หน่วยความจำหนึ่งไบท์ ในการเก็บสถานะอินพุท เอาท์พุทหนึ่งจุดก็ได้ ซึ่งก็มีข้อที่น่าพิจารณาหลายอย่างดังนี้

7	6	5	4	3	2	1	Ø	BIT
Ø/1	Ø/1	Ø/1	Ø/1	Ø/1	Ø/1	Ø/1	Ø/1	

ก) ลักษณะการเก็บข้อมูลหนึ่งบิทต่อ หนึ่งอินพุท/เอาท์พุท

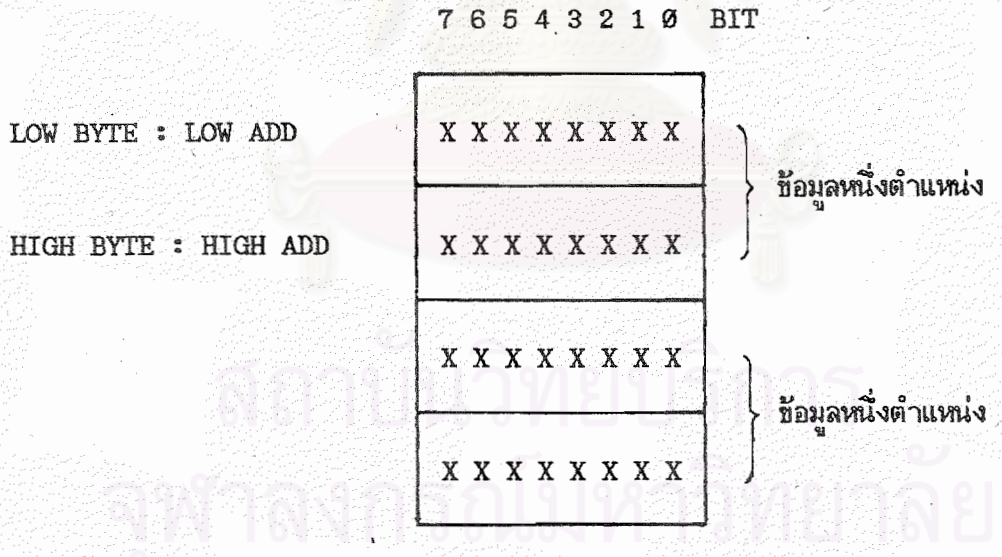
7	6	5	4	3	2	1	Ø	BIT
Ø/1	X	X	X	X	X	X	X	BYTE
Ø/1	X	X	X	X	X	X	X	BYTE
Ø/1	X	X	X	X	X	X	X	BYTE

ข) ลักษณะการเก็บข้อมูลหนึ่งไบท์ ต่อ หนึ่งอินพุท/เอาท์พุท

รูปที่ 2.4 แสดงการเก็บข้อมูลในลักษณะบิท หรือไบท์

การเก็บข้อมูลในลักษณะบิท โดยแทนหนึ่งบิทต่อหนึ่งอินพุท/เอาท์พุทนั้นมีข้อดีคือ ทำให้ประหยัดหน่วยความจำที่ใช้ในการเก็บข้อมูล แต่มีข้อเสียคือการอ่านและเขียนข้อมูลที่ตำแหน่งต่าง ๆ จะยุ่งยากและเสียเวลาในการดีโค๊ด (Decode) และเอนโค๊ด (Encode) ข้อมูล ซึ่งจะทำเวลาที่ทำงานใน 1รอบมีค่ามากขึ้น สำหรับการจัดเก็บข้อมูลในลักษณะหนึ่งไบท์แทนข้อมูลอินพุท/เอาท์พุทหนึ่งจุด จะทำให้การประมวลผลของคำสั่งแลดเดอร์รวดเร็วยิ่งขึ้น และจะมีความสะดวกในการอ้างอิงตำแหน่งอินพุท/เอาท์พุทต่าง ๆ อีกด้วย แต่ก็ใช้หน่วยความจำในการเก็บข้อมูลมากขึ้น แต่หน่วยความจำในปัจจุบันมีราคาไม่สูง ในการวิจัยนี้จึงใช้วิธีเก็บข้อมูลในลักษณะไบท์แทนสถานะอินพุทและเอาท์พุท การใช้ข้อมูลในการเก็บเพียงหนึ่งบิทนี้ ทำให้ฮาร์ดแวร์ของเครื่อง PC ในส่วนของการรับ-ส่งข้อมูลระหว่างโมดูลอินพุท/เอาท์พุทกับโมดูลหน่วยประมวลผล สามารถใช้ดาต้าบัสเพียงหนึ่งเส้นเท่านั้น ทำให้การออกแบบระบบสะดวกขึ้น

การเก็บข้อมูลที่เป็นค่าจำนวน สำหรับใช้ในคำสั่งด้านการจัดการข้อมูลนั้น กำหนดให้มีขนาด 16 บิท ซึ่งจะต้องใช้หน่วยความจำ 2 ไบท์ ในการเก็บข้อมูลหนึ่งตำแหน่งโดยข้อมูลที่เก็บอาจเป็นเลขไบนารี่หรือเลข BCD ก็ได้ ขึ้นกับคำสั่งในการใช้งาน การอ้างอิงตำแหน่งของข้อมูลจะชี้โดยแอดเดรสไบท์ต่ำของข้อมูลตำแหน่งนั้นๆ ดังแสดงในรูปที่ 2.5



รูปที่ 2.5 แสดงการจัดเก็บข้อมูลที่ใช้ในคำสั่งด้านการจัดการข้อมูล

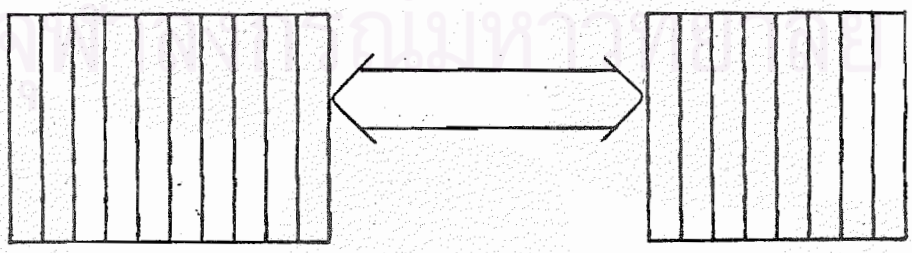
บทที่ 3

ฮาร์ดแวร์ของเครื่องควบคุมชนิดโปรแกรมได้

บทนี้จะกล่าวถึงฮาร์ดแวร์ของเครื่องควบคุมชนิดโปรแกรมได้ ที่ออกแบบสร้างในการวิจัยครั้งนี้ โดยจะเริ่มจากลักษณะโครงสร้างทางฮาร์ดแวร์และการทำงานของระบบ ต่อจากนั้นจะกล่าวถึงฮาร์ดแวร์ของโมดูลต่าง ๆ ที่ออกแบบสร้าง รวมถึงฮาร์ดแวร์ของตัวป้อนโปรแกรม (Hand-held programming console)

3.1 ลักษณะโครงสร้างของระบบ

ระบบประกอบด้วยสองส่วนใหญ่ๆ คือ แผงช่องเสียบ(Panel board) กับโมดูลต่างๆ โดยโมดูลจะเสียบต่อบนแผงช่องเสียบ โมดูลต่าง ๆ ที่ออกแบบได้แก่ โมดูลหน่วยประมวลผล โมดูลแสดงผลและคีย์บอร์ด โมดูลอินพุทและเอาท์พุทแบบต่าง ๆ ระบบที่ออกแบบจะมีช่องเสียบสำหรับโมดูลประมวลผล โมดูลแสดงผลและคีย์บอร์ดโดยเฉพาะ ส่วนที่เหลือนอกนั้นเป็นช่องเสียบสำหรับโมดูลอินพุท และโมดูลเอาท์พุทชนิดต่าง ๆ แผงช่องเสียบของระบบแบ่งออกเป็น 2 ส่วน คือ Basic unit และ Expansion unit โดยที่แต่ละยูนิตจะสามารถต่อโมดูลอินพุท/เอาท์พุทได้ถึง 8 โมดูล *

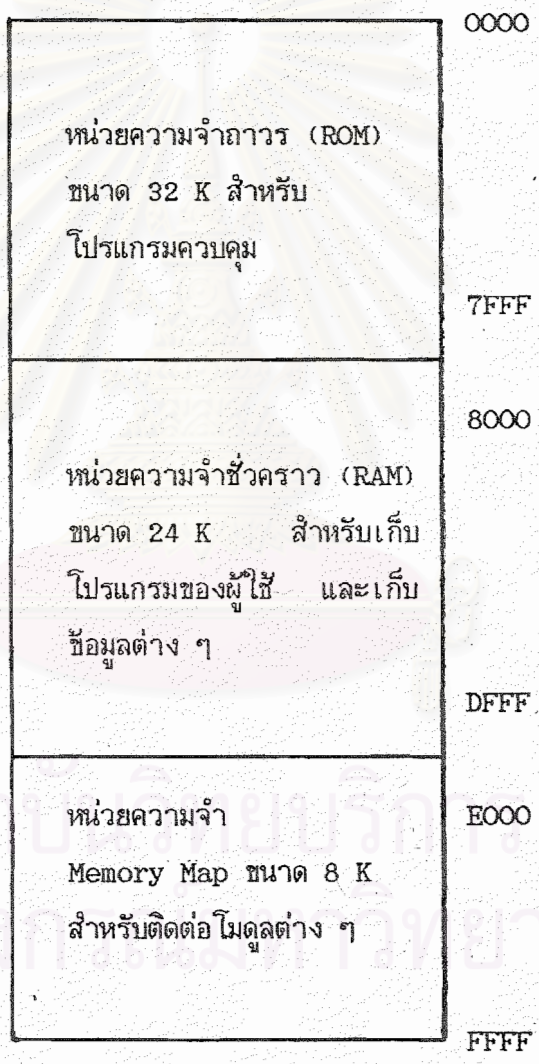


Basic unit Expansion unit

รูปที่ 3.1 แสดงลักษณะช่องเสียบ (panel board) ของเครื่อง PC

* สำหรับโมดูลหน่วยประมวลผลและโมดูลแสดงผลและคีย์บอร์ดจะต่ออยู่ที่ Basic unit เสมอ

เครื่อง PC ที่ออกแบบนี้จะใช้ซีพียูเป็นไมโครโปรเซสเซอร์ขนาด 8 บิท เบอร์ Z80A เป็นตัวควบคุมการทำงานทั้งหมด โดยที่วงจรในส่วนควบคุมทั้งหมดจะอยู่ในโมดูลประมวลผล ส่วนโมดูลอื่น ๆ จะติดต่อกับโมดูลประมวลผลและถูกควบคุมโดยโมดูลประมวลผล การติดต่อระหว่างโมดูลประมวลผลกับโมดูลอื่นๆ จะติดต่อในลักษณะของ I/O Map และ Memory Map สำหรับการจัดแบ่งหน่วยความจำของระบบเป็นดังรูปที่ 3.2



รูปที่ 3.2 แสดงการจัดแบ่งหน่วยความจำของเครื่อง PC

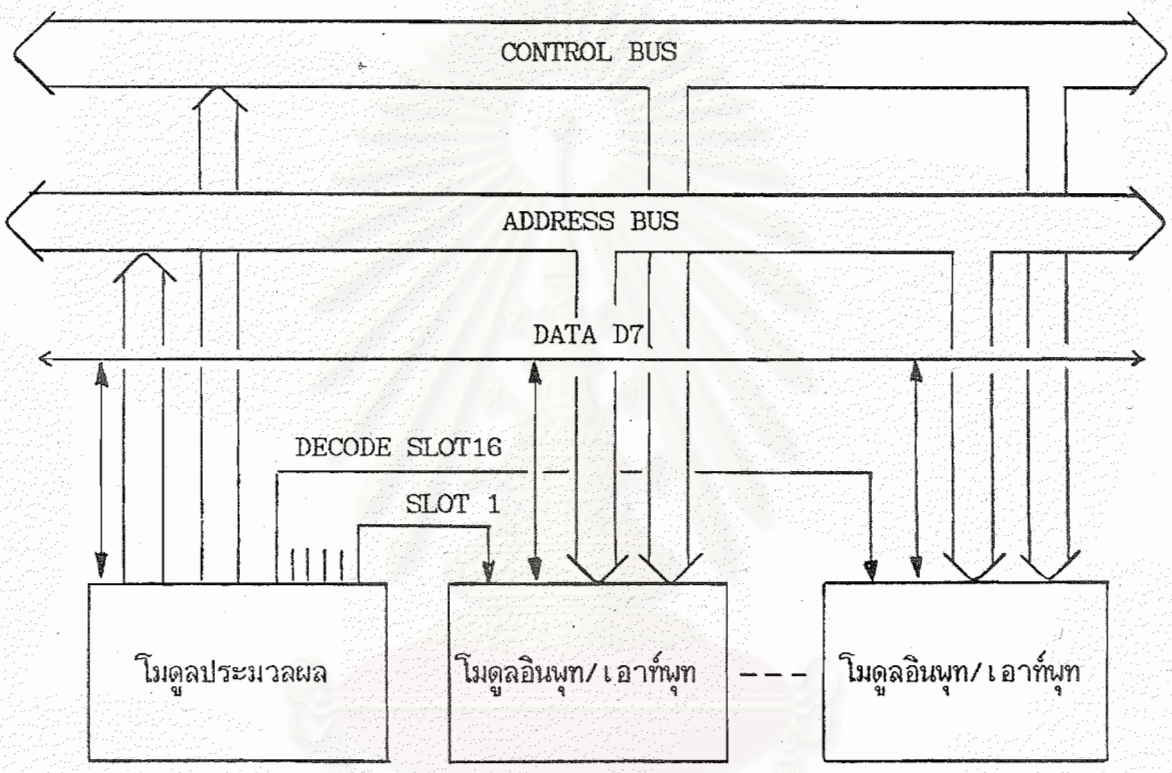
การติดต่อระหว่างโมดูลประมวลผลและโมดูลอินพุท/เอาท์พุท จะใช้การติดต่อผ่านหน่วยความจำ Memory Map ขนาด 8K นี้ในเครื่อง PC ที่ออกแบบนี้จะให้สามารถต่อโมดูลอินพุท/เอาท์พุทได้สูงสุด 16 โมดูล โดยมีการจัดแบ่งMemory Mapสำหรับการติดต่อของโมดูลต่างๆ ดังนี้

E000

Slot 1 : E000 - E1FF
Slot 2 : E200 - E3FF
Slot 3 : E400 - E5FF
Slot 4 : E600 - E7FF
Slot 5 : E800 - E9FF
Slot 6 : EA00 - EBFF
Slot 7 : EC00 - EDFF
Slot 8 : EE00 - EF00
Slot 9 : F000 - F1FF
Slot 10 : F200 - F3FF
Slot 11 : F400 - F5FF
Slot 12 : F600 - F7FF
Slot 13 : F800 - F9FF
Slot 14 : FA00 - FBFF
Slot 15 : FC00 - FDFF
Slot 16 : FE00 - FFFF

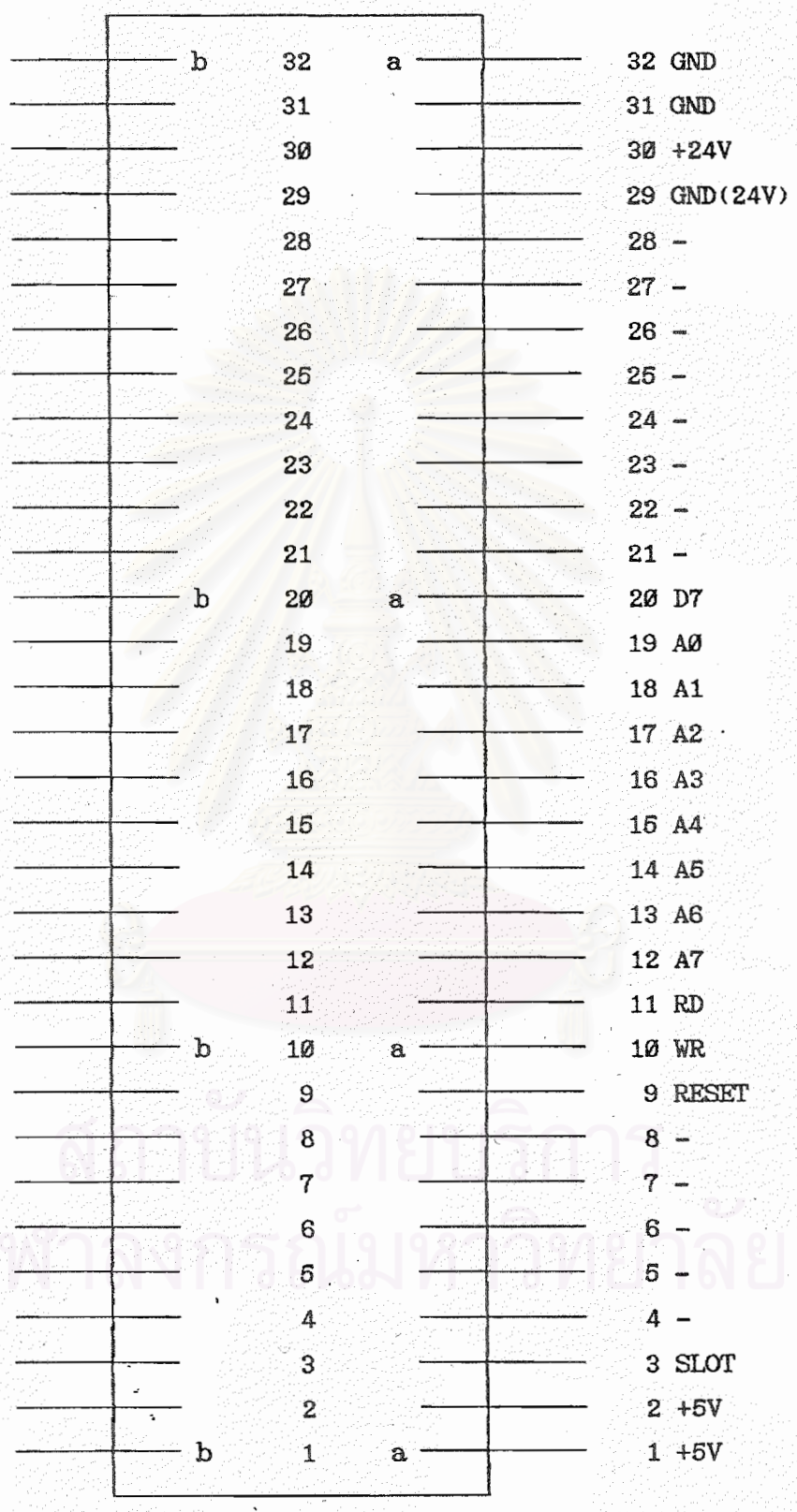
FFFF

การติดต่อของโมดูลอินพุท/เอาท์พุทต่างๆ กับโมดูลประมวลผลจะติดต่อผ่านหน่วยความจำ Memory Map ของ Slot นั้นๆ ที่โมดูลเสียบอยู่ สำหรับสัญญาณควบคุมการเลือกของ Memory Map แต่ละ Slot นั้น จะทำการดีโค้ดและส่งมาจากโมดูลประมวลผลไปยังสล๊อตต่าง ๆ สำหรับลักษณะการติดต่อสัญญาณของโมดูลต่าง ๆ จะเป็นดังนี้



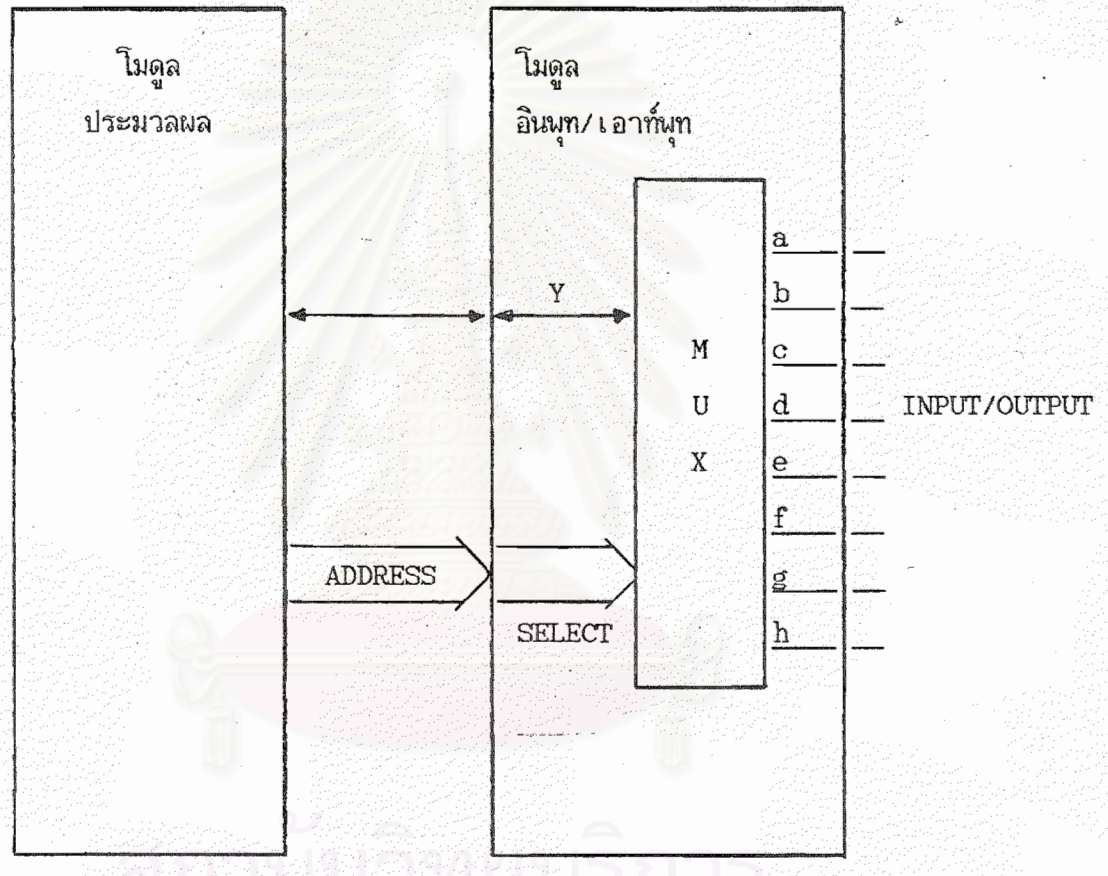
รูปที่ 3.3 แสดงสัญญาณที่ใช้ในการติดต่อของโมดูลต่าง ๆ

สัญญาณที่ติดต่อระหว่างโมดูลประมวลผล กับโมดูลอินพุท/เอาท์พุทต่าง ๆ ประกอบด้วยสัญญาณแอดเดรสบัส ซึ่งสามารถอ้างอิงตำแหน่งความจำต่าง ๆ ของ Memory Map แต่ละสล๊อตได้ สัญญาณ Control bus เป็นสัญญาณควบคุมการทำงานจากซีพียูมาจากโมดูลประมวลผล สัญญาณดาต้าบัส D7 สำหรับรับส่งข้อมูลระหว่างโมดูลประมวลผลและโมดูลอินพุท/เอาท์พุทต่าง ๆ และสัญญาณดีโค้ด Memory Map ของแต่ละสล๊อตสัญญาณต่าง ๆ ในการติดต่อเหล่านี้จะต่อขนานกับทุก ๆ สล๊อต โดยใช้คอนเน็กเตอร์ขนาด 64 ขา ในการต่อกับโมดูลต่าง ๆ โดยมีตำแหน่งสัญญาณของขาต่าง ๆ ดังนี้



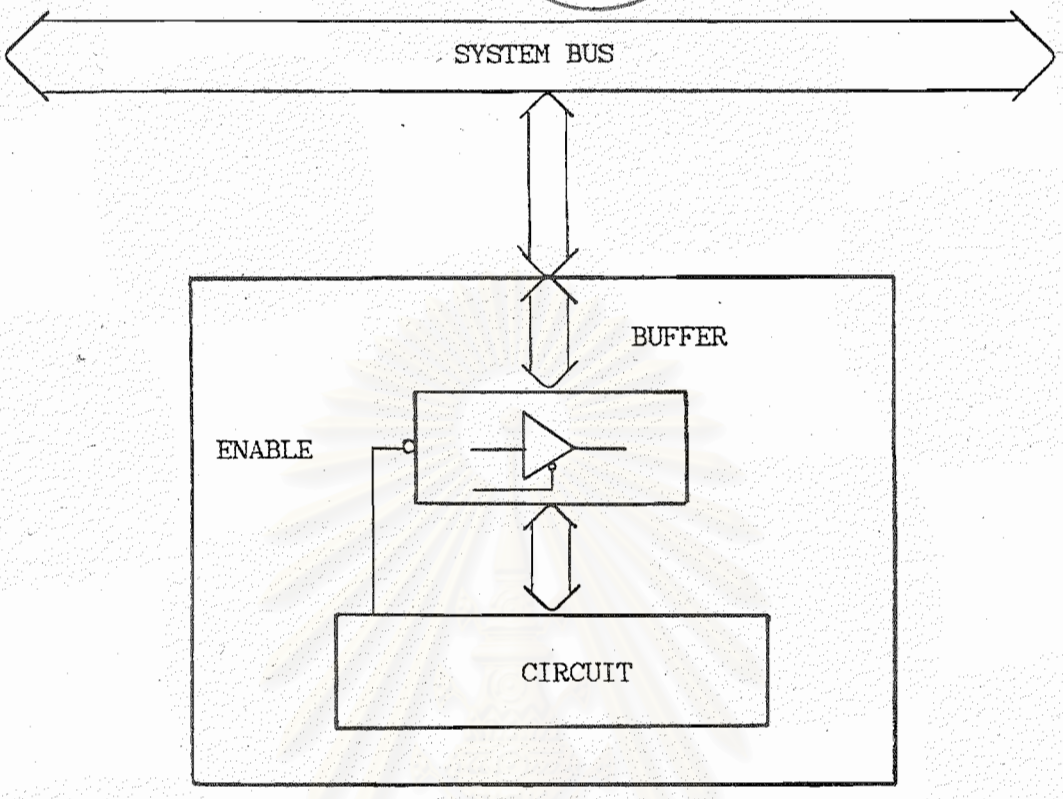
รูปที่ 3.4 แสดงขั้วต่อของสัญญาณต่าง ๆ

การส่งข้อมูลระหว่างโมดูลประมวลผลและโมดูลอินพุท/เอาท์พุทนั้น จะใช้ดาต้าบัสในการส่งข้อมูลเพียง 1 เส้น เพราะการจัดเก็บข้อมูลของอินพุทและเอาท์พุท จะเก็บในลักษณะ 1 ไบท์ต่ออินพุทหรือเอาท์พุท1จุด ซึ่งจะใช้ข้อมูลในการเก็บ 1 บิทเท่านั้น สำหรับการรับส่งข้อมูลหลายๆ ตัว เราก็รับส่งในลักษณะการมัลติเพ็กซ์ โดยจะมีวงจรนี้อยู่ในโมดูลอินพุท/เอาท์พุทต่างๆ ซึ่งมีลักษณะในการรับส่งข้อมูลดังนี้



รูปที่ 3.5 แสดงการรับส่งข้อมูลระหว่างโมดูลประมวลผลกับโมดูลอินพุท/เอาท์พุท

ในการทำงานของเครื่อง PC นั้นการติดต่อระหว่างโมดูลประมวลผล กับโมดูลอินพุท/เอาท์พุทต่าง ๆ จะไม่ได้ติดต่อกันตลอดเวลา แต่จะติดต่อกันเป็นช่วง ๆ คือทุก ๆ 1 รอบการทำงาน (SCAN TIME) ในการทำงานตามโปรแกรมที่ผู้ใช้เขียนดังนั้นเพื่อให้เครื่องมีเสถียรภาพดีขึ้น จึงไม่ควรเชื่อมต่อโมดูลต่างๆ เข้ากับบัสของระบบตลอดเวลา ควรจะเชื่อมเฉพาะเมื่อต้องการติดต่อรับส่งข้อมูลเท่านั้น การทำลักษณะนี้จะช่วยปัญหาลดการรบกวนจากสัญญาณภายนอกมายังระบบซึ่งช่วยทำให้เสถียรภาพของเครื่องดีขึ้น



รูปที่ 3.6 แสดงการเชื่อมต่อบัสของโมดูลกับระบบ

โมดูลอินพุทและเอาท์พุทชนิดต่าง ๆ สามารถต่อเข้ากับเครื่อง PC นี้ ที่ตำแหน่งสล๊อตใด ๆ ก็ได้ โดยผู้ใช้เป็นผู้เลือกตามความสะดวกในการใช้งาน ระบบที่ออกแบบนี้จะต้องสามารถรับรู้ว่ามีโมดูลอะไรต่ออยู่ที่สล๊อตอะไรบ้าง ซึ่งในเครื่อง PC นี้ใช้การสร้างรหัสไว้ที่โมดูลอินพุทและเอาท์พุทต่าง ๆ โดยโมดูลแต่ละชนิดจะมีรหัสที่แตกต่างกันเมื่อเริ่มเปิดเครื่อง (POWER ON) จะต้องมีการตรวจสอบว่ามีโมดูลอะไรต่ออยู่ในเครื่องที่ตำแหน่งใดบ้าง และมีการสร้างตารางข้อมูลเพื่อใช้ในการติดต่อกับโมดูลเหล่านี้เก็บไว้

3.2 ฮาร์ดแวร์ของส่วนต่าง ๆ

ในหัวข้อที่ผ่านมาได้กล่าวถึงลักษณะโครงสร้างของระบบมาแล้ว ซึ่งคงทำให้เข้าใจหลักการทำงานของเครื่องในลักษณะรวมมาแล้ว สำหรับในหัวข้อนี้จะได้กล่าวถึงลักษณะและการทำงานของแต่ละโมดูลที่ออกแบบสร้างไว้ ซึ่งได้แก่

- โมดูลประมวลผล
- โมดูลแสดงผลคีย์บอร์ด และสื่อสาร

- โมดูลอินพุท
- โมดูลเอาท์พุท
- โมดูลอนาลอกอินพุท
- โมดูลกำหนดค่าตัวเลข
- ตัวป้อนโปรแกรม (Hand-held programming console)

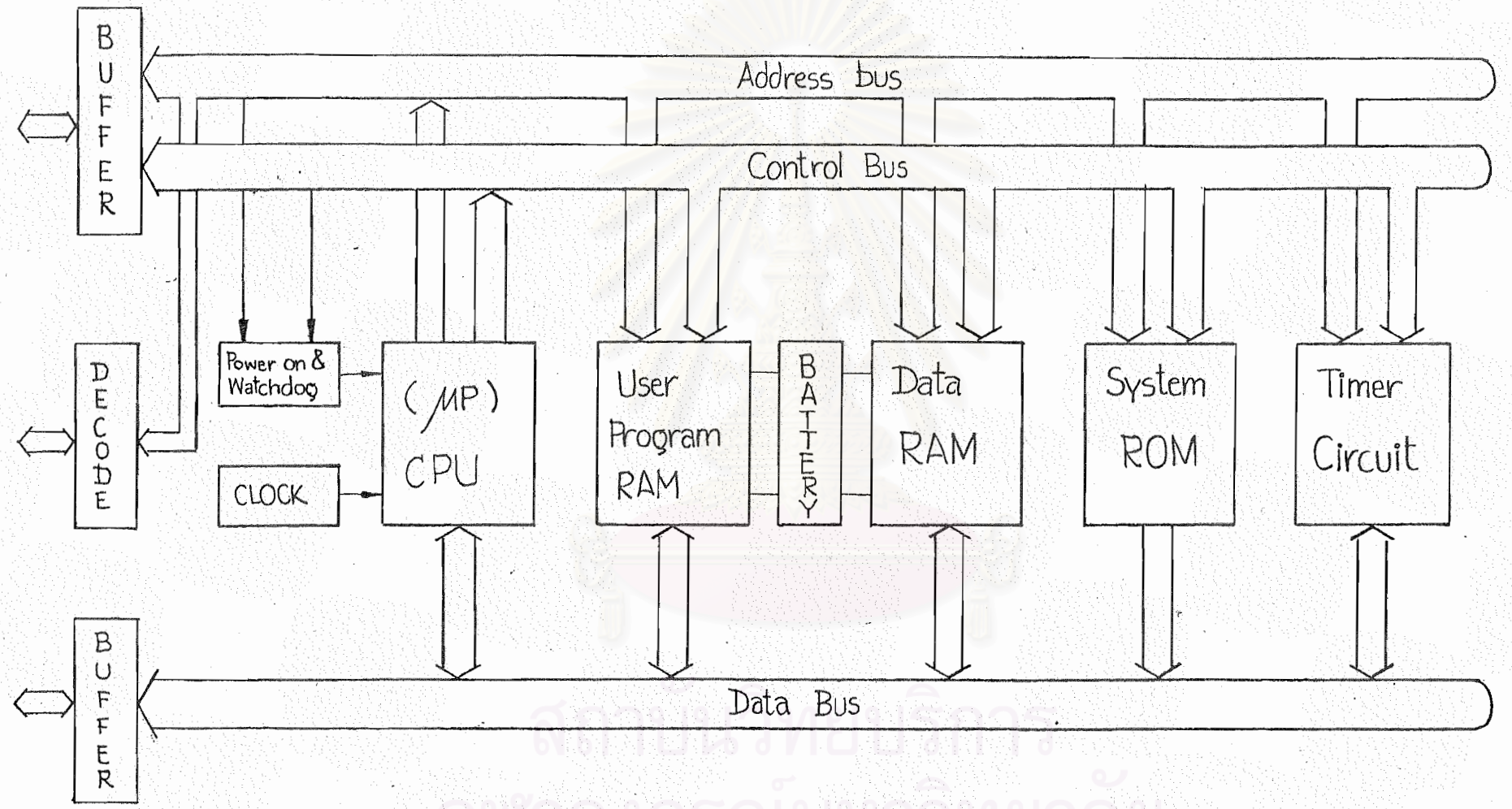
3.2.1 โมดูลประมวลผล

โมดูลนี้จะมีลักษณะเป็นไมโครคอมพิวเตอร์เครื่องหนึ่ง ซึ่งมีวงจรพิเศษบางอย่างเพิ่มขึ้น โมดูลนี้ออกแบบให้สามารถใช้งานเอนกประสงค์ด้วยคือ สามารถนำโมดูลนี้ไปใช้พัฒนากับงานอื่นได้ การทำงานของโมดูลประมวลผลไม่จำเป็นต้องอาศัยโมดูลอื่น ตัวมันเองสามารถทำงานได้โดยอิสระอยู่แล้ว ส่วนประกอบที่สำคัญของโมดูลนี้สามารถแสดงได้ด้วย รูปที่ 3.7 และรูปที่ ก.1 - ก.2 (ภาคผนวก ก)

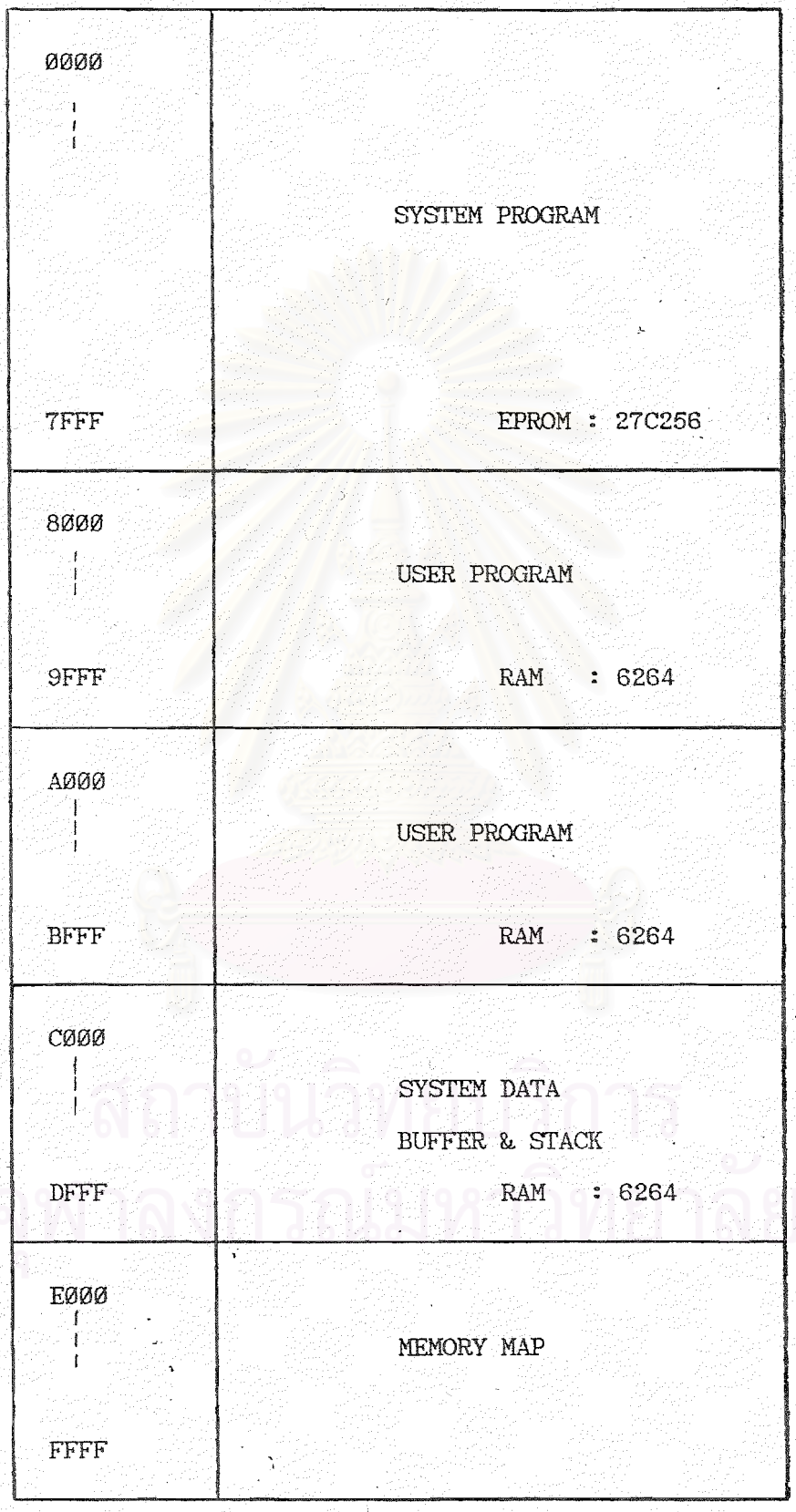
ซีพียูใช้ขนาด 8 บิท เบอร์ Z80A ทำงานที่ความเร็วนาฬิกา 4 MHz ระบบบัสของซีพียูจะต่อตรงกับอุปกรณ์ต่างๆ ในโมดูลเลย แต่จะต่อผ่านบัฟเฟอร์สำหรับบัสที่ต่อออกไปภายนอกโมดูล

หน่วยความจำ ซึ่งแบ่งออกได้ 3 ส่วนคือ หน่วยความจำที่ใช้เก็บโปรแกรมควบคุมการทำงานต่าง ๆ หน่วยความจำส่วนนี้ใช้ EPROM เบอร์ 27C256 เป็นตัวเก็บโปรแกรม มีแอดเดรสในการติดต่อระหว่าง 0000H-7FFFH สามารถเก็บโปรแกรมได้ 32 กิโลไบท์ หน่วยความจำส่วนที่ 2 คือ หน่วยความจำที่เก็บโปรแกรมของผู้ใช้ (User program) ซึ่งไอซี RAM เบอร์ 6264 จำนวน 2 ตัว มีตำแหน่งแอดเดรสในการทำงานระหว่าง 8000H ถึง BFFFH สามารถเก็บข้อมูลได้ทั้งหมด 16 กิโลไบท์ หน่วยความจำส่วนนี้จำเป็นต้องมีแหล่งจ่ายไฟสำรอง (Battery backup) เพื่อป้องกันไม่ให้โปรแกรมของผู้ใช้ที่เก็บไว้ภายในสูญหายเมื่อปิดเครื่อง (Power off) หน่วยความจำส่วนที่ 3 คือ หน่วยความจำที่เก็บข้อมูลต่าง ๆ ของอินพุท และเอาท์พุทในการทำงานของเครื่องรวมทั้งใช้เป็นที่เก็บสถานะต่าง ๆ ในการทำงานของโปรแกรมควบคุมเครื่องและใช้บางส่วนเป็นสแตกค์ (Stack) ในการทำงานของโปรแกรม หน่วยความจำส่วนที่ 3 นี้ก็จะมีแหล่งจ่ายไฟสำรอง (Battery backup) ด้วยเพราะข้อมูลอินพุท/เอาท์พุทของเครื่อง PC บางชนิด (Retentive memory) จำเป็นต้องเก็บข้อมูลเดิมไว้เมื่อปิดเครื่อง PC

วงจรฐานเวลา (Time base) เป็นวงจรสำหรับสร้างสัญญาณฐานเวลาให้แก่ระบบการทำงานจะใช้ไอซี Z80-CTC เป็นตัวทำงาน ซึ่งจะมีทั้งหมด 4 แชนแนล โดยแชลแนล 0 จะใช้เป็นสัญญาณนาฬิกาสำหรับใช้ในการติดต่อสื่อสาร RS-232 ของโมดูลแสดงผล และคีย์บอร์ด แชลแนลที่ 1 จะเป็นสัญญาณนาฬิกาสำหรับวงจรการอ่านคีย์บอร์ด ซึ่งจะมีความถี่ประมาณ 80KHz



รูปที่ 3.7 แสดงโครงสร้างของโมดูลประมวลผล



รูปที่ 3.8 แสดงการจัดเก็บหน่วยความจำของระบบ

สำหรับแชลแนลที่ 2 และแชลแนลที่ 3 จะใช้งานร่วมกันทำหน้าที่สร้างสัญญาณฐานเวลาสำหรับการทำงานของคำสั่ง TIM และ TIMH ของเครื่อง PC โดยจะสร้างฐานเวลา ๐.๐๐5 วินาทีให้แก่ระบบ

วงจรวอท์ชดอกไทเมอร์ ทำหน้าที่คอยตรวจเช็คความผิดพลาดของการทำงานของซีพียู โดยมีหลักการทำงานคือ ถ้าซีพียูทำงานตามปกติจะมีสัญญาณมารีเซ็ทวงจรวอท์ดอกไทเมอร์ ทำให้ไม่มีการรีเซ็ทซีพียู แต่ถ้าซีพียูทำงานผิดพลาด คือไม่ทำงานตามโปรแกรมที่วางไว้ก็จะไม่มีสัญญาณมารีเซ็ทวงจรวอทช์ดอกไทเมอร์ ดังนั้นเมื่อถึงเวลาที่ตั้งไว้วงจรวอท์ชดอกไทเมอร์จะส่งสัญญาณไปรีเซ็ทซีพียู ทำให้ซีพียูกลับมาเริ่มทำงานที่จุดเริ่มต้นตามโปรแกรมที่ต้องการใหม่

วงจรจ่ายไฟสำรอง เพื่อรักษาโปรแกรมและข้อมูลของเครื่องที่เก็บอยู่ในหน่วยความจำ RAM ขณะที่ปิดเครื่อง โดยจะใช้แบบตอรี่ Lithium จ่ายไฟให้กับหน่วยความจำ RAM 6264 โดยการใช้ไดโอดเป็นสวิทช์ควบคุมการทำงาน ขณะที่เปิดเครื่อง (Power on) หน่วยความจำเหล่านี้จะรับไฟจากแหล่งจ่ายไฟปกติ แต่เมื่อปิดเครื่องจะใช้ไฟจากแบตตอรี่ การตรวจสอบว่าแบตตอรี่หมดหรือยังนั้น ในเครื่องจะมีวงจรเปรียบเทียบแรงดัน โดยใช้ IC เบอร์ LM 393 เป็นตัวเปรียบเทียบเอาท์พุทของวงจรเปรียบเทียบจะแสดงโดย LED ที่โมดูลแสดงผล และคีย์บอร์ดว่าแบตตอรี่หมดหรือไม่ และเอาท์พุทนี้จะถูกอ่านผ่านพอร์ทโดยโปรแกรมควบคุมระบบด้วยเพื่อนำไปใช้เป็นข้อมูลในการแสดงสถานะของแบตตอรี่

3.2.2 โมดูลแสดงผลคีย์บอร์ด และสื่อสาร

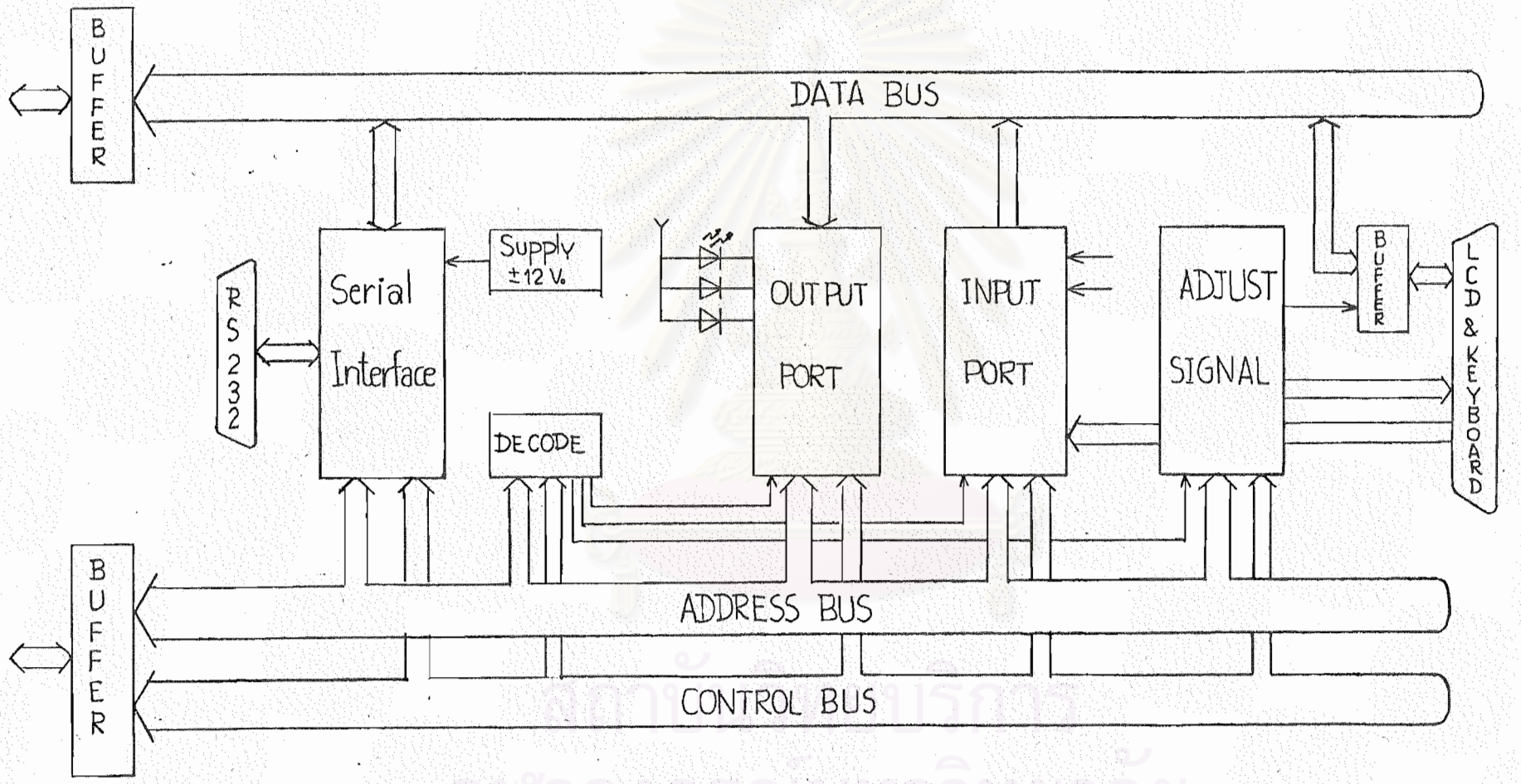
โมดูลนี้ออกแบบมาเพื่อต่อกับตัวป้อนโปรแกรมชนิดมือถือ และพอร์ทสื่อสาร โมดูลนี้มีส่วนประกอบที่สำคัญคือ

วงจรสื่อสารชนิดอนุกรมแบบ RS-232 ซึ่งใช้ไอซีเบอร์ 8251 เป็นตัวควบคุมการทำงานของวงจรนี้สัญญาณนาฬิกาสำหรับวงจรนี้จะส่งมาจากวงจรสร้างฐานเวลา ซึ่งอยู่ในโมดูลประมวลผล

พอร์ทเอาท์พุท ใช้สำหรับขับ LED เพื่อแสดงผลของสถานะต่าง ๆ ในการทำงานของเครื่อง เช่น สถานะทำงานผิดพลาดชนิด ALARM หรือ ERROR โหมดการทำงาน PROGRAM หรือ RUN และสถานะ Low Battary ของแบตตอรี่ เอาท์พุทพอร์ทนี้จะใช้ไอซีบัฟเฟอร์ชนิดที่สามารถแลทช์ข้อมูลได้

พอร์ทอินพุท ใช้อ่านสถานะต่าง ๆ คือ อ่านสวิทช์ในการเลือกโหมดการทำงานว่าเป็นชนิด Program, Monitor หรือ Run ใช้อ่านสถานะการต่อตัวป้อนโปรแกรมชนิดมือถือ และใช้อ่านสถานะของแบตตอรี่

วงจรเชื่อมต่อกับตัวป้อนโปรแกรมแบบมือถือ วงจรนี้จะทำหน้าที่จัดสัญญาณควบคุมต่าง ๆ



รูปที่ 3.9 แสดงโครงสร้างของโมดูลแสดงผล คีย์บอร์ด และสื่อสาร

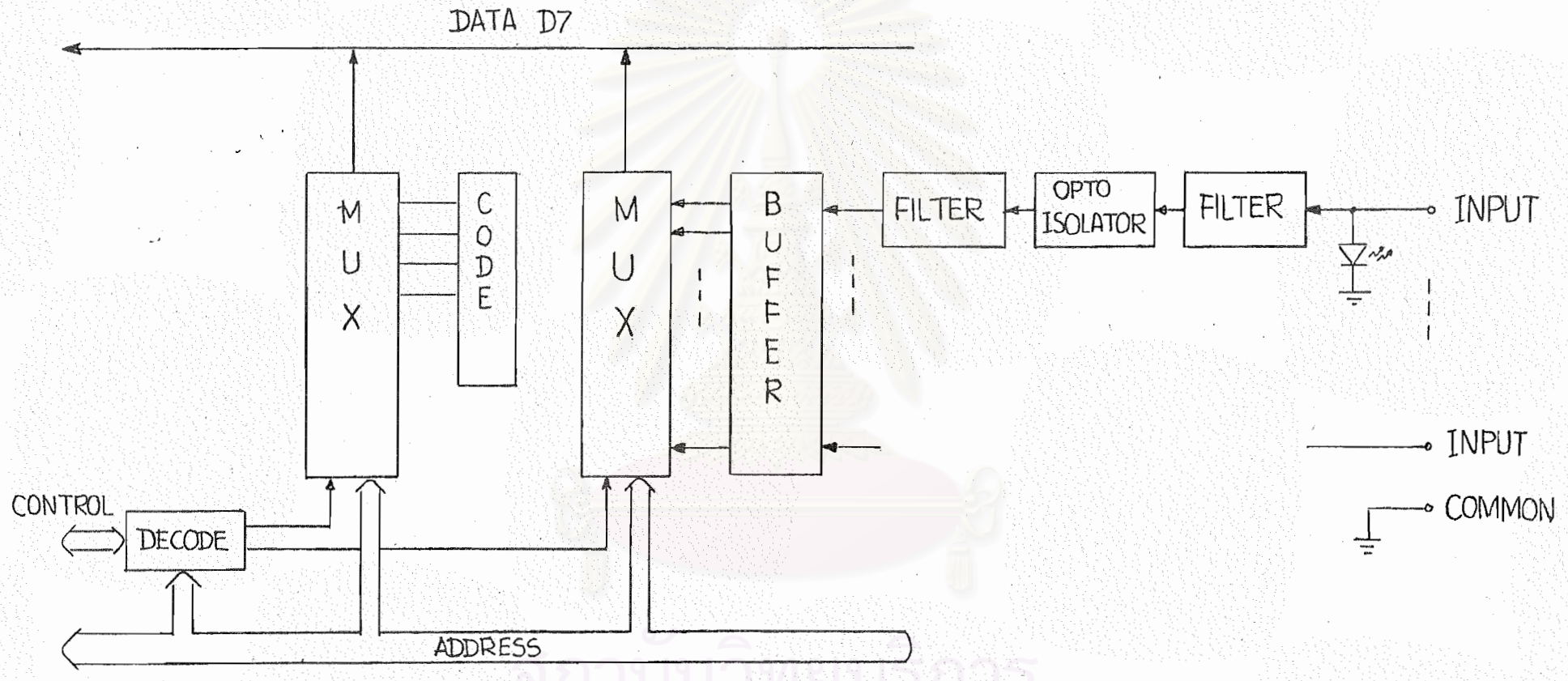
ให้เหมาะกับการต่อกับตัวแสดงผล LCD และเป็นบัฟเฟอร์ระหว่างโมดูลนี้กับตัวป้อนโปรแกรมชนิดมือถือ

3.2.3 โมดูลอินพุท

ทำหน้าที่รับสัญญาณอินพุทจากอุปกรณ์ภายนอก เช่น สวิทช์ ปุ่มกด และเซนเซอร์ (Sensor) ต่าง ๆ ลักษณะของอุปกรณ์ที่มาต่อจะต้องมีเอาท์พุทในลักษณะการเปิด - ปิด โดยต่อเข้าระหว่างขาของอินพุท และขาร่วม (COMMON) ถ้าอุปกรณ์ที่มาให้เอาท์พุทในลักษณะปิดวงจรก็จะมีกระแสไฟครบวงจรในลักษณะกระแสตรงไหลเข้า (DC Sink Type) ซึ่งกระแสนี้มาจากแหล่งจ่ายไฟขนาด 24 V ของระบบ ซึ่งต่อมาให้ที่โมดูลอินพุทแล้ว

หลักการทำงานของโมดูลนี้คือ สัญญาณที่เข้าที่ขั้วอินพุทของโมดูล จะผ่านวงจรฟิลเตอร์แล้วคับปลิ้งผ่านตัวแยกสัญญาณ (Opto Isolator) แล้วผ่านวงจรฟิลเตอร์ความถี่ต่ำไปยังบัฟเฟอร์แบบชมิทก์ สัญญาณที่ผ่านบัฟเฟอร์นี้จะไปยังไอซีเลือกสัญญาณ (Data Selector) เบอร์ 74 LS 251 ซึ่งทำหน้าที่คล้ายการมัลติเพ็กซ์ ไอซีเลือกสัญญาณนี้จะเลือกอ่านสัญญาณอินพุทครั้งละ 1 อินพุท ส่งไปยังดาต้าบัส D7 สัญญาณเลือกอินพุทนี้ จะใช้สัญญาณแอดเดรสที่ส่งมาจากโมดูลประมวลผล

วงจรอีกส่วนหนึ่งของโมดูลนี้คือ วงจรอ่านรหัสของโมดูล ซึ่งใช้ลักษณะการอ่านเหมือนการอ่านอินพุท รหัสของโมดูลนี้ใช้รหัส "ØØØØ"



รูปที่ 3.10 แสดงโครงสร้างของโมดูลอินพุท

3.2.4 โมดูลเอาท์พุท

ทำหน้าที่ส่งสัญญาณเอาท์พุทไปควบคุมอุปกรณ์ภายนอก โดยเอาท์พุทของโมดูลนี้จะเป็นหน้าสัมผัสของรีเลย์ สามารถต่อเอาท์พุทได้ขนาด 2A/220VAC

การทำงานของโมดูลนี้จะรับสัญญาณข้อมูล D7 และสัญญาณแอดเดรสจากโมดูลประมวลผล โดยไอซีเบอร์ 74LS259 ซึ่งทำหน้าที่ในการเลือกส่งข้อมูลและคงค่าข้อมูลไว้ (Latch) สัญญาณที่ได้จะผ่านวงจรดับปลิ้งสัญญาณ โดยใช้ Opto Isotator แล้วขยายสัญญาณส่งไปขับรีเลย์เอาท์พุทอีกครั้งเอาท์พุทของโมดูลนี้จะเป็นหน้าที่สัมผัสของรีเลย์นี้ ซึ่งจะมีวงจรกรองความถี่ต่ออยู่ด้วย เพื่อช่วยลดสัญญาณรบกวน

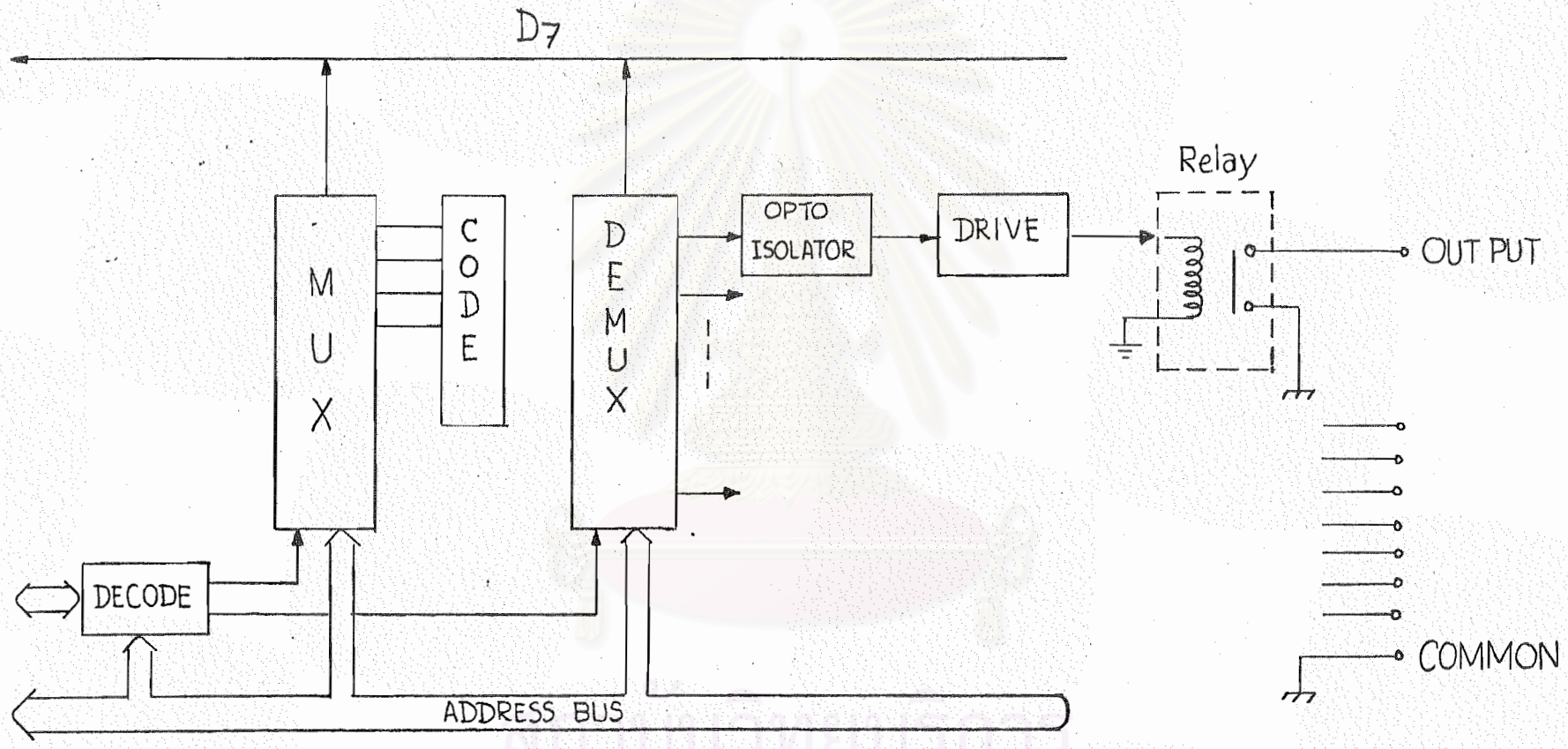
โมดูลเอาท์พุทนี้จะมีวงจรอ่านรหัสของโมดูล ซึ่งมีลักษณะการทำงานเหมือนวงจรอ่านรหัสของโมดูลอินพุท สำหรับรหัสของโมดูลเอาท์พุทนี้คือ "ØØØ1"

3.2.5 โมดูลอนาลอกอินพุท

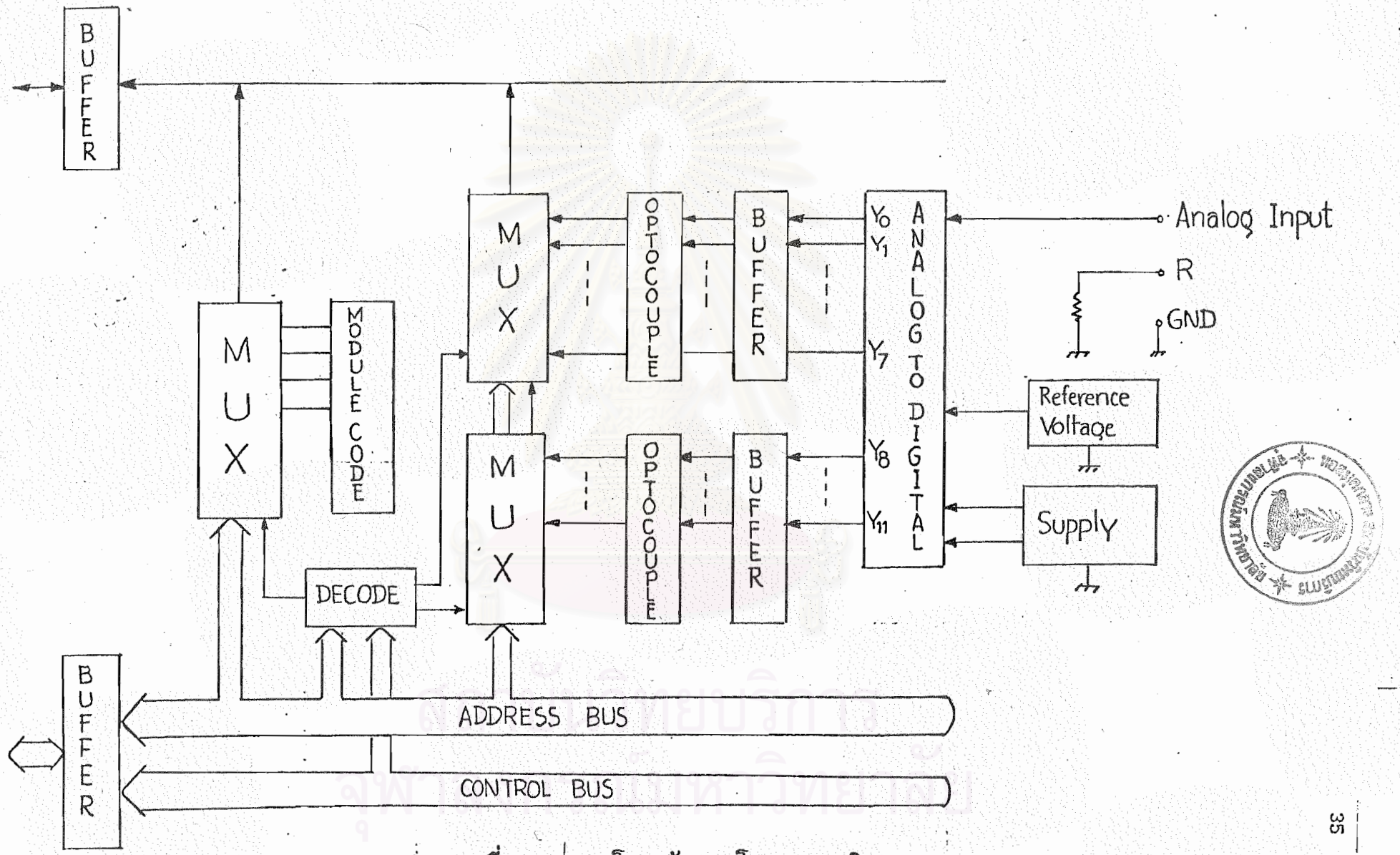
โมดูลนี้จะรับสัญญาณอนาลอกขนาด 1-5 Vdc หรือ 4-2Ø mA แล้วแปลงให้เป็นสัญญาณดิจิตอลขนาด 12 บิท ส่งไปยังโมดูลประมวลผล

การทำงานของโมดูลนี้จะใช้ไอซีเบอร์ 71Ø9 เป็นตัวแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอลขนาด 12 บิท สำหรับแรงดันอ้างอิงมาตราฐานจะใช้ไอซีเบอร์ MC 14Ø4A เป็นตัวสร้าง ไอซีแปลงสัญญาณอนาลอกเป็นดิจิตอลนี้ ต้องใช้แหล่งจ่ายไฟบวกและลบในการทำงาน จึงต้องมีวงจรในการสร้างแรงดันบวกและลบให้วงจรด้วย ซึ่งจะใช้ไอซี 555 และไอซี 78M15 78MØ6, 78MØ5 ในการทำงาน สัญญาณดิจิตอล 12 บิทที่ได้จะต่อผ่านบัฟเฟอร์เบอร์ 74LS245 จำนวน 2 ตัว แล้วผ่านวงจรดับปลิ้งสัญญาณด้วย Opto Isolator ส่งเข้าไอซี 74LS251 ซึ่งทำหน้าที่เป็นตัวเลือกสัญญาณบิทต่าง ๆ ส่งไปยังดาต้าบัส D7 ของโมดูลประมวลผล การเลือกอ่านสัญญาณบิทต่างๆ จะใช้สัญญาณแอดเดรส AØ-A7 โดยจะต่อผ่านไอซีดีโค้ดสัญญาณเบอร์74LS 138 ด้วย สำหรับบัฟเฟอร์ของบัสต่างๆ ในโมดูลจะใช้ไอซีเบอร์ 74LS245 เป็นบัฟเฟอร์

วงจรการอ่านรหัสของโมดูลอนาลอกอินพุทจะเหมือนของโมดูลอินพุท คือใช้ไอซีเลือกสัญญาณเบอร์ 74LS251 ในการรหัส สัญญาณเลือกการอ่านรหัสจะถูกดีโค้ดโดยไอซี 74LS138 เช่นกัน รหัสของโมดูลอนาลอกอินพุทที่ใช้คือ "Ø1ØØ"



รูปที่ 3.11 แสดงโครงสร้างของโมดูลเอาท์พุท



รูปที่ 3.12 แสดงโครงสร้างของโมดูลอนาลอกอินพุท

3.2.6 โมดูลกำหนดค่าตัวเลข

โมดูลนี้ใช้สำหรับตั้งค่าตัวเลขฐานสิบขนาด 4 หลัก จำนวน 4 แชลแนล เพื่อนำไปเป็นค่าพารามิเตอร์ของฟังก์ชั่นในการทำงานของโปรแกรม การตั้งค่าตัวเลขจะใช้ Thumb wheel switch

โมดูลนี้ไม่มีสัญญาณติดต่อกับภายนอกจึงไม่ต้องมีวงจรคับปลิ้งสัญญาณต่างๆ การทำงานของวงจรจะใช้สัญญาณ BCD code จากตัว Thumb wheel switch เป็นสัญญาณอินพุทของตัวเลือกสัญญาณ (Data Selector) โดยตรง สัญญาณการเลือกตำแหน่งอินพุทต่างๆ จะใช้สัญญาณแอดเดรส AØ-A7 โดยต่อผ่านไอซีดีโค๊ดเบอร์ 74LS138 2 ตัว บัฟเฟอร์ของสัญญาณสำหรับโมดูลนี้ใช้ไอซีเบอร์ 74LS541

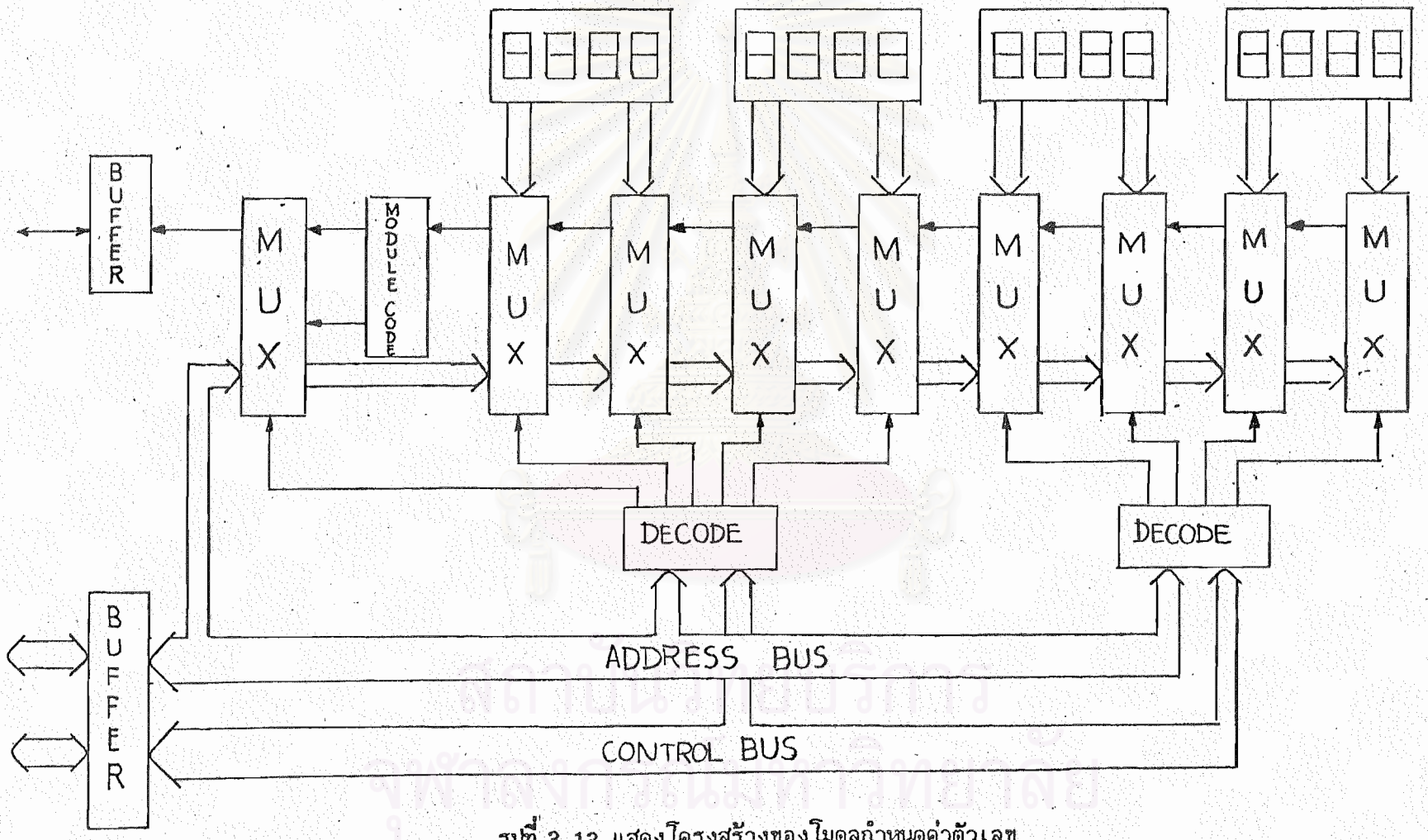
รหัสของโมดูลอนาลอกอินพุทคือ "1ØØØ" ซึ่งถูกอ่านผ่านไอซี 74LS251 โดยมีการทำงานเหมือนของโมดูลอินพุท

3.2.7 ตัวป้อนโปรแกรมแบบมือถือ (Hand held programming console)

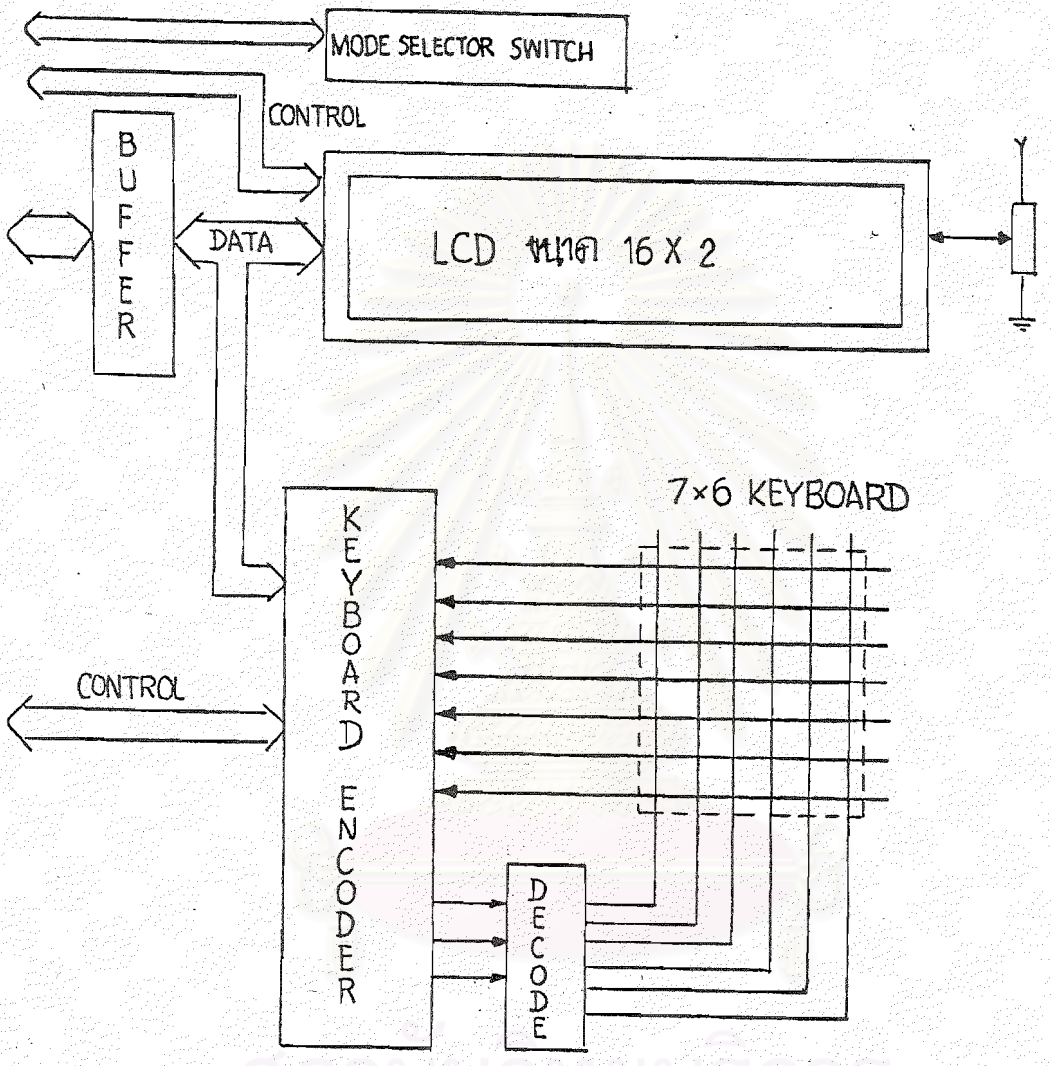
วงจรของตัวป้อนโปรแกรมแบบมือถือประกอบด้วย 2 ส่วนใหญ่ ๆ คือ วงจรแสดงผล LCD ซึ่งจะเป็นโมดูลแสดงผลขนาด 16x2 ตัวอักษร วงจรดีโค๊ดและเปลี่ยนสัญญาณสำหรับตัว LCD อยู่ที่โมดูลแสดงผลและคีย์บอร์ดแล้ว ดังนั้นจึงต่อสัญญาณต่างๆ โดยตรงไปยังตัว LCD

วงจรส่วนที่ 2 คือ วงจร Keyboard encoder ซึ่งใช้ไอซีเบอร์ 8279 ซึ่งเป็นไอซีเฉพาะในการอ่านคีย์บอร์ด สัญญาณการสแกนแถวของคีย์บอร์ดจากไอซี 8279 จะต่อผ่านไอซีดีโค๊ดเบอร์ 74LS138 ก่อน แล้วส่งเป็นสัญญาณเลือกคอลัมภ์ของ Key Matrix คีย์บอร์ดที่ใช้จะมีขนาด 7 แถว x 6 คอลัมภ์ รวมเป็น 42 คีย์

บัฟเฟอร์ของดาต้าบัสของตัวป้อนโปรแกรมแบบมือจะใช้ไอซีเบอร์ 74LS245 ในการทำงาน



รูปที่ 3.13 แสดงโครงสร้างของโมดูลกำหนดค่าตัวเลข



รูปที่ 3.14 แสดงโครงสร้างของตัวป้อนโปรแกรมแบบมือถือ

บทที่ 4

ซอฟท์แวร์ของเครื่องควบคุมชนิดโปรแกรมได้

โปรแกรมซอฟท์แวร์ควบคุมระบบของเครื่อง PC ในการวิจัยนี้เป็นภาษาแอสเซมบลี้ (Assembly language) เพราะเป็นภาษาที่มีความเร็วในการทำงานสูง และสามารถเขียนโปรแกรมควบคุมในรายละเอียดส่วนต่าง ๆ ของวงจรได้ดี การเขียนโปรแกรมจะเป็นลักษณะโมดูล แล้วนำมาเชื่อมต่อกัน (Link) เป็นโปรแกรมรวมอีกครั้ง การพัฒนาหรือแก้ไขเพิ่มเติมในอนาคตสามารถทำได้สะดวก โปรแกรมควบคุมการทำงานของเครื่องจะแบ่งออกได้เป็น 2 ส่วนที่สำคัญคือ โปรแกรมควบคุมในโหมดโปรแกรม และโปรแกรมควบคุมในโหมดทำงาน สำหรับเนื้อหาของบทนี้นอกจากจะกล่าวถึงโปรแกรมควบคุมการทำงานหลัก 2 ส่วนข้างต้นแล้วยังจะกล่าวถึงส่วนอื่นๆ อีกเช่น การอินเตอร์รัพท์ การแสดงผล การจัดเก็บข้อมูล และคำสั่ง การทำงานของคำสั่งฟังก์ชั่นต่าง ๆ

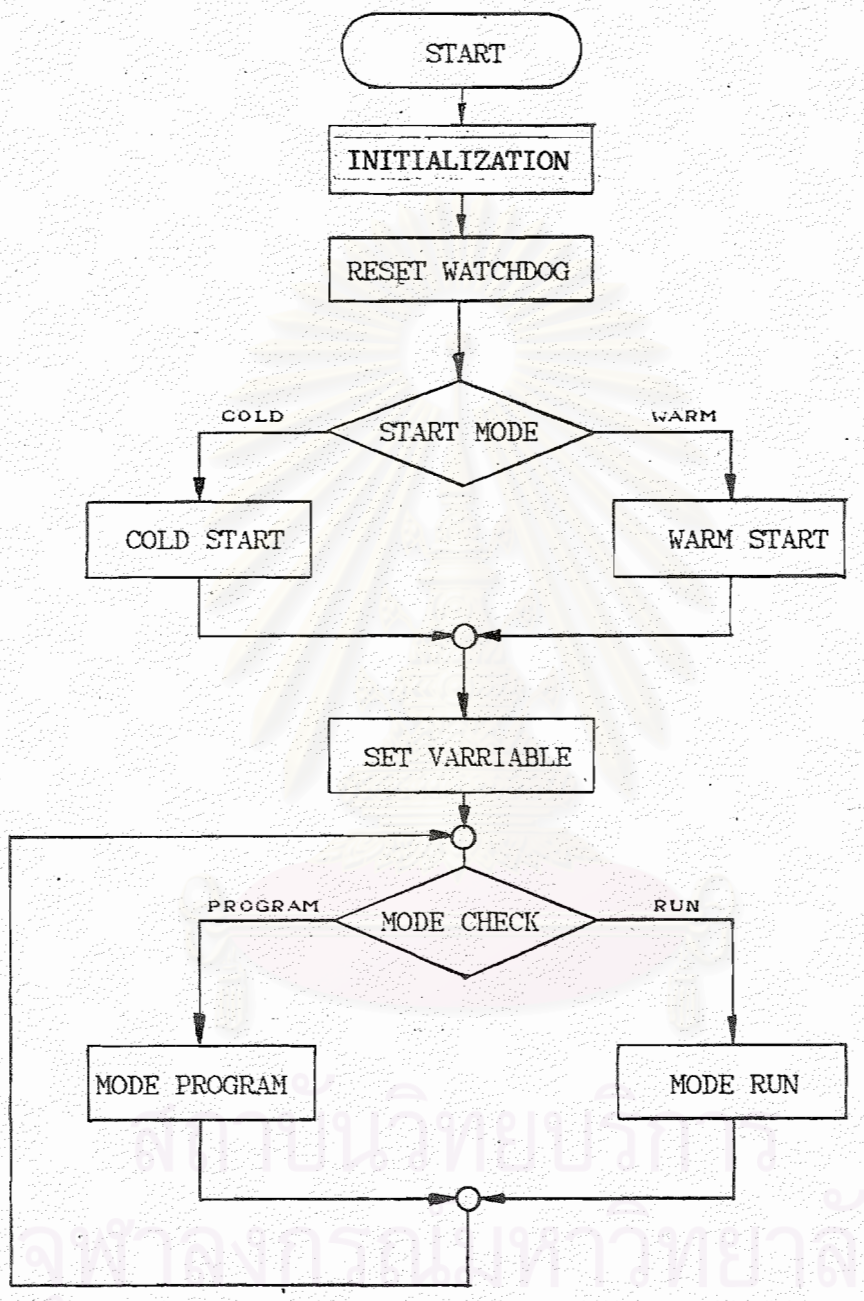
4.1 การทำงานของโปรแกรมควบคุม

ลักษณะการทำงานของโปรแกรมควบคุมเครื่องสามารถแสดงด้วยรูปที่ 4.1 โดยเมื่อเริ่มต้นทำงานจะมีการกำหนดค่าเริ่มต้นต่างๆ ของตัวแปร และตั้งโปรแกรมของอุปกรณ์ต่างๆ มีการรีเซ็ทวงจรวอชท์ดอก แล้วจึงตรวจสอบว่าการเริ่มต้นทำงานเป็น Cold start* หรือ Warm start* ลำดับต่อมาจะเป็นการตรวจสอบว่าเป็นการทำงานในโหมดโปรแกรม (Program mode) หรือโหมดการทำงาน (Run mode) ในขณะที่มีการทำงานในโหมดต่าง ๆ จบหนึ่งครั้งก็จะมีการตรวจสอบดูว่ามีการเปลี่ยนโหมดการทำงานหรือไม่ สำหรับรายละเอียดในการทำงานของโปรแกรมส่วนต่าง ๆ ที่สำคัญ จะอธิบายเป็นส่วน ๆ ต่อไป

4.1.1 การเริ่มต้นทำงาน

การเริ่มต้นทำงานของโปรแกรมควบคุมเครื่องจะเริ่มที่ตำแหน่งแอดเดรส 0000 โดย

* จะอธิบายในหัวข้อ 4.1.1



รูปที่ 4.1 แสดงโฟล์วชาร์ทหลักในการทำงานของเครื่อง

มีการเริ่มต้นทำงานได้ 2 วิธีคือ

1. การเริ่มต้นทำงานด้วยการเปิดเครื่อง หรือเรียกว่า Cold start

2. การเริ่มต้นทำงานจากคำสั่งหรือสัญญาณภายในเครื่อง หรือเรียกว่า Warm start

การแยกการเริ่มต้นทำงานของโปรแกรมออกเป็น 2 แบบนี้ เพื่อต้องการให้เครื่อง PC สามารถทำงานต่อเนื่องโดยอัตโนมัติต่อไปได้ เมื่อมีสัญญาณรบกวนจากภายนอกมาทำให้เครื่อง PC ทำงานผิดพลาดจนถูกรีเซ็ทโดยวงจรวอชท์ดอกไทเมอร์

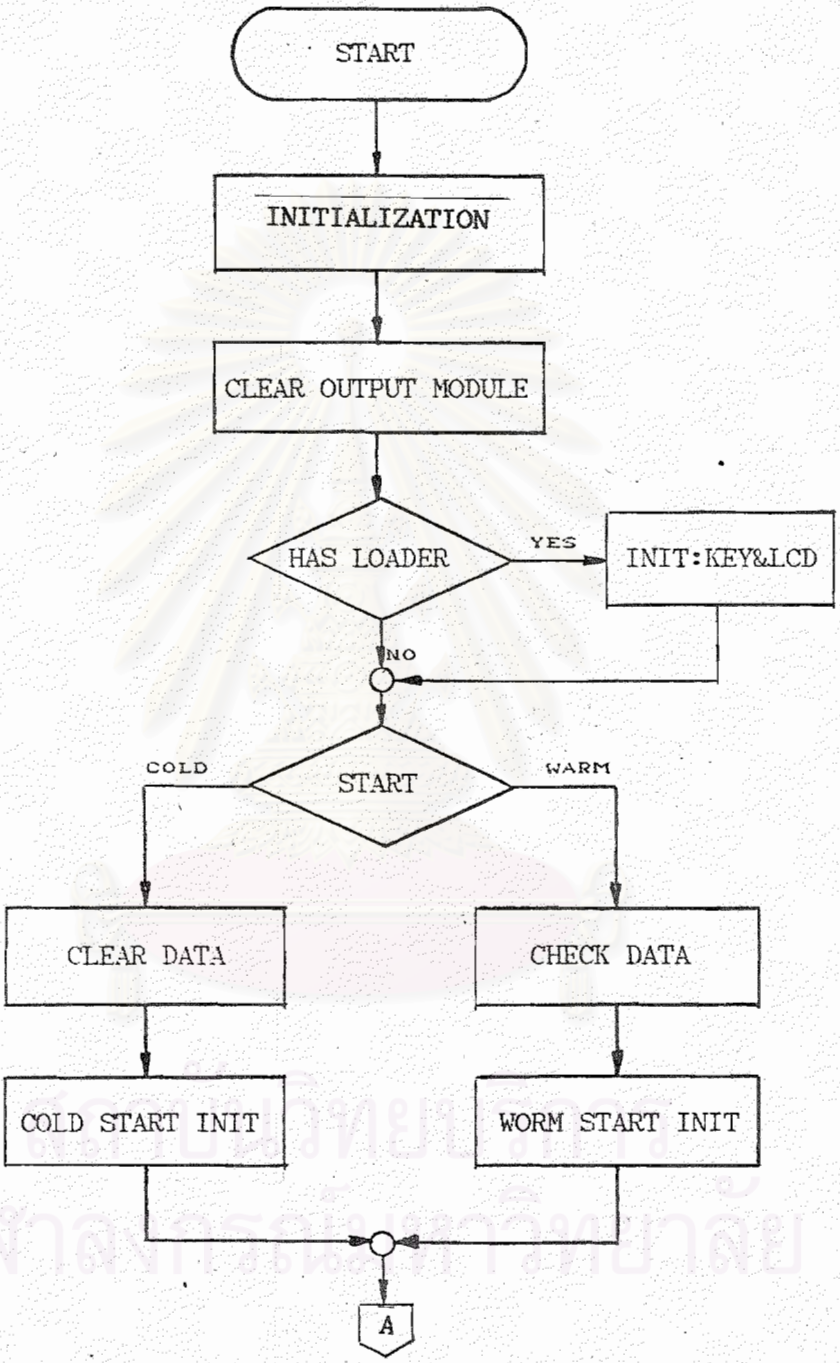
เนื่องจากการรีเซ็ทโดยวงจร Power on reset หรือโดยวงจร Watchdog timer และการกระโดดมาทำงานที่แอดเดรส 0000 นั้น ซีพียูไม่สามารถรู้ได้ว่าเป็นการเริ่มต้นทำงานโดยวิธีใด ดังนั้นการตรวจสอบว่าเป็น Cold start หรือ Warm start จะใช้วิธีการอ่านค่าพารามิเตอร์ของอุปกรณ์บางตัวในวงจร ถ้าเป็น Cold start หรือเริ่มเปิดเครื่องค่าพารามิเตอร์นี้จะมีค่าไม่แน่นอนเพราะยังไม่มีการโปรแกรมลงไป แต่ถ้า Warm start จะมีค่าตรงกับที่โปรแกรมไว้ ซึ่งทำให้สามารถแยกสถานะทั้งสองได้

การทำงานของ Warm start นี้จะช่วยให้เสถียรภาพของเครื่องดีขึ้นคือ เมื่อเครื่อง PC กำลังทำงานในโหมดทำงานอยู่ และมีสัญญาณรบกวนจากภายนอกเข้ามาทำให้ซีพียูทำงานผิดพลาด และไม่อยู่ในโปรแกรมที่ให้ทำงาน ซึ่งเมื่อเวลาผ่านไปวงจรวอชท์ดอกไทเมอร์จะส่งสัญญาณรีเซ็ทไปยังซีพียู ซีพียูจะเริ่มต้นทำงานใหม่และตรวจสอบพบว่าเป็น Warm start ก็จะไม่ลบข้อมูลต่าง ๆ ของเดิมทิ้งโดยจะทำงานต่อเนื่องไป ทำให้กระบวนการที่เครื่อง PC ควบคุมอยู่สามารถทำงานต่อเนื่องไปได้โดยไม่หยุดชะงัก

การทำงานของโปรแกรมควบคุมตอนเริ่มต้น ดังแสดงในรูปที่ 4.2 ประกอบด้วยบล็อกต่าง ๆ ซึ่งมีหน้าที่ดังนี้คือ

INITIALIZATION — เป็นโปรแกรมกำหนดค่าเริ่มต้น ในการทำงานของเครื่อง PC คือ
- เซ็ทอินเตอร์รัพท์โหมด และสถานะการอินเตอร์รัพท์
- เซ็ทค่าสแตก
- กำหนดค่าเริ่มต้นให้ตัวแปรต่าง ๆ ในการทำงาน
- ตรวจสอบโมดูลต่าง ๆ ที่ต่อในระบบ และสร้างตารางเก็บสถานะของโมดูลต่าง ๆ ไว้

Clear output module — เนื่องจากเมื่อเริ่มเปิดเครื่องสถานะของเอาท์พุทพอร์ทของโมดูลต่าง ๆ จะมีค่าไม่แน่นอน จำเป็นต้องเคลียร์สถานะให้เป็น OFF ทั้งหมด



รูปที่ 4.2 แสดงโฟล์วชาร์ทการทำงานของเครื่อง PC

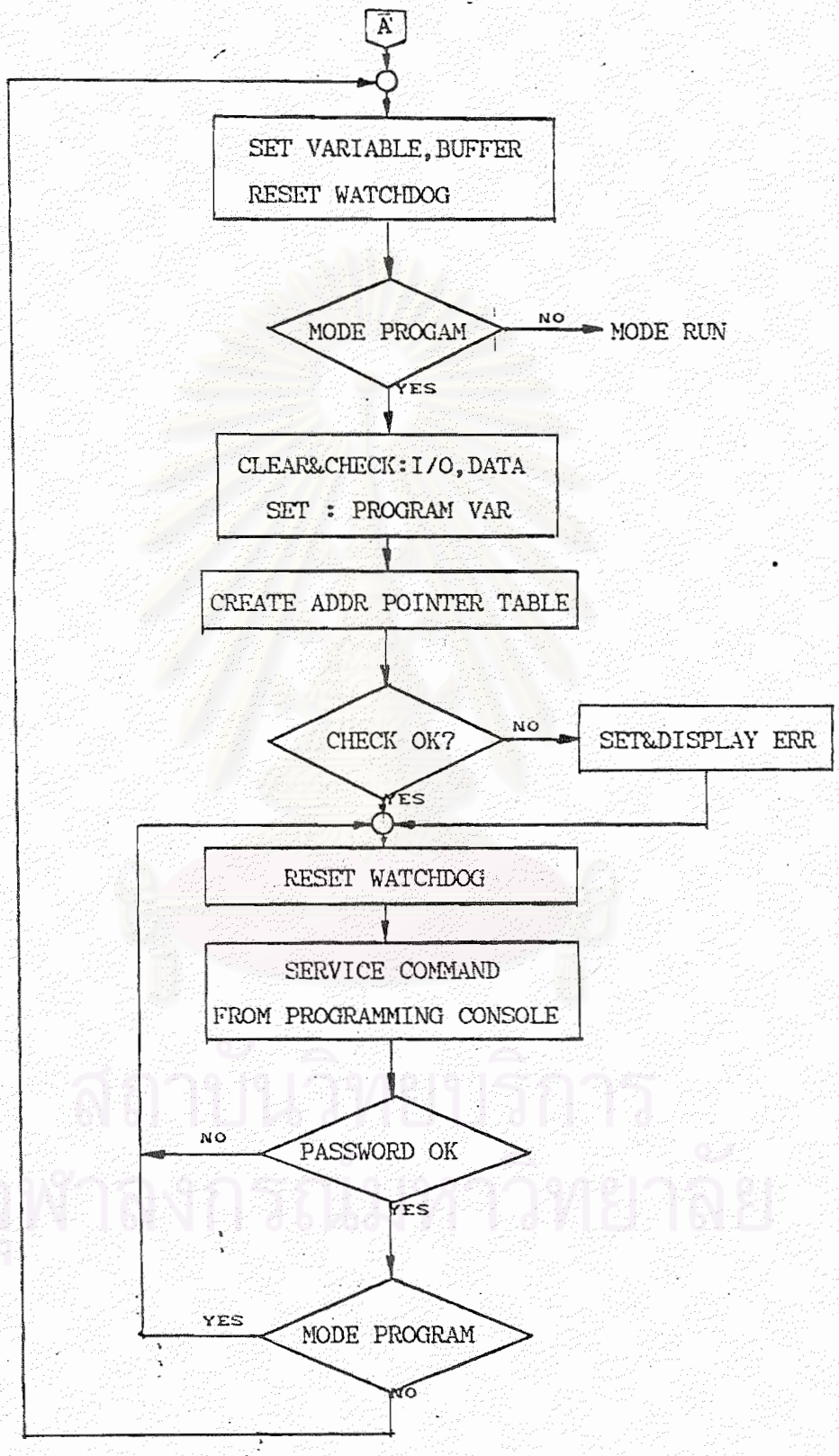
Keyboard & LCD Initialization	เป็นการโปรแกรมการทำงานของวงจรคีย์บอร์ด (ไอซี 8279) และการเซ็ทโหมดการทำงานต่างๆ ของวงจร LCD ที่ใช้แสดงผล กรณีที่ตัวป้อนโปรแกรมแบบมือไม่ได้ต่ออยู่กับเครื่อง PC จะมีการเซ็ทค่าตัวแปรเพื่อป้องกันโปรแกรมควบคุมการทำงาน ไม่ให้เสียเวลาในการอ่านคีย์บอร์ด และแสดงผล
Clear data	เป็นโปรแกรมเคลียร์ข้อมูลอินพุท/เอาท์พุทต่าง ๆ ในหน่วยความจำ และเคลียร์ข้อมูลของตัวตั้งเวลา (Timer, ข้อมูลสำหรับคำสั่งจัดการข้อมูล)
Check data	เป็นโปรแกรมจะตรวจสอบข้อมูลต่างๆ ในหน่วยความว่าถูกต้องหรือไม่ ถ้ามีค่าไม่ถูกต้องก็จะเคลียร์ข้อมูลที่ตำแหน่งนั้น ถ้าข้อมูลมีรูปแบบถูกต้องก็ยังคงสถานะเดิมไว้
Cold start initialization	เป็นการโปรแกรมฟังก์ชั่นการทำงานของอุปกรณ์ต่างๆ เช่น Z80-CTC โมดูล LCD
Warm start initialization	เป็นการกำหนดค่าเริ่มต้นให้ตัวแปรต่างๆ

4.1.2 <u>การทำงานโหมดโปรแกรม</u> (Program mode)

โหมดโปรแกรมออกแบบมาให้ใช้งานเกี่ยวกับ การป้อนโปรแกรมขั้นบันไดลงในหน่วยความจำ การแก้ไขคำสั่งหรือโอเปอร์แรนด์ (Operand) ของคำสั่งต่าง ๆ การลบคำสั่ง (Delete) การสอดแทรกคำสั่ง (Insert) การค้นหาตำแหน่งของคำสั่ง (Search) การดูสถานะหรือค่าของข้อมูลต่างๆ (Monitor) และการเซ็ทหรือรีเซ็ทสถานะของข้อมูลรีเลย์ และการทำงานของคำสั่งช่วยของเครื่อง เช่น การลบโปรแกรม การลบข้อมูล การป้อนรหัส Password

การทำงานของโปรแกรมควบคุมในโหมดโปรแกรมแสดงเป็นโฟล์วชาร์ทในรูปที่ 4.3 ซึ่งมีขั้นตอนการทำงานดังนี้คือ

1. รีเซ็ทวงจรวอทช์ดอกไทเมอร์ และกำหนดค่าเริ่มต้นให้ตัวแปรที่ใช้ร่วมในโหมดโปรแกรมและโหมดทำงาน



รูปที่ 4.3 แสดงโฟล์วชาร์ทการทำงานโหมดโปรแกรม

2. ตรวจสอบว่าเป็นการทำงานโหมดใด ถ้าเป็นโหมดทำงานก็กระโดดไปที่โหมดทำงาน ถ้าเป็นโหมดโปรแกรมก็ทำงานในขั้นต่อไป

3. ลบค่าของข้อมูลรีเลย์ และข้อมูลแบบ Non Retentive ในเครื่อง PC ทั้งหมด ตรวจสอบข้อมูลแบบ Retentive ถ้ามีรูปแบบที่ผิดก็ลบข้อมูลตัวนั้นทิ้ง และกำหนดค่าให้ตัวแปรที่ต้องใช้งานในโหมดโปรแกรม

4. เป็นการสร้างตารางชี้ตำแหน่งโปรแกรมขั้นบันไดของผู้ใช้ เพื่อใช้ในการเลื่อนไปยังตำแหน่งต่าง ๆ ของโปรแกรมขั้นบันได เพราะโปรแกรมขั้นบันไดแต่ละคำสั่งมีความยาวไปเท่ากัน

5. ตรวจสอบโปรแกรมขั้นบันไดที่เก็บในหน่วยความจำว่ามีที่ผิดพลาดหรือเปล่า ถ้ามีก็จะเซ็ทค่าตัวแปรไว้ และแสดงค่าที่ผิดพลาดบอกให้ผู้ใช้ทราบ

6. รีเซ็ทวงจรวอทช์ดอกไทเมอร์

7. เป็นการตรวจสอบการกดคีย์บอร์ดของผู้ใช้ ถ้ามีการกดคีย์บอร์ดก็จะไปทำงานที่โปรแกรมจัดการคีย์บอร์ด ซึ่งใช้อัลกอริธึมแบบ Keyboard Parser โปรแกรมการทำงานส่วนนี้จะคล้ายกับโปรแกรมในส่วน Service peripheral device ในโหมดทำงาน

8. หลังจากทำงานตามการกดคีย์แล้วจะตรวจสอบสถานะของ Password ถ้ายังไม่ป้อน Password จะไม่ยอมเปลี่ยนโหมด โดยจะกลับไปทำงานในขั้นตอนที่ 6 ซ้ำใหม่

9. ตรวจสอบว่ามีการเปลี่ยนโหมดหรือไม่ ถ้าไม่เปลี่ยนจะไปทำงานในขั้นตอนที่ 6 ซ้ำใหม่ ถ้ามีการเปลี่ยนโหมดเป็นโหมดทำงาน ก็จะข้ามไปยังขั้นตอนที่ 1 อีกครั้ง

4.1.3 <u>โหมดทำงาน</u> (Run mode)

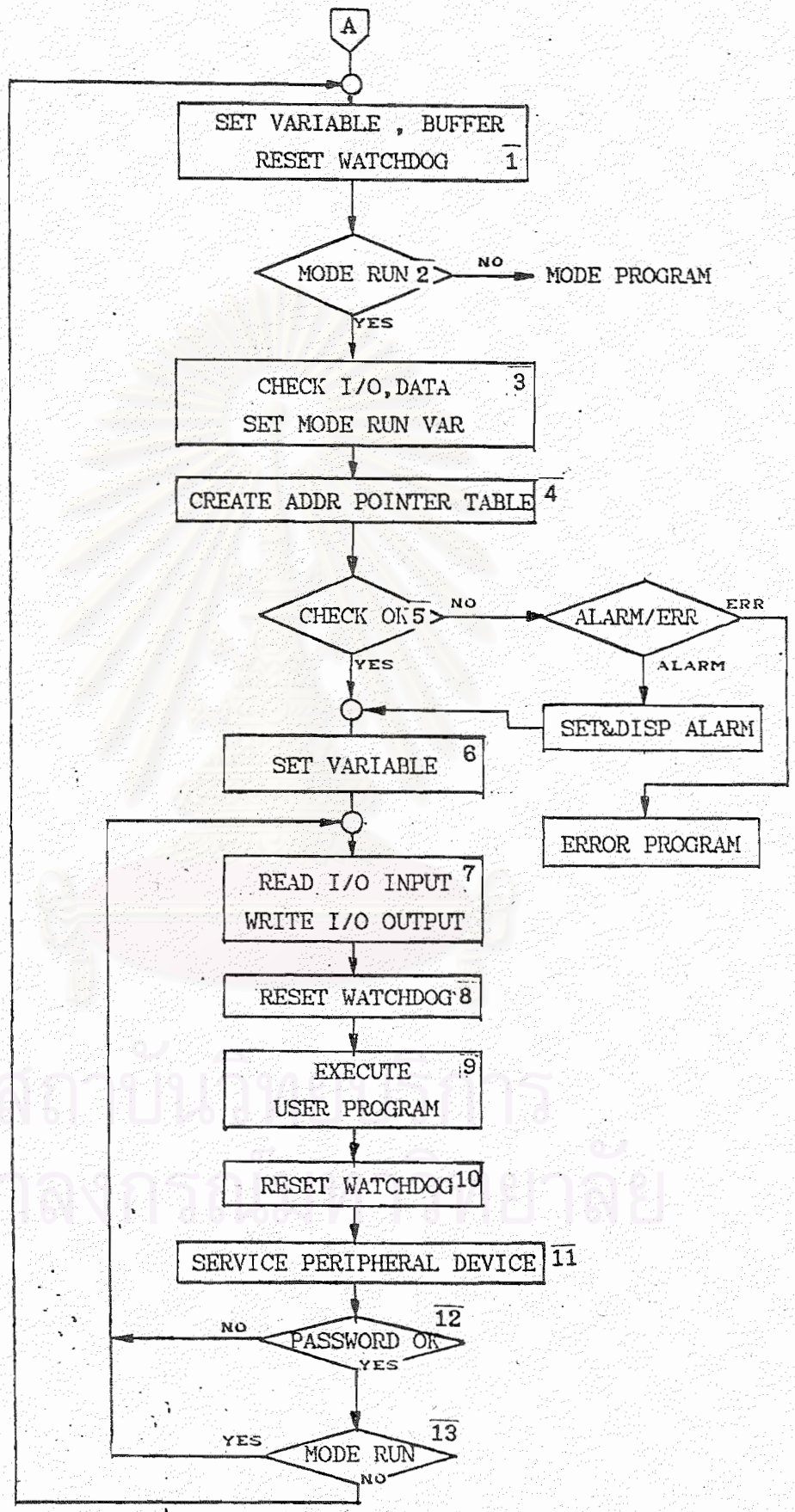
การทำงานของโหมดทำงานที่สำคัญคือ การทำงานตามโปรแกรมขั้นบันไดของผู้ใช้ที่เก็บไว้ในหน่วยความจำ ซึ่งต้องการให้ทำงานเร็วที่สุด สำหรับการแปลงโปรแกรมขั้นบันไดของเครื่องนี้จะใช้วิธี Compiler[8] ร่วมกับวิธี Call[8] ซึ่งได้กล่าวมาแล้วในบทที่ 2 สำหรับการอ่าน I/O อินพุท และการเขียน I/O เอาท์พุท ในโหมดทำงานนี้จะกล่าวถึงอีกครั้งในหัวข้อถัดไป การทำงานในโหมดทำงานมีขั้นตอนดังนี้

1. รีเซ็ทวงจรวอทช์ดอกไทเมอร์ และกำหนดค่าให้กับตัวแปรต่าง ๆ ที่ใช้ร่วมกันทั้งโหมดโปรแกรมและโหมดทำงาน

2. ตรวจสอบโหมดการทำงาน และไปทำงานตามโหมดนั้น ๆ

3. ตรวจสอบข้อมูลของ I/O รีเลย์ต่าง ๆ ถ้ามีรูปแบบที่ผิดพลาดจะเคลียร์ข้อมูลตำแหน่งนั้น และเซ็ทค่าเริ่มต้นสำหรับตัวแปรของฟังก์ชั่น TIM, TIMH, SFT, CNT, DIFU, DIFD

4. สร้างตารางชี้ตำแหน่งโปรแกรมขั้นบันไดที่เก็บในหน่วยความจำ เพื่อใช้ในการ



รูปที่ 4.4 แสดงโฟลว์ชาร์ทการทำงานของโหมดทำงาน

เลื่อนไปยังตำแหน่งต่างๆ ของคำสั่ง ในการเรียกดูโปรแกรมและการค้นหาโปรแกรมของผู้ใช้

5. ตรวจสอบความผิดพลาด ซึ่งแบ่งออกได้เป็น 2 แบบคือ ความผิดพลาดเล็กน้อยสามารถทำงานได้เรียกว่า ALARM เช่น สถานะข้อมูลที่เก็บไว้ผิดพลาด เป็นต้น และความผิดพลาดที่ระบบไม่สามารถทำงานได้เรียกว่า ERROR เช่น ผู้ใช้ลืมโปรแกรมคำสั่งจบโปรแกรม (END) ในโปรแกรมชั้นบันไดที่เขียน เป็นต้น

กรณีที่เกิด ERROR ขึ้นระบบจะไม่ทำงานตามโปรแกรมชั้นบันได ผู้ใช้ต้องเปลี่ยนการทำงานไปโหมดโปรแกรมแล้วแก้ไขโปรแกรมให้ถูกต้องเสียก่อน

6. กำหนดค่าเริ่มต้นให้กับตัวแปรหรือพารามิเตอร์ต่าง ๆ ที่ใช้ในโหมดทำงาน เช่น ข้อมูลเริ่มต้น ตำแหน่งเริ่มทำงาน

7. อ่านข้อมูลอินพุทพอร์ทของโมดูลต่าง ๆ มาเก็บในหน่วยความจำ และเขียนข้อมูลจากหน่วยความจำไปยังพอร์ทเอาท์พุทของโมดูลต่าง ๆ

8. รีเซ็ทวงจรวอทช์ดอกไทเมอร์

9. ทำงานตามโปรแกรมชั้นบันไดที่เก็บในหน่วยความจำ โดยเริ่มทำงานจากคำสั่งแรกจนถึงคำสั่งจบโปรแกรม (END)

10. รีเซ็ทวงจรวอทช์ดอกไทเมอร์

11. ตรวจสอบสถานะของอุปกรณ์ที่มาต่อว่าจะต้องทำงานอะไรให้บ้างหรือไม่ เช่น ถ้ามีการกดคีย์บอร์ด ก็จะทำงานตามโปรแกรมควบคุมคีย์บอร์ด ซึ่งมีลักษณะเหมือนการทำงานในโหมดโปรแกรม

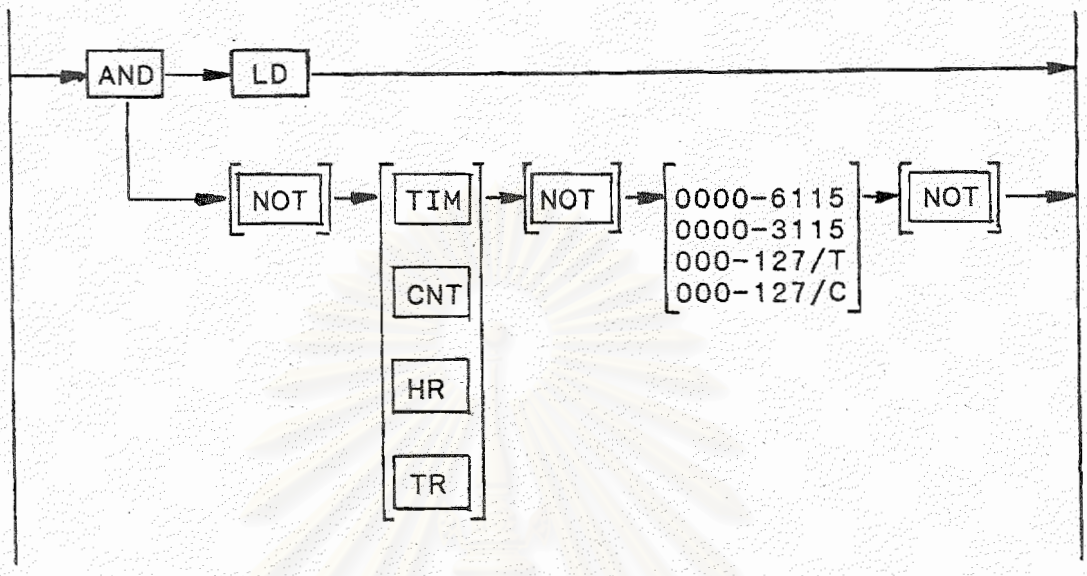
12. ตรวจสอบว่ามีการป้อน (Password) หรือยัง ถ้ายังไม่ป้อนจะกลับไปทำงานซ้ำขั้นตอนที่ 7 ใหม่

13. ตรวจสอบการเปลี่ยนโหมดการทำงาน ถ้าไม่มีการเปลี่ยนจะกลับไปทำงานขั้นตอนที่ 7 ถ้ามีการเปลี่ยนเป็นโหมดโปรแกรมจะไปทำงานขั้นตอนที่ 1

4.1.4 โปรแกรมจัดการคีย์บอร์ด

ลำดับขั้นการโต้ตอบการกดคีย์บอร์ดของผู้ใช้ในเครื่อง PC มีลักษณะที่ยุ่งยากซับซ้อน เนื่องจากมีปุ่มคีย์บอร์ด 42 คีย์ และมีคำสั่งชั้นบันไดต่าง ๆ ที่มีลักษณะแตกต่างกันเป็นจำนวนมาก การเขียนโปรแกรมควบคุมการทำงานในส่วนนี้จะใช้อัลกอริธึมที่เรียกว่า Keyboard Parser [10] ซึ่งมีลักษณะการทำงานแบบลำดับ โดยมีการเก็บสถานะลำดับปัจจุบันไว้

ตัวอย่างแสดงลักษณะการกดคีย์ "AND" เพื่อป้อนโปรแกรมซึ่งสามารถกดคีย์บอร์ดลำดับต่อไปหลายแบบ ดังแสดงในรูปที่ 4.5



รูปที่ 4.5 แสดงลำดับขั้นการป้อนคำสั่ง AND

เพื่อความสะดวกในการทำงานของโปรแกรมควบคุมการกดคีย์บอร์ด เราจะมีการแปลงรหัสของคีย์บอร์ดที่อ่านได้จากการกดคีย์ โดยคีย์บอร์ดที่กดมา 1 ครั้งจะแทนด้วยพารามิเตอร์ 2 ตัวคือ FNKY และ NUMB

FNKY หมายถึง รหัสฟังก์ชั่นในการทำงานของคีย์นั้น

NUMB หมายถึง ค่าข้อมูลของรหัส (FNKY) คีย์นั้น ๆ

ลักษณะที่เห็นได้ชัดคือ คีย์ตัวเลข 0, 1, 2, 3,..., 9 จะแทนด้วย FNKY ค่าเดียวกัน แต่มี NUMB ต่างกัน เพราะหน้าที่การทำงานของคีย์ตัวเลข 0-9 จะเหมือนกัน แต่มีค่าข้อมูลต่างกันค่า FNKY และ NUMB ของคีย์ต่าง ๆ แสดงในตารางที่ 4.1

หน้าที่ของโปรแกรมควบคุมการกดคีย์บอร์ดที่สำคัญ แบ่งออกเป็น 3 ส่วนคือ

1. อ่านค่าคีย์บอร์ดที่มีการกด และแปลงเป็นรหัส FNKY และ NUMB
2. ตรวจสอบค่า FNKY ของคีย์ที่กด เทียบกับสถานะ (State) เดิมว่าควรเปลี่ยนสถานะเดิมเป็นสถานะใหม่อะไร
3. ทำโปรแกรมตอบสนองการกดคีย์นั้น

การทำงานของโปรแกรมควบคุมการกดคีย์บอร์ด มีตัวแปรที่เกี่ยวข้องในการทำงานดังนี้

FNKY เป็นตัวแปรที่เก็บรหัสการกดคีย์บอร์ด

NUMB เป็นตัวแปรเก็บค่าตัวเลขของการกดคีย์บอร์ด

PREST เป็นตัวแปรที่ใช้เก็บสถานะปัจจุบัน (Present state) ซึ่งจะใช้อ้างอิงในการเปรียบเทียบเพื่อหาสถานะต่อไป

PARSER STATE TABLE หมายถึงตารางที่ใช้ในการเปรียบเทียบเพื่อหาสถานะต่อไป ตารางนี้ประกอบด้วย FNKYT, NEXST และ ARNO

FNKYT เป็นรหัสของคีย์ที่เก็บไว้ในตาราง PARSER STATE ซึ่งจะเก็บไว้สำหรับเปรียบเทียบกับตัวแปร FNKY เพื่อหาค่า NEXST และ ARNO

NEXST เป็นค่าของสถานะที่เก็บไว้ในตาราง PARSER STATE เพื่อที่จะนำไปเปลี่ยนค่าของสถานะปัจจุบัน

ARNO ACTION ROUNTINE NUMBER เก็บไว้ใน PARSER STATE TABLE ใช้เป็นตัวชี้ตำแหน่งโปรแกรมตอบสนองการกดคีย์

โฟลว์ชาร์ทการทำงานของโปรแกรมควบคุมคีย์บอร์ดแสดงในรูปที่ 4.6 ซึ่งมีลักษณะการทำงานดังนี้

1. ตรวจสอบว่ามีการกดคีย์บอร์ด ถ้ามีจะนำรหัสคีย์บอร์ดไปแปลงเป็นค่า FNKY และ NUMB กรณีที่ FNKY ไม่เท่ากับ OFFH ก็จะทำขั้นต่อไป

2. ใช้ค่าสถานะปัจจุบัน (PREST) เป็นตัวชี้หาตำแหน่งเริ่มต้นของตาราง PARSER STATE TABLE แล้วนำค่า FNKY ไปเปรียบเทียบกับค่า FNKYT ในตารางจนกว่าจะพบค่าที่เท่ากัน หรือจนกว่าจะพบเครื่องหมาย "*"

3. นำค่า NEXST ที่ได้จากตารางไปแทนค่า PREST

4. ตรวจสอบดูค่าของ PREST ถ้าเท่ากับ Initial state ก็กลับไปทำขั้นตอนที่ 2 ใหม่

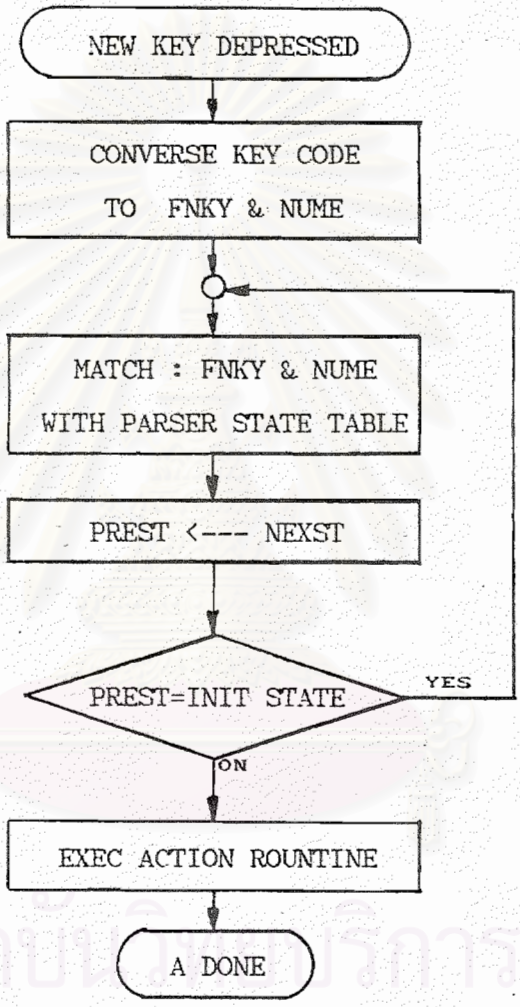
5. ทำโปรแกรมตอบสนองการกดคีย์บอร์ดตามค่า ARNO ที่ได้จากตาราง

KEY LABEL	KEY CODE	FNKY	NUMB
0	06	01	0
1	05	01	1
2	0D	01	2
3	15	01	3
4	04	01	4
5	0C	01	5
6	14	01	6
7	03	01	7
8	0B	01	8
9	13	01	9
LD	02	02	0
AND	01	03	0
OR	09	04	0
OUT	0A	05	0
TIM	12	06	0
CNT	11	07	0
NOT	10	08	0
HR	29	09	0
TR	19	0A	0
DM	1A	0B	0
*	22	0C	0
#	2A	0D	0
SHIFT	28	0E	0
FUN	00	0F	0
CLR	16	10	0
WRITE	26	11	0

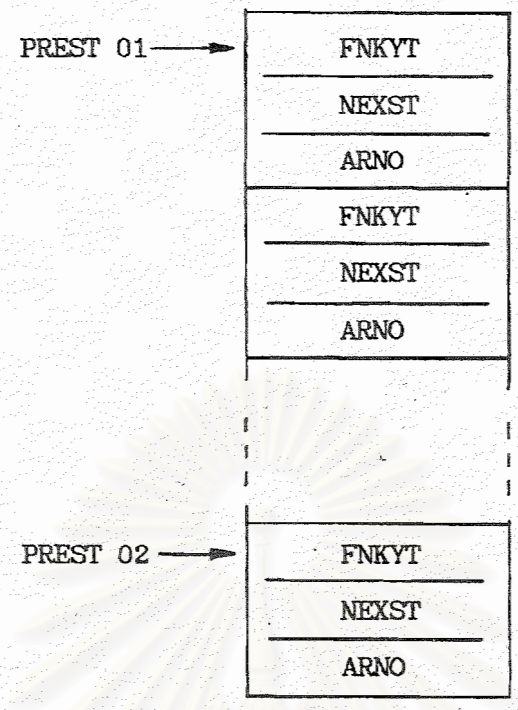
ตารางที่ 4.1 แสดงการแปลงรหัสของคีย์บอร์ด

KEY LABEL	KEY CODE	FNKY	NUMB
DEL	24	12	0
INS	25	13	0
MONTR	2C	14	0
SRCH	2B	15	0
CHG	23	16	0
SET	1C	17	0
RESET	1D	18	0
⇧	2D	19	0
⇩	2E	1A	0
SFT	08	1B	0
A	06	1C	0A
B	05	1C	0B
C	0D	1C	0C
D	15	1C	0D
E	04	1C	0E
F	0C	1C	0F
CH	2A	1D	0
CONT	22	1E	0
---	---	FF	0

ตารางที่ 4.1 (ต่อ) แสดงการแปลงรหัสของคีย์บอร์ด



รูปที่ 4.6 แสดงโฟลว์ชาร์ทการทำงานของโปรแกรมควบคุมคีย์บอร์ด



รูปที่ 4.7 แสดงตำแหน่งของการจัดเก็บของ PARSER STATE TABLE

4.1.5 การอินเตอร์รัพท์ (Interrupts)

การอินเตอร์รัพท์ของเครื่อง PC นี้จะใช้สัญญาณ INT จากไอซี Z80-CTC ซึ่งโปรแกรมไว้ให้สร้างฐานเวลา 0.005 วินาที โหมดของการอินเตอร์รัพท์ที่ใช้เป็นโหมด 1 เมื่อซีพียูได้รับสัญญาณ INT จะกระโดดไปทำงานที่แอดเดรส 0038H

หน้าที่ที่สำคัญของโปรแกรมบริการอินเตอร์รัพท์คือ

1. เซ็ทพารามิเตอร์สำหรับฐานเวลา 0.01 และ 0.1 วินาที เพื่อใช้ในคำสั่ง TIMH และ TIM
2. สร้างสัญญาณพัสช์สำหรับรีเลย์ช่วยต่อไปนี้

รีเลย์เบอร์	6300	สัญญาณ 0.1 วินาที
	6301	สัญญาณ 0.2 วินาที
	6302	สัญญาณ 1 วินาที

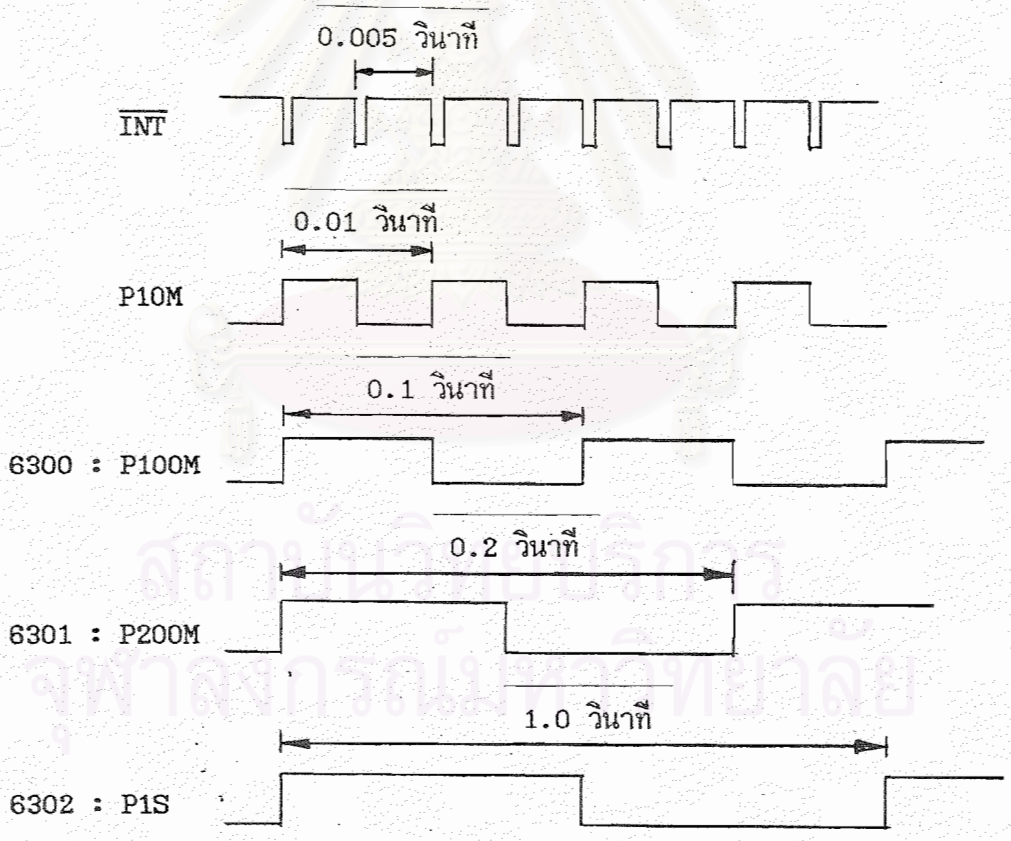
3. แสดงผลของข้อมูลที่ LCD
4. แสดงผลของข้อมูล สำหรับการแสดงสถานะ (MONITOR)

การอินเตอร์รัพท์จะเกิดขึ้นทุกๆ 0.005 วินาที ดังนั้นโปรแกรมการอินเตอร์รัพท์ต้องออกแบบให้กระทัดรัด เพื่อให้การทำงานรวดเร็ว ซึ่งรวมถึงตำแหน่งของตัวแปรที่ใช้ในโปรแกรม

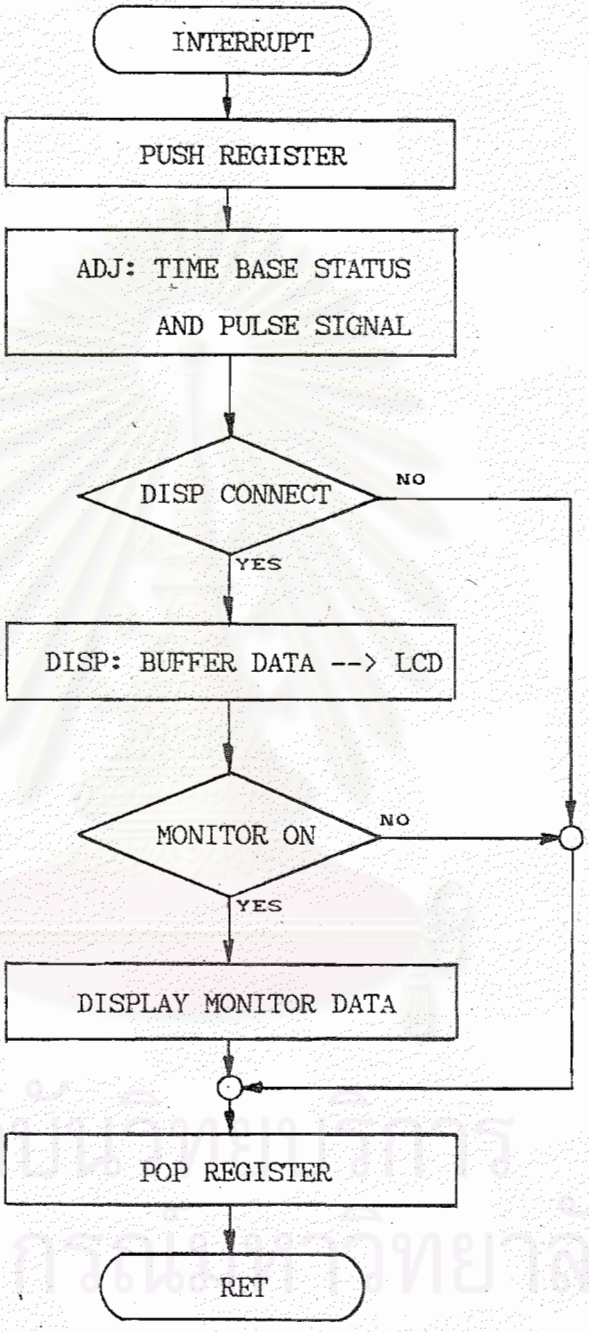
ควรมีการวางตำแหน่งที่เหมาะสมเพื่อให้โปรแกรมทำงานได้เร็ว

การทำงานของฐานเวลา 0.01 วินาทีและ 0.1 วินาที สำหรับคำสั่ง TIMH (High speed timer) และคำสั่ง TIM (Timer) จะทำงานโดยโปรแกรมอินเตอร์รัพท์จะเซ็ทสถานะ F10M ให้เป็น 01 ทุก ๆ 0.01 วินาทีและเซ็ทสถานะ F100M ให้เป็น 01 ทุก ๆ 0.1 วินาที เมื่อซีพียูทำงานตามโปรแกรมขั้นบันได(Ladder) และพบคำสั่ง TIMH หรือ TIM ก็จะตรวจสอบสถานะ F10M/F100M ถ้ามีค่าสถานะเป็น 01 ก็จะทำตามคำสั่งนั้นคือ ลดค่าจำนวนเวลาลง 1 และทำงานตามโปรแกรมขั้นบันไดต่อไปเรื่อยๆ เมื่อทำงานครบหนึ่งรอบการทำงาน (Scantime) เครื่องก็จะเซ็ทสถานะของ F10M และ F100M ให้เป็น 00

การแสดงผลข้อมูลที่ LCD จะทำทุก ๆ 0.01 วินาที โดยจะแสดงผลเพียง 1 ตัวอักษร ดังนั้นถ้าแสดงผลข้อมูลหลาย ๆ ตัว จะต้องคอยสัญญาณอินเตอร์รัพท์หลายครั้งจึงแสดงผลหมดทุกตัวอักษร สำหรับการทำงานในการแสดงข้อมูลของการดูสถานะ (Monitor) จะมีลักษณะเหมือนที่กล่าวมาเพียงแต่จะแสดงผลทุก ๆ 0.05 วินาที



รูปที่ 4.8 แสดงลักษณะของสัญญาณฐานเวลาต่าง ๆ

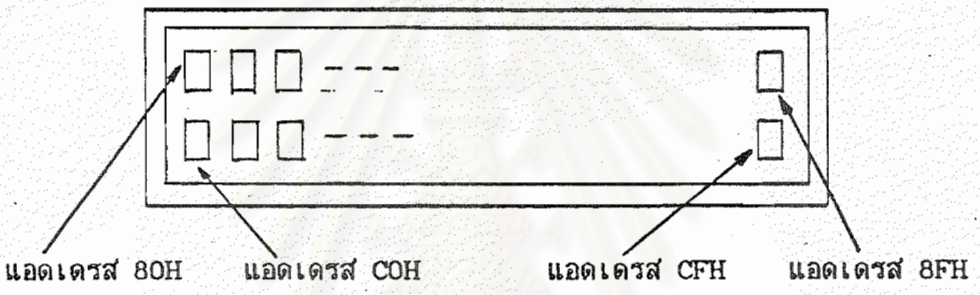


รูปที่ 4.9 แสดงโฟลว์ชาร์ทการทำงานของโปรแกรมอินเตอร์รัปท์

4.1.6 การแสดงผลของ LCD

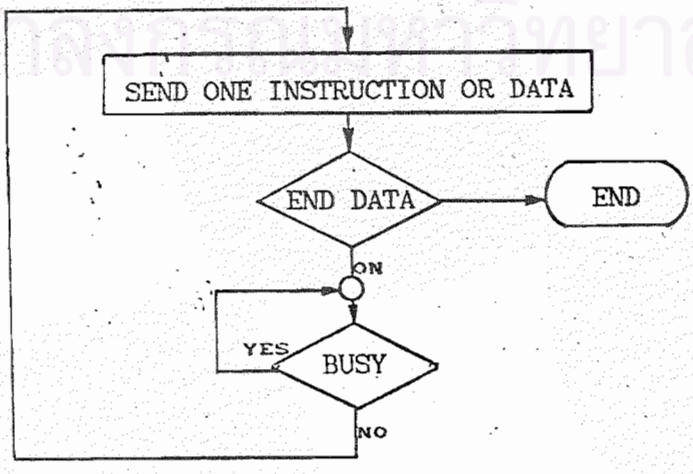
การแสดงผลของเครื่อง PC ที่ออกแบบนี้ใช้จอ LCD ขนาด 16 ตัวอักษร 2 แถว โดยตัวอักษรที่แสดงเป็นดือทซ์เมทริกขนาด 5 x 7 รหัสในการแสดงข้อมูลจะคล้ายรหัสแอสกี้ แต่มีรหัสคำสั่งควบคุมต่างหาก สำหรับการเซ็ทโหมดการทำงานของ LCD เราเซ็ทไว้แล้วในส่วนเริ่มต้นการทำงานของโปรแกรมควบคุม (Initialization) คำสั่งควบคุม LCD ที่ใช้ในการแสดงผลที่สำคัญได้แก่ คำสั่งในการเคลียร์การแสดงผลทั้งหมด และคำสั่งในการกำหนดตำแหน่งสำหรับการแสดงผล การส่งสัญญาณไปยัง LCD จะแบ่งออกเป็น 2 ชนิดคือ

1. การส่งสัญญาณควบคุมการแสดงผล ใช้พอร์ทหมายเลข 80H
2. การส่งสัญญาณรหัสตัวอักษรที่แสดงผล ใช้พอร์ทหมายเลข 81H



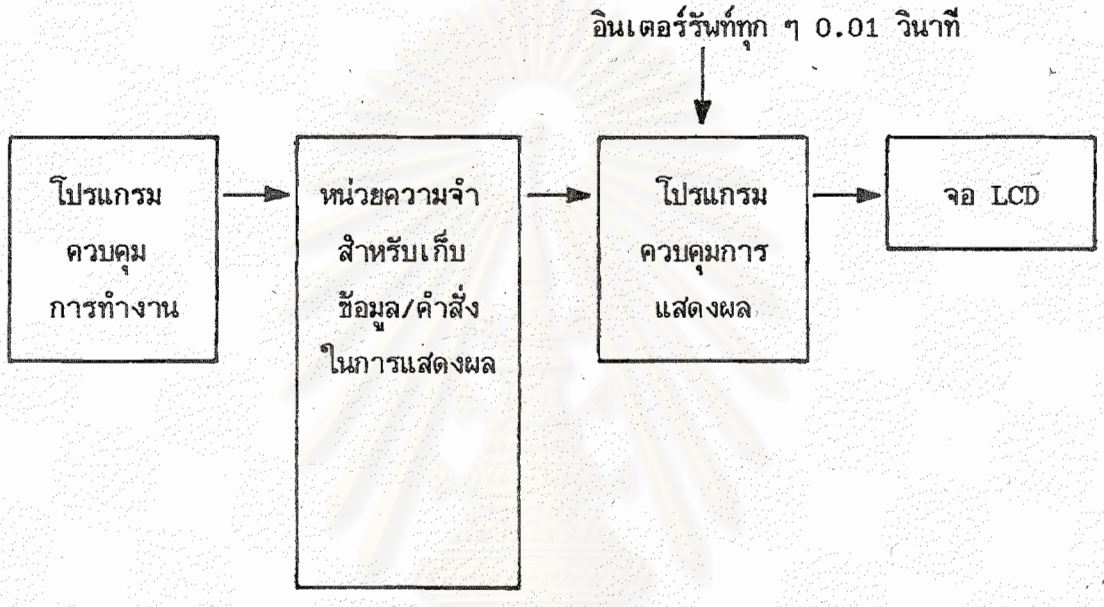
รูปที่ 4.10 แสดงตำแหน่งในการแสดงผลของ LCD

เนื่องจากตัวแสดงผล LCD ไม่สามารถรับข้อมูลในการแสดงผล หรือคำสั่งติดต่อกันได้หลายตัวในการส่งข้อมูลไปให้ตัวแสดงผล LCD 1 ครั้ง ซีพียูต้องคอยตรวจสอบว่าตัว LCD พร้อมที่จะรับข้อมูลตัวต่อไปหรือยัง ซึ่งข้อมูลและคำสั่งแต่ละชนิดมีเวลาในการทำงาน (Delay time) ไม่เท่ากัน เช่น คำสั่งเคลียร์จอ LCD ใช้เวลาในการทำงานประมาณ 1.64 mSec การแสดงข้อมูลตัวอักษรใช้เวลาในการทำงานประมาณ 40 μSec



รูปที่ 4.11 แสดงโฟลว์ชาร์ทการแสดงผลทางจอ LCD แบบปกติ

ถ้าใช้การแสดงผลแบบปกตินี้คือ ให้ซีพียูคอยตรวจสอบสัญญาณ BUSY ของ LCD นี้ จะทำให้ซีพียูเสียเวลาในการคอยมาก ซึ่งจะทำให้เครื่อง PC ทำงานได้ช้าลง วิธีการแสดงผลแบบนี้จึงไม่เหมาะสม ดังนั้นในการแสดงผลของเครื่อง PC นี้ จึงใช้สัญญาณอินเตอร์รัพท์เป็นตัวช่วยในการแสดงผล โดยออกแบบให้มีการส่งคำสั่งหรือข้อมูลไปยังตัว LCD ทุก ๆ 0.01 วินาที เมื่อมีการแสดงผล



รูปที่ 4.12 แสดงลักษณะการทำงานของโปรแกรมควบคุมการแสดงผล

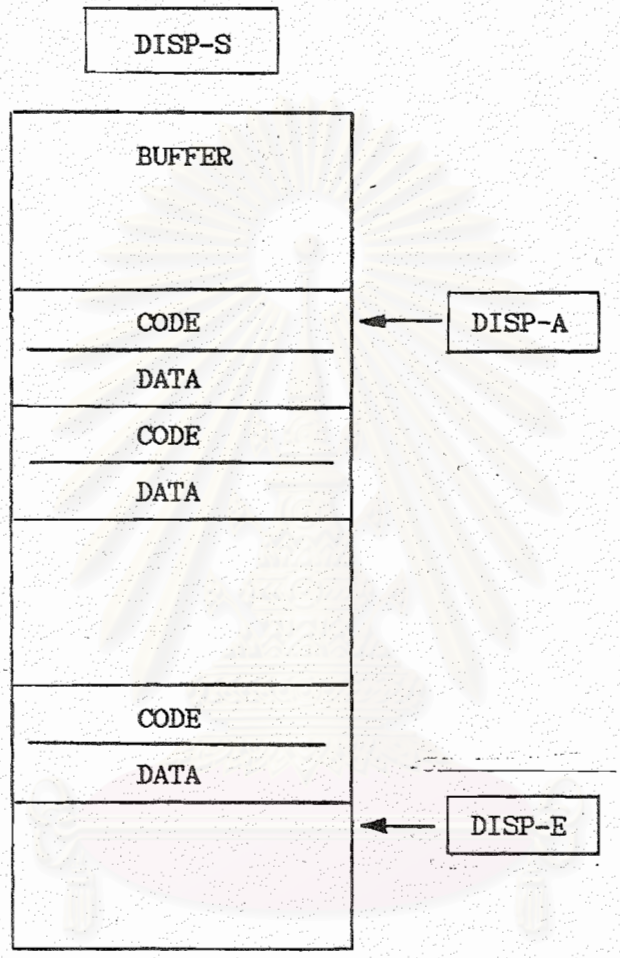
การทำงานของการแสดงผลนี้จะมีการสร้างพื้นที่สำหรับเก็บข้อมูลในการแสดงผล (Display data buffer) และตัวแปรอีก 3 ตัวอีก

DISP-S เป็นตัวเก็บสถานะว่ามีข้อมูลในการแสดงผล หรือไม่
รหัส 01 แทนยังมีข้อมูลในการแสดงผล
รหัส 00 แทนไม่มีข้อมูลในการแสดงผล

DISP-A เก็บตำแหน่งของข้อมูลที่ต้องแสดงผล ตัวแปรนี้จะชี้ตำแหน่งของข้อมูลในบัฟเฟอร์ตัวต่อไปที่ต้องแสดงผล ตัวแปรต้องปรับค่า (Update) ทุกครั้งที่มีการแสดงผล

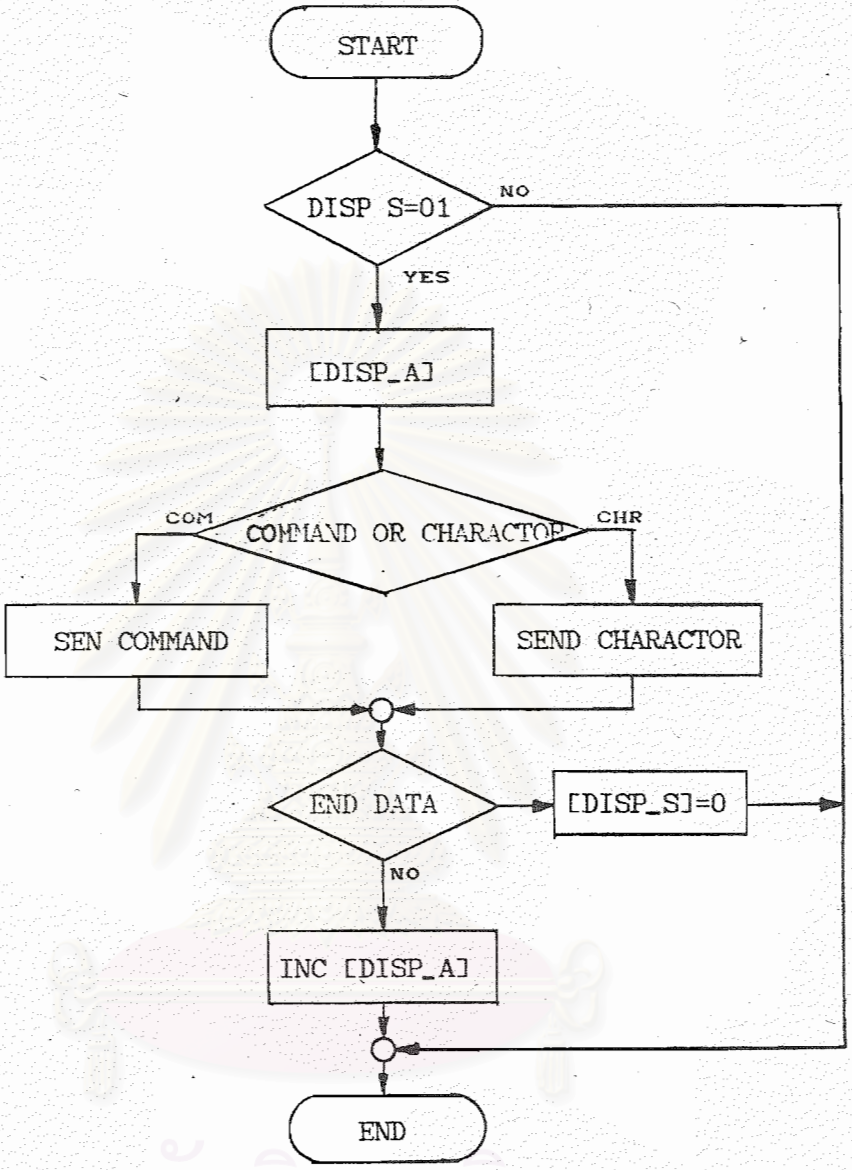
DISP-E เก็บตำแหน่งสุดท้ายของข้อมูลในบัฟเฟอร์ เพื่อใช้ในการเพิ่มข้อมูลที่ต้องแสดงผลลงไป

ข้อมูลที่จะเก็บลงในบัฟเฟอร์ 1 ตัว ประกอบด้วย 2 ส่วนคือ รหัสเพื่อบอกว่าเป็นคำสั่งหรือตัวอักษร และส่วนที่สองคือ ข้อมูล



รูปที่ 4.13 แสดงการจัดเก็บข้อมูลในการแสดงผล

การทำงานของโปรแกรมควบคุมการแสดงผลจะตรวจสอบสถานะ DISP-S ถ้าเท่ากับ 01 ก็จะไปอ่านข้อมูลที่ชี้ตำแหน่งโดย DISP-A แล้วตรวจสอบว่าข้อมูลดังกล่าวเป็นคำสั่งหรือตัวอักษร แล้วส่งข้อมูลไปยัง LCD และตรวจดูว่าหมดข้อมูลหรือยัง ถ้ายังก็เพิ่มค่าของ DISP-A เป็นตำแหน่งตัวถัดไป ถ้าข้อมูลหมดแล้วก็เซ็ทค่า DISP-S เป็น 00



รูปที่ 4.14 แสดงโฟลว์ชาร์ทการทำงานของโปรแกรมควบคุมการแสดงผล

4.1.7 การอ่าน และเขียนอินพุท/เอาท์พุท

การอ่าน และเขียนอินพุท/เอาท์พุทของเครื่อง PC สามารถแบ่งออกได้ 2 วิธี [11] คือวิธี Continuous updating และวิธี Mass input/output copy ทั้ง 2 วิธีมีลักษณะที่แตกต่างกันคือ

1. วิธี Continuous updating

ซีพียูจะอ่าน และเขียนอินพุท/เอาท์พุท เมื่อกำลังทำงานคำสั่งนั้นในโปรแกรมขั้นบันได โดยตรง ซึ่งถ้าเป็นอินพุทก็จะได้ค่าจริงของข้อมูลที่มี delay time ตามอินพุทพอร์ทนั้น ถ้าเป็นเอาท์พุทซีพียูก็จะเขียนเอาท์พุทออกที่เอาท์พุทพอร์ทโดยตรงการทำงานของวิธีนี้มีลักษณะดังต่อไปนี้

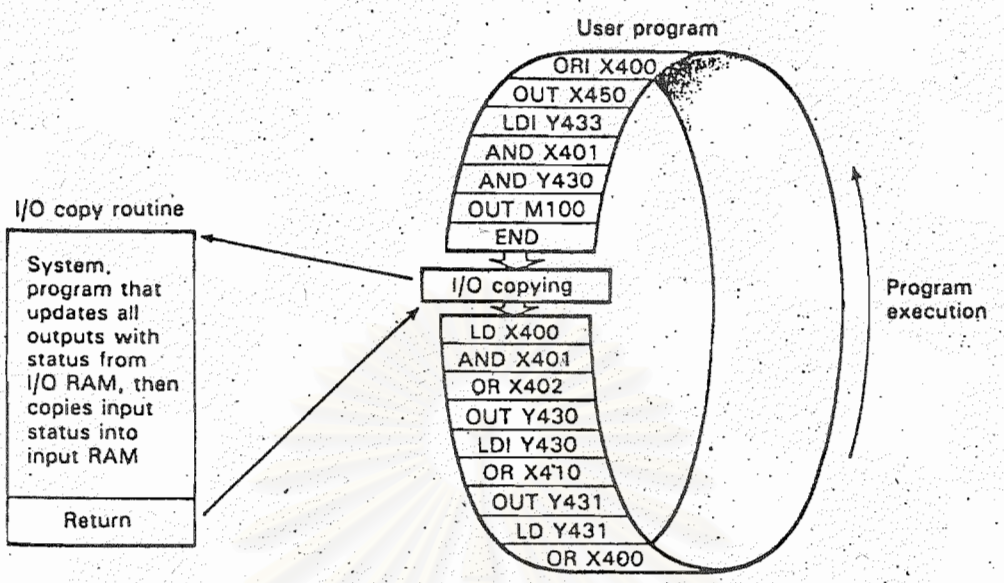
EXECUTE FIRST INSTRUCTION	SCAN I/O	NEXT INSTRUCTION	SCAN I/O	NEXT INSTRUCTION	SCAN I/O

2. วิธี Mass input/output copy

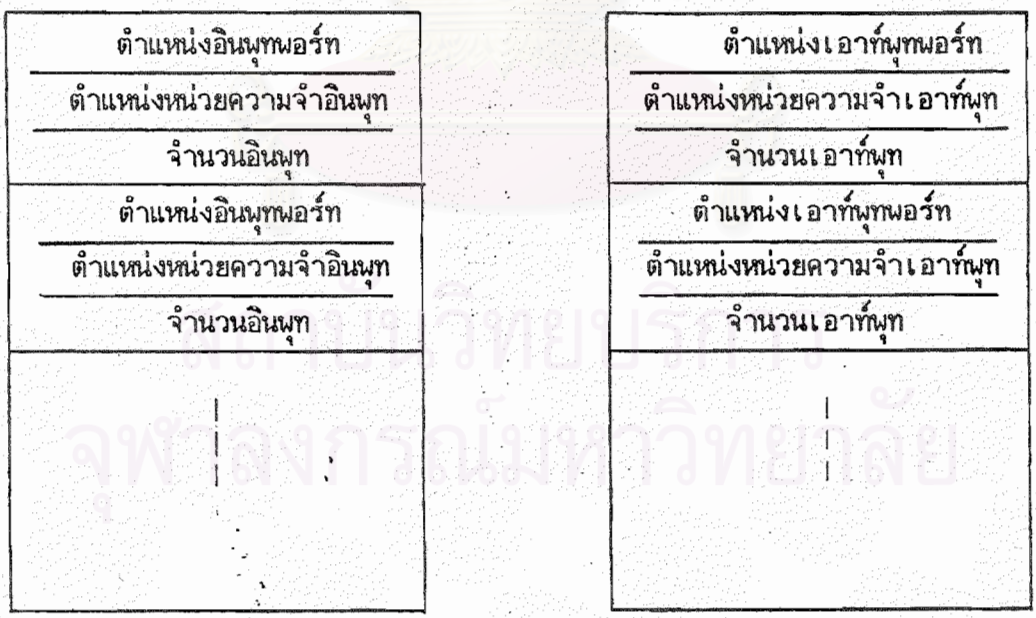
ในเครื่อง PC ขนาดใหญ่ที่มีจำนวนอินพุท/เอาท์พุทมาก และมีโปรแกรมขั้นบันไดยาว ในสถานะของอินพุท ขณะที่ทำงานต้นโปรแกรมกับสถานะอินพุทเมื่อทำงานปลายโปรแกรม อาจมีค่าแตกต่างกัน ซึ่งอาจมีผลทำให้การทำงานของโปรแกรมควบคุมไม่ถูกต้องได้

การอ่าน และเขียนอินพุท/เอาท์พุทวิธีนี้ ดูรูปที่4.15 จะสร้างตารางในการเก็บสถานะของอินพุท/เอาท์พุทในหน่วยความจำขึ้นมา และในการทำงานของซีพียูตามคำสั่งโปรแกรมขั้นบันไดก็จะติดต่อกับตารางหน่วยความจำที่เก็บข้อมูลแทน เมื่อทำงานครบหนึ่งรอบทำงาน (Scantime) ก็จะอ่านข้อมูลจากอินพุทพอร์ทต่าง ๆ มาเก็บในตาราง และนำค่าเอาท์พุทที่เก็บในตารางส่งไปยังเอาท์พุทพอร์ท ซึ่งจะมีการคงสถานะ (Latch) ของข้อมูลไว้จนกว่าจะมีการเขียนครั้งใหม่ วิธีนี้จะทำให้ค่าของอินพุท/เอาท์พุทมีค่าเหมือนกันตลอดการทำงานหนึ่งรอบการทำงาน

ในเครื่อง PC ที่ออกแบบจะใช้วิธีอ่านและเขียนอินพุทแบบ Mass input/output copy การสร้างตารางในการเก็บตำแหน่งต่าง ๆ ของอินพุท/เอาท์พุท จะแยกเป็น 2 ตาราง คือ ตารางสำหรับอินพุทพอร์ท และตารางสำหรับเอาท์พุทพอร์ท ตารางเหล่านี้จะถูกสร้างตอนเริ่มต้นขณะ Initialization



รูปที่ 4.15 แสดงการอ่าน และเขียนแบบ Mass input/output copy



รูปที่ 4.16 แสดงตารางในการเก็บอินพุท/เอาท์พุทพอร์ท

4.2 โครงสร้าง และการจัดเก็บข้อมูล

ข้อมูลที่ใช้ในเครื่อง PC มีหลายอย่างเช่น ข้อมูลอินพุท/เอาท์พุท ข้อมูลของตัวตั้งเวลาหรือตัวนับ และข้อมูลที่ใช้ในการจัดการข้อมูล การจัดแบ่งข้อมูลในเครื่อง PC นี้แบ่งออกได้หลายชนิดคือ

1. I/O Relay
2. Auxilary Relay
3. Holding Relay
4. Data memory
5. Timer/Counter Data
6. Temporary Relay
7. Timer/Counter Contact

I/O Relay หมายถึง หน่วยความจำที่เก็บสถานะของอินพุทและเอาท์พุทในการทำงานใน 1 รอบการทำงานของเครื่อง PC จะมีการอ่านค่าจากโมดูลอินพุทไปเก็บลงใน I/O Relay ตามหมายเลขที่กำหนด และจะมีการอ่านค่าของ I/O Relay ส่งไปยังโมดูลเอาท์พุท จำนวนของ I/O Relay ในเครื่องที่ออกแบบมี 512 ตำแหน่ง ในกรณีที่ I/O Relay ใช้งานไม่ทั้งหมด ส่วนที่เหลือสามารถใช้งานเป็น Auxilary relay ได้

Auxilary relay หมายถึง รีเลย์ช่วยในการทำงานต่างกับ I/O relay ตรงที่ไม่สามารถคิดต่อกับอุปกรณ์ภายนอกได้โดยตรง ถ้าต้องการติดต่อกับอุปกรณ์ภายนอกต้องติดต่อผ่าน I/O Relay จำนวนของ I/O Relay ในเครื่องมีทั้งหมด 512 ตำแหน่ง แต่มีบางส่วนประมาณ 48 ตำแหน่งที่นำไปใช้เป็นรีเลย์ช่วยพิเศษ (Special auxilary relay) ถ้า I/O relay มีจำนวนไม่เพียงพอ เครื่อง PC ที่ออกแบบนี้ผู้ใช้สามารถนำ Auxilary relay ไปใช้แทนได้ การใช้งานต่าง ๆ ของ Auxilary relay จะเหมือนกับ I/O relay โดยปกติเมื่อมีการเปิดเครื่องหรือเปลี่ยนจากโหมดโปรแกรมไปเป็นโหมดทำงาน ข้อมูลต่าง ๆ ใน Auxilary relay จะถูกเคลียร์ก่อนเสมอ

Holding relay หมายถึง Auxilary relay แบบ Rententive การทำงานจะเหมือน Auxilary relay ทุกอย่างแต่ข้อมูลของ Holding Relay จะไม่ถูกเคลียร์เมื่อเปิดเครื่องหรือเปลี่ยนโหมดจากโหมดโปรแกรมเป็นโหมดการทำงาน Holding relay ในเครื่องถูกออกแบบให้มีทั้งหมด 512 ตำแหน่ง และในการใช้งานจะเรียกชื่อย่อว่า HR

Data memory เป็นหน่วยความจำสำหรับเก็บข้อมูลที่ใช้ในคำสั่งเกี่ยวกับการจัดการข้อมูล หน่วยความจำนี้มีขนาด 16 บิทแบ่งออกเป็นแบบ Rententive และแบบ Non-rententive

อย่างละ 256 ตำแหน่ง

Timer/counter data หมายถึงหน่วยความจำที่ใช้เก็บข้อมูลของคำสั่ง TIM หรือคำสั่ง CNT การเก็บข้อมูลจะมีขนาด 16 บิท ต่อหนึ่งตำแหน่ง ในเครื่อง PC นี้มี Timer/counter data ทั้งหมด 128 ตำแหน่ง

Temporary relay หมายถึงรีเลย์ที่ไว้เก็บค่าในการประมวลผลชั่วคราว ซึ่งจะใช้ได้เฉพาะคำสั่ง LD และ OUT เท่านั้น ในเครื่องที่ออกแบบมี Temperary relay ทั้งหมด 8 ตำแหน่ง ในการใช้งานจะเรียกชื่อย่อว่า TR

Timer/counter contact เป็นรีเลย์ที่เก็บผลลัพธ์ของคำสั่งเกี่ยวกับตัวตั้งเวลาและตัวนับ ซึ่งจะมีค่าเป็น ON หรือ OFF ในเครื่องมี TIM/CNT รวมกัน 128 ตัว แต่การเก็บข้อมูลเอาท์พุท จะแยกเป็นของ Timer 128 ตัว ของ Counter 128 ตัว เพื่อความสะดวกในการตรวจสอบของโปรแกรมควบคุม ซึ่งตำแหน่งที่เก็บข้อมูลจริงนั้นรวมกันทั้ง Timer และ Counter ไม่เกิน 128 ตำแหน่ง

ลักษณะของข้อมูลที่เก็บในเครื่องสามารถแบ่งออกได้เป็น 2 ชนิดคือ

1. ข้อมูลที่เป็นสถานะของอินพุท/เอาท์พุท

ข้อมูลชนิดนี้ได้แก่ I/O relay, Auxilary relay, Holding relay ข้อมูลเหล่านี้จะมีเพียง 2 สถานะคือ สถานะ ON หรือ OFF ในการจัดเก็บจะใช้หน่วยความจำ 1 ไบท์ ในการเก็บข้อมูลหนึ่งตำแหน่ง โดยให้ข้อมูล 00H แทนสถานะ OFF และข้อมูล 80H แทนสถานะ ON

1	0	0	0	0	0	0	0	สถานะ ON

0	0	0	0	0	0	0	0	สถานะ OFF

เพื่อความเหมาะสมในการเคลื่อนย้าย และในการจัดการข้อมูลต่างๆ ข้อมูลชนิดนี้ถูกจัดแบ่งออกเป็นกลุ่ม กลุ่มละ 16 ตำแหน่ง และเรียกข้อมูลแต่ละกลุ่มว่าแชลแนล (Channel) และในการอ้างอิงตำแหน่งของข้อมูลชนิดนี้จะมี 2 แบบคือ

1. อ้างอิงตำแหน่งข้อมูล จะใช้ตัวเลขฐานสิบขนาด 4 หลักในการอ้างอิงตำแหน่งของข้อมูลนั้น โดยใช้ตัวเลข 2 หลักแรกแทนหมายเลขแชลแนล และตัวเลข 2 ตำแหน่งหลังแทนตำแหน่งของข้อมูล ซึ่งมีค่าระหว่าง 00 ถึง 15

ตำแหน่งข้อมูล : XX XX

หมายเลขแชลแนล — หมายเลขตำแหน่ง (00-15)

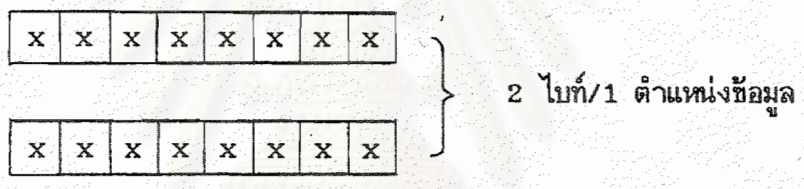
2. อ้างอิงตำแหน่งของกลุ่มข้อมูลจะใช้ตัวเลขฐานสิบขนาด 2 หลักในการอ้างอิงกลุ่มของข้อมูล โดยจะเรียกว่าหมายเลขแชลแนล ซึ่งหมายถึงกลุ่มข้อมูล 16 ตำแหน่ง

ตำแหน่งกลุ่มข้อมูล : xx
└── หมายเลขแชลแนล

สำหรับการบอกชนิดต่าง ๆ ของข้อมูลทั้ง 2 แบบ จะระบุชนิดของข้อมูลลงไว้ด้านหน้าของตำแหน่งข้อมูล และด้านหน้าของตำแหน่งกลุ่มข้อมูล

2. ข้อมูลที่เก็บค่าตัวเลข

ข้อมูลชนิดนี้ได้แก่ Data memory และ Timer/counter data ข้อมูลชนิดนี้มีขนาด 16 บิท ซึ่งข้อมูลหนึ่งตำแหน่งจะใช้หน่วยความจำในการเก็บ 2 ไบท์



ค่าตัวเลขของข้อมูลชนิดนี้อาจเป็นเลข BCD ขนาด 16 บิท หรือเลข Binary ขนาด 16 บิทก็ได้ โดยข้อมูลบิท 0 จะเก็บที่บิทต่ำของไบท์แอดเดรสต่ำ และบิท 15 จะเก็บที่บิทสูงของไบท์แอดเดรสสูง

การอ้างอิงตำแหน่งข้อมูลจะใช้ตัวเลขฐานสิบขนาด 3 หลักในการอ้างอิง โดยมีชนิดของข้อมูลระบุไว้ การอ้างอิงข้อมูลที่บิทต่าง ๆ ไม่สามารถทำได้โดยตรง แต่อาจใช้วิธีย้ายข้อมูลไปยังรีเลย์ช่วย (Auxilary relay) แล้วอ้างอิงตำแหน่งของรีเลย์ช่วยแทน

DM xxx
└── หมายเลขตำแหน่ง

การอ้างอิงตำแหน่งของข้อมูลชนิดต่างๆ ที่กล่าวมาแล้วข้างต้นสามารถสรุปเป็นตารางได้ดังนี้

	จำนวน	อ้างอิงตำแหน่ง	อ้างอิงแชลแนล
I/O Relay	512	0000 - 3115	00 - 31
Auxilary relay	512	3200 - 6315	32 - 63
Holding relay	512	HR 0000 - HR 3115	HR 00 - HR 31
Data memory			
- Retentive	256	-	DM 000 - DM 255
- Non retentive	256	-	DM 256 - DM 512
Timer/counter	128	-	TIM/CNT 000 - 127
Temporary	8	TR0 - TR7	-

ตารางที่ 4.2 แสดงชนิดของข้อมูลต่าง ๆ

Name	NO.	Relay number															
		0000 - 3115															
		00CH	01CH	02CH	03CH	04CH	05CH	06CH	07CH	08CH	09CH	10CH	11CH	12CH	13CH	14CH	15CH
		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
		01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
		02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
		03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
		04	04	04	04	04	04	04	04	04	04	04	04	04	04	04	04
		05	05	05	05	05	05	05	05	05	05	05	05	05	05	05	05
		06	06	06	06	06	06	06	06	06	06	06	06	06	06	06	06
		07	07	07	07	07	07	07	07	07	07	07	07	07	07	07	07
		08	08	08	08	08	08	08	08	08	08	08	08	08	08	08	08
		09	09	09	09	09	09	09	09	09	09	09	09	09	09	09	09
		10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
		11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
		12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
		13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
Input/		14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
Output	512	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
relay		16CH	17CH	18CH	19CH	20CH	21CH	22CH	23CH	24CH	25CH	26CH	27CH	28CH	29CH	30CH	31CH
		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
		01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
		02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
		03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
		04	04	04	04	04	04	04	04	04	04	04	04	04	04	04	04
		05	05	05	05	05	05	05	05	05	05	05	05	05	05	05	05
		06	06	06	06	06	06	06	06	06	06	06	06	06	06	06	06
		07	07	07	07	07	07	07	07	07	07	07	07	07	07	07	07
		08	08	08	08	08	08	08	08	08	08	08	08	08	08	08	08
		09	09	09	09	09	09	09	09	09	09	09	09	09	09	09	09
		10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
		11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
		12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
		13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
		14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
		15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15

ตารางที่ 4.3 แสดงการเก็บข้อมูลของ I/O Relay

Name	NO.	Relay number															
		3200 - 6015															
		32CH	33CH	34CH	35CH	36CH	37CH	38CH	39CH	40CH	41CH	42CH	43CH	44CH	45CH	46CH	47CH
		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
		01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
		02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
		03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
		04	04	04	04	04	04	04	04	04	04	04	04	04	04	04	04
		05	05	05	05	05	05	05	05	05	05	05	05	05	05	05	05
		06	06	06	06	06	06	06	06	06	06	06	06	06	06	06	06
		07	07	07	07	07	07	07	07	07	07	07	07	07	07	07	07
		08	08	08	08	08	08	08	08	08	08	08	08	08	08	08	08
		09	09	09	09	09	09	09	09	09	09	09	09	09	09	09	09
		10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
		11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
		12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
		13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
		14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
Auxiliary relay	464	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
		48CH	49CH	50CH	51CH	52CH	53CH	54CH	55CH	56CH	57CH	58CH	59CH	60CH			
		00	00	00	00	00	00	00	00	00	00	00	00	00			
		01	01	01	01	01	01	01	01	01	01	01	01	01			
		02	02	02	02	02	02	02	02	02	02	02	02	02			
		03	03	03	03	03	03	03	03	03	03	03	03	03			
		04	04	04	04	04	04	04	04	04	04	04	04	04			
		05	05	05	05	05	05	05	05	05	05	05	05	05			
		06	06	06	06	06	06	06	06	06	06	06	06	06			
		07	07	07	07	07	07	07	07	07	07	07	07	07			
		08	08	08	08	08	08	08	08	08	08	08	08	08			
		09	09	09	09	09	09	09	09	09	09	09	09	09			
		10	10	10	10	10	10	10	10	10	10	10	10	10			
		11	11	11	11	11	11	11	11	11	11	11	11	11			
		12	12	12	12	12	12	12	12	12	12	12	12	12			
		13	13	13	13	13	13	13	13	13	13	13	13	13			
		14	14	14	14	14	14	14	14	14	14	14	14	14			
		15	15	15	15	15	15	15	15	15	15	15	15	15			

ตารางที่ 4.4 แสดงการเก็บข้อมูลของ Auxiliary relay

CH NO.	Relay No.	หน้าที่การทำงาน
61	6100	ถ้า ON เอาท์พุททั้งหมดจะปิด (OFF)
	6101	ถ้า ON ข้อมูลของ I/O และ Auxiliary relay จะไม่ถูกลบ
	6200	ON เมื่อเกิด Alarm
	6201	ON เมื่อเกิด Error
	6202	ON เมื่อแบตตอรี่ในเครื่องไม่ปกติ
	6203	จะ ON หนึ่งรอบการทำงานแรกเท่านั้น
	6204	ON ตลอด
	6205	OFF ตลอด
62	6206	-
	6207	-
	6208	รหัสแสดงความผิดพลาด
	6209	
	6210	
	6211	
	6212	
	6213	
	6214	
	6215	
	6300	ใช้เป็นสัญญาณนาฬิกา 0.1 วินาที
	6301	ใช้เป็นสัญญาณนาฬิกา 0.2 วินาที
	6302	ใช้เป็นสัญญาณนาฬิกา 1.0 วินาที
63	6303	ON เมื่อข้อมูล BCD มีข้อผิดพลาด
	6304	ON เมื่อเกิดตัวทดจากการคำนวณ (CY)
	6305	ON เมื่อการเปรียบเทียบได้ผลเป็นมากกว่า (>)
	6306	ON เมื่อการเปรียบเทียบได้ผลเป็นเท่ากับ (=)
	6307	ON เมื่อการเปรียบเทียบได้ผลเป็นน้อยกว่า (<)
	0	เป็นรีเลย์ชั่วคราวตัวที่ 0
	1	เป็นรีเลย์ชั่วคราวตัวที่ 1
	2	เป็นรีเลย์ชั่วคราวตัวที่ 2
Tempo	3	เป็นรีเลย์ชั่วคราวตัวที่ 3
rary	4	เป็นรีเลย์ชั่วคราวตัวที่ 4
relay	5	เป็นรีเลย์ชั่วคราวตัวที่ 5
	6	เป็นรีเลย์ชั่วคราวตัวที่ 6
(TR)	7	เป็นรีเลย์ชั่วคราวตัวที่ 7

ตาราง 4.5 แสดงรีเลย์พิเศษ

Name	NO.	Relay number															
		HR 0000 - 3115															
		00CH	01CH	02CH	03CH	04CH	05CH	06CH	07CH	08CH	09CH	10CH	11CH	12CH	13CH	14CH	15CH
		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
		01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
		02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
		03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
		04	04	04	04	04	04	04	04	04	04	04	04	04	04	04	04
		05	05	05	05	05	05	05	05	05	05	05	05	05	05	05	05
		06	06	06	06	06	06	06	06	06	06	06	06	06	06	06	06
		07	07	07	07	07	07	07	07	07	07	07	07	07	07	07	07
		08	08	08	08	08	08	08	08	08	08	08	08	08	08	08	08
		09	09	09	09	09	09	09	09	09	09	09	09	09	09	09	09
		10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
		11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
		12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
		13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
		14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
Holding	512	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
relay		16CH	17CH	18CH	19CH	20CH	21CH	22CH	23CH	24CH	25CH	26CH	27CH	28CH	29CH	30CH	31CH
		00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
		01	01	01	01	01	01	01	01	01	01	01	01	01	01	01	01
		02	02	02	02	02	02	02	02	02	02	02	02	02	02	02	02
		03	03	03	03	03	03	03	03	03	03	03	03	03	03	03	03
		04	04	04	04	04	04	04	04	04	04	04	04	04	04	04	04
		05	05	05	05	05	05	05	05	05	05	05	05	05	05	05	05
		06	06	06	06	06	06	06	06	06	06	06	06	06	06	06	06
		07	07	07	07	07	07	07	07	07	07	07	07	07	07	07	07
		08	08	08	08	08	08	08	08	08	08	08	08	08	08	08	08
		09	09	09	09	09	09	09	09	09	09	09	09	09	09	09	09
		10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
		11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
		12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
		13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
		14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
		15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15

ตารางที่ 4.6 แสดงการเก็บข้อมูลของ Holding relay

Name	NO.	Data memory number															
		DM 000 - 511															
		000	010	020	030	040	050	060	070	080	090	100	110	120	130	140	150
		001	011	021	031	041	051	061	071	081	091	101	111	121	131	141	151
		002	012	022	032	042	052	062	072	082	092	102	112	122	132	142	152
		003	013	023	033	043	053	063	073	083	093	103	113	123	133	143	153
		004	014	024	034	044	054	064	074	084	094	104	114	124	134	144	154
		005	015	025	035	045	055	065	075	085	095	105	115	125	135	145	155
		006	016	026	036	046	056	066	076	086	096	106	116	126	136	146	156
		007	017	027	037	047	057	067	077	087	097	107	117	127	137	147	157
		008	018	028	038	048	058	068	078	088	098	108	118	128	138	148	158
		009	019	029	039	049	059	069	079	089	099	109	119	129	139	149	159
		160	170	180	190	200	210	220	230	240	250	260	270	280	290	300	310
		161	171	181	191	201	211	221	231	241	251	261	271	281	291	301	311
		162	172	182	192	202	212	222	232	242	252	262	272	282	292	302	312
		163	173	183	193	203	213	223	233	243	253	263	273	283	293	303	313
		164	174	184	194	204	214	224	234	244	254	264	274	284	294	304	314
		165	175	185	195	205	215	225	235	245	255	265	275	285	295	305	315
		166	176	186	196	206	216	226	236	246	256	266	276	286	296	306	316
		167	177	187	197	207	217	227	237	247	257	267	277	287	297	307	317
		168	178	188	198	208	218	228	238	248	258	268	278	288	298	308	318
Data	512 X	169	179	189	199	209	219	229	239	249	259	269	279	289	299	309	319
memory	16 bit	320	330	340	350	360	370	380	390	400	410	420	430	440	450	460	470
		321	331	341	351	361	371	381	391	401	411	421	431	441	451	461	471
		322	332	342	352	362	372	382	392	402	412	422	432	442	452	462	472
		323	333	343	353	363	373	383	393	403	413	423	433	443	453	463	473
		324	334	344	354	364	374	384	394	404	414	424	434	444	454	464	474
		325	335	345	355	365	375	385	395	405	415	425	435	445	455	465	475
		326	336	346	356	366	376	386	396	406	416	426	436	446	456	466	476
		327	337	347	357	367	377	387	397	407	417	427	437	447	457	467	477
		328	338	348	358	368	378	388	398	408	418	428	438	448	458	468	478
		329	339	349	359	369	379	389	399	409	419	429	439	449	459	469	479
		480	490	500	510												
		481	491	501	511												
		482	492	502													
		483	493	503													
		484	494	504													
		485	495	505													
		486	496	506													
		487	497	507													
		488	498	508													
		489	499	509													

ตารางที่ 4.7 แสดงการเก็บข้อมูลของ Data memory

Name	NO.	Timer/Counter number												
		TIM/CNT 000 - 127												
Timer/ Counter	128	000	010	020	030	040	050	060	070	080	090	100	110	120
		001	011	021	031	041	051	061	071	081	091	101	111	121
		002	012	022	032	042	052	062	072	082	092	102	112	122
		003	013	023	033	043	053	063	073	083	093	103	113	123
		004	014	024	034	044	054	064	074	084	094	104	114	124
		005	015	025	035	045	055	065	075	085	095	105	115	125
		006	016	026	036	046	056	066	076	086	096	106	116	126
		007	017	027	037	047	057	067	077	087	097	107	117	127
		008	018	028	038	048	058	068	078	088	098	108	118	
		009	019	029	039	049	059	069	079	089	099	109	119	

ตารางที่ 4.8 แสดงการเก็บข้อมูลของ Timer/counter

การจัดตำแหน่งของหน่วยความจำเพื่อเก็บข้อมูลชนิดต่างๆ ต้องพิจารณาถึงความเหมาะสมและความสะดวกในการเขียนโปรแกรมควบคุมด้วย เครื่องที่ออกแบบนี้ถูกจัดตำแหน่งข้อมูลให้ลงตัวเป็นหน้า (Page) โดยมีขนาดหน้าเท่ากับ 128 ไบท์ ซึ่งมีตำแหน่งในการเก็บข้อมูลต่าง ๆ ดังนี้

ชนิดข้อมูล	ขนาด	แอดเดรสที่เก็บ
I/O Relay	512 : 0000-3115	C000-C1FF
Auxiliary relay	512 : 3200-6315	C200-C3FF
Holding relay	512 : HR0000-3115	C400-C5FF
TIM/CNT relay	127 : TIM000-127	C600-C6FF
Data memory	512 : DM 000-511	C000-CFFF

ตารางที่ 4.9 แสดงตำแหน่งหน่วยความจำที่เก็บข้อมูลต่าง ๆ

ในการทำงานของคำสั่งขั้นบันไดบางคำสั่งจำเป็นต้องใช้หน่วยความจำในการเก็บสถานะในการทำงานบางอย่าง สถานะเหล่านี้ผู้ใช้ไม่จำเป็นต้องรู้ แต่จะใช้เฉพาะในการทำงานของคำสั่งนั้น คำสั่งที่ต้องใช้หน่วยความจำในการเก็บสถานะบางอย่างได้แก่ คำสั่ง SFT, DIFU, DIFD และคำสั่งเกี่ยวกับ Timer หรือ Counter การเก็บของสถานะเหล่านี้จะใช้ลักษณะเป็นหน้า (Page) เช่นเดียวกันเพื่อสะดวกในการอ้างอิง

TYPE	MEMORY ADDRESS																NO.
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
I/O 0000-3115	1	1	0	0	0	0	0	x	x	x	x	x	x	x	x	x	512
AUX 3200-6315	1	1	0	0	0	0	1	x	x	x	x	x	x	x	x	x	512
HR 0000-3115	1	1	0	0	0	1	0	x	x	x	x	x	x	x	x	x	512
TIM 000-127	1	1	0	0	0	1	1	0	x	x	x	x	x	x	x	0	128
CNT 000-127	1	1	0	0	0	1	1	0	x	x	x	x	x	x	x	1	128
DIFU/DIFD	1	1	0	0	0	1	1	1	0	x	x	x	x	x	x	x	128
SFT	1	1	0	0	0	1	1	1	1	x	x	x	x	x	x	x	128
C/T, DSET	1	1	0	0	1	0	0	0	x	x	x	x	x	x	x	0	128
PUL-ST	1	1	0	0	1	0	0	1	x	x	x	x	x	x	x	0	128
COUNT VALUE	1	1	0	0	1	0	1	0	x	x	x	x	x	x	x	0	128
PRESET VALUE	1	1	0	0	1	0	1	1	x	x	x	x	x	x	x	0	128
DM 000-511	1	1	0	0	1	1	x	x	x	x	x	x	x	x	x	x	512
SYSTEM RESERVE	1	1	0	1	x	x	x	x	x	x	x	x	x	x	x	x	4096

ตารางที่ 4.10 แสดงแอดเดรสของหน่วยความจำในการเก็บข้อมูล

4.3 การจัดเก็บคำสั่ง Ladder

การเขียนโปรแกรมในเครื่อง PC ของใช้เขียนจะเป็นภาษาขั้นบันได (Ladder)และฟังก์ชั่นบล๊อก (Function block) โดยการป้อนโปรแกรมจะแบ่งออกเป็นสองส่วนคือ นิวโมนิคและโอเปอร์แรนด์ บางคำสั่งอาจมีโอเปอร์แรนด์หลายตัว โปรแกรมที่ผู้ใช้เขียนจะถูกแปลงให้เป็นคำสั่งภาษาเครื่องที่ซีพียูสามารถทำงานได้ และเก็บลงในหน่วยความจำสำหรับเก็บโปรแกรม (User program area) สำหรับการแปลคำสั่งขั้นบันไดมาเป็นภาษาเครื่องนี้ จะเป็นการทำงานของโปรแกรมควบคุมการกดคีย์บอร์ด เมื่อมีการกดคำสั่งเขียน (write) ก็จะมีการตรวจสอบความถูกต้องของคำสั่งขั้นบันได และแปลงเป็นภาษาเครื่องเก็บลงหน่วยความจำ

คำสั่งที่ใช้ในการเขียนโปรแกรมขั้นบันไดของเครื่อง PC นี้ จะจำแนกออกได้เป็น 2 ชนิดคือ

1. คำสั่งเบื้องต้น เป็นคำสั่งเกี่ยวกับการอ่าน/เขียน และทำลอจิกต่าง ๆ ของรีเลย์ ซึ่งมีทั้งหมด 10 คำสั่งคือ

LD	LD-NOT	OR
OR-NOT	AND	AND-NOT
AND-LD	OR-LD	OUT
OUT-NOT		

2. คำสั่งฟังก์ชั่นเป็นคำสั่งในการจัดการข้อมูลต่าง ๆ รวมทั้งคำสั่งในการควบคุมการทำงานของโปรแกรมขั้นบันได ประกอบด้วยคำสั่งทั้งหมด 54 คำสั่งคือ

NOP (FUN00)	END (FUN01)	IL (FUN02)
ILC (FUN03)	JMP (FUN04)	JME (FUN05)
F06 (FUN06)	F07 (FUN07)	TIM (FUN08)
CNT (FUN09)	SFT (FUN10)	KEEP(FUN11)
CNTR(FUN12)	DIFU(FUN13)	DIFD(FUN14)
TIMH(FUN15)	WSFT(FUN16)	MOVL(FUN17)
MOVH(FUN18)	F19 (FUN19)	CMP (FUN20)
MOV (FUN21)	MVN (FUN22)	BIN (FUN23)
BCD (FUN24)	ASL (FUN25)	ASR (FUN26)
ROL (FUN27	ROR (FUN28)	COM (FUN29)
ADD (FUN30)	SUB (FUN31)	MUL (FUN32)
DIV (FUN33)	ANDW(FUN34)	ORW (FUN35)

XORW(FUN36)	XNRW(FUN37)	INC (FUN38)
DEC (FUN39)	STC (FUN40)	CLC (FUN41)
BKM (FUN42)	BKS (FUN43)	DAEX(FUN44)
ODSL(FUN45)	ODSR(FUN46)	DECO(FUN47)
ENCO(FUN48)	7SEG(FUN49)	F50 (FUN50)
F51 (FUN51)	F52 (FUN52)	F53 (FUN53)

การจัดเก็บคำสั่งเบื้องต้นจะมีลักษณะเป็นเอ๊กซ์คิ้วโปรแกรมคือ เป็นคำสั่งภาษาเครื่องที่ซีพียูสามารถทำงานได้เลย โดยค่าโอเปอร์แรนด์ของคำสั่งขั้นบันไดจะถูกแปลง และเก็บไว้ในเอ๊กซ์คิ้วโปรแกรมแล้ว เช่น

คำสั่ง LD 0001

จะมีการเก็บเป็นภาษาเครื่องคือ

SRL	:	CB 3F
LD HL, C001	:	21 01 C0
OR (HL)	:	B6

คำสั่ง LD 0001 = CB 3F 21 01 C0 B6

จะเห็นว่าคำสั่งขั้นบันได LD 0001 ถูกแปลงเป็นภาษาเครื่องมีความยาว 6 ไบท์ และโอเปอร์แรนด์ 0001 จะถูกแปลงเป็นแอดเดรสของหน่วยความจำที่เก็บข้อมูลตำแหน่ง 0001 ซึ่งมีค่าเท่ากับ 0C001H

สำหรับคำสั่งฟังก์ชั่นต่าง ๆ นั้น แบ่งออกได้ 2 อย่างคือ คำสั่งที่มีโอเปอร์แรนด์ และคำสั่งที่ไม่มีโอเปอร์แรนด์ การจัดเก็บคำสั่งฟังก์ชั่นนี้จะเก็บในลักษณะเป็นคำสั่งการเรียกไปยังตารางของสับรูทีน(Subroutine) ตามด้วยชนิดของโอเปอร์แรนด์และโอเปอร์แรนด์ที่ 1, 2 และ 3 ตามลำดับ การดูว่าคำสั่งที่เก็บนี้เป็นฟังก์ชั่นอะไร สามารถหาได้จากค่าของตำแหน่งในตารางที่สับรูทีน (Subroutine) ต้องไปทำงาน โดยตารางนี้จะเก็บไว้ที่ตำแหน่ง 0300H ถึง 0400H

ตัวอย่าง คำสั่ง MOV (21) -

\- 02

\- HR 05

จะมีการเก็บคำสั่งเป็น

CALL 033F : CD 03 3F

Data type : 00

S (Source) : 20 C0

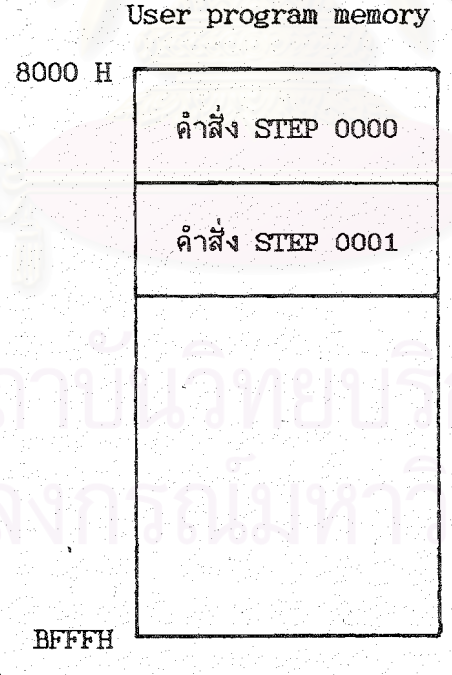
D (Destination) : 50 C4

คำสั่ง : MOV (21) - จะเก็บในหน่วยความจำคือ CD 03 3F 00 20 C0 50 C4

- 02

- HR05

เครื่อง PC ที่ออกแบบนี้สามารถเขียนโปรแกรมได้ประมาณ 4000 คำสั่ง ซึ่งจำนวนที่แน่นอนขึ้นอยู่กับความยาวของแต่ละคำสั่งที่เขียนโปรแกรม โปรแกรมที่ผู้ใช้เขียนนี้จะเก็บในหน่วยความจำขนาด 16K bytes ตำแหน่ง 8000H ถึง BFFFH การเก็บจะมีลักษณะเป็นการนำคำสั่งต่าง ๆ ที่แปลงเป็นภาษาเครื่องแล้ว มาเรียงต่อกันโดยเริ่มจากสเต็ป 0000 เก็บที่ตำแหน่ง 8000H แล้วเก็บคำสั่งสเต็ปอื่น ๆ เรียงต่อไปตามลำดับ



FUN NO	INSTRUCTION	1	2	3	4	5	6	7	8	9	10	REMARK
	LD XXXX	SRL		LD HL,XXXX			OR HL					
		CB	3F	21	XXXX		B6					
	LD NOT XXXX	SRL		LD HL,XXXX			OR HL	XOR C				
		CB	3F	21	XXXX		B6	A9				
	OR XXXX	LD HL,XXXX			OR HL							
		21	XXXX		B6							
	OR NOT XXXX	LD B,A	LD A,(XXXX)			XOR C	OR B					
		47	3A	XXXX		A9	B0					
	AND XXXX	LD B,A	LD A,(XXXX)			OR 7FH		AND B				
		47	3A	XXXX		F6	F7	A0				
	AND NOT XXXX	LD B,A	LD A,(XXXX)			XOR C	OR 7FH		AND B			
		47	3A	XXXX		A9	F6	7F	A0			
	AND LD	ADD A,A	JR C,02		AND 7FH							
		87	38	02	E6	7F						
	OR LD	ADD A,A	JR NC,01		OR C							
		87	30	01	B1							
	OUT XXXX	AND C	LD (XXXX),A									
		A1	32	XXXX								
	OUT NOT XXXX	AND C	XOR C	LD (XXXX),A			XOR C					
		A1	A9	32	XXXX		A9					

ตารางที่ 4.11 แสดงการเก็บคำสั่งเบื้องต้น

FUN NO	INSTRUCTION	1	2	3	4	5	6	7	8	9	10	REMARK
00	NOP	NOP	NOP	NOP	NOP							
		00	00	00	00							
01	END	CALL 0304										
		CD	04	03								
02	IL	CALL 0308										
		CD	08	03								
03	ILC	CALL 030C										
		CD	0C	03								
04	JMP	CALL 0310										
		CD	10	03								
05	JME	CALL 0314										
		CD	14	03								
06	F06	CALL 0318										
		CD	18	03								
07	F07	CALL 031C										
		CD	1C	03								
08	TIM	CALL 0320			NO.	DT	D1					
	D1	CD	20	03	XX	- 00**	XXXX					
09	CNT	CALL 0324			NO.	DT	D1					
	D1	CD	24	03	XX	- 00**	XXXX					

ตารางที่ 4.12 แสดงการเก็บคำสั่งฟังก์ชั่น

FUN NO	INSTRUCTION	1	2	3	4	5	6	7	8	9	10	REMARK
10	SFT	CALL 0328			DT	D1		D2				
	D1,D2	CD	28	03	- ****	XXXX		XXXX				
11	KEEP XXXX	CALL 032C			OPERAND I							
		CD	2C	03	XXXX							
12	CNTR XXX	CALL 0330			NO.	DT	D1					
	D1	CD	30	03	XX	- 00**	XXXX					
13	DIFU XXXX	CALL 0334			OPERAND I							
		CD	34	03	XXXX							
14	DIFD XXXX	CALL 0338			OPERAND I							
		CD	38	03	XXXX							
15	TIMH XXX	CALL 033C			NO.	DT	D1					
	D1	CD	3C	03	XX	- 00**	XXXX					
16	WSFT	CALL 0340			DT	D1		D2				
	D1,D2	CD	40	03	- ****	XXXX		XXXX				
17	MOVL	CALL 0344			DT	S1		D1				
	S1,D1	CD	44	03	- ****	XXXX		XXXX				
18	MOVH	CALL 0348			DT	S1		D1				
	S1,D1	CD	48	03	- ****	XXXX		XXXX				
19	F19	CALL 034C										
		CD	4C	03								

ตารางที่ 4.13 แสดงการเก็บคำสั่งฟังก์ชั่น (ต่อ)

FUN NO	INSTRUCTION	1	2	3	4	5	6	7	8	9	10	REMARK
20	CMP	CALL 0350			DT	S1		S2				
	S1,S2	CD	50	03	- ****	XXXX		XXXX				
21	MOV	CALL 0354			DT	S1		D1				
	S1,D1	CD	54	03	- ****	XXXX		XXXX				
22	MVN	CALL 0358			DT	S1		D1				
	S1,D1	CD	58	03	- ****	XXXX		XXXX				
23	BIN	CALL 035C			DT	S1		D1				
	S1,D1	CD	5C	03	- ****	XXXX		XXXX				
24	BCD	CALL 0360			DT	S1		D1				
	S1,D1	CD	60	03	- ****	XXXX		XXXX				
25	ASL	CALL 0364			DT	D						
	D	CD	64	03	- 00**	XXXX						
26	ASR	CALL 0368			DT	D						
	D	CD	68	03	- 00**	XXXX						
27	ROL	CALL 036C			DT	D						
	D	CD	6C	03	- 00**	XXXX						
28	ROR	CALL 0370			DT	D						
	D	CD	70	03	- 00**	XXXX						
29	COM	CALL 0374			DT	D						
	D	CD	74	03	- 00**	XXXX						

ตารางที่ 4.14 แสดงการเก็บคำสั่งฟังก์ชั่น (ต่อ)

FUN NO	INSTRUCTION	1	2	3	4	5	6	7	8	9	10	REMARK
30	ADD	CALL 0378			DT	S1		S2		D		
	S1,S2,D	CD	78	03	-******	XXXX		XXXX		XXXX		
31	SUB	CALL 037C			DT	S1		S2		D		
	S1,S2,D	CD	7C	03	-******	XXXX		XXXX		XXXX		
32	MUL	CALL 0380			DT	S1		S2		D		
	S1,S2,D	CD	80	03	-******	XXXX		XXXX		XXXX		
33	DIV	CALL 0384			DT	S1		S2		D		
	S1,S2,D	CD	84	03	-******	XXXX		XXXX		XXXX		
34	ANDW	CALL 0388			DT	S1		S2		D		
	S1,S2,D	CD	88	03	-******	XXXX		XXXX		XXXX		
35	ORW	CALL ,038C			DT	S1		S2		D		
	S1,S2,D	CD	8C	03	-******	XXXX		XXXX		XXXX		
36	XORW	CALL 0390			DT	S1		S2		D		
	S1,S2,D	CD	90	03	-******	XXXX		XXXX		XXXX		
37	XNRW	CALL 0394			DT	S1		S2		D		
	S1,S2,D	CD	94	03	-******	XXXX		XXXX		XXXX		
38	INC	CALL 0398			DT	D						
	D	CD	98	03	- 00**	XXXX						
39	DEC	CALL 039C			DT	D						
	D	CD	9C	03	- 00**	XXXX						

ตารางที่ 4.15 แสดงการเก็บคำสั่งฟังก์ชั่น (ต่อ)

FUN NO	INSTRUCTION	1	2	3	4	5	6	7	8	9	10	REMARK
40	STC	CALL 03A0										
		CD	A0	03								
41	CLC	CALL 03A4										
		CD	A4	03								
42	BKM	CALL 03A8			DT	W		S		D		
	W,S,D	CD	A8	03	-******	XXXX		XXXX		XXXX		
43	BKS	CALL 03AC			DT	S		D1		D2		
	S,D1,D2	CD	AC	03	-******	XXXX		XXXX		XXXX		
44	DAEX	CALL 03B0			DT	D1		D2		-		
	D1,D2,-	CD	B0	03	-******	XXXX		XXXX		-		
45	ODSL	CALL 03B4			DT	D1		D2		-		
	D1,D2,-	CD	B4	03	-******	XXXX		XXXX		-		
46	ODSR	CALL 03B8			DT	D1		D2		-		
	D1,D2,-	CD	B8	03	-******	XXXX		XXXX		-		
47	DECO	CALL 03BC			DT	S		K		D		
	S,K,D	CD	BC	03	-******	XXXX		0-3		XXXX		
48	ENCO	CALL 03C0			DT	S		K		D		
	S,K,D	CD	C0	03	-******	XXXX		0-3		XXXX		
49	7SEG	CALL 03C4			DT	S		K		D		
	S,K,D	CD	C4	03	-******	XXXX		0-3		XXXX		

ตารางที่ 4.16 แสดงการเก็บคำสั่งฟังก์ชั่น (ต่อ)

FUN NO	INSTRUCTION	1	2	3	4	5	6	7	8	9	10	REMARK
50	F50	CALL 03C8										
		CD	C8	03								
51	F51	CALL 03CC										
		CD	CC	03								
52	F52	CALL 03CF										
		CD	CF	03								
53	F53	CALL 03D4										
		CD	D4	03								

ตารางที่ 4.17 แสดงการเก็บคำสั่งฟังก์ชั่น (ต่อ)

4.4 การทำงานของคำสั่งเบื้องต้น

คำสั่งเบื้องต้นเป็นคำสั่งเกี่ยวกับการอ่านค่าอินพุท การทำตรรกศาสตร์และการส่งผลลัพธ์ไปยังเอาท์พุท ซึ่งได้แก่คำสั่ง LD, LD NOT, OR, OR NOT, AND, AND NOT, AND LD, OR LD, OUT และ OUT NOT รูปแบบของคำสั่งจะประกอบด้วยอ๊อฟโค๊ด (Opcode) และโอเปอร์แรนด์ (Operand)

โอเปอร์แรนด์จะประกอบด้วยชนิดของข้อมูล และตำแหน่งของข้อมูล คำสั่งแต่ละคำสั่งจะสามารถใช้กับโอเปอร์แรนด์ได้แตกต่างกันดังนี้

Content of Operand	LD	LD NOT OR OR NOT AND AND NOT	OUT	OUT NOT	AND LD OR LD
I/O and AUX relay	0000-6307		0000-6015		-
Holding relay	HR 0000-3115				-
Timer, counter	TIM/CNT 000-127		-		-
Temporary relay	TRO-7	-	TRO-7	-	-

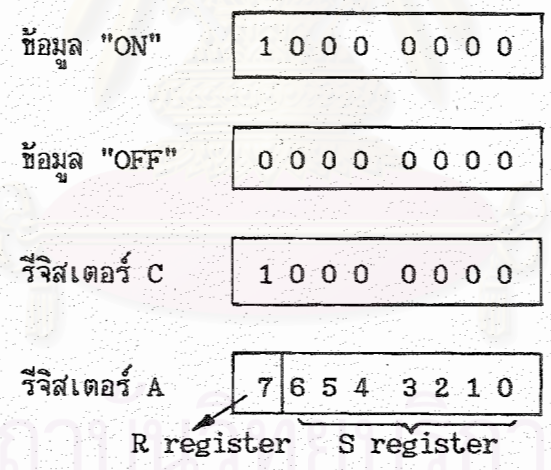
ตารางที่ 4.18 แสดงการใช้โอเปอร์แรนด์ของคำสั่งเบื้องต้น

การทำงานของคำสั่งเบื้องต้นเหล่านี้จะเกี่ยวข้องกับข้อมูล 2 สถานะคือ สถานะ ON และสถานะ OFF โดยสถานะ ON จะมีข้อมูลเป็น 80 H และสถานะ OFF จะมีข้อมูลเป็น 00H ในการใช้งานของผู้ใช้จะมองเสมือนว่ามีการทำงานของคำสั่งจะเกี่ยวข้องกับรีจิสเตอร์ 2 ชนิด คือ Result register และ Stack register

Result register (R) หมายถึงที่เก็บข้อมูลของผลลัพท์ ในการทำงานของคำสั่งต่าง ๆ

Stack register (S) หมายถึงที่พักข้อมูลชั่วคราว สำหรับไว้ใช้ในคำสั่งถัดไป

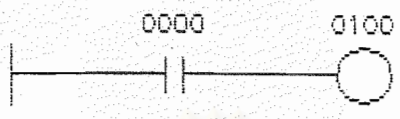
ในเครื่อง PC ที่ออกแบบนี้จะใช้แอคคิวมูเลเตอร์รีจิสเตอร์ของซีพียู (A register) เป็นที่เก็บข้อมูลของ R และ S โดย R จะถูกเก็บที่บิท 7 และ S จะเก็บที่บิท 6 จนถึงบิท 1 ซึ่งทำให้มี S ถึง 7 ตัว ทำให้สามารถโปรแกรมขั้นบันไดที่ต้องเก็บสถานะข้อมูลไว้ได้ถึง 7 ขั้น การทำงานคำสั่งเหล่านี้ของซีพียูจะต้องให้รีจิสเตอร์ C มีค่าเป็น 80H ด้วย ซึ่งจะอธิบายรายละเอียดในการทำงานของแต่ละคำสั่งต่อไป



รูปที่ 4.17 แสดงการเก็บข้อมูลของรีจิสเตอร์ภายในขณะทำงาน

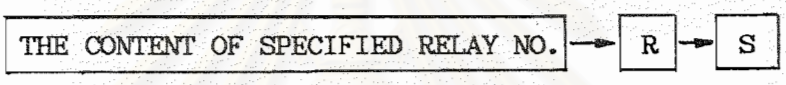
4.4.1 คำสั่ง LOAD (LD)

จะใช้คำสั่ง LD เป็นคำสั่งเริ่มต้นสำหรับรีเลย์อินพุทตัวแรกของแต่ละแถว ที่มีหน้าสัมผัสเป็นแบบปกติเปิด (Normally open)



CODING

STEP	PECODE	OPERAND
0100	LD	0000
0101	OUT	0100



รูปที่ 4.18 แสดงการใช้งาน และการทำงานของคำสั่ง LD

การทำงานของคำสั่ง LD จะอ่านข้อมูลอินพุทตามหมายเลขที่ระบุในโอเปอร์แรนด์ของคำสั่งมาเก็บไว้ในรีจิสเตอร์ R และนำค่าสถานะเดิมของรีจิสเตอร์ R ไปเก็บไว้ที่รีจิสเตอร์ S

การทำงานภาษาเครื่องของคำสั่ง LD

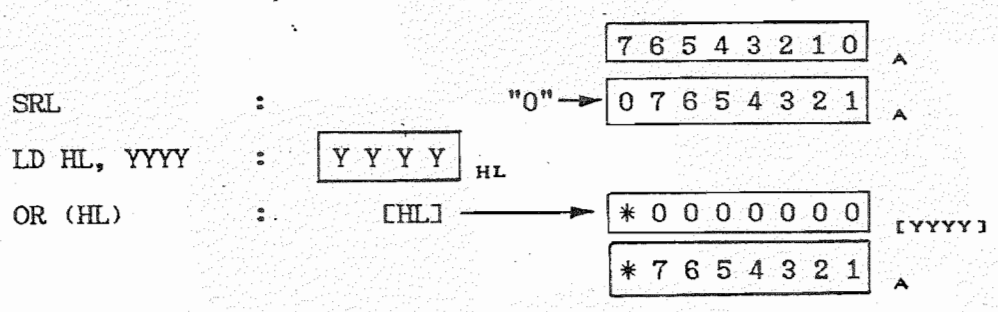
คำสั่ง LD เมื่อแปลงเป็นภาษาเครื่องมีคำสั่งดังนี้

```
LD xxxx  :  SRL
            LD  HL,YYYY
            OR  (HL)
```

คำสั่ง SRL เป็นการเลื่อนค่า S ไปหนึ่งบิท และย้ายข้อมูลจาก R ไปยัง S

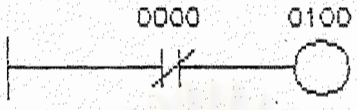
คำสั่ง LD HL, YYYY เป็นการกำหนดตัวชี้ตำแหน่งข้อมูล

คำสั่ง OR (HL) อ่านข้อมูลสถานะของอินพุทไปเก็บไว้ที่ R



4.4.2 คำสั่ง LOAD-NOT (LD-NOT)

จะใช้คำสั่ง LD-NOT เป็นคำสั่งเริ่มต้นสำหรับรีเลย์อินพุทตัวแรกของแต่ละแถวที่มีหน้าสัมผัสเป็นแบบปกติปิด (Normally close)



CODING

STEP	OPCODE	OPERAND
0102	LD NOT	0000
0103	OUT	0100

รูปที่ 4.19 แสดงการใช้งาน และการทำงานของคำสั่ง LD-NOT

การทำงานของคำสั่ง LD-NOT จะอ่านข้อมูลสถานะอินพุทตามหมายเลขที่ระบุในโอเปอร์แรนด์ของคำสั่ง แล้ว Inverse ข้อมูลและนำไปเก็บในรีจิสเตอร์ R นำค่าสถานะเดิมของรีจิสเตอร์ R ไปเก็บไว้ในรีจิสเตอร์ S

การทำงานภาษาเครื่องของคำสั่ง LD-NOT

คำสั่ง LD-NOT เมื่อแปลงเป็นภาษาเครื่องจะมีคำสั่งดังนี้

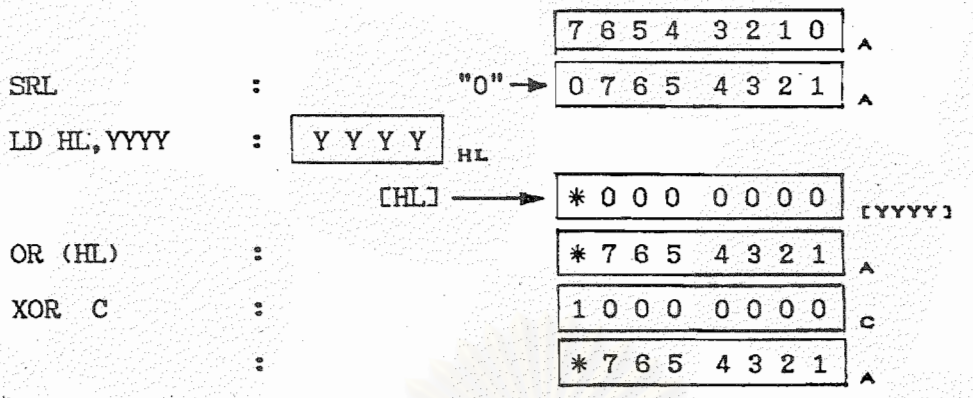
```
LD-NOT xxxx    :   SRL
                   LD  HL, YYYY
                   OR  (HL)
                   XOR C
```

คำสั่ง SRL เป็นการเลื่อนค่าของ S และย้ายค่ารีจิสเตอร์ R ไปยังรีจิสเตอร์ S

คำสั่ง LD HL, YYYY เป็นการกำหนดตัวชี้ตำแหน่งข้อมูล

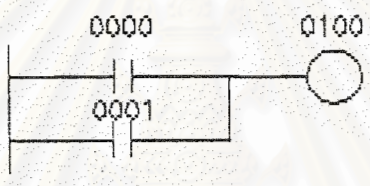
คำสั่ง OR (HL) เป็นการอ่านข้อมูลสถานะอินพุทไปเก็บไว้ที่รีจิสเตอร์ R

คำสั่ง XOR C เป็นการ Inverse ค่าของรีจิสเตอร์ R



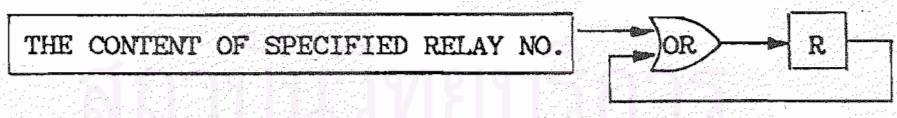
4.4.3 คำสั่ง OR

คำสั่ง OR จะใช้แทนรีเลย์อินพุทแบบปกติเปิด (NO) ที่ต่อขนานอยู่



CODING

STEP	OPCODE	OPERAND
0000	LD	0000
0001	OR	0001
0002	OUT	0100



รูปที่ 4.20 แสดงการใช้งาน และการทำงานของคำสั่ง OR

การทำงานของคำสั่ง OR จะเป็นการอ่านสถานะข้อมูลที่ถูกระบุตำแหน่งโดยโอเปอร์แรนด์มาทำลอจิก OR กับค่าของรีจิสเตอร์ R แล้วนำค่าไปเก็บไว้ที่รีจิสเตอร์ R

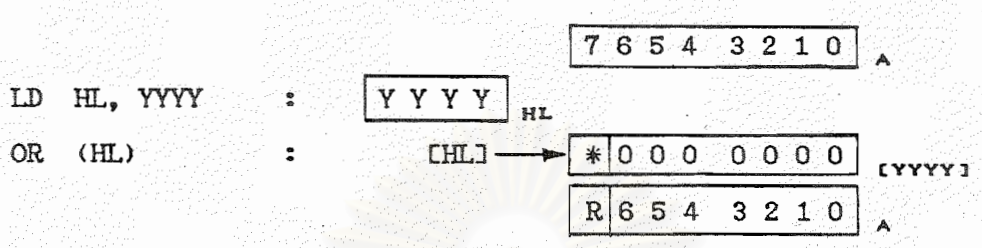
การทำงานภาษาเครื่องของคำสั่ง OR

คำสั่ง OR เมื่อแปลงเป็นภาษาเครื่องมีคำสั่งดังนี้

OR xxxx : LD HL, xxxx
OR (HL)

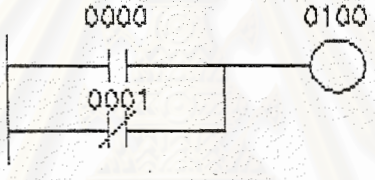
คำสั่ง LD HL, xxxx เป็นการกำหนดตัวชี้ตำแหน่งของข้อมูล

คำสั่ง OR (HL) เป็นการ OR ข้อมูลอินพุทกับ R และเก็บไว้ที่ R



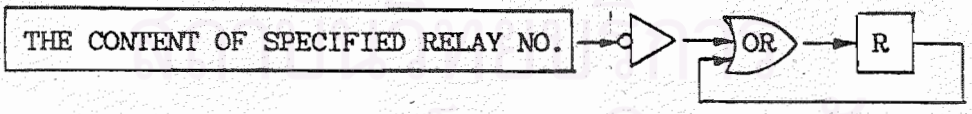
4.4.4 คำสั่ง OR-NOT

คำสั่ง OR-NOT จะใช้แทนรีเลย์อินพุทแบบปกติปิด (NC) ที่ต่อขนานอยู่



CODING

STEP	OPCODE	OPERAND
0000	LD	0000
0001	OR NOT	0001
0002	OUT	0100



รูปที่ 4.21 แสดงการใช้งาน และการทำงานของคำสั่ง OR-NOT

การทำงานของคำสั่ง OR-NOT จะเป็นการอ่านสถานะข้อมูลที่ตำแหน่ง ซึ่งถูกระบุโดยโอเปอร์แรนด์ในคำสั่ง และ Inverse ข้อมูล แล้วทำลอจิก OR กับรีจิสเตอร์ R นำค่าผลลัพธ์ไปเก็บไว้ในรีจิสเตอร์ R

การทำงานภาษาเครื่องของคำสั่ง OR-NOT

คำสั่ง OR NOT xxxx เมื่อแปลงเป็นภาษาเครื่องมีคำสั่งดังนี้

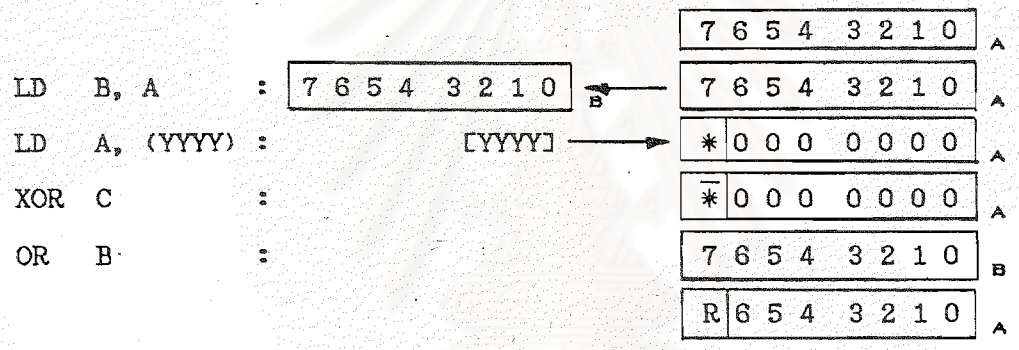
```
OR xxxx  :  LD   B, A
            LD   A, (YYYY)
            XOR  C
            OR   B
```

คำสั่ง LD B,A เก็บค่าของ R และ S ไว้ที่รีจิสเตอร์ B

คำสั่ง LD A, (YYYY) อ่านค่าข้อมูลสถานะของอินพุท

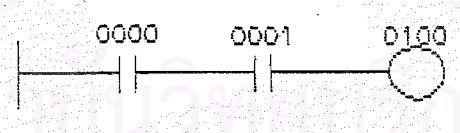
คำสั่ง XOR C Inverse ค่าข้อมูลสถานะของอินพุท

คำสั่ง OR B นำค่า Inverse ของข้อมูลไปทำลอจิก OR กับค่า R และเก็บไว้ที่ R



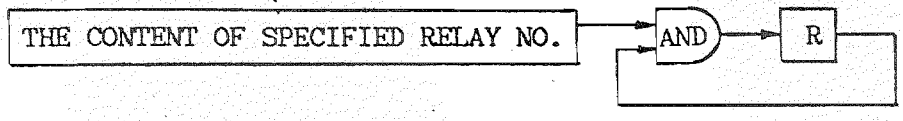
4.4.5 คำสั่ง AND

คำสั่ง AND จะใช้แทนรีเลย์อินพุทแบบปกติเปิด (NO) ที่ต่ออนุกรมอยู่



CODING

STEP	OPCODE	OPERAND
0000	LD	0000
0001	AND	0001
0002	OUT	0100



รูปที่ 4.22 แสดงการใช้งาน และการทำงานของคำสั่ง AND

การทำงานของคำสั่ง AND จะอ่านสถานะข้อมูลที่ตำแหน่งซึ่งระบุโดยโอเปอร์แรนด์ในคำสั่ง นำมาทำลอจิก AND กับค่าในรีจิสเตอร์ R แล้วเก็บผลลัพท์ไว้ในรีจิสเตอร์ R

การทำงานภาษาเครื่องของคำสั่ง AND

คำสั่ง AND xxxx แปลงเป็นภาษาเครื่องมีคำสั่งดังนี้

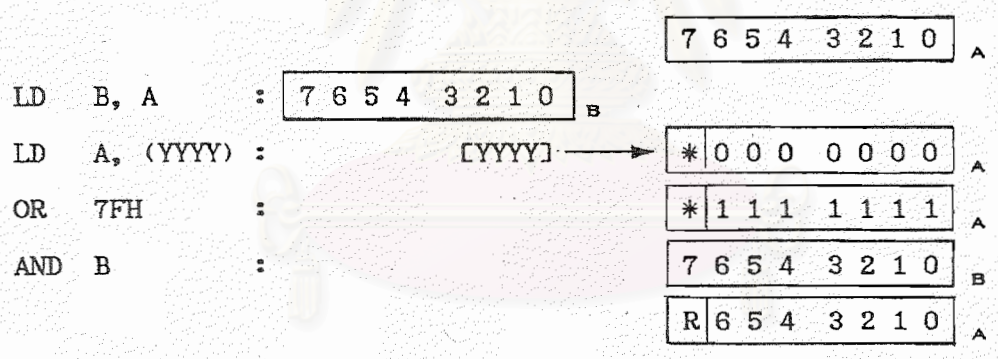
```
AND xxxx  :  LD   B, A
             LD   A, (YYYY)
             OR   7FH
             AND  B
```

คำสั่ง LD B, A ย้ายค่าของ R และ S ไปเก็บที่รีจิสเตอร์ B

คำสั่ง LD A, (YYYY) อ่านค่าอินพุทไปเก็บที่รีจิสเตอร์ A

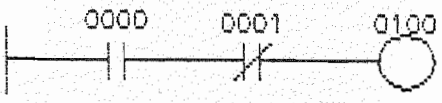
คำสั่ง OR 7FH จัดค่าข้อมูลอินพุทให้เหมาะสม

คำสั่ง AND B นำค่าของข้อมูลอินพุทไปทำลอจิก AND กับค่า R และเก็บไว้ที่ R



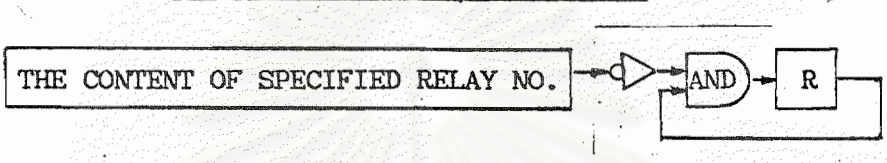
4.4.6 <u>คำสั่ง AND NOT</u>

คำสั่ง AND NOT ใช้แทนรีเลย์อินพุทแบบปกติปิด (NC) ที่ต่ออนุกรมอยู่



CODING

STEP	OPCODE	OPERAND
0000	LD	0000
0001	AND NOT	0001
0002	OUT	0100



รูปที่ 4.23 แสดงการใช้งาน และการทำงานของคำสั่ง AND NOT

การทำงานของคำสั่ง AND-NOT จะอ่านสถานะข้อมูลที่ตำแหน่ง ซึ่งระบุโดยโอเปอร์แรนด์ของคำสั่ง นำไป Inverse และทำลอจิก AND กับรีจิสเตอร์ R นำผลลัพธ์ไปเก็บไว้ที่รีจิสเตอร์ R

การทำงานภาษาเครื่องของคำสั่ง AND-NOT

```
คำสั่ง AND NOT xxxx  :  LD   B, A
                        LD   A, (YYYY)
                        XOR  C
                        OR   7FH
                        AND  B
```

คำสั่ง LD B, A นำค่าของ R และ S ไปเก็บที่รีจิสเตอร์ B

คำสั่ง LD A, (YYYY) อ่านค่าข้อมูลสถานะของอินพุท

คำสั่ง XOR C Inverse ค่าข้อมูลสถานะของอินพุท

คำสั่ง OR 7FH จัดค่าข้อมูลอินพุทให้เหมาะสม

คำสั่ง AND B นำค่าข้อมูลอินพุทไปทำลอจิก AND กับค่า R แล้วนำไปเก็บที่ R

การทำงานของคำสั่ง

1. จากคำสั่ง LD 0000 และ OR 0001 จะได้ผลลัพธ์ของลอจิก OR ในบล๊อก A และจะเก็บผลลัพธ์ไว้ในรีจิสเตอร์ B
2. จากคำสั่ง LD 0002 จะทำให้ค่าในรีจิสเตอร์ R ถูกย้ายไปเก็บในรีจิสเตอร์ S และผลลัพธ์ของการทำลอจิกของคำสั่ง LD 0002 และคำสั่ง OR-NOT 0003 จะถูกเก็บในรีจิสเตอร์ R
3. คำสั่ง AND-LD จะเป็นการทำลอจิก AND ระหว่างรีจิสเตอร์ R และรีจิสเตอร์ S ผลลัพธ์ที่ได้จะเก็บในรีจิสเตอร์ R

การทำงานภาษาเครื่องของคำสั่ง AND-LD

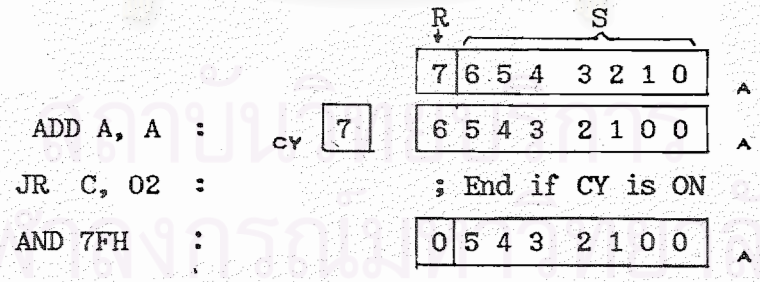
คำสั่ง AND-LD เมื่อแปลงเป็นภาษาเครื่องจะมีคำสั่งดังนี้

```
AND-LD  :  ADD  A, A
           JR   C, 02
           AND  7FH
```

คำสั่ง ADD A, A เลื่อนค่า S ไปเก็บที่ R และตรวจสอบค่า R เดิม

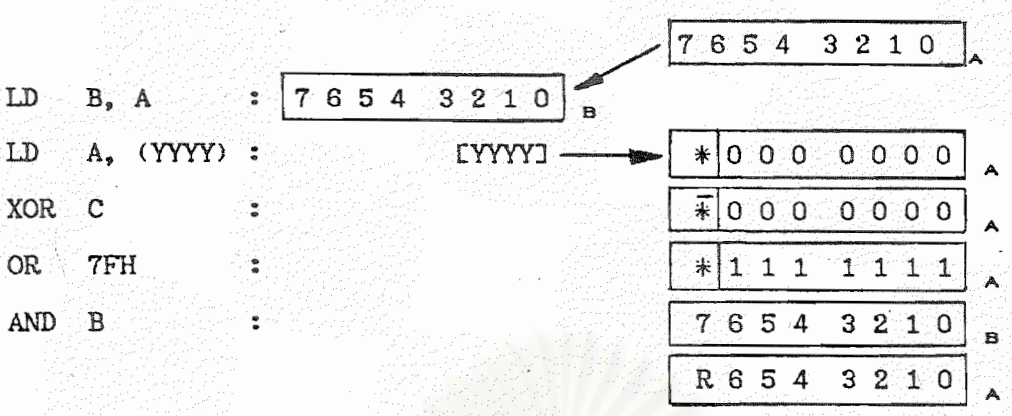
คำสั่ง JR C, 02 ถ้าค่า R เดิมมีสถานะ ON ค่าผลลัพธ์จะเท่ากับค่าของ S ที่เก็บอยู่ที่ R ในปัจจุบัน

คำสั่ง AND 7FH ถ้าค่า R เดิมมีสถานะ OFF ผลลัพธ์ที่ R ปัจจุบันต้องเซ็ทเป็น OFF



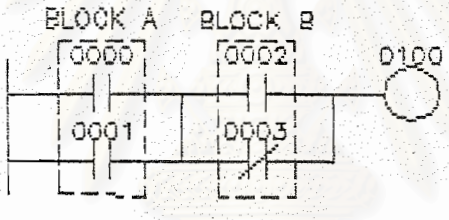
4.4.8 <u>คำสั่ง OR-LOAD</u> (OR-LD)

คำสั่ง OR-LD จะใช้ในการ OR กันของบล๊อกภายในตั้งแต่ 2 บล๊อก หรือมากกว่าดังแสดงในรูปที่ 4.25



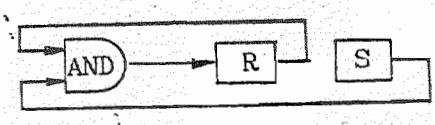
4.4.7 คำสั่ง AND LOAD (AND LD)

คำสั่ง AND LD จะใช้ในการ AND กันของบล๊อกภายในตั้งแต่ 2 บล๊อก หรือมากกว่า ดังแสดงในรูปที่ 4.24

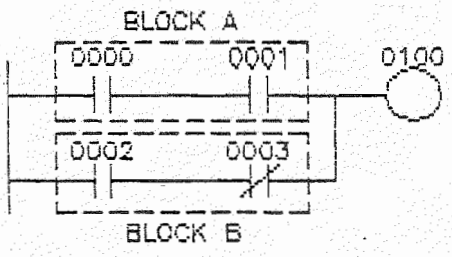


CODING

STEP	OPCODE	OPERAND
0300	LD	0000
0301	OR	0001
0302	LD	0002
0303	OR NOT	0003
0304	AND LD	-
0305	OUT	0100

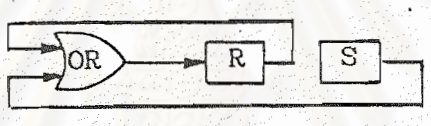


รูปที่ 4.24 แสดงการใช้ และการทำงานของคำสั่ง AND LD



CODING

STEP	OPCODE	OPERAND
0300	LD	0000
0301	AND	0001
0302	LD	0002
0303	AND-NOT	0003
0304	OR-LD	-
0305	OUT	0100



รูปที่ 4.25 แสดงการใช้ และการทำงานของคำสั่ง OR LD

การทำงานของคำสั่ง

1. จากคำสั่ง LD 0000 และ AND 0001 จะได้ผลลัพธ์ของลอจิก OR ในบล๊อก A เก็บผลลัพธ์ในรีจิสเตอร์ R
2. จากคำสั่ง LD 0002 จะทำให้ค่าในรีจิสเตอร์ R ถูกย้ายไปเก็บในรีจิสเตอร์ S และผลลัพธ์การทำลอจิกของคำสั่ง LD 0002 และคำสั่ง AND-NOT 0003 ในบล๊อก B จะเก็บในรีจิสเตอร์ R
3. คำสั่ง OR-LD จะเป็นการทำลอจิก OR ระหว่างค่าในรีจิสเตอร์ R และค่าในรีจิสเตอร์ S ผลลัพธ์ที่ได้เก็บไว้ในรีจิสเตอร์ R

การทำงานภาษาเครื่องของคำสั่ง OR-LD

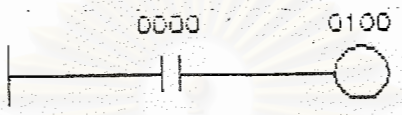
คำสั่ง OR-LD เมื่อแปลงเป็นภาษาเครื่องจะมีคำสั่งดังนี้

```
OR-LD   :   ADD   A, A
            JR    NC, 01
            OR    C
```

คำสั่ง ADD A,A จะเลื่อนค่าของ S ไปเก็บที่ R และตรวจสอบค่าของ R เดิม
คำสั่ง JR NC, 01 ถ้าค่า R เดิมมีสถานะ OFF ก็จบ
คำสั่ง OR C ถ้าค่า R เดิมมีสถานะ ON ค่าผลลัพธ์ที่ R ปัจจุบันต้องเซ็ทให้ ON

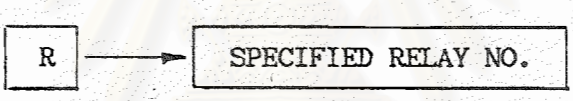
4.4.9 คำสั่ง OUT

คำสั่ง OUT ใช้สำหรับแทนเอาท์พุทรีเลย์



CODING

STEP	OPCODE	OPERAND
0200	LD	0000
0201	OUT	0100



รูปที่ 4.26 แสดงการใช้ และการทำงานของคำสั่ง OUT

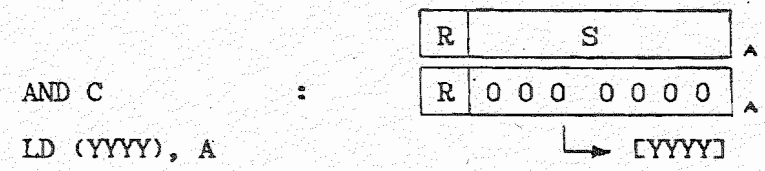
การทำงานของคำสั่ง OUT จะเป็นการนำค่าในรีจิสเตอร์ R ส่งไปที่เอาท์พุทของรีเลย์ ซึ่งถูกระบุโดยโอเปอร์แรนด์ของคำสั่ง การทำคำสั่งนี้ค่าสถานะของรีจิสเตอร์ R ยังคงเดิม แต่ค่าของรีจิสเตอร์ S จะถูกลบไป

การทำงานภาษาเครื่องของคำสั่ง OUT

คำสั่ง OUT เมื่อแปลงเป็นภาษาเครื่องจะมีคำสั่งดังนี้

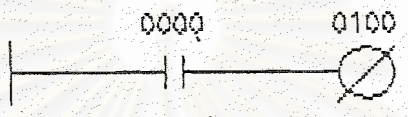
```
OUT xxxx  :  AND  C
             LD   [YYYY], A
```

คำสั่ง AND C เป็นการจัดข้อมูลสำหรับเอาท์พุทให้เหมาะสม
คำสั่ง LD (YYYY), A เป็นการนำค่าของรีจิสเตอร์ R ไปเก็บในตำแหน่งเอาท์พุทที่กำหนด



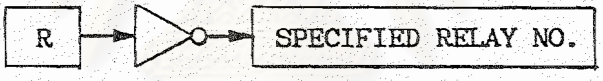
4.4.10 คำสั่ง OUT-NOT

คำสั่ง OUT-NOT ใช้แทนเอาท์พุทรีเลย์ที่เป็น Inverse



CODING

STEP	OPCODE	OPERAND
0200	LD	0001
0201	OUT NOT	0100



รูปที่ 4.27 แสดงการใช้ และการทำงานของคำสั่ง OUT-NOT

การทำงานของคำสั่ง OUT-NOT จะเป็นการนำค่าของรีจิสเตอร์ R มาทำ Inverse แล้วส่งไปที่เอาท์พุทของรีเลย์ ซึ่งถูกระบุโดยโอเปอร์แรนด์ของคำสั่ง การทำคำสั่งนี้ค่าของรีจิสเตอร์ R ยังคงเดิมแต่ค่าของรีจิสเตอร์ S จะถูกลบไป

การทำงานภาษาเครื่องของคำสั่ง OUT-NOT

คำสั่ง OUT-NOT เมื่อแปลงเป็นภาษาเครื่องจะมีคำสั่งดังนี้

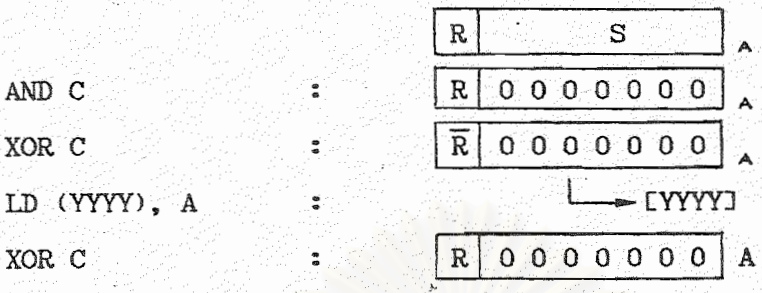
OUT-NOT xxxx : AND C
XOR C
LD [YYYY], A
XOR C

คำสั่ง AND C เป็นจัดข้อมูลสำหรับเอาท์พุทให้เหมาะสม

คำสั่ง XOR C เป็นการ Inverse ค่าของรีจิสเตอร์ R

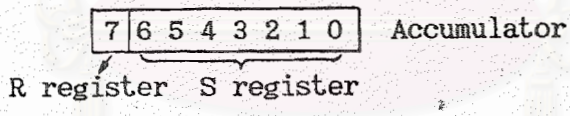
คำสั่ง LD (YYYY), A เป็นการนำค่าข้อมูลไปเก็บในตำแหน่งเอาท์พุทที่กำหนด

คำสั่ง XOR C เป็นการจัดข้อมูลของรีจิสเตอร์ R ให้ถูกต้อง



4.5 การทำงานของคำสั่งฟังก์ชั่น

คำสั่งฟังก์ชั่นมีทั้งหมด 54 คำสั่ง มีหมายเลขฟังก์ชั่นตั้งแต่ 00 ถึง 53 เงื่อนไขการทำงานของฟังก์ชั่นส่วนใหญ่ขึ้นอยู่กับค่าสถานะของรีจิสเตอร์ R ถ้ามีสถานะ ON จะทำงานตามฟังก์ชั่นนั้น ถ้ามีสถานะ OFF จะไม่ทำงานตามฟังก์ชั่นนั้น แต่มีฟังก์ชั่นบางคำสั่งที่ใช้ค่ารีจิสเตอร์ R และค่าในรีจิสเตอร์ S เป็นเงื่อนไขในการทำงาน ค่าของรีจิสเตอร์ R และ S นี้จะเป็นผลลัพธ์ของการทำคำสั่งเบื้องต้นต่าง ๆ ซึ่งจะเก็บอยู่ในแอคคิวมูเลเตอร์ของซีพียู

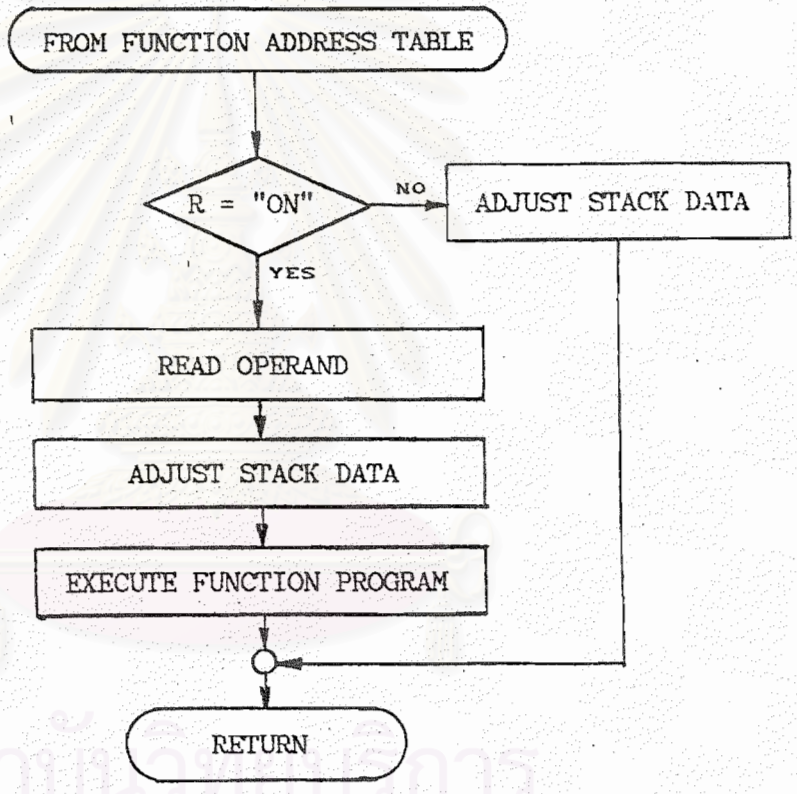


รหัสคำสั่งภาษาเครื่องของฟังก์ชั่นต่างๆ จะเป็นคำสั่งให้ซีพียูกระโดดไปทำงานที่สับรูทีนต่าง ๆ ซึ่งกำหนดไว้ในตาราง สำหรับฟังก์ชั่นที่มีโอเปอร์แรนด์ ก็จะเก็บโอเปอร์แรนด์ไว้ต่อท้ายคำสั่งกระโดดไปทำงานสับรูทีนด้วย

CALL 0354		DT	S1	D1
CD	54 03	0000 $*_2$ $*_1$	XXXX	XXXX

รูปที่ 4.28 แสดงภาษาเครื่องของคำสั่ง MOV(21)

ลักษณะการทำงานของคำสั่งฟังก์ชั่น แสดงในโฟล์วชาร์ทรูปที่ 4.29 โดยเริ่มทำงานจากคำสั่งกระโดดไปทำงานสับรูทีน (CALL) ซึ่งจะไปทำงานยังตารางที่สร้างไว้ตารางนี้จะเป็นตัวชี้ให้ไปทำงานตามโปรแกรมสำหรับฟังก์ชั่นต่างๆ ขณะนั้นค่าในสแต็กจะเป็นตำแหน่งแอดเดรสถัดจากคำสั่ง CALL ซึ่งจะเป็นตัวชี้ตำแหน่งโอเปอร์แรนด์ให้กับฟังก์ชั่นนั้น เมื่อโปรแกรมของฟังก์ชั่นทำงานเสร็จแล้ว ก็จะปรับค่าในสแต็กให้ชี้ไปยังตำแหน่งของคำสั่งขั้นบันไดตัวต่อไป แล้วทำคำสั่ง RET เพื่อไปทำงานตามคำสั่งขั้นบันไดคำสั่งต่อไป สำหรับฟังก์ชั่นที่ไม่มีโอเปอร์แรนด์ก็ไม่ต้องปรับค่าข้อมูลในสแต็ก

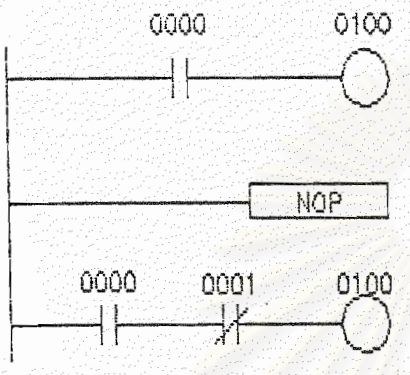


รูปที่ 4.29 แสดงโฟลว์ชาร์ทการทำงานของฟังก์ชั่นทั่วไป

ฟังก์ชั่นต่าง ๆ เหล่านี้สามารถแบ่งออกตามหน้าที่การทำงานได้หลายกลุ่ม เช่น คำสั่งเกี่ยวกับการควบคุม คำสั่งคำนวณ คำสั่งการเปรียบเทียบ คำสั่งลอจิก และคำสั่งในการแปลงรหัสต่าง ๆ สำหรับการอธิบายการทำงานและการใช้ฟังก์ชั่นต่าง ๆ ในลำดับต่อไปจะเรียงตามลำดับหมายเลขฟังก์ชั่นเป็นหลัก โดยไม่จำแนกเป็นกลุ่มต่าง ๆ

4.5.1 คำสั่ง NOP (FUN00)

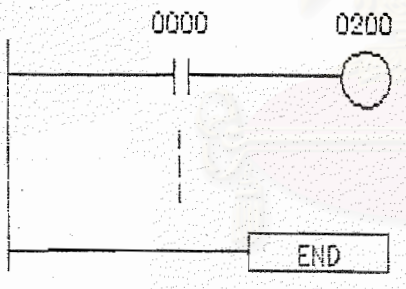
คำสั่งนี้เหมือนกับไม่มีการทำงานจะไม่ผลต่อข้อมูลต่าง ๆ ของเครื่อง คำสั่งนี้สามารถใส่เพื่อเป็นช่องว่างระหว่างคำสั่งอื่น ๆ ได้ ทุกครั้งที่มีการลบโปรแกรมทั้งหมดของผู้ใช้คำสั่ง NOP จะถูกใส่ลงหน่วยความจำทั้งหมด



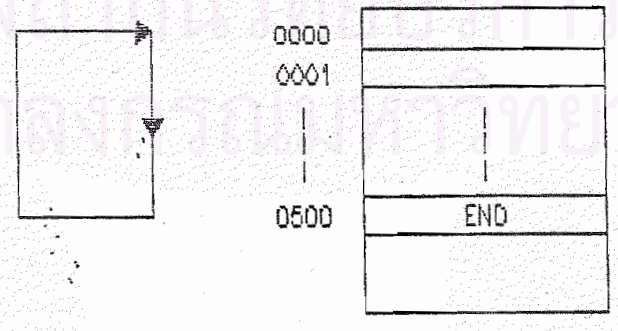
STEP	OPERAND	OPCODE
0000	LD	0000
0001	OUT	0100
0002	NOP (00)	
0003	LD	0001
0004	AND NOT	0002
0005	OUT	0101

4.5.2 คำสั่ง END (FUN01)

คำสั่ง END จะใส่เป็นคำสั่งสุดท้ายในการเขียนโปรแกรม เพื่อให้เครื่องรู้ว่าจบโปรแกรมและกลับมาเริ่มทำงานคำสั่งแรกใหม่



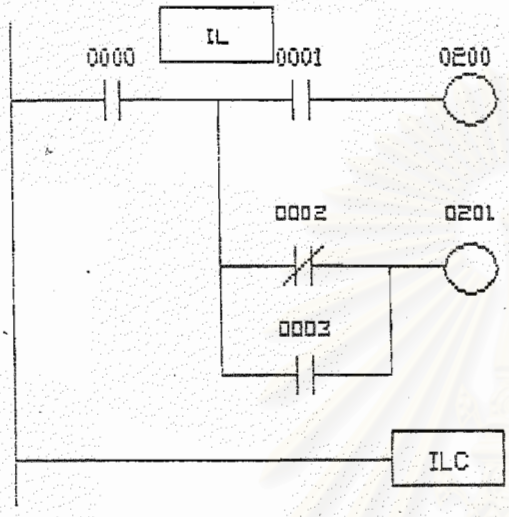
STEP	OPCODE	OPERAAND
0000	LD	0000
0500	END (01)	



ถ้าไม่มีคำสั่ง END ในโปรแกรมเครื่องจะไม่ยอมทำงาน และแสดง Error มีข้อความที่จอ LCD ว่า "NO END INSTR"

4.5.3 คำสั่ง INTERLOCK [IL (FUN02)] และ INTERLOCK CLEAR [FUN03)]

คำสั่ง IL และ ILC จะใช้ร่วมกันเป็นคู่ โดยใช้ IL เป็นจุดเริ่มต้นและ ILC เป็นจุดสุดท้ายในช่วงโปรแกรมที่ควบคุม การทำงานจะเหมือนว่าเอาท์พุททั้งหมดในช่วงคำสั่ง IL ถึง LC จะถูกควบคุมด้วยผลลัพธ์ของรีจิสเตอร์ R ขณะพบคำสั่ง IL



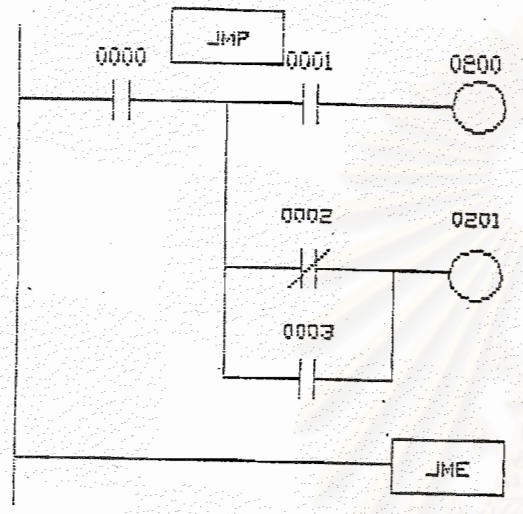
STEP	OPCODE	OPERAND
0100	LD	0000
0101	IL(02)	
0102	LD	0001
0103	OUT	0200
0104	LD NOT	0002
0105	OR	0003
0106	OUT	0201
0107	ILC(03)	

ถ้าสถานะของคำสั่ง IL เป็น ON การทำงานของโปรแกรมจะเหมือนไม่มีคำสั่ง IL และ ILC ในโปรแกรม แต่ถ้าสถานะของคำสั่ง IL เป็น OFF จะมีผลทำให้เอาท์พุท และข้อมูลต่าง ๆ ที่อยู่ระหว่างคำสั่ง IL และ ILC เป็นดังนี้

รีเลย์เอาท์พุท, รีเลย์ช่วย	OFF
ตัวตั้งเวลา	รีเซ็ท
ตัวนับ, ชิสรีจิสเตอร์, HR รีเลย์	คงสถานะเดิม

4.5.4 คำสั่ง JUMP [JMP(FUN04) I และ JUMP END [JME(FUN05)]

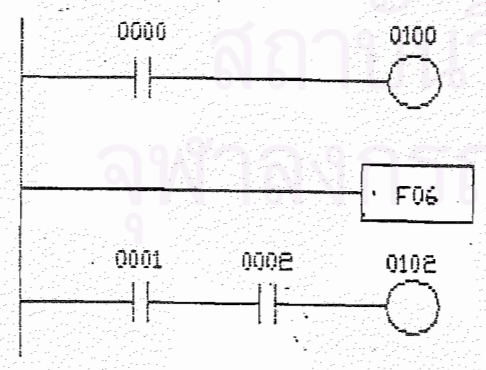
การใช้งานคำสั่ง JMP จะร่วมกับคำสั่ง JME ซึ่งมีการทำงานคือ ถ้าสถานะของคำสั่ง JMP เป็น ON โปรแกรมจะทำงานเหมือนปกติ (เหมือนไม่มีคำสั่ง JMP และ JME) แต่ถ้าสถานะของคำสั่ง JMP เป็น OFF โปรแกรมชั้นบันไดที่อยู่ระหว่างคำสั่ง JMP และคำสั่ง JME จะถูกข้ามไป



STEP	OPCODE	OPERAND
0200	LD	0000
0201	JMP(04)	
0202	LD	0001
0203	OUT	0200
0204	LD NOT	0002
0205	OR	0003
0206	OUT	0201
0207	JME(05)	

4.5.6 คำสั่ง F06 (FUN06), F07 (FUN07), F19(FUN19), F50(FUN50), F51(FUN51), F52(FUN52) และ F53(FUN52)

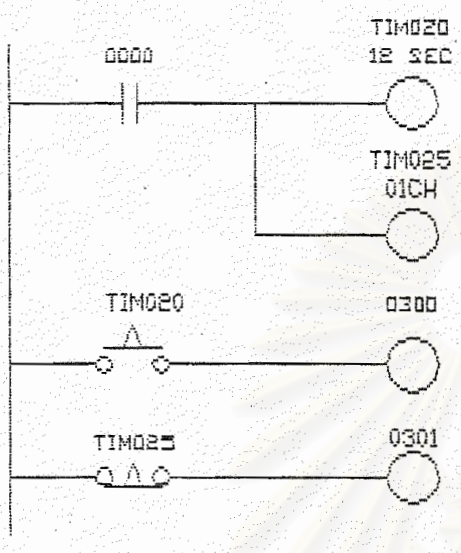
คำสั่งเหล่านี้เป็นคำสั่งว่างไม่มีผลต่อการทำงานของโปรแกรมชั้นบันได จะมีหน้าที่คล้ายคำสั่ง NOP คำสั่งเหล่านี้สามารถนำมาทำเพิ่มเป็นคำสั่งอื่น ๆ ในอนาคตได้



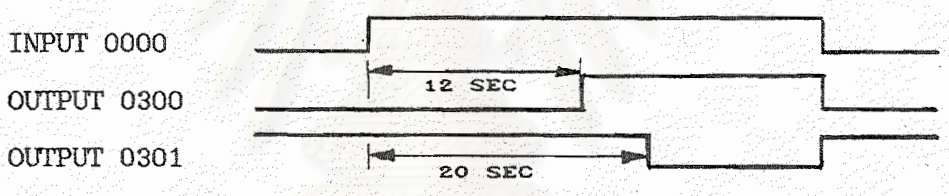
STEP	OPCODE	OPERAND
0100	LD	0000
0101	OUT	0100
0102	F06	
0103	LD	0001
0104	AND	0002
0105	OUT	0102

4.5.7 คำสั่ง TIMER [TIM(08)]

เป็นคำสั่งตัวตั้งเวลา ซึ่งเป็นแบบ ON-delay timer การทำงานสามารถแสดงได้ด้วยรูป 4.30



STEP	OPCODE	OPERAND
0200	LD	0000
0201	TIM	020
		# 0120
0202	TIM	025
		01
0203	LD	TIM 020
0204	OUT	0300
0205	LD NOT TIM	025
0206	OUT	0301



รูปที่ 4.30 แสดงการทำงานของคำสั่ง TIM

หมายเลขของ TIM อยู่ระหว่าง 000-127 ซึ่งจะต้องใช้ร่วมกับคำสั่ง Timer และ Counter อื่น ๆ ดังนั้นการใช้งานต้องระวังหมายเลขซ้ำกันด้วย

ข้อมูลที่ใช้เป็นเวลาของคำสั่ง TIM มีความละเอียด 0.1 วินาที หมายถึงค่าข้อมูลหลัง TIM จะต้องหารด้วย 10 จึงมีหน่วยเป็นวินาที

ข้อมูลที่จะใช้เป็นเวลาของคำสั่ง TIM มีหลายชนิดคือ

I/O, AUX Channel : 00-63

HR relay : HR00-31

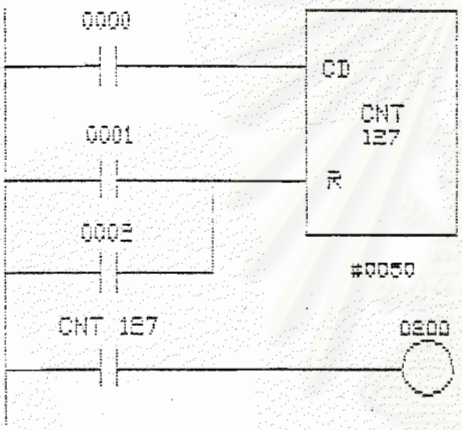
Constant : 0000-9999

เมื่อไฟดับหรือปิดเครื่องตัวตั้งเวลาจะถูกรีเซ็ทไปเริ่มต้นใหม่ ถ้าต้องการให้ยังคงสถานะข้อมูลเวลาเดิมไว้ทำงานต่อ จะต้องดัดแปลงใช้ตัวนับ (Counter) มาทำงานแทนโดยนำสัญญาณ 0.1 วินาที (I/O relay เบอร์ 6300) มาเป็นอินพุทของตัวนับ

4.5.8 คำสั่ง COUNTER [CNT(FUN09)]

คำสั่ง CNT เป็นตัวตั้งเวลาแบบนับลง (Count down) หมายถึงเมื่อมีอินพุทเข้ามาข้อมูลภายในจะถูกนับลงทีละหนึ่งจนเป็นศูนย์ หน้าสัมผัส NO ของตัวนับจะทำงานสัญญาณเข้าของตัวนับจะมี 2 สัญญาณคือ CP เป็นสัญญาณอินพุทสำหรับการนับและ R เป็นสัญญาณรีเซ็ทให้ตัวนับเริ่มทำงานใหม่

เมื่อ Counter นับลงจนข้อมูลเป็นศูนย์แล้วรีเลย์เอาท์พุทของตัวนับจะ ON ต่อเนื่องตลอดเวลาถึงแม้จะมีสัญญาณ CP เพิ่มเข้ามา จนกว่าจะมีสัญญาณรีเซ็ทเข้ามาตัวนับจึงเริ่มเซ็ทค่าจำนวนนับใหม่ และรีเลย์เอาท์พุทจะ OFF



STEP	OPCODE	OPERAND
0100	LD	0000
0101	LD	0001
0102	OR	0002
0103	CNT	127
		# 0050
0104	LD	CNT 127
0105	OUT	0200

หมายเลขของ COUNTER จะมีค่าระหว่าง 000-127 ซึ่งหมายเลขนี้ใช้ร่วมกับคำสั่ง TIMER และ COUNTER อื่น ๆ ผู้ใช้ต้องระวังหมายเลขซ้ำกัน

ข้อมูลจำนวนที่นับ และสถานะเอาท์พุทรีเลย์ของตัวนับจะยังคงค่าเดิมอยู่เมื่อไฟดับ หรือเริ่มทำงานใหม่

ข้อมูลที่จะใช้เป็นจำนวนนับของคำสั่ง CNT มีหลายชนิดคือ

I/O, AUX Channel : 00-63

HR relay : HR00-31

Constant : 0000-9999

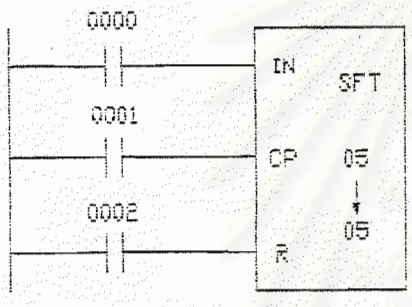
4.5.9 คำสั่ง SHIFT REGISTER [SFT (FUN10)]

เป็นคำสั่งในการเลื่อนข้อมูลครั้งละ 1 ตำแหน่ง โดยเลื่อนจากตำแหน่งต่ำไปยังตำแหน่งสูง ขนาดข้อมูลที่เลื่อนกำหนดเป็นแชลแนล ซึ่งสามารถเลื่อนมากกว่าครั้งละหนึ่งแชลแนลได้ สัญญาณของคำสั่ง SFT มี 3 สัญญาณคือ

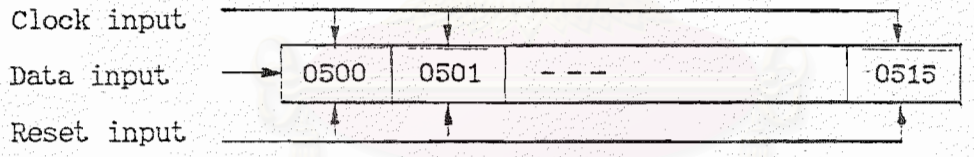
Data input (IN) เป็นข้อมูลที่จะเลื่อนเข้า

Clock input (CP) เป็นสัญญาณควบคุมการเลื่อนข้อมูล โดยใช้ขอบขาขึ้น (Leading edge)

Reset input (R) เป็นสัญญาณลบข้อมูลทั้งหมด



STEP	OPCODE	OPERAND
0300	LD	0000
0301	LD	0001
0302	LD	0002
0303	SFT	05
		05



รูปแบบคำสั่งของ SFT ประกอบด้วย

SFT D1 (Start channel number)

\- D2 (End channel number)

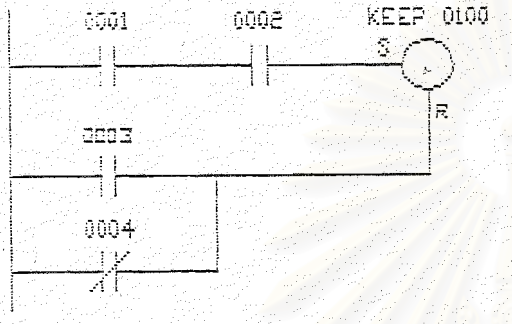
ชนิดของข้อมูลโอเปอร์แรนด์ประกอบด้วย

I/O, AUX relay : 00-60

Holding relay : HR 00-31

4.5.10 คำสั่ง KEEP (FUN11)

เป็นคำสั่งที่ใช้แทน Latchang relay คำสั่งนี้มีสัญญาณอินพุท 2 สัญญาณคือ สัญญาณเซ็ท (S) และสัญญาณรีเซ็ท (R) การทำงานของคำสั่งนี้รีเลย์เอาท์พุทจะ ON เมื่อมีสัญญาณ S เข้ามา และรีเลย์เอาท์พุท OFF เมื่อมีสัญญาณ R เข้ามา ถ้าสัญญาณ S และ R เข้ามาพร้อมกัน รีเลย์เอาท์พุทจะ OFF



STEP	OPCODE	OPERAND
0500	LD	0001
501	AND	0002
0502	LD	0003
0503	OR NOT	0004
0506	KEEP (11)	0100

รูปแบบคำสั่ง KEEP

KEEP Operand

โอเปอร์แรนด์ของคำสั่งประกอบด้วย

I/O relay : 0000-3115

Aux relay : 3200-6015

Holding relay : HR 0000-3115

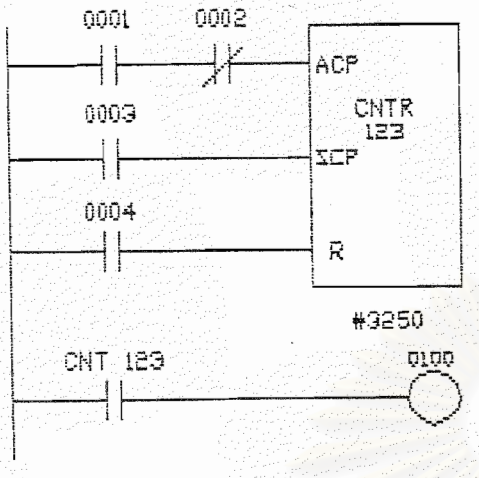
4.5.11 คำสั่ง UP-DOWN COUNTER [CNTR(FUN 12)]

เป็นคำสั่งสร้างตัวนับชนิดนับขึ้นลงได้ มีลักษณะคล้ายตัวนับแบบ CNT แต่เพิ่มสัญญาณอินพุทเข้ามาอีกหนึ่งสัญญาณ สัญญาณอินพุทจะแบ่งออกเป็น

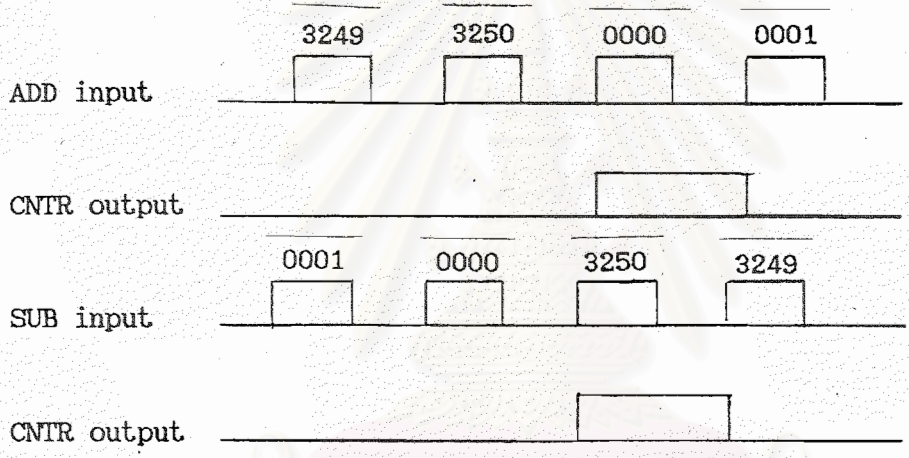
สัญญาณ ADD input (ACP) เป็นสัญญาณให้เพิ่มจำนวนนับทีละหนึ่ง

สัญญาณ SUBTRACT input (SCP) เป็นสัญญาณให้ลดจำนวนนับลงทีละหนึ่ง

สัญญาณ Reset input (R) เป็นการรีเซ็ทให้จำนวนนับเป็นศูนย์



STEP	OPCODE	OPERAND
0500	LD	0001
0501	AND-NOT	0002
0502	LD	0003
0503	LD	0004
0504	CNTR(12)	123
		# 3250
0505	LD	CNT 123
0506	OUT	0100



สัญญาณ ADD input และ SUB input เป็นแบบ Leading edge

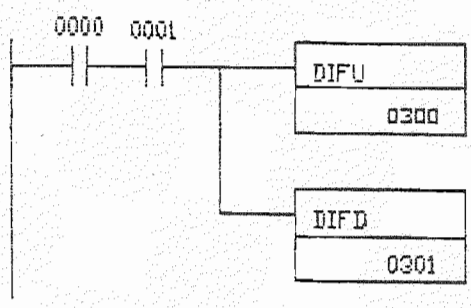
หมายเลขของตัวนับมีค่าระหว่าง 000-127 ซึ่งใช้ร่วมกับคำสั่ง TIMER และ Counter อื่น ๆ ผู้ใช้ต้องระวังไม่ให้ซ้ำกัน

ข้อมูลที่ใช้เป็นจำนวนนับของคำสั่ง CNTR ประกอบด้วย

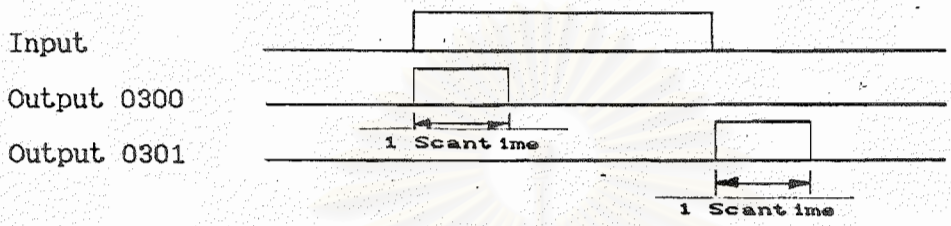
I/O, AUX channel	:	00-63
HR channel	:	HR 00-31
Constant	:	0000-9999

4.5.12 คำสั่ง DIFFERENTIATION UP [DIFU(FUN13)] และคำสั่ง DIFF[RENTIATION DOWN [DIFD(FUN14)]

คำสั่ง DIFU และ DIFD เป็นคำสั่งที่ให้เอาท์พุทเพียง 1 รอบการทำงาน (Scan time) โดยคำสั่ง DIFU จะให้เอาท์พุทที่ขอบขาขึ้น (Leading edge) ของอินพุท และคำสั่ง DIFD จะให้เอาท์พุทที่ขอบขาลง (Failing edge) ของอินพุท



STEP	OPCODE	OPERAND
0020	LD	0000
0021	AND	0001
0022	DIFU(13)	0300
0023	DIFD(14)	0301



โอเปอร์แรนด์ของคำสั่งประกอบด้วย

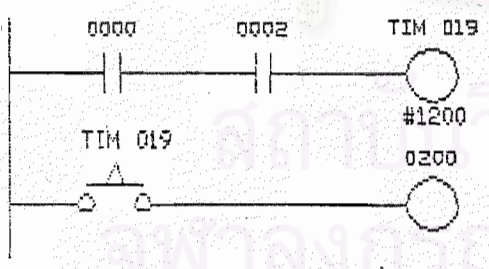
I/O, AUX relay : 0000-6015

Holding relay : HR 0000-3115

คำสั่ง DIFU และ DIFD ทั้งหมดภายในโปรแกรมมีได้ไม่เกิน 128 คำสั่ง

4.5.13 คำสั่ง HIGH-SPEED TIMER [TIMH(FUN15)]

คำสั่ง TIMH จะมีรูปแบบคำสั่ง และลักษณะการทำงานเหมือนคำสั่ง TIM ทุกอย่าง แต่คำสั่ง TIMH มีความละเอียดของเวลามากกว่าคือ มีหน่วยเป็น 0.01 วินาที

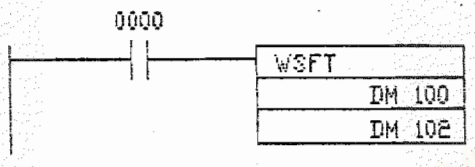


STEP	OPCODE	OPERAND
0500	LD	0000
0501	AND NOT	0001
0502	TIM	019
		# 1200
0503	LD	TIM 019
0504	OUT	0200

หมายเลขของ TIMH มีค่าระหว่าง 000-127 ซึ่งใช้ร่วมกับคำสั่ง Timer และ Counter อื่น ๆ ผู้ใช้ต้องระวังไม่ให้ซ้ำกัน

4.5.14 คำสั่ง WORD SHIFT [WSFT(FUN16)]

คำสั่ง WSFT ใช้เลื่อนข้อมูลทีละแชลแนล เมื่อทำคำสั่งนี้แล้วค่าของแชลแนลเริ่มต้นจะเป็นศูนย์



STEP	OPCODE	OPERAND
0500	LD	0000
0501	WSFT(16)	-
		DM 100
		DM 102

เมื่อค่าของรีจิสเตอร์ R เป็น 1 (สถานะ ON) คำสั่ง WSFT จะทำงานทุก ๆ รอบการทำงาน ถ้าต้องการให้ทำงานหนึ่งรอบการทำงานต้องใช้คำสั่ง DIFU และ DIFD ช่วย

รูปแบบของคำสั่ง WSFT

WSFT -
- D1 : Start channel
- D2 : End channel

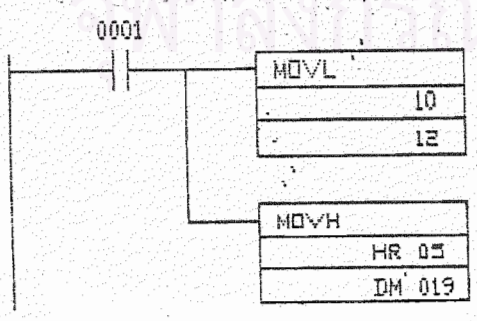
ชนิดของข้อมูลโอเปอร์แรนด์ประกอบด้วย

I/O, AUX channel	:	00-63
Holding relay	:	HR 00-31
Data memory	:	DM 000-511

4.5.15 คำสั่ง MOVE TO LOW BYTE [MOVL(FUN17)] และคำสั่ง MOVE TO H16H BYTE [MOVH(FUN18)]

คำสั่ง MOVL เป็นการคัดลอกข้อมูลไบท์สูง (บิท 8-15) ไปยังไบท์ต่ำ (บิท 0-7) ตามตำแหน่งที่ระบุในโอเปอร์แรนด์

คำสั่ง MOVH เป็นการคัดลอกข้อมูลไบท์ต่ำ (บิท 0-7) ไปยังไบท์สูง (บิท 8-15) ตามตำแหน่งที่ระบุในโอเปอร์แรนด์



STEP	OPCODE	OPERAND
0500	LD	0001
0501	MOVL(17)	-
		00
		12
0502	MOVH(18)	-
		HR05
		DM019

รูปแบบของคำสั่ง MOVL/MOVH -

\- S1 : Transfer data

D1 : Tranfer destinating CH NO.

ชนิดข้อมูลของโอเปอร์แรนด์

	S1	D1
I/O, AUX relay	: 00-63	: 00-60
Holding relay	: HR 00-31	: HR 00-31
Data memory	: DM 000-511	: DM 000-511

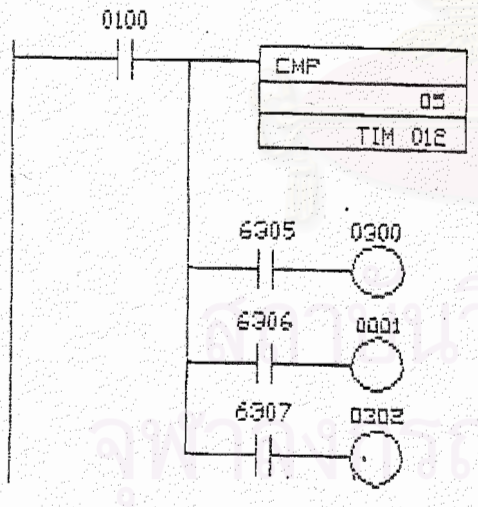
4.5.16 คำสั่ง COMPARE [CMP(FUN20)]

คำสั่ง CMP ใช้สำหรับเปรียบเทียบข้อมูล 2 แชลแนล ในลักษณะเลขไบนารี 16 บิท ซึ่งผลลัพธ์ของการเปรียบเทียบคือ

รีเลย์ 6305 จะมีสถานะ ON ถ้า S1 มีค่ามากกว่า S2

รีเลย์ 6306 จะมีสถานะ ON ถ้า S1 มีค่าเท่ากับ S2

รีเลย์ 6307 จะมีสถานะ ON ถ้า S1 มีค่าน้อยกว่า S2



STEP	OPCODE	OPERAND
0500	LD	0001
0501	OUT	TR0
0502	CMP(20)	-
		05
		TIM02
0503	LD	TR0
0504	AND	6305
0505	OUT	0300
0506	LD	TR0
0507	AND	6306
0508	OUT	0301
0509	LD	TR0
0510	AND	6307
0511	OUT	0302

รูปแบบคำสั่ง CMP –

S1 : Compare data 1

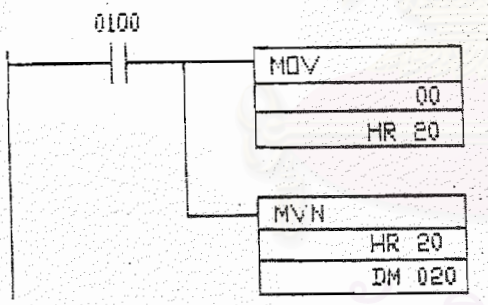
S2 : Compare data 2

ชนิดของข้อมูลโอเปอร์แรนด์ S1 และ S2

I/O, AUX relay	:	00-63
Holding relay	:	HR 00-31
Timer/counter	:	TIM/CNT 000-127
Data memory	:	DM 000-511
Constant	:	0000-FFFF

4.5.17 คำสั่ง MOVE[FUN21)] และ MOVE NOT [MVN(FUN21)]

คำสั่ง MOV เป็นการคัดลอกข้อมูลขนาด 16 บิทจากแชลแนลต้นทางไปยังแชลแนลปลายทาง โดยมีตำแหน่งระบุไว้ในโอเปอร์แรนด์ คำสั่ง MVN ทำงานเหมือนคำสั่ง MOV แต่จะ Inverse ข้อมูลก่อน



STEP	OPCODE	OPERAND
0200	LD	0100
0201	MOV	–
		00
		HR 20
0202	MVN	–
		HR 20
		DM 020

รูปแบบคำสั่ง MOV/MVN –

S : Transfer data

D : Destination channel

ชนิดของข้อมูลของโอเปอร์แรนด์

	S	D
I/O, AUX relay	: 00-63	: 00-60
Holding relay	: HR 00-31	: HR 00-31
Timer/counter	: TIM/CNT 000-127	: -
Data memory	: DM 000-511	: DM 000-511
Constant	: 0000-FFFF	: -

4.5.18 คำสั่ง BCD-TO-BIN CONVERTION [BIN(FUN23)]

คำสั่ง BIN ใช้ในการแปลงข้อมูลฐานสิบขนาด 4 หลักไปเป็นเลขไบนารีขนาด 16 บิท ตำแหน่งต่าง ๆ ของข้อมูลระบุที่โอเปอร์แรนด์

รูปแบบคำสั่ง BIN -
- S : Conversion data
- D : Destination Channel

ชนิดข้อมูลของโอเปอร์แรนด์

	S	D1
I/O, AUX relay	: 00-63	: 00-60
Holding relay	: HR 00-31	: HR 00-31
Timer/Counter	: TIM/CNT 000-127	: -
Data memory	: DM 000-511	: DM 000-511

ถ้าผลลัพธ์ของคำสั่ง BIN เป็นศูนย์สถานะของรีเลย์ 6306 จะเป็น ON
ถ้าข้อมูลอินพุทไม่ใช่เลข BCD คำสั่งนี้จะไม่ทำงาน และรีเลย์ 6303 จะมีสถานะ ON

4.5.19 คำสั่ง BIN-TO-BCD CONVERSION [BCD(FUN24)]

คำสั่ง BCD ใช้แปลงข้อมูลไบนารีขนาด 16 บิท ไปเป็นเลข BCD ขนาด 4 หลัก ตำแหน่งของข้อมูลต่าง ๆ ระบุที่โอเปอร์แรนด์

รูปแบบคำสั่ง BCD -

- S : Conversion data
- D : Destination channel

ชนิดข้อมูลของโอเปอร์แรนด์

	S	D1
I/O, AUX relay	: 00-63	: 00-60
Holding relay	: HR 00-31	: HR 00-31
Data memory	: DM 000-511	: DM 000-511

ถ้าผลลัพธ์ของคำสั่งเป็นศูนย์รีเลย์ 6306 จะมีสถานะ ON

ถ้าผลลัพธ์ของคำสั่งมีค่ามากกว่า 9999 รีเลย์ 6303 จะมีสถานะ ON

4.5.20 คำสั่ง ASL (FUN25), ASR(FUN26), ROL(FUN27), ROR(FUN28) และ COM(FUN29)

รูปแบบคำสั่ง Function code -

- D : Data channel

ชนิดข้อมูลของโอเปอร์แรนด์

I/O, AUX relay	: 00-60
Holding relay	: HR 00-31
Data memory	: DM 000-511

การทำงานของคำสั่ง

ASL : Arithmetic shift left เป็นคำสั่งเลื่อนข้อมูลจากซ้ายไปขวาหนึ่งตำแหน่ง (บิท) เมื่อทำคำสั่งนี้แล้วข้อมูลบิท 0 จะมีค่าเป็น "0" และข้อมูลเดิมของบิท 15 จะไปเก็บที่รีเลย์ตัวทด (6304)

ASR : Arithmetic shift right เป็นคำสั่งเลื่อนข้อมูลจากขวาไปซ้ายหนึ่งตำแหน่ง (บิท) เมื่อทำคำสั่งนี้แล้วข้อมูลบิท 15 จะมีค่าเป็น "0" และข้อมูลเดิมของบิท 0 จะนำไปเก็บที่รีเลย์ตัวทด (Carry)

ROL : Rotate left เป็นคำสั่งหมุนข้อมูลผ่านรีเลย์ตัวทด (6304) ในทิศทางซ้ายมือ ซึ่งจะทำให้ข้อมูลบิท 15 เลื่อนไปอยู่ในรีเลย์ตัวทด (6304) และข้อมูลรีเลย์ตัวทดจะเลื่อนไปยังบิท 0

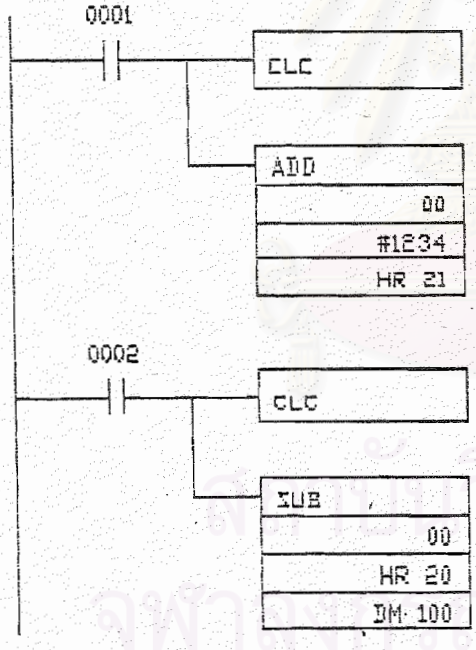
ROR : Rotate right เป็นคำสั่งหมุนข้อมูลทางขวาผ่านรีเลย์ตัวทด (6304) การทำงานจะทำให้ข้อมูลตัวทด (6304) เลื่อนไปยังบิท 15 และข้อมูลบิท 0 เลื่อนไปยังรีเลย์ตัวทด

COM : Complement เป็นคำสั่งทำ Inverse ข้อมูล โดยเปลี่ยนค่าทุกบิทของข้อมูลจาก 0 เป็น 1 และจาก 1 เป็น 0

4.5.21 คำสั่ง ADD (FUN30) และ SUBTRACT [SUB(FUN31)]

คำสั่ง ADD เป็นการบวกเลข BCD 4 หลัก 2 จำนวน โดยจำนำค่า S1 มาบวกกับ S2 และบวกรีเลย์ตัวทด (6304) นำผลลัพธ์ไปเก็บที่ D

คำสั่ง SUB เป็นการลบเลข BCD 4 หลัก 2 จำนวน โดยนำค่า S1 มาลบด้วย S2 และลบกับรีเลย์ตัวทด (6304) นำผลลัพธ์ไปเก็บที่ D



STEP	OPCODE	OPERAND
0500	LD	0001
0501	CLC	-
0502	ADD	-
		00
		# 1234
		HR 21
0503	LD	0002
0504	CLC	-
0505	SUB	-
		00
		HR 20
		DM 100

รูปแบบคำสั่ง ADD/SUB -

\- S1 : Data 1

\- S2 : Data 2

\- D : Result channel

ชนิดข้อมูลของโอเปอร์แรนด์

	S1, S2	D
I/O, AUX relay	: 00-63	: 00-60
Holding relay	: HR 00-31	: HR 00-31
Timer/counter	: T/C 000-127	: -
Data memory	: DM 000-511	: DM 000-511
Constant	: 0000-9999	: -

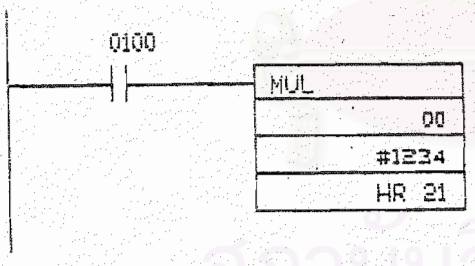
ก่อนทำคำสั่งซีพียูจะตรวจสอบข้อมูลถ้าไม่ใช่เลข BCD รีเลย์ 6303 จะมีสถานะ ON และคำสั่งนี้จะไม่ทำงาน

ถ้าผลลัพธ์ของการทำคำสั่ง ADD หรือ SUB เป็นศูนย์รีเลย์ 6306 จะมีสถานะ ON

ถ้าผลลัพธ์ของคำสั่ง ADD มากกว่า 9999 หรือผลลัพธ์ของคำสั่ง SUB มีค่าน้อยกว่า 0000 รีเลย์ 6304 (Carry) จะมีสถานะ ON

4.5.22 คำสั่ง MULTIPLY [MUL(FUN 32)]

MUL เป็นการคูณเลข BCD 4 หลักที่เก็บใน S1 และ S2 ผลลัพธ์ที่ได้เป็น BCD ขนาด 8 หลัก (2 Channel) ซึ่งถูกเก็บไว้ที่ D และแชลแนลถัดไป



STEP	OPCODE	OPERAND
0000	LD	0100
0001	MUL	-
		00
		# 1234
		HR 21

รูปแบบคำสั่ง MUL -

- S1 : Data 1
- S2 : Data 2
- D : Result channel

ชนิดข้อมูลของโอเปอร์แรนด์

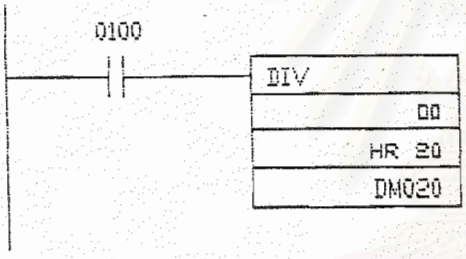
	S1, S2	D
I/O, AUX relay	: 00-63	: 00-60
Holding relay	: HR 00-31	: HR 00-31
Timer/counter	: T/C 000-127	: -
Data memory	: DM 000-511	: DM 000-511
Constant	: 0000-9999	: -

ถ้าผลลัพธ์ของการคำนวณเป็น 0 รีเลย์ 6306 จะมีสถานะ ON

ถ้าข้อมูลที่นำมาคูณไม่ใช่เลข BCD คำสั่ง MUL จะไม่ทำงาน และรีเลย์ 6303 จะ ON

4.5.23 คำสั่ง DIVIDE [DIV (FUN 33)]

คำสั่ง DIV จะนำค่า S1 หารด้วยค่าของ S2 นำผลลัพธ์ไปเก็บที่ D และนำเศษของการหารไปเก็บที่ตำแหน่งถัดไป



STEP	OPCODE	OPERAND
0000	LD	0100
0001	DIV	-
		00
		HR 20
		DM 020

รูปแบบคำสั่ง DIV -

\- S1 : Data 1

\- S2 : Divide data

\- D : Result channel

ชนิดข้อมูลของโอเปอร์แรนด์

	S1, S2	D
I/O, AUX relay	: 00-63	: 00-60
Holding relay	: HR 00-31	: HR 00-31
Timer/counter	: T/C 000-127	: -
Data memory	: DM 000-511	: DM 000-511
Constant	: 0000-9999	: -

ซีพียูจะตรวจสอบข้อมูล S1 และ S2 ถ้ามีความผิดพลาดจะเซ็ทรีเลย์ 6303 ให้ ON และไม่ทำคำสั่ง DIV นี้

4.5.24 คำสั่ง ANDW (FUN 34), ORW (FUN 35), XORW (FUN 36), XNRW (FUN 37)

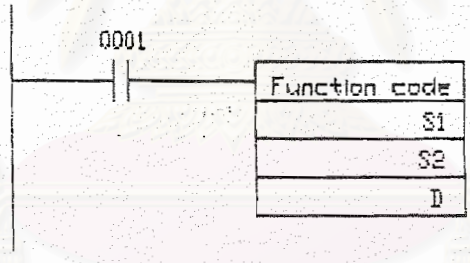
คำสั่งเหล่านี้เป็นคำสั่งเกี่ยวกับการทำลอจิกของข้อมูล

รูปแบบของคำสั่ง Function code -

- S1 : Data 1
- S2 : Data 2
- D : Result channel

ชนิดข้อมูลของโอเปอร์แรนด์

	S1, S2	D
I/O, AUX relay	: 00-63	: 00-60
Holding relay	: HR 00-31	: HR 00-31
Timer/counter	: T/C 000-127	: -
Data memory	: DM 000-511	: DM 000-511
Constant	: 0000-FFFF	: -



ANDW : And word เป็นการทำลอจิก AND ระหว่าง S1 และ S2 ผลลัพธ์ไปเก็บที่ D

ORW : Or word เป็นการทำลอจิก OR ระหว่าง S1 และ S2 ผลลัพธ์ไปเก็บที่ D

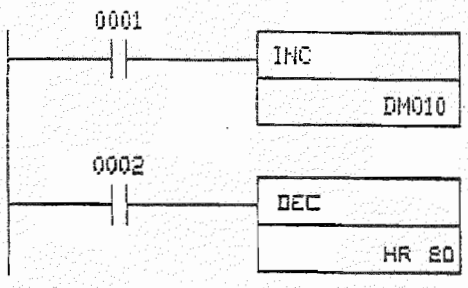
XORW : Exclusive word เป็นการทำลอจิก XOR ระหว่าง S1 และ S2 ผลลัพธ์ไปเก็บที่ D

ถ้าผลลัพธ์ของการทำคำสั่งเหล่านี้เป็น 0 รีเลย์ 6306 จะเซ็ทเป็น ON

4.5.25 คำสั่ง INCREMENT [INC(FUN 38)] และคำสั่ง DECREMENT [DEC (FUN 39)]

คำสั่ง INC จะเพิ่มค่าของข้อมูลอีกหนึ่ง คำสั่ง DEC เป็นการลดค่าของข้อมูลลงหนึ่ง ถ้าผลลัพธ์การทำงานของคำสั่ง INC หรือ DEC เป็นศูนย์รีเลย์ 6306 จะถูกเซ็ทเป็น ON

ในการทำงานซีพียูจะตรวจสอบค่าข้อมูล ถ้าไม่ใช่รหัส BCD จะเซ็ทรีเลย์ 6303 เป็น ON และจะไม่ทำคำสั่งนี้ด้วย



STEP	OPCODE	OPERAND
0000	LD	0001
0001	INC	-
		DM 010
0002	DEC	-
		HR 20

รูปแบบคำสั่ง INC/DEC -

\- D : Data channel

ชนิดข้อมูลของโอเปอร์แรนด์

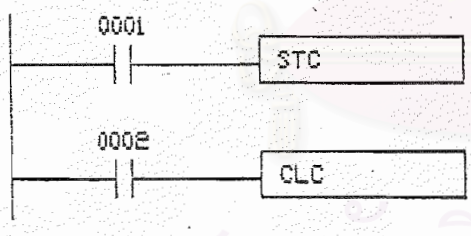
I/O, AUX relay : 00-60

Holding relay : HR 00-31

Data memory : DM 000-511

4.5.26 คำสั่ง SET CARRY [STC(FUN 40)] และคำสั่ง CLEAR CARRY [CLC (FUN 41)]

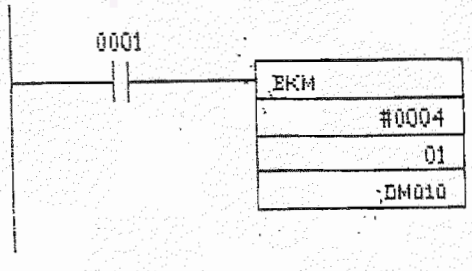
ใช้คำสั่ง STC สำหรับเซ็ทรีเลย์ตัวทด (6304) ให้เป็น "1" และคำสั่ง CLC สำหรับรีเซ็ทรีเลย์ตัวทด (6304) ให้เป็น "0" คำสั่งเหล่านี้จะใช้ร่วมกับคำสั่งการคำนวณ



STEP	OPCODE	OPERAND
0100	LD	0001
0101	STC	-
0102	LD	0002
0103	CLC	-

4.5.27 คำสั่ง BLOCK MOVE [BKM(FUN 42)]

เป็นคำสั่งย้ายข้อมูลครั้งละหลาย ๆ แชลแนล



STEP	OPCODE	OPERAND
0000	LD	0001
0001	BKM (42)	-
		# 0004
		01
		DM 010

รูปแบบคำสั่ง BKM -

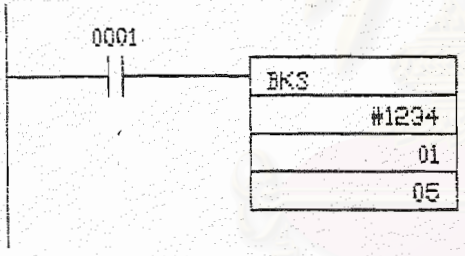
- W : No. of channel
- S : Start channel
- D : Destination channel

ชนิดข้อมูลของโอเปอร์แรนด์

	W	S	D
I/O, AUX relay	:	: 00-63	: 00-60
Holding relay	:	: HR 00-31	: HR 00-31
Timer/counter	:	: T/C 000-127	: T/C 000-127
Data memory	:	: DM 000-511	: DM 000-511
Constant	: 0000-0511	: -	: -

4.5.28 คำสั่ง BLOCK SET [BKS(FUN 43)]

เป็นคำสั่งกำหนดค่าของข้อมูลครั้งละหลาย ๆ แชลแนล



STEP	OPCODE	OPERAND
0000	LD	0001
0001	BKS	-
		# 1234
		01
		05

รูปแบบคำสั่ง BKS -

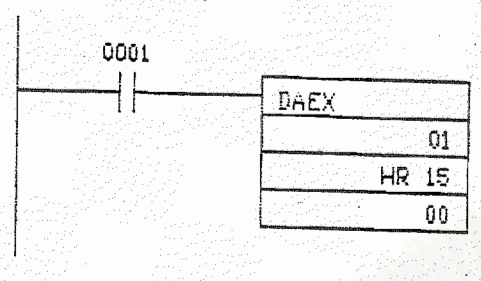
- S : Setting data
- D1 : Start channel
- D2 : End channel

ชนิดข้อมูลของโอเปอร์แรนด์

	S	D1, D2
I/O, AUX relay	: 00-63	: 00-60
Holding relay	: HR 00-31	: HR 00-31
Timer/counter	: T/C 000-127	: T/C 000-127
Data memory	: DM 000-511	: DM 000-511
Constant	: 0000-FFFF	: -

4.5.29 คำสั่ง DATA EXCHANGE [DAEX(FUN 44)]

คำสั่ง DAEX เป็นสลับข้อมูลของแชลแนลหนึ่งกับแชลแนลอื่น



STEP	OPCODE	OPERAND
0000	LD	0001
0001	DAEX	-
		01
		HR 15
		00

รูปแบบคำสั่ง DAEX -

\- D1 : Data 1

\- D2 : Data 2

\- -

ชนิดข้อมูลของโอเปอร์แรนด์

I/O, AUX relay : 00-60

Holding relay : HR 00-31

Data memory : DM 000-511

4.5.30 คำสั่ง ONE DIGIT SHIFT LEFT [ODSL(FUN 45)] และ
คำสั่ง ONE DIGIT SHIFT RIGHT [ODSR(FUN 46)]

รูปแบบคำสั่ง ODSL/ODSR -

\- D1 : Start channel

\- D2 : End channel

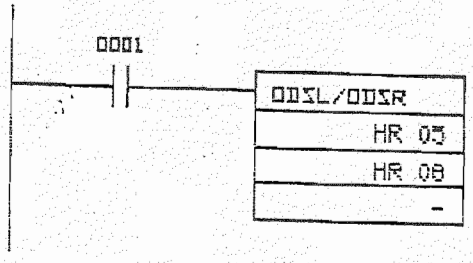
\- -

ชนิดข้อมูลของโอเปอร์แรนด์ D1, D2

I/O, AUX relay : 00-60

Holding relay : HR 00-31

Data memory : DM 000-511



คำสั่ง ODSL จะเป็นการเลื่อนข้อมูลระหว่าง D1 ถึง D2 ไปทางซ้ายมือครั้งละ 4 บิท และนำค่า 0000 ใส่ที่บิท 0-3 ของ D1

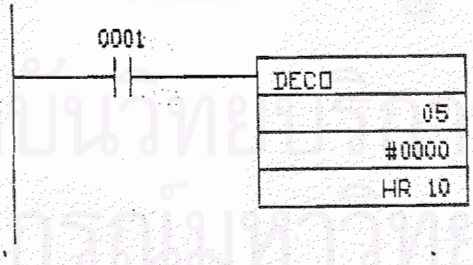
คำสั่ง ODSR เป็นการเลื่อนข้อมูลระหว่าง D1 ถึง D2 ไปทางขวามือครั้งละ 4 บิท และนำค่า 0000 ใส่ที่บิท 15-12 ของ D2

4.5.31 คำสั่ง 4-TO-16 DECODER [DECO(FUN 47)], คำสั่ง 16-TO-4 ENCODER [ENCO(FUN 48)], คำสั่ง 7-SEGMENT DECODER [7 SEG(FUN 49)]

รูปแบบคำสั่ง Function code -
- S : Conversion data
- K : Digit No.
- D : Result channel

ชนิดข้อมูลของโอเปอร์แรนด์

	S	K	D
I/O, AUX relay	00-63	-	00-60
Holding relay	HR 00-31	-	HR 00-31
Timer/counter	T/C 000-127	-	T/C 000-127
Data memory	DM 000-511	-	DM 000-511
Constant	-	0000-0003	-



คำสั่ง DECO เป็นการดีโค้ดค่าไบนารีของข้อมูล S ณ หลัก (Digit) ที่กำหนดโดย K และเก็บผลลัพธ์ไว้ที่ D

คำสั่ง ENCO เป็นการหาค่าตำแหน่งบิทสูงสุดของข้อมูล "1" ของ S แล้วแปลงค่าเป็นเลขไบนารีเก็บลงที่ D ณ หลัก (Digit) ที่กำหนดโดย K

คำสั่ง 7SEG เป็นการแปลงค่าไบนารีของข้อมูล S ณ หลัก (Digit) ที่กำหนดโดย K ไปเป็นรหัส 7 Segment เก็บไว้ที่ไบท์ต่ำของ D

บทที่ 5

การสร้างเครื่องควบคุม และการทดสอบ

การสร้างเครื่อง PC สามารถแบ่งออกเป็นขั้นตอนที่สำคัญได้ 2 อย่างคือ การสร้างฮาร์ดแวร์ และการพัฒนาซอฟท์แวร์

5.1 การสร้างฮาร์ดแวร์

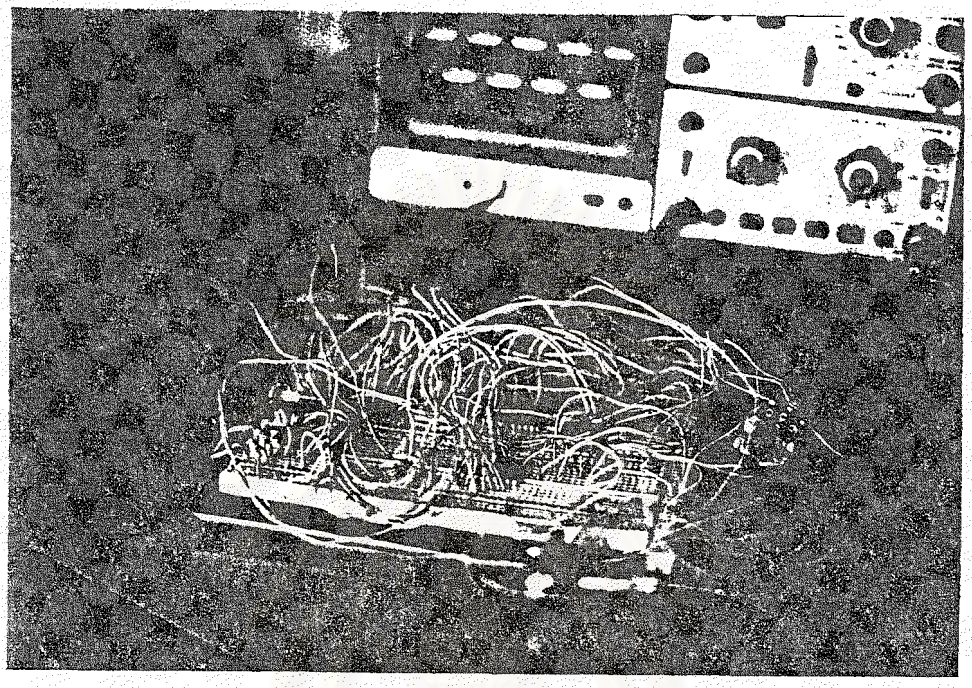
การสร้างฮาร์ดแวร์ของเครื่อง PC ก็เหมือนวงจรอีเลคทรอนิคส์อื่น ๆ คือ ต้องมีการทดลองวงจรมีการแก้ไขปรับปรุงวงจรเพื่อให้มีเสถียรภาพดีที่สุด

การสร้างฮาร์ดแวร์ของเครื่อง PC ที่ออกแบบนี้เริ่มจากการออกแบบวงจรต่าง ๆ ในกระดาษก่อน โดยเริ่มทำวงจรส่วนที่สำคัญก่อนคือ โมดูลประมวลผล โมดูลแสดงผล และคีย์บอร์ด และตามด้วยโมดูลอินพุท/เอาท์พุทต่าง ๆ การออกแบบนี้จะต้องตรวจสอบอุปกรณ์ต่าง ๆ ที่ใช้ว่าสามารถทำงานร่วมกันได้ เช่น ความเร็ว, ระดับของสัญญาณต่าง ๆ

5.1.1 การทดลองฮาร์ดแวร์

จากวงจรที่ออกแบบแล้วควรมีการทดลองก่อนที่จะนำไปสร้างวงจรจริง เพื่อให้แน่ใจว่าวงจรสามารถทำงานได้ถูกต้อง ถ้ามีวงจรส่วนใดทำงานผิดพลาดจะได้แก้ไขให้ถูกต้องเสียก่อน การทดลองฮาร์ดแวร์ของเครื่อง PC ที่ออกแบบนี้จะใช้การทดลองในแผ่นโฟโต้บอร์ด โดยเลือกทดลองวงจรเฉพาะส่วนที่ยุ่งยาก หรือวงจรที่อาจมีปัญหาในการทำงาน สำหรับวงจรพื้นฐานหรือวงจรที่เคยใช้งานมาแล้ว บางส่วนอาจนำไปออกแบบวงจรจริงเลยได้

การทดลองวงจรฮาร์ดแวร์นี้ไม่อาจทดลองพร้อมกันได้ทั้งหมด เนื่องจากวงจรทั้งระบบมีขนาดใหญ่ ซึ่งสามารถทำได้โดยทดลองวงจรส่วนที่สำคัญแล้วนำไปสร้างวงจรจริง แล้วจึงทดลองวงจรส่วนอื่นโดยต่อเชื่อมกับวงจรที่สร้างเสร็จแล้ว

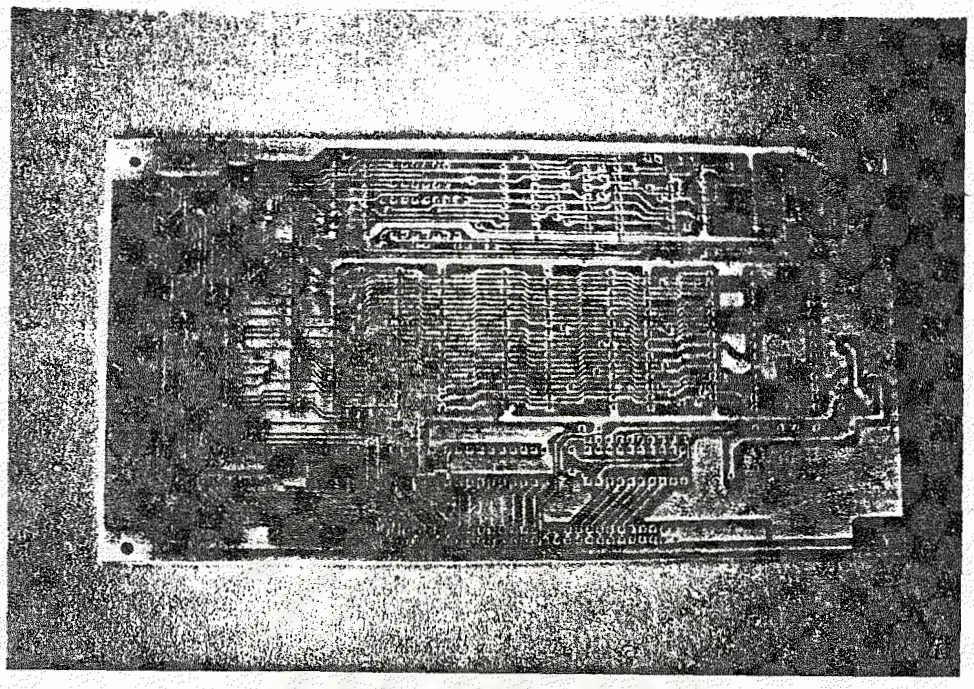


รูปที่ 5.1 แสดงรูปวงจรที่ทดลองฮาร์ดแวร์

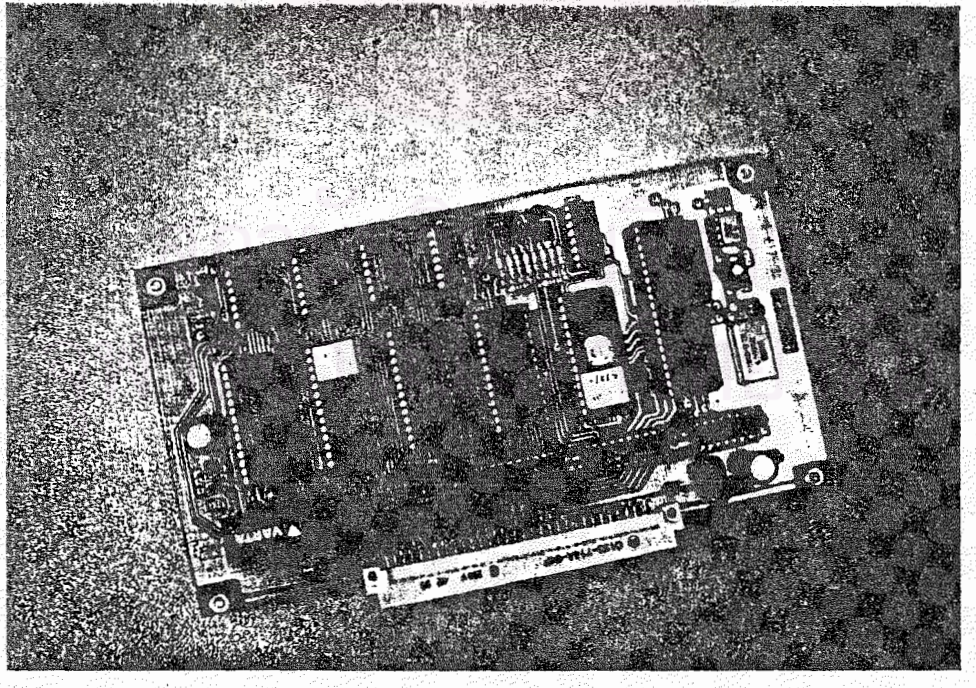
5.1.2 การออกแบบแผ่นวงจรพิมพ์

เมื่อทดลองการทำงานของฮาร์ดแวร์ที่ทดลองจนเครื่องสามารถทำงานได้ถูกต้องแล้ว ขั้นต่อไปคือการออกแบบแผ่นวงจรพิมพ์ ซึ่งการออกแบบแผ่นวงจรพิมพ์ต้องคำนึงถึงวัตถุประสงค์การใช้งานของเครื่องด้วย เนื่องจากเครื่อง PC จะต้องใช้งานสำหรับการควบคุมในโรงงานอุตสาหกรรมเป็นส่วนใหญ่ ซึ่งการทำงานภายใต้สภานแวดล้อมเช่นนั้น ต้องคำนึงถึงสัญญาณรบกวนเป็นสำคัญ การออกแบบแผ่นวงจรพิมพ์นั้นจะต้องพยายามให้ป้องกันการรบกวนจากสัญญาณภายนอกให้มากที่สุด เช่น การจัดกลุ่มของอุปกรณ์ที่ติดต่อสัญญาณภายนอก กับกลุ่มอุปกรณ์ในการประมวลผลต้องพยายามให้แยกกัน การทำ Ground plane เพื่อช่วยลดการเหนี่ยวนำของสัญญาณรบกวน หรือการต่อคาปาซิเตอร์คับปลิ้งสัญญาณรบกวนต้องพยายามต่อให้ใกล้กับตัวอุปกรณ์ การวางตำแหน่งของอุปกรณ์พยายามให้สายสัญญาณต่าง ๆ สั้นที่สุด

การออกแบบแผ่นวงจรพิมพ์ของเครื่อง PC ที่สร้างนี้ จะใช้ซอฟท์แวร์ Smart work ช่วยในการออกแบบ ซึ่งทำให้การแก้ไขปรับปรุงต่าง ๆ ทำได้สะดวกรวดเร็ว



รูปที่ 5.2 แสดงแผ่นวงจรพิมพ์ที่ออกแบบ



รูปที่ 5.3 แสดงแผ่นวงจรพิมพ์ที่ประกอบวงจรแล้ว

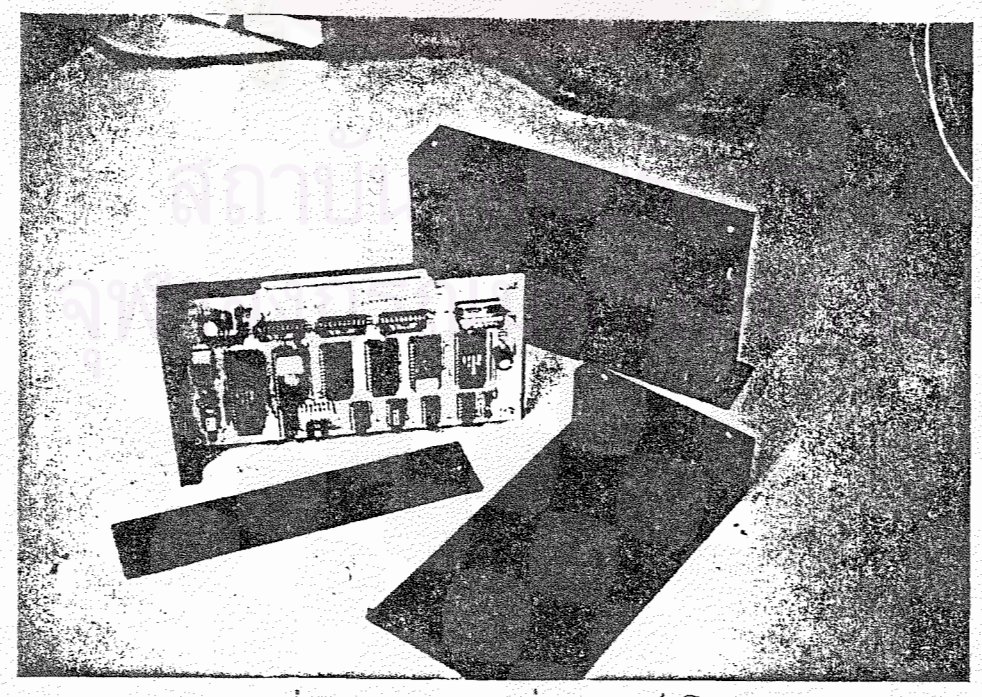
5.1.3 การประกอบวงจร และทดสอบ

เมื่อออกแบบแผ่นวงจรพิมพ์เสร็จแล้วก็นำไปทำแผ่นวงจรพิมพ์สำหรับประกอบวงจร โดยก่อนที่จะประกอบวงจรจะต้องตรวจสอบความถูกต้องของแผ่นวงจรพิมพ์ด้วย เช่น มีสายสัญญาณขาด หรือมีส่วนของวงจรที่ต่อกัน (Short) หรือไม่ ถ้ามีก็แก้ไขปรับปรุงให้เรียบร้อย การประกอบวงจรควรเลือกอุปกรณ์ที่คุณภาพ และต้องแน่ใจว่าอุปกรณ์ที่ใส่ลงไปนั้นสามารถทำงานได้ การประกอบวงจรนั้นก็เป็นส่วนที่สำคัญต่อเสถียรภาพของเครื่องด้วย ต้องให้ความระมัดระวังด้วย เช่น ถ้ามีการบัดกรีไม่ดี (หลวม) เวลานำไปใช้งานอาจมีปัญหาทำงานได้บ้างไม่ได้บ้าง

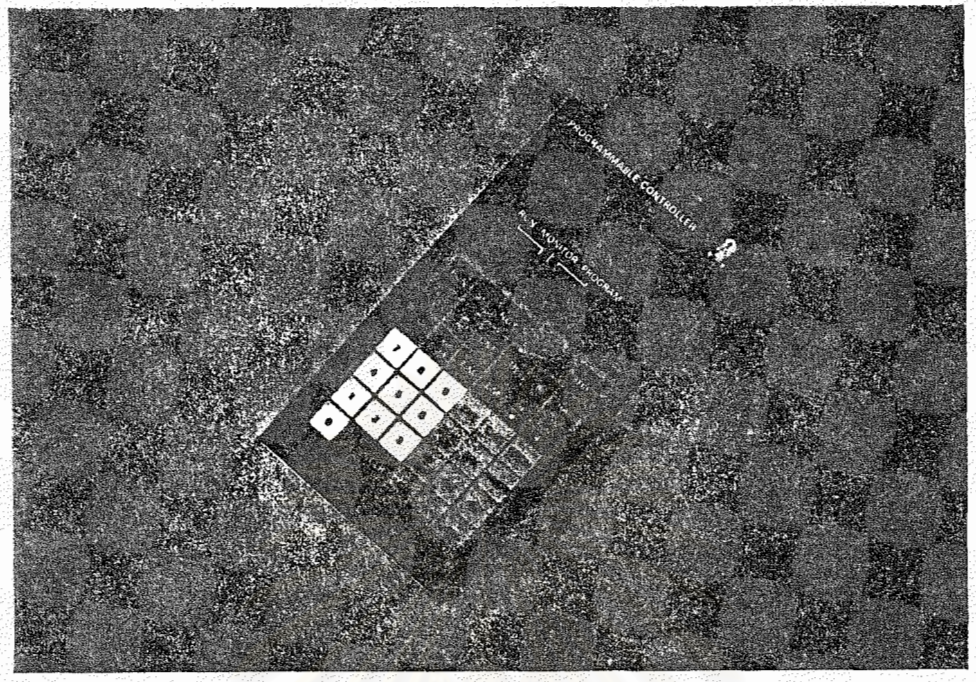
หลังจากที่ประกอบวงจรเสร็จแล้ว ก็ต้องมีการทดสอบว่าวงจรนั้นสามารถทำงานได้ถูกต้องหรือไม่ ซึ่งจะทดสอบโดยใช้อิมูเลเตอร์ต่อกับวงจรที่สร้าง และส่งคำสั่งจากเครื่องคอมพิวเตอร์ IBM ไปยังอิมูเลเตอร์ เพื่อทดสอบการทำงานของวงจรส่วนต่าง ๆ ว่าทำงานได้ถูกต้องหรือไม่ บางครั้งการทดสอบฮาร์ดแวร์บางส่วนจำเป็นต้องเขียนซอฟท์แวร์ขึ้นมา เพื่อช่วยในการทดสอบด้วย เช่น การอินเตอร์รัพท์

เมื่อประกอบวงจรต่าง ๆ ของเครื่อง PC เสร็จแล้วงานขั้นต่อไปคือ การพัฒนาซอฟท์แวร์โปรแกรมควบคุม ซึ่งจะใช้อิมูเลเตอร์ในการพัฒนาและทดลองโปรแกรมส่วนต่าง ๆ หลังจากที่พัฒนาโปรแกรมเสร็จแล้ว จะต้องนำโปรแกรมไปอัดลงหน่วยความจำ EPROM แล้วจึงนำไปประกอบกับเครื่องให้สมบูรณ์

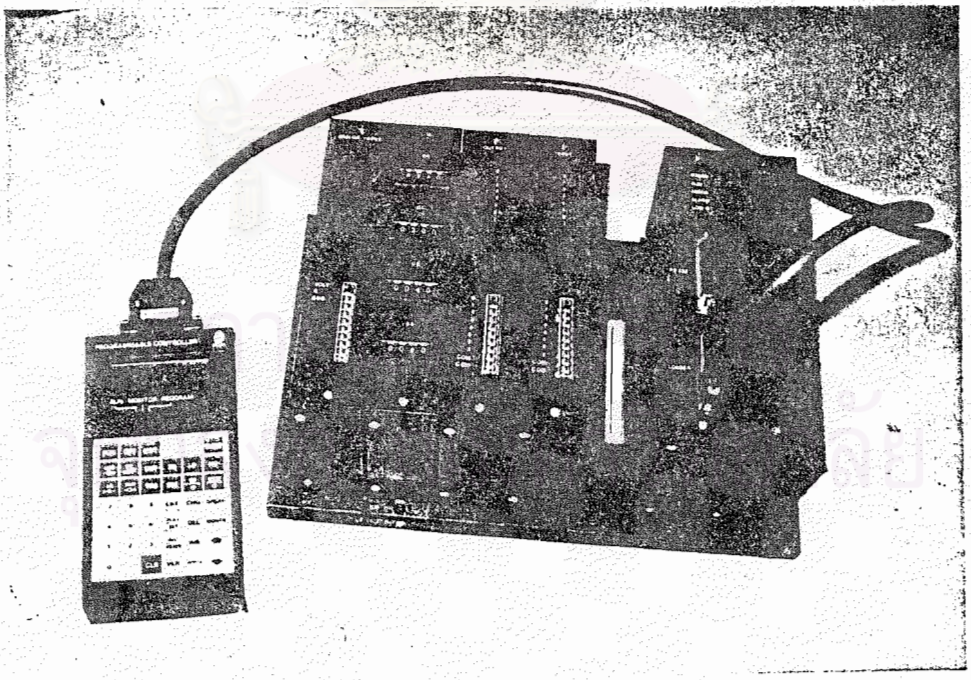
หลังจากนั้นก็ประกอบฮาร์ดแวร์ส่วนต่าง ๆ ลงกล่องที่ออกแบบไว้แล้ว นำไปทดสอบการใช้งานต่อไป



รูปที่ 5.4 แสดงวงจรที่ประกอบเป็นโมดูล



รูปที่ 5.5 แสดงรูปตัวป้อนโปรแกรมแบบมือถือ



รูปที่ 5.6 แสดงเครื่อง PC ที่สร้างเสร็จสมบูรณ์

5.2 การพัฒนาซอฟท์แวร์

การพัฒนาซอฟท์แวร์โปรแกรมควบคุมการทำงานของเครื่อง PC แบ่งออกเป็น 2 ขั้นตอนคือ

5.2.1 การเขียนโปรแกรม

เป็นการเขียนโปรแกรมภาษาแอสเซมบลี้ที่ใช้ควบคุมเครื่อง มีขั้นตอนคือ

1. ออกแบบโปรแกรม และเขียนโฟล์วชาร์ทในกระดาษ

2. เขียน และป้อนโปรแกรมควบคุมภาษาแอสเซมบลี้โดยใช้โปรแกรม Text editor ช่วย ซึ่งก็จะได้เป็น Text file โมดูลต่าง ๆ เก็บไว้

3. คอมพายล์โปรแกรม Text file ที่มีนามสกุล .SRC ให้เป็น Object program โดยโปรแกรม Z80ASM ในการคอมพายล์ ซึ่งถ้ามี Error เกิดขึ้นก็กับไปแก้ไขในขั้นตอนที่ 2 ใหม่

4. ลิ๊งค์ (Link) Object program ต่าง ๆ ที่ได้ให้ต่อกันเป็น FILE.ASM โดยใช้โปรแกรม LODZ80

5.2.2 การทดสอบโปรแกรม

ในการเขียนโปรแกรมควบคุมเครื่อง ไม่สามารถเขียนโปรแกรมครั้งเดียวทั้งหมดได้ จะต้องมีการเขียนเป็นส่วน ๆ และทดสอบการทำงานว่าถูกต้องหรือไม่ ถ้าไม่ถูกต้องให้กลับไปแก้ไข ถ้าถูกต้องแล้วก็เขียนโปรแกรมส่วนต่อไป หลังจากที่เขียนได้ทั้งหมดแล้วก็นำส่วนต่าง ๆ มาเชื่อมต่อกัน แล้วทดสอบการทำงานทั้งหมดอีกครั้ง

การทดสอบการทำงานของโปรแกรมจะใช้เครื่องอิมูเลเตอร์ช่วยในการทดสอบ โดยส่งคำสั่งทดสอบต่าง ๆ ผ่านเครื่องคอมพิวเตอร์ IBM ไปยังอิมูเลเตอร์ แล้วดูสถานะการทำงานต่าง ๆ ว่าโปรแกรมควบคุมทำงานถูกต้องหรือไม่ การใช้อิมูเลเตอร์นี้ทำให้สามารถแก้ไข และตรวจสอบการทำงานของโปรแกรมได้เป็นอย่างดี การทดสอบการทำงานของโปรแกรมควบคุมแล้วพบข้อผิดพลาดจะต้องกลับไปขั้นตอนการเขียนโปรแกรมแล้วแก้ไขโปรแกรมใหม่ ซึ่งบางครั้งการแก้ไขอาจต้องเขียนโฟลว์ชาร์ทใหม่เลย

5.3 การทดสอบเครื่อง

การทดสอบการทำงานของเครื่อง จะแบ่งเป็น 2 ส่วนใหญ่ ๆ คือการทดสอบว่าเครื่องทำงานตามคำสั่งและฟังก์ชั่นต่าง ๆ ที่ออกแบบไว้ได้ถูกต้อง หรือไม่ และการนำเครื่องไปใช้

ควบคุมจริง ซึ่งใช้ทดสอบกับระบบสายพานลำเลียง

5.3.1 การทดสอบคำสั่งต่าง ๆ

การทดสอบคำสั่งต่าง ๆ ของเครื่องว่าทำงานถูกต้องหรือไม่ ก็จะต้องเขียนโปรแกรมขึ้นมันไดสำหรับทดลองการทำงานของแต่ละคำสั่ง และป้อนโปรแกรมเข้าไปแล้วลองให้เครื่องทำงานโหมดทำงาน (RUN) ว่าเครื่องทำงานถูกต้องตามต้องการหรือไม่ การทดลองนี้จะทำกับทุก ๆ คำสั่ง

ผลการทดสอบ คำสั่งทุกคำสั่งสามารถทำงานได้ถูกต้อง

ในการทดสอบคำสั่งต่าง ๆ นั้นบางครั้งต้องมีการสร้างสัญญาณและข้อมูลอินพุทด้วย ซึ่งในการทดสอบจะใช้สวิทช์โยกในการป้อนสัญญาณอินพุท และใช้โมดูลกำหนดค่าตัวเลข ช่วยในการข้อมูลตัวเลข

ถ้าการทดสอบ ถ้าพบข้อผิดพลาดก็ต้องแก้ไขโปรแกรมในส่วนของคำสั่งที่ทำงานผิดพลาด ซึ่งถ้าการหาข้อผิดพลาดยุ่งยากซับซ้อน ก็อาจนำอีมูเลเตอร์มาช่วยหาโดยให้ทำงานเป็นสเต็ป (Step) แล้วดูค่ารีจิสเตอร์ต่าง ๆ เพื่อหาตำแหน่งที่ผิดพลาดและแก้ไขต่อไป

5.3.2 การทดสอบเวลาในการทำงานของคำสั่ง

การหาเวลาในการทำงานของแต่ละคำสั่งต่าง ๆ สามารถหาได้ 2 วิธีคือ

1. โดยวิธีคำนวณ
2. โดยการวัดจากการทำงานจริงของเครื่อง

ในการหาเวลาการทำงานของคำสั่งต่อไปนี้ จะเป็นการหาเวลาการทำงานของคำสั่งเบื้องต้นเท่านั้นเพราะคำสั่งเหล่านี้จะใช้ในการเขียนโปรแกรมเป็นส่วนใหญ่

เวลาในการทำงานของคำสั่งขึ้นอยู่กับสัญญาณนาฬิกา (Clock) ของซีพียูที่ทำงานด้วย สำหรับเครื่องที่ออกแบบนี้ใช้สัญญาณนาฬิกาขนาด 4 MHz ในการทำงานของซีพียู ดังนี้จะใช้เป็นฐานเวลานี้ในการคำนวณความเร็วการทำงานของคำสั่งต่าง ๆ

1. การหาเวลาทำงานของคำสั่งโดยการคำนวณ

คำสั่ง LD	Assembly program	Execute clock
	SRL A	8
	LD HL, xxxx	10
	OR (HL)	7
	รวม	25 Clocks
	ใช้เวลาในการทำงาน	6.25 μsec

คำสั่ง LD NOT	Assembly program	Execute clock
	SRL A	8
	LD HL, xxxx	10
	OR (HL)	7
	XOR C	4
	รวม	29 Clocks
	ใช้เวลาในการทำงาน	7.25 μsec

คำสั่ง OR	Assembly program	Execute program
	LD HL, xxxx	10
	OR (HL)	7
	รวม	17 Clocks
	ใช้เวลาในการทำงาน	4.25 μsec

คำสั่ง OR NOT	Assembly program	Execute program
	LD B, A	4
	LD A, xxxx	13
	XOR C	4
	OR B	4
	รวม	25 Clocks
	ใช้เวลาในการทำงาน	6.25 μsec

คำสั่ง AND	Assembly program	Execute program
	LD B, A	4
	LD A, xxxx	13
	OR 7FH	7
	AND B	4
	รวม	28 Clocks
	ใช้เวลาในการทำงาน	7 μsec

คำสั่ง AND NOT	Assembly program	Execute clock
	LD B, A	4
	LD A, xxxx	13
	XOR C	4
	OR 7FH	7
	AND B	4
	รวม	82 Clocks
	ใช้เวลาในการทำงาน	8 μsec

คำสั่ง AND LD	Assembly program	Execute clock
	ADD A, A	4
	JR C, 02	7/12
	AND 7FH	7
	รวม	18/23 clocks
	ใช้เวลาในการทำงาน	4.5/5.75 μsec

คำสั่ง OR LD	Assembly program	Execute clock
	ADD A, A	4
	JR NC, 01	7/12
	OR C	4
	รวม	15/20 clocks
	ใช้เวลาในการทำงาน	3.75/5 μsec

คำสั่ง OUT	Assembly program	Execute clock
	AND C	4
	LD (xxxx), A	13
	รวม	17 clocks

ใช้เวลาในการทำงาน 4.25 μsec

คำสั่ง OUT NOT	Assembly program	Execute clock
	AND C	4
	XOR C	4
	LD (xxxx), A	13
	XOR C	4
	รวม	25 clocks

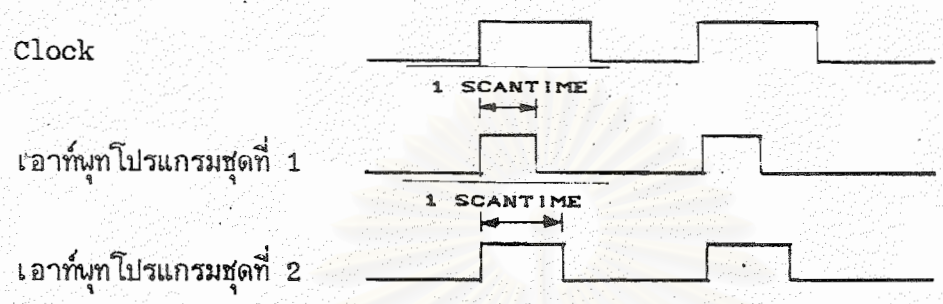
ใช้เวลาในการทำงาน 6.25 μsec

คำสั่ง	เวลาที่ใช้ทำงาน (μSEC)
LD	6.25
LD NOT	7.25
OR	4.25
OR NOT	6.25
AND	7.0
AND NOT	8.0
AND LD	4.5/5.75
OR LD	3.75/5
OUT	4.25
OUT NOT	6.25
เฉลี่ย	5.7

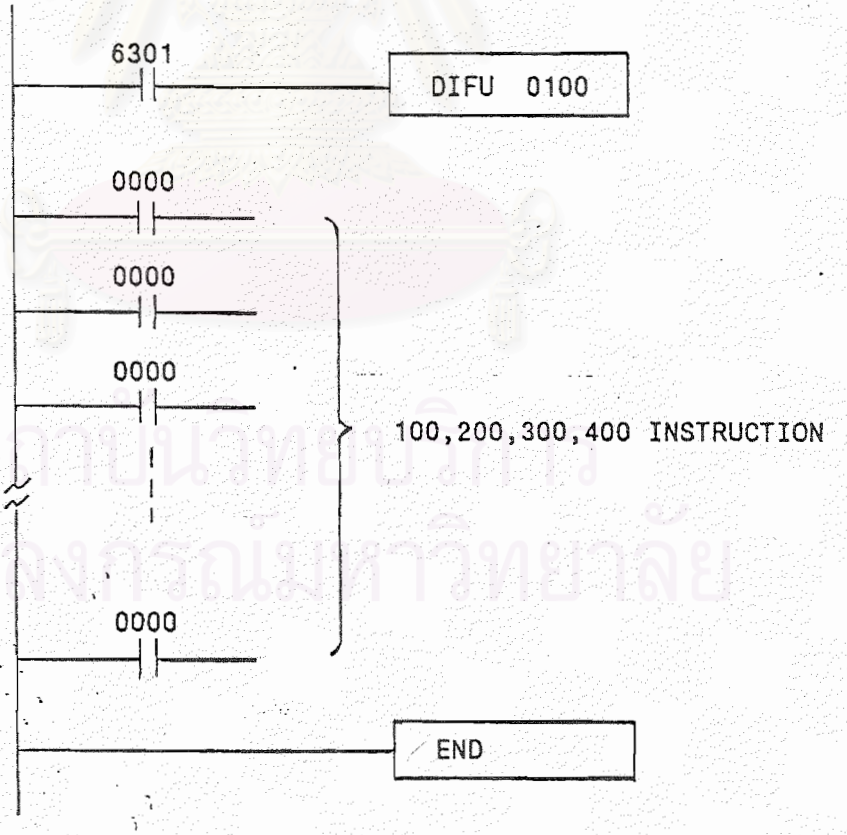
ตาราง 5.1 แสดงเวลาการทำงานของคำสั่งเบื้องต้น

2. การหาเวลาการทำงานของคำสั่ง โดยการวัดจากเครื่องจริง

การหาเวลาในการทำงานของคำสั่งโดยการวัด ถ้าวัดโดยตรงจะมีความยุ่งยาก จึงต้องมีการดัดแปลงวิธีวัดโดยจะใช้วิธีหาค่าความแตกต่างของเวลาในการทำงานหนึ่งรอบ (1 scand time) ของโปรแกรมสองชุดที่มีขนาดแตกต่างกัน

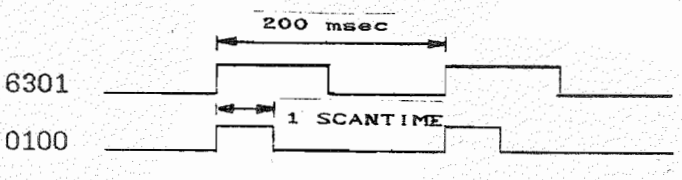


โปรแกรมขั้นบันไดที่จะใช้ทดสอบจะแสดงในรูปที่ 5.6 ซึ่งเป็นการวัดการทำงานของคำสั่ง LD ถ้าต้องการวัดคำสั่งอื่นก็เปลี่ยนจากคำสั่ง LD เป็นคำสั่งอื่นแทนก็สามารถวัดได้เช่นเดียวกัน



รูปที่ 5.7 แสดงวงจรขั้นบันไดที่ใช้ทดสอบวัดเวลาการทำงาน

รีเลย์ 6301 เป็นตัวกำเนิดสัญญาณนาฬิกา 0.2 วินาที ของเครื่อง
รีเลย์ 0100 เป็นรีเลย์เอาท์พุทพอร์ทของเครื่อง



การวัดเวลาการทำงานของรีเลย์เอาท์พุท 0100 แสดงในตารางที่ 5.2

จำนวนของคำสั่ง	เวลาของเอาท์พุท 0100 (msec)
100	1.07
200	1.69
300	2.32
400	2.95

ตารางที่ 5.2 แสดงค่าการวัดเวลาทำงานของคำสั่ง

จากตารางที่ 5.2 คำนวณหาเวลาของคำสั่ง LD ดังนี้

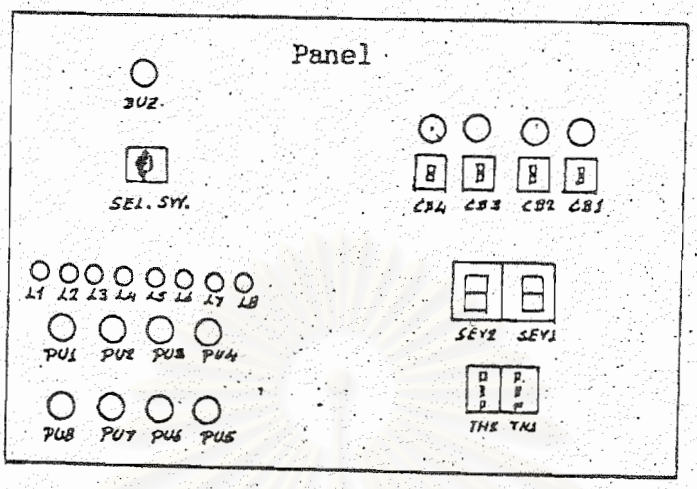
คำสั่ง LD จำนวน 100 คำสั่งใช้เวลาทำงาน = 1.69 - 1.07 = 0.62 msec

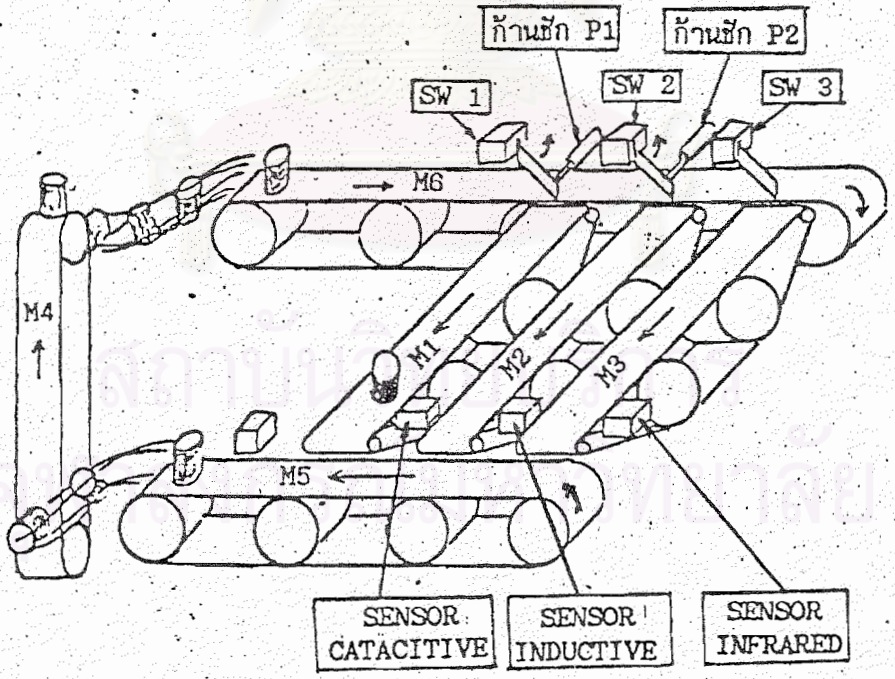
ดังนั้นคำสั่ง LD หนึ่งคำสั่งจะใช้เวลาทำงาน 6.2 μsec

จะเห็นว่าค่าเวลาการทำงานของคำสั่ง LD จากการวัด และจากการคำนวณมีค่าใกล้เคียงสำหรับการวัดของคำสั่งอื่นๆ ก็สามารถใช้วิธีเดียวกันได้ ซึ่งก็จะต้องมีค่าใกล้เคียงกับค่าจากการคำนวณ

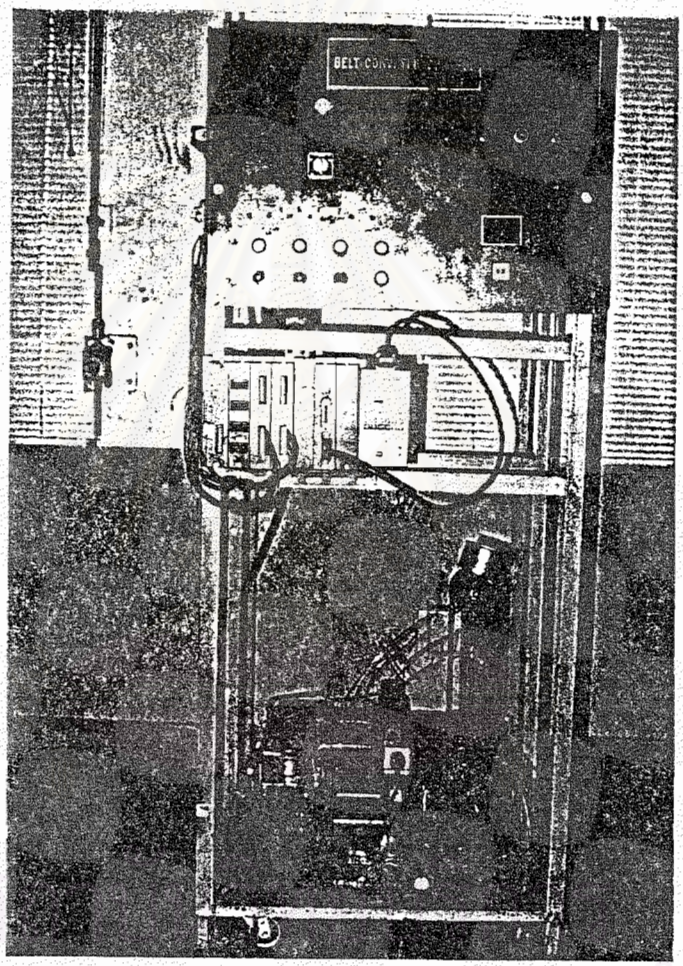
5.3.3 การทดสอบกับระบบจำลอง

ระบบจำลองที่เป็นชุดทดสอบของเครื่อง PC นี้ จะเป็นระบบจำลองของสายพานลำเลียง ระบบจำลองนี้มีลักษณะเป็นสายพานลำเลียงกระป๋องให้เคลื่อนเป็นวงรอบ โดยมีการเคลื่อนที่ทั้งแนวราบและแนวดิ่ง ระบบลำเลียงนี้สามารถเลือกช่องทางเคลื่อนที่ของกระป๋องได้ 3 ช่องทาง ซึ่งการเปลี่ยนทิศทางการเคลื่อนที่ของกระป๋องจะใช้ก้านชักซึ่งบังคับด้วยลม สายพานทั้งหมดจะมี 6 สายพาน ซึ่งควบคุมการเคลื่อนที่โดยมอเตอร์ 6 ตัว ในระบบจะมีอุปกรณ์ตรวจจับการเคลื่อนที่ของกระป๋อง ได้แก่ ลิมิตสวิทช์, ตัวตรวจจับชนิดแสงอินฟาเรด, ตัวตรวจจับแบบความจุ (Capacitive) และตัวตรวจจับแบบอินดัคแตนซ์ (Inductive) และมีแผงควบคุมประกอบด้วย สวิทช์และตัวแสดงผลต่าง ๆ ลักษณะของอุปกรณ์ควบคุมต่าง ๆ ของระบบจำลองแสดงในรูปที่ 5.8





รูปที่ 5.8 แสดงอุปกรณ์ของระบบจำลองสายพานลำเลียง



รูปที่ 5.9 แสดงระบบจำลองสายลำเลียงที่เครื่องทดสอบ

5.3.3.1 ส่วนประกอบของระบบจำลองสายพานลำเลียง

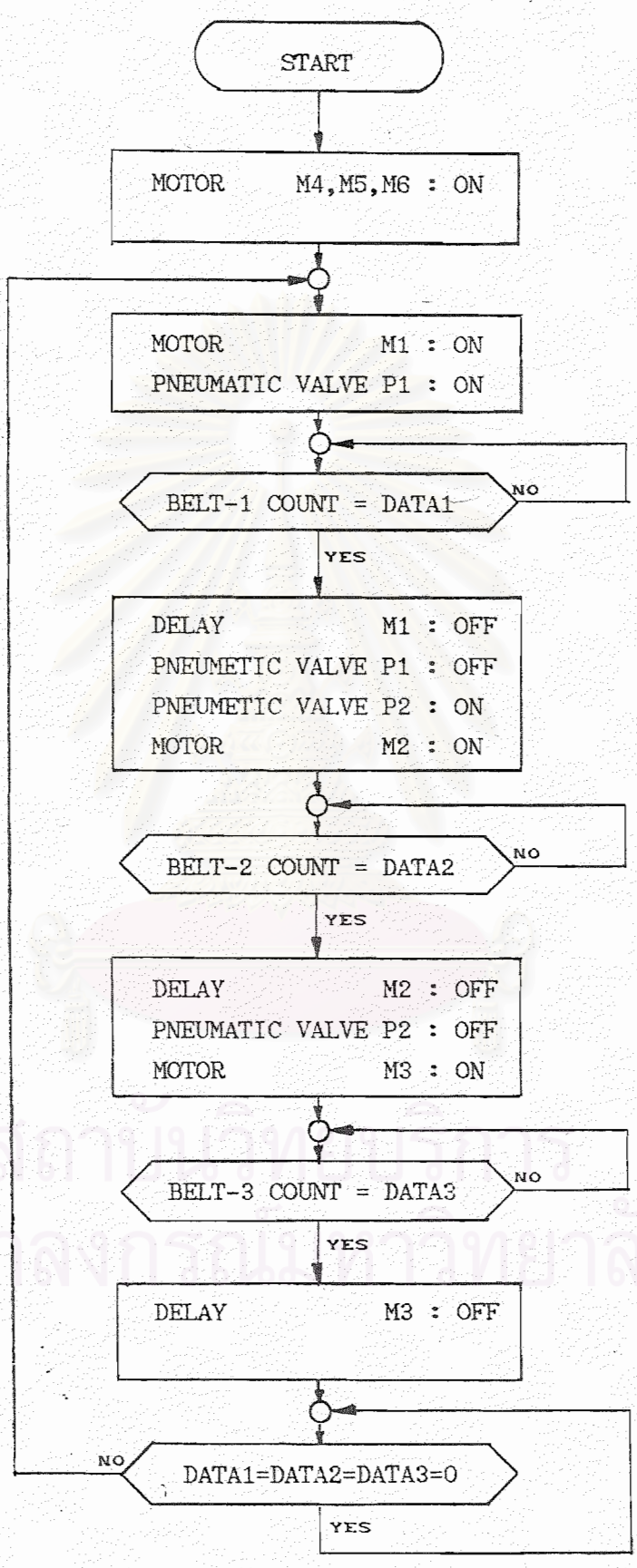
ส่วนประกอบของระบบจำลองสายพานลำเลียงมีดังนี้

1. โครงเหล็กขนาด 75 x 100 x 180 ซม. พร้อมล้อเลื่อนและหูยก
2. อุปกรณ์ต่าง ๆ สำหรับการทดลอง
   - 2.1 สายพานแนวดิ่งพร้อมมอเตอร์ขับเคลื่อน จำนวน 1 ชุด
   - 2.2 สายพานแนวนอนพร้อมมอเตอร์ขับเคลื่อน จำนวน 5 ชุด
   - 2.3 ก้านชักเปลี่ยนทิศทางกระป๋องพร้อมรีเลย์ลม (Pneumatic Relay) จำนวน 2 ชุด
   - 2.4 ตัวตรวจจับ (Sensor) ชนิดไม่สัมผัส
     - 2.4.1 แบบใช้แสงอินฟาเรด (Infrared) จำนวน 1 ชุด
     - 2.4.2 แบบเปลี่ยนแปลงความจุ (Capacitive) จำนวน 1 ชุด
     - 2.4.3 แบบเปลี่ยนแปลงอินดัคแตนซ์ (Inductive) จำนวน 1 ชุด
   - 2.5 ตัวตรวจจับ (Sensor) แบบสัมผัส : Proximity Switches จำนวน 4 ตัว
   - 2.6 Selector Switch จำนวน 1 ตัว
   - 2.7 Pushbutton Switches จำนวน 8 ตัว
   - 2.8 Thumbwheel Switches จำนวน 2 ตัว
   - 2.9 Light Emitting Diodes จำนวน 8 ตัว
   - 2.10 7 - Segment Displays จำนวน 2 ชุด
   - 2.11 Buzzer จำนวน 1 ตัว
   - 2.12 Relay Contactors พร้อมฐานรอง จำนวน 8 ชุด
3. อุปกรณ์สำหรับแหล่งจ่ายไฟ (220 โวลท์ 50 เฮิร์ท)
   - 3.1 Fuse พร้อม Fuse Holders จำนวน 4 ชุด
   - 3.2 Circuit Breakers จำนวน 4 ตัว
   - 3.3 Pilot Lamps จำนวน 4 ตัว
   - 3.4 หม้อแปลง 220/110 โวลท์ จำนวน 3 ตัว

5.3.3.2 โปรแกรมทดสอบระบบจำลอง

เครื่อง PC ที่ใช้ทดสอบจะประกอบด้วย โมดูลอินพุท โมดูลเอาท์พุทและโมดูลกำหนดค่าตัวเลข การทดสอบต้องการควบคุมการเคลื่อนที่ของกระป๋องว่าจะใช้สายพานที่ 1 หรือ 2 หรือ 3 โดยให้ผู้ใช้กำหนดจำนวนกระป๋องที่ต้องการให้ผ่านในแต่ละสายพานได้ ซึ่งจะใช้โมดูลกำหนดค่าตัวเลขเป็นตัวกำหนดจำนวนกระป๋อง โดยเริ่มต้นให้กระป๋องเคลื่อนที่ผ่านสายพานที่ 1 ก่อน เมื่อครบแล้วจึงไปเคลื่อนที่ผ่านสายพานที่ 2 และเมื่อครบจำนวนที่กำหนดของสายพานที่ 2 ก็ให้เคลื่อนที่สายพานที่ 3 และนับจนครบจำนวนที่กำหนด ก็ให้เคลื่อนที่ผ่านสายพานที่ 1 ใหม่ ผู้ใช้สามารถเปลี่ยนแปลงค่าจำนวนกระป๋องของแต่ละสายพาน ในขณะที่เครื่องกำลังทำงานได้

การทำงานเมื่อกดปุ่ม START มอเตอร์ M4, M5, M6 จะทำงานพร้อมกัน หลังจากนั้นมอเตอร์ M1 และก้านชัก P1 จะทำงาน หลังจากนั้นจะนับกระป๋องที่ผ่านสายพานที่ 1 โดยใช้ SW1 เมื่อนับครบจำนวนแล้ว ก้านชัก P1 จะ OFF มอเตอร์ M2 และก้านชัก P2 จะทำงาน สายพานที่ 1 จะหยุดทำงานเมื่อกระป๋องผ่านตัวตรวจจับ Capacitive ไปแล้ว 2 วินาที เมื่อ P2 และ M2 ทำงานแล้วกระป๋องจะเคลื่อนที่ผ่านสายพานที่ 2 และจะนับจำนวนกระป๋องที่ผ่านโดยใช้ SW2 เมื่อนับครบ P2 จะ OFF มอเตอร์ M3 จะทำงาน สายพานที่ 2 จะหยุดทำงานหลังจากกระป๋องที่นับครบได้ผ่านตัวตรวจจับ Inductive เป็นเวลา 2 วินาทีแล้ว เมื่อกระป๋องเคลื่อนที่ผ่านสายพานที่ 3 SW3 จะนับกระป๋องที่ผ่านสายพานที่ 3 เมื่อนับครบจะกลับไปเริ่มที่สายพาน 1 ซ้ำใหม่ไปเรื่อย ๆ สายพานที่ 3 จะหยุดหลังจากกระป๋องใบที่นับครบผ่านตัวตรวจจับอินฟราเรดเป็นเวลา 2 วินาทีแล้ว เมื่อต้องการให้ระบบหยุดทำงานจะต้องกดปุ่ม STOP เท่านั้น



รูปที่ 5.10 แสดงโฟลว์ชาร์ทการทำงานของโปรแกรมทดสอบระบบจำลอง

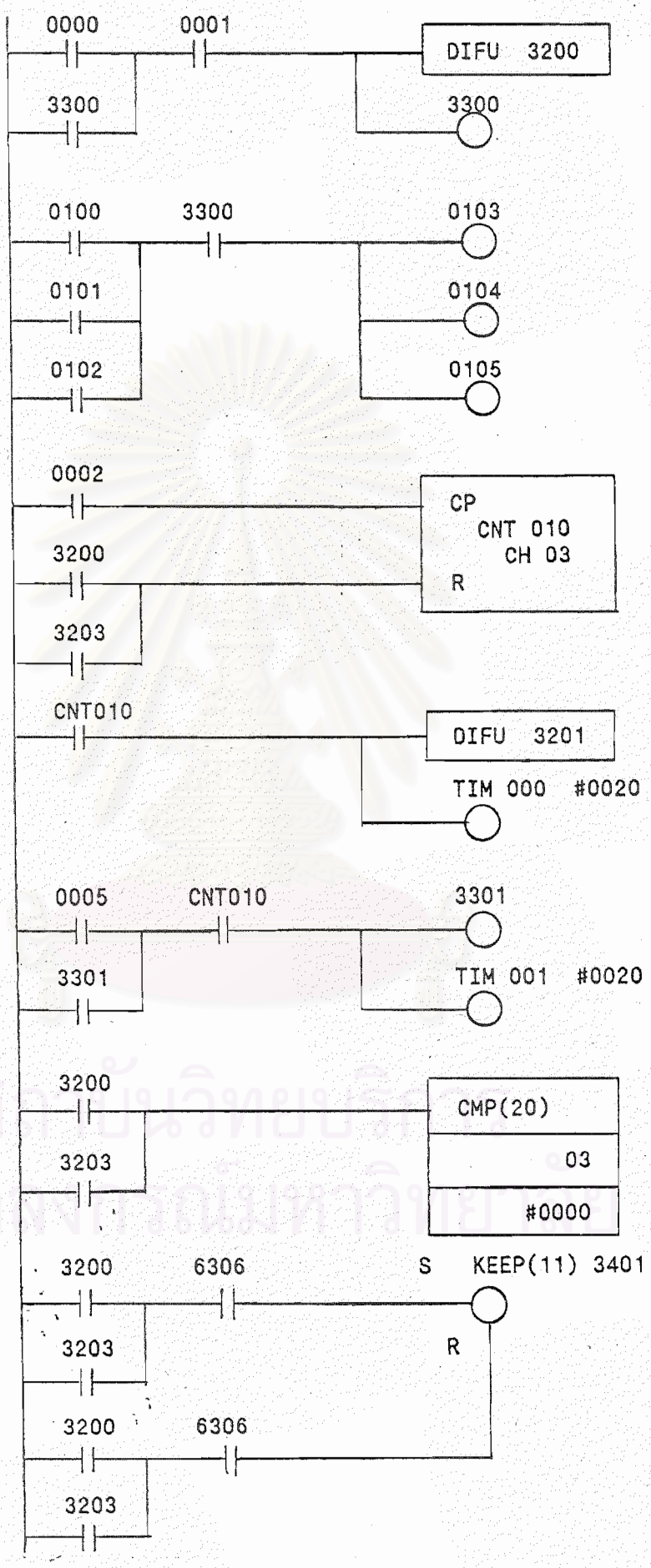
5.3.3.3 ขั้นตอนการทดสอบ

1. กำหนดเบอร์อินพุท/เอาท์พุทพอร์ทของอุปกรณ์ต่าง ๆ

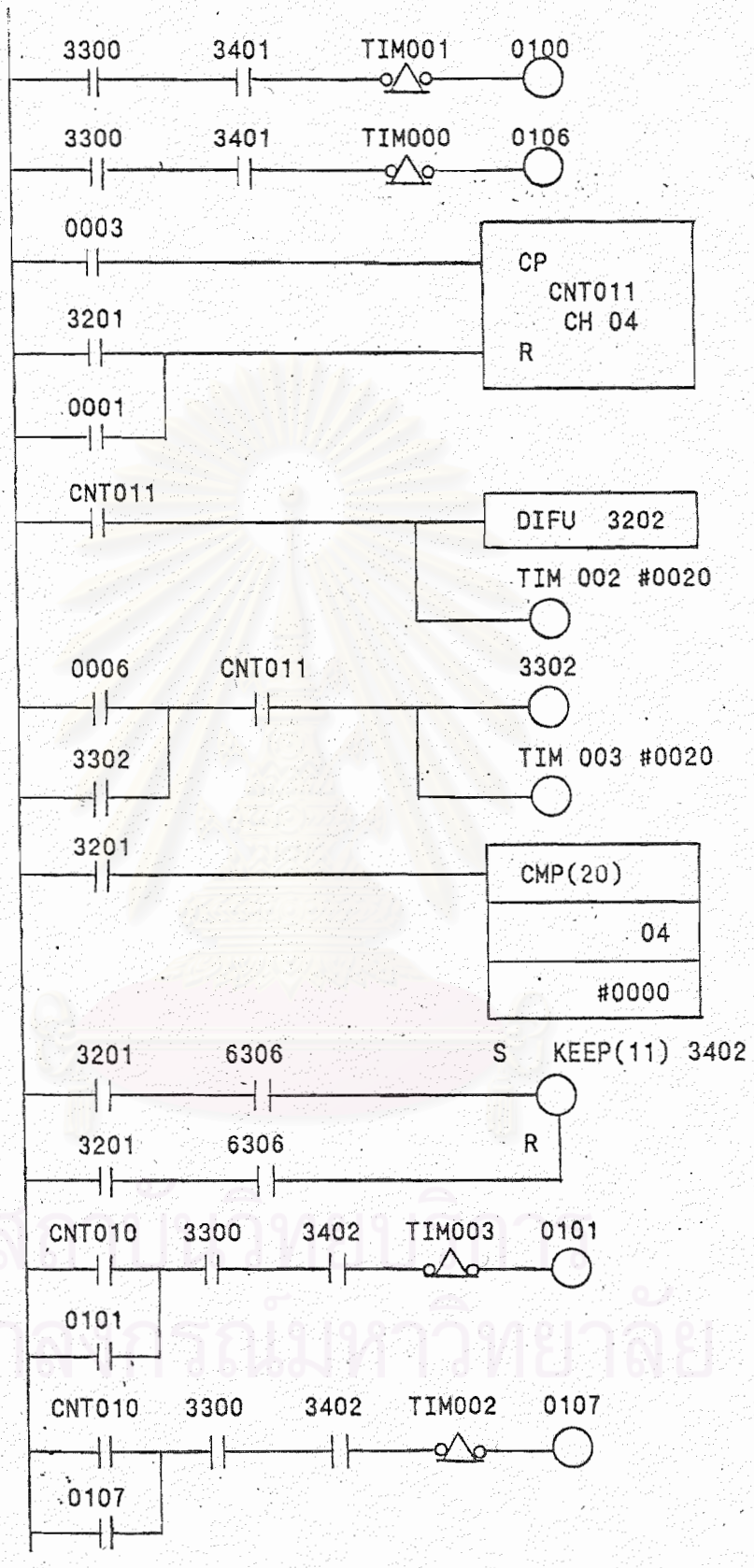
อินพุท	:	START	0000
		STOP	0001
		SW1	0002
		SW2	0003
		SW3	0004
		SENSOR CAPACITIVE	0005
		SENSOR INDUCTIVE	0006
		SENSOR INFRARED	0007
เอาท์พุท	:	MOTOR 1	0100
		MOTOR 2	0101
		MOTOR 3	0102
		MOTOR 4	0103
		MOTOR 5	0104
		MOTOR 6	0105
		ก้านชัก P1	0106
		ก้านชัก P2	0107

2. ออกแบบวงจรควบคุม (Ladder diagram) สำหรับการทดสอบ แสดงในรูปที่ 5.10
3. แปลงวงจรขั้นบันไดให้เป็นโปรแกรมนีมอนิค แล้วป้อนโปรแกรมลงในเครื่องโปรแกรมนีมอนิค นี้แสดงในรูปที่ 5.11
4. ต่อสาย อินพุท/เอาท์พุทต่าง ๆ เข้าเครื่อง PC
5. ทดสอบการทำงานต่าง ๆ ว่าถูกต้องหรือไม่

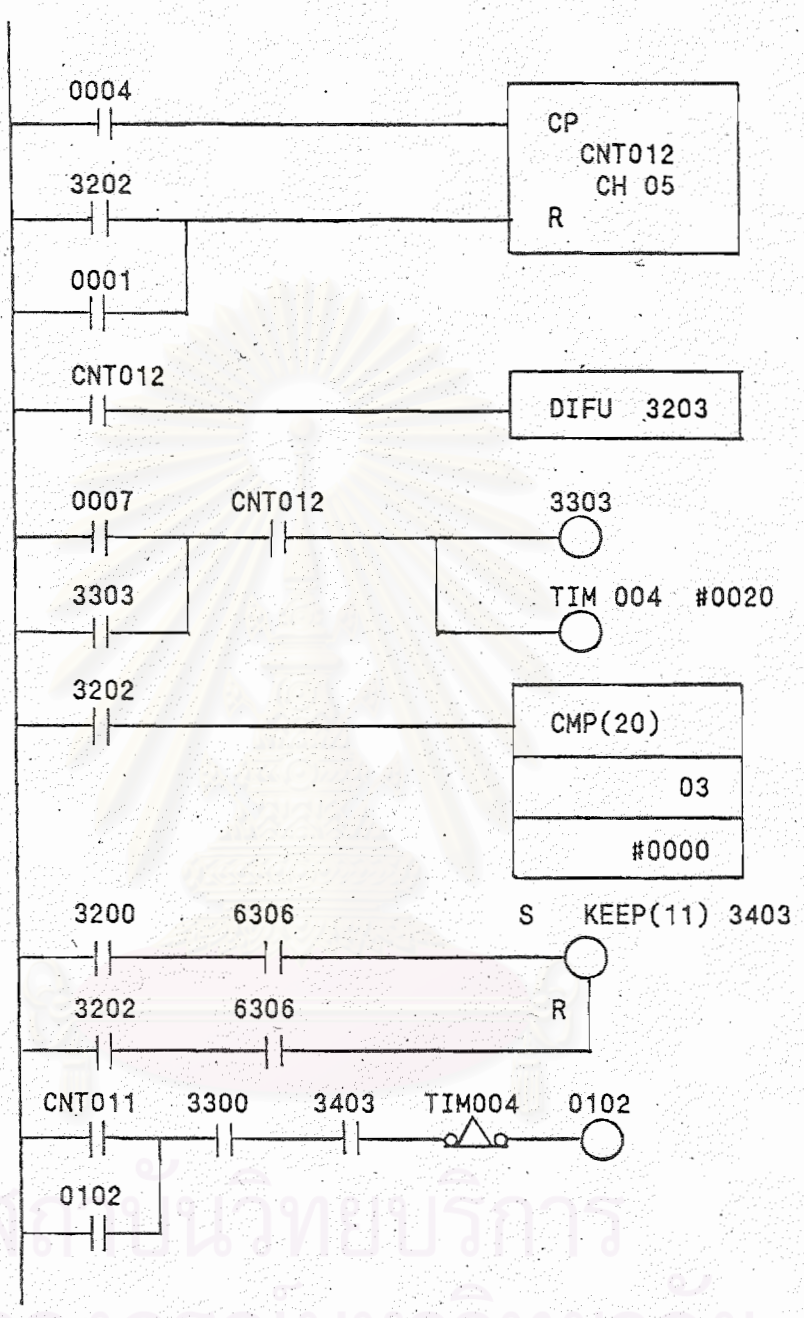
ผลการทดสอบ เครื่อง PC สามารถควบคุมการทำงานของระบบจำลองสายพานลำเลียงได้ถูกต้องตามโปรแกรมขั้นบันไดทุกประการ



รูปที่ 5.11 แสดงวงจรขั้นบันไดที่ใช้ทดสอบ



รูปที่ 5.11 แสดงวงจรขั้นบันไดที่ใช้ทดสอบ (ต่อ)



รูปที่ 5.11 แสดงวงจรขั้นบันไดที่ใช้ทดสอบ (ต่อ)

STEP	OPCODE	OPERAND
0000	LD	0000
0001	OR	3300
0002	AND NOT	0001
0003	DIFU(13)	3200
0004	OUT	3300
0005	LD	0100
0006	OR	0101
0007	OR	0102
0008	AND	3300
0009	OUT	0103
0010	OUT	0104
0011	OUT	0105
0012	LD	0002
0013	LD	3200
0014	OR	3203
0015	CNT	010
	-	CH 03
0016	LD	CNT 010
0017	DIFU(13)	3201
0018	TIM	000
	-	#0020
0019	LD	0005
0020	OR	3301
0021	AND	CNT 010
0022	OUT	3301
0023	TIM	001
	-	#0020
0024	LD	3200
0025	OR	3203
0026	CMP(20)	-
	-	03
	-	#0000
0027	LD	3200
0028	OR	3203
0029	AND	6306
0030	LD	3200

รูปที่ 5.12 แสดงคำสั่งโปรแกรมที่ใช้ทดสอบ

0031	OR	3203
0032	AND NOT	6306
0033	KEEP(11)	3401
0034	LD	3300
0035	AND NOT	3401
0036	AND NOT	TIM 001
0037	OUT	0100
0038	LD	3300
0039	AND NOT	3401
0040	AND NOT	TIM 000
0041	OUT	0106
0042	LD	0003
0043	LD	3201
0044	OR	0001
0045	CNT	011
	-	CH 04
0046	LD	CNT 011
0047	DIFU(13)	3202
0048	TIM	002
	-	#0020
0049	LD	0006
0050	OR	3302
0051	AND	CNT 011
0052	OUT	3302
0053	TIM	003
	-	#0020
0054	LD	3201
0055	CMP(20)	-
	-	04
	-	#0000
0056	LD	3201
0057	AND	6306
0058	LD	3201
0059	AND NOT	6306
0060	KEEP(11)	3402
0061	LD	CNT 010
0062	OR	0101
0063	AND	3300

รูปที่ 5.12 แสดงคำสั่งโปรแกรมที่ใช้ทดสอบ(ต่อ)

0064	AND NOT	3402
0065	AND NOT	TIM 003
0066	OUT	0101
0067	LD	CNT 010
0068	OR	0107
0069	AND	3300
0070	AND NOT	3402
0071	AND NOT	TIM 002
0072	OUT	0107
0073	LD	0004
0074	LD	3202
0075	OR	0001
0076	CNT	012
	-	CH 05
0077	LD	CNT 012
0078	DIFU(13)	3203
0079	LD	0007
0080	OR	3303
0081	AND	CNT 012
0082	OUT	3303
0083	TIM	004
	-	#0020
0084	LD	3202
0085	CMP(20)	-
	-	05
	-	#0000
0086	LD	3202
0087	AND	6306
0088	LD	3202
0089	AND NOT	6306
0090	KEEP(11)	3403
0091	LD	CNT 011
0092	OR	0102
0093	AND	3300
0094	AND NOT	3403
0095	AND NOT	TIM 004
0096	OUT	0102
0097	END(01)	-

รูปที่ 5.12 แสดงคำสั่งโปรแกรมที่ใช้ทดสอบ(ต่อ)

บทที่ 6
บทสรุป และข้อเสนอแนะ

6.1 สรุปผลการทำวิจัย

ผลการวิจัยครั้งนี้เป็นไปตามวัตถุประสงค์ของการวิจัยคือ พัฒนาเครื่องควบคุมชนิดโปรแกรมได้ ให้มีความสามารถในการจัดการข้อมูล และให้มีลักษณะโครงสร้างเป็นโมดูล ผลการทดสอบเครื่องสามารถทำงานได้ดีเป็นที่น่าพอใจ

เครื่อง PC ที่ออกแบบสร้างมีลักษณะคุณสมบัติดังนี้

- โครงสร้างมีลักษณะเป็นโมดูล สามารถติดตั้งและซ่อมบำรุงได้สะดวก ผู้ใช้สามารถเลือกชนิดจะจำนวนของโมดูลอินพุท/เอาท์พุท ในการใช้งานต่างๆ ได้เองตามความเหมาะสมของงาน โมดูลอินพุท/เอาท์พุทแบบต่างๆ สามารถเสียบในช่องเสียบ (Slot) ใดของเครื่องก็ได้เครื่องจะตรวจสอบ และจัดค่าของพารามิเตอร์ต่างๆ ให้เอง

- ในระบบที่ออกแบบสามารถสร้างโมดูลอินพุท/เอาท์พุท สำหรับติดต่อกับสัญญาณพิเศษแบบต่างๆ ได้ง่าย เครื่อง PC ที่สร้างนี้ประกอบด้วยโมดูลประมวลผล, โมดูลแสดงผลคีย์บอร์ดและสื่อสาร โมดูลอินพุท โมดูลเอาท์พุท โมดูลอนาลอกอินพุท โมดูลกำหนดค่าตัวเลข และตัวป้อนโปรแกรมแบบมือถือ

- มีความสามารถในการจัดการข้อมูล มีคำสั่งฟังก์ชันต่างๆ ในการจัดการข้อมูลได้แก่ คำสั่งการคำนวณ คำสั่งการเปรียบเทียบ คำสั่ง[illegible]ื่อนย้ายข้อมูล คำสั่งทางลอจิก คำสั่งเกี่ยวกับการ[illegible]งรหัส

- เครื่องที่ออกแบบสามารถติดต่ออินพุท/เอาท์พุท ได้ทั้งหมดถึง 512 จุด โดยไม่จำกัดจำนวนของอินพุทและจำนวนเอาท์พุทว่าต้องเท่ากัน แต่รวมกันแล้วไม่ควรเกิน 512 จุด ถ้าต้องการจำนวนอินพุท/เอาท์พุทมากกว่าสามารถใช้ได้สูงสุดถึง 976 จุด แต่จะทำให้รีเลย์ช่วย (Auxlirary relay) ของเครื่องลดลง

- การแปลงคำสั่งขั้นบันไดใช้วิธี Compile ร่วมกับวิธี Call
- ความเร็วเฉลี่ยในการทำงานคำสั่งเบื้องต้น 5.7 μsec
- ผู้ใช้สามารถเขียนโปรแกรมคำสั่งขั้นบันไดได้สูงสุดถึง 4000 คำสั่ง
- โปรแกรม Ladder interpreter ที่ใช้ในการแปลงภาษา Ladder เป็นภาษาเครื่องนี้ สามารถนำไปติดตั้งในเครื่องควบคุมระบบอื่นที่ใช้ไมโครโปรเซสเซอร์ Z-80A ได้โดยง่าย สามารถย่นย่อเวลาในการพัฒนาโปรแกรมได้มาก โดยเฉพาะระบบควบคุมที่ต้องมีการควบคุมซีเควันซ์ และการคำนวณตัวเลข

6.2 ข้อเสนอแนะ

1. PC ที่พัฒนาขึ้นทดลองกับระบบจำลองในห้องปฏิบัติการเท่านั้น แม้ผลการทดลองจะไม่ปรากฏความผิดพลาด ผู้วิจัยคาดว่า PC ที่พัฒนาขึ้นนี้สามารถนำไปพัฒนาเชิงพาณิชย์ได้ โดยต้องนำไปทดลองใช้ควบคุมในโรงงานอุตสาหกรรมจริงๆ สักระยะหนึ่งเพื่อความแน่ใจในการลงทุนผลิต ในการวิจัยครั้งนี้ยังไม่ได้นำเครื่อง PC ไปทดลองกับเครื่องจักรจริง เนื่องจากต้องใช้เวลาในการทดลองนาน และมีค่าใช้จ่ายที่สูง งบประมาณที่ใช้ในการวิจัยไม่เพียงพอกับการทดลอง

2. อาร์ดแวร์ของ PC สามารถลดขนาดให้เล็กลงอีกได้ เพราะปัจจุบันมีการผลิต MPU Z-80A ที่มีขนาดเล็กและเป็นแบบ Chip Carrier มีองค์ประกอบอื่นๆ เช่น Real time clock UART Watch dog Timer ในตัว ทำให้ใช้งานได้ง่ายขึ้นมาก นอกจากนั้น องค์ประกอบวงจรอื่นๆ เช่น RAM และ ROM จะมีความจุสูงขึ้น เป็นการลดขนาด และจำนวนของอุปกรณ์ลงได้ การลดองค์ประกอบวงจรลงจะทำให้ราคาของเครื่องที่ผลิตถูกลงได้

3. แนวโน้มของการโปรแกรมเครื่อง PC ด้วยภาษา Function Chart มีมากขึ้นจากการออกมาตรฐาน IEC-848 ซึ่งกำหนดมาตรฐานภาษา Function Chart ของ IEC ซึ่งเป็นมาตรฐานสากล ทำให้หลายบริษัทหันมาให้ความสนใจ การเขียนโปรแกรมด้วย Function Chart มากขึ้น การพัฒนาโปรแกรมสำหรับแปลภาษา Function Chart จึงน่าจะเป็นเรื่องที่จะทำการวิจัยกันต่อไป

บรรณานุกรม

1. กฤษดา วิศวธีรานนท์, เอกสารประกอบการสัมมนาเรื่อง PLC, สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
2. OMRON, USER'S MANUAL PROGRAMMABLE CONTROLLER SYSMAC C-SERIES
3. OMRON, USER'S MANUAL PROGRAMMABLE CONTROLLER SYSMAC -S6
4. FUJI ELECTRIC, USER'S MANUAL PROGRAMMABLE CONTROLLER
5. GE-FUNUC, USER'S MANUAL PROGRAMMABLE CONTROLLER
6. กฤษดา วิศวธีรานนท์, สมบูรณ์ จงชัยกิจ และสมพงษ์ ฉัตรแสงอุทัย, การพัฒนาโปรแกรมควบคุมการทำงานของ PLC , การประชุมวิชาการ วิศวกรรมไฟฟ้า 9 สถาบันครั้งที่ 11
7. กฤษดา วิศวธีรานนท์, สมบูรณ์ จงชัยกิจ และสิทธิพร ประวัติรุ่งเรือง, เครื่องควบคุมชนิดโปรแกรมได้ซึ่งมีความสามารถในการจัดการข้อมูล, การประชุมวิชาการวิศวกรรมไฟฟ้า 9 สถาบันครั้งที่ 12
8. สมพงษ์ ฉัตรแสงอุทัย, การพัฒนาตัวควบคุมแบบลำดับที่โปรแกรมได้ขนาดเล็ก วิทยานิพนธ์ปริญญาวิทยาศาสตรมหาบัณฑิต จุฬาลงกรณ์มหาวิทยาลัย 2532
9. Thomas E.Kissel, Understanding and Using Programmable Controller, Rentice-Hall, Inc.,Englewood Cliffs, New Jersey.
10. J.B.Peatman, Microcomputer-based Design, New York MC-GRAW Hill 1977
11. Warnock,lan G.,Programmable controllers operation and Application, Prentice Hall International (UK) Ltd. LONDON, 1988
12. JOB DENOTTER, Programmable Logic controller operation, interfare and programming, Prentice-Hall, Inc. Englewood Cliffs, New Jersey
13. Junji Yamada, Batch Process control and recipe management using a distributed control system, Computer Applications in industrial Instrumentation (DCS) Training I , Chulalongkorn University 1990
14. O J Struger, R D Gray, R Hall, Conference on programmable Controller ,1st London England,1985
15. E Norup, Small PC design and application , Electromatic, Denmark
16. Joseph A.Benedetto, Preventing Errors in Programmable Control, Machine Design October 1980
17. Martyn Jones, A Modular Approach to Process Control ,Electronics & Power November/December 1985

18 กฤษดา วิศวธีรานนท์ "การพัฒนาเครื่องควบคุมชบวนการผสมสาหรับการผลิตในโรงงานอาหารสัตว์ โดยใช้ไมโครคอมพิวเตอร์" การประชุมวิชาการทางวิศวกรรมไฟฟ้า 8 สถาบันอุดมศึกษา ครั้งที่ 7 2527

19 สมศักดิ์ ทาทอง,กฤษดา วิศวธีรานนท์ "การประยุกต์ไมโครคอมพิวเตอร์ในระบบควบคุมกระบวนการผสมในโรงงานอาหารสัตว์" การประชุมวิชาการวิศวกรรมไฟฟ้า 8 สถาบันอุดมศึกษา ครั้งที่ 8 2528

20 กฤษดา วิศวธีรานนท์ "การประยุกต์ไมโครคอมพิวเตอร์ในการควบคุมกระบวนการผลิตในโรงงานอาหารสัตว์" วิศวกรรมสาร เล่มที่ 2 เม.ย. 2529

21 กฤษดา วิศวธีรานนท์ "ประยุกต์ไมโครโปรเซสเซอร์กับการบรรจุหีบห่อ" วารสารเทคนิค ฉบับที่ 30 ก.ค. 2530

22 กฤษดา วิศวธีรานนท์, นรังสรรค์ วิไลสกุลยง "เครื่องควบคุมลิฟท์โดยสารเดี่ยวโดยใช้ไมโครคอมพิวเตอร์" วิศวกรรมสาร เล่มที่ 3 2530 วสท. แห่งประเทศไทย ครั้งที่ 9 2529

23 กฤษดา วิศวธีรานนท์, นรังสรรค์ วิไลสกุลยง "เครื่องควบคุมลิฟท์โดยสารเดี่ยวโดยใช้ไมโครคอมพิวเตอร์" การประชุมวิชาการทางวิศวกรรมไฟฟ้าสถาบันอุดมศึกษาแห่งประเทศไทย ครั้งที่ 9 2529

24 นรังสรรค์ วิไลสกุลยง "การพัฒนาเครื่องควบคุมลิฟท์โดยสารเดี่ยวโดยใช้ไมโครคอมพิวเตอร์" วิทยานิพนธ์ วิศวกรรมศาสตร์มหาบัณฑิต จุฬาลงกรณ์มหาวิทยาลัย 2530

25 กฤษดา วิศวธีรานนท์ PC เบื้องต้น เอสารประกอบการอบรมสัมมนา สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) 17-18 ม.ย. 2529

26 ภากร หุตะสังกาศ, กิตติ ตีรเศรษฐ "ไมโครคอมพิวเตอร์กับเครื่องควบคุมที่โปรแกรมได้" การประชุมทางวิชาการวิศวกรรมไฟฟ้า สถาบันอุดมศึกษาแห่งประเทศไทย ครั้งที่ 9 ธ.ค. 2529

27 ภากร หุตะสังกาศ, กิตติ ตีรเศรษฐ "การประยุกต์ใช้ไมโครคอมพิวเตอร์ในการจาลองการทางานของเครื่องควบคุมที่โปรแกรมได้" การประชุมวิชาการวิศวกรรมไฟฟ้า สถาบันอุดมศึกษาแห่งประเทศไทย ครั้งที่ 10 พ.ย. 2530

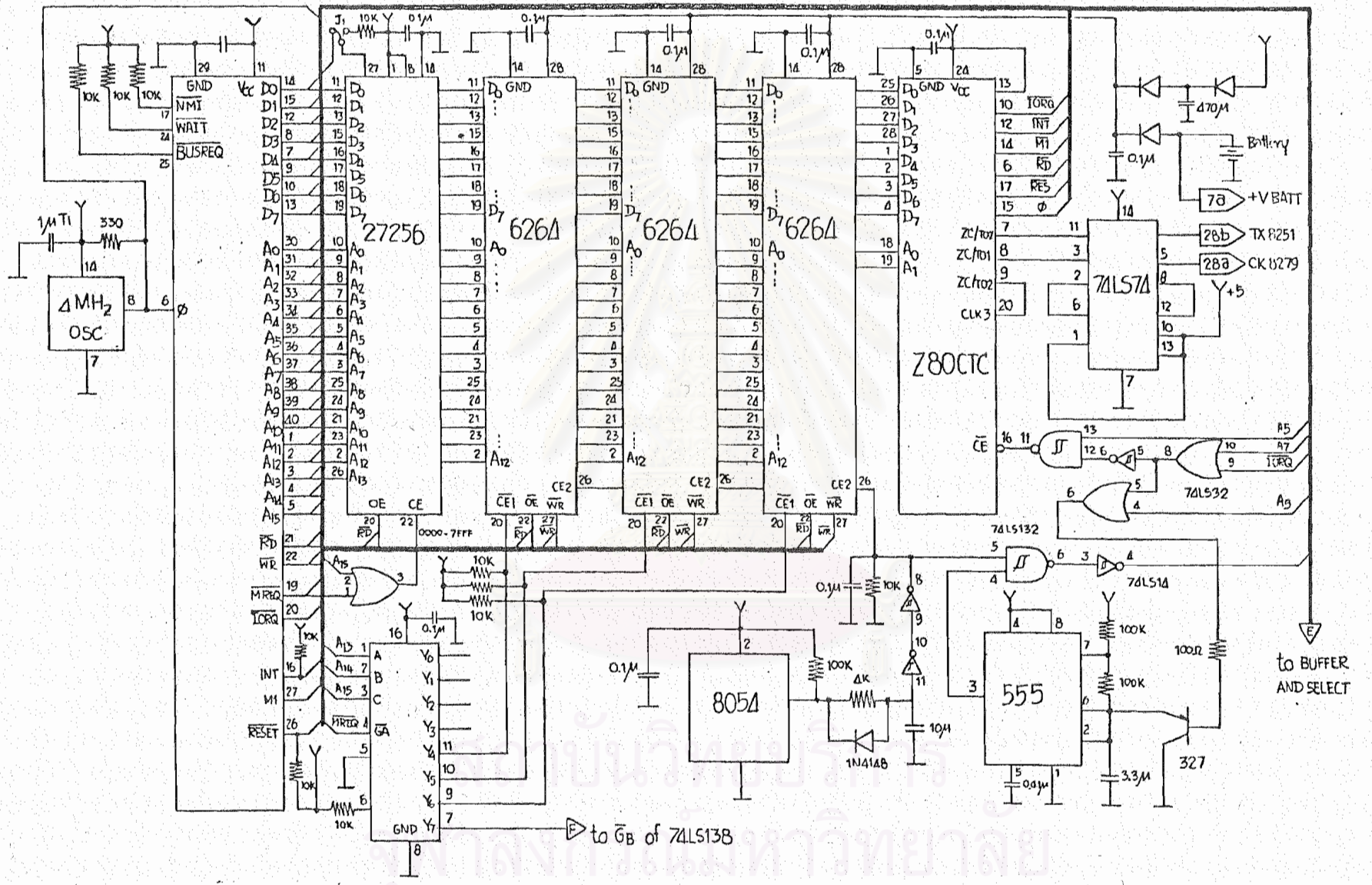
28 กฤษดา วิศวธีรานนท์, สมบูรณ์จงชัยกิจ "เครื่องควบคุมที่โปรแกรมได้" รายงานการวิจัยเสนอต่อศูนย์เทคโนโลยีอิเล้ทรอนิกส์และคอมพิวเตอร์แห่งชาติ ภายใต้โครงการหลักโครงการพัฒนาวงจรอิเล็กทรอนิกส์เพื่ออุตสาหกรรม ปี 2532

ภาคผนวก ก

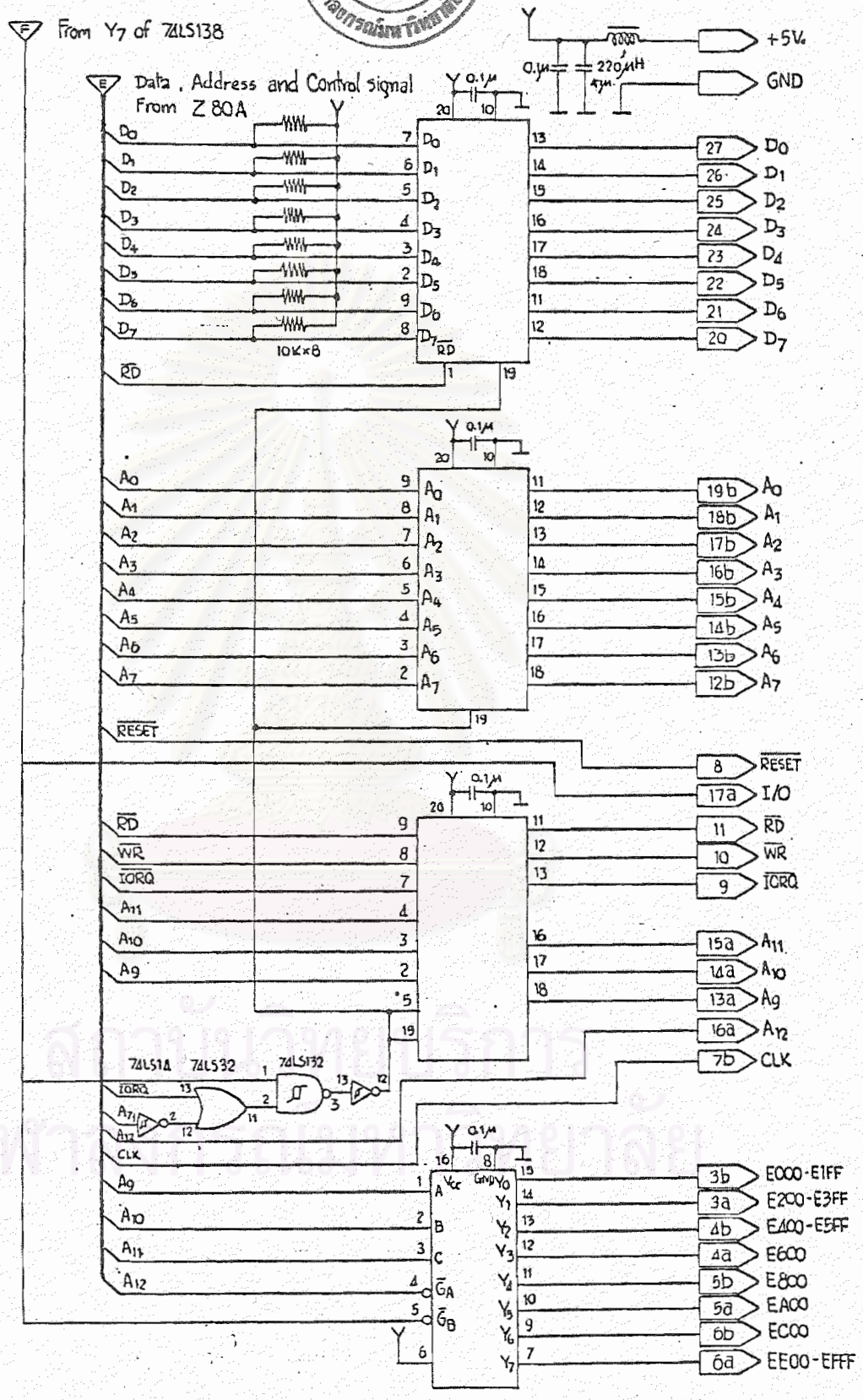
รายละเอียดฮาร์ดแวร์ของเครื่อง

วงจรของเครื่อง PC ที่ออกแบบสร้างทั้งหมด แสดงในรูปที่ ก.1 ถึงรูปที่ ก.10 โดยแบ่งเป็นวงจรของโมดูลต่าง ๆ ดังนี้

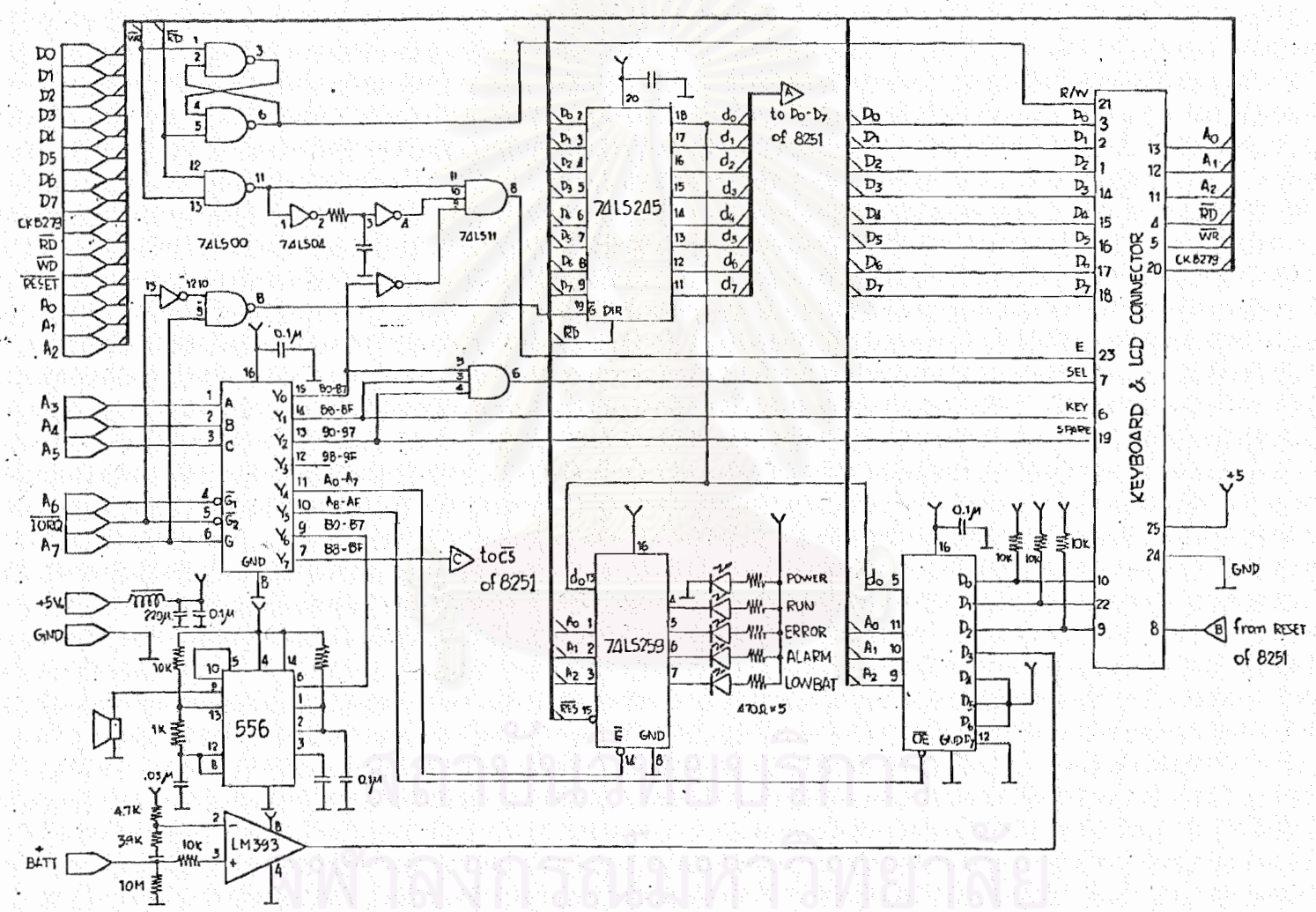
โมดูลประมวลผล	แสดงในรูปที่ ก.1 และ ก.2
โมดูลแสดงผลคีย์บอร์ด และสื่อสาร	แสดงในรูปที่ ก.3 และ ก.4
โมดูลอินพุท	แสดงในรูปที่ ก.5
โมดูลเอาท์พุท	แสดงในรูปที่ ก.6
โมดูลอนาลอกอินพุท	แสดงในรูปที่ ก.7 และ ก.8
โมดูลกำหนดค่าตัวเลข	แสดงในรูปที่ ก.9
ตัวป้อนโปรแกรมแบบมือถือ	แสดงในรูปที่ ก.10



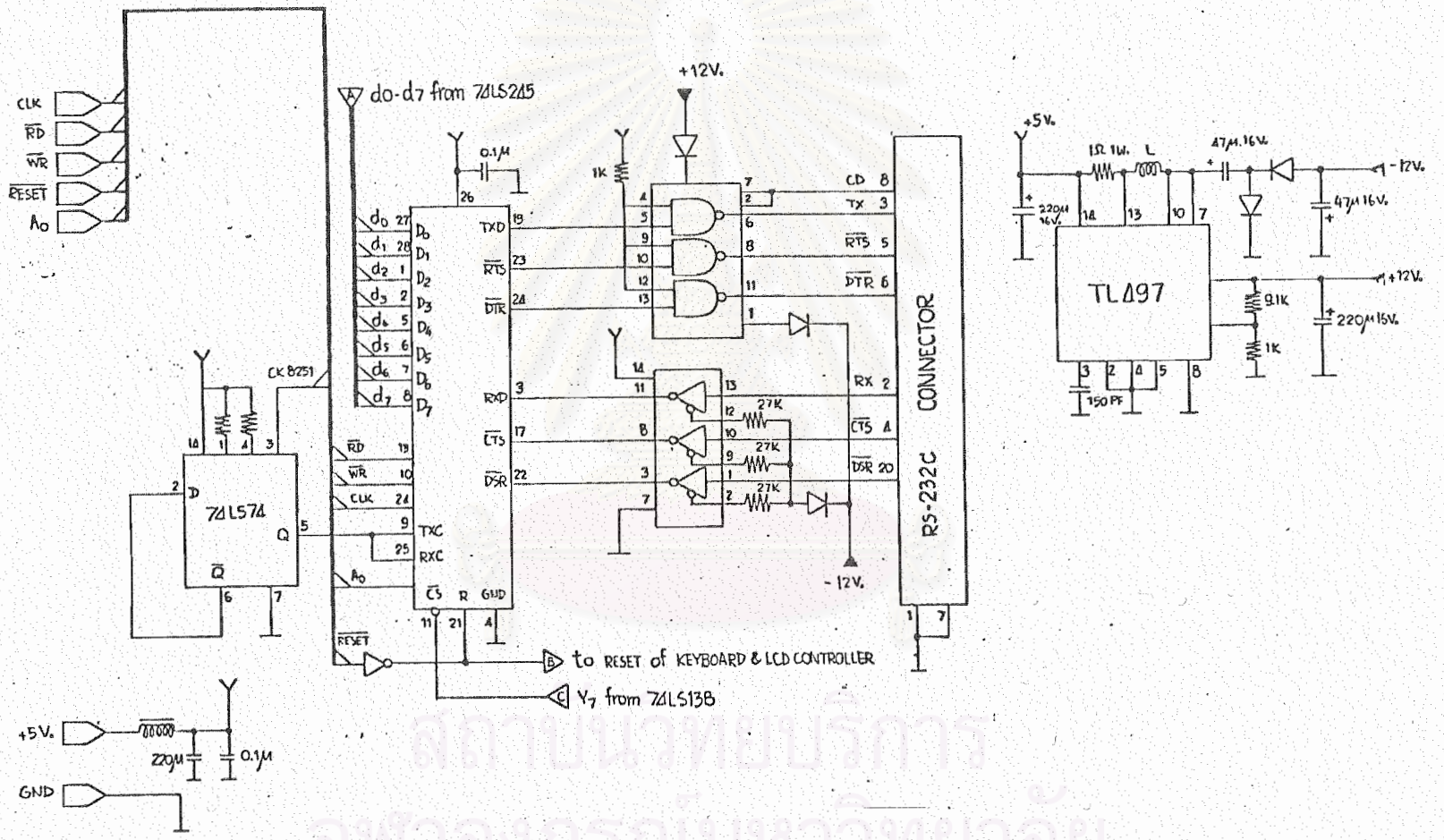
รูปที่ ก.1 แสดงวงจรของโมดูลประมวลผล



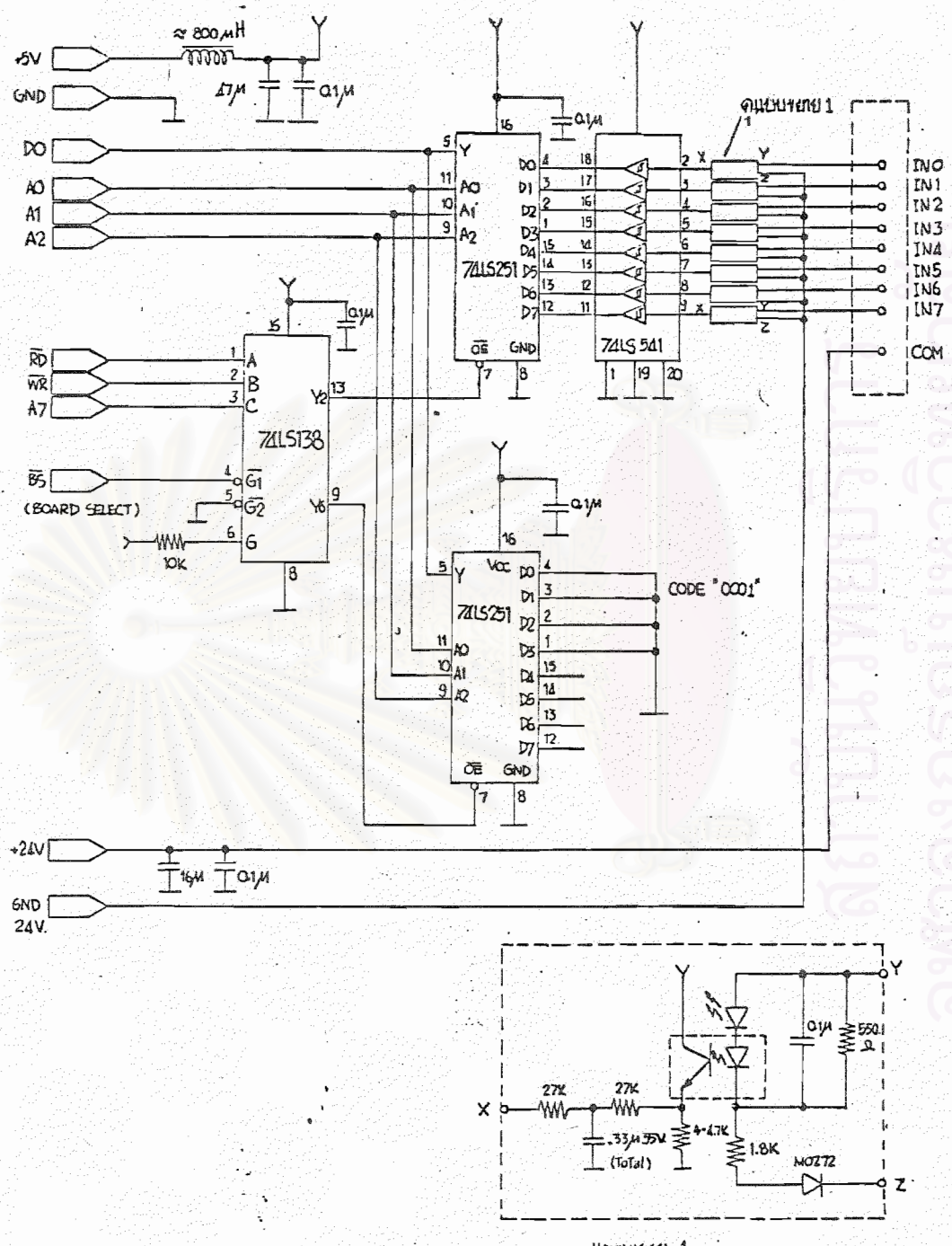
รูปที่ ก.2 แสดงวงจรของโมดูลประมวลผล (ต่อ)



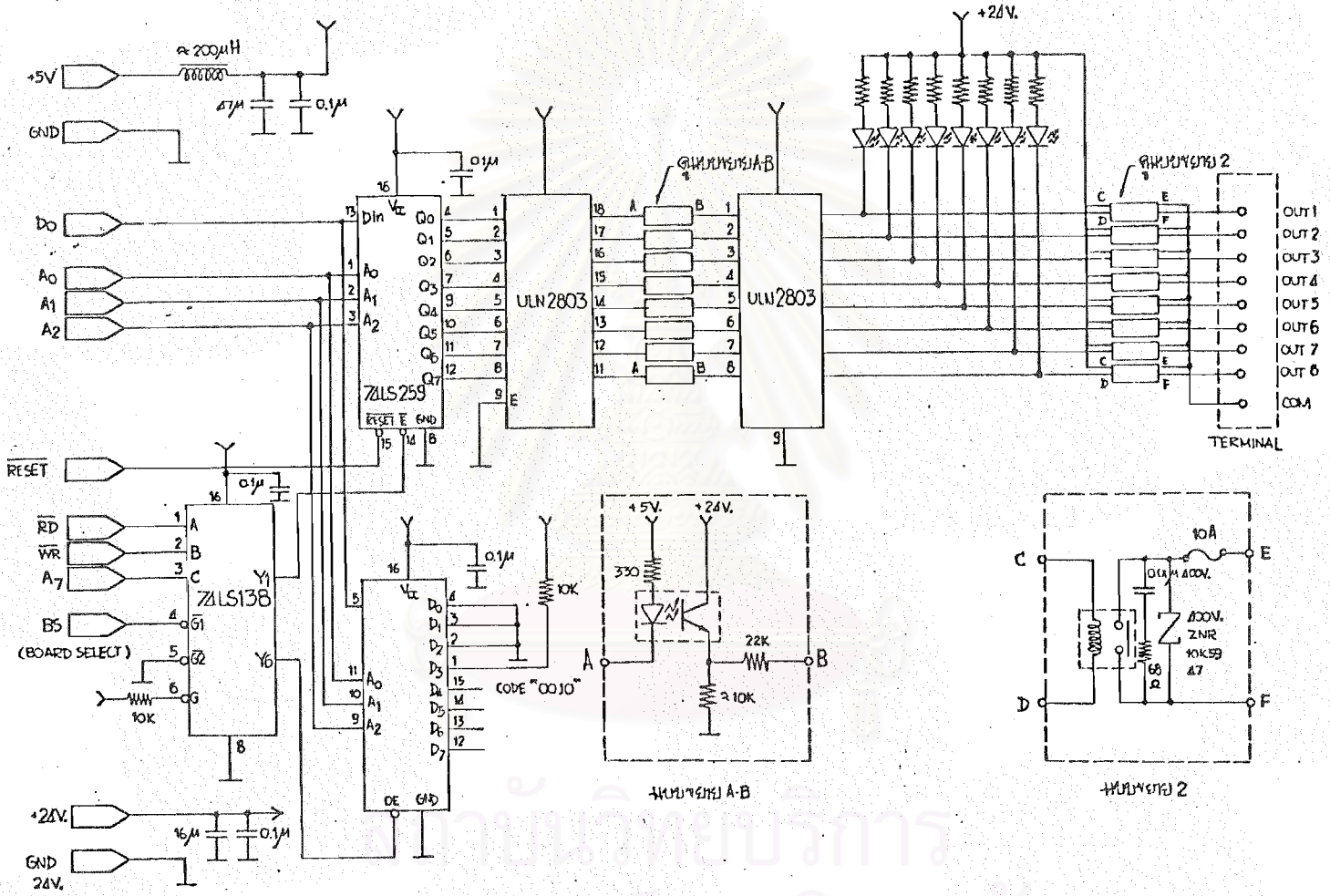
รูปที่ ก.3 แสดงวงจรของโมดูลแสดงผล คีย์บอร์ด และสื่อสาร



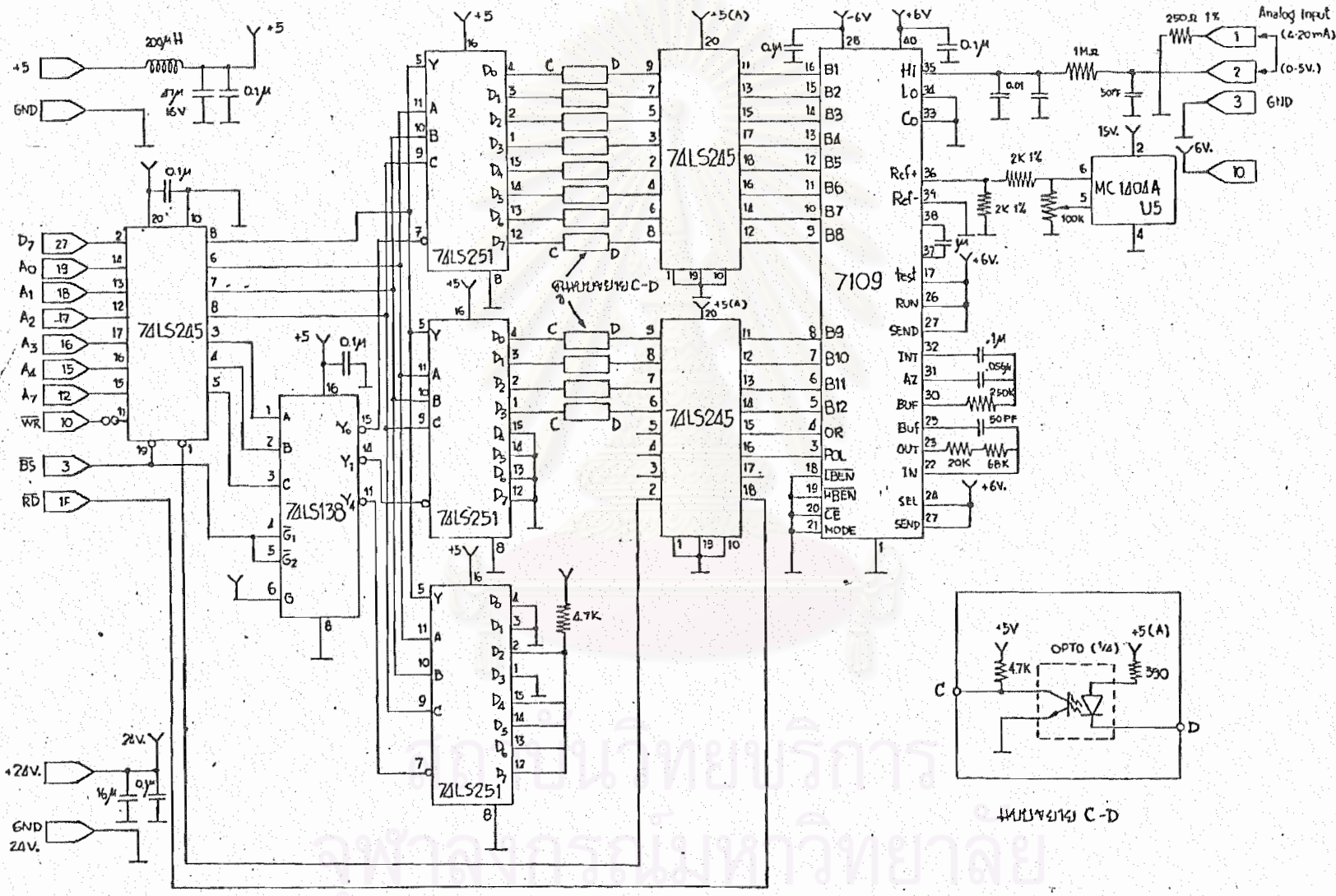
รูปที่ ก.4 แสดงวงจรของโมดูลแสดงผล คีย์บอร์ด และสื่อสาร (ต่อ)



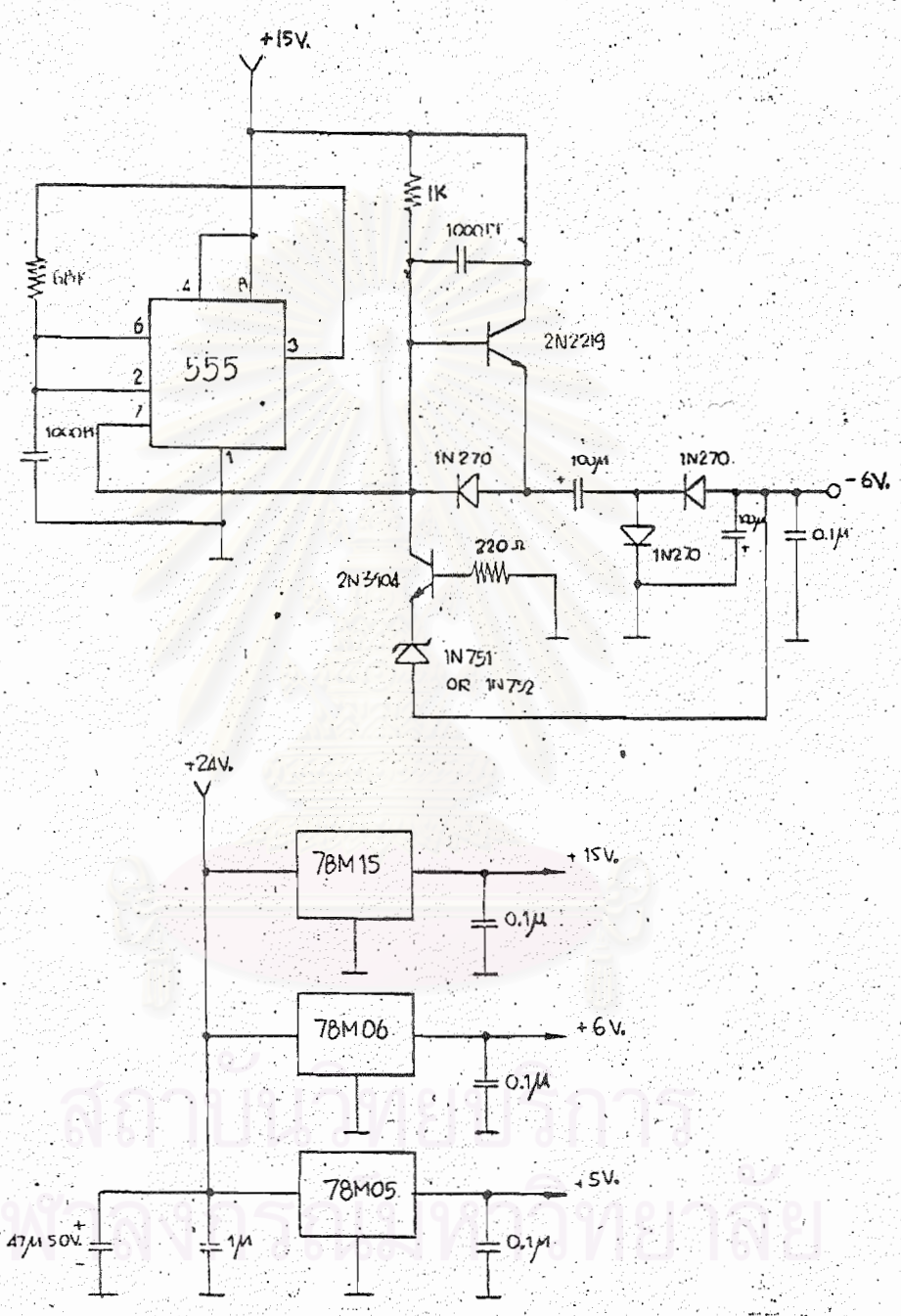
รูปที่ ก.5 แสดงวงจรของโมดูลอินพุท



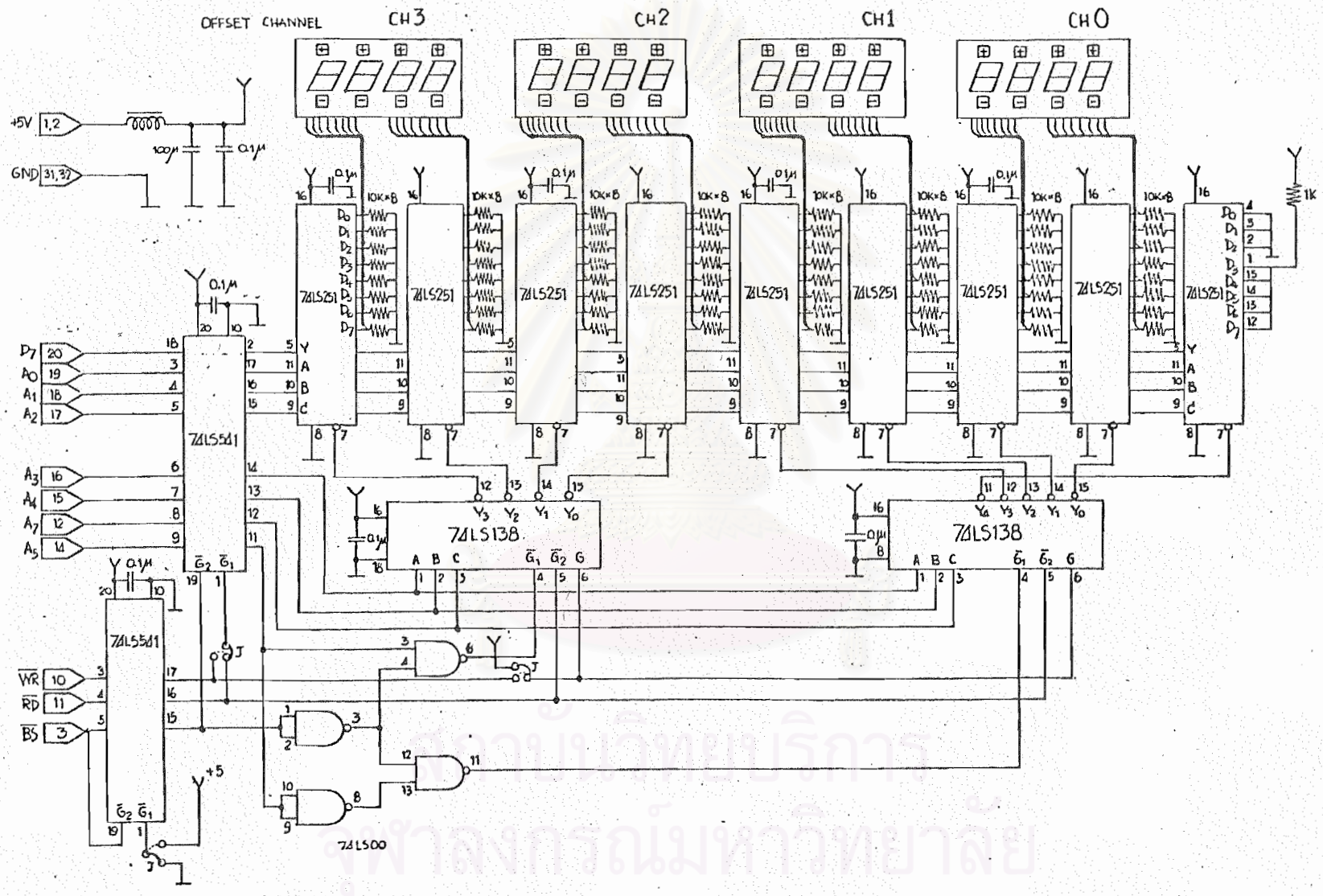
รูปที่ ก.6 แสดงวงจรของโมดูลเอาท์พุท



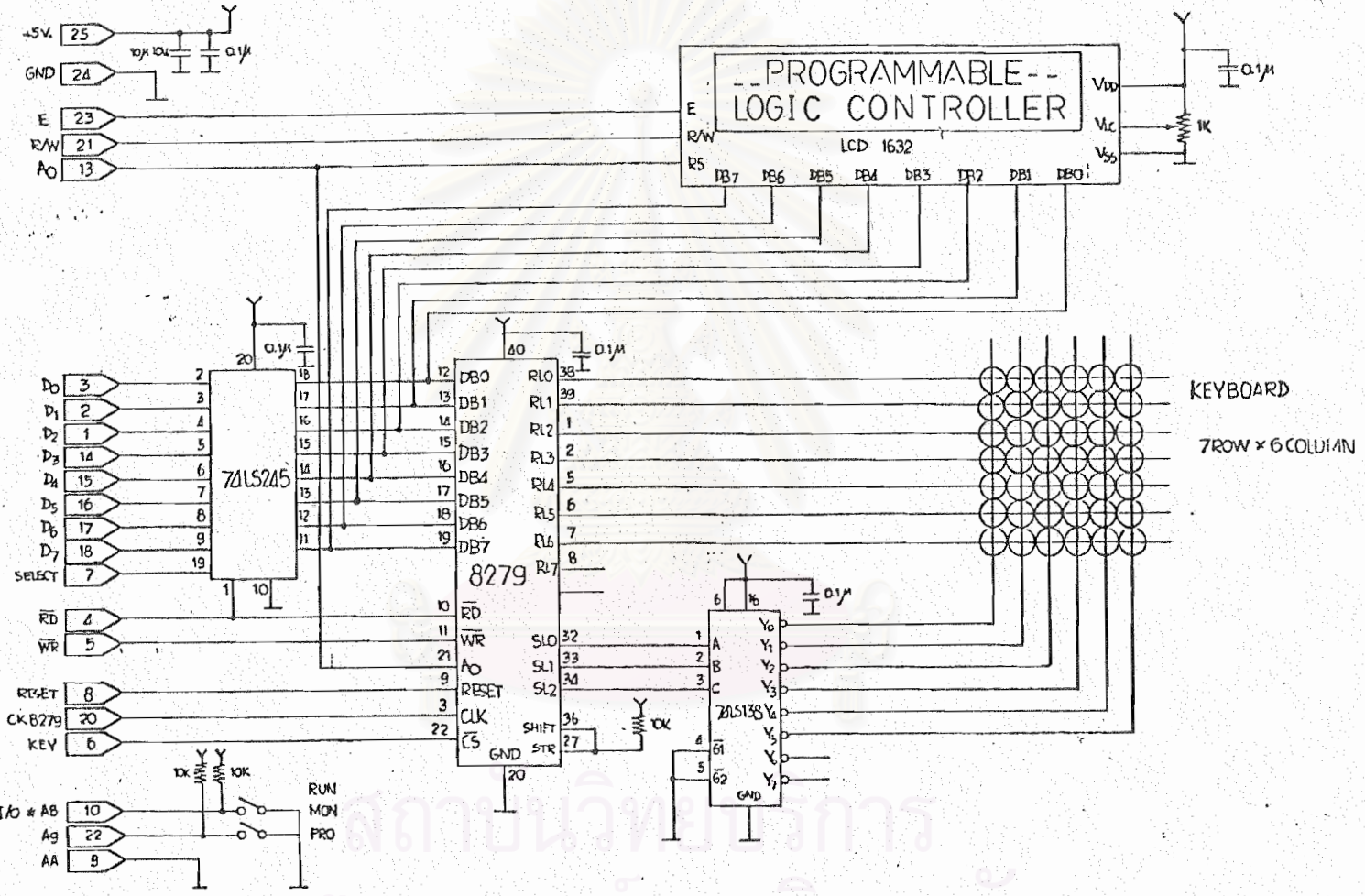
รูปที่ ก.7 แสดงวงจรของโมดูลอนาลอกอินพุท



รูปที่ ก.8 แสดงวงจรของโมดูลอนาลอกอินพุท (ต่อ)



รูปที่ ก.9 แสดงวงจรของโมดูลกำหนดค่าตัวเลข



รูปที่ ก.10 แสดงวงจรของตัวป้อนโปรแกรมแบบมือถือ

ภาคผนวก ข

คุณสมบัติของเครื่อง

ลักษณะสมบัติจำเพาะของเครื่อง PC ที่ออกแบบสร้าง เป็นดังนี้คือ

ลักษณะโครงสร้าง	แบบโมดูล
ลักษณะตัวถัง	โลหะเหล็กพ่นสี
แรงดันไฟ	5 vdc, 24 vdc
อุปกรณ์หลักในการควบคุม	LSI, TTL
การควบคุม	โปรแกรมเก็บไว้ในเครื่อง
ภาษาที่โปรแกรม	ภาษาขั้นบันได (Ladder)
จำนวนคำสั่ง	64 คำสั่ง
ความยาวของคำสั่ง	3 ถึง 10 ไบท์
ความเร็วการทำงาน	เฉลี่ย 5.7 μsec/ คำสั่ง (เบื้องต้น)
จำนวนโปรแกรมสูงสุด	ประมาณ 4000 คำสั่ง

จำนวนอินพุท/เอาท์พุท	512 จุด
จำนวนรีเลย์ช่วย (AUX relay)	464 จุด
จำนวน Holding relay	512 จุด
จำนวน Timer/Counter	128 ตัว
จำนวน Data memory	512 คำ (16 บิท/คำ)
จำนวน Temporary relay	8 จุด
จำนวนรีเลย์พิเศษ	40 จุด
การป้องกันข้อมูล	HR relay, Counter, Data memory ยังคงสถานะเดิมเมื่อไฟดับ
แบตตอรี่	3 ปี
การตรวจสอบความผิดพลาด	CPU ผิดพลาด (วงจรไทเมอร์) ถอดรื้อออก Battery failure Memory error Program error

ลักษณะสมบัติจำเพาะของโมดูลอินพุท

แรงดันอินพุท	DC 24 V ± 10%
ความต้านทานอินพุท	1.8 K
กระแสอินพุท	10 mA
จำนวนอินพุท	8/16 จุด กราวด์ร่วม
แหล่งจ่ายไฟฟ้าเลี้ยงวงจร	DC 5 V ± 5%

ลักษณะสมบัติจำเพาะของโมดูลเอาท์พุท

แรงดันเอาท์พุทสูงสุด	AC 220 V, DC 24 V
กระแสเอาท์พุท	AC 2 A, DC 24
ชนิดของเอาท์พุท	หน้าสัมผัสรีเลย์
แหล่งจ่ายไฟขับรีเลย์ภายใน	DC 24 V ± 10%
แหล่งจ่ายไฟฟ้าเลี้ยงวงจร	DC 5 V ± 5%
จำนวนเอาท์พุท	8/16 จุด กราวด์ร่วม

ลักษณะสมบัติจำเพาะของโมดูลอนาลอกอินพุท

ชนิดของสัญญาณอินพุท	กระแส/แรงดัน
แรงดันอินพุท	1-5 VDC
กระแสอินพุท	4-20 mA
ความละเอียด	12 บิท
เอาท์พุท	0000-OFFFH (HEX)
แหล่งจ่ายไฟของวงจร	5 VDC ± 5% , 24 VDC ± 10%

ลักษณะสมบัติจำเพาะของโมดูลกำหนดค่าตัวเลข

ชนิดของตัวเลข	BCD 4 หลัก/แชลแนล
จำนวนแชลแนล	4 แชลแนล
แหล่งจ่ายไฟของวงจร	5 VDC ± 5%

ภาคผนวก ค

คู่มือการใช้เครื่องควบคุมชนิดโปรแกรมได้ (IIRL-III)

การใช้เครื่อง PC ที่สร้างจะแบ่งการใช้งานออกเป็น 2 โหมดคือ

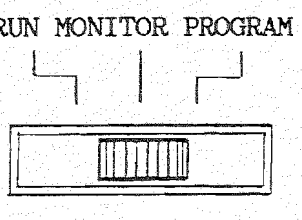
1. โหมดโปรแกรม (Program)
2. โหมดทำงาน (Run)

หน้าที่การทำงานของทั้งสองโหมดคือ โหมดโปรแกรมใช้สำหรับการป้อนโปรแกรม การแก้ไขโปรแกรมต่างๆ โหมดทำงานใช้สำหรับให้เครื่องทำงานตามโปรแกรมชั้นบันไดที่ผู้ใช้เขียน การแสดงชนิดของโหมดที่ใช้งานอยู่ดูได้จาก LED ที่โมดูลแสดงผล หรือดูจากจอLCD ขณะที่เปิดเครื่อง

การเลือกโหมดที่ใช้

ถ้าตัวป้อนโปรแกรมต่ออยู่กับเครื่อง PC เราจะเลือกโหมดโดยการเลื่อนสวิทช์ที่ตัวป้อนโปรแกรม ซึ่งมีตำแหน่งในการเลื่อน 3 ตำแหน่งคือ RUN, MONITOR และ PROGRAM เมื่อเลื่อนสวิทช์ไปที่ตำแหน่ง "PROGRAM" จะเป็นการทำงานโหมดโปรแกรม เมื่อเลื่อนสวิทช์ไปที่ตำแหน่ง "RUN" หรือตำแหน่ง "MONITOR" จะหมายถึงเป็นการทำงานโหมดทำงาน (RUN)

ถ้าโมดูลแสดงผล คีย์บอร์ดและสื่อสาร หรือตัวป้อนโปรแกรมไม่ได้ต่ออยู่กับเครื่อง PC เครื่องจะทำงานในโหมดทำงาน (RUN) โดยอัตโนมัติ



การใช้คีย์บอร์ด

คีย์บอร์ดของตัวป้อนโปรแกรมมีขนาด 7 แถว x 6 คอลัมภ์ ซึ่งมีบางคีย์บอร์ดที่ไม่ได้ใช้งาน รูปของคีย์บอร์ดแสดงในรูปที่ ค.1

FUN	SFT	NOT			SHIFT
AND	OR	CNT	TR .		HR
LD	OUT	TIM	DM	CH *	CONT #
7	8	9			SRCH
E 4	F 5	6	SET	DEL	MONTR
B 1	C 2	D 3	RESET	INS	
A 0		CLR		WRITE	

รูปที่ ค.1 แสดงตำแหน่งคีย์บอร์ดของเครื่อง PC

คีย์บอร์ดต่างๆ เหล่านี้สามารถแบ่งออกตามกลุ่มการใช้งานได้ดังนี้

คีย์ตัวเลข

คีย์เหล่านี้ ได้แก่ คีย์ตัวเลข Ø-9 ซึ่งใช้ในการป้อนค่าตัวเลข สำหรับข้อมูลของคำสั่งขั้นบันไดต่างๆ ใช้ป้อนหมายเลขขั้น(Step) ใช้ป้อนหมายเลขของตัวตั้งเวลา ตัวนับและรีเลย์ต่างๆ

การป้อนตัวเลขฐานสิบหก A-F นั้นจะใช้คีย์ Ø ถึงคีย์ 5 ร่วมกับการคีย์ SHIFT โดยกดคีย์ SHIFT ตามด้วยหมายเลขเหล่านั้น

คีย์ CLR

ใช้สำหรับการลบการแสดงผลที่จอ LCD ใช้ในการลบข้อมูลคำสั่งขั้นบันได และใช้ในการป้อน Password

คีย์ควบคุมการทำงาน

เป็นคีย์ที่ใช้เป็นคำสั่งควบคุมการทำงานต่าง ๆ ของเครื่องได้แก่

⇧ ใช้เลื่อนโปรแกรมไปข้างหลัง 1 ขั้น

⇩ ใช้เลื่อนโปรแกรมไปข้างหน้า 1 ขั้น

WRITE ใช้เขียนคำสั่งลงในหน่วยความจำ

INS ใช้แทรกคำสั่งลงในหน่วยความจำ

DEL ใช้ลบคำสั่งออกจากหน่วยความจำ

SRCH ใช้ในการค้นหาคำสั่ง

MONTR ใช้ดูสถานะของรีเลย์ และข้อมูล

SET ใช้เซ็ทสถานะรีเลย์ที่ Monitor อยู่ให้เป็น ON

RESET ใช้เซ็ทสถานะรีเลย์ที่ Monitor อยู่ให้เป็น OFF

คีย์คำสั่ง

ใช้ในการป้อนคำสั่งต่าง ๆ รวมทั้งชนิดของโอเปอร์แรนด์ ได้แก่

FUN ใช้เลื่อนคำสั่งฟังก์ชั่นต่าง ๆ โดยกดคีย์ FUN ตามด้วยหมายเลขฟังก์ชั่น

SFT ใช้ป้อนคำสั่ง SFT(10)

NOT ใช้ป้อนคำสั่งของหน้าสัมผัสปกติปิด (NC)

AND ใช้สำหรับคำสั่ง AND

OR ใช้สำหรับคำสั่ง OR

CNT ใช้สำหรับคำสั่ง Counter หรือหมายถึงรีเลย์ Counter

LD ใช้สำหรับคำสั่ง LD

OUT ใช้สำหรับคำสั่ง OUT

TIM ใช้สำหรับคำสั่ง Timer หรือหมายถึงรีเลย์ Timer

TR ใช้สำหรับป้อน Temperary relay

HR ใช้สำหรับป้อน Holding relay

DM ใช้สำหรับป้อน Data memory

ใช้สำหรับป้อนค่าคงที่ตัวเลข

CH ใช้ป้อนค่า Channel ในการ Monitor

CONT ใช้ป้อนรีเลย์ในการ Monitor

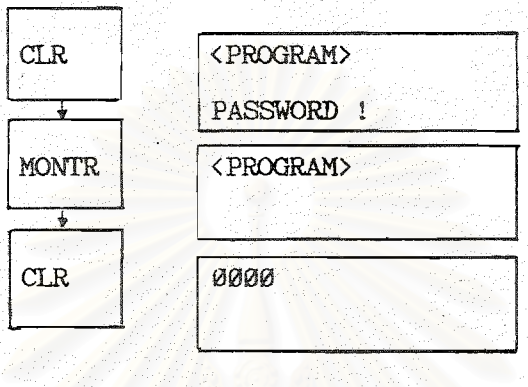
การป้อน PASSWORD ของเครื่อง

เมื่อเริ่มเปิดเครื่อง (Power on) จะต้องป้อน Password ให้ถูกต้องก่อนจึงสามารถใช้ตัวป้อนโปรแกรมสำหรับคำสั่งต่าง ๆ ได้ และถ้าผู้ใช้เลื่อนสวิทช์เปลี่ยนโหมดที่ใช้งาน เครื่องจะไม่เปลี่ยนโหมดให้จนกว่าจะป้อน Password แล้ว

การป้อน Password ของเครื่องทำโดยการกดปุ่ม CLR MONTR CLR ตามลำดับ

โหมดการใช้งาน

RUN	MONITOR	PROGRAM
X	X	X

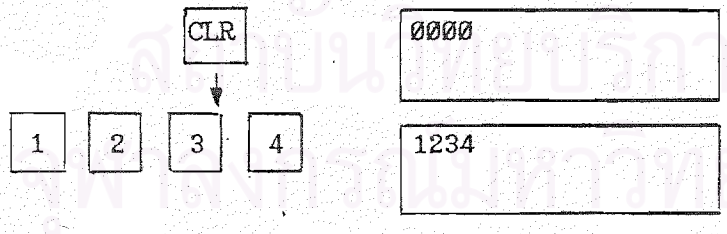


การกำหนดแอดเดรส

เป็นการกำหนดแอดเดรส (Step) ของโปรแกรมขั้นบันไดเพื่อจะใช้งานกับการทำงานอื่นได้แก่ การอ่าน การเขียน การเพิ่ม การลบ การค้นหา การใช้งานโดยกดปุ่ม CLR ตามด้วยหมายเลขแอดเดรสที่ต้องการ

โหมดการใช้งาน

RUN	PROGRAM
X	X



ถ้าต้องการแอดเดรส 0000 หลังจากกดคีย์ CLR แล้ว ไม่จำเป็นต้องกดตัวเลขก็ได้ การกำหนดแอดเดรสนี้เครื่องจะไม่แสดงโปรแกรมที่แอดเดรสนั้น ถ้าต้องการดูโปรแกรมที่แอดเดรสนั้นต้องกดคีย์ ⇧ หรือ ⇩

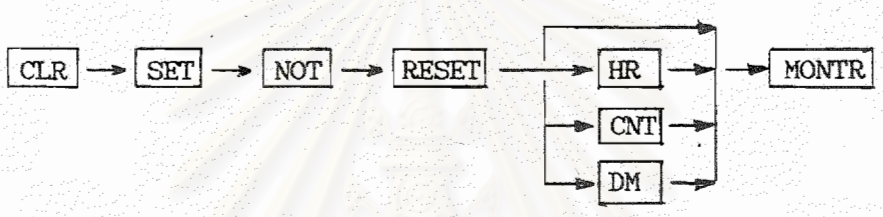
การลบโปรแกรมทั้งหมด

เป็นคำสั่งสำหรับลบโปรแกรมชั้นบันไดทั้งหมดในหน่วยความจำ หลังจากทำงานแล้วโปรแกรมในหน่วยความจำจะเป็นคำสั่ง NOP(ØØ) ทั้งหมด การลบโปรแกรมทั้งหมดนี้เครื่องจะลบข้อมูลใน Holding relay, Counter และ Data memory ด้วย ถ้าไม่ต้องการให้ลบข้อมูลต่าง ๆ เหล่านี้จะต้องกดคีย์ชนิดของข้อมูลนั้นด้วย

โหมดการใช้งาน

RUN	PROGRAM
-	X

รูปแบบการใช้



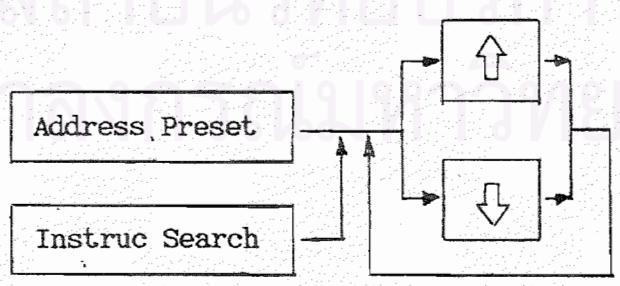
การอ่านโปรแกรม

เป็นคำสั่งสำหรับอ่านโปรแกรมชั้นบันไดที่เก็บอยู่ในหน่วยความจำ

โหมดการใช้งาน

RUN	PROGRAM
X	X

รูปแบบการใช้



- เมื่อกดคีย์ ⇧ จะเป็นการแสดงโปรแกรมที่ตำแหน่งก่อนหน้าหนึ่งตำแหน่ง
- เมื่อกดคีย์ ⇩ จะเป็นการแสดงโปรแกรมที่ตำแหน่งถัดไปหนึ่งตำแหน่ง
- ถ้าแอดเดรสที่กำหนดมีค่ามากกว่าแอดเดรสสูงสุดของโปรแกรม เครื่องจะไม่ทำงานคำสั่งนี้

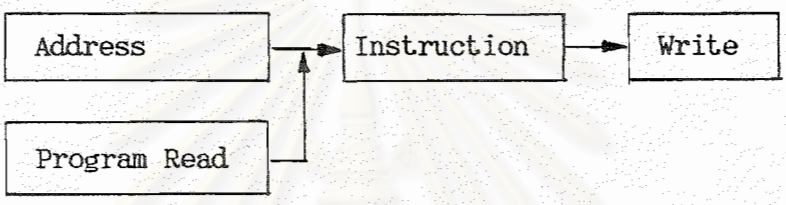
การเขียนคำสั่ง

เป็นการเขียนคำสั่งที่แสดงบนจอ LCD ขณะนั้นลงไปในหน่วยความจำ ณ ตำแหน่งแอดเดรสที่แสดงบนจอ LCD คำสั่งเดิมที่เก็บในหน่วยความจำ ณ ตำแหน่งนั้นจะถูกแทนด้วยคำสั่งใหม่ที่เขียนลงไป

โหมดที่ใช้งาน

RUN	PROGRAM
-	X

รูปแบบการใช้งาน



- หลังจากทำงานแล้ว เครื่องจะนำคำสั่งที่ตำแหน่งถัดไปมาแสดงบนจอ LCD ให้ดู
- เมื่อคำสั่งใหม่ที่จะเขียนมีความยาวกว่าคำสั่งเดิม และหน่วยความจำไม่พอเก็บ คำสั่งนี้จะไม่ถูกทำงาน

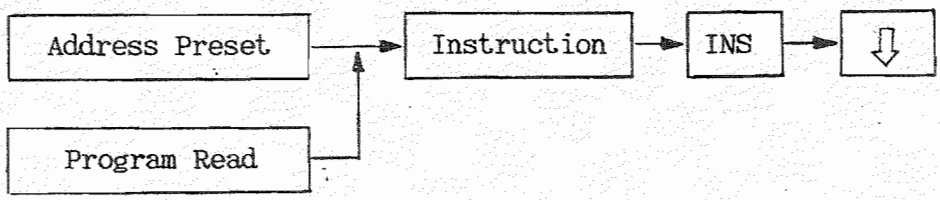
การเพิ่มคำสั่ง

เป็นการแทรกคำสั่งที่แสดงบนจอ LCD ลงไป ณ ตำแหน่งแอดเดรสที่แสดงบนจอ LCD โปรแกรมเดิมตั้งแต่แอดเดรสนั้นจนจบโปรแกรมจะถูกเลื่อนลงไปข้างล่าง

โหมดการใช้งาน

RUN	PROGRAM
-	X

รูปแบบการใช้



- หลังจากทำงานแล้วเครื่องจะนำคำสั่งที่แอดเดรสต่อไปมาแสดงที่จอ LCD
- ถ้าหน่วยความจำที่เก็บโปรแกรมเต็ม คำสั่งนี้จะไม่ทำงาน

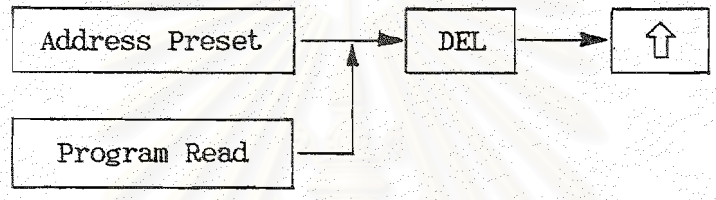
การลบคำสั่ง

เป็นการลบคำสั่ง ณ ตำแหน่งแอดเดรสที่กำหนดบนจอ LCD ออกจากหน่วยความจำและคำสั่งตั้งแต่ตำแหน่งแอดเดรสต่อไปจนจบโปรแกรมจะถูกเลื่อนมาข้างหน้าด้วย

โหมดการใช้งาน

RUN	PROGRAM
-	X

รูปแบบการใช้



- เมื่อกดคีย์ DEL เครื่องจะนำคำสั่ง ณ ตำแหน่งแอดเดรสนั้นจากหน่วยความจำออกมาแสดงที่จอ LCD อีกครั้งเพื่อให้แน่ใจ และจะลบเมื่อกดคีย์ ⇧
- หลังจากทำคำสั่งนี้แล้วเครื่องจะนำคำสั่งใหม่ ณ ตำแหน่งแอดเดรสเดิมมาแสดงที่จอ LCD

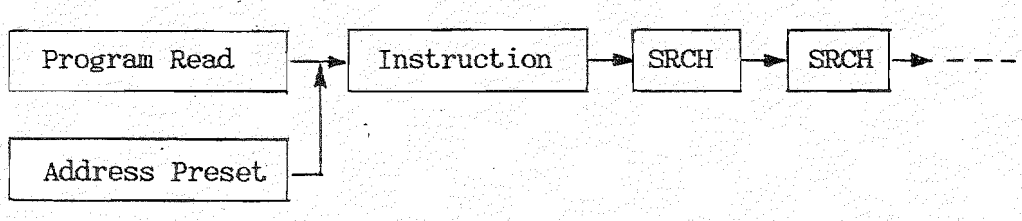
การค้นหาคำสั่ง

เป็นการหาคำสั่งที่กำหนดว่าอยู่ที่ตำแหน่งเท่าใดในโปรแกรม และนำมาแสดงที่จอ LCD การค้นหาจะเริ่มจากแอดเดรสขณะนั้นไปเรื่อย ๆ จนพบคำสั่งนั้น หรือพบคำสั่ง END จึงหยุดและนำคำสั่งนั้นมาแสดงที่จอ LCD ถ้าการค้นจนถึงคำสั่งสุดท้ายแล้วยังไม่พบก็จะแสดงข้อความบอกว่าหาไม่พบ

โหมดการใช้งาน

RUN	PROGRAM
-	X

รูปแบบการใช้



การใช้คำสั่งค้นหาสามารถใช้ต่อเนื่องได้หมายถึงว่า เมื่อกดคีย์ SRCH จะเป็นการหา

คำสั่งนั้นตัวแรก ถ้าต้องการหาตัวที่สอง, สามต่อ ก็กดคีย์ SRCH หาตัวต่อไปได้ โดยจะต้องห้ามกดตัวอื่นในระหว่างนั้น

การดูสถานะ (Monitor)

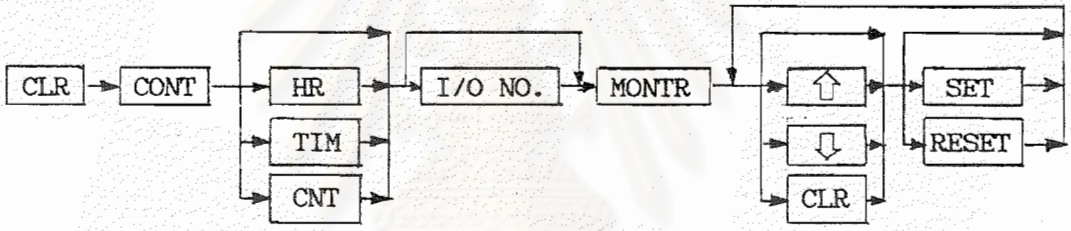
การดูสถานะจะมี 2 อย่างคือ ดูสถานะของรีเลย์ว่าเป็น ON หรือ OFF และดูค่าข้อมูลของแชลแนล ซึ่งแสดงเป็นเลขฐานสิบหกจำนวน 4 หลัก

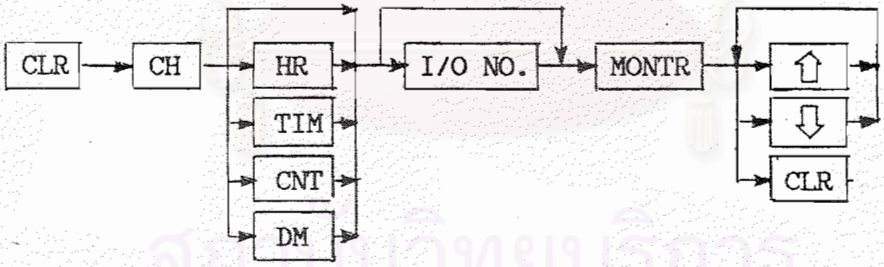
การดูสถานะของรีเลย์แล้วผู้ใช้สามารถ เซ็ท/รีเซ็ทข้อมูลของรีเลย์ที่แสดงสถานะนั้นได้ โดยการกดคีย์ SET หรือ RESET

โหมดการใช้งาน

RUN	PROGRAM
X	X

รูปแบบการใช้





ขณะที่กำลังดูสถานะของรีเลย์ หรือข้อมูลอยู่ผู้ใช้สามารถกดคีย์ ⇧ เพื่อดูสถานะของตำแหน่งถัดไปข้างบน หรือกดคีย์ ⇩ เพื่อดูสถานะของตำแหน่งถัดมา

การใช้งานโหมดทำงานของเครื่องอาจพบข้อผิดพลาด ซึ่งจะแสดงโดย LED ที่โมดูลแสดงผลและแสดงที่จอ LCD ซึ่งผู้ใช้จะต้องเปลี่ยนโหมดมาเป็นโหมดโปรแกรมก่อนแล้ว จึงแก้ไขที่ผิดพลาดให้ถูกต้อง

ข้อผิดพลาดที่อาจเกิดขึ้นได้แก่

"NO END INSTR" หมายถึงไม่มีคำสั่ง END ในโปรแกรมขั้นบันได เครื่องไม่สามารถทำงานได้ ผู้ใช้ต้องกลับไปโหมดโปรแกรมเพื่อป้อนคำสั่ง END ก่อน

"MEMORY ERR" หมายถึงมีคำสั่งขั้นบันไดขั้นบันได บางคำสั่งที่เก็บอยู่ในหน่วยความจำมีการผิดพลาด ผู้ใช้ต้องกลับไปโหมดโปรแกรมและลบคำสั่งที่ผิดพลาดทิ้ง

"LOW BATT" หมายถึงแบตตอรี่ที่ต่อกับหน่วยความจำมีกำลังไฟต่ำ แล้วควรจะเปลี่ยนใหม่

คำสั่งขั้นบันไดต่างๆ ที่ใช้เขียนโปรแกรมมีทั้งสิ้น 64 คำสั่ง ซึ่งมีรายละเอียดอยู่ในบทที่ 4 แล้ว