ระบบ
ควบคุมเครื่องปรับอากาศ
ขนาดใหญ่

AIR CONDITIONING
SYSTEM CONTROL

โดย

นาย ศุภชัย มั่งคั่ง
นาย สงบ[illegible] สังขจินดา
นาย สุรินทร์ จิตบุญญาพินิจ

ปริญญานิพนธ์ฉบับนี้
เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาอุตสาหกรรมศาสตร์บัณฑิต
สาขาเทคโนโลยีคอมพิวเตอร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้า
เจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2535

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
สาขาวิชาเทคโนโลยีคอมพิวเตอร์อุตสาหกรรม
คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า

เจ้าคุณทหารลาดกระบัง

เรื่อง

ระบบควบคุมเครื่องปรับอากาศขนาดใหญ่

AIR CONDITIONING SYSTEM CONTROL

โดย

นาย ศุภชัย	มั่งคั่ง	34162231
นาย สงบ	ส้งขจินดา	34162232
นาย สุรินทร์	จิตบุญญาพินิจ	34162240

..... อาจารย์ที่ปรึกษา

(ผศ. พิพัฒน์ เลาหสงคราม)

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เป็นการศึกษาโครงสร้างการใช้งาน Chip Microcontroller ตระกูล MCS-51 ซึ่งได้เลือกเบอร์ 80C31 และการนำมาประยุกต์ใช้ในการควบคุมระบบปรับอากาศ ไม่ว่าจะเป็นแบบติดหน้าต่าง แบบติดข้างฝา แบบแขวน หรือรวมทั้งเครื่องปรับอากาศขนาดใหญ่ เพราะตัว chip microcontroller มีความเหมาะสมที่จะนำมาประยุกต์ควบคุมเครื่องปรับอากาศได้ ซึ่งข้อดีของ chip microcontoller มี port ที่ติดต่อกับอุปกรณ์ภายนอกได้สะดวก

จากการศึกษาโครงสร้างการใช้งาน chip microcontroller เบอร์ 80C31 และระบบควบคุมเครื่องปรับอากาศขนาดใหญ่ ทำให้ทราบว่าสามารถนำมาใช้ในการควบคุมเครื่องปรับอากาศขนาดใหญ่ได้ จะเน้นชนิด chiller typeการทำความเย็นโดยน้ำเย็นที่ถูกขับด้วยปั๊มแรงดันสูง ให้ไหลผ่านคอยล์เย็นของคอมเพรสเซอร์ และส่งไปตามท่อไปยังที่ต่าง ๆ ที่ต้องการทำความเย็น การทำงานของระบบปรับอากาศขนาดใหญ่นี้จำเป็นต้องมีอุปกรณ์ทำงานร่วมกับคอมเพรสเซอร์ เช่น ปั๊มน้ำเย็น ปั๊นน้ำระบายความร้อน และพัดลมเป่าระบายความร้อน อุปกรณ์ดังกล่าวทั้งหมดจะทำงานร่วมกัน หากตัวใดตัวหนึ่งขัดข้องจะมีผลต่อระบบทั้งหมด โดยทั่วไปแล้วการควบคุมเครื่องปรับอากาศชนิดนี้ ยังใช้ระบบ manual อยู่การเปิด-ปิดของปั๊มและ คอมเพรสเซอร์แต่ละชุดจะเป็นอิสระไม่ขึ้นต่อกัน เนื่องจากอุปกรณ์ดังกล่าวนั้นมีอยู่ด้วยกันหลายชุด และแต่ละชุดจะเป็นอิสระในการทำงาน

จากหลักการทำงานดังกล่าวสามารถนำ microcontroller มาประยุกต์ใช้ในการควบคุมระบบปรับอากาศให้มีประสิทธิภาพกว่าระบบเก่า โดยการนำมาควบคุมการเปิด-ปิดของปั๊มน้ำและคอมเพรสเซอร์ให้เป็นไปตามที่กำหนด โดยโปรแกรม การสตาร์ทของคอมเพรสเซอร์และ ปั๊มแต่ละตัวจะกินกระแสมากในการเริ่มเดินจึงมีการกำหนดห้วง-เวลาเพื่อไม่ให้เกิดการกระชากของกระแส และหากว่าคอมเพรสเซอร์หรือปั๊มตัวใดตัวหนึ่งเกิดการขัดข้องก็จะส่งสัญญาณ trip มาให้ซีพียูรับทราบเพื่อไม่ให้เกิดการเสียหายแก่ระบบ ซีพียูจะสั่งให้ระบบหยุดทำงาน และแจ้งสัญญาณ alarm ทางบัสเซอร์พร้อมกับบอกตำแหน่งที่เกิดการขัดข้องโดยแจ้งผลผ่านทาง lcd เมื่อเหตุการณ์ขัดข้องได้รับการแก้ไขระบบก็พร้อมที่จะทำงานได้ใหม่

โดยทั่วไปแล้วตามสถานที่ต่าง ๆ ที่ใช้ระบบปรับอากาศชนิด chiller type นี้จะมีชุดคอมเพรมเซอร์ และปั้นน้ำสองชุดขึ้นไป เพื่อใช้สำรองในกรณีชุดใดชุดหนึ่งเกิดการขัดข้อง หรือกรณีที่มีการใช้โหลดมากในช่วงกลางวันก็จะใช้ทั้งสองชุดทำงานร่วมกัน แต่เมื่อโหลดลดลงหรือในช่วงอากาศเย็นความจำเป็นในการใช้งานจะเหลือเพียงชุดเดียว เพื่อเป็นการประหยัดพลังงานจึงใช้ชุดควบคุมอุณหภูมิตรวจจับความเย็นและส่งสัญญาณให้ซีพียู สั่งให้ชุดทำความเย็นชุดใดชุดหนึ่งทำงานหรือทำงานร่วมกันทั้งสองชุด เมื่อเลือกสวิทช์มาที่โหมด AUTO ถ้าอุณหภูมิสูงกว่า 22 องศาเซลเซียล ซีพียูจะสั่งให้ทั้งสองชุดทำงานพร้อมกัน ถ้าอุณหภูมิต่ำกว่า 18 องศาเซลเซียล ชุดทำความเย็นจะทำงานเพียงชุดเดียว ส่วนในโหมด manual เราสามารถเลือกชุดใดชุดหนึ่ง หรือทั้งสองชุดทำงานร่วมกันได้

เงื่อนไขการทำงานต่าง ๆ ถูกกำหนดด้วยโปรแกรม และถูกนำไปเก็บไว้ใน EPROM ซึ่งสามารถนำไปแก้ไขได้ตามต้องการ

Thesis : Air conditioning system control
(Chiller Type)

Student : Mr.Supachai mangkang 34.162231
Mr.Sa-ngob Sangkhachinda 34.162232
Mr.Surin Chitboonyapinit 34.162240

Advisor : Mr.Pipat Lalhasongkram

Education level : Bachelor of industrial instrument technology

Education year : 1992

Abstract

The purpose of this thesis is intended to study the structure and application of "MCS-51" series single ship micro controller which utilize the "80C31" to control and operation of air conditioning system emphasis here on chiller type.

In general there are many kind of air conditioning system but most of familiar type are small type such as window type or split type that we are not mention here.

Chiller type air conditioning system is a big one wildely use most in big area demanded high vollum of colled air. In an operation a set of chiller type air conditioning system consist of at least chiller compressor,chilled water pump,colling

water pump,cooling fan motor and other acessories. In practical most of early models each of pumps and compressor are running in seperately and dependent controlled by manual operation. When any of these out of order the consequence can be effective to other devices that leaded system into failure.

To utilize microcontroller with this system the main point that 8031 microcontroller ship has been assigned to act as following.

First,to drive each of motor pumps and compressor through I/o port and driver part by specify step and period of starting time in difference in order to protection inrush current at starting.

Second,when any of these component in failure the trip signal is sent through input port to cpu. To prevent of any damage the system must be stopped automatically,alarm signal is loudly and positioning failure can be reading on lcd display.The system ready to operation again after problem has been solved.

Third, here we have two system of air conditioning system cpu will assign one or two system working due to temperature sensor and temperature setting value in program.

Farther develloping and application can be done by the same principle as above.

สารบัญ

บทที่ 1

บทนำ

ในปัจจุบันนี้ เทคโนโลยีสมัยใหม่ได้เข้ามามีบทบาทมากเป็นอย่างยิ่งและยังมีความสำคัญอีกด้วย นอกจากนี้ยังมีอิทธิพลต่อความเป็นอยู่ในชีวิตประจำวันเป็นอย่างมาก และมีแนวโน้มจะเพิ่มความสำคัญยิ่งขึ้นในอนาคต ไม่ว่าจะเป็นวงการอุตสาหกรรม วิทยาศาสตร์ ด้านธุรกิจ การแพทย์ และเกษตรกรรม เป็นต้น เพื่อให้ทันกับสมัย เวลา และประสิทธิภาพสูงสุด จึงได้นำเอาเทคโนโลยีชั้นสูงเข้ามาใช้ ไม่ว่าจะเป็นคอมพิวเตอร์ หรือ ระบบควบคุมอัตโนมัติ จะถูกนำมาใช้อย่างกว้างขวาง แทนระบบที่ต้องพึ่งพามนุษย์เป็นผู้ควบคุม จึงต้องศึกษาเรียนรู้เกี่ยวกับตัวไมโครโปรเซสเซอร์ในด้านต่าง ๆ

ฉะนั้น จุดประสงค์ของการทำโปรเจ็กนี้ ก็เพื่อที่จะให้ผู้จัดทำเกิดความเข้าใจในการทำงาน การออกแบบ การนำไมโครโปรเซสเซอร์ไปประยุกต์ใช้งานในระบบควบคุมอัตโนมัติ

วัตถุประสงค์ของปริญญานิพนธ์

เพื่อให้นักศึกษาที่ผ่านปริญญานิพนธ์นี้แล้ว นักศึกษาจะสามารถ

1. อธิบายโครงสร้างของตัวไมโครโปรเซสเซอร์ได้
2. อธิบายการทำงานของตัวไมโครโปรเซสเซอร์ได้
3. นำมาใช้งานในด้านการควบคุม ร่วมกับอุปกรณ์อื่นได้

ขอบเขตของปริญญานิพนธ์

เพื่อเป็นการศึกษาและการนำไมโครคอนโทรเลอร์มาประยุกต์ใช้กับงานควบคุมระบบเครื่องปรับอากาศขนาดใหญ่(CHILLER TYPE) โดยกำหนดให้ไมโครคอนโทรเลอร์ทำหน้าที่ ดังนี้

- ใช้เป็นตัวขับอุปกรณ์ของระบบปรับอากาศ เช่น มอเตอร์ปั๊ม และคอมเพรสเซอร์
- ใช้เป็นตัวรับค่าสัญญาณที่เกิดจากการขัดข้องของระบบ เช่น เกิดจากการ over current ของอุปกรณ์แต่ละตัว และสั่งให้ระบบหยุดทำงานพร้อมกับส่งสัญญาณเตือน

การขัดข้อง

- ใช้ในการแสดงผลโดยแสดงผลออกทางจอ LCD เช่น แสดงเวลาการ running ของมอเตอร์หรือคอมเพรสเซอร์ในช่วงสตาร์ท แสดงตำแหน่งของการขัดข้องว่าเกิดจากอุปกรณ์ตัวใดซึ่งสามารถทราบที่มาของเหตุขัดข้องได้ในทันที และใช้ตรวจเช็คอุณหภูมิเพื่ออ่านค่าในขณะนั้น ๆ ได้
- ใช้ควบคุมอุณหภูมิโดยกำหนดให้ระบบปรับอากาศมีสองชุด การเลือกให้ชุดใดชุดหนึ่งทำงาน หรือทั้งสองชุดทำงานพร้อมกันขึ้นอยู่กับค่าอุณหภูมิที่กำหนดไว้ในโปรแกรม

บทที่ 2

ระบบการทำงานของเครื่องปรับอากาศขนาดใหญ่

AIR CONDITIONING SYSTEM CONTROL (CHILLER TYPE)

การทำงานของระบบปรับอากาศ

เครื่องปรับอากาศทั่ว ๆ ไปที่ใช้อยู่ตามบ้านหรือสำนักงานเล็ก ๆ มักจะใช้ชนิด window typeโดยให้ตัว commpressor และ evaporator อยู่ใน package เดียวกัน หรือ ชนิดsplit type โดยให้ตัว commpressor อยู่ภายนอกอาคาร และ evaporator หรือชุดคอยล์เย็นอยู่ภายในอาคาร ซึ่งเป็นเครื่องปรับอากาศขนาดเล็กและมีอุปกรณ์ควมคุมไม่ยุ่งยากมากนัก ส่วนเครื่องทำความเย็นตามอาคารใหญ่ ๆ ที่ใช้ในสถานที่ซึ่งต้องการความเย็นมาก ๆ โดยทั่วไปจะใช้น้ำเย็น(chilled water) จากตัวทำความเย็น(chiller) ซึ่งถูกขับโดย pump แรงดันสูง มายังจุดต่าง ๆ ที่ต้องการทำ-ความเย็น น้ำเย็นดังกล่าวจะไหลผ่านตัว air handling unit หรือ fancoil unit พัดลมที่อยู่ในอุปกรณ์ดังกล่าวจะเป่าลมผ่านคอยล์เย็น ซึ่งทำให้ความเย็นออกมาตามต้องการ

หลักการทำงานเบื้องต้นของเครื่องปรับอากาศ ตัว compressor ซึ่งถูกขับโดย motor จะทำหน้าที่ดูดน้ำยาที่เป็นแกส และอัดส่งไปยัง discharge line ซึ่งมีอุณหภูมิ และแรงดันสูงผ่าน condensor ทำให้น้ำยาที่มีสถานะเป็นแกสกลายเป็นของเหลว ซึ่งใช้หลักการของการควบแน่น ส่งผ่านไปเข้า expansion valve น้ำยาจะถูกลดแรงดันลง และฉีดเข้าไปยัง evaporator และน้ำยาที่เป็นแกสจะวิ่งกลับเข้ากลับเข้าท่อทางดูด (suction line) ใน compressor เมื่อ evaporator เย็นลง ความเย็นจะถูกนำพาไป โดยอาจจะใช้พัดลมเป่าเพื่อให้ความเย็นกระจาย ซึ่งนิยมใช้กับเครื่องปรับอากาศขนาดเล็ก ส่วยในเครื่องปรับอากาศขนาดใหญ่ จะใช้น้ำเป็นตัวนำความเย็น ไปยังตัวกระจายความเย็น (air handling unit หรือ fancoil unit) ส่วนความร้อนที่มาจากตัว condensor ในระบบปรับอากาศขนาดเล็กจะใช้พัดลมเป่า เพื่อระบายความร้อนโดยตรง ส่วนในระบบปรับอากาศขนาดใหญ่จะใช้น้ำเป็นตัวนำความร้อน เพื่อไประบายความร้อน และจะใช้พัดลมเป่าน้ำเพื่อระบายความร้อนอีกทอดหนึ่ง

จากหลักการคร่าวๆ ดังกล่าว ชุดทำความเย็นที่ใช้ chiller นี้จะประกอบด้วยอุปกรณ์หลัก ๆ ดังต่อไปนี้

1. ตัวchiller ซึ่งมีมอเตอร์สำหรับขับcommpressor อยู่ในตัวซึ่งจะเป็นตัวที่ใช้กำลังสูง เพื่ออัดน้ำยา(refrigerant) สำหรับทำให้เกิดความเย็น

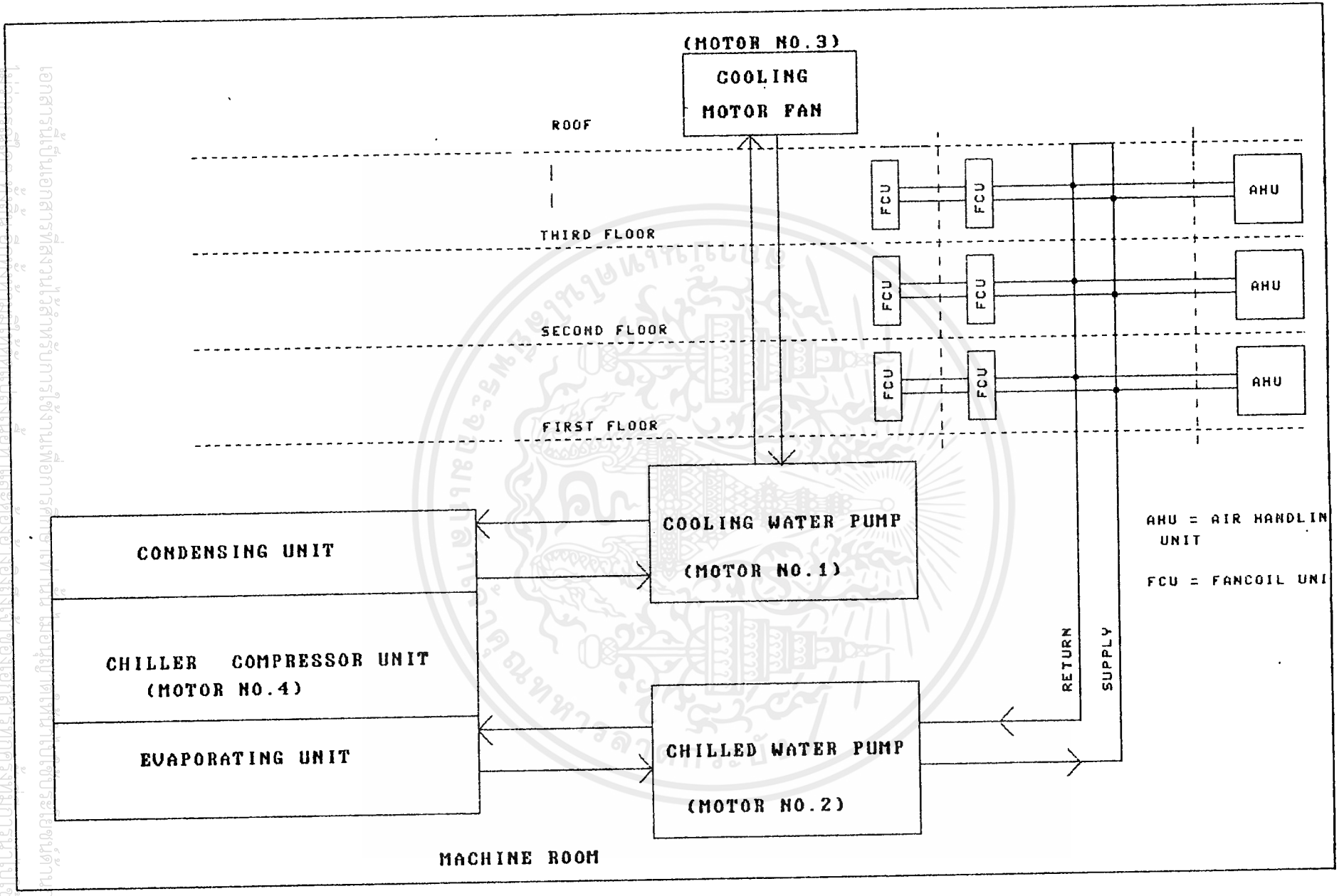
2. chilled water pump เป็นตัวขับน้ำเย็นที่ไหลผ่าน ขดลวดความเย็น (evaporator) ใน chiller ไปยังส่วนต่าง ๆ ที่ต้องการทำความเย็น โดยใช้ fan-coil unit (fcu) หรือ airhandling unit เป็นตัวกระจายความเย็น

3. cooling water pump เป็นตัวขับน้ำผ่านตัวระบายความร้อนจาก-condencing unit ซึ่งอยู่ภายในตัว chiller ไปยังตัวระบายความร้อนภายนอกอาคาร หรือส่วนบนสุดของอาคาร

4. fan cooling motor เป็นพัดลมเป่าระบายความร้อนจากน้ำร้อนที่มาจาก cooling water pump ซึ่งนำความร้อนมาจาก condencing unit เพื่อระบายความร้อนสู่อากาศภายนอก

นอกจากนี้ยังมีอุปกรณ์ควบคุมต่าง ๆ อีกมากแต่จะไม่ขอกล่าวในที่นี้

Block diagram แสดงการทำงานของระบบปรับอากาศ chiller type ดังแสดงในรูป



BLOCK DIAGRAM แสดงการทำงานของเครื่องปรับอากาศชนิด CHILLER TYPE

บทที่ 3

การเชื่อมต่อ CPU กับระบบเครื่องปรับอากาศ

การส่งและรับค่าของซีพียูกับอุปกรณ์ภายนอก ติดต่อได้โดยผ่านไอโอพอร์ทของไอซีเบอร์ 8255 และพอร์ทของซีพียูเอง โดยมีรายละเอียดดังนี้

1. <u>การควบคุมมอเตอร์ปั๊มและคอมเพรสเซอร์</u> ซีพียูจะควบคุมมอเตอร์ปั๊มและคอมเพรสเซอร์โดยผ่านทางพอร์ทบีของไอซีเบอร์8255(PB.0-PB.7) กำหนดให้เป็นเอาท์พุทพอร์ทผ่านไปยังชุดไดร์เวอร์ซึ่งใช้ไอซีชนิด OPTO ISOLATOR SW. ทำงานร่วมกับ TRIAC ซึ่งตัว TRIAC นี้จะช่วยแก้ปัญหาในการ SPIKE ของหน้าสัมผัสแทนที่จะใช้รีเลย์ โดยทั่วไปโดยกำหนดให้ PB.0-PB.3 ไปควยคุมมอเตอร์ปั๊มละคอมเพรสเซอร์ชุดที่หนึ่ง และให้ PB.4-PB.7 ควบคุมมอเตอร์และคอมเพรสเซอร์ชุดที่สอง ดังแสดงในรูป 3.1

2. <u>การรับสัญญาณที่เกิดจากการขัดข้องของระบบ</u> กำหนดให้พอร์ทเอของไอซีเบอร์ 8255 เป็นอินพุทพอร์ท(PA.0-PA.7) รับสัญญาณที่เกิดจากการ TRIP ของมอเตอร์ปั๊มและคอม-เพรสเซอร์อาจเกิดจากการโอเวอร์เคอร์เรนท์ หรือสาเหตุอื่น ๆ ผ่านไปยังซีพียู โดยโดยกำหนดให้ PA.0-PA.3 รับสัญญาณ TRIP ของระบบปรับอากาศชุดที่หนึ่ง และ PA.4-PA.7 รับสัญญาณ TRIP ของระบบปรับอากาศชุดที่สอง ในสภาวะปกติที่ขาของ PA.0-PA.7 จะถูกพูลอัพไว้ด้วย Vccซึ่งจะมีค่าเป็นหนึ่งอยู่ตลอด และสัญญาณดังกล่าวต่อร่วมกับ AND GATE โดยให้เอาท์พุทของ AND GATE ส่งค่าไปยังขาอินพุทของซีพียู ซึ่งสภาวะปกติจะมีค่าเป็นหนึ่ง เมื่อเกิดการ TRIP ของอุปกรณ์ตัวใดตัวหนึ่งของ AND GATE จะมีค่าเป็นศูนย์ และทำให้ซีพียูสั่งให้มอเตอร์ปั๊มและคอมเพรสเซอร์หยุดการทำงานพร้อมกันนั้นบิตที่เป็นศูนย์ซึ่งเกิดจากการขัดข้องถูกส่งไปยังซีพียู และแสดงผลบอกนำแหน่งโดยออกทางจอ LCD รูป 3.2

3. <u>การควบคุมการเปิด-ปิดของเครื่องปรับอากาศ</u> กำหนดให้ PC.0-PC.3ของไอซีเบอร์ 8255 เป็นอินพุทพอร์ททำหน้าที่รับสัญญาณจาก PUSH BUTTON SWITCH ดังนั้น

PC.0-PC.1 เป็น START และ STOP SW. ของระบบปรับอากาศชุดที่หนึ่ง

PC.2-PC.3 เป็น START และ STOP SW. ของระบบปรับอากาศชุดที่สอง

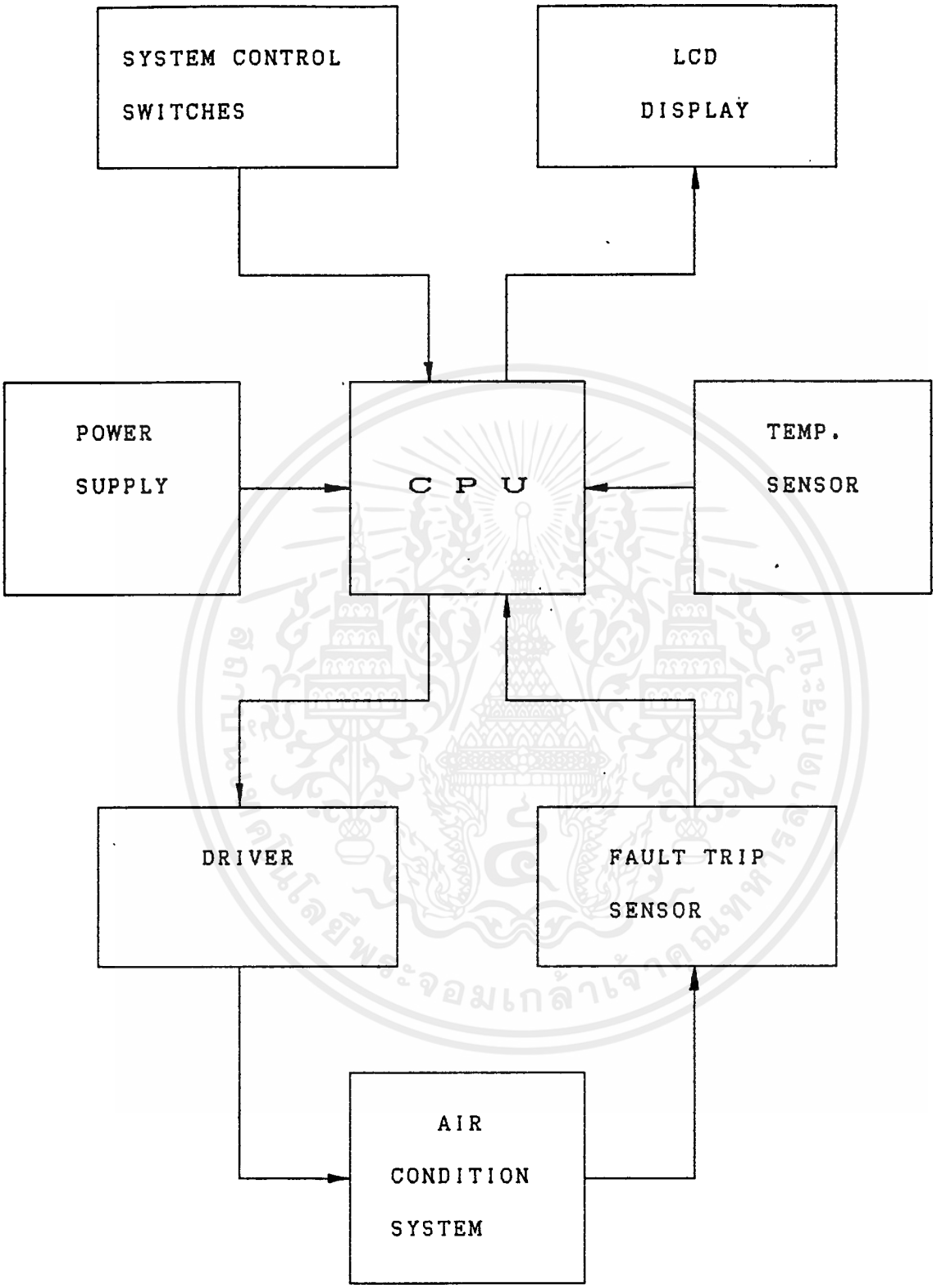
เมื่อ SELECTOR SW.อยู่ในตำแหน่ง MANNUAL เราสามารถ ON-OFF ระบบปรับอากาศทั้งสองระบบได้โดยตรงและเมื่อ SELECTOR SW.อยู่ในตำแหน่ง AUTO การทำ ON-OFF

ของเครื่องปรับอากาศชุดที่สองถูกควบคุมโดยชุดควบคุมอุณหภูมิ ซึ่งส่งสัญญาณผ่านทางPC.5 มาขับรีเลย์โดยให้ CONTACT NC ของรีเลย์ต่ออยู่กับ PC.3 และ NO.ต่อกับ PC.2 ส่วน COMMON RELAY จะต่อผ่าน MANNUAL - AUTO SELECTOR SW. และอีกด้านหนึ่งของสวิทช์ต่ออยู่กับ ground ดังแสดงในรูป 3.3

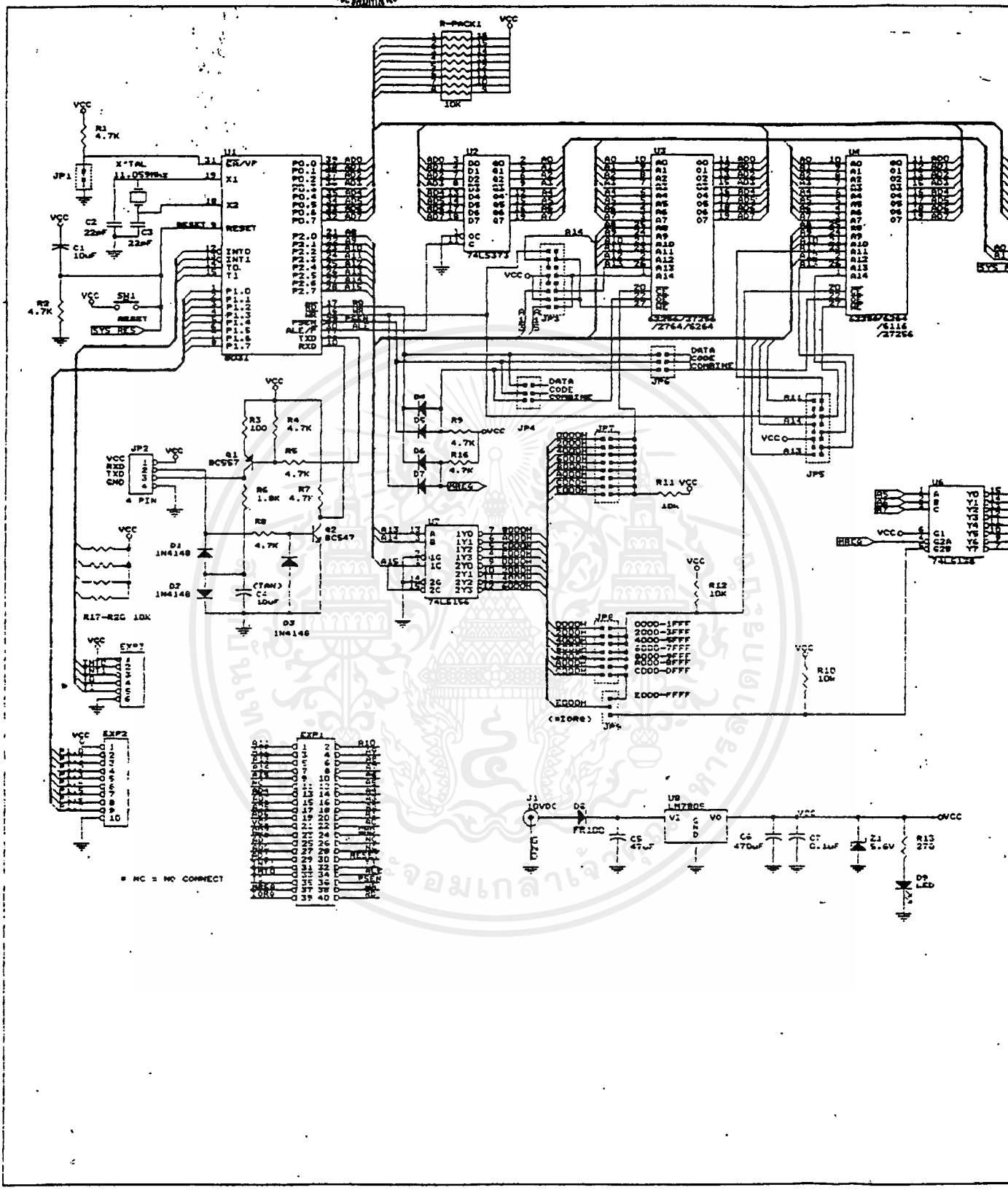
4. ระบบสัญญาณแจ้งการขัดข้องของระบบ เมื่อสัญญาณขัดข้องเกิดขึ้นจากอุปกรณ์แต่ละตัว ซีพียูจะได้รับสัญญาณ INTO. และส่งสัญญาณออกมาที่ PC.4ของไอซีเบอร์ 8255 พร้อมกันนั้นบัสเซอร์จะส่งสัญญาณ alarm แจ้งเหตุให้ทราบ การหยุดการทำงานของบัสเซอร์ทำได้โดยกดปุ่ม ALARM RESET

5. การรับค่าข้อมูลและการตรวจสอบอุณหภูมิ กำหนดให้พอร์ทหนึ่งของซีพียู (Exp.2 ของ PC-SB 31 BARD)เป็นอินพุทพอร์ท รับค่าข้อมูลมาจากชุด ADC 0804 โดยเข้ามาทางขา P1.0-P1.6 ส่วนการดูค่าอุณหภูมิในขณะนั้น ๆ กำหนด P1.7 เป็นอินพุทรับค่าจาก TEMP.CHECK SW. ซึ่งปกติจะพูลอัพไว้

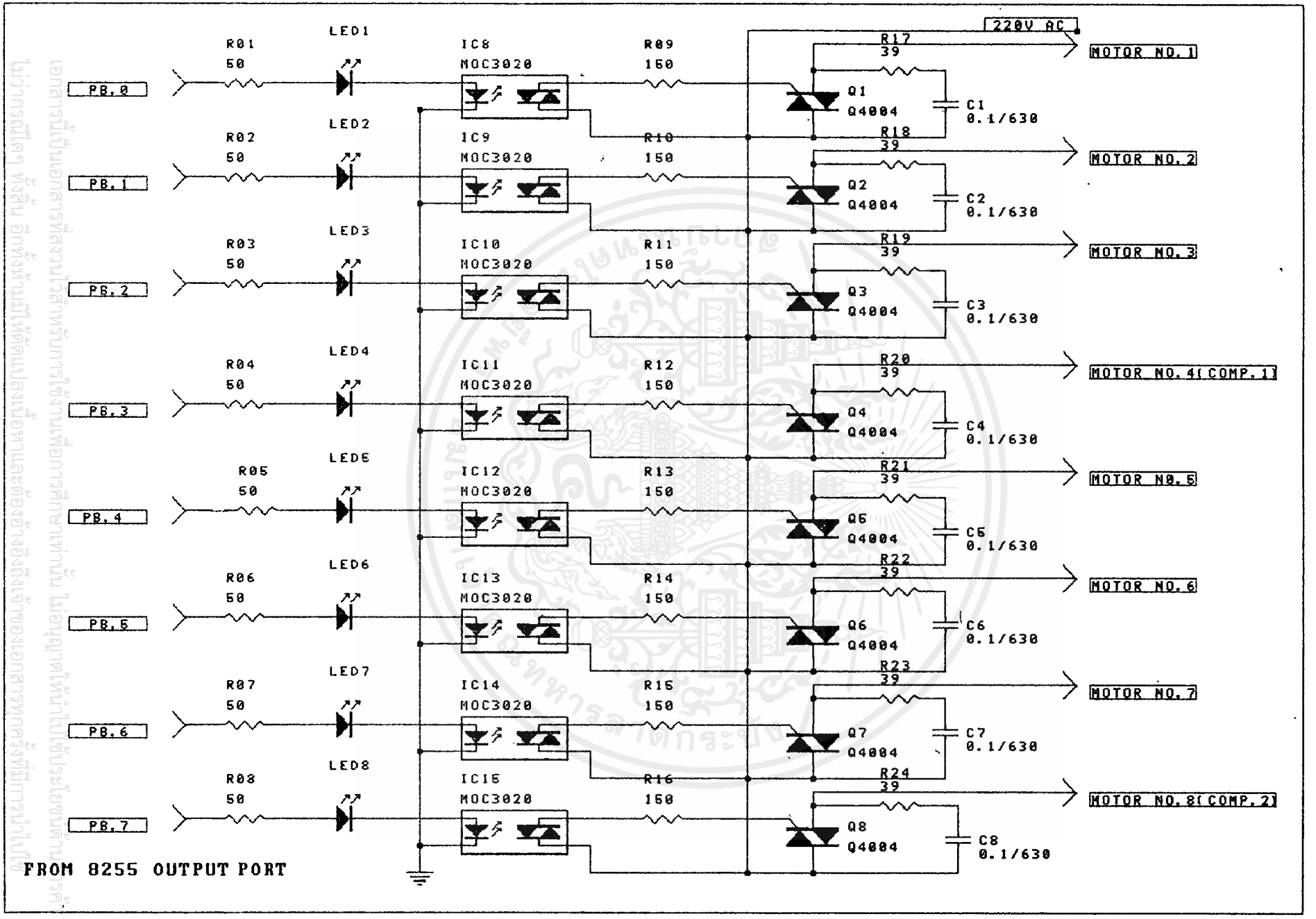
6. LCD DISPLAY กำหนดให้ EXPAN.5 ของ PC-SB 31 เป็นพอร์ทที่ใช้ติดต่อกับLCD โดยตรง กำหนดให้ D0-D7 ต่ออยู่กับขา ดาต้าของ LCD ขา A0 และขา A1 ของแอด-เดรสบัสต่ออยู่กับขา R/W และ RS ของ LCD ตามลำดับส่วนขา E ของ LCD จะต่อร่วมอยู่กับ DECODE ของไอซีเบอร์ 74LS138 ดังแสดงในรูป 3.4



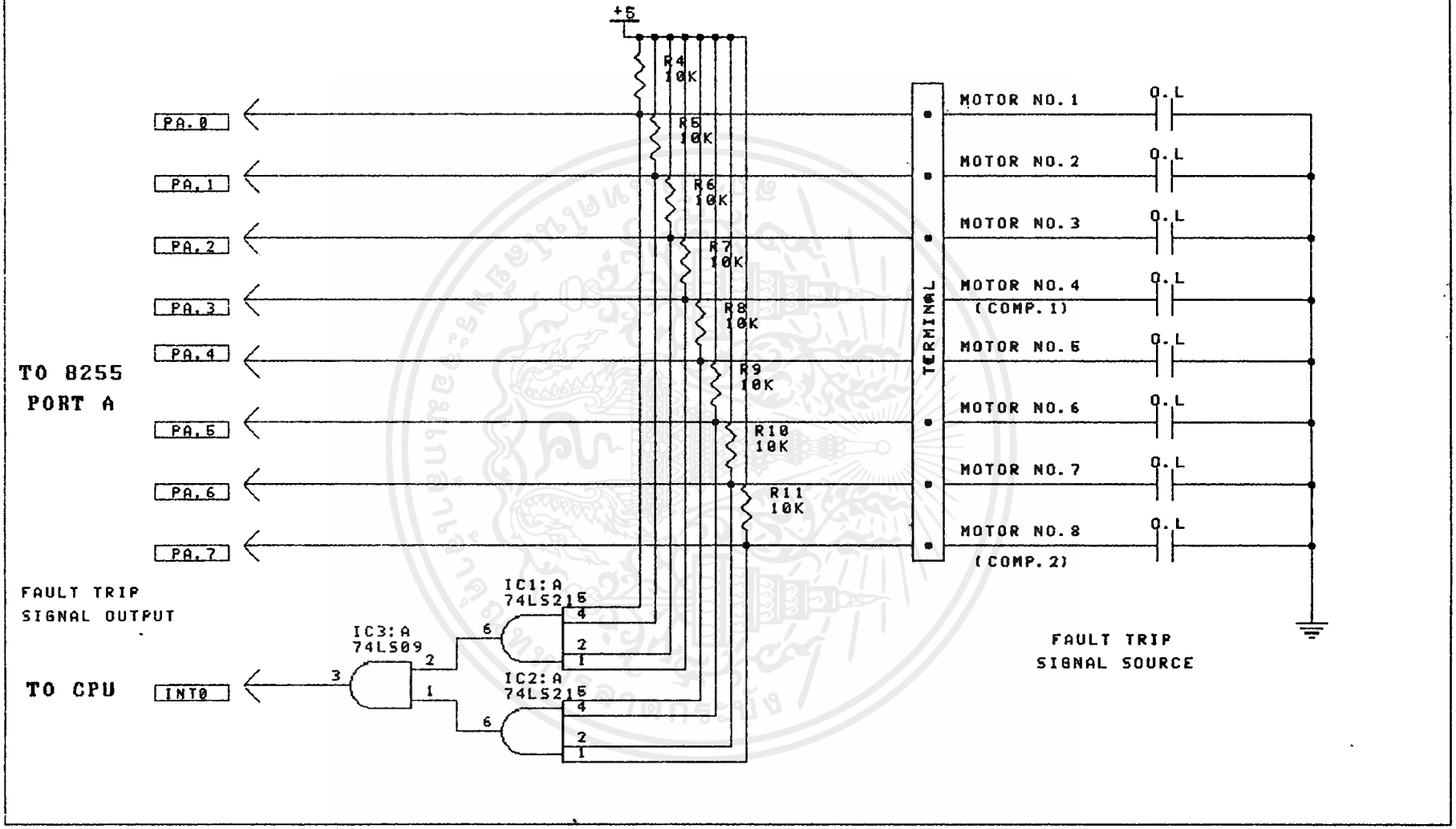
BLOCK แสดงการนำ MICROCONTROLLER มาใช้ควบคุมเครื่องปรับอากาศ



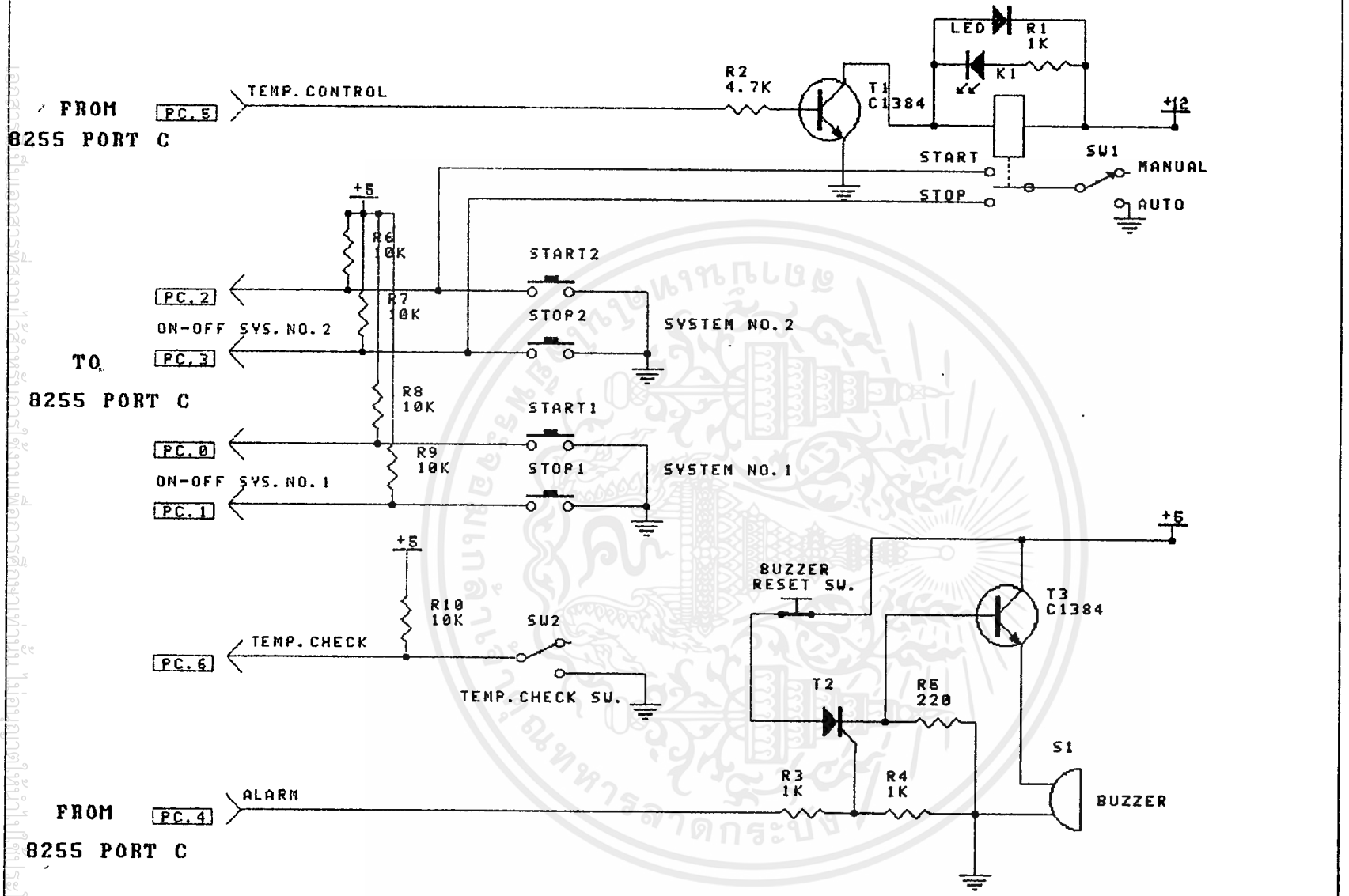
รูปแสดงวงจรของ PC-SB31



รูป 3.1 แสดงวงจร DRIVER

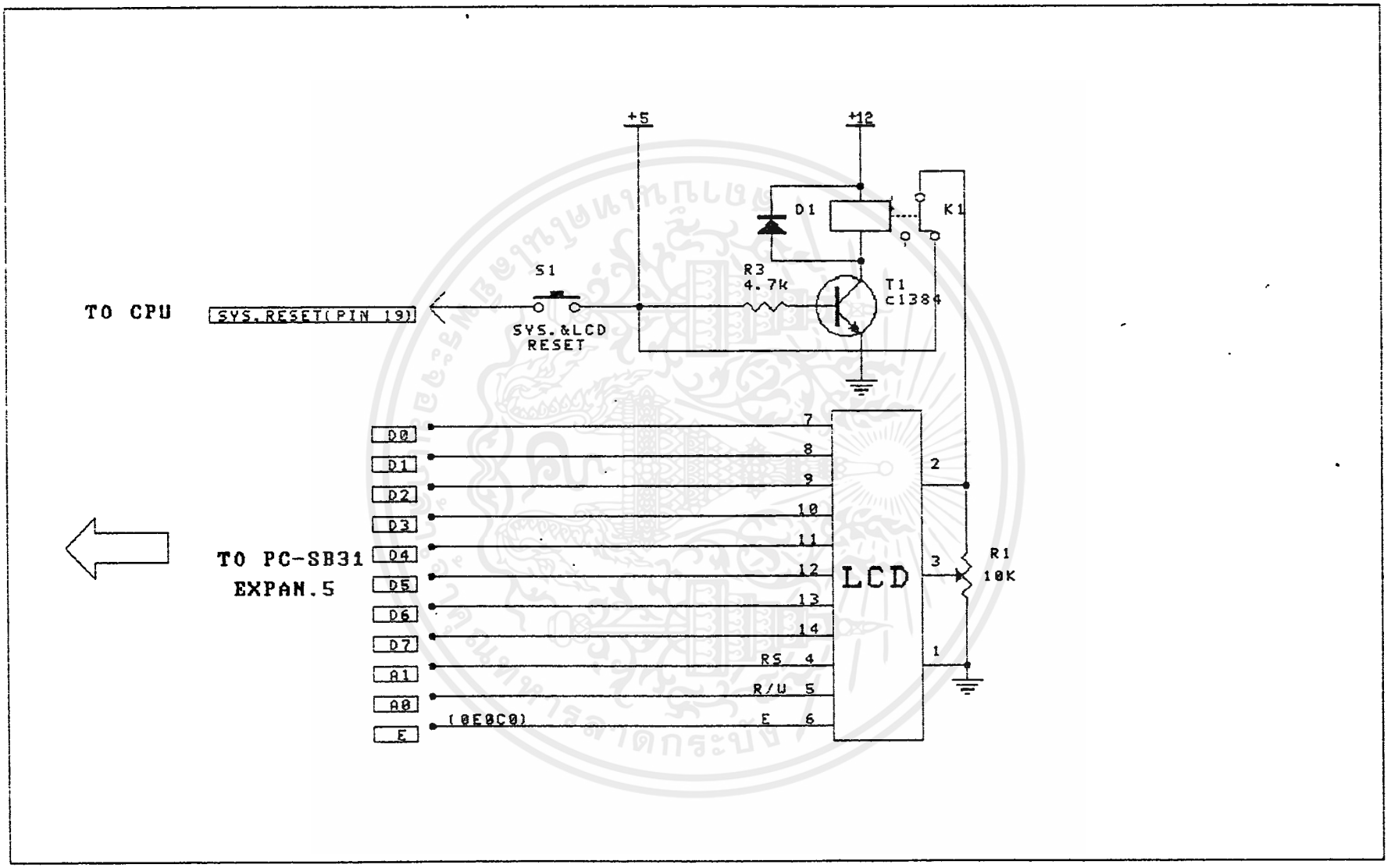


รูป 3.2 แสดงการรับสัญญาณ TRIP และการส่งสัญญาณ INTERRUPT ไปยัง CPU



SYS.CONTROL AND ALARM DIAGRAM

รูป 3.3 การต่องจรควบคุมและBUZZER ALAEM



รูป 3.4 แสดงการต่อ LCD ไปยัง PORT ของ PC-SB31 การ RESET CPU และ LCD

บทที่ 4

การทำงานของไมโครคอนโทรลเลอร์ในการควบคุมเครื่องปรับอากาศ

ในระบบควบคุมที่เราออกแบบมานี้จะใช้ cpu เบอร์ 8031 เป็นตัวควบคุมการทำงานของระบบ ซึ่งการทำงานทั้งหมดของ cpu จะถูกควบคุมด้วยซอฟแวร์ ซึ่งสามารถที่จะอธิบายการทำงานได้ด้วย flow chart ดังรูป

หลังจากเปิดเครื่องแล้ว cpu จะถูกหน่วงเวลาเอาไว้ประมาณ 30 ms. เพื่อให้ภาคจ่ายไฟเสถียรภาพเสียก่อน จึงจะเริ่มการทำงาน

cpu จะเริ่มเข้าสู่การควบคุมการทำงานของระบบปรับอากาศซึ่งมีอยู่ 2 ระบบ คือ system NO.1 และ system NO.2 ได้ก็ต่อเมื่อไม่มีสัญญาณ interrupt เข้ามาที่ขา interrupt โดยการทำงานของระบบปรับอากาศทั้งสอง มีรายละเอียดและลำดับขั้นตอนการทำงานคือ cpu จะทำการ scan switch ที่ใช้ในการควบคุมระบบตามลำดับดังนี้ คือ

1. เมื่อ switch start system NO.1 ถูกกด cpu จะส่งสัญญาณไปควบคุมระบบปรับอากาศชุดที่ 1 (system NO.1) ให้ทำงาน ตามลำดับขั้นตอน คือ เริ่มจาก cpu จะส่งสัญญาณไปควบคุมชุด driver ให้ motor NO.1,2,3 และ 4 สตาร์ท ตามลำดับ โดยมีระยะเวลาการสตาร์ทต่างกัน คือ 5 วินาที,10 วินาที,10 วินาที และ 3 นาที ตามลำดับ หมายความว่า cpu จะถูกหน่วงเวลาด้วยซอฟแวร์ประมาณ 5 วินาที ก่อนแล้วจึงส่งสัญญาณให้ motor NO.1 สตาร์ท จากนั้นก็จะหน่วงเวลา 10 วินาที motor NO.2 จึงถูกสตาร์ท และหลังจากนั้นก็เป็นในลักษณะเดียวกัน คือ cpu จะถูกหน่วงเวลา 10 วินาที แล้วจึงส่งสัญญาณให้ motor NO.3 สตาร์ท และเมื่อ motor NO.3 สตาร์ทไปแล้ว 3 นาที cpu จึงจะสั่งให้ motor NO.4 สตาร์ท จึงเป็นอันว่า ระบบปรับอากาศชุดที่ 1 (system NO.1) ทำงานเรียบร้อยแล้ว จุดประสงค์ในการหน่วงเวลานี้ เนื่องจาก motor pump และ compressor จะกินกระแสมากในการสตาร์ท จึงกำหนดให้เวลาในการสตาร์ทแตกต่างกัน
2. เมื่อ switch start system NO.2 ถูกกด cpu ก็จะส่งสัญญาณไปควบคุม ให้ชุด driver สั่งให้ระบบปรับอากาศชุดที่ 2 (system NO.2) ทำงาน โดยจะมีผลทำให้ motor NO.5,6,7 และ 8 ทำงานเหมือนระบบปรับอากาศชุดที่ 1 ทุกประการ คือ มีระยะเวลาก่อน ช่วงการสตาร์ท motor แต่ละตัวเท่ากัน คือ 5 วินาที,10 วินาที,10 วินาที และ 3 นาที

ข้อสังเกตประการหนึ่งเกี่ยวกับระบบปรับอากาศชุดที่ 2 คือว่า ระบบปรับอากาศชุดที่ 2 จะสามารถทำงานได้ โดยการเลือก mode ควบคุมการทำงานระบบ ซึ่งแบ่งออกเป็น 2 mode คือ

2.1 ถ้าเราเลือก mode การทำงานของระบบปรับอากาศชุดที่ 2 ให้เป็นแบบ manual ดังนั้นเวลาที่เราต้องการให้ระบบปรับอากาศชุดที่ 2 ทำงาน เราจะต้องไปกด switch start system NO.2 ให้อยู่ในสภาวะ on ระบบปรับอากาศชุดดังกล่าวจึงจะทำงานได้

2.2 แต่ถ้าเราเลือก mode การทำงานของระบบปรับอากาศชุดที่ 2 ให้เป็นแบบ auto การทำงานของระบบปรับอากาศชุดที่ 2 จะอยู่ในการควบคุมของชุดควบคุมอุณหภูมิ คือเมื่อ มีอุณหภูมิสูงกว่า 22°C ขึ้นไป หรือตามค่าอุณหภูมิที่กำหนดไว้ในในโปรแกรม ระบบปรับอากาศชุดที่ 2 ซึ่งการควบคุมอยู่ใน mode auto จะถูก CPU สั่งให้ทำงานตามขั้นตอนดังที่ได้กล่าวมาแล้วข้างต้น และ เมื่ออุณหภูมิมีอุณหภูมิต่ำกว่า 18°C ระบบปรับอากาศชุดที่ 2 ก็จะหยุดการทำงานโดยอัตโนมัติ

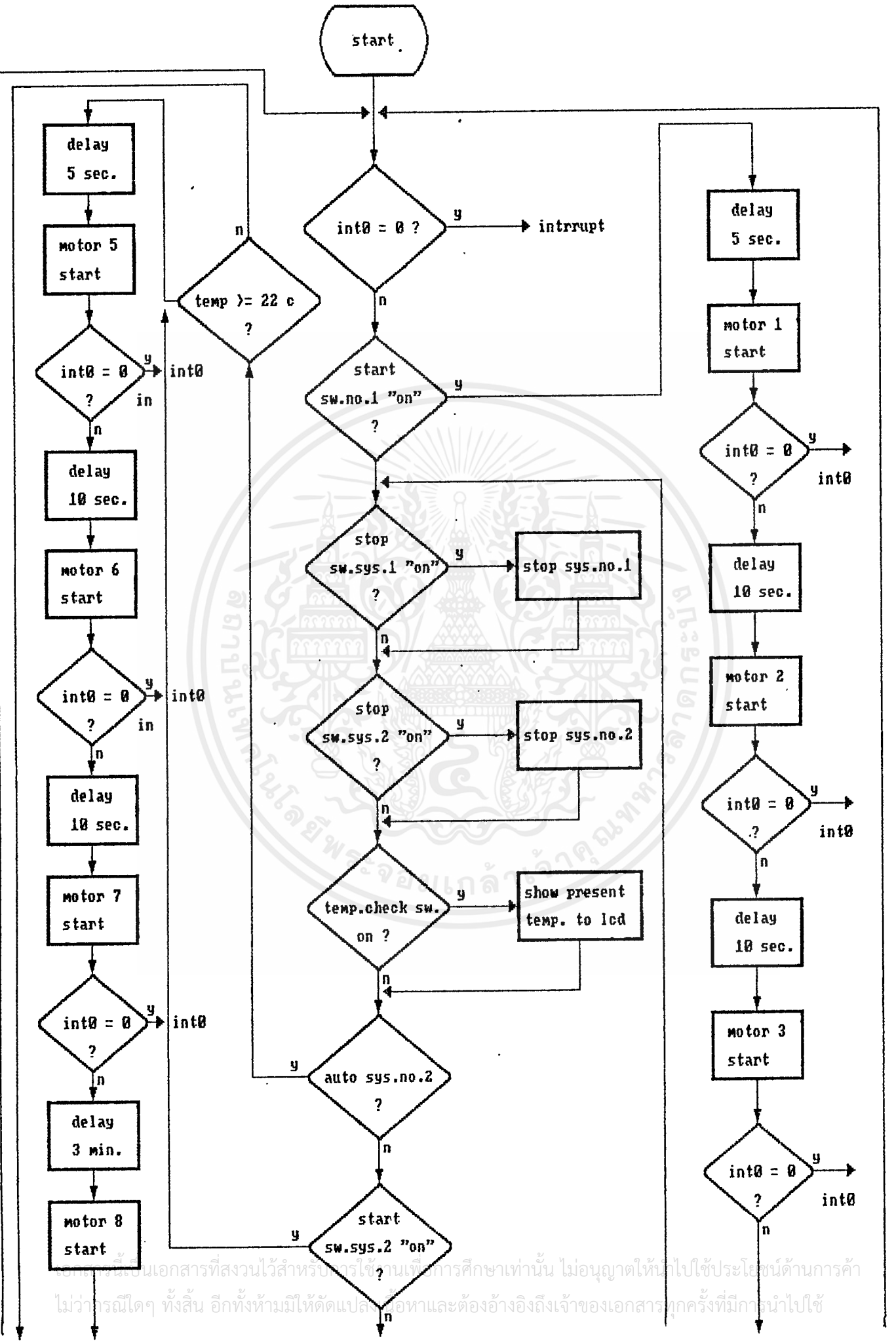
3. เมื่อ switch stop system NO.1 ถูกกด CPU จะสั่งให้ระบบปรับอากาศชุดที่ 1 หยุดการทำงาน

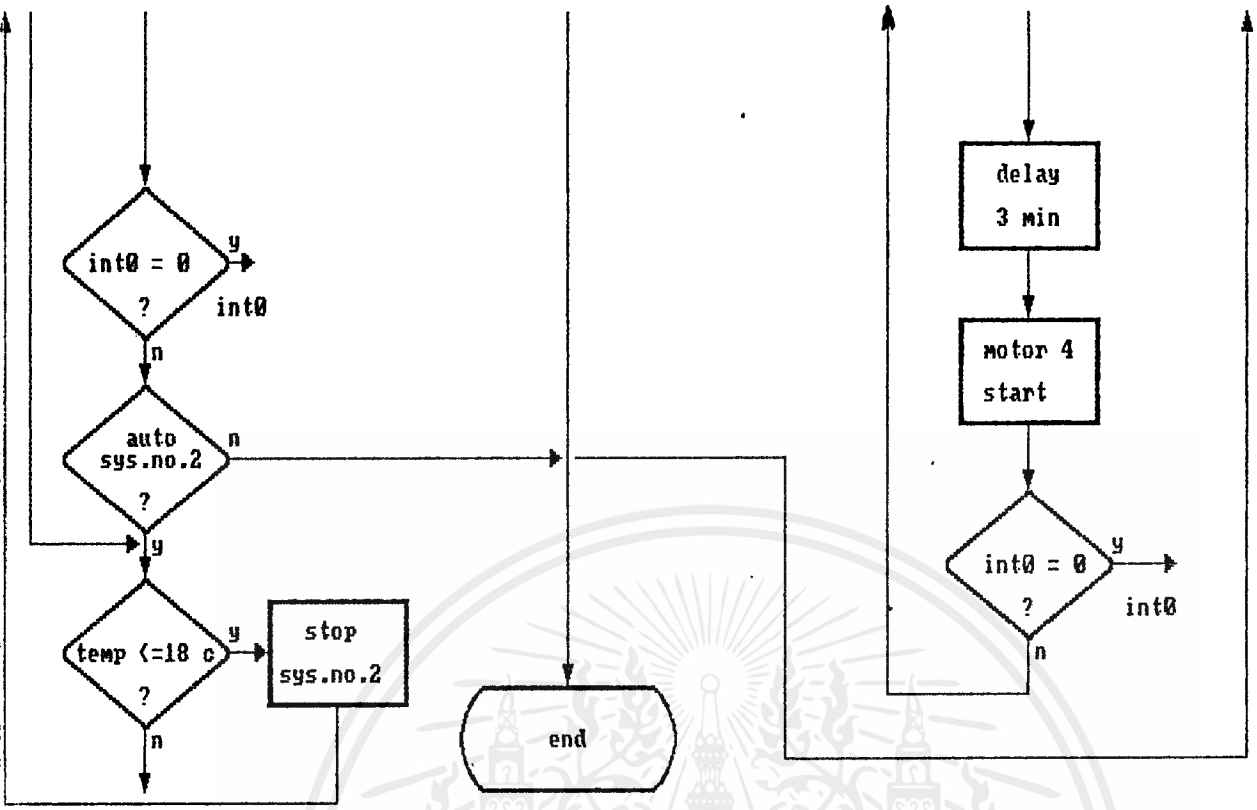
4. เมื่อ switch stop system NO.2 ถูกกด ไม่ว่าจะอยู่ใน mode manual หรือ auto เราสามารถ stop ระบบปรับอากาศชุดที่ 2 ได้ แต่มีข้อควรระวัง คือ เมื่อเรา stop ระบบปรับอากาศชุดที่ 2 ควรให้ selector switch สำหรับเลือก mode ให้ไปอยู่ในตำแหน่ง manual เพราะไม่เช่นนั้นอาจทำให้เครื่องปรับอากาศชุดที่ 2 ทำงานใหม่ได้

5. เมื่อ CPU ได้รับสัญญาณ interrupt นั่นหมายความว่า มีอุปกรณ์ตัวใดตัวหนึ่งของระบบปรับอากาศชุดใดชุดหนึ่งใน 2 ชุดนั้นเกิดขัดข้อง ดังนั้น CPU จะส่งสัญญาณไปควบคุมให้ระบบปรับอากาศในชุดที่มีอุปกรณ์ขัดข้องนั้นหยุดการทำงานทั้งระบบ พร้อมกันนั้นก็จะส่งสัญญาณให้ระบบ alarm ส่งสัญญาณเสียงเตือนให้ผู้ใช้เครื่องทราบ และที่หน้าจอแสดงผล LCD ของเครื่องจะแสดงชื่อและตำแหน่งของอุปกรณ์ที่เกิดขุดข้องให้ผู้ใช้เครื่องทราบเพื่อจะได้ไปทำการแก้ไขได้อย่างถูกต้อง หากผู้ใช้ต้องการ reset ระบบ alarm ก็สามารถทำได้ โดยการกดปุ่ม reset alarm จะทำให้เสียงที่ดังอยู่นั้นหยุดส่งเสียง จากนั้นผู้ใช้จึงสามารถไปทำการแก้ไขอุปกรณ์ที่ขัดข้องนั้น ๆ ได้ตามลำดับต่อไป

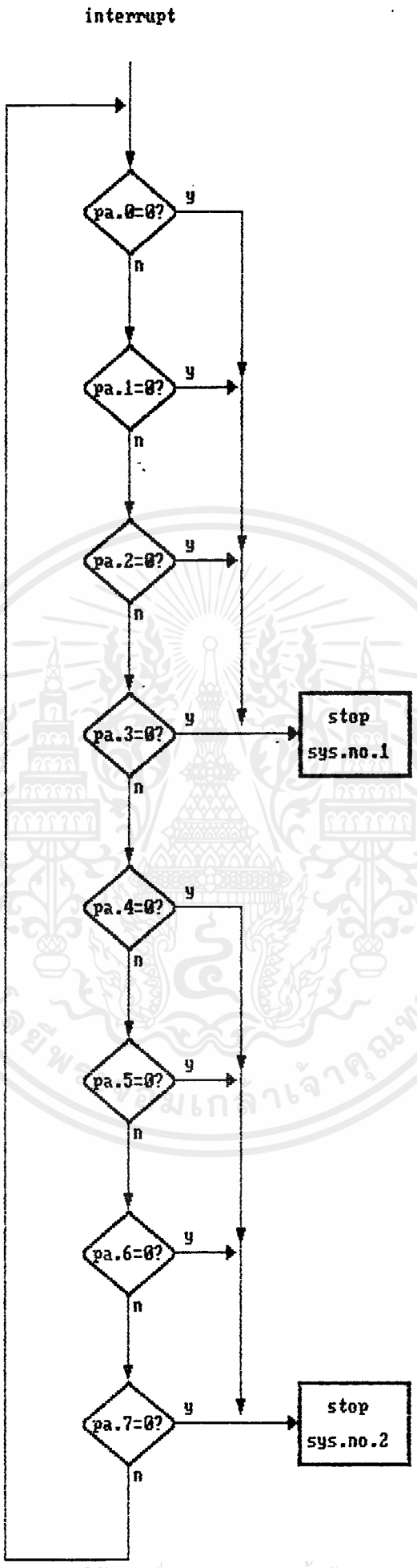
6. การแสดง display จาก block diagram ภาคแสดงผล (LCD display) หรือแสดงสภาวะจะบอกสภาวะการทำงานของระบบ คือเมื่อเปิดเครื่อง LCD จะแสดงสถานะของระบบว่าพร้อมว่าที่จะทำงานหรือไม่ในกรณีที่อุปกรณ์ตัวใดของระบบปรับอากาศชุดใด ๆ เกิดขัดข้อง LCD จะแสดงชื่อและตำแหน่งของอุปกรณ์ตัวที่ขัดข้องนั้น ๆ ให้เรา

ทราบเพื่อที่จะทำให้เราไปทำการแก้ไขอุปกรณ์ตัวนั้น ๆได้อย่างถูกต้องและถ้าไม่มีอุปกรณ์ตัวใดขัดข้อง คือ ทั้งระบบปรับอากาศชุดที่ 1 และชุดที่ 2 (system NO.1, NO.2) อยู่ในสภาวะปกติ LCD จะแสดงข้อความบอกให้รู้ว่าระบบพร้อมที่จะทำงานตามขั้นตอน คือ เมื่อ CPU สั่ง start motor NO.1-8 LCD จะแสดงข้อความบอกให้ทราบว่าขณะนี้ motor ตัวใดของระบบปรับอากาศชุดใดกำลังทำงานอยู่ และพร้อมทั้งจะบอกระยะเวลาหรือระยะห่างที่ motor แต่ละตัวจะทำงานเรียงตามลำดับกันไปตามที่ได้กำหนดไว้ในโปรแกรม และใช้แสดงค่าอุณหภูมิในขณะนั้น ๆโดยการที่เรากด check temp.-switch และนอกจากนี้ LCD จะแสดงสภาวะทุกสภาวะที่เกิดขึ้นกับระบบปรับอากาศทั้งหมด





FLOWCHAT แสดงขั้นตอนการทำงานของระบบ



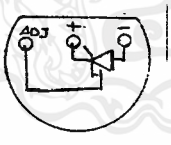
FLOWCHAT แสดงการรับสัญญาณ INTERRUPT

บทที่ 5

การควบคุมอุณหภูมิ (Temperature and Sensor Control)

ตัววัดอุณหภูมิมีวิธีการหลายวิธีในการวัดอุณหภูมิ โดยเครื่องมือแต่ละชนิดอาศัยหลักการเปลี่ยนแปลงคุณสมบัติ เฉพาะของสาร คือ จะต้องมีการเปลี่ยนแปลงที่วัดได้ เมื่ออุณหภูมิที่วัดเปลี่ยนไปและ การเปลี่ยนแปลงที่วัดได้ต้องคงที่แน่นอน ส่วนตัววัดอุณหภูมิทาง-อิเลคทรอนิคส์นั้นจะแสดงค่าอุณหภูมิออกมาเป็นตัวเลข อย่างเที่ยงตรง(บอกเป็นแรงดัน) ตัวอุปกรณ์ที่ว่านี้เราใช้อุปกรณ์ทางเซมิคอนดัคเตอร์ โดยหลักการทำงาน คือ เมื่ออุณหภูมิมีการเปลี่ยนทำให้จังชั่น พี-เอ็น มีการเปลี่ยนแปลงด้วย และลักษณะตัววัดเสมือน ซีเนอร์-ไดโอด จึงทำให้การรักษาระดับแรงดันมีการเปลี่ยนแปลงไปตามอุณหภูมิ โดยเมื่ออุณหภูมิสูงจะได้ค่าแรงดันที่เอาท์พุทมีค่าแรงดันมาก และเมื่ออุณหภูมิเริ่มต่ำลงค่าแรงดันที่เอาท์พุทก็จะลดลงด้วย การคัดเลือกตัววัดอุณหภูมิ เห็นว่าเบอร์ IC เบอร์ LM335 มีคุณสมบัติตามที่กล่าวมาซึ่ง LM335เป็นวงจรอินทิเกรทเซอร์คิท ซึ่งใช้ทำเป็นตัวตรวจจับอุณหภูมิสำหรับใช้ในย่านอุณหภูมิตั้งแต่ 0 องศา จนถึง +100 องศา มีลักษณะรูปร่างดังรูปที่ 5.1

plastic package



bottom view

Order Number LLM335Z or LM335AZ

See NS Paackage Number Z03A

รูปที่ 5.1 แสดงลักษณะของ Sensor

โดยพื้นฐานแล้ว LM335 ก็ทำงานเช่นเดียวกับซีเนอร์ไดโอดดังวงจร แรง-ดันพังทลายซึ่งหมายถึงแรงดันเอาท์พุทจากวงจรนี้จะแปรโดยตรงกับอุณหภูมิสมบูรณ์ โดยมีค่าเท่ากับ 10 mv ต่ออุณหภูมิที่เพิ่มขึ้น 1 องศา ในย่านอุณหภูมิที่ออกแบบมาใช้งาน ค่าของตัวต้านทานจะเป็นตัวกำหนดค่าของกระแสที่ไหลผ่านอุปกรณ์ตัวนี้ แต่เนื่องจากค่าไดนามิคอิม

พีแดนซ์ที่กระแส 1 ma จะมีค่าโดยปกติ 0.6 โอห์ม อุปกรณ์ตัวนี้จึงสามารถทำงานได้ในย่านกระแสตั้งแต่ 400 ไมโครแอมป์ โดยเสมือนว่าไม่มีการเปลี่ยนคุณสมบัติของตัวมันเลย และค่า reverse current or forward current สูงสุดโดยไม่ทำให้อุปกรณ์เสียหายควรเป็น 10 ma ถ้ากระแสสูงกว่านี้จะทำให้เกิดความเสียหายต่ออุปกรณ์ตัววัดได้ ที่อุณหภูมิ 25 องศาเซลเซียล และที่reverse 1 ma แรงดันเอาท์พุทจากวงจรจะมีค่าตามที่ออกแบบไว้เท่ากับ 2.98 โวลท์

ค่าเปรียบเทียบจุดอ้างอิงของมาตราฐาน

จุดศูนย์สมบูรณ์ = -273.15 องศาเซลเซียล มีค่า = 0 องศาเควิน
จุดเยือกแข็งของน้ำ = 0 องศาเซลเซียล มีค่า = 273.15 องศาเควิน

สูตรการเปลี่ยนองศาเซลเซียลเป็นองศาเควิน

$\ldots\ldots\ldots^{\circ}_{K} = \ldots\ldots\ldots\ldots^{\circ}_{C} + 273.15$

จากคุณสมบัติการเปลี่ยนแปลง 10 mv / $^{\circ}_{C}$

ถ้า 0 $^{\circ}_{C}$ มีค่า = 273.15 $^{\circ}_{K}$
เทียบค่าเป็นแรงดัน = 273 X 10 (10 : 10 mv/$^{\circ}_{C}$)
= 2730 mv
= 2.73 v

เทียบค่า 25 $^{\circ}_{C}$ เป็นแรงดัน = 25 X 10 (10 : 10 mv/$^{\circ}_{C}$)
= 250 mv
= 0.25 v

ทำการเปลี่ยนค่าองศาเควินเป็นองศาเซลเซียล

จะได้ 0 $^{\circ}_{C}$ = 273 $^{\circ}_{K}$
ถ้า 25 $^{\circ}_{C}$ = 2.73 + 0.25
ฉะนั้นที่ 25 $^{\circ}_{C}$ = 2.98 V

และจากการเปลี่ยนที่ 10 mv ต่อ 1 $^{\circ}_{C}$ จะได้ค่าแรงดันดังตาราง

°c	แรงดัน Vout (v)
18	2.91
19	2.92
20	2.93
21	2.94
22	2.95

ความเป็นเชิงเส้น จุดเด่นของตัววัดที่เป็น semiconductor LM 335 จะให้แรงดันเอาท์พุท ที่เป็นเชิงเส้นเมื่อเทียบกับอุณหภูมิที่เปลี่ยนแปลงในย่านที่กำหนดซึ่ง ไม่-เหมือนกับเอาท์พุทที่ได้จากตัวตารางจับอุณหภูมิส่วนใหญ่ซึ่งเอาท์พุทจะไม่เป็นเชิงเส้น ซึ่งถ้าเรานำเอาค่าแรงดันเอาท์พุทมาเขียนเป็นกราฟ ระหว่างค่าแรงดันกับอุณหภูมิตลอดย่านการทำงานของ LM335 เมื่อลากเส้นไปตัดแกนอุณหภูมิที่ 0 องศาสัมบูรณ์ ค่าแรงดันเอาท์พุทที่อ่านได้จากกราฟจะเป็น 0 v จากความเป็นเชิงเส้นที่กล่าวจะต้องขึ้นอยู่กับการปรับความถูกต้องของอุณหภูมิเพียงครั้งเดียวจะให้ความถูกต้องตลอดย่านอุณหภูมิที่ใช้งานนั้นก็เนื่องจากเอาท์พุทจะแปรผันโดยตรงอย่างเที่ยงตรงกับอุณหภูมิสัมบูรณ์ โดยเอาท์พุทจะลดลงเป็น 0 v ที่อุณหภูมิองศาสัมบูรณ์ ดังนั้นการปรับความลาดชันที่อุณหภูมิค่าหนึ่งให้ถูกต้อง จะทำให้เกิดความถูกต้องตลอดย่านอุณหภูมิ

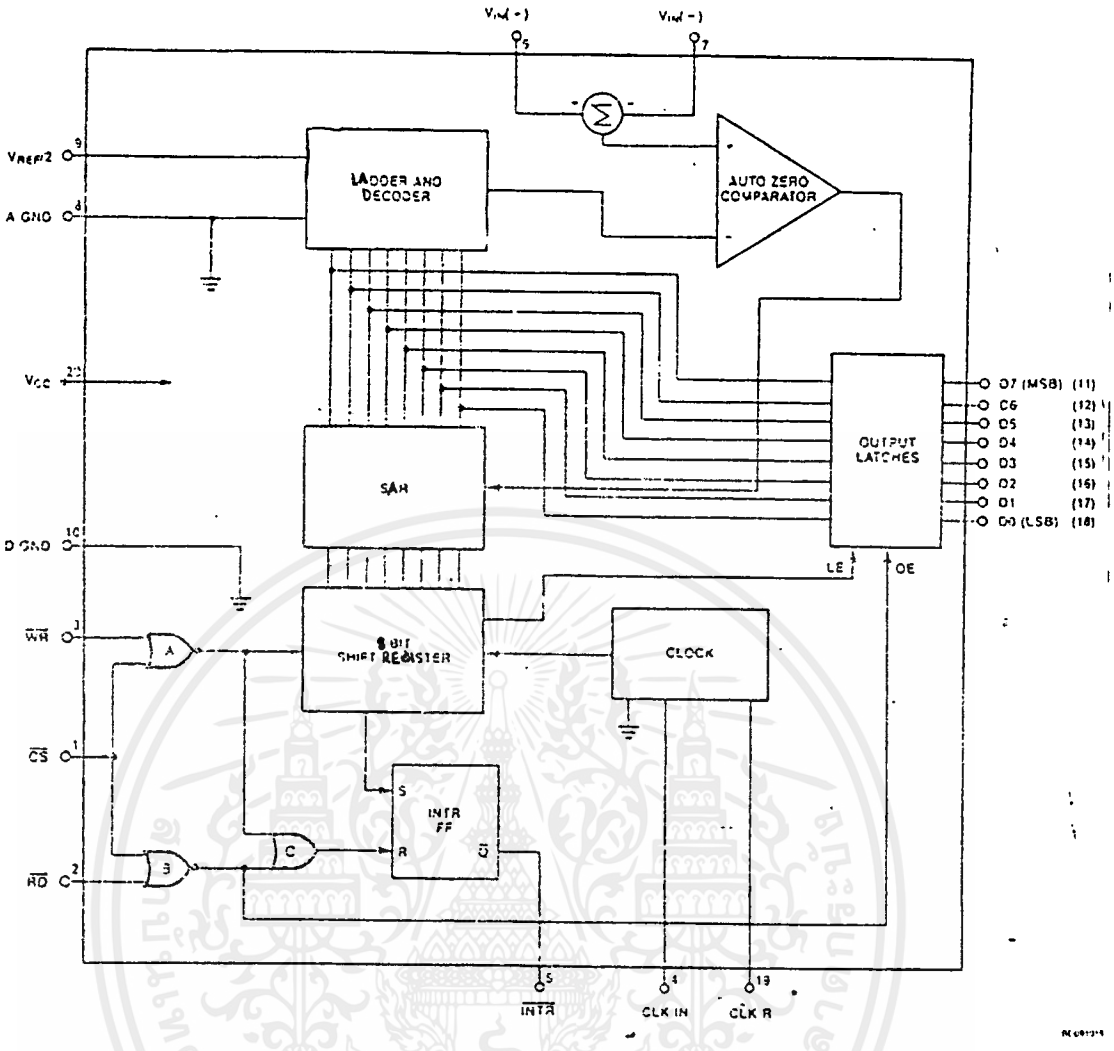
ความร้อนที่เกิดขึ้นในตัว เช่นเดียวกับตัวตรวจจับอุณหภูมิ ความร้อนใด ๆ ที่เกิดขึ้นจากกระแสที่ไหลผ่านอุปกรณ์ที่ตัวตรวจจับ จะมีผลต่อค่าอุณหภูมิของตัวมันตลอดจนค่า-แรงดันเอาท์พุทที่เกิดขึ้นสำหรับ LM335 นั้นควรจะให้ทำงานที่กระแสต่ำสุดซึ่งเพียงพอที่จะขับให้วงจรภายในไอซี ทำงานได้โดยกระแสประมาณ 400 ma จะเป็นกระแสต่ำสุดที่ไอซีจะทำงานได้ตามปกติ (ค่ากระแสนี้จาก spacb)

ตัวแปลงสัญญาณ

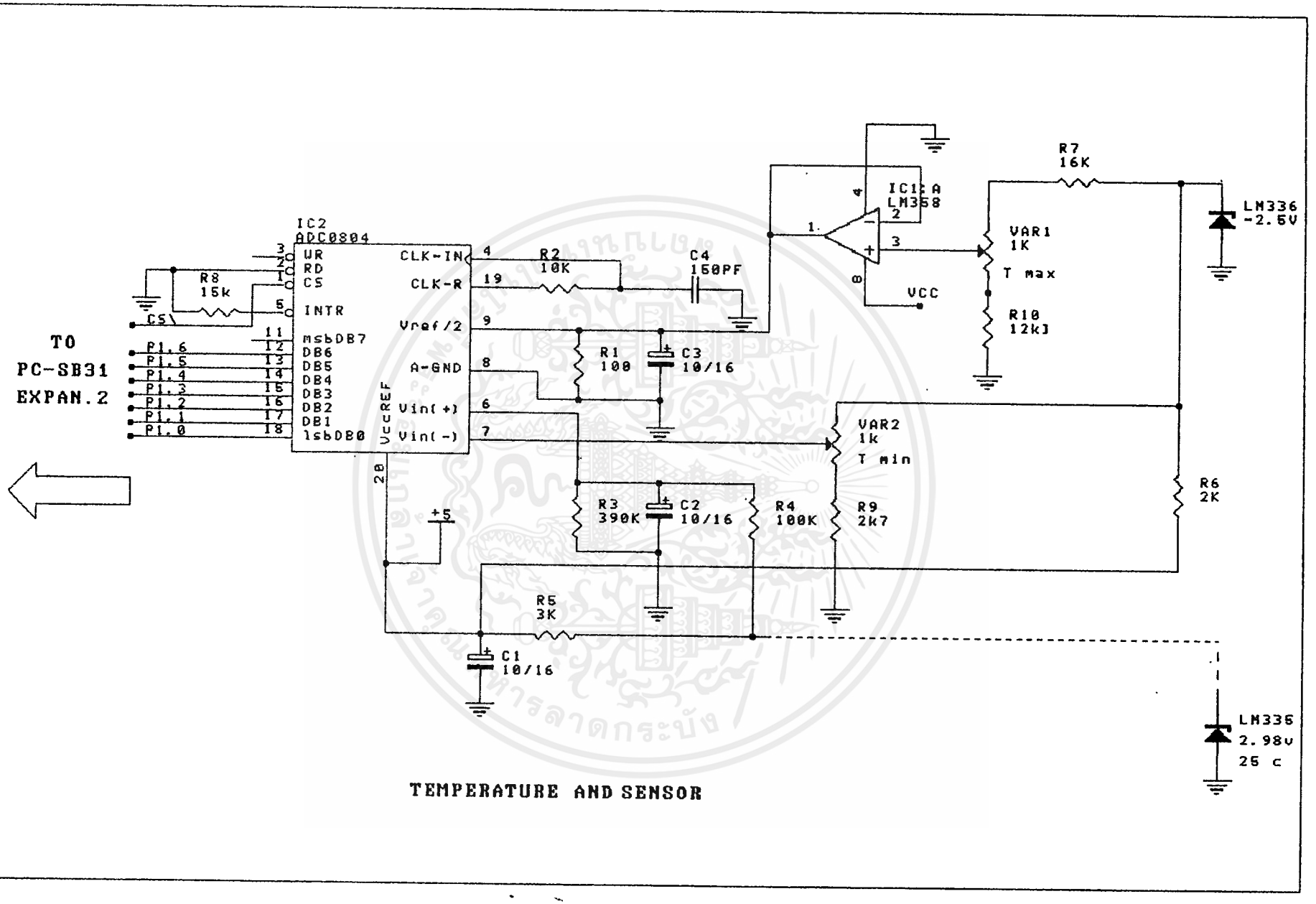
จากตัววัดอุณหภูมิซึ่งใช้ IC LM 335 ซึ่งให้เอาท์พุทเป็นแรงดันที่เปลี่ยนแปลงค่าตามอุณหภูมิที่เปลี่ยนไป และแรงดันที่ได้นี้เป็นสัญญาณแบบอะนาล็อคฉะนั้นจำเป็นอย่างยิ่งที่จะต้องทำการแปลงค่าแรงดันซึ่งเป็นอะนาล็อคให้เป็นแบบดิจิตอล เพื่อให้ค่าที่เป็นไบนารี่-เป็นค่าดาต้าส่งค่าทาง ดาต้าบัสให้กับซีพียู เพื่อทำการโปรเซสและให้เอาท์พุทไปควบคุมระบบเครื่องปรับอากาศในการแปลงสัญญาณอะนาล็อคซึ่งเป็นสัญญาณดิจิตอลนี้เราใช้ IC เบอร์ ADC 0804 จะเป็นตัวแปลงสัญญาณซึ่ง ADC 0804 จะมีหนึ่งอินพุทที่รับสัญญาณอะนาล็อคและเอาท์พุทเป็นดิจิตอล (เป็นค่าไบนารี่) ขนาด 8 บิต ซึ่งเหมาะสมกับวงจรที่ออกแบบไว้

คุณสมบัติ ADC 0804 เป็นตระกูล ซีมอส ขนาด 8 บิต ใช้วิธีการแปลงค่า A/Dแบบการประมาณค่า (succesteve approximation)ซึ่งการเปรียบเทียบภายในใช้ความต้านทานแบบ ladder และสามารถเปรียบเทียบใน auto-zero สามารถทำงาน-ร่วมกับ micro processerได้โดยมีอุปกรณ์ต่อร่วมภายนอกเพียงเล็กน้อยเอาท์พุทเป็นแบบ 3-state โดยต่อเข้ากับดาต้าบัสโดยตรง แรงดันอินพุทที่เป็นสัญญาณอะนาล็อคสามารถเพิ่มความต่างของสัญญาณได้ ทำการ offset ได้และสามารถปรับค่าอินพุทที่มีสัญญาณต่ำ ๆโดยให้เอาท์พุทเต็ม 8 บิต ได้โครงสร้างภายในดังรูปที่ 4.2

การทำงานของวงจรการนำไอซี ADC 0804 ต่อร่วมใช้งานกับตัววัดอุณหภูมิ LM335 ซึ่ง LM335 จะให้แรงดันที่เปลี่ยนไปตามอุณหภูมิที่เปลี่ยนไปจาการวัดให้ค่าแรงดันที่ขา v_{++} ADC 0804 ก่อนการใช้งานจำเป็นต้องทำการปรับค่าของการวัดเสียก่อน ซึ่งการปรับมีสองขั้นตอนด้วยกัน



รูปที่ 5.2 แสดงวงจรภายในของไอซี ADC 0804



TEMPERATURE AND SENSOR

รูปแสดงวงจรการต่อ ADC 0804 ร่วมกับตัว Sensor

ขั้นตอนที่หนึ่ง ปรับที่ variable registor tmax ซึ่งจะต่อกับขา vref ของ ADC 0804 (ขา 9) โดยผ่านบัฟเฟอร์ LM358 การปรับ tmax เพื่อกำหนดค่าว่าแรงดัน v_{i+} ที่รับเข้ามามีการเปลี่ยนแปลงเท่าไร ทำให้ค่าไบนารี่ที่เอาท์พุทเปลี่ยนแปลงกี่บิต โดยใช้สมการในการหาค่าที่ vref ดังนี้

Verf =(I/P ที่เปลี่ยนแปลง 1 บิต X บิต full scall)

- ฉะนั้นต้องการค่าที่ I/Pเปลี่ยนแปลง 10 mv ทำให้เอาท์พุทเปลี่ยนไป 1 บิต
- เอาท์พุทขนาด 8 บิต ที่full scale จะได้ 2^8 = 256 ค่า

จาก 00000000(B) ถึง 11111111(B) ได้

- Vref ที่ 5 โวลท์ (Vcc) จะได้

Vref = 10 mv X 256

= 2.56 v

Vref/2 = (10 mv X 256)/2

= 1.28 V

ฉะนั้นทำการปรับค่า tmax ให้ได้ 1.28 โวลท์

ADC 0804 จะทำการแปลงค่าเมื่อ V_{i+} รับแรงดันมีการเปลี่ยนแปลง 10 mv ที่เอาท์พุทจะเปลี่ยนไป 1 บิต

ขั้นตอนที่สอง ปรับที่ variable resistor tmin ซึ่งจะต่อเข้ากับขา V_{i-} ของ ADC 0804 (ขา 7)การปรับ tminเพื่อกำหนดค่าของเอาท์พุทที่เริ่มต้น(00000000) ว่าเป็นอุณหภูมิต่ำสุดที่เท่าไรของการวัด วิธีการหาค่าในการปรับ tmin จากคุณสมบัติของตัววัดอุณหภูมิจะให้แรงดัน 2.98 โวลท์จากรูปที่ 5.3 เขียนใหม่เสียซึ่งวงจรจะเป็นลักษณะ voltage divider ดังรูปที่ 5.4 ต้องทำการหาค่าแรงดันที่ V_{i+} โดยใช้สมการของ voltage divider

$V_i/V_T = R_3/R_T$

$V_i = (R_3/R_T)V_T$

$=(390K/490K)\ 2.98$

V_i ที่ 25 $^{\circ}_{C}$ จะได้ = 2.372 V

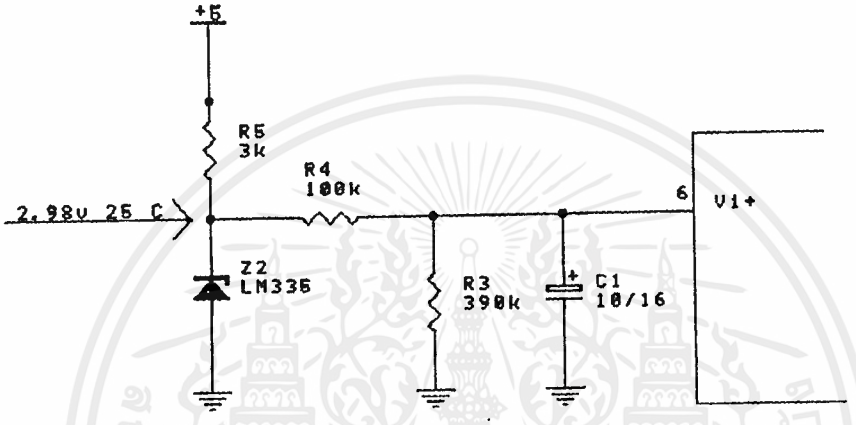
จาก V_T ที่ 0 $^{\circ}_{C}$ จะได้ = 2.73 V

$V_i = (R_3/R_T)V_T$

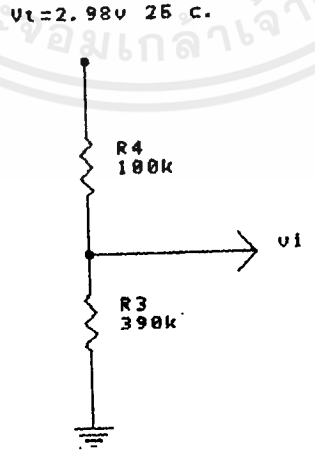
$= (390K/490K)\ 2.73$

V_i ที่ 0 $^{\circ}_{C}$ = 2.173 V (v_i คือ V_{i+})

เมื่อ V_{i+} ที่ 0 $^{\circ}_{c}$ = 2.17 โวลท์ จากนั้นจึงทำการปรับที่ tmin ให้ได้ 2.17 โวลท์ ที่เอาท์พุท (ทั้ง 8 บิต) จะเป็น 0 หมด และเมื่อที่ตัววัดให้แรงดันที่เปลี่ยนแปลงเป็นแบบ BCD จากเลขนับสำคัญต่ำสุด (DB0) ไปจนถึงเลขนับสำคัญสูงสุด (DB7) จะได้ค่าต่ำสุด คือ 00000000(B) จนถึง full scale จะเท่ากับ 11111111(B) แต่ในย่านของอุณหภูมิที่เลือกใช้งานตั้งแต่อุณหภูมิ 18 $^{\circ}_{c}$ จนถึง 22 $^{\circ}_{c}$ จะได้ค่าแรงดันที่ตัววัด ค่าแรงดันที่ V_{i+} (ขา6) ของADC 0804และค่าเอาท์พุทที่เป็นเลขดิจิตอลตามตาราง



รูปที่ 5.3 แสดงวงจรการใช้งานจริง



รูปที่ 5.4 แสดงวงจรการแปลงเป็น Voltage Divider ของรูปที่ 5.3

อุณหภูมิ($^{\circ}C$)	แรงดันที่ตัววัด (V)	แรงดันที่ V_{i+} (V)	เอาท์พุทดิจิตอล (HEX)
18	2.91	2.316	12
19	2.92	2.324	13
20	2.93	2.332	14
21	2.94	2.340	15
22	2.95	2.348	16

ค่าแรงดันV_{i+} ของแต่ละอุณหภูมิในตารางหาได้จากสมการของ voltage divider ข้างต้นโดยที่ค่าของ V_T จะเปลี่ยนไปตามแต่ละอุณหภูมิและค่า V_T นี้ได้เปรียบเทียบเป็นอุณหภูมิขององศาเซลเซียล โดยเปรียบเทียบค่าจาก $^{\circ}C$ เป็น $^{\circ}C$ จากสมการเปรียบเทียบจุดอ้างอิงมาตรฐาน เมื่อเราได้ค่าเอาท์พุทจาก ADC 0804 แล้วที่เอาท์พุทของ ADC 0804 จะต่ออยู่กับ expan.2 ของ microcontroller ชุดบอร์ด PC-SB31 ซึ่งต่อโดยตรงมาจากพอร์ทของซีพียู ได้กำหนดให้อินพุทพอร์ทดาต้าของ ADC 0804 ที่ซีพียูรับเข้ามาซีพียูจะทำการโปรเซสและให้เอาท์พุทตามที่โปรแกรมกำหนด โดยให้สัญญาณควบคุมส่งผ่านทาง 8255 ไปควบคุมรีเลย์เพื่อเลือกให้ระบบใดระบบหนึ่งทำงานหรือทั้งสองระบบทำงานร่วมกัน โดยกำหนดให้ เครื่องปรับอากาศทั้งสองชุดทำงานร่วมกันเมื่ออุณหภูมิของน้ำเย็นสูงกว่า 22 องศาเซลเซียล และตัดให้ทำงานเพียงระบบเดียวเมื่ออุณหภูมิต่ำกว่า 18 องศาเซลเซียล การใช้อุณหภูมิตัด ต่อให้ทำงานระบบเดียวหรือทั้งสองระบบ จะทำได้เมื่อสวิทช์ควบคุมอยู่ที่โหมด AUTO

เห็นได้ว่าย่านของอุณหภูมิที่เราใช้งานนั้นจะอยู่ในช่วง 18-22 องศาเซลเซียส และค่าสูงสุดของอุณหภูมิที่ใช้คือ 22 องศาเซลเซียส ค่าเอาท์พุทดิจิตอลตามตารางคือ 16H ซึ่งค่าสูงสุดของ ADC 0804 นี้ให้ได้ถึง FFH (256 ค่า) ฉะนั้นที่บิตนัยสำคัญสูงสุดไม่มีความจำเป็นต้องใช้ เพราะไม่มีการเปลี่ยนแปลงค่าบิต เราจึงเลือกเอาพอร์ท P1.7 ของ CPU มาใช้งานได้โดยทำให้เป็นอินพุท รับค่าจาก TEMP.CHECK SW. ที่ต่อกับ Vcc หรือต่อลงกราวน์ซึ่งมีตัวต้านทานต่ออนุกรมอยู่ด้วย TEMP.CHECK SW. นี้ใช้ตรวจสอบค่าอุณหภูมิในขณะนั้น ๆ โดยแสดงผลออกทาง LCD

บทสรุป

ผลของการทำงานในการทดลองนี้ ได้บรรลุวัตถุประสงค์เป็นที่น่าพอใจในระดับหนึ่ง คือได้เรียนรู้ โครงสร้างและการใช้คำสั่งของไมโครคอนโทรลเลอร์เบอร์ 8031 ซึ่งเป็นชิพ ในตระกูล MCS-51 ของบริษัท INTEL มาใช้ในงานควบคุม พร้อมกันนี้ ทำให้เข้าใจในหลักการนำมาประยุกต์ เพื่อใช้ต่อร่วมกับอุปกรณ์ภายนอก การพัฒนาโครงงานนี้ให้สมบูรณ์ขึ้น สามารถทำได้ โดยการเปลี่ยนแปลงแก้ไขทางโปรแกรม (SOFTWARE) และเพิ่มเติมทาง HARDWARE โดยทางด้านโปรแกรมได้บันทึกลงไว้ใน EPROM ซึ่งสามารถแก้ไขได้ และสามารถนำไปใช้ในการควบคุมในลักษณะต่าง ๆ ได้อย่างกว้างขวาง

กิตติกรรมประกาศ

โครงงานและปริญญานิพนธ์นี้สำเร็จได้ด้วยดี ทั้งนี้ก็ด้วยความช่วยเหลือจากบุคคลหลาย ๆ ฝ่ายด้วยกันซึ่งต้องขอขอบคุณ ผศ. พิพัฒน์ เลาหสงคราม ที่ช่วยชี้แนะในด้านต่าง ๆ และให้คำปรึกษาทั้งทางด้านซอร์ฟแวร์และฮาร์ดแวร์

ขอขอบคุณทางภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม ที่ให้การสนับสนุนโครงงาน ในด้านเครื่องมือและอุปกรณ์ในการทดลองซึ่งมีส่วนทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี

ภาคผนวก

โปรแกรมควบคุมการทำงาน

```
            ORG     0000H           ;KRUK TE1 PC-SB31

PORTA       EQU     0E0E0H          ;PA0-PA3=IN SYS.1 , PA4-PA7=IN SYS.2
PORTB       EQU     0E0E1H          ;DRIVER PB0-PB3=SYS.1 , PB4-PB5=SYS.2
PORTC       EQU     0E0E2H          ;PC0=STA1,PC1=STO1,PC2=STA2,PC3=STO2
CONTP       EQU     0E0E3H          ;PORT CONTROL
COMMAND     EQU     0E0C0H          ;Read-Write Register
READBUSY    EQU     0E0C1H          ;Read BF (Busy Flag) and address
WRITEDATA   EQU     0E0C2H          ;Write character
READDATA    EQU     0E0E3H          ;Read Data from DD ram

; ******** MAIN PROGRAM ********

            LJMP    TZR             ; set PORT


; ******** INITERRUPT0 *********

            LJMP    INTERR          ;INTERRUPT 0

; ******** SET PORT 8255 *******

TZR:        LCALL   DELAYS1         ;DELAY 1 Sec.

            MOV     70H,#0
            MOV     71H,#0
            MOV     R0,#5

STPORT:     MOV     DPTR,#CONTP     ;set PORT CONTROL OF 8255
            MOV     A,#10010001B    ;PA&PC lower=IN,PB&PC upper=OUT
            MOVX    @DPTR,A

            DJNZ    R0,STPORT

; ****** ACKNOW INTERRUPT *****

ACKINT:     MOV     88H,#40H        ;set TCON
            MOV     0A8H,#0F5H      ;set IE
            MOV     0B8H,#15H       ;set IP

; **** TEST SYSTEM, READY *****

STLCD:      LCALL   INITLCD         ;set LCD
            MOV     A,#1
            MOV     R4,#0           ;Line 1 "SYSTEM NO.1 READY"
            LCALL   WRLINE
            MOV     A,#2
            MOV     R4,#1           ;Line 2 "SYSTEM NO.2 READY"
            LCALL   WRLINE

SCSW:       MOV     A,71H
            CJNE    A,#1,NZM
            LJMP    WPT
NZM:        MOV     DPTR,#PORTC
            MOVX    A,@DPTR
            JNB     ACC.0,NECT20
            JNB     ACC.1,NECT21
            MOV     A,P1
            JNB     ACC.7,NECT24
            MOV     A,P1
            ANL     A,#7FH
            CLR     C
            CJNE    A,#7FH,AUTOM
```

```
                LJMP    SCSW
AUTOM:          CJNE    A,#29H,MR1
AUTOM2:         MOV     DPTR,#PORTC
                MOVX    A,@DPTR
                ORL     A,#00100000B
                MOV     DPTR,#PORTC
                MOVX    @DPTR,A
                LJMP    ST2
MR1:            JNC     AUTOM2
                LJMP    SCSW
NECT20:         LJMP    CODE0
NECT21:         LJMP    CODE1
NECT22:         LJMP    CODE2
NECT23:         LJMP    CODE3
NECT24:         LJMP    CODE4

ST2:            MOV     DPTR,#PORTC
                MOVX    A,@DPTR
                JNB     ACC.2,ST22
                LJMP    SCSW

ST22:           LJMP    SYSTEM2

ST3:            MOV     DPTR,#PORTC
                MOVX    A,@DPTR
                JNB     ACC.3,SCSW
                LJMP    SCSW

WPT:            MOV     DPTR,#PORTC
                MOVX    A,@DPTR
                JNB     ACC.0,CODE0
                JNB     ACC.1,CODE1
                JNB     ACC.3,NECT11
                MOV     A,P1
                JNB     ACC.7,NECT24
                MOV     A,P1
                ANL     A,#7FH
                CJNE    A,#7FH,MRN18
                LJMP    WPT
MRN18:          CJNE    A,#25H,MN18RN
ASLS:           MOV     DPTR,#PORTC
                MOVX    A,@DPTR
                ANL     A,#11011111B
                MOVX    @DPTR,A
                LJMP    WPT
MN18RN:         JC      ASLS
                LJMP    SCSW

NECT9:          LJMP    SCSW
NECT10:         LJMP    CODE2
NECT11:         LJMP    CODE3
NECT12:         LJMP    CODE4
NECT13:         LJMP    CODE1


CODE0:          MOV     DPTR,#PORTC
                MOVX    A,@DPTR
                JNB     ACC.1,CODE1
                LJMP    SYSTEM1

CODE1:          MOV     A,71H
                CJNE    A,#1,HIT1
                MOV     DPTR,#PORTB
```

```
                MOV     A,#0F0H
                MOVX    @DPTR,A
                SJMP    KILL1
HIT1:           MOV     DPTR,#PORTB
                MOV     A,#00H
                MOVX    @DPTR,A
KILL1:          MOV     70H,#0

                MOV     A,71H
                CJNE    A,#1,QU1
                LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#0           ;Line 1 "SYSTEM NO.1 READY"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#11          ;Line 2 "SYSTEM NO.2 RUNNING"
                LCALL   WRLINE
                LJMP    SCSW
QU1:            LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#0           ;Line 1 "SYSTEM NO.1 READY"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#1           ;Line 2 "SYSTEM NO.2 READY"
                LCALL   WRLINE
                LJMP    SCSW

CODE2:          MOV     DPTR,#PORTC
                MOVX    A,@DPTR
                JNB     ACC.3,CODE3
                LJMP    SYSTEM2


CODE3:          MOV     A,70H
                CJNE    A,#1,HIT3
                MOV     DPTR,#PORTB
                MOV     A,#0FH
                MOVX    @DPTR,A
                SJMP    KILL3
HIT3:           MOV     DPTR,#PORTB
                MOV     A,#00H
                MOVX    @DPTR,A
KILL3:          MOV     71H,#0

                MOV     A,70H
                CJNE    A,#1,QU3
                LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#10          ;Line 1 "SYSTEM NO.1 RUNNING"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#1           ;Line 2 "SYSTEM NO.2 READY"
                LCALL   WRLINE
                MOV     71H,#0
                LJMP    SCSW
QU3:            LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#0           ;Line 1 "SYSTEM NO.1 READY"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#1           ;Line 2 "SYSTEM NO.2 READY"
                LCALL   WRLINE
                MOV     71H,#0
```

```
                LJMP    SCSW

CODE4:          LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#34
                LCALL   WRLINE

TEMOC:          LCALL   TEMPCH
                MOV     A,#2
                LCALL   WRLITEM
                MOV     A,P1
                JNB     ACC.7,TEMOC
                LJMP    GGV

TEMPCH:         LCALL   DEB3
                MOV     A,P1
                ANL     A,#7FH
                CJNE    A,#7FH,NBV
                LJMP    TEMPCH
NBV:            CLR     C
                SUBB    A,#13H

                MOV     35H,#0FFH
D100LP:         INC     35H
                CLR     C
                SUBB    A,#100
                JNC     D100LP
                ADD     A,#100

                MOV     36H,#0FFH
D10LP:          INC     36H
                CLR     C
                SUBB    A,#10
                JNC     D10LP
                ADD     A,#10
                MOV     R5,A
                MOV     A,36H
                SWAP    A
                ORL     A,R5
                MOV     36H,A

                MOV     DPTR,#COUNTEM
                MOV     A,35H
                MOVX    @DPTR,A
                INC     DPTR
                MOV     A,36H
                MOVX    @DPTR,A

CONVTEM:        MOV     DPTR,#COUNTEM
                MOV     R1,DPH
                MOV     R2,DPL
                MOV     DPTR,#DISBTEM
                MOV     R3,DPH
                MOV     R4,DPL

                MOV     R0,#02

DONETEM:        MOV     DPL,R2
                MOV     DPH,R1
                MOVX    A,@DPTR
                SWAP    A
                LCALL   ADJTEM
```

```
                MOV     DPL,R4
                MOV     DPH,R3
                INC     DPTR
                MOV     R3,DPH
                MOV     R4,DPL
                MOV     DPL,R2
                MOV     DPH,R1
                MOVX    A,@DPTR
                LCALL   ADJTEM

                MOV     DPL,R4
                MOV     DPH,R3
                INC     DPTR
                MOV     R3,DPH
                MOV     R4,DPL
                MOV     DPL,R2
                MOV     DPH,R1
                INC     DPTR
                MOV     R1,DPH
                MOV     R2,DPL
                DJNZ    R0,DONETEM

ONGSAC:         MOV     DPTR,#DISBTEM+4
                MOV     A,#00H
                MOVX    @DPTR,A
                INC     DPTR
                MOV     A,#43H
                MOVX    @DPTR,A

                RET

ADJTEM:         ANL     A,#0FH
                MOV     DPTR,#TABOTEM
                CLR     C
                ADD     A,DPL
                MOV     DPL,A
                JC      RELTEM
                SJMP    LDATEM
RELTEM:         INC     DPH
LDATEM:         MOVX    A,@DPTR
                MOV     DPL,R4
                MOV     DPH,R3
                MOVX    @DPTR,A

                RET


COUNTEM:        DS      2
DISBTEM:        DS      6


TABOTEM:        DB      30H
                DB      31H,32H,33H
                DB      34H,35H,36H
                DB      37H,38H,39H


WRLITEM:        CJNE    A,#2,IUY
                SJMP    WRTEMP

IUY:            NOP
                RET
```

```
WRTEMP:   MOV     A,#0D3H
          MOV     DPTR,#COMMAND
          MOVX    @DPTR,A
          LCALL   WAITBF
          MOV     DPTR,#DISBTEM+1

WRLI:     MOV     R0,#5
WRLII:    MOVX    A,@DPTR
          MOV     R3,A
          PUSH    DPH
          PUSH    DPL
          LCALL   WRBYTE
          POP     DPL
          POP     DPH
          INC     DPTR
          DJNZ    R0,WRLII

          RET


GGV:      MOV     A,70H
          CJNE    A,#1,GGV3
          MOV     A,71H
          CJNE    A,#1,GGV2
          LCALL   INITLCD
          MOV     A,#1
          MOV     R4,#10          ;Line 1 "SYSTEM NO.1 RUNNING"
          LCALL   WRLINE
          MOV     A,#2
          MOV     R4,#11          ;Line 2 "SYSTEM NO.2 RUNNING"
          LCALL   WRLINE
          LJMP    SCSW

GGV2:     LCALL   INITLCD
          MOV     A,#1
          MOV     R4,#10          ;Line 1 "SYSTEM NO.1 RUNNING"
          LCALL   WRLINE
          MOV     A,#2
          MOV     R4,#1           ;Line 2 "SYSTEM NO.2 READY"
          LCALL   WRLINE
          LJMP    SCSW

GGV3:     MOV     A,71H
          CJNE    A,#1,GGV4
          LCALL   INITLCD
          MOV     A,#1
          MOV     R4,#0           ;Line 1 "SYSTEM NO.1 READY"
          LCALL   WRLINE
          MOV     A,#2
          MOV     R4,#11          ;Line 2 "SYSTEM NO.2 RUNNING"
          LCALL   WRLINE
          LJMP    SCSW

GGV4:     LCALL   INITLCD
          MOV     A,#1
          MOV     R4,#0           ;Line 1 "SYSTEM NO.1 READY"
          LCALL   WRLINE
          MOV     A,#2
          MOV     R4,#1           ;Line 2 "SYSTEM NO.2 READY"
          LCALL   WRLINE
          LJMP    SCSW
```

```
; ********** 00:00 ************

CLCOT:      MOV     A,#99H
            MOV     DPTR,#COUNT
            MOVX    @DPTR,A
            MOV     DPTR,#COUNT+1
            MOV     A,#59H
            MOVX    @DPTR,A
            MOV     DPTR,#COUNT+2
            MOVX    @DPTR,A
            LCALL   NUBTOE
            SJMP    OURMAI

NUBTOE:     LCALL   UPTIME
            LCALL   CONVT
            MOV     A,#2
            LCALL   WRLINT
            LCALL   DELAYS1

            RET

OURMAI:     MOV     A,32H
            CJNE    A,#00H,MMM
            SJMP    WWW
MMM:        DJNZ    32H,TOUTOE
WWW:        MOV     A,31H
            CJNE    A,#00H,YYY
            SJMP    EEE
YYY:        LCALL   NUBTOE
            DEC     31H
            MOV     32H,#0FFH
            SJMP    TOUTOE
EEE:        MOV     A,30H
            CJNE    A,#00H,KKK
            SJMP    XXX
KKK:        LCALL   NUBTOE
            DEC     30H
            MOV     31H,#0FFH
            MOV     30H,#0FFH
            SJMP    TOUTOE

TOUTOE:     LCALL   NUBTOE
            SJMP    OURMAI

XXX:        NOP

            RET

;********* FINE XX:XX *********

; ***** SYSTEM NO.1 START ******

SYSTEM1:    MOV     70H,#0

WRW5S1:     LCALL   INITLCD         ;SET LCD
            MOV     A,#1
            MOV     R4,#30
            LCALL   WRLINE          ;P. W. 5 S.
            LCALL   DELAYS1

00I051:     MOV     30H,#00         ;00.00.05
            MOV     31H,#00
            MOV     32H,#06
```

```
            LCALL   CLCOT

STM1:       LCALL   INITLCD
            MOV     A,#1
            MOV     R4,#2           ;Line 1 "MOTOR NO.1 RUNNING (S.1)"
            LCALL   WRLINE          ;display "MOTOR NO.1 RUNNING (S.1)"
            MOV     DPTR,#PORTB
            MOVX    A,@DPTR
            ORL     A,#00000001B
            MOVX    @DPTR,A         ;MOTOR NO.1 RUN
            LCALL   DELAYS3         ;DELAY 3 S.
            LCALL   WRW10S
            LJMP    STM2

WRW10S:     LCALL   INITLCD
            MOV     A,#1
            MOV     R4,#31          ;Line 2 "WAIT FOR 10 Sec."
            LCALL   WRLINE
            LCALL   DELAYS1

OOI10:      MOV     30H,#00
            MOV     31H,#00
            MOV     32H,#11
            LCALL   CLCOT


            RET

STM2:       LCALL   INITLCD
            MOV     A,#1
            MOV     R4,#3           ;Line 1 "MOTOR NO.2 RUNNING (S.1)"
            LCALL   WRLINE          ;display "MOTOR NO.2 RUNNING (S.1)"
            MOV     DPTR,#PORTB
            MOVX    A,@DPTR
            ORL     A,#00000011B
            MOVX    @DPTR,A         ;MOTOR NO.2 RUN

            LCALL   DELAYS3         ;DELAY 3 S.
            LCALL   WRW10S

STM3:       LCALL   INITLCD
            MOV     A,#1
            MOV     R4,#4           ;Line 1 "MOTOR NO.3 RUNNING (S.1)"
            LCALL   WRLINE          ;display "MOTOR NO.3 RUNNING (S.1)"
            MOV     DPTR,#PORTB
            MOVX    A,@DPTR
            ORL     A,#00000111B
            MOVX    @DPTR,A         ;MOTOR NO.3 RUN

            LCALL   DELAYS3         ;DELAY 3 S.
            LCALL   WRW3MN

            SJMP    STM4

WRW3MN:     LCALL   INITLCD
            MOV     A,#1
            MOV     R4,#32          ;Line 2 "WAIT FOR 3 Min."
            LCALL   WRLINE
            LCALL   DELAYS1         ;DELAY 1 S.

03I00:      MOV     30H,#00
            MOV     31H,#00
```

```
                MOV     32H,#181
                LCALL   CLCOT

                RET

STM4:           LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#5           ;Line 1 "MOTOR NO.4 RUNNING (S.1)"
                LCALL   WRLINE          ;display "MOTOR NO.4 RUNNING (S.1)"
                MOV     DPTR,#PORTB
                MOVX    A,@DPTR
                ORL     A,#00001111B
                MOVX    @DPTR,A         ;MOTOR NO.4 RUN

                LCALL   DELAYS3         ;DELAY 3 S.

SENSO:          MOV     70H,#1

FSTEM:          MOV     A,71H
                CJNE    A,#1,SHOW1
                SJMP    SHOW2


SHOW1:          LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#10          ;Line 1 "SYSTEM NO.1 RUNNING"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#1           ;Line 2 "SYSTEM NO.2 READY"
                LCALL   WRLINE          ;display "SYSTEM NO.1 RUNNING
                                        ;         SYSTEM NO.2 READY"
                                        ;SYSTEM NO.1 IS RUNNING**********
                LJMP    SCSW

SHOW2:          LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#10          ;Line 1 "SYSTEM NO.1 RUNNING"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#11          ;Line 2 "SYSTEM NO.2 RUNNING"
                LCALL   WRLINE          ;display "SYSTEM NO.1 RUNNING
                                        ;         SYSTEM NO.2 RUNNING"

                LJMP    SCSW

; ***** SYSTEM NO.2 START ******

SYSTEM2:        MOV     71H,#0

WRW5S2:         LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#30          ;W. P. 5 S.
                LCALL   WRLINE
                LCALL   DELAYS1

OOI052:         MOV     30H,#00         ;00.00.05
                MOV     31H,#00
                MOV     32H,#06
                LCALL   CLCOT

STAMT5:         LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#6           ;Line 1 "MOTOR NO.5 RUNNING (S.2)"
```

```
        LCALL   WRLINE          ;display "MOTOR NO.5 RUNNING (S.2)"
        MOV     DPTR,#PORTB
        MOVX    A,@DPTR
        ORL     A,#00010000B
        MOVX    @DPTR,A         ;MOTOR NO.5 RUN
        LCALL   DELAYS3         ;DELAY 3 S.
        LCALL   WRW10S
        LJMP    STAMT6

STAMT6: LCALL   INITLCD
        MOV     A,#1
        MOV     R4,#7           ;Line 1 "MOTOR NO.6 RUNNING (S.2)"
        LCALL   WRLINE          ;display "MOTOR NO.6 RUNNING (S.2)"
        MOV     DPTR,#PORTB
        MOVX    A,@DPTR
        ORL     A,#00110000B
        MOVX    @DPTR,A         ;MOTOR NO.6 RUN
        LCALL   DELAYS3         ;DELAY 3 S.
        LCALL   WRW10S
        LJMP    STAMT7

STAMT7: LCALL   INITLCD
        MOV     A,#1
        MOV     R4,#8           ;Line 1 "MOTOR NO.7 RUNNING (S.2)"
        LCALL   WRLINE          ;display "MOTOR NO.7 RUNNING (S.2)"
        MOV     DPTR,#PORTB
        MOVX    A,@DPTR
        ORL     A,#01110000B
        MOVX    @DPTR,A         ;MOTOR NO.7 RUN
        LCALL   DELAYS3         ;DELAY 3 S.
        LCALL   WRW3MN
        LJMP    STAMT8

STAMT8: LCALL   INITLCD
        MOV     A,#1
        MOV     R4,#9           ;Line 1 "MOTOR NO.8 RUNNING (S.2)"
        LCALL   WRLINE          ;display "MOTOR NO.8 RUNNING (S.2)"
        MOV     DPTR,#PORTB
        MOVX    A,@DPTR
        ORL     A,#11110000B
        MOVX    @DPTR,A         ;MOTOR NO.8 RUN

        LCALL   DELAYS3         ;DELAY 3 S.


SENSO2: MOV     71H,#1

FSYSTS2: MOV    A,70H
        CJNE    A,#1,SHOW3
        SJMP    SHOW4

SHOW3:  LCALL   INITLCD
        MOV     A,#1
        MOV     R4,#0           ;Line 1 "SYSTEM NO.1 READY"
        LCALL   WRLINE
        MOV     A,#2
        MOV     R4,#11          ;Line 2 "SYSTEM NO.2 RUNNING"
        LCALL   WRLINE          ;display "SYSTEM NO.1 READY
                                ;         SYSTEM NO.2 RUNNING"
        LJMP    SCSW

SHOW4:  LCALL   INITLCD
```

```
            MOV     A,#1
            MOV     R4,#10          ;Line 1 "SYSTEM NO.1 RUNNING"
            LCALL   WRLINE
            MOV     A,#2
            MOV     R4,#11          ;Line 2 "SYSTEM NO.2 RUNNIG"
            LCALL   WRLINE          ;display "SYSTEM NO.1 RUNNING
                                    ;          SYSTEM NO.2 RUNNING"

            LJMP    SCSW

; ***********INTERRUPT**********

INTERR:     MOV     DPTR,#PORTC
            MOVX    A,@DPTR
            ORL     A,#00010000B            ;ALARM (((((((((((((
            MOVX    @DPTR,A

;****** CHECK PA_BIT ? = 0 *****

READINA:    MOV     DPTR,#PORTA     ;read signal form PORTA
            MOVX    A,@DPTR

BIT_0:      JNB     ACC.0,G0        ;PA0=0 ?
            SJMP    BIT_1           ;no
G0:         LCALL   DEB             ;yes
            JNB     ACC.0,NECT0

BIT_1:      JNB     ACC.1,G1        ;PA1=0 ?
            SJMP    BIT_2           ;no
G1:         LCALL   DEB             ;yes
            JNB     ACC.1,NECT1

BIT_2:      JNB     ACC.2,G2        ;PA2=0 ?
            SJMP    BIT_3           ;no
G2:         LCALL   DEB             ;yes
            JNB     ACC.2,NECT2

BIT_3:      JNB     ACC.3,G3        ;PA3=0 ?
            SJMP    BIT_4           ;no
G3:         LCALL   DEB             ;yes
            JNB     ACC.3,NECT3

BIT_4:      JNB     ACC.4,G4        ;PA4=0 ?
            SJMP    BIT_5           ;no
G4:         LCALL   DEB             ;yes
            JNB     ACC.4,NECT4

BIT_5:      JNB     ACC.5,G5        ;PA5=0 ?
            SJMP    BIT_6           ;no
G5:         LCALL   DEB             ;yes
            JNB     ACC.5,NECT5

BIT_6:      JNB     ACC.6,G6        ;PA6=0 ?
            SJMP    BIT_7           ;no
G6:         LCALL   DEB             ;yes
            JNB     ACC.6,NECT6

BIT_7:      JNB     ACC.7,G7        ;PA7=0 ?
            LJMP    READINA         ;no
G7:         LCALL   DEB             ;yes
            JNB     ACC.7,NECT7
            LJMP    READINA         ;around
```

```
;********** CONNECT 0-7 ********

NECT0:      LJMP    ZERO
NECT1:      LJMP    ONE
NECT2:      LJMP    TWO
NECT3:      LJMP    THREE
NECT4:      LJMP    FOUR
NECT5:      LJMP    FIVE
NECT6:      LJMP    SIX
NECT7:      LJMP    SEVEN


;******* WHEN PA_BIT = 0 *******

ZERO:       LCALL   CUT1            ;OFF SYSTEM  1

            LCALL   INITLCD
            MOV     A,#1
            MOV     R4,#12          ;Line 1 "SYS N.1 EROR"
            LCALL   WRLINE
            MOV     A,#2
            MOV     R4,#14          ;Line 2 "M 1 OVER LOAD"
            LCALL   WRLINE

READ0:      MOV     DPTR,#CONTP
            MOV     A,#91H
            MOVX    @DPTR,A
            MOV     DPTR,#PORTA     ;yes
            MOVX    A,@DPTR
            JNB     ACC.0,READ0     ;PA0=0 ?

            LCALL   INITLCD         ;no
            MOV     A,#1
            MOV     R4,#0           ;Line 1 "SYSTEM 1 READY"
            LCALL   WRLINE
            MOV     A,#2
            MOV     R4,#22          ;Line 2 "M.1 READY"
            LCALL   WRLINE
            MOV     R7,#1
            LCALL   DELAYM
            LCALL   DYREA1

ONE:        LCALL   CUT1            ;OFF SYSTEM 1

            LCALL   INITLCD
            MOV     A,#1
            MOV     R4,#12          ;Line 1 "SYS N.1 ERROR"
            LCALL   WRLINE
            MOV     A,#2
            MOV     R4,#15          ;Line 2 "M.2 OVER LOAD"
            LCALL   WRLINE

READ1:      MOV     DPTR,#CONTP
            MOV     A,#91H
            MOVX    @DPTR,A
            MOV     DPTR,#PORTA     ;yes
            MOVX    A,@DPTR
            JNB     ACC.1,READ1     ;PA1=0 ?

            LCALL   INITLCD         ;no
            MOV     A,#1
            MOV     R4,#0           ;Line 1 "SYSTEM 1 READY"
            LCALL   WRLINE
            MOV     A,#2
            MOV     R4,#23          ;Line 2 "M.2 READY"
            LCALL   WRLINE
```

```
                MOV     R7,#1
                LCALL   DELAYM
                LCALL   DYREA1

TWO:            LCALL   CUT1            ;OFF SYSTEM N.1

                LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#12          ;Line 1 "SYS N.1 EROR"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#16          ;Line 2 "M.3 OVER LOAD"
                LCALL   WRLINE

READ2:          MOV     DPTR,#CONTP
                MOV     A,#91H
                MOVX    @DPTR,A
                MOV     DPTR,#PORTA     ;yes
                MOVX    A,@DPTR
                JNB     ACC.2,READ2     ;PA2=0 ?

                LCALL   INITLCD         ;no
                MOV     A,#1
                MOV     R4,#0           ;Line 1 "SYSTEM 1 READY"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#24          ;Line 2 "M.3 READY"
                LCALL   WRLINE
                MOV     R7,#1
                LCALL   DELAYM
                LCALL   DYREA1

THREE:          LCALL   CUT1            ;OFF SYSTEM NO.1

                LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#12          ;Line 1 "SYS N.1 EROR"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#17          ;Line 2 "M.4 OVER LOAD"
                LCALL   WRLINE

READ3:          MOV     DPTR,#CONTP
                MOV     A,#91H
                MOVX    @DPTR,A
                MOV     DPTR,#PORTA     ;yes
                MOVX    A,@DPTR
                JNB     ACC.3,READ3     ;PA3=0 ?

                LCALL   INITLCD         ;no
                MOV     A,#1
                MOV     R4,#0           ;Line 1 "SYSTEM 1 READY"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#25          ;Line 2 "M.4 READY"
                LCALL   WRLINE
                MOV     R7,#1
                LCALL   DELAYM
                LCALL   DYREA1

FOUR:           LCALL   CUT2            ;OFF SYSTEM NO.2

                LCALL   INITLCD
```

```
                MOV     A,#1
                MOV     R4,#13          ;Line "SYS N.2 ERROR"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#18          ;Line 2 "M.5 OVER LOAD"
                LCALL   WRLINE

READ4:          MOV     DPTR,#CONTP
                MOV     A,#91H
                MOVX    @DPTR,A
                MOV     DPTR,#PORTA     ;yes
                MOVX    A,@DPTR
                JNB     ACC.4,READ4     ;PA4=0 ?

                LCALL   INITLCD         ;no
                MOV     A,#1
                MOV     R4,#1           ;Line 1 "SYSTEM 2 READY"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#26          ;Line 2 "M.5 READY"
                LCALL   WRLINE
                MOV     R7,#1
                LCALL   DELAYM
                LCALL   DYREA2

FIVE:           LCALL   CUT2            ;OFF SYSTEM N.2

                LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#13          ;Line 1 "SYSTEM N.2 ERROR"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#19          ;Line 2 "M.6 OVER LOAD"
                LCALL   WRLINE

READ5:          MOV     DPTR,#CONTP
                MOV     A,#91H
                MOVX    @DPTR,A
                MOV     DPTR,#PORTA     ;yes
                MOVX    A,@DPTR
                JNB     ACC.5,READ5     ;PA5=0 ?

                LCALL   INITLCD         ;no
                MOV     A,#1
                MOV     R4,#1           ;Line 1 "SYSTEM 2 READY"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#27          ;Line 2 "M.6 READY"
                LCALL   WRLINE
                MOV     R7,#1
                LCALL   DELAYM
                LCALL   DYREA2

SIX:            LCALL   CUT2            ;OFF SYSTEM N.2

                LCALL   INITLCD
                MOV     A,#1
                MOV     R4,#13          ;Line 1 "SYSTEM NO.2 ERROR"
                LCALL   WRLINE
                MOV     A,#2
                MOV     R4,#20          ;Line 2 "M.7 OVER LOAD"
                LCALL   WRLINE
```

```
READ6:      MOV     DPTR,#CONTP
            MOV     A,#91H
            MOVX    @DPTR,A
            MOV     DPTR,#PORTA   ;yes
            MOVX    A,@DPTR
            JNB     ACC.6,READ6   ;PA6=0 ?

            LCALL   INITLCD       ;no
            MOV     A,#1
            MOV     R4,#1         ;Line 1 "SYSTEM 2 READY"
            LCALL   WRLINE
            MOV     A,#2
            MOV     R4,#28        ;Line 2 "M.7 READY"
            LCALL   WRLINE
            MOV     R7,#1
            LCALL   DELAYM
            LCALL   DYREA2

SEVEN:      LCALL   CUT2          ;OFF SYSTEM N.2

            LCALL   INITLCD
            MOV     A,#1
            MOV     R4,#13        ;Line 1 "SYSTEM N.2 ERROR"
            LCALL   WRLINE
            MOV     A,#2
            MOV     R4,#21        ;Line 2 "M.8 OVER LOAD"
            LCALL   WRLINE

READ7:      MOV     DPTR,#CONTP
            MOV     A,#91H
            MOVX    @DPTR,A
            MOV     DPTR,#PORTA   ;yes
            MOVX    A,@DPTR
            JNB     ACC.7,READ7   ;PA7=0 ?

            LCALL   INITLCD       ;no
            MOV     A,#1
            MOV     R4,#1         ;Line 1 "SYSTEM 2 READY"
            LCALL   WRLINE
            MOV     A,#2
            MOV     R4,#28        ;Line 2 "M.8 READY"
            LCALL   WRLINE
            MOV     R7,#1
            LCALL   DELAYM
            LJMP    DYREA2

CUT1:       MOV     A,71H
            CJNE    A,#1,ALONE1
            MOV     DPTR,#PORTB
            MOV     A,#0F0H
            MOVX    @DPTR,A
            MOV     70H,#0
            SJMP    TUC1
ALONE1:     MOV     DPTR,#PORTB
            MOV     A,#00H
            MOVX    @DPTR,A
            MOV     70H,#0

TUC1:       RET

CUT2:       MOV     A,70H
            CJNE    A,#1,ALONE2
```

```
            MOV     DPTR,#PORTB
            MOV     A,#0FH
            MOVX    @DPTR,A
            MOV     71H,#0
            SJMP    TUC2
ALONE2:     MOV     DPTR,#PORTB
            MOV     A,#00H
            MOVX    @DPTR,A
            MOV     71H,#0

TUC2:       RET

DYREA1:     MOV     88H,#40H
            MOV     0A8H,#0F5H
            MOV     0B8H,#15H
            MOV     A,71H
            CJNE    A,#1,NECT8
            MOV     DPTR,#PORTB
            MOV     A,#0F0H
            MOVX    @DPTR,A
            LJMP    SHOW3

DYREA2:     MOV     88H,#40H
            MOV     0A8H,#0F5H
            MOV     0B8H,#15H
            MOV     A,70H
            CJNE    A,#1,NECT8
            MOV     DPTR,#PORTB
            MOV     A,#0FH
            MOVX    @DPTR,A
            LJMP    SHOW1
NECT8:      LJMP    STLCD

;********** INITIAL LCD ********

INITLCD:    MOV     DPTR,#COMMAND
            MOV     A,#00111000B  ;function set  38H
                                  ;DL=1 8 bit,N=1 1/16 duty,F=0 5*7
            MOVX    @DPTR,A
            LCALL   WAITBF        ;enable signal pluse

            MOV     A,#00001111B  ;display on/off control
                                  ;D=1 off,C=1 cursor on,B=1 blink
            MOVX    @DPTR,A
            LCALL   WAITBF

            MOV     A,#00000110B  ;entry mode set
                                  ;I/D=1  increment,S=0 right
            MOVX    @DPTR,A
            LCALL   WAITBF

CLEARL:     MOV     A,#00000001B  ;clear all display
            MOVX    @DPTR,A
            LCALL   WAITBF


            RET

; * by mean of check busy flag *
WAITBF:     PUSH    DPH
            PUSH    DPL
```

```
            MOV     DPTR,#READBUSY
RDY1:       MOVX    A,@DPTR
            JB      ACC.7,RDY1      ;Busy Flag

            POP     DPL
            POP     DPH

            RET

DELAYL:     MOV     R5,#07H         ;4.45 mS
BAA:        MOV     R6,#0FFH
BAA1:       DJNZ    R6,BAA1
            DJNZ    R5,BAA

            RET

;****** WRITE LINE 24 CHAR *****

WRLINE:     CJNE    A,#01H,NO1
            SJMP    WRL1
NO1:        CJNE    A,#02H,NO2
            SJMP    WRL2

NO2:        RET                     ;NO WRITE LINE

WRL1:       MOV     A,#00H
            MOV     DPTR,#COMMAND
            MOVX    @DPTR,A
            LCALL   WAITBF
            LCALL   ADDRLINE
            LJMP    WRLM

ADDRLINE:   MOV     B,#24
            MOV     A,R4
            MUL     AB
            MOV     DPTR,#TAB1
            CLR     C
            ADDC    A,DPL
            MOV     DPL,A
            MOV     A,B
            ADDC    A,DPH
            MOV     DPH,A

            RET

WRL2:       MOV     A,#0C0H
            MOV     DPTR,#COMMAND
            MOVX    @DPTR,A
            LCALL   WAITBF
            LCALL   ADDRLINE
            LJMP    WRLM

WRLM:       MOV     R0,#24D         ;24 CHR
WRL:        MOVX    A,@DPTR
            MOV     R3,A
            PUSH    DPH
            PUSH    DPL
            LCALL   WRBYTE
            POP     DPL
            POP     DPH
            INC     DPTR
            DJNZ    R0,WRL
```

```
            RET

WRBYTE:     MOV     DPTR,#WRITEDATA
            MOV     A,R3            ;data byte
            MOVX    @DPTR,A
            LCALL   WAITBF          ;Wait LCD module ready

            RET

DELAYM:     MOV     R2,#25H         ;1 round / 5 S.
            MOV     R0,#0FFH
BOON:       MOV     R1,#0FFH
BOON1:      DJNZ    R1,BOON1
            DJNZ    R0,BOON
            DJNZ    R2,BOON
            DJNZ    R7,DELAYM

            RET

DELAYS3:    MOV     R2,#14H         ;DELAY 3 S.
            MOV     R0,#0FFH
HUU:        MOV     R1,#0FFH
HUU1:       DJNZ    R1,HUU1
            DJNZ    R0,HUU
            DJNZ    R2,HUU

            RET

DELAYS1:    MOV     R2,#07H         ;DELAY 1 S.
            MOV     R0,#00H
            MOV     R1,#00H
KEEN:       DJNZ    R1,KEEN
            DJNZ    R0,KEEN
            DJNZ    R2,KEEN
            MOV     R2,#15H
KEEN1:      MOV     R0,#35H
KEEN2:      DJNZ    R0,KEEN2
            DJNZ    R2,KEEN1

            RET


DEB:        MOV     R5,#14H         ;DEBOUNCE 10 mS.
BEE:        MOV     R6,#0FFH
BEE1:       DJNZ    R6,BEE1
            DJNZ    R5,BEE

            RET

DEB2:       MOV     R5,#34H         ;DEBOUNCE 30 ms.
BEE2:       MOV     R6,#0FFH
BEE3:       DJNZ    R6,BEE3
            DJNZ    R5,BEE2

            RET

DEB3:       MOV     R5,#0B2H
PZN:        MOV     R6,#0FFH
PZN1:       DJNZ    R6,PZN1
            DJNZ    R5,PZN

            RET
```

```
TAB1:       DB      "SYSTEM NO.1 READY          "     ;0
            DB      "SYSTEM NO.2 READY          "     ;1
            DB      "MOTOR NO.1 RUNNING (S.1)"        ;2
            DB      "MOTOR NO.2 RUNNING (S.1)"        ;3
            DB      "MOTOR NO.3 RUNNING (S.1)"        ;4
            DB      "MOTOR NO.4 RUNNING (S.1)"        ;5
            DB      "MOTOR NO.5 RUNNING (S.2)"        ;6
            DB      "MOTOR NO.6 RUNNING (S.2)"        ;7
            DB      "MOTOR NO.7 RUNNING (S.2)"        ;8
            DB      "MOTOR NO.8 RUNNING (S.2)"        ;9
            DB      "SYSTEM NO.1 RUNNING        "     ;10
            DB      "SYSTEM NO.2 RUNNING        "     ;11
            DB      "SYSTEM NO.1 ERROR          "     ;12
            DB      "SYSTEM NO.2 ERROR          "     ;13
            DB      "MOTOR NO.1  OVER LOAD      "     ;14
            DB      "MOTOR NO.2  OVER LOAD      "     ;15
            DB      "MOTOR NO.3  OVER LOAD      "     ;16
            DB      "MOTOR NO.4  OVER LOAD      "     ;17
            DB      "MOTOR NO.5  OVER LOAD      "     ;18
            DB      "MOTOR NO.6  OVER LOAD      "     ;19
            DB      "MOTOR NO.7  OVER LOAD      "     ;20
            DB      "MOTOR NO.8  OVER LOAD      "     ;21
            DB      "MOTOR NO.1  READY          "     ;22
            DB      "MOTOR NO.2  READY          "     ;23
            DB      "MOTOR NO.3  READY          "     ;24
            DB      "MOTOR NO.4  READY          "     ;25
            DB      "MOTOR NO.5  READY          "     ;26
            DB      "MOTOR NO.6  READY          "     ;27
            DB      "MOTOR NO.7  READY          "     ;28
            DB      "MOTOR NO.8  READY          "     ;29
            DB      "WAIT FOR 5 Sec.            "     ;30
            DB      "WAIT FOR 10 Sec.           "     ;31
            DB      "WAIT FOR 3 Min.            "     ;32
            DB      "SYSTEM READY TO RUN NEW!"        ;33
            DB      "CURRENT TEMPERATURE        "     ;34


COUNT:      DS      3
DISBUF:     DS      8

UPTIME:     MOV     DPTR,#COUNT+2 ;set TIME XX XX XX + 1
            MOVX    A,@DPTR
            ADD     A,#01H
            DA      A
            CLR     C
            CJNE    A,#60H,UPSEC
            CLR     A
UPSEC:      MOV     DPTR,#COUNT+2 ;set SECOUND
            MOVX    @DPTR,A
            JC      OUTT

            MOV     DPTR,#COUNT+1
            MOVX    A,@DPTR
            ADD     A,#01H
            DA      A
            CLR     C
            CJNE    A,#60H,UPMIN
            CLR     A
UPMIN:      MOV     DPTR,#COUNT+1 ;set MINUTE
            MOVX    @DPTR,A
            JC      OUTT
```

```
                MOV     DPTR,#COUNT
                MOVX    A,@DPTR
                ADD     A,#01H
                DA      A
                MOVX    @DPTR,A

OUTT:           NOP

                RET

CONVT:          MOV     R0,#03H         ;increment TIME
                MOV     DPTR,#DISBUF
                MOV     R3,DPH
                MOV     R4,DPL
                MOV     DPTR,#COUNT
                MOV     R1,DPH
                MOV     R2,DPL

DONE:           MOV     DPL,R2
                MOV     DPH,R1
                MOVX    A,@DPTR
                SWAP    A
                LCALL   ADJUST

                MOV     DPL,R4
                MOV     DPH,R3
                INC     DPTR
                MOV     R3,DPH
                MOV     R4,DPL
                MOV     DPL,R2
                MOV     DPH,R1
                MOVX    A,@DPTR
                LCALL   ADJUST

                MOV     DPL,R4
                MOV     DPH,R3
                INC     DPTR
                INC     DPTR
                MOV     R3,DPH
                MOV     R4,DPL
                MOV     DPL,R2
                MOV     DPH,R1
                INC     DPTR
                MOV     R1,DPH
                MOV     R2,DPL
                DJNZ    R0,DONE

WRPOI:          MOV     DPTR,#DISBUF+2;set MIDDLE POINT
                MOV     A,#3AH
                MOVX    @DPTR,A
                MOV     DPTR,#DISBUF+5
                MOVX    @DPTR,A

                RET

WRLINT:         CJNE    A,#2,SON        ;WRITE TIME XXIXX TO LCD
                SJMP    WRLT2

SON:            NOP
                RET
```

```
WRLT2:      MOV     A,#0D0H
            MOV     DPTR,#COMMAND
            MOVX    @DPTR,A
            LCALL   WAITBF
            MOV     DPTR,#DISBUF

WRLMT:      MOV     R0,#8           ;8 CHAR
WRLT:       MOVX    A,@DPTR
            MOV     R3,A
            PUSH    DPH
            PUSH    DPL
            LCALL   WRBYTE
            POP     DPL
            POP     DPH
            INC     DPTR
            DJNZ    R0,WRLT

            RET

ADJUST:     ANL     A,#0FH          ;sub of CONVT arrange TIME CODE
            MOV     DPTR,#TAB0
            CLR     C
            ADD     A,DPL
            MOV     DPL,A
            JC      REL
            SJMP    LDATA
REL:        INC     DPH
LDATA:      MOVX    A,@DPTR
            MOV     DPL,R4
            MOV     DPH,R3
            MOVX    @DPTR,A

            RET

TAB0:       DB      30H
            DB      31H,32H,33H
            DB      34H,35H,36H
            DB      37H,38H,39H

            END
```

การใช้งานไมโครคอนโทรลเลอร์ #8031

ความเจริญเติบโตทางเทคโนโลยีในปัจจุบันได้รุดหน้าไปอย่างรวดเร็วความสะดวกสบายในชีวิตประจำวันก็มีมากขึ้นเนื่องจากได้นำเอาเทคโนโลยีในสาขาต่าง ๆ มาประกอบกัน และประดิษฐ์เป็นเครื่องอำนวยความสะดวกต่าง ๆ เช่น เครื่องปรับอากาศ เครื่องซักผ้า เตาอบไมโครเวฟ ฯลฯ สิ่งเหล่านี้เกิดจากการผสมผสานกันระหว่างเทคโนโลยีในสาขาต่าง ๆ แต่สิ่งหนึ่งที่เข้ามามีบทบาทในชีวิตประจำวันของเรามากขึ้นนั่นก็ คือ ไมโครคอมพิวเตอร์ถูกนำมาใช้ควบคุมอุปกรณ์ต่างๆที่กล่าวมาข้างต้นโดยฝังตัว(enbeded) อยู่ในอุปกรณ์นั้น ๆ

แต่เป็นที่น่าเป็นห่วงสำหรับแวดวงนักเล่นคอมพิวเตอร์ฮาร์ดแวร์ของบ้านเรา ซึ่งเกิดช่องว่างระหว่างเวลากับเทคโนโลยีขึ้น คือในช่วงระยะเวลาสั้น ๆเทคโนโลยีของคอมพิวเตอร์ก็ก้าวหน้าไปในขณะที่นักคอมพิวเตอร์ฮาร์ดแวร์สมัครเล่นต้องการเวลาในการศึกษาค้นคว้าตลอดจนเสาะแสวงหาชุดสนับสนุน เช่น บอร์ดพัฒนาระบบไมโครคอมพิวเตอร์ ข้อมูลเกี่ยวกับไอซีใหม่ ๆ เมื่อหลายปีก่อนบ้านเรามี cpu เพียง Z80 ปัจจุบันมี cpu ที่ถูกผลิตขึ้นมาให้มีคุณสมบัติใช้งานในด้านการคอนโทรล หรือที่เรียกว่า single chip-microcontroller

8031 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ 8031 นี้เป็น cpu ในตระกูล MCS51 ของบริษัทอินเทล ซึ่งเป็นที่นิยมมากในปัจจุบัน คุณสมบัติคร่าง ๆ มีดังนี้

- cpu ขนาดแปดบิท
- หน่วยความจำภายใน 128 ไบท์ (ram)
- พอร์ทสื่อสารข้อมูลอนุกรม หนึ่งช่อง
- timer/counter สองช่อง
- พอร์ท I/O ขนาดแปดบิท สี่พอร์ท
- ขาอินเตอร์รัพต์ภายนอก สองช่อง
- มีวงจรกำเนิดสัญญาณคล็อกภานในตัว
- ติดต่อหน่วยความจำภายนอกได้ 128 KB (แยกเป็นหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลอย่างละ 64 KB)

ความใจร้อนอยากรู้เป็นนิสัยของช่าง หรือนักประดิษฐ์ในที่นี้จะกล่าวถึงการสร้างบอร์ด

- ชื่อบอร์ด control pack single board 8031(CP-SB31)
- cpu เบอร์ 8031
- หน่วยความจำผู้ใช้สามารถเลือกได้อย่างอิสระตลอด 128 KB โดยการเลือกเซ็ทจัมเปอร์ให้ถูกต้อง
- พอร์ท I/O 8255 หนึ่งตัว
- อินเตอร์เฟสกับ LCD ได้ทันที
- พอร์ทสื่อสารข้อมูลอนุกรม หนึ่งช่อง
- อินเตอร์เฟสกับคีย์บอร์ดขนาด 4 X 4 ได้
- เพาเวอร์ซัพพลาย 5 โวลท์บนบอร์ด
- พัฒนาโปรแกรมร่วมกับ PC ผ่านพอร์ทอนุกรม RS-232
- ขนาดบอร์ด 7.6 x14 cm.

<u>การต่อ cpu 8031 กับหน่วยความจำภายนอก</u>

ขา 31 ของ cpu($\overline{EA}$/VP) ถ้าต่อขานี้ลงกราวด์จะเป็น การติดต่อกับโปรแกรมภายนอกซึ่งจะมีผลต่อพอร์ทP0 และ P2 ดังนี้

P0 (ขา32-39) จะมัลติเพล็กซ์ระหว่าง data D),D7 และ address A0-A7.

P2 (ขา21-28) จะเป็นแอดเดรส A8-A15 ในการใช้งานในลักษณะนี้พอร์ทP0 จะต้องมีตัวต้านทานพูลอัพด้วย

ขา 30 (ALE/Pบาร์) จะเป็นสัญญูรณแอดเดรสแลตช์อีนีเบิลออกจากตัวcpu บอกอุปกรณ์ภายนอกให้ทำการแลตช์แอดเดรสไบต์ต่ำที่ถูกส่งออกมาในขณะเฟตช์คำสั่ง

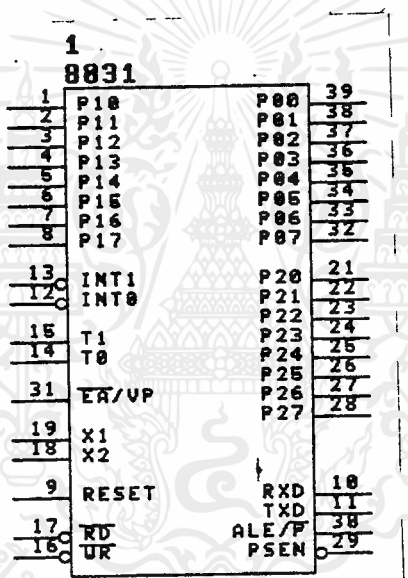
ขา 17 (RDบาร์) เป็นขาสัญญาณการอ่านข้อมูลจากหน่วยความจำข้อมูล-ภายนอก

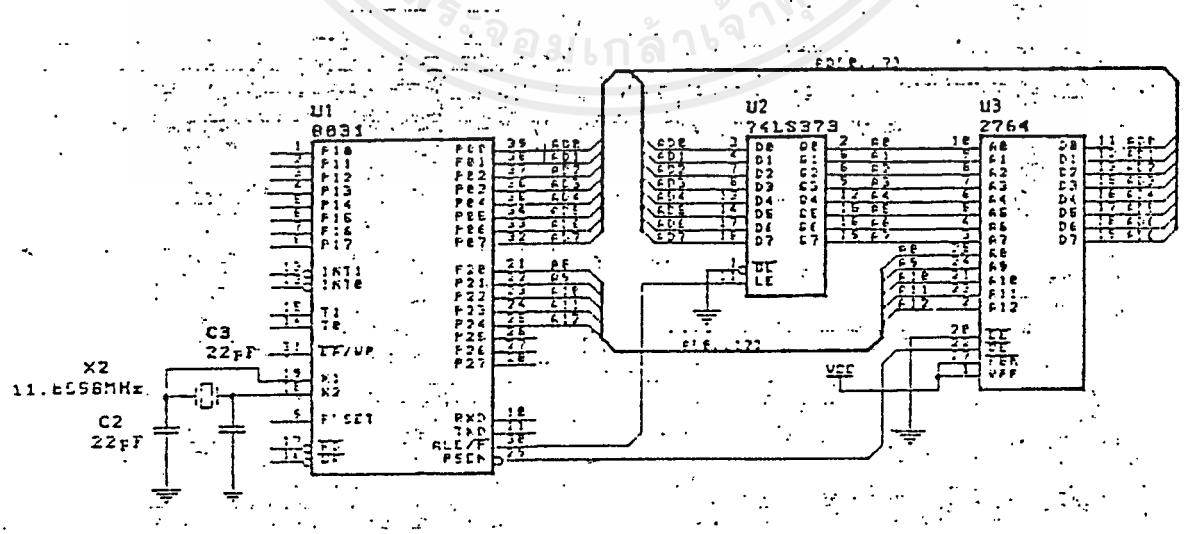
ขา 16 (WRบาร์) เป็นขาสัญญาณการเขียนข้อมูลจาก cpu ไปยังหน่วย-ความจำข้อมูลภายนอก

ขา 29 (PSEW:program store enable) เป็นขาสัญญาณออกจาก cpu ขณะทำการเฟตช์สัญญาณจากหน่วยความจำโปรแกรม

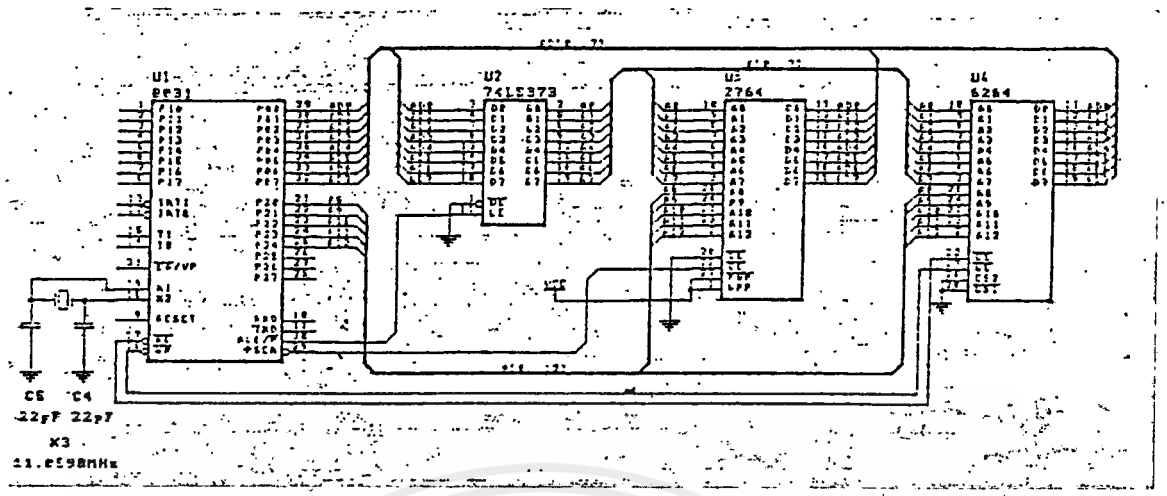
จากรูป พอร์ทP0 จะทำหน้าที่เป็นทั้งแอดเดรส และดาต้าโดยเมื่อ cpuต้องการอ่านโปรแกรมจะส่งแอดเดรสไบต์ต่ำออกทางพอร์ท P0 และ ส่งสัญญาณ ALE ให้กับ

741s373 ทำการแลตช์ เพื่อเป็นแอดเดรสของหน่วยความจำต่อไป ส่วน A8-A15 จะออกทางพอร์ท P2 เมื่อถึงไซเกิลของการเฟตช์คำสั่งที่ขาPSEN(29) ของcpu จะส่งพัลส์ลอจิก"0"ออกมา ซึ่งจากวงจรขาPSENถูกต่ออยู่กับ 2764 ข้อมูลใน 2764 จะถูกส่งออกทาง D0-D7 ไปยังพอร์ท P0 ของ cpu เพื่อนำคำสั่งไปแปลงต่อไป ส่วนอีกรูปจะมีการเพิ่มหน่วยความจำข้อมูล(data memory)ซึ่งเป็นแรมจากวงจรจะเห็นว่า cpu จะติดต่อกับแรมด้วยสัญญาณ RDบาร์/WRบาร์ ซึ่งสัญญาณทั้งสองนี้จะมีออกมาก็ต่อเมื่อ cpu ทำคำสั่งเกี่ยวกับการอ่าน/เขียนหน่วยความจำภายนอก ดังนั้นหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลจึงสามารถอยู่ในตำแหน่งเดียวกันได้ เพราะในขณะที่ cpu อ่านโปรแกรมจากรอมสัญญาณ PSEN จะแอกทีฟ ส่วนการอ่าน/เขียน แรม ซีพียู จะใช้สัญญาณ RDบาร์/WRบาร์ ในการติดต่อ จึงทำให้สามารถใช้หน่วยความจำอยู่ในตำแหน่งเดียวกันได้





รูปที่ 1 แสดงขาต่างๆ ของ CPU 8031 และการต่อกับ ROM ภายนอก

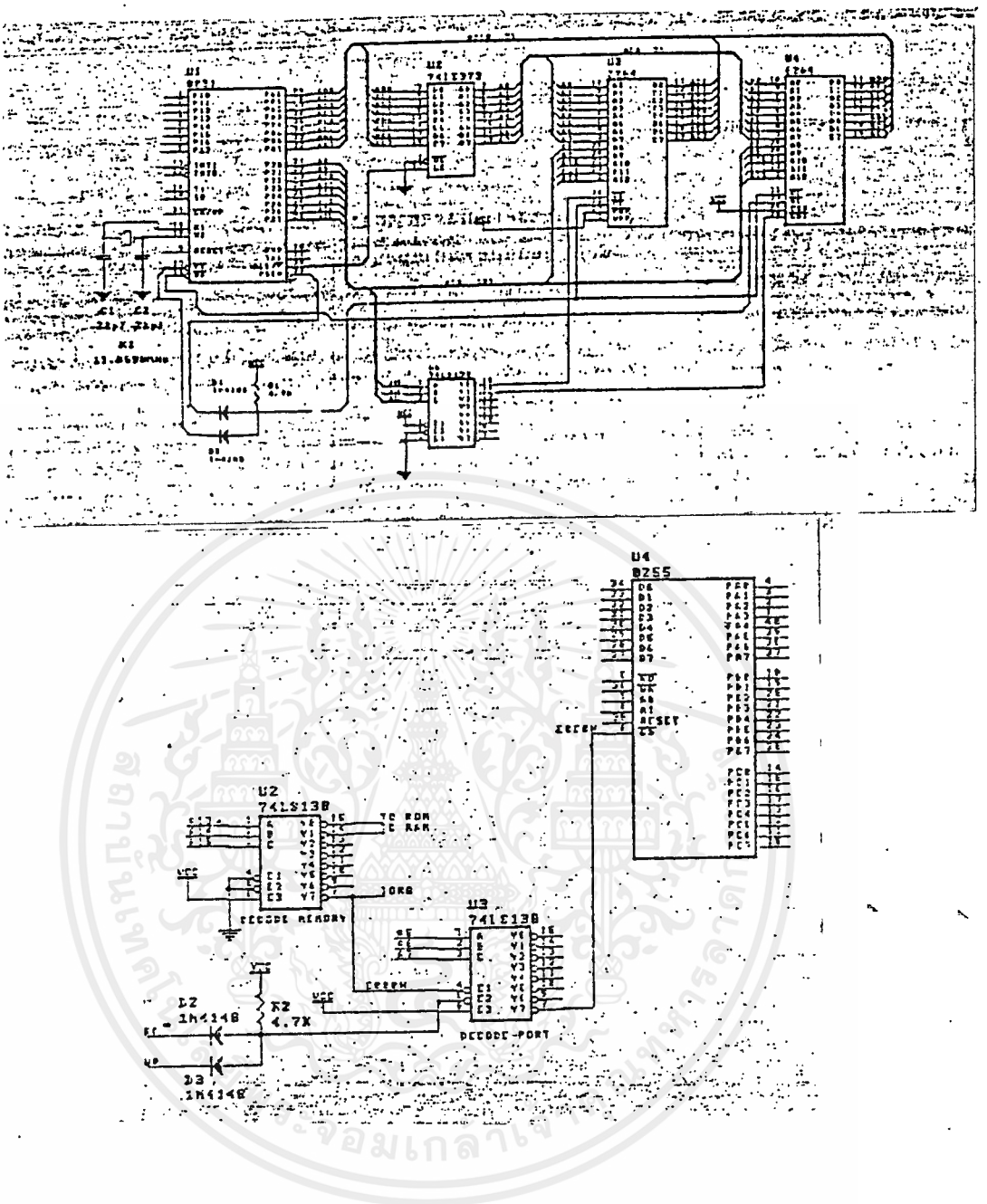


รูปที่ 2 แสดงการต่อ CPU กับ ROM และ RAM ภายนอก

จากรูป2 เป็นการต่อ 8031 กับหน่วยความจำ รอม และ แรม โดยหน่วยความจำทั้งสองชนิดจะอยู่ในหน้าเดียวกัน และการต่อแบบนี้จะอ้างหน่วยความจำได้สูงสุด 64 KB วิธีการต่อกระทำโดยการนำสัญญาณ PSENและRDบาร์มา AND กันแต่วงจรในรูป3 ใช้ไดโอดมาแทนแอนด์เกต เมื่อมีตำแหน่งหน่วยความจำอยู่ในหน้าเดียวกัน จำเป็นจะต้องกำหนดตำแหน่งของหน่วยแต่ละตัวว่าอยู่ที่ตำแหน่งใดวงจรในรูป3 จึงใช้ 74ls138 เป็นตัวดีโค้ด โดย รอมมีตำแหน่งอยู่ที่(0000-1FFFH) และแรมมีตำแหน่งอยู่ที่ (2000-3FFFH)

ถึงแม้ว่าจะเหลือพอร์ทอินพท/เอาท์พุทขนาดแปดบิทไว้ใช้ แต่ในบางครั้งก็ไม่เพียงพอกับความต้องการของผู้ใช้ จำเป็นต้องต่อพอร์ทขยายออกไปอีกบอร์ด ที่เรากำลังสร้างอยู่นี้เลือกขยายด้วย 8255

ปัญหาอยู่ที่ว่า 8031ไม่มีสัญญาณ IORQบาร์ เหมือน Z80 ที่ใช้ติดต่อกับพอร์ทภายนอก ดังนั้นจึงใช้วิธี "I/O map memory" คือการให้ cpu มองหน่วยความจำ-ตำแหน่งหนึ่งเป็นพอร์ทในที่นี้เราใช้หน่วยความจำตำแหน่ง E000H เป็นต้นไปเป็นตำแหน่งของI/O และที่ตำแหน่งE000H นี้เราใช้เป็นสัญญาณ IORQบาร์ เทียมเพื่อให้บอร์ดที่สร้างขึ้นนี้สามารถใช้กับอุปกรณ์ที่ต้องการสัญญาณ IORQบาร์ ด้วย ส่วนตำแหน่งของพอร์ทจะถูกกำหนดด้วยแอดเดรสไบต์ต่ำ A5-A7 โดยผ่าน 74ls138 เป็นตัวดีโค้ดเบอร์พอร์ทจากวงจรใช้การดีโค้ดสองชั้นแบบนี้ในทางทฤษฎีแล้วอาจจะวิกฤตไปบ้างแต่ในทางปฏิบัติแล้วใช้งานได้ดี



รูปที่ 3 แสดงการต่อ 8031 กับหน่วยความจำโดยการรวมกับหน่วยความจำโปรแกรม และ หน่วยความจำข้อมูล ไว้ในหน้าเดียวกัน เบอร์ของชิพหน่วยความจำ

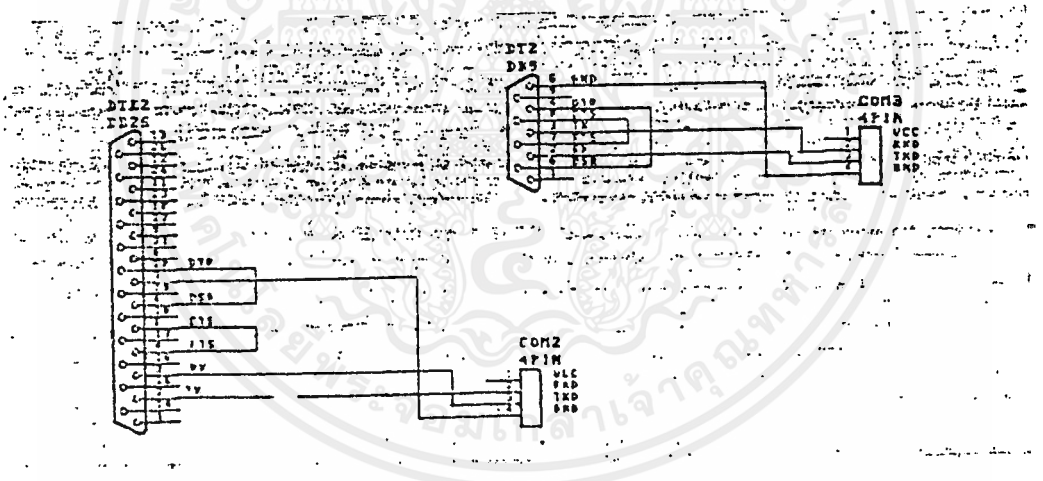
พอร์ทสื่อสารข้อมูลอนุกรม

8031 มีขา TX และ RX สำหรับส่งและรับข้อมูลอนุกรมตามลำดับซึ่งอยู่ที่ขา

11 และขา 10 แต่เนื่องจากสัญญาณที่ขาทั้งสองอยู่ในระดับ TTL แต่สัญญาณที่ช่องRS-232 อยู่ในช่วง15 โวลต์ฉะนั่นจึงต้องมีวงจรอินเตอร์เฟส เพื่อจัดระดับสัญญาณให้เหมาะสมจากวงจรสมบูรณ์ในรูป5 จะได้ระดับสัญญาณประมาณ 8 โวลต์ ซึ่งก็เพียงพอกับการสื่อสารข้อมูลในระยะที่ไม่ไกลนัก

สร้างสายเชื่อมต่อ RS-232 กับ CP-SB31

การสื่อสารข้อมูลอนุกรมระหว่างCP-SB31 กับ พีซี นั้นเราใช้สัญญาณเพียงสามเส้น คือ TX, RX และ GND แต่ในการส่ง-รับข้อมูลของ พีซี นั้นโปรแกรมจะทำแฮนด์เช็ค ทางฮาร์ดแวร์ด้วย แต่เนื่องจาก CP-SB31 ไม่มีสัญญาณที่จะทำแฮนด์เช็คโดยตรง เราจึงจำเป็นต้องหลอก พีซี ด้วยการทำการป้อนกลับสัญญาณนี้ให้กับตัวเองโดยต่อขา RTS เข้ากับขา CTS (ขา4 กับขา5) และต่อขา DSR เข้ากับขา DTR (ขา6 กับขา20)



รูปที่ 4 การต่อสัญญาณ RS-232

ACALL addr11

ความหมาย เป็นคำสั่งเรียนแบบไม่มีเงื่อนไข ไปยังตำแหน่งที่ชี้ด้วยค่าแอดเดรสที่กำหนด โปรแกรมเคาเตอร์จะเพิ่มค่าขึ้นสองครั้ง เพื่อรับแอดเดรสในคำสั่งที่ตามมาจากนั้นใส่ผลลัพธ์ 16 บิท ลงบนสแตค (8บิทต่ำก่อน) แล้วเพิ่มค่าสแตคพอนย์เตอร์ขึ้นสองครั้ง แอดเดรสปลายทางได้รับจาก 5 บิทสูงของโปรแกรมเคาเตอร์ที่เพิ่มขึ้น ออปโค๊ดบิทที่ 5-7 และไบท์ที่สองต่อจากคำสั่ง โปรแกรมย่อยที่ถูกเรียกจะต้องเริ่มภายในบล็อก (2K) ของหน่วยความจำโปรแกรมเดียวกัน ซึ่งอยู่ในไบท์แรก ของคำสั่งต่อจาก ACALL คำสั่งนี้ไม่มีผลต่อแฟลก

ตัวอย่าง เริ่มต้นให้สแตคพอนย์เตอร์มีค่า 07H และ label"SUBRTN" ในหน่วยความจำโปรแกรมอยู่ที่ตำแหน่ง 0345H หลังจากทำคำสั่ง ACALL SUBRTN แล้วที่ตำแหน่ง 0123H สแตคพอนย์เตอร์จะมีข้อมูล 09H หน่วยความจำข้อมูลภายในตำแหน่ง 08H และ 09H จะมีค่า 23H และ 01Hตามลำดับ โปรแกรมเคาเตอร์จะมีค่า 0345H

ENCODE :

a_{10}	a_9	a_8	1	0	0	0	1	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

ADD A,<src-byte>

ความหมาย จะทำการบวกค่าในตัวแปร (8บิท) กับแอคคิวมูเลเตอร์แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์ ถ้ามีตัวทดจากบิท 7 C-แฟลกจะเซ็ทค่าเป็น 1 ถ้ามีตัวทดออกจากบิท 3 AC-แฟลกจะเซ็ทค่าเป็น 1 มิฉะนั้นจะถูกเคลียร์ เมื่อทำการบวกจำนวนเต็มแบบไม่คิดเครื่องหมาย C-แฟลกจะเป็นตัวชี้โอเวอร์โฟล์วที่เกิดขึ้น OV-แฟลกจะถูกเซ็ทเมื่อมีตัวทดออกจากบิท 6 แต่ไม่ออกจากบิท 7 หรือมีตัวทดออกจากบิท 7แต่ไม่ออกจากบิท 6 มิฉะนั้นแล้ว OV-แฟลกจะเป็นตัวชี้ค่าลบของผลลัพธ์ จากการบวกค่าทั้งสองที่เป็นบวก หรือจะชี้ผลบวกที่เป็นบวกจากการบวกเลขลบทั้งสองก็ได้ โอเปอแรนด์ที่ใช้อ้างแอดเดรสมี 4 โหมด REGISTOR,DIRECT,REG-INDIRECT,IMMEDIATE

ตัวอย่าง แอคคิวมูเลเตอร์มีค่า 0C3H(11000011B) และR0 มีค่า 0AAH(10101010B) หลังจากทำคำสั่ง ADD A,R0 จะได้ผลลัพธ์ 6DH(01101101B) เก็บไว้ในแอคคิวมูเลเตอร์ AC-แฟลกจะถูกเคลียร์ C และ OV-แฟลกจะเซ็ทค่าเป็น 1

ADD A,Rn

ความหมาย ทำการบวกค่าของแอคคิวมูเลเตอร์กับข้อมูลใน Rn แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE :

0 0 1 0	0 r r r

ADD A,DIRECT

ความหมาย ทำการบวกค่าในแอคคิวมูเลเตอร์กับข้อมูลในแอดเดรสที่กำหนด แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE :

0 0 1 0	0 1 0 1	direct address

ADD A,@Ri

ความหมาย ทำการบวกค่าในแอคคิวมูเลเตอร์กับ ข้อมูลในแอดเดรสที่กำหนดด้วย Ri แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE :

0 0 1 0	0 1 0 i

ADD A,#data

ความหมาย ทำการบวกค่าในแอคคิวมูเลเตอร์กับ ข้อมูลที่เข้ามากับคำสั่งแล้วเก็บผลลัพธ์ไว้ในแอคคคิวมูเลเตอร์

ENCODE :

0 0 1 0	0 1 0 1	immediate data

ADDC A,<src-byte>

ความหมาย ทำการบวกพร้อมกัน ระหว่างค่าในตัวแปร (8บิท) C-แฟลก แอคคิวมูเลเตอร์

แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์ C-แฟลกจะถูกเซ็ทถ้ามีตัวทดออกจากบิท7 และAC-แฟลกจะถูกเซ็ทถ้ามีตัวทดออกจากบิท3 มิฉะนั้นจะถูกเคลียร์เมื่อทำการบวกเลขจำนวนเต็มแบบไม่คิดเครื่องหมาย C-แฟลกจะเป็นตัวชี้ OVที่เกิดขึ้น

OV-แฟลกจะถูกเซ็ทถ้ามีตัวทดออกจากบิท6 แต่ไม่ออกจากบิท7 หรือมีตัวทดออกจากบิท7 แต่ไม่ออกจากบิท6 มิฉะนั้นจะถูกเคลียร์ เมื่อทำการบวกเลขจำนวนเต็มแบบคิดเครื่องหมาย จะชี้ค่าลบของผลบวกระหว่างเลขบวกสองค่าหรือจะชี้ค่าบวกจากการบวกระหว่างเลขลบสองค่า โอเปอแรนด์ที่ใช้อ้างแอดเดรสมี สี่โหมดคือ REGISTOR,DIRECT REG-INDIRECT,IMMEDIATE

ADDC A,Rn

<u>ความหมาย</u> ทำการบวกค่าในแอคคิวมูเลเตอร์ C-แฟลก และRn เข้าด้วยกันแล้วเก็บผลลัพธ์ที่ได้ไว้ในแอคคิวมูเลเตอร์

ENCODE :

0 0 1 1	1 r r r

ADDC A,DIRECT

<u>ความหมาย</u> ทำการบวกค่าในแอคคิวมูเลเตอร์ c-แฟลก กับข้อมูลในแอดเดรสที่กำหนดแล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE :

0 0 1 1	0 1 0 1	direct address

ADDC A,@Ri

<u>ความหมาย</u> ทำการบวกค่าในแอคคิวมูเลเตอร์ C-แฟลก กับข้อมูลในแอดเดรสที่กำหนดด้วย Riแล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE :

0 0 1 1	0 1 1 i

ADDC A,#data

ความหมาย ทำการบวกค่าในแอคคิวมูเลเตอร์ C-แฟลก กับข้อมูลที่เข้ามากับคำสั่งแล้วเก็บผลลัพธ์ไว้ในแอคคคิวมูเลเตอร์

ENCODE :

0 0 1 1	0 1 0 0	immediate data

AJMP addr11

ความหมาย ทำการย้ายการปฏิบัติโปรแกรมไปยังแอดเดรสที่เรากำหนด ซึ่งโปรแกรมเคาเตอร์จะเพิ่มค่าขึ้นสองครั้ง แอดเดรสปลายทางต้องอยู่ในบล็อกสอง เค ภายในหน่วยความจำโปรแกรมเดียวกัน โดยจะใส่ไว้ในไบท์แรกต่อจากคำสั่ง AJMP

ตัวอย่าง label"JMPADR"ในหน่วยความจำโปรแกรมอยู่ที่ตำแหน่ง 0123H คำสั่ง AJMP JMPADR อยู่ที่แอดเดรส 0345H ผลลัพธ์จะโหลดค่าโปรแกรมเคาเตอร์ด้วย 0123H

ENCODE :

a_{10} a_9 a_8 0	0 0 0 1	a_7 a_6 a_5 a_4	a_3 a_2 a_1 a_0

ANL <dest-byte>,<src-byte>

ความหมาย กระทำlogic-AND ระหว่างค่าในตัวแปรทั้งสองและเก็บผลลัพธ์ไว้ในตัวแปรปลายทาง คำสั่งนี้ไม่มีผลต่อแฟลก โอเปอแรนด์ทั้งสองอ้างแอดเดรสได้ สี่โหมด คือ register, direct, reg-indirect,immediate และเมื่อปลายทางอ้างแอดเดรสแบบ direct, source สามารถอ้างแอดเดรสแบบ immediate หรือแอคคิวมูเลเตอร์

ตัวอย่าง ถ้าแอคคิวมูเลเตอร์มีค่า 0C03H(11000011B)และRO มีค่า 55H(01010101B) เมื่อทำคำสั่ง ANL A,RO จะได้ผลลัพธ์ 441H(01000001B) เก็บไว้ในแอคคิวมูเลเตอร์ เมื่อปลายทางอ้างแอดเดรสแบบ direct คำสั่งจะเคลียร์ทุกบิทในแรม บางตำแหน่งหรือ hardware register, mask byte จะหารูปแบบบิทที่ถูกเคลียร์ถ้าไม่เป็นค่าคงที่ในคำสั่งก็จะเป็นค่าที่คำนวนในแอคคิวมูเลเตอร์ ขณะรันต์คำสั่ง ANL P1,#01110011B จะทำการเคลียร์บิท7 บิท3 และบิท2 ของพอร์ท 1

ANL A,Rn

ความหมาย ทำlogic-ANDระหว่างแอคคิวมูเลเตอร์กับข้อมูลในRn แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE :

0 1 0 1	1 r r r

ANL A,DIRECT

ความหมาย ทำlogic-AND ระหว่างแอคคิวมูเลเตอร์กับข้อมูลในแอดเดรสที่กำหนด แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE :

0 1 0 1	0 1 0 1	direct address

ANL A,@Ri

ความหมาย ทำlogic-AND ระหว่างแอคคิวมูเลเตอร์กับ ข้อมูลในแอดเดรสที่กำหนดด้วย Ri แล้วเก็บผลลัพธ์ไว้ในแอคคิวมูเลเตอร์

ENCODE :

0 1 0 1	0 1 1 i

ANL A,#data

ความหมาย ทำlogic-AND ระหว่างแอคคิวมูเลเตอร์กับ ข้อมูลที่เข้ามากับคำสั่งแล้วเก็บผลลัพธ์ไว้ในแอคคคิวมูเลเตอร์

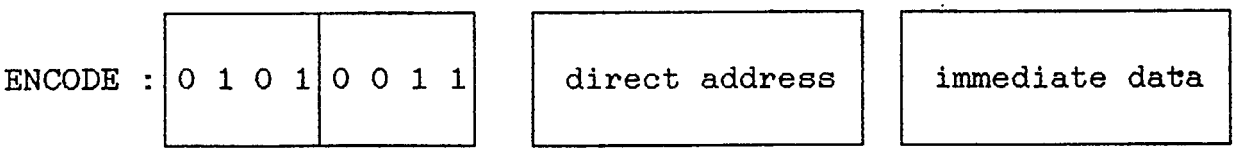
ENCODE :

0 1 0 1	0 1 0 0	immediate data

ANL direct,#data

ความหมาย ทำlogic-AND ระหว่างข้อมูลในแอดเดรสที่กำหนด กับข้อมูลที่เข้ามากับคำสั่ง

แล้วเก็บผลลัพธ์ไว้ในแอดเดรสที่กำหนดนั่น



ANL C,<src-bit>

ความหมาย ถ้าค่าของsrc-bit มีlogic 0 จะเป็นการเคลียร์ C-แฟลก มิฉะนั้นจะย้าย C_แฟลกไปสู่สถานะ current state เครื่องหมาย"/"แสดงให้เห็นว่าเป็นการคอมพลีเมนท์บิทของแอดเดรส และใช้เป็น source เท่จะไม่มีลต่อ source bitและแฟลกอื่นๆ src-bit อ้างแอดเดรสแบบ direct เท่านั้น

ANL C,bit

ความหมาย ทำlogic-AND ระหว่างแอดเดรสบิทที่กำหนดกับ C-แฟลก แล้วเก็บไว้ใน C-แฟลก

CJNE<dest-byte>,<src-byre>,rel

ความหมาย ทำการเปรียบเทียบค่าของโอเปอร์แรนด์แรก และจะกระโดดไปถ้าผลลัพธ์ไม่ไม่เท่ากัน แอดเดรสปลายทางจะคำนวนได้โดยการบวกแบบคิดเครื่องหมาย relative แล้วส่งในคำสั่งไบท์สุดท้ายไปยังโปรแกรมเคาเตอร์ หลังจากนั้นเพิ่มค่าโปรแกรมเคาเตอร์ไปที่จุดเริ่มต้นของคำสั่งต่อไป c-แฟลกจะถูกเซ็ทถ้าเลขจำนวนเต็มแบบไม่คิดเครื่องหมายของ<dest-byte>น้อยกว่า<src-byte> นอกนั้นc-แฟลกจะถูกเคลียร์ คำสั่งนี้ไม่มีผลต่อโอเปอร์แรนด์ทั้งสอง ซึ่งในสองโอเปอร์แรนด์แรกสามารถอ้างแอดเดรสได้ สี่โหมด แอคคิวมูเลเตอร์อาจจะถูกเปรียบเทียบกับ ไบท์ที่อ้างแอดเดรสแบบ direct หรือ แบบ immediate และอ้างตำแหน่งของแรม หรือโดยอ้างผ่าน register แล้วเปรียบเทียบกับค่าคงที่ immediate

ตัวอย่าง แอคคิวมูเลเตอร์มีค่า34H, R7มีค่า 56H เมื่อทำคำสั่ง

CJNE R7, #60H,NOT]EQ

```
            .........           ;R7=60H
NOT]EQ:     JC  REQ]LOW         ;IF R7<60H
            .........           ;R7>60H
```

จะทำการเซ็ทค่า c-แฟลก และแยกสาขาไปทำคำสั่งที่ label"NOT]EQ"คำสั่งนี้จะกำหนดว่า R7 มากกว่าหรือน้อยกว่า 60H โดยการทดสอบc-แฟลกถ้าข้อมูลที่พอร์ท 1 เป็น34H ดังนั้นเมื่อทำคำสั่ง

```
WAIT:  CJNE A,P1,WAIT
```

จะเคลียร์c-แฟลก และทำคำสั่งต่อไป เมื่อข้อมูลที่อ่านจากพอร์ท1 เท่ากับแอคคิวมูเลเตอร์ ถ้าพอร์ท1 เป็นค่าอื่นโปรแกรมก็จะวนที่จุดนี้จนกว่าพอร์ท1 จะเปลี่ยนข้อมูลเป็น 34H

CJNE A,direct,rel

ความหมาย ทำการเปรียบเทียบแอคคิวมูเลเตอร์กับข้อมูลในแอดเดรสที่กำหนด ถ้าไม่เท่ากัน จะกระโดดไปยังแอดเดรสที่ต้องการ แต่ถ้าเท่ากันจะทำคำสั่งต่อไป

ENCODE :

1 0 1 1	0 1 0 1	direct address	rel.address

CJNE A,#data,rel

ความหมาย ทำการเปรียบเทียบแอคคิวมูเลเตอร์กับข้อมูลที่เข้ามากับคำสั่งถ้าไม่เท่ากันจะกระโดดไปยังแอดเดรสที่ต้องการ แต่ถ้าเท่ากันจะทำคำสั่งต่อไป

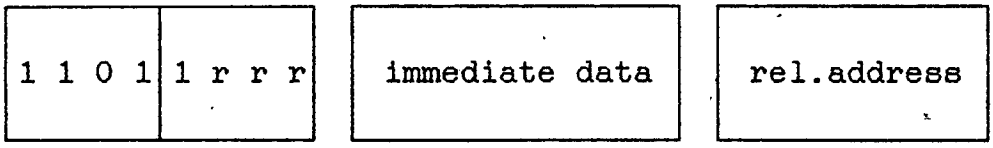
ENCODE :

1 0 1 1	0 1 0 1	immediate data	rel.address

CJNE Rn,data,rel

ความหมาย ทำการเปรียบเทียบกับข้อมูลในRn กับข้อมูลที่เข้ามาในคำสั่ง ถ้าไม่เท่ากันจะกระโดดไปยังแอดเดรสที่ต้องการ แต่ถ้าเท่ากันจะทำคำสั่งต่อไป

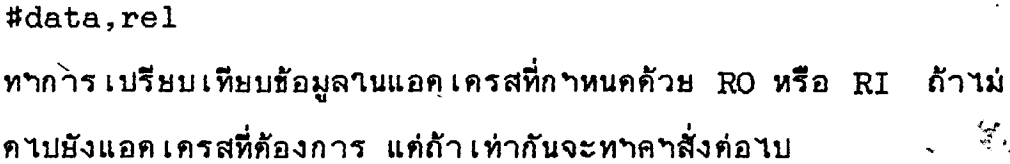
ENCODE :



CJNE @Ri,#data,rel

ความหมาย ทำการเปรียบเทียบข้อมูลในแอคเครสที่กำหนดด้วย R0 หรือ R1 ถ้าไม่เท่ากันจะกระโดดไปยังแอดเดรสที่ต้องการ แต่ถ้าเท่ากันจะทำคำสั่งต่อไป

ENCODE :

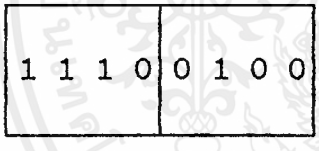


CLR A

ความหมาย เคลียร์แอคคิวมูเลเตอร์ให้ทุกบิทเป็นศูนย์ และไม่มีผลต่อแฟลก

ตัวอย่าง แอคคิวมูเลเตอร์มีค่า 5CH(01011101B) เมื่อทำคำสั่ง CLR A จะเซ็ทค่าในแอคคิวมูเลเตอร์เป็น 00H

ENCODE :



CLR bit

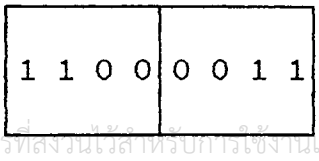
ความหมาย เคลียร์บิทที่ถูกชี้ คำสั่งนี้ไม่มีผลต่อแฟลกอื่นๆและสามารถทำบนc-แฟลก หรือบิทใด ๆ ที่อ้างแอดเดรสโดยตรง

ตัวอย่าง พอร์ท1 ถูกเขียนด้วยข้อมูล 5DH(01011101B)มาก่อนเมื่อทำคำสั่งCLR P1..2 จะได้ 059H(0101111001B) เก็บไว้ในพอร์ท1

CLR C

ความหมาย เคลียร์c-แฟลก ให้เป็นศูนย์

ENCODE :



CLR bit

ความหมาย เคลียร์บิทใด ๆ ในแอคเครสที่ต้องการ

ENCODE :	1 1 0 0	0 0 1 1	bit address

CPL A

ความหมาย คอมพลีเมนต์ค่าในแอคคิวมูเลเตอร์ แต่ละบิทของแอคคิวมูเลเตอร์จะถูกคอมพลีเมนต์โดยบิทซึ่งเคยเป็น1 ก็จะถูกเปลี่ยนเป็น0 และในทางกลับกัน คำสั่งนี้ไม่มีผลต่อแฟลก

ตัวอย่าง แอคคิวมูเลเตอร์มีค่า 5CH(01011100B) เมื่อทำคำสั่ง CPL A จะได้ค่า 0A3H(10100011B) เก็บไว้ยังแอคคิวมูเลเตอร์

ENCODE :	1 1 1 1	0 1 0 0

CPL bit

ความหมาย บิทที่กำหนดจะถูกคอมพลีเมนต์ โดยบิทที่เคยเป็น1 ก็จะถูกเปลี่ยนมาเป็น0 และในทางกลับกัน คำสั่งนี้ไม่มีผลต่อแฟลกอื่น ๆ ซึ่งคำสั่งนี้ยังสามารถกระทำบนc-แฟลก หรืออ้างแอดเดรสแบบบิทdirect ก็ได้

ตัวอย่าง พอร์ท1 ถูกเขียนด้วย 5BH(01011011B) มาก่อน เมื่อทำคำสั่ง

```
CLR  P1.1
CLR  P1.2
```

จะเซ็ทข้อมูลที่พอร์ท1 ได้เป็น 59H(01011001B)

CPL bit

ความหมาย คอมพลีเมนท์บิทใด ๆ ในแอดเดรสที่ต้องการ

ENCODE :	1 0 1 1	0 0 1 0	bit address

DA A

ความหมาย ทำการปรับค่า แปดบิทในแอคคิวมูเลเตอร์ซึ่งเป็นผลลัพธ์จากการบวกเกินสองตัวแปรและสร้าง BCD 4บิทขึ้นมาสองหลัก โดยอาจใช้คำสั่ง ADD หรือ ADDC เพื่อทำการบวกก็ได้ ถ้าแอคคิวมูเลเตอร์บิทศูนย์ ถึงบิทสามมีค่าเกิน9(xxxx1010-xxxx1111)หรือถ้าac-แฟลกถูกเซ็ท จะบวกหกเข้ากับแอคคิวมูเลเตอร์ เพื่อสร้าง BCDที่ต้องในบิทต่ำการบวกที่เกิดขึ้นภายในนี้จะเซ็ทc-แฟลก ถ้ามีตัวทดออกจากสี่บิทต่ำ ถ่ายทอดออกไปผ่านสี่บิทสูง แต่จะไม่ทำการเคลียร์c-แฟลก

ถ้าc-แฟลกถูกเซ็ทหรือถ้าสี่บิทสูงมีค่าเกิน9(1010xxxx-1111xxxx)จะบวกหกเข้ากับสี่บิทสูงนี้ เพื่อสร้าง BCDที่ถูกต้องและจะเซ็ทc-แฟลกถ้ามีตัวทดออกจากสี่บิทสูงดังนั้นc-แฟลกจะชี้ผลบวกของ BCD ทั้งสองเกิน100 เพื่อความแม่นยำของการบวกเลขฐานสิบ คำสั่งนี้ไม่มีผลต่อOV-แฟลก ทุกอย่างที่เกิดขึ้นในระหว่างไซเกิลคำสั่ง จำเป็นที่จะต้องเปลี่ยนเป็นฐานสิบโดยการบวก 00,06,60,66เข้ากับแอคคิวมูเลเตอร์ ตามค่าเริ่มแรกของแอคคิวมูเลเตอร์และสภาวะของPSW

หมายเหตุ คำสั่งDA Aไม่สามารถแปลงเลขฐานสิบหกในแอคคิวมูเลเตอร์ให้เป็นค่าBCD ได้และใช้กับการลบเลขฐานสิบไม่ได้เช่นกัน

ตัวอย่าง แอคคิวมูเลเตอร์มีค่า 56H(01010110B) ถูกแบ่งเป็นBCD สองหลักและR3มีค่า 67h(01100111b) ถูกแบ่งเป็นBCD สองหลักขณะที่c-แฟลกถูกเซ็ท เมื่อทำคำสั่ง

```
ADDC A,R3
DA  A
```

ขั้นแรกจะทำการบวกแบบ 2's complement ได้ผลลัพธ์ คือ 0BEH เก็บไว้ในแอคคิวมูเลเตอร์ C และ AC-แฟลกถูกเคลียร์

คำสั่งจะเปลี่ยนค่าแอคคิวมูเลเตอร์เป็น 24H(00100100B) แบ่งเป็นBCD สองหลัก c-แฟลกจะถูกเซ็ทด้วยคำสั่งนี้แสดงว่าเกิด overflowในฐานสิบ ผลบวกที่ถูกต้องของ56, 67,1คือ 124

ค่า BCD สามารถเพิ่มหรือลดโดยการบวก01H หรือ99H ถ้าแอคคิวมูเลเตอร์เริ่มแรกมีค่า30H เมื่อทำคำสั่ง

```
ADD A,#99H
DA  A
```

จะย้ายข้อมูลใน c-แฟลก และข้อมูล 29H ไว้ในแอคคิวมูเลเตอร์เมื่อ 30+99=129,บิทต่ำของผลบวกนั้นสามารถอธิบายด้วย 30-1=29

ENCODE :

1 1 0 1	0 1 0 0

DEC byte

ความหมาย ทำการลดค่าตัวแปรลงหนึ่ง ถ้าค่าเริ่มต้นเป็น00H จะunderflow เป็น0FFH คำสั่งนี้ไม่มีผลต่อแฟลก สามารถอ้างแอดเดรสได้สี่โหมดคือ accumulator, register direct,register-indirect

ตัวอย่าง R0มีค่า7FH(01111111B)แรมภายในตำแหน่ง 7EH และ7FH มีข้อมูล 00Hและ 40H ตามลำดับ เมื่อทำคำสั่ง

DEC @R0

DEC R0

DEC @R0

R0 จะมีข้อมูล7EH และแรมภายในตำแหน่ง7EHจะมีข้อมูล 0FFH และ03FH ตามลำดับ

DEC A

ความหมาย ลดค่าแอคคิวมูเลเตอร์ลงหนึ่ง

ENCODE :

0 0 0 1	0 1 0 0

DEC Rn

ความหมาย ลดค่าRn ลงหนึ่ง

ENCODE :

0 0 0 1	1 r r r

DEC direct

ความหมาย ลดค่าของข้อมูลในตำแหน่งที่กำหนดลงหนึ่ง

ENCODE :

0 0 0 1	0 1 0 1	direct address

DEC @Ri

ความหมาย ลดค่าของข้อมูลในแอดเดรสที่กำหนดด้วยR0 หรือ R1 ลงหนึ่ง

ENCODE :

1 0 0 0	0 1 0 1

DIV AB

ความหมาย ทำการหารจำนวนเต็ม แบบบิทโดยไม่คิดเครื่องหมายซึ่งอยู่ในแอคคิวมูเลเตอร์ด้วยจำนวนเต็มชนิดเดียวกันใน reg. B ผลหารเก็บไว้ในแอคคิวมูเลเตอร์และเศษเก็บไว้ใน reg. B, c และ OV-แฟลกจะถูกเคลียร์ ถ้าreg. มีค่า00H ผลลัพธ์จะไม่มีค่า และOV-แฟลกจะถูกเซ็ท c-แฟลกจะถูกเคลียร์

ตัวอย่าง แอคคิวมูเลเตอร์มีค่า251(0FBH)และมีค่า18(12H)เมื่อทำคำสั่ง DIV AB จะได้ 13เก็บไว้ในแอคคิวมูเลเตอร์(0DH)และ17(11H)เก็บไว้ใน reg. B

เมื่อ 251=(13*18)+17 cและOV_แฟลกจะถูกเคลียร์

ENCODE :

1 0 0 0	0 1 0 0

DJNZ <byte>,<rel-addr>

ความหมาย ทำการลดค่าในตำแหน่งที่ถูกชี้ลงหนึ่งและ กระโดดไปยังตำแหน่งที่ถูกชี้ด้วยโอเปอแรนด์ที่สอง ถ้าผลลัพธ์ไม่เป็นศูนย์ ค่าที่เริ่มต้นจาก 00Hจะunderflow เป็น0FFH คำสั่งไม่มีผลต่อแฟลกแอดเดรสปลายทาง สามารถคำนวนได้โดยการบวกแบบคิดเครื่องหมายrelative ในคำสั่งไบท์สุดท้ายไปยังโปรแกรมเคาเตอร์ หลักจากนั้นจะเพิ่มค่าโปรแกรมเคาเตอร์ไปที่ไบท์แรกของคำสั่งต่อไป ตำแหน่งที่ถูกลดค่าอาจเป็น register หรือaddress byte

ตัวอย่าง แรมภายในตำแหน่ง40H,50Hและ60H มีข้อมูล 01H,70Hและ15Hตามลำดับเมื่อ

ทาคาสั่ง

```
DJNZ  40H,LABEL]1
DJNZ  50H,LABEL]2
DJNZ  60H,LABEL]3
```

จะกระโดดไปทาคาสั่งที่ label"LABEL]2"ด้วยค่า 00H,6FH,15Hในตา-แหน่งทั้งสามของแรม การกระโดดครั้งแรกจะไม่เกิดขึ้นเพราะผลลัพธ์เป็นศูนย์คาสั่งนี้เป็นตัวอย่างโปรแกรมลูป โดยให้การหน่วงเวลาพอประมาณด้วยคาสั่งเดียว

```
      MOV   R2,#8
TOOGLE :  CPL   P1.7
      DJNZ R2,TOGGLE
```

จะทาการtoggle port 1.7 8ครั้งซึ่งจะได้สี่เอาท์พุทพัลซ์ที่บิท7 ของเอาท์พุทพอร์ท1 แต่ละพัลซ์จะมีค่า 3 machine cycle, 2mcสาหรับDJNZ และ 1mc สาหรับการเปลี่ยนค่าที่พอร์ท

DJNZ Rn,rel

ความหมาย ลดค่าRnลงหนึ่งและจะกระโดดไปยังแอดเดรสที่ต้องการถ้าRnไม่เท่ากับศูนย์

ENCODE :

1 1 0 1 1 r r r	rel address

DJNZ direct,rel

ความหมาย ลดค่าในแอดเดรสที่กาหนด ถ้าค่ายังไม่เป็นศูนย์ จะบวกค่าโปรแกรมเคา-เตอร์เดิมเข้ากับrel-address แล้วเก็บไว้ในโปรแกรมเคาเตอร์ใหม่

ENCODE :

1 1 0 1 0 1 0 1	direct address	rel.address

INC <byte>

ความหมาย ทาการเพิ่มค่าตัวแปรที่ถูกชี้ขึ้นหนึ่ง ถ้าค่าเริ่มต้นเป็น 0FFH จะoverflow เป็น 00H และไม่มีผลต่อแฟลก

หมายเหตุ เมื่อใช้คำสั่งนี้เพื่อเปลี่ยนแปลงพอร์ทเอาท์พุทค่าที่ใช้ เป็นพอร์ทเริ่มต้นจะอ่านได้จากข้อมูลที่แลทช์ทางเอาท์พุทไม่ใช่ขาอินพุท

INC A

ความหมาย เพิ่มค่าแอคคิวมูเลเตอร์ขึ้นหนึ่ง

ENCODE :

0 0 0 0	0 1 0 0

INC Rn

ความหมาย เพิ่มค่า Rn ขึ้นหนึ่ง

ENCODE :

0 0 0 0	0 r r r

INC direct

ความหมาย เพิ่มค่าในแอดเดรสที่กำหนดขึ้นหนึ่ง

ENCODE :

0 0 0 0	0 1 0 1	direct address

INC @Ri

ความหมาย เพิ่มค่าในแอดเดรสที่กำหนดด้วย R0หรือR1ขึ้นหนึ่ง

ENCODE :

0 0 0 0	0 1 1 i

INC DPTR

ความหมาย ทำการเพิ่มค่าตัวชี้ข้อมูลขนาด16บิทขึ้นหนึ่ง คำสั่งนี้ไม่มีผลต่อแฟลก

ตัวอย่าง reg DPH และ DPL มีข้อมูล12H และ 0FEH เมื่อทำคำสั่ง

INC DPTR

INC DPTR

INC DPTR

จะเปลี่ยนDPH & DPL เป็น13 และ01 ตามลำดับ

ENCODE :

1 0 1 0	0 0 1 1

JB bit,rel

ความหมาย ถ้าบิทที่ถูกชี้มีค่าเป็น1 จะกระโดดไปสู่แอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไป ตำแหน่งปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมายrelativeในไบท์ที่สามของคำสั่ง เพื่อไปยังโปรแกรมเคาเตอร์ จากนั้นเพิ่มค่าโปรแกรมเคาเตอร์ไปที่ไบท์แรกของคำสั่งต่อไป บิทที่ตรวจสอบจะไม่ถูกเปลี่ยนแปลงและไม่มีผลต่อแฟลก

ตัวอย่าง ข้อมูลที่อินพุทพอร์ท1 เป็น(11001010B) ข้อมูลในแอคคิวมูเลเตอร์มีค่า 56(01010110B) เมื่อทำคำสั่ง

JB P1.2,LABEL1

JB ACC.2,LABEL2

โปรแกรมจะกระโดดไปทำที่ LABEL2

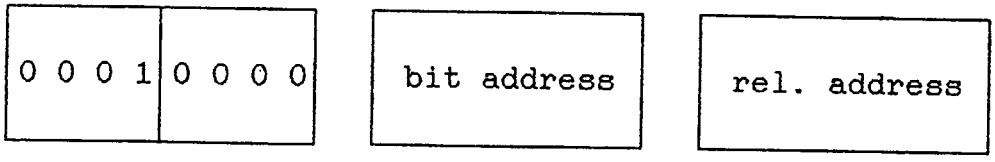
ENCODE :

0 0 1 0	0 0 0 0	bit address	rel. address

JBC bit,rel

ความหมาย ถ้าบิทที่ถูกชี้มีค่าเป็น1 จะกระโดดไปยังแอดเดรสที่ต้องการ แล้วเคลียร์บิทนั้นด้วย นอกนั้นจะทำคำสั่งต่อไป ถ้าบิทเป็นศูนย์แล้วจะไม่ถูกเคลียร์ แอดเดรสปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมายrelateve ในไบท์ที่สามของคำสั่ง เพื่อส่งค่าไปยังโปรแกรมเคาเตอร์จากนั้นเพิ่มค่าโปรแกรมเคาเตอร์ไปที่ไบท์แรกของคำสั่งต่อไป และไม่มีผลต่อแฟลก

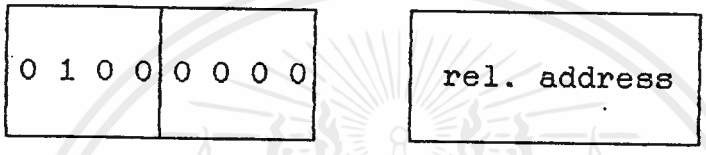
ENCODE :



JC rel

ความหมาย ถ้า c-แฟลกถูกเซ็ทจะกระโดดไปยังแอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไป แอดเดรสปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมาย relative ในไบท์ที่สองของคำสั่งเพื่อไปยังโปรแกรมเคาเตอร์ จากนั้นจะเพิ่มค่าโปรแกรมเคาเตอร์ขึ้นสองครั้ง คำสั่งนี้ไม่มีผลต่อแฟลก

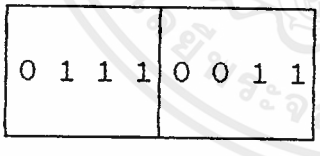
ENCODE :



JMP @A+DPTR

ความหมาย บวกค่าแปดบิทแบบไม่คิดเครื่องหมายที่อยู่ในแอคคิวมูเลเตอร์กับข้อมูลสิบหกบิทใน DPTR และเก็บผลลัพธ์ไว้ในโปรแกรมเคาเตอร์เพื่อเป็นค่าแหน่งสำหรับ fetchคำสั่งตัวทดจากแปดบิทต่ำจะส่งผ่านไปทางแปดบิทสูง ซึ่งค่าของแอคคิวมูเลเตอร์และDPTR จะไม่เปลี่ยนแปลง และไม่มีผลต่อแฟลก

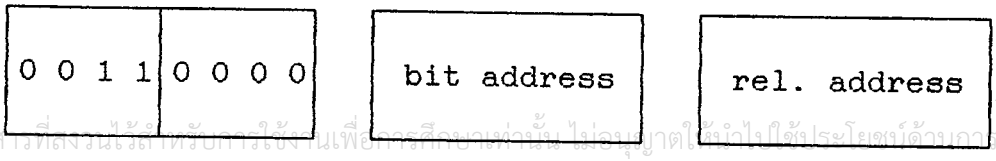
ENCODE :



JNB bit,rel

ความหมาย ถ้าบิทที่ถูกชี้มีค่าเป็น0 จะกระโดดไปยังแอดเดรสที่ต้องการมิฉะนั้นจะทำคำสั่งต่อไป แอดเดรสปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมายrelateve ในไบท์ที่-สามของคำสั่งเพื่อไปยังโปรแกรมเคาเตอร์จากนั้นเพิ่มค่าโปรแกรมเคาเตอร์ไปที่ไบท์แรกของคำสั่งต่อไป บิทที่ตรวจสอบไม่เปลี่ยนแปลงและไม่มีผลต่อแฟลก

ENCODE :



JNC rel

ความหมาย ถ้า c-แฟลกถูกเซ็ทเป็น0 จะกระโดดไปยังแอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไป แอดเดรสปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมาย relativeในไบท์ที่สองของคำสั่ง เพื่อไปยังโปรแกรมเคาเตอร์ จากนั้นจะเพิ่มค่าโปรแกรมเคาเตอร์ขึ้นสองครั้ง เพื่อชี้คำสั่งต่อไป และc-แฟลกจะไม่เปลี่ยนแปลง

ENCODE :

0 1 0 1	0 0 0 0	rel. address

JNZ rel

ความหมาย ถ้าบิทใดบิทหนึ่งของแอคคิวมูเลเตอร์เป็น1 จะกระโดดไปยังแอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไป แอดเดรสปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมาย relativeในไบท์ที่สองของคำสั่ง เพื่อไปยังโปรแกรมเคาเตอร์จากนั้นจะเพิ่มค่าโปรแกรมเคาเตอร์ขึ้นสองครั้ง แอคคิวมูเลเตอร์ และแฟลกจะไม่เปลี่ยนแปลง

ENCODE :

0 1 1 1	0 0 0 0	rel. address

JZ rel

ความหมาย ถ้าบิทใดบิทหนึ่งของแอคคิวมูเลเตอร์เป็น0 จะกระโดดไปยังแอดเดรสที่ต้องการ มิฉะนั้นจะทำคำสั่งต่อไป แอดเดรสปลายทางคำนวนโดยการบวกแบบคิดเครื่องหมาย relativeในไบท์ที่สองของคำสั่ง เพื่อไปยังโปรแกรมเคาเตอร์จากนั้นจะเพิ่มค่าโปรแกรมเคาเตอร์ขึ้นสองครั้ง แอคคิวมูเลเตอร์ และแฟลกจะไม่เปลี่ยนแปลง

ENCODE :

0 1 1 0	0 0 0 0	rel. address

LCALL addr16

ความหมาย ทำการเรียกโปรแกรมย่อย ที่อยู่ในตำแหน่งที่ถูกชี้โปรแกรมเคาเตอร์จะเพิ่มค่าขึ้นสามครั้ง เพื่อเป็นแอดเดรสของคำสั่งต่อไป และใส่ผลลัพธ์ 16บิทลงบนสแตคจากนั้น

สแตคพอน์ยเตอร์จะเพิ่มค่าสองครั้ง แล้วโหลดค่าโปรแกรมเคาเตอร์แปดบิทสูงและแปดบิทต่ำตามลำดับเป็นไบท์ที่สองและสามของคำสั่ง LCALL การexecuteโปรแกรม จะต่อเนื่องกับแอดเดรสนี้ โปรแกรมย่อยอาจจะริ่มต้นที่แอดเดรสใด ๆ ก็ได้ในหน่วยความจำโปรแกรมและไม่มีผลต่อแฟลก

ENCODE :	0 0 0 1	0 0 1 0	addr15-addr8	addr7 - addr0

LJMP addr16

<u>ความหมาย</u> ทำการกระโดดไปยังแอดเดรสที่ถูกชี้อย่างไม่มีเงื่อนไข โดยการโหลดค่าใส่ไบท์สูงและไบท์ต่ำของโปรแกรมเคาเตอร์ด้วยข้อมูลในไบท์ที่สองและสามของคำสั่งตามลำดับ แอดเดรสปลายทางจะเป็นค่าแห่งใดๆ ในหกสิบสี่เค.ของหน่วยความจำโปรแกรมคำสั่งนี้ไม่มีผลต่อแฟลก

ENCODE :	0 0 0 0	0 0 1 0	addr15-addr8	addr7 - addr0

MOV <dest-byte>,<src-byte>

<u>ความหมาย</u> ย้ายค่าในโอเปอแรนด์ที่สอง มาไว้ในโอเปอแรนด์แรกและไม่มีผลต่อ<src-byte>และรีจิสเตอร์อื่น ๆ

MOV A,Rn

<u>ความหมาย</u> โหลดค่าในRn มาไว้ในแอคคิวมูเลเตอร์

ENCODE :	1 1 1 0	1 r r r

MOV A,direct

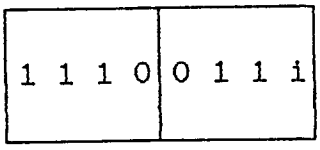
<u>ความหมาย</u> โหลดค่าในแอดเดรสที่กำหนดมาไว้ในแอคคิวมูเลเตอร์

ENCODE :	1 1 1 0	0 1 0 1	direct address

MOV A,@Ri

ความหมาย โหลดค่าในแอดเดรสที่กำหนดด้วยR0หรือR1มาไว้ในแอคคิวมูเลเตอร์

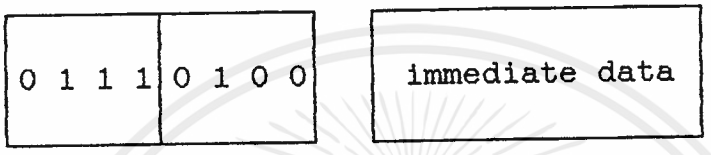
ENCODE :



MOV A,#data

ความหมาย โหลดข้อมูลที่เข้ามากับคำสั่ง ไว้ในแอคคิวมูเลเตอร์

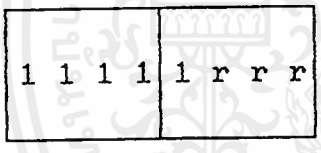
ENCODE :



MOV Rn,A

ความหมาย โหลดข้อมูลในแอคคิวมูเลเตอร์มาไว้ในRn

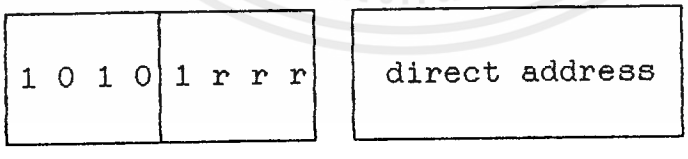
ENCODE :



MOV Rn,direct

ความหมาย โหลดข้อมูลในแอคคิวมูเลเตอร์ที่กำหนดมาไว้ในRn

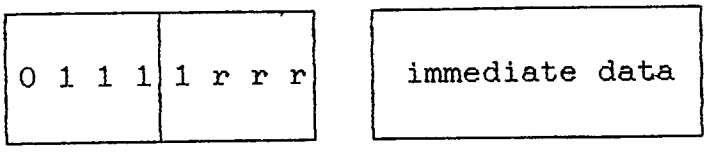
ENCODE :



MOV Rn,#data

ความหมาย โหลดข้อมูลที่เข้ามากับคำสั่ง เก็บไว้ในRn

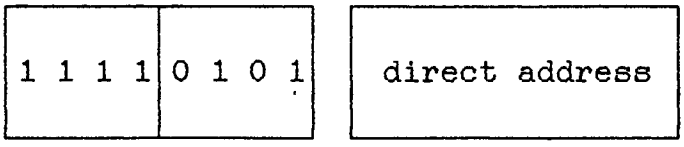
ENCODE :



MOV direct,A

ความหมาย โหลดข้อมูลในแอคคิวมูเลเตอร์มาเก็บไว้ในแอดเดรสที่กำหนด

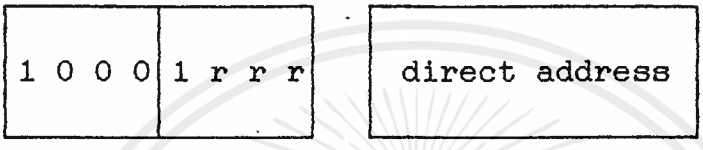
ENCODE :



MOV direct,Rn

ความหมาย โหลดค่า Rnมาเก็บไว้ยังแอดเดรสที่กำหนด

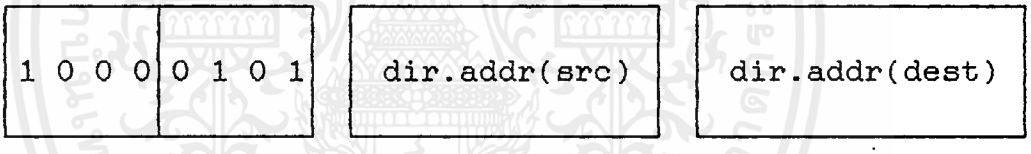
ENCODE :



MOV direct,direct

ความหมาย โหลดข้อมูลจากแอดเดรสหนึ่งมาเก็บไว้ในอีกแอดเดรสหนึ่ง

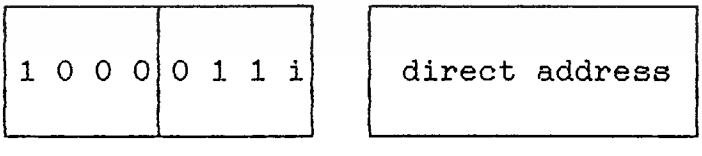
ENCODE :



MOV direct,@Ri

ความหมาย โหลดข้อมูลจากแอดเดรสที่กำหนดโดยRo หรือ R1มาเก็บไว้ในแอดเดรสที่กำหนด

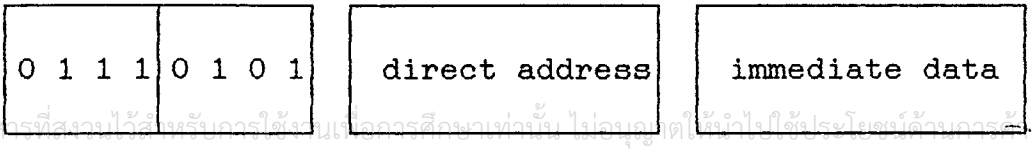
ENCODE :



MOV direct,#data

ความหมาย โหลดข้อมูลที่เข้ามากับคำสั่ง มาเก็บไว้ในแอดเดรสที่กำหนด

ENCODE :



MOV @Ri,A

ความหมาย โหลดข้อมูลในแอคคิวมูเลเตอร์มาเก็บไว้ในแอดเดรสที่กำหนดโดย Ro หรือ R1

ENCODE :

1 1 1 1	0 1 1 i

MOV @Ri,direct

ความหมาย โหลดข้อมูลจากแอดเดรสที่กำหนด มาเก็บไว้ในแอดเดรสที่กำหนด โดย Roหรือ R1

ENCODE :

1 0 1 0	0 1 1 i	direct address

MOV @Ri,#data

ความหมาย โหลดข้อมูลที่เข้ามากับคำสั่งมา เก็บไว้ในแอดเดรสที่กำหนดโดย Roหรือ R1

ENCODE :

0 1 1 1	1 r r r	immediate data

MOV <dest-bit>,<src-bit>

ความหมาย โหลดค่าหนึ่งบิทในโอเปอแรนด์ที่สองมาไว้ในโอเปอแรนด์แรก คำสั่งนี้ไม่มีผลต่อแฟลกหรือรีจิสเตอร์ใดๆ

MOV C,bit

ความหมาย โหลดค่าหนึ่งบิทที่ต้องการ มาเก็บไว้ในc-แฟลก

ENCODE :

1 0 1 0	0 0 1 0	bit address

MOV bit,C

ความหมาย โหลดค่าจาก c-แฟลก มาเก็บไว้ในแอดเดรสบิทที่ต้องการ

ENCODE :

1 0 0 1	0 0 1 0	bit address

MOV DPTR,#data16

ความหมาย ทำการโหลด DPTRด้วยข้อมูลสิบหกบิท โดยข้อมูลจะโหลดเข้าไปในไบท์ที่สองและสามของคำสั่ง โดยในไบท์ที่สอง เก็บแปดบิทสูงและไบท์สาม เก็บแปดบิทต่ำ คำสั่งนี้ไม่มีผลต่อแฟลก

ENCODE :

1 0 0 1	0 0 0 0	immed.data15-8	immed.data7-0

MOVC A,@A+<base-reg)

ความหมาย ทำการโหลดค่าแอคคิวมูเลเตอร์ด้วยรหัสแปดบิทหรือค่าคงที่จากหน่วยความจำโปรแกรม แอดเดรสดังกล่าวเป็นผลบวกของข้อมูลแปดบิทแบบไม่คิดเครื่องหมายในแอคคิวมูเลเตอร์กับข้อมูลในbase-regแบบสิบหกบิท ซึ่งอาจเป็นDPTR หรือPCก็ได้ ถ้าเป็น พีซี จะทำการเพิ่มค่าเพื่อไปยังแอดเดรสของ คำสั่งที่ตามมาก่อนที่จะบวกกับแอคคิวมูเลเตอร์ มิฉะนั้นbase-regจะไม่เปลี่ยนแปลงการบวก แบบสิบหกบิทจะเกิดตัวทดออกจากแปดบิทต่ำและอาจจะผ่านเข้าไปยังบิทสูงได้คำสั่งจะไม่มีผลต่อแฟลก

MOVC A,@A+DPTR

ความหมาย ทำการบวกข้อมูลในหน่วยความจำที่กำหนดตำแหน่งโดยแอคคิวมูเลเตอร์และDPTRแบบคิดตัวทดด้วย นำผลลัพธ์ที่ได้เก็บไว้ในแอคคิวมูเลเตอร์

ENCODE :

1 0 0 1	0 0 1 1

MOVC A,@A+PC

ความหมาย ทำการบวกข้อมูลในหน่วยความจำที่กำหนดตำแหน่ง โดยแอคคิวมูเลเตอร์และPC แบบคิดตัวทดด้วย นำผลลัพธ์ที่ได้เก็บไว้ในแอคคิวมูเลเตอร์

ENCODE :

1	0	0	0	0	0	1	1

MOVX <dest-byte>,<src-byte>

ความหมาย ทำการย้ายข้อมูลระหว่างแอคคิวมูเลเตอร์ และข้อมูลแปดบิทจากหน่วยความจำข้อมูลภายนอก สังเกตจาก "x"ที่ต่อท้ายMOV โดยมีคำสั่งอยู่สองชนิด คือ ย้ายค่าแปดบิทหรือสิบหกบิทที่อ้างแอดเดรสแบบdirectไปยังแรมภายนอก

ชนิดแรก ข้อมูลในR0หรือR1ในรีจิสเตอร์ แบงก์ จะให้addr 8 bitmultiplexed กับข้อมูลบนP0ค่าแปดบิทนี้เพียงพอสำหรับการdecode การขยายI/O ภายนอกหรือสำหรับแรมขนาดเล็ก ถ้าพื้นที่ค่อนข้างใหญ่ขาพอร์ทเอาท์พุทจะใช้ต่อไปยังเอาท์พุทของแอดเดรสบิทสูง ขาเหล่านี้จะถูกควบคุมด้วยคำสั่งoutputที่มาก่อนคำสั่งmovx

ชนิดที่สองDPTRจะสร้างแอดเดรสสิบหกบิท P2จะเอาท์แอดเดรสแปดบิทสูง ในขณะที่P0 multiplexed address แปดบิทต่ำกับคาค้า P2(sfr)จะเก็บค่าไว้ก่อน ขณะที่P2(o/p buffer)จะปล่อยข้อมูลของDPH ลักษณะนี้จะเร็วและมีประสิทธภาพเมื่อจัดการกับข้อมูลที่ใหญ่มาก และไม่ต้องเพิ่มคำสั่งในการตั้งเอาท์พุทพอร์ท

MOVX A,@Ri

ความหมาย โหลดค่าจากหน่วยความจำข้อมูลภายนอก โดยกำหนดแอดเดรสผ่านR0 หรือ R1 เข้ามาเก็บไว้ในแอคคิวมูเลเตอร์

ENCODE :

1	1	1	0	0	0	1	1

MOVX A,@DPTR

ความหมาย โหลดค่าจากหน่วยความจำข้อมูลภายนอก โดยกำหนดแอดเดรสผ่านDPTR เข้ามาเก็บไว้ในแอคคิวมูเลเตอร์

ENCODE :

1	1	1	0	0	0	0	0

รายละเอียดและการใช้งานชุดไมโครคอน-โทรลเลอร์ PC-SB 31

การติดตั้งหน่วยความจำให้กับ PC-SB31

เนื่องจาก PC-SB31 ถูกสร้างมาให้เป็นอิสระในการเลือกใช้หน่วยความจำได้หลายขนาดทั้ง EPROM และ RAM รวมทั้งตำแหน่งแอดเดรสของหน่วยความจำผู้ใช้ก็ยังสามารถกำหนดได้ตามต้องการ ซึ่งทั้งหมดนี้ขึ้นอยู่กับการใส่ตำแหน่งของ JUMPER ต่างๆให้ถูกต้อง ซึ่ง U3 และ U4 จะถูกควบคุมด้วย JUMPER ซึ่งมีรายละเอียดดังนี้ :-

U3

JP3 เลือกเบอร์ของชิปหน่วยความจำที่ใส่อยู่บน U3 (2764 , 27256 , 27512 6264 , 62556)

JP4 สำหรับเลือกว่าจะให้หน่วยความจำที่ U3 เป็น DATA MEMORY หรือ CODE MEMORY หรือเป็นทั้ง CODE และ DATA MEMORY

JP7 เลือกตำแหน่งเริ่มต้นและขนาดของหน่วยความจำ U3

U4

JP5 เลือกเบอร์ของชิปหน่วยความจำที่ใส่อยู่บน U4 (27256 , 6116 , 6264 , 62256)

JP6 เลือกลักษณะการทำงานของ U4 ว่าจะให้เป็น DATA MEMORY หรือ CODE MEMORY หรือเป็นทั้ง DATA และ CODE MEMORY

JP8 เลือกตำแหน่งเริ่มต้นและขนาดของหน่วยความจำ U4

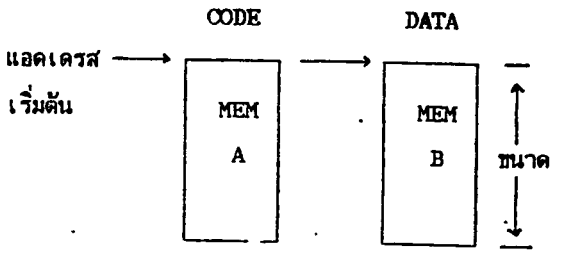
JP9 เลือกว่าจะอนุญาตให้มีการใช้ I/O พอร์ต ภายนอกอีกหรือไม่ ถ้าไม่มีพอร์ตภายนอก U4 จะมีขนาดสูงสุดได้ถึง 32 KB

ข้อจำกัดในการติดตั้งหน่วยความจำแบบแยก DATA และ PROGRAM ออกจากกัน

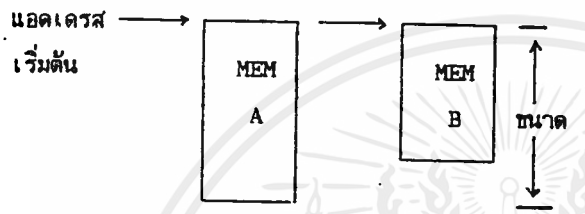
เมื่อดูจากวงจรของ PC-SB31 จะเห็นว่า ส่วนของ JP7 และ JP8 ด้านหนึ่งจะถูกต่อถึงกันในตำแหน่งที่ตรงกัน (เช่น 0000 ของ JP7 จะต่อกับ 0000 ของ JP8) จึงทำให้เกิดการซ้อน PAGE ขึ้นในกรณีที่ขนาดของหน่วยความจำ U3 และ U4 มีขนาดไม่เท่ากัน ถึงแม้ว่าผู้ใช้จะใส่หน่วยความจำบน U3 และ U4 มีขนาดไม่เท่ากัน โปรแกรมก็ยังทำงานได้ตามปกติแต่ผู้ใช้ต้องระลึกอยู่เสมอว่าหน่วยความจำแต่ละตัวเริ่มต้นที่ใดและควรสิ้นสุดที่ใด

ตัวอย่างเช่น ที่ U3 ใส่หน่วยความจำ 8 KB และที่ U4 ใส่หน่วยความจำขนาด 2 KB ที่ตำแหน่งของหน่วยความจำน้อย (U4) จะเกิดการซ้อนตัวเอง เช่น ในตำแหน่ง 2000H และตำแหน่ง 2800H จะเป็นตำแหน่งเดียวกัน (เกิดการซ้อน)

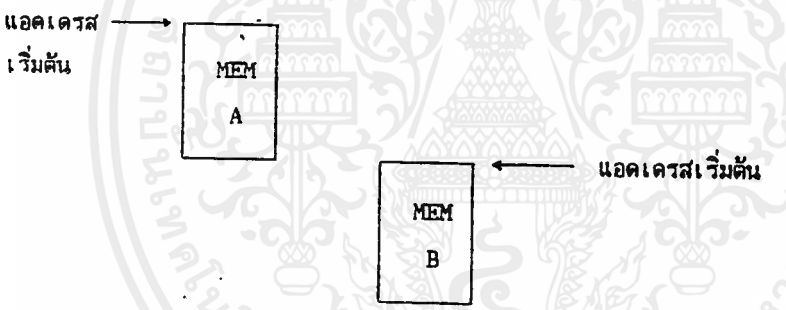
ลักษณะของการเลือกขนาดของหน่วยความจำชนิดแยก DATA และ CODE MEMORY



รูปที่ 1. (A) ไม่เกิดการซ้อน PAGE ของตัวเอง เมื่อ MEM A = MEMB



รูปที่ 1. (B) หน่วยความจำตำแหน่ง B จะเกิดการซ้อนตัวเอง

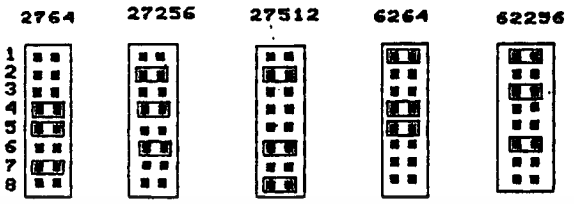


รูปที่ 1. (C) จุดเริ่มต้นต่อกันจะไม่เกิดการซ้อน

รูปที่ 1. ลักษณะการจัดหน่วยความจำ

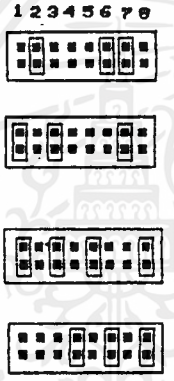
การใส่ JUMPER ให้ถูกต้องตามเบอร์ของชิปหน่วยความจำ

JP3 (สำหรับ U3)



รูปที่ 2. การใส่ JUMPER JP3

JP5 (สำหรับ U4)



รูปที่ 3. การใส่ JUMPER JP5

JP4 & JP6 (เลือกลักษณะของ U3 และ U4)

DATA = DATA MEMORY

CODE = CODE MEMORY

COMBINE = DATA AND CODE MEMORY

รูปที่ 4. การใส่ JUMPER JP4 & JP6

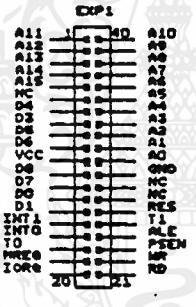
PC-SB31 VS ET DEBUGGER-31

การพัฒนาระบบอาจจะไม่สะดวกถ้าหากว่าขาดเครื่องมือพัฒนาระบบการทำงานให้สำเร็จลุล่วงอาจจะกินเวลามาก ฉะนั้น PC-SB31 จึงมีซอฟต์แวร์ในการพัฒนาโปรแกรม ชื่อว่า ET DEBUGGER-31 ซึ่งโปรแกรมส่วนนี้ถูกบรรจุอยู่ใน EPROM 2764 ซึ่งนำมาใช้ร่วมกับบอร์ด PC-SB31 ได้ทันที โดยผู้ใช้ต้องจัดหน่วยความจำให้เหมาะสม ซึ่งมีรายละเอียดอยู่ในคู่มือ ET DEBUGGER-31 หรืออาจจะใช้การใส่ JUMPER ตามแบบในรูปที่ 7.

รายละเอียดเกี่ยวกับ CONNECTER

PC-SB31 ได้ถูกออกแบบมาให้ใช้ได้กับบอร์ดอินเตอร์เฟสต่างๆของบริษัทอีทีทีโดยเฉพาะอย่างยิ่งบอร์ดประเภทอินพุท/เอ้าท์พุท ดังมีรายละเอียดดังต่อไปนี้

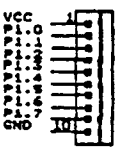
EXP1 เป็น CONNECTER ขนาด 40 PIN ซึ่งมีสัญญาณที่ขาต่างๆคล้ายกับสัญญาณที่ต่อจาก CPU Z80 ทั้งนี้เพื่อสนับสนุนบอร์ดต่างๆเช่น 72IO , RTC เป็นต้น แต่เนื่องจาก PC-SB31 ใช้ CPU คนละตระกูลกับ Z80 จึงมีสัญญาณบางสัญญาณไม่ตรงกัน ดังจะดูได้ในรูปที่ 9.



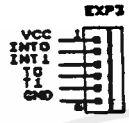
รูปที่ 9. Z80 CONNECTER (COMPATIBLE)

หมายเหตุ สำหรับผู้ที่ต่อกับ 72IO ให้ทำการดัดแปลงสัญญาณรีเซทบนบอร์ด 72IO ให้เป็นแบบ POWER ON RESET (ดูคู่มือ 72IO)

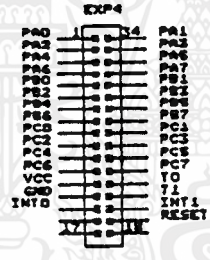
EXP2 เป็น INPUT/OUTPUT พอร์ตอิสระขนาด 8 บิท โดยมีรายละเอียดดังนี้



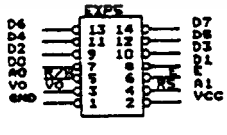
EXP3 เป็นขาของ INTERUPT และ TIMER/COUNTER ของตัว CPU โดยมีรายละเอียดดังนี้



EXP4 เป็น CONNECTER ขนาด 34 ขา ซึ่งมีการวางตำแหน่งของขาตรงกับบอร์ด 72IO ซึ่งสามารถต่อกับอุปกรณ์ I/O ต่างๆ เช่น SSRAC , ET-AD เป็นต้น



EXP5 เป็น CONNECTER ขนาด 14 ขา ที่ใช้ต่อกับ LCD ได้โดยตรง โดยมีรายละเอียดดังนี้



การติดต่อและควบคุม LCD

ต่อ LCD เข้าทาง EXP5 โดยที่ EXP5 ถูก DECODE ไว้ที่พอร์ตหมายเลขดังนี้ :-

- E0C0H = พอร์ตเกี่ยวกับการเขียนคำสั่งไปยัง LCD (RS , R/W = 00)
- EXC1H = พอร์ตอ่าน BUSY FLAG และแอดเดรสของ CURSOR (RS , R/W = 01)
- E0C2H = พอร์ตเขียนข้อมูลไปยัง DD RAM หรือ CG RAM (RS , R/W = 10)
- E0C3H = พอร์ตอ่านข้อมูลจาก DD RAM หรือ CG RAM (RS , R/W = 11)

* ดูตัวอย่างโปรแกรมในหน้าถัดไปรายละเอียดเกี่ยวกับ LCD หาดูได้จากคู่มือ LCD หรือติดต่อ บริษัท อีทีที จำกัด *

รายละเอียด และการประยุกต์ใช้งาน

DOT MATRIX LCD MODULE

DOT MATRIX LCD MODULE

อุปกรณ์ในปัจจุบันนี้ในส่วนแสดงผลนั้นจะใช้ LCD เสียเป็นส่วนใหญ่ไม่ว่าจะเป็นเครื่องเล่น VEDIO, เครื่องถ่ายเอกสาร, เครื่องมือวัดคุมต่างๆ, เครื่องคอมพิวเตอร์ เราพอจะแบ่ง DOT MATRIX LCD MODULE นี้ออกได้เป็นพวกๆดังนี้ :-

1. CHARACTER LCD MODULE
2. GRAPHIC LCD MODULE
3. SEGMENT DISPLSY TYPE LCD MODULE

โดยในแต่ละแบบนี้ก็จะมีส่วนประกอบใหญ่ๆแบ่งได้เป็น

1. DOT MATRIX LCD เป็นตัวแสดงผลให้เรามองเห็นในลักษณะการปิดและเปิดตัวเองกับแสงก็คือ ส่วนของที่เป็นตัวกระจกบรรจุผลึก

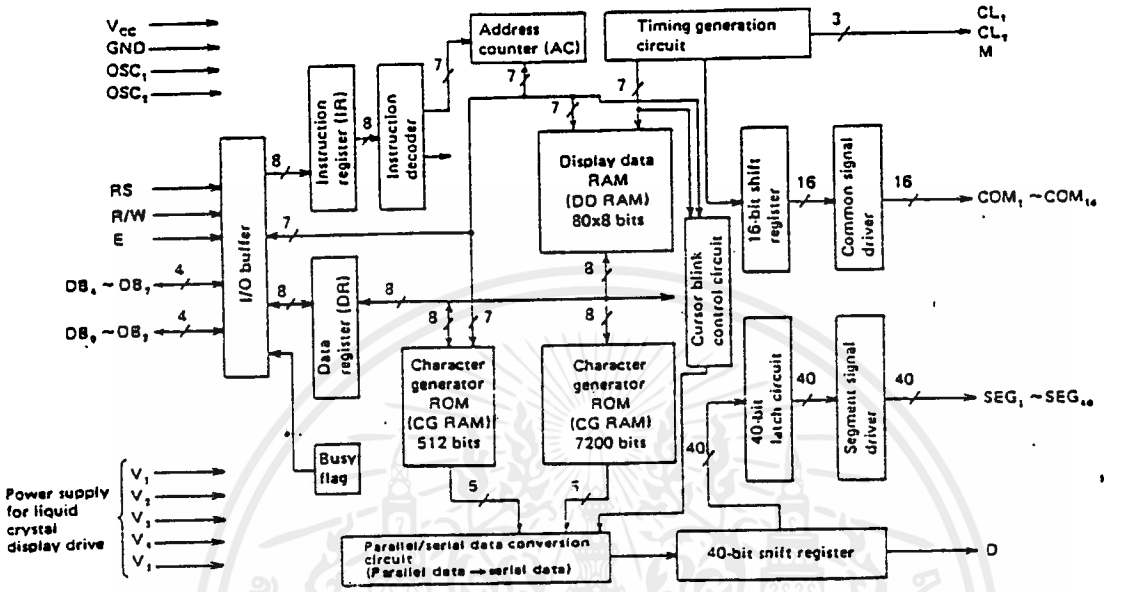
2. DRIVER เป็นตัวรับสัญญาณจากตัวควบคุมมาขับผลึก LCD อีกทีหนึ่งโดยมีเบอร์ที่นิยมใช้ใน LCD MODULE เช่น HD44100H, MSM5259

3. CONTROLLER เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาและจัดกวรควบคุม LCD MODULE ให้ทำงานแสดงผลต่างๆเช่น การลบจอภาพ, การเกิดตัวอักษร, เป็นต้น โดยมีเบอร์ IC ที่นิยมใช้กันคือ HD44780 ซึ่งจะใช้ในแบบ CHARACTER LCD MODULE เป็นส่วนใหญ่ เบอร์ IC HD61830 จะใช้ในแบบ GRAPHIC LCD MODULE

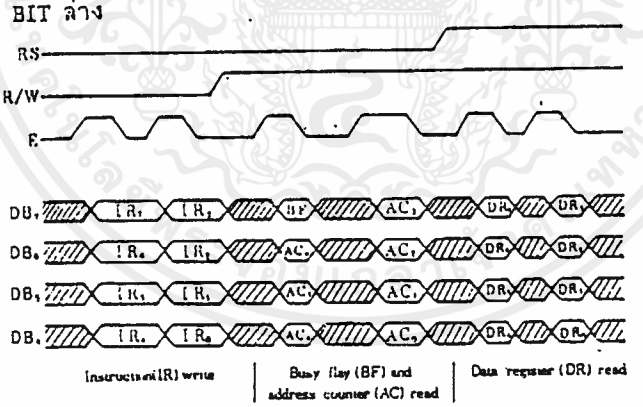
ในการศึกษาการทำงานและใช้งาน LCD MODULE นั้นไม่ใช่เรื่องยากเลยถ้าเราสามารถทำความเข้าใจในส่วนของ CONTROLLER ได้ก็เพียงพอแล้วและโดยมาก LCD MODULE ในแต่ละบริษัทแล้วจะใช้ตัว CONTROLLER ที่มีหลักการทำงานเหมือนๆกันเป็นส่วนใหญ่และใน LCD MODULE แต่ละขนาดจำนวนตัวอักษรหรือจำนวนบรรทัดก็มีหลักการทำงานแบบเดียวกันทั้งหมด IC ที่นิยมมากที่สุดตัวหนึ่งที่เป็น CONTROLLER LCD ก็คือ เบอร์ HD44780 โดยรูปแบบการทำงานของมันได้เป็นมาตรฐานให้กับ CONTROLLER LCD ตัวอื่นๆด้วย

HD44780 เป็นไอซี LSI ตัวหนึ่งใช้ควบคุม LCD โดยแสดงผลในรูปตัวอักษรหรือสัญลักษณ์ต่างๆด้วยมันเองสามารถต่อใช้งานแบบ 4 BIT หรือ 8 BIT ก็ได้ โดยถ้าเราต่อแบบ 4 BIT จะต่อใช้งานที่ DB7-DB4

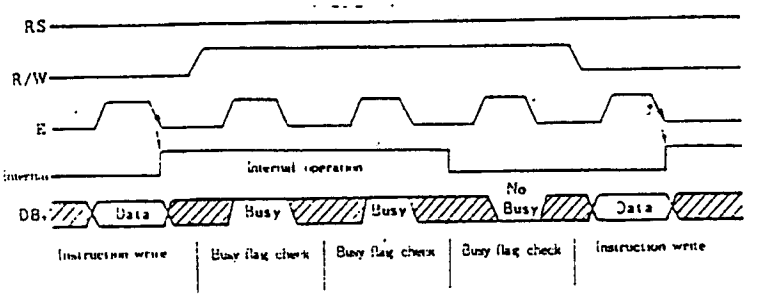
Block diagram of HD44780 interior



เท่านั้นโดยข้อมูลครั้งแรกที่ส่งนั้น HD44780 จะถือเป็นข้อมูล 4 BIT บน และข้อมูลที่ส่งต่อมานั้น เป็นข้อมูล 4 BIT ล่าง

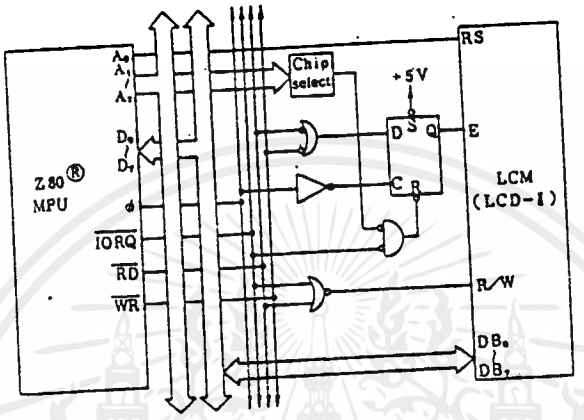


4-bit data transfer example

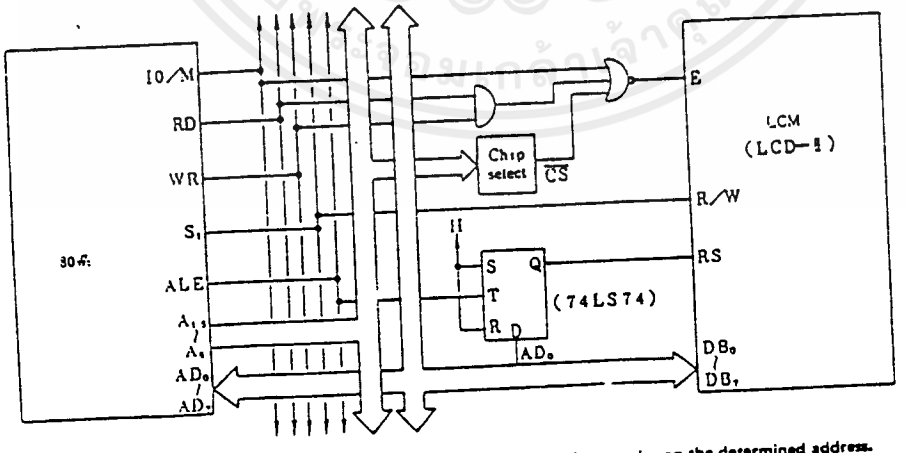


Example of busy flag check timing sequence

Example of interfacing to Z80 MPU



เราสามารถต่อ LCD MODULE (HD44780 เป็น CONTROLER) เข้ากับระบบไมโครได้หลายรูปแบบดังรูป



Example of connection with LCM being used as a part of memories on the determined address.

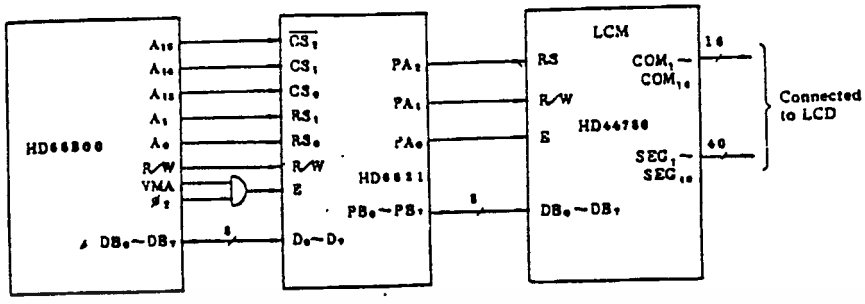
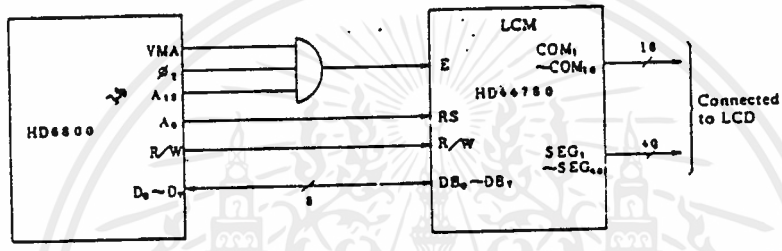
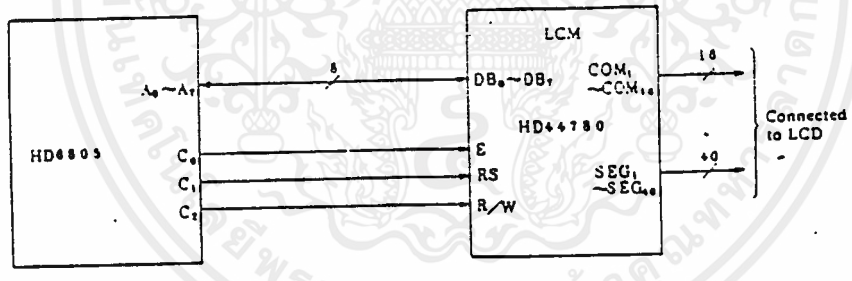


Fig. 4 Example of Interface to HD68B00 using PIA (HD68B21)

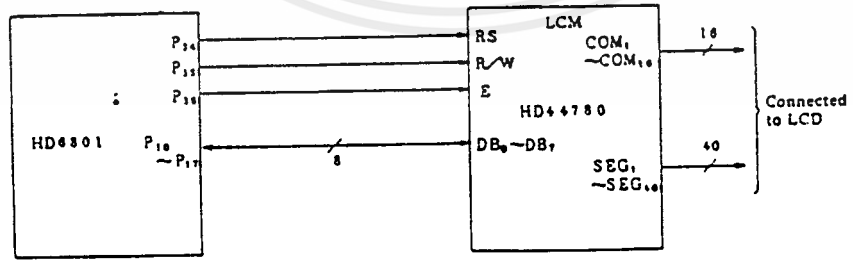
Connecting directly to the 8 bit MPU bus line



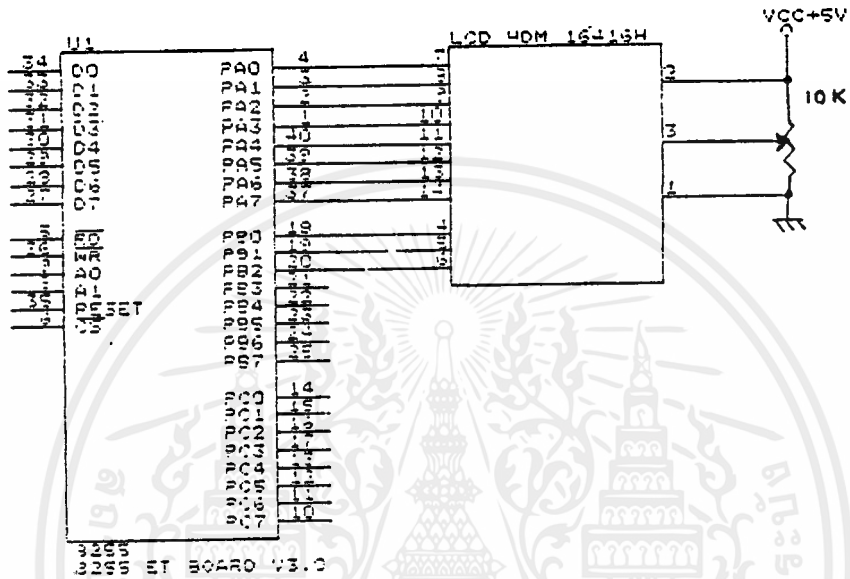
Example of interfacing to the HD6805



Example of interfacing to the HD6301



ตัวอย่างการต่อใช้งานจริงกับ ET-BOARD V3.0



จากวงจรเป็นการต่อ 8255 ให้เข้าใช้กับ LCD โดยเราจะจำลองสัญญาณต่างๆขึ้นมาโดยการใช้ PORT A และ PORT B โดย PORT A นั้นเราให้เป็น DATA PORT และ PORT B นั้นเราให้เป็นสัญญาณควบคุมไปใช้

เมื่อเราเริ่มเปิดไฟป้อนให้ HD44780 นั้นก็จะทำการ RESET ตัวมันเองโดยจะใช้เวลาประมาณ 10 ms หลังจากไฟ VDD มีค่า 4.5 VOLT แล้ว โดยจะ SET ตัวเองดังนี้ :-

1. DISPLAY CLEAR จะทำการลบข้อมูลจอภาพ LCD

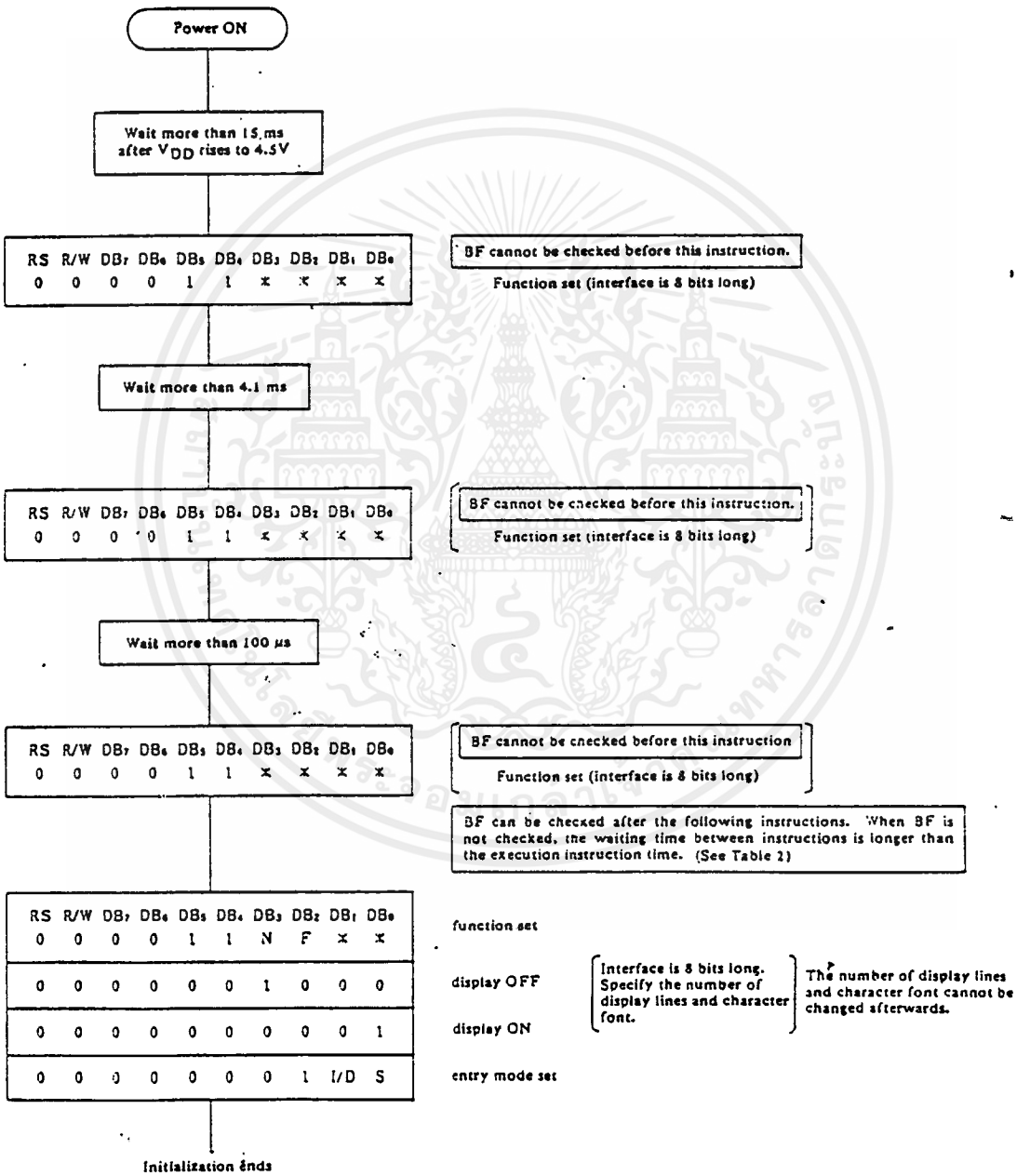
2. FUNCTION SET โดยจะ SET ค่าภายใน
   - DL = 1 : เป็นการ SET ให้การติดต่อแบบ 8 BIT
   - N = 0 : SET เป็น 1 บรรทัดการแสดงผล
   - F = 0 : 5X7 DOT ต่อหนึ่งต่ออักษร

3. DISPLAY ON/OFF
   - D = 0 : DISPLAY OFF
   - C = 0 : CURSOR OFF
   - B = 0 : BLINK OFF

4. ENTRY MODE SET I/D = 1 : +1 (เพิ่มค่า COUNTER ขึ้น 1)

S = 0 : NO SHIFT

เมื่อเราเริ่มเปิดเครื่องทำงานแล้วก็จะต้องส่งคำสั่งควบคุมให้มันเริ่มทำงานดังตาราง



ตารางคำสั่ง HD44780

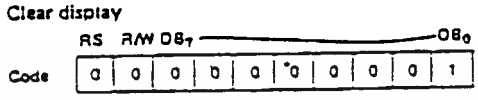
Instruction	Code										Description	Execution time (when fosc is 250 kHz) Note 1	Execution time (when fosc is 160 kHz) Note 2
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Clear display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	82 μs – 1.64 ms	120 μs – 4.9 ms
Return home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0). Also returns the display being shifted to the original position. DD RAM contents remain unchanged.	40 μs – 1.6 ms	120 μs – 4.8 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40 μs	120 μs
Display ON/ OFF control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40 μs	120 μs
Cursor and display shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RAM contents	40 μs	120 μs
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL) number of display lines (L) and character font (F).	40 μs	120 μs
Set CG RAM address.	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data is sent and received after this setting.	40 μs	120 μs
Set DD RAM address	0	0	1	A_{DD}							Sets the DD RAM address. DD RAM data is sent and received after this setting.	40 μs	120 μs
Read busy flag & address	0	1	BF	AC							Reads Busy flag (BF) indicating internal operation is being performed and reads address counter contents.	1 μs	1 μs
Write data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40 μs	120 μs
Read data to CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40 μs	120 μs
	I/D = 1: Increment (+1) I/D = 0: Decrement (−1) S = 1: Accompanies display shift. S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right. R/L = 0: Shift to the left. DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5 x 10 dots F = 0: 5 x 7 dots BF = 1: Internally operating BF = 0: Can accept instruction										DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM address A_{DD}: DD RAM address Corresponds to cursor address. AC: Address counter used for both of DD and CG RAM address.	Execution time changes when frequency changes. (Example) When fosc is 270 kHz: $40\ \mu s \times \frac{250}{270} = 37\ \mu s$	

*No effect

Notes 1. Applied to models driven by 1/8 duty or 1/11 duty.
2. Applied to models driven by 1/16 duty.

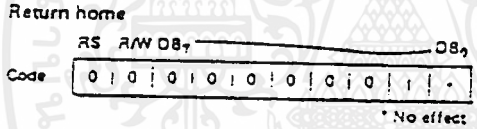
รายละเอียดของคำสั่ง HD44780

1. CLEAR DISPLAY



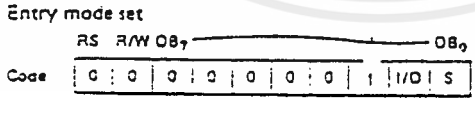
คำสั่งนี้จะเป็นการเขียนช่องว่างหรือ SPACE (ASCII 20H) เข้าไปใน DD RAM ทั้งหมดและทำการ SET DD RAM ADDRESSER เป็นศูนย์ ตัว CURSOR จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจอภาพ SET I/D = 1, S ไม่มีการเปลี่ยน

2. RETURN HOME



คำสั่งนี้จะทำการ SET DD RAM ADDRESSER เป็นศูนย์ ตัว CURSOR จะกลับไปอยู่ตำแหน่งบนสุดซ้ายมือของจอภาพข้อมูลในจอภาพไม่เปลี่ยน

3. ENTRY MODE SET



BIT I/D : โดยจะเป็นตัวกำหนดให้ว่าเมื่อเขียนหรืออ่านข้อมูลแล้วจะทำให้ DD RAM ADDRESS เพิ่มขึ้นหนึ่งหรือลดลงหนึ่งโดย 1 = เพิ่ม 0 = ลดลงหนึ่ง

BIT S : เป็นตัวกำหนดแสดงผลโดยถ้า S = 1 จะเป็นการใส่ข้อมูลแล้วตัว CURSOR อยู่กับที่ข้อมูลจะถูกดันไปทางซ้าย ถ้า S = 0 ข้อมูลจะอยู่กับที่ตัว CURSOR จะถูกดันไปทางขวามือ

4. DISPLAY ON/OFF CONTROL

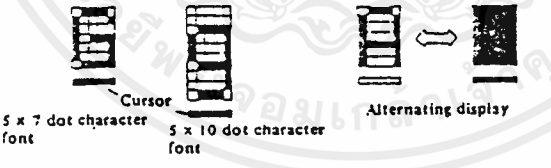
Display ON/OFF control

	RS	R/W	DB_7						DB_0
Code	0	0	0	0	0	0	1	D	C	B

BIT D : เป็น BIT ให้เปิดปิดหน้าจอภาพโดยถ้า D = 1 จะ ON และ D = 0 จะ OFF

BIT C : จะให้แสดง CURSOR ให้ BIT C = 1 และถ้าไม่ต้องการแสดง CURSOR BIT C = 0 โดยตัว CURSOR จะอยู่ที่ LINE ที่ 8 ในแบบ 5X7 DOT และจะอยู่ LINE ที่ 11 ในแบบ 5X10 DOT

BIT B : เป็น BIT SET การกระพริบของ CURSOR โดย B = 1 มีการกระพริบ B = 0 ไม่มีการกระพริบ โดยมีระยะเวลาการกระพริบประมาณ 379.2 ms



(a) Cursor Display Example (b) Blink Display Example

5. CURSOR OR DISPLAY SHIFT

Cursor or display shift

	RS	R/W	DB_7						DB_0
Code	0	0	0	0	0	1	S/C	R/L	*	*

* No effect

เป็นคำสั่งกำหนดให้ตำแหน่ง CURSOR หรือข้อมูลไปเกิดทางซ้ายหรือขวา โดยไม่ต้องใช้คำสั่งเขียนหรืออ่าน โดย

S/C	R/L	
0	0	ทำการย้าย CURSOR ไปจากตำแหน่งเดิมไปซ้ายมือ 1 ตำแหน่ง
0	1	ทำการย้าย CURSOR ไปจากตำแหน่งเดิมไปขวามือ 1 ตำแหน่ง
1	0	เป็นการดันตัวอักษรที่เกิดไปทางซ้าย
1	1	เป็นการดันตัวอักษรที่เกิดไปทางขวามือ

6. FUNCTION SET

Function set

	RS	R/W	DB_7						DB_0
Code	0	0	0	0	1	DL	N	F	*	*

* No effect

BIT DL : เป็นการ SET การติดต่อว่าจะให้เป็นแบบ 8 BIT หรือ 4 BIT โดยถ้าต้องการติดต่อ 4 BIT DL = 0 และ 8 BIT DL = 1

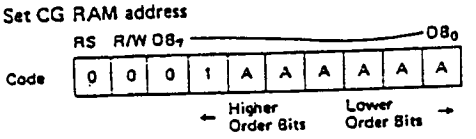
N : เป็นการ SET บรรทัดการแสดงผล N = 0 แสดง 1 บรรทัด N = 1 แสดง 2 บรรทัด ในกรณีมากกว่า 2 บรรทัด ก็ให้ SET N = 1

F : เป็นการ SET ขนาด DOT การแสดงผล 5X7 หรือ 5X10 โดย F = 0 เป็นแบบ 5X7 และ F = 1 เป็นแบบ 5X10

N	F	No. of display lines	Character font	Duty factor	Remarks
0	0	1	5 x 7 dots	1/8	
0	1	1	5 x 10 dots	1/11	
1	*	2	5 x 7 dots	1/16	Cannot display 2 lines with 5 x 10 dot character font.

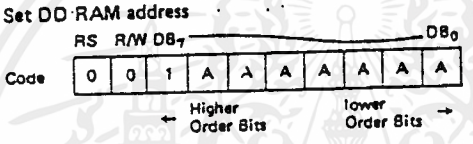
* No effect

7. SET CG RAM ADDRESS



ใน HD44780 นั้นจะมีหน่วยความจำอยู่ 2 ชุด คือ DISPLAY DATA RAM (DD RAM) จำนวน 80X8 BIT และ CHARACTER GENERATOR ROM CG RAM จำนวน 512 BIT และ 7200 BIT คำสั่งนี้จะเป็นการ SET ADDRESS ใน CG RAM โดยต้องทำการ SET ADDRESS ก่อนเขียนหรืออ่านข้อมูลจาก CG RAM ด้วย

8. SET DD RAM ADDRESS



เป็นคำสั่ง SET ค่า ADDRESS ใน DD RAM ในการเขียนหรืออ่านค่าจาก DD RAM (DD RAM คือ ส่วนที่จะแสดงผลหน้าจอ LCD) โดยจำนวน ADDRESS ที่จะเกิดขึ้นบนจอ LCD จะอยู่กับ SET ค่า N ด้วย

ถ้า N = 0 (1 บรรทัด) ADDRESS จะอยู่ 00H-4FH

ถ้า N = 1 (2 บรรทัด) ADDRESS จะอยู่ 00H-27H สำหรับบรรทัดที่ 1 และ 40H-67H สำหรับบรรทัดที่ 2

ตัวอย่างการจัด ADDRESS ของ DD RAM หน้าจอ LCD แบบ 16 ตัวอักษร 4 บรรทัด และ 20 ตัวอักษร 2 บรรทัด HDM-16416H, HDM-20216H

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	— display position
1-line	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	— DD RAM address
2-line	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	
3-line	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	
4-line	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	

HDM-16416H

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	← display position
1-line	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	← DD RAM address
2-line	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	
3-line	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27	
4-line	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	.61	62	63	64	65	66	67	

(Note) Shift display is as same as 2-line type.

9. READ BUSY FLAG AND ADDRESS

Read busy flag & address

	RS	R/W	DB_7							DB_0
Code	0	1	BF	A	A	A	A	A	A	A

← Higher Order Bits Lower Order Bits →

เป็นคำสั่งอ่านค่า BUSY FLAG ซึ่งจะเป็นตัวบอกว่าตัว HD44780 นี้อยู่ในขบวนการทำงานภายในอยู่หรืออยู่ในสภาวะพร้อมจะรับข้อมูล โดย

BF = 1 อยู่ในขบวนการทำงานภายในไม่พร้อมจะรับข้อมูลหรือคำสั่ง

BF = 0 พร้อมจะรับข้อมูลหรือคำสั่งได้

และนอกจากนี้ยังเป็นคำสั่งอ่านค่าข้อมูล ADDRESS ของ CG RAM หรือ DD RAM ด้วย

10. WRITE DATA TO CG หรือ DD RAM

Write data to CG or DD RAM

	RS	R/W	DB_7							DB_0
Code	1	0	D	D	D	D	D	D	D	D

← Higher Order Bits Lower Order Bits →

เป็นคำสั่งเขียนข้อมูลเข้าไปใน CG หรือ DD RAM โดยเมื่อเขียนข้อมูลและ ADDRESS จะเพิ่มหรือลดโดยอัตโนมัติตามคำสั่งที่ SET ใน ENTRY MODE ข้อกำหนดที่จะรู้ว่าเป็นการเขียนข้อมูลของ CG RAM หรือ DD RAM ทำได้โดยการ SET ADDRESS ของ CG RAM หรือ DD RAM อันมาก่อนจะเขียนข้อมูล

11. READ DATA FROM CG OR DD RAM

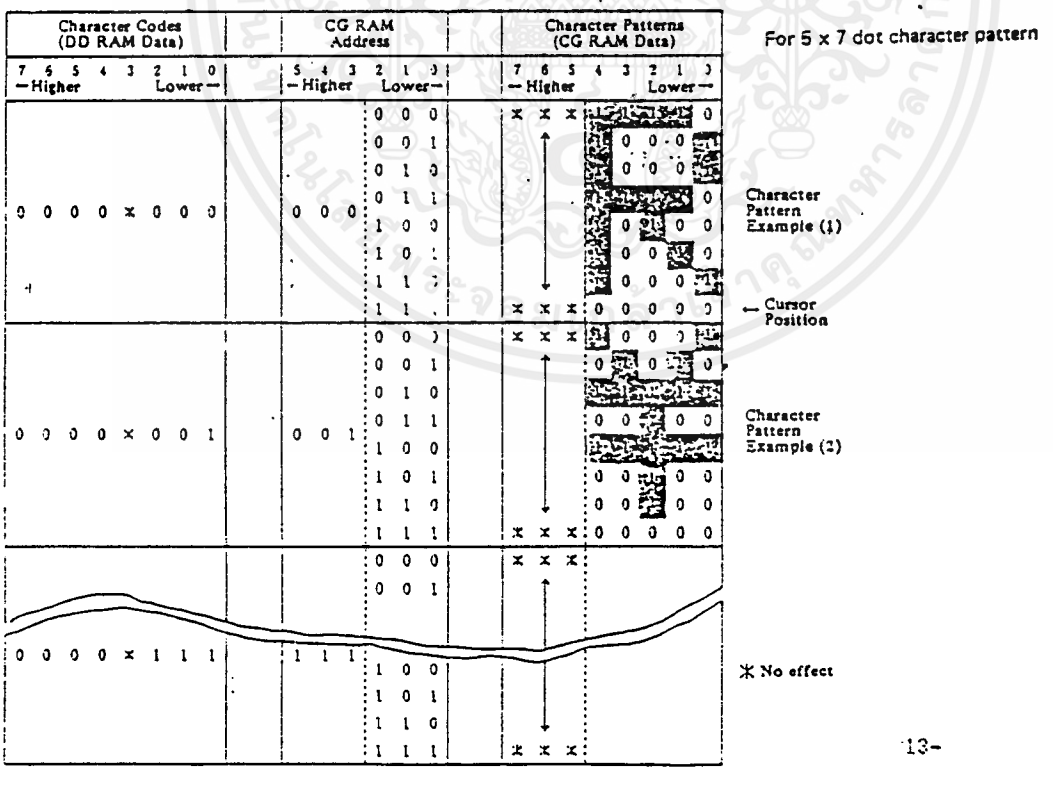
Read data from CG or DD RAM

	RS	R/W	DB_7							DB_0
Code	1	1	D	D	D	D	D	D	D	D

← Higher Order Bits — Lower Order Bits →

เป็นคำสั่งอ่านค่าข้อมูลจาก CG RAM หรือ DD RAM โดยก่อนอ่านค่าจาก DD RAM หรือ CG RAM นี้ควรจะใช้คำสั่ง SET ADDRESS ก่อนเพื่อให้รู้ว่าข้อมูลที่อ่านได้นั้นเป็น DD หรือ CG RAM

จากตารางการทำงานจะเห็นว่าการใช้งาน LCD MODULE นั้นง่ายเพียงแต่เราส่งคำสั่งเริ่มแรกและ SET ความต้องการในขนาดตัวอักษร, CURSOR หลังจากนั้นเราก็สามารถเขียนตัวอักษรเข้าไปใน DD RAM ตามตารางตัวอักษรที่ให้มานั้นก็จะเกิดอักษรในจอภาพ LCD เรายังสามารถกำหนดตำแหน่งตัวอักษรที่จะให้เกิดบนจอได้โดยการ SET DD RAM ADDRESS ตามตารางที่ให้มาในหัวข้อ SET DD RAM ADDRESS ขอให้ทดสอบทำความเข้าใจกับตัวโปรแกรมที่ใช้กับ ET-BOARD V3.0 นี้ที่ให้มาจะเห็นว่าจะมีส่วนเริ่มต้นก็คือ ส่วนการ INITIAL LCD เพื่อกำหนดหน้าที่การทำงานต่างๆ



ส่วนประกอบของโปรแกรม

EPLUSE จะเป็นส่วนกำเนิดสัญญาณ ENABLE SIGNAL โดยการใช้ PORT B BIT ที่ 2 กำเนิด PLUSE สัญญาณ ENABLE ขึ้น

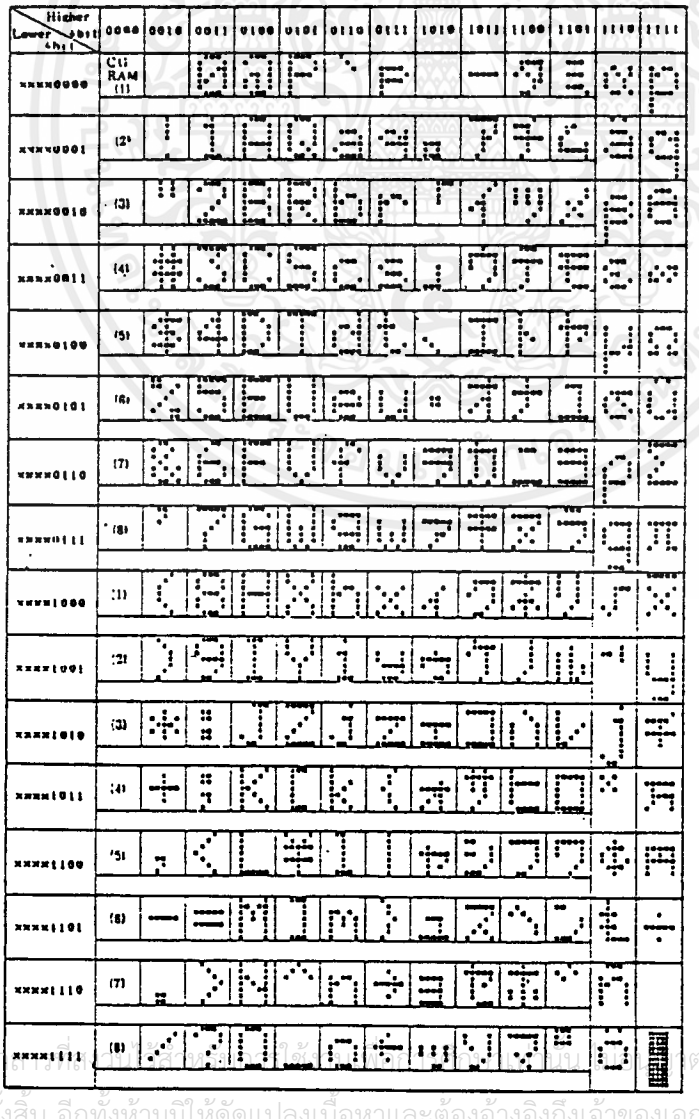
GOTO จะเป็นส่วนกำหนดตำแหน่งของส่วน DD RAM ADDRESS ที่จะเขียนข้อมูล โดยจากโปรแกรม INITIAL ที่เรา SET ไว้ เมื่อเขียนข้อมูลเข้าไปใน DD RAM แล้ว ADDRESS ของ DD RAM จะเพิ่มขึ้น 1 โดยทันที

WRBYTE เป็นส่วนเขียนข้อมูล 1 BYTE เข้าไปในตำแหน่ง ADDRESS ของ DD RAM ขณะนั้นๆ

WRLINE เป็นส่วนในการเขียนข้อมูลทีละ 1 LINE เพราะตำแหน่ง DD RAM ที่เกิดบนจอภาพ LCD นั้นแต่ละตำแหน่งจะไม่ต่อกันไปในแต่ละบรรทัด

จากตัวอย่างที่ให้จะมี LCD แบบ 20 ตัวอักษร 2 บรรทัด และ 16 ตัวอักษร 4 บรรทัด

CHARACTER FONT TABLE



NOTE: CGRAM is a CHARACTER GENERATOR RAM having a storage function of character pattern which enable to change freely by user's program.

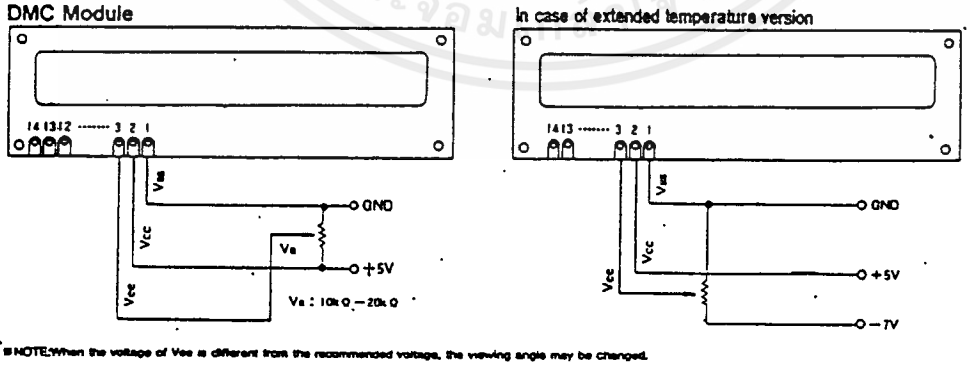
นอกจากนี้ LCD MODULE (HD44780) นี้จะยังมีส่วนหนึ่งของ CHARACTER GENERATOR ที่เราสามารถเขียนข้อมูลในการเกิดตัวอักษรขึ้นได้เอง จากตารางตัวอักษร 5X7 DOT นั้นจะเห็นว่าคือ ตำแหน่งในตาราง 00H ถึง 07H ส่วนตำแหน่ง 08H-0FH จะเป็นตำแหน่งเดียวกับ 00H-07H จะเห็นว่าจะมี CHARACTER GENERATOR 8 ตัวที่เราสามารถเขียนข้อมูลกำหนดเองได้และถ้าเป็นแบบ 5X10 DOT จะเขียนได้ 4 ตัวอักษร ซึ่งจากข้อพิเศษนี้ทำให้เราสามารถเขียนตัวอักษรสัญลักษณ์หรืออักษรภาษาไทยได้

การเขียนข้อมูล CHARACTER GENERATOR

เราสามารถเขียนข้อมูลได้โดยกำหนด ADDRESS ของ CG RAM ก่อนโดยเขียนได้ 64 ตำแหน่ง BIT 5-BIT 0 และเมื่อกำหนด ADDRESS แล้วก็จะทำการเขียนข้อมูลลงใน CG RAM โดยเป็นลักษณะ BIT ต่อ BIT บนจอ 1 ตัวอักษร คือ 5X7 DOT นั้นจะใช้ข้อมูล BIT 4 ถึง BIT 0 ต่อ 1 BYTE เท่านั้น 1 ตัวอักษรจะใช้ข้อมูล 8 BYTE ด้วยกันให้ดูจากตารางประกอบไปด้วยและเมื่อเขียนข้อมูลลงใน CG RAM แล้วเวลาเราจะใช้งานก็ให้เขียนข้อมูลใน DD RAM คือ ข้อมูลตำแหน่งในตาราง CHARACTER ที่ตำแหน่ง 00H-07H

ตัวอย่างโปรแกรมการเขียนข้อมูลตัวหนังสือภาษาไทยเป็นตัว (อ), (ท), และตัว (ี) เข้าไปใน CG RAM ตำแหน่งที่ 00H, 01H และ 02H และนำมาแสดงผลทางจอ LCD โดยใช้ 2 บรรทัดในการแสดงผล

สรุป การใช้งาน LCD MODULE นั้นที่สำคัญคือ ต้องเข้าใจในตัว CONTROLLER ของ LCD MODULE นั้น โดย CONTROLLER ทุกๆบริษัทจะมีการทำงานที่เหมือนๆกันเป็นส่วนใหญ่



ขาต่างๆในการต่อใช้งาน HD44780

1. RS (REGISTOR SELECTION) จะเป็นขาเลือก REGISTOR ภายในซึ่งมีอยู่ 2 ตัวคือ INSTRUCTION REGISTOR (IR) และ DATA REGISTOR (DR) โดยถ้าเป็น 1 จะเป็นการเลือก DATA และถ้าเป็น 0 จะเป็นการเลือก INSTRUCTION

2. R/W (READ/WRITE) เป็นตัวเลือกว่าจะเขียนหรือจะอ่านข้อมูลจากตัว IC โดยอ่านข้อมูล = 1, เขียนข้อมูล = 0

3. E (ENABLE SIGNAL) เป็นขากำหนดสภาพการรับเขียนอ่านข้อมูล

The relation between the operation and the combination of RS, R/W

RS	RW	E	OPERATION
0	0		Write instruction code
0	1		Read busy flag and address counter
1	0		Write data
1	1		Read data

When performing data and instruction code by 4 bit, transfer RS, R/W every time.

4. DB0-DB7 เป็นขารับส่งข้อมูลจากตัว IC

5. VDD ไฟเลี้ยงตัววงจร

6. VSS เป็นขา GND

7. VO เป็นขารับ VOLTAGE ในการขับ LCD ให้สว่างหรือมืด

หนังสืออ้างอิง

- สมยศ ภูงามแปลง. "MCS-51ไมโครคอนโทรลเลอร์แบบชิพเดียว" วารสารเซมิคอนดัคเตอร์อิเลคทรอนิคส์. ฉบับที่ 93,2532 หน้า 264-269
- ผศ.พิพัฒน์ เลาหสงคราม. "ไมโครคอนโทรลเลอร์ MCS-51 (1)" วิศวสารลาดกระบัง, หน้า 60-68
- ผศ.พิพัฒน์ เลาหสงคราม. "ไมโครคอนโทรลเลอร์ MCS-51 (2)" วิศวสารลาดกระบัง, หน้า 49-60
- HANDBOOK MCS-51, INTEL CORPORATION, 1985
- CURTIS D. JOHNSON "MICROPROCESSOR-BASED PROCESS CONTROL" PRETICE-HALL, 1984
- JAZZ 31 "SINGLE BOARD MICROCONTROLLER", USER MANUAL V1.1 SILA RESEARCH CO., LTD.
- อาจารย์นรินทร์ เนาวประทีป. "การใช้งาน Z-80", PHYSICS CENTER