ปริญญานิพนธ์

ตัวควบคุมลำดับโดยใช้ MCS-51

Programmable Logic Control by MCS-51

นางสาวนัยนา วาณิชยพงศ์

นายบัญชา ดอกคำ

นายสมภพ จีนปั่น

นายอภิเชษฐ์ อนุตรวณิชกุล

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์

ภาควิชา ครุศาสตร์วิศวกรรม

คณะ ครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2537

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อปริญญานิพนธ์ ตัวควบคุมลำดับโดยใช้ MCS-51

Programmable Logic Control.by MCS-51

ชื่อนักศึกษา

1.นางสาวนัยนา วาณิชยพงศ์

2.นายบัญชา ดอกคำ

3.นายสมภพ จีนปั่น

4.นายอภิเชษฐ์ อนุตรวณิชกุล

อาจารย์ผู้ควบคุมปริญญานิพนธ์

1.อาจารย์กิติพงค์ มะโน

2.อาจารย์วรวิทย์ สมหา

3.อาจารย์สันติ ต้นตระกูล

คณะกรรมการสอบปริญญานิพนธ์	ลายมือ
อาจารย์กิติพงค์ มะโน	
อาจารย์พีระวุฒิ สุวรรณจันทร์	
อาจารย์สุชิน อาจหาญ	
อาจารย์สันติ ต้นตระกูล	
อาจารย์โกศล ตราชู	

วัน/เดือน/ปี ที่สอบ วันที่ 24 เดือนธันวาคม พ.ศ.2537 เวลา 13.30 น.ถึง 14.30 น.

สถานที่สอบ ห้อง ค.301 คณะครุศาสตร์อุตสาหกรรม

ลงนาม ....

(ผศ.ดร.[illegible] ณ อยุธยา)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

ปริญญานิพนธ์

เรื่อง ตัวควบคุมลำดับโดยใช้ MCS-51

Programmable Logic Control by MCS-51

ผู้จัดทำ

1. นางสาวนัยนา วาณิชยพงศ์
2. นายบัญชา ดอกคำ
3. นายสมภพ จีนปั่น
4. นายอภิเชษฐ์ อนุตรวณิชกุล

อาจารย์ที่ปรึกษา

ลงนาม .....

(อาจารย์ กิติพงศ์ มะโน)

ลงนาม .....

(อาจารย์ วรวิทย์ สมหา)

ลงนาม .....

(อาจารย์ สันติ ตันตระกูล)

หัวหน้าภาควิชา

ลงนาม .....

(ผศ.ดร.ธีรพล เทพหัสดิน ณ อยุธยา)

ปริญญานิพนธ์

ตัวควบคุมลำดับโดยใช้ MCS-51

Programmable Logic Control by MCS-51

จุดประสงค์

1. เพื่อศึกษาระบบควบคุมลำดับ
2. เพื่อออกแบบวงจรตัวควบคุมลำดับโดยใช้ MCS-51
3. เพื่อสร้างวงจรตัวควบคุมลำดับโดยใช้ MCS-51
4. เพื่อแสดงการทำงานของตัวควบคุมลำดับโดยใช้ MCS-51
5. เพื่อสามารถนำไปให้นักศึกษาหรือผู้สนใจได้ศึกษา หรือนำไปใช้งานในด้านการควบคุม

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้ และ เข้าใจในทฤษฎี หลักการทำงานของระบบควบคุมลำดับ
2. ออกแบบวงจรตัวควบคุมลำดับโดยใช้ MCS-51 ได้
3. สร้างวงจรตัวควบคุมลำดับโดยใช้ MCS-51 ได้
4. เขียนโปรแกรมควบคุมของตัวควบคุมลำดับโดยใช้ MCS-51 ได้
5. เพื่อให้ผู้สนใจได้ศึกษาเป็นแนวทางในการศึกษาต่อไป

ตัวควบคุมลำดับโดยใช้ MCS-51

นางสาวนัยนา วาณิชยพงศ์

นายบัญชา ดอกคำ

นายสมภพ จีนปั่น

นายอภิเชษฐ์ อนุตรวณิชกุล

อาจารย์ที่ปรึกษา

อาจารย์กิติพงค์ มะโน

อาจารย์วรวิทย์ สมหา

อาจารย์สันติ ตันตระกูล

ปีการศึกษา 2537

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เสนอ ตัวควบคุมลำดับโดยใช้ MCS-51 ซึ่งเป็นตัวควบคุมที่สร้างขึ้นเพื่อให้นักศึกษาได้รู้จักกับตัวควบคุมลำดับแบบพื้นฐาน นักศึกษาได้ทดลองเขียนโปรแกรมอย่างง่ายๆ และจะเป็นพื้นฐานในการใช้ตัวควบคุมลำดับชนิดอื่นๆได้ เราจะใช้ไมโครคอนโทรเลอร์ 8 บิท เบอร์ 8031 เหมาะสำหรับงานคอนโทรลต่างๆตัวควบคุมที่สร้างขึ้นนี้เหมาะสำหรับใช้ในการทดสองหรือควบคุมงานเล็กๆ เป็นต้น

PROGRAMMABLE LOGIC CONTROL BY MCS-51

MISS.NAIYANA WANITCHAYÀPONG

MR.BUNCHA DORGKUM

MR.SOMPOB JEENPUN

MR.APICHET ANUTARAVANICHAKUL

ADVISOR

MR.KITIPHONG MANO

MR.WORAVIT SOMHA

MR.SANTI TUNTRAKOOL

1994

ABSTRACT

THIS THISIS PROPOSES THE PROGRAMMABLE LOGIC CONTROL BY MCS-51, THIS IS CONTROLLER THAT BUILD FOR STUDENT WHO WANT TO KNOW THE BASIC OF PROGRAMMABLE LOGIC CONTROL, THAT FOR STUDENT HOW TO WRITE THE' SIMPLE PROGRAM, IT WILL BE THE BASIC FOR USE ANOTHER PROGRAMMABLE LOGIC CONTROL. THIS WE WILL USE 8 BITS MICROCONTROLLER 8031 SERIES,IT SUITABLE FOR USE IN CONTROL WORK. THIS CONTROLLER, THAT BUILD UP. IT SUITABLE FOR USE IN EXPERIMENTAL OR SMALL CONTROLLER ETC.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงมาได้เพราะพระคุณของ คุณแม่ คุณพ่อ และ ความกรุณาจากท่านอาจารย์ธีระพล เทพหัสดิน ณ อยุธยา และ โดยเฉพาะอย่างยิ่งอาจารย์กิติพงค์ มะโน ซึ่งได้ช่วยเหลือคณะผู้จัดทำมาโดยตลอดทั้งให้คำแนะนำ และให้ข้อมูลมาศึกษาและให้แนวทางในการทำ ซึ่งในโอกาศนี้คณะผู้จัดทำโครงงานรู้สึกซาบซึ้งในพระคุณของทุกๆท่านจึงขอขอบพระคุณเป็นอย่างยิ่งมา ณ โอกาสนี้ด้วย

สารบัญ

เรื่อง	หน้า

สารบัญตาราง

ตาราง **หน้า**

สารบัญภาพ

รูปที่ หน้า

บทที่ 1

บทนำ

ในปัจจุบันนี้ในโรงงานอุตสาหกรรมในประเทศไทยได้ขยายตัวไปอย่างมากซึ่งโรงงานส่วนใหญ่ก็ต้องมีเครื่องจักรที่ใช้ในการผลิตสินค้า ซึ่งก็ต้องมีการควบคุมเกิดขึ้น ดังนั้นจึงได้เกิดอุปกรณ์ชนิดหนึ่งเกิดขึ้นคือ **PLC (Programmable Logic Control)**

ซึ่งอุปกรณ์ชนิดนี้นำมาใช้ควบคุมระบบการผลิตในโรงงานแทนการใช้ระบบรีเลย์แบบเก่า ซึ่งในตัวPLCนี้จะมีไมโครโปรเซสเซอร์คอยควบคุมอยู่ มีวิธีการเขียนโปรแกรมคล้ายกับการเขียนโปรแกรมคอมพิวเตอร์โดยทั่วไป เมื่อเขียนเสร็จแล้วก็นำลงเก็บในหน่วยความจำ ซึ่งต่างกับระบบรีเลย์ซึ่งต้องมีการเดินสายที่ยุ่งยากการเปลี่ยนแปลงทำได้ลำบาก แต่ในPLCเปลี่ยนแปลงทำได้ง่ายเพียงแต่เปลี่ยนตัวโปรแกรม และเปลี่ยนอุปกรณ์ที่ต่อทางด้าน อินพุท และ เอาท์พุท เท่านั้นเอง

แต่ในปัจจุบันราคาของPLCในท้องตลาดนั้นยังมีราคาสูงอยู่มาก ดังนั้นจึงได้คิดที่จะทำ PLC ขึ้นโดยใช้ซีพียูที่มีอยู่ในท้องตลาดซึ่งเบอร์ที่เหมาะสมก็คือเบอร์ในตระกูล MCS-51 ซึ่งใช้ในงานคอนโทรลได้ดีซึ่งในที่นี้เราได้นำเอาชุดโมดูลที่มีในท้องตลาด นำมาประกอบกันเป็นเครื่อง PLC แล้วเขียนโปรแกรมควบคุมโดยจะอาศัยคำสั่งจากPLCที่มีอยู่ในท้องตลาดมาเป็นมาตรฐาน เช่นPLC ของ **Mitsubishi** เป็นต้น โดยในต้นแบบที่ทำขึ้นนี้จะเน้นไปที่ให้บุคคลที่ต้องการศึกษาในเรื่องการใช้งานPLCและนักศึกษาที่เรียนวิชาคอนโทรลซึ่งมีรายละเอียดในแต่ละบทดังนี้

บทที่ 1 บทนำ

บทที่ 2 ทฤษฎีการทำงานของ PLC องค์ประกอบในระบบ PLC หน่วยประมวลผลกลาง หน่วยอินพุท/เอาท์พุท หน่วยป้อนข้อมูล

บทที่ 3 การออกแบบการนำเอาโมดูลแต่ละชุดมารวมกัน การเขียนโปรแกรมควบคุม

บทที่ 4 ผลการทดลองของPLC

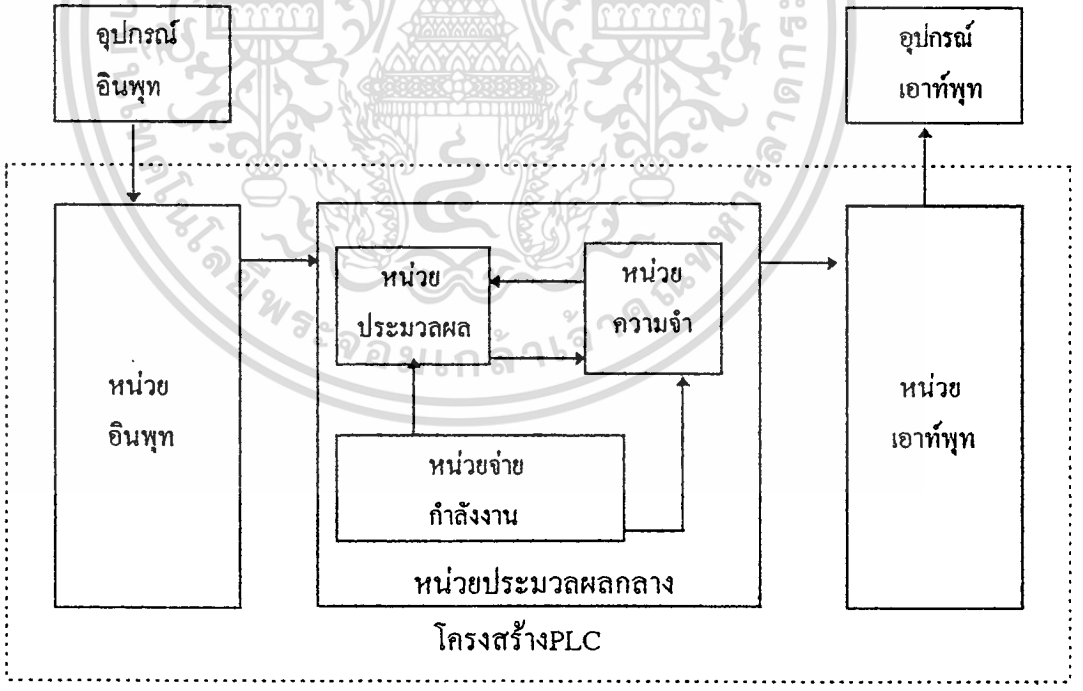
บทที่ 5 บทสรุปและวิจารณ์ผลการทดลองและแนวทางการปรับปรุงและพัฒนาต่อไป

บทที่ 2

ทฤษฎี และ หลักการ

2.1 ทฤษฎีการทำงานของ PLC

PLC เป็นคอมพิวเตอร์ชนิดหนึ่งที่ถูกออกแบบมาเพื่อใช้กับงานควบคุมในโรงงานอุตสาหกรรม ซึ่งต้องทนต่ออุณหภูมิ ความชื้น ฝุ่นละออง ระบบไฟฟ้าที่มีการรบกวนไม่สม่ำเสมอ แรงกระแทกและแรงสั่นสะเทือนต่างๆ ได้ดี โครงสร้างพื้นฐานของ PLC ยังคงคล้ายคลึงกับระบบคอมพิวเตอร์ทั่วไป คือประกอบด้วย หน่วยอินพุท (Input Interface) หน่วยประมวลผลกลาง(Processor) และหน่วยเอาท์พุท (Output Interface) ซึ่งในการติดตามควบคุมการทำงานของระบบนั้น PLC ยังต้องมีการติดต่ออุปกรณ์ภายนอกต่างๆ คือ อุปกรณ์อินพุทและอุปกรณ์เอาท์พุท ซึ่งถึงแม้จะไม่ใช่ส่วนประกอบของ PLC แต่เราถือว่าเป็นส่วนหนึ่งของระบบPLC ดังรูปที่ 2.1



รูปที่ 2.1 รูปแสดงระบบPLC

2.2 การทำงานของระบบPLC

ประกอบด้วยขั้นตอนต่างๆ ดังนี้

2.1.1 อ่านสภาวะของอุปกรณ์อินพุท

ซึ่งอาจจะเป็นสวิตช์ต่างๆหรืออุปกรณ์ตรวจจับสัญญาณ(Sensor)ที่วัดค่าสัญญาณอะนาล็อกเข้ามาทางหน่วยอินพุท

2.1.2 หน่วยอินพุท

ทำการแปลงสภาวะของอุปกรณ์อินพุท ให้อยู่ในรูปแบบที่หน่วยประมวลผลสามารถเข้าใจได้ แล้วส่งไปเก็บไว้ในหน่วยความจำของหน่วยประมวลผลกลาง

2.1.3 หน่วยประมวลผล

นำโปรแกรมของผู้ใช้มาปฏิบัติตามทีละ 1 คำสั่ง โดยเริ่มจากคำสั่งแรกจนสิ้นสุดโปรแกรมในหน่วยความจำโดยหน่วยประมวลผลจะใช้สภาวะต่างๆของอุปกรณ์อินพุทที่เก็บไว้ในหน่วยความจำ มาทำการประมวลผลตามเงื่อนไขที่กำหนดไว้ในแต่ละคำสั่ง ถ้าผลการปฏิบัติของ คำสั่งใดทำให้สภาวะของอุปกรณ์เอาท์พุทมีการเปลี่ยนแปลง ผลดังกล่าวจะถูกบันทึกเอาไว้ในหน่วยความจำก่อน จากนั้นหน่วยประมวลผลจะนำคำสั่งต่อไปของผู้ใช้ขึ้นมาปฏิบัติต่อ

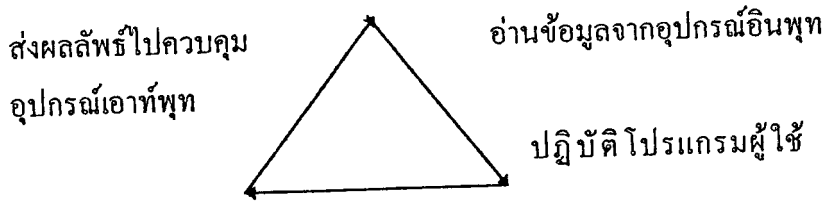
2.1.4 หน่วยเอาท์พุท

นำสภาวะของเอาท์พุทจุดต่างๆในหน่วยความจำ มาแปลงให้อยู่ในรูปแบบสัญญาณที่สามารถควบคุมอุปกรณ์เอาท์พุทได้ เพื่อส่งต่อไปยังอุปกรณ์เอาท์พุท

2.1.5 อุปกรณ์เอาท์พุท

ซึ่งอาจเป็น หลอดไฟฟ้า กระดิ่งไฟฟ้า มอเตอร์ไฟฟ้า หรือวาล์วควบคุมจะทำการปิดเปิดตามสัญญาณที่ส่งมาควบคุมโดยหน่วยเอาท์พุท

การทำงาน 1 รอบของ PLC จากขั้นตอนที่ 1 ถึงขั้นตอนที่ 5 เราเรียกว่า การสแกน และเรียกช่วงเวลาที่ PLC ใช้ในการสแกน 1 รอบว่า ช่วงเวลาสแกน (Scan Timer) ซึ่งจะใช้เวลาประมาณ 1 ถึง 100 มิลลิวินาที ขึ้นอยู่กับขนาดความยาวของโปรแกรมผู้ใช้ประเภทและจำนวนของอุปกรณ์อินพุท รวมทั้งคุณลักษณะของ PLC เอง



รูปที่ 2.2 การสแกนของ PLC

ช่วงเวลาสแกนของPLCจะกำหนดขีดความสามารถของ PLC ในการตรวจจับการเปลี่ยนแปลงของอุปกรณ์ภายนอกและการควบคุมเครื่องจักรเช่น PLC ที่มีช่วงเวลาสแกน 10 มิลลิวินาที ย่อมไม่สามารถรับสภาวะแท้จริงของอุปกรณ์ที่มีการเปลี่ยนแปลงทุก 8 มิลลิวินาที ได้ถ้าใช้ PLC ดังกล่าวการควบคุมจะผิดพลาดหมด

อุปกรณ์อีกชนิดหนึ่งในระบบ PLC คือ หน่วยรับข้อมูล (Programming device) ซึ่งใช้สำหรับป้อนโปรแกรมผู้ใช้เข้าไปยังหน่วยความจำของ PLC โดยโปรแกรมผู้ใช้จะเป็นตัวกำหนดเงื่อนไขที่หน่วยประมวลผลนำไปใช้ในการตัดสินใจว่าส่งสัญญาณไปควบคุมอุปกรณ์เอาท์พุทในลักษณะใด เมื่อสภาวะอุปกรณ์อินพุทมีการเปลี่ยนแปลง

2.3 องค์ประกอบในระบบPLC

2.3.1 หน่วยประมวลผลกลาง

มีหน้าที่ควบคุมการทำงานทั้งหมดของเครื่อง PLC ประกอบด้วยหน่วยประมวลผล หน่วยความจำ และหน่วยจ่ายกำลังงาน โดยในแต่ละส่วนมีหน้าที่ดังนี้

หน่วยประมวลผล มีหน้าที่นำโปรแกรมผู้ใช้ (User Program) มาปฏิบัติ เพื่อควบคุมอุปกรณ์ภายนอกตามเงื่อนไขการควบคุมที่ผู้เขียนโปรแกรมต้องการควบคุมการติดต่อรับส่งข้อมูลระหว่างหน่วยประมวลผลกลางกับหน่วยอินพุท/เอาท์พุทและหน่วยอินพุท/เอาท์พุทกับอุปกรณ์ภายนอก ติดต่อกับผู้ใช้และอุปกรณ์ร่วม ตรวจสอบสภาพการทำงานของ PLC โดยมีโปรแกรมบริหารระบบ (Operating System) เป็นผู้ควบคุมอีกที

โดยทั่วไปหน่วยประมวลผลจะเป็นไมโครโปรเซสเซอร์ ซึ่งนอกจากจะทำหน้าที่แทนวงจรรีเลย์ในการควบคุมแบบ ON/OFF เหมือนวงจรตรรกแล้วยังสามารถคำนวณทางคณิตศาสตร์และติดค่อกับอุปกรณ์ภายนอกได้

โปรแกรมบริหารระบบ (Operating System)	SYSTEM MEMORY
ข้อมูลจากการทำงานโปรแกรมระบบ (Scratch Pad)	

ตารางที่ 2.1 หน่วยความจำของระบบ PLC

ระบบโปรแกรมผู้ใช้ (User Program)	SYSTEM MEMORY
ข้อมูลจากการทำงานโปรแกรมระบบ (Scratch Pad)	
ตารางข้อมูล (Data Table)	

ตารางที่ 2.2 แสดงการแบ่งหน่วยความจำของผู้ใช้ PLC

ของการปฏิบัติการชั่วคราวของโปรแกรมผู้ใช้งาน นอกจากนี้หน่วยความจำในส่วนนี้ ยังใช้เป็นที่เก็บสภาวะต่างๆของอุปกรณ์ภายนอกและอุปกรณ์ภายใน หรือที่เรียกว่าตารางข้อมูล ซึ่งเราสามารถแบ่งออกเป็น 4 ส่วนด้วยกันคือ

ตารางอินพุท (Input Table)

ตารางเอาท์พุท (Output Table)

ตารางอุปกรณ์ภายใน (Internal Discrete Table)

ตารางรีจิสเตอร์ (Register Table)

ตารางอินพุท มีหน้าที่เก็บข้อมูลที่แสดงสภาวะของอุปกรณ์อินพุทภายนอก โดยจะเก็บค่า "1" แทนสภาวะการ ON และเก็บค่า "0" แทนสภาวะการ OFF ของอุปกรณ์เอาท์พุท โดยอุปกรณ์เอาท์พุทภายนอกที่เชื่อมต่อกับหน่วยเอาท์พุทจะเปลี่ยนสภาวะตามตารางเอาท์พุทก่อนที่จะสิ้นสุดการสแกนทุกครั้ง

ตารางอุปกรณ์ภายใน มีหน้าที่เก็บสภาวะของอุปกรณ์ภายในที่ไม่มีจุดเชื่อมต่อกับอุปกรณ์ภายนอก เช่น สภาวะของอุปกรณ์หน่วงเวลา อุปกรณ์นับจำนวน หรือรีเลย์ภายใน ซึ่งใช้ในการเก็บสภาวะการควบคุมชั่วคราว เช่นเดียวกับรีเลย์ควบคุมในวงจรรีเลย์

ตารางรีจิสเตอร์ มีหน้าที่เก็บข้อมูลที่เป็นตัวเลขหรือข้อมูลอะนาลอก ซึ่งไม่สามารถใช้ตารางอินพุท/เอาท์พุทหรือตารางอุปกรณ์ภายในเก็บรักษาได้ เพราะมีค่าเป็นตัวเลข ไม่ใช่สภาวะ ON/OFF ตารางรีจิสเตอร์ของ PLC ประกอบด้วย อินพุทรีจิสเตอร์ เอาท์พุทรีจิสเตอร์ และรีจิสเตอร์ภายใน

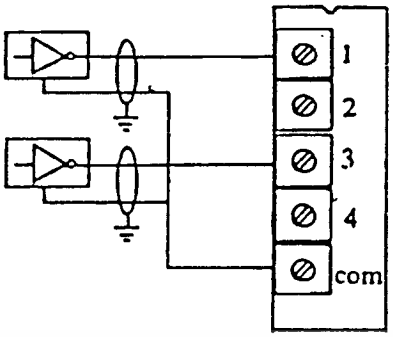
อินพุทรีจิสเตอร์ จะเก็บข้อมูลจากอุปกรณ์อินพุทซึ่งจะเปลี่ยนค่าเมื่อเริ่มการสแกนทุกครั้ง เอาท์พุทรีจิสเตอร์จะเก็บข้อมูลที่ควบคุมอุปกรณ์เอาท์พุท และส่งค่าเมื่อสิ้นสุดการสแกนทุกครั้ง ส่วนรีจิสเตอร์ภายในจะเก็บข้อมูลชั่วคราวที่ได้จากการประมวลผลและการคำนวณทางคณิตศาสตร์ ค่าคงที่ของการหน่วงเวลาและจำนวนนับ รวมทั้งค่าเวลาปัจจุบันและจำนวนนับของอุปกรณ์ภายใน

หน่วยจ่ายกำลังงาน ทำหน้าที่จ่ายกระแสไฟฟ้าให้กับส่วนต่างๆ ของ PLC คือหน่วยประมวลผล หน่วยความจำและหน่วยอินพุท/เอาท์พุท โดยรักษาแรงดันไฟฟ้าให้คงที่ และแจ้งให้หน่วยประมวลผลทราบเมื่อระบบไฟฟ้ามีอาการผิดปกติ

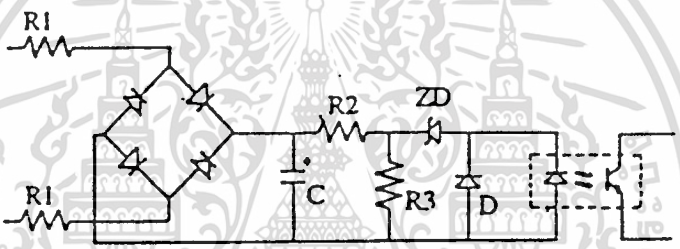
2.3.2 หน่วยอินพุท/เอาท์พุท

ทำหน้าที่ติดต่อระหว่าง PLC กับอุปกรณ์ภายนอก ซึ่งมีทั้งหน่วยอินพุท/เอาท์พุท แบบตรรก (Digital Input/Output) ซึ่งใช้ในการควบคุมแบบ ON/OFF และหน่วยอินพุท/เอาท์พุทแบบตัวเลข (Analog Input/Output) ซึ่งใช้ตรวจจับระดับสัญญาณต่าง ๆ หรือตำแหน่งของเครื่องจักร ทำให้ขอบเขตการใช้PLCในงานควบคุมกว้างขึ้น

สัญญาณไฟฟ้าที่อุปกรณ์อินพุท/เอาท์พุทแบบตรรกใช้มีหลายค่าทั้งแบบแรงดันไฟฟ้ากระแสตรงและกระแสสลับ ซึ่งเราพอสรุปสัญญาณมาตรฐานที่หน่วยอินพุท/เอาท์พุทใช้ในการติดต่อกับ PLC ได้ดังตารางที่ 2.3



รูปที่ 2.4 การเชื่อมต่อหน่วยอินพุทแบบ TTL

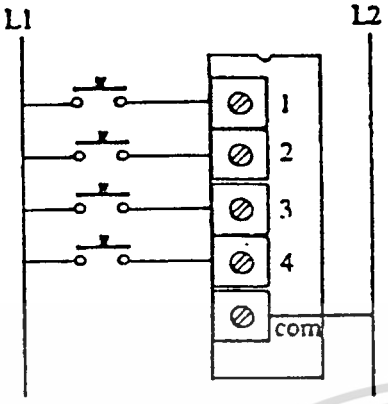


รูปที่ 2.5 วงจรหน่วยอินพุทแบบ AC/DC

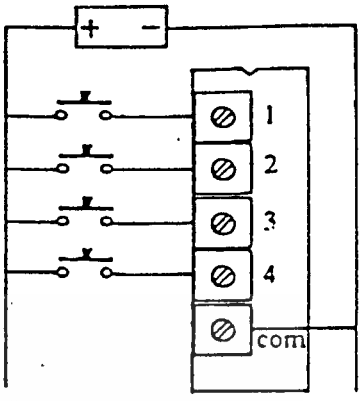
หน่วยเอาท์พุทแบบ TTL มีหน้าที่ควบคุมการทำงานของอุปกรณ์อิเล็กทรอนิกส์ที่เป็นแบบTTLรืออุปกรณ์ที่ใช้สัญญาณไฟฟ้า 5 V_{DC} ซึ่งมีวิธีการเชื่อมต่อดังรูปที่ 2.7

หน่วยเอาท์พุทแบบ AC จะเปลี่ยนสภาวะควบคุมจากหน่วยประมวลผลกลางให้เป็นระดับแรงดันไฟฟ้ากระแสสลับเพื่อควบคุมการทำงานของอุปกรณ์เอาท์พุทโดยประกอบด้วย วงจรตรรกซึ่งติดต่อกับหน่วยประมวลผลกลาง วงจรเชื่อมต่อแบบออปติก สวิตช์อิเลกทรอนิกส์ และวงจรกรองความถี่

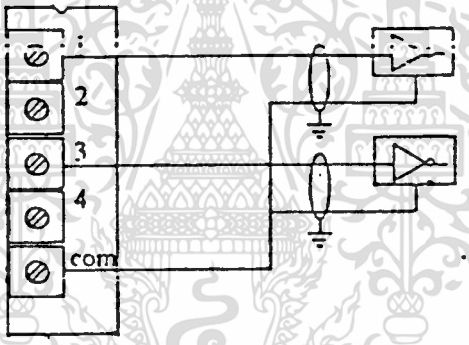
วงจรเชื่อมแบบออปติก จะทำหน้าที่แยกระบบไฟของชุด PLC ออกจากอุปกรณ์ภายนอก (Electrical Isolate) เพื่อป้องกันไม่ให้ระบบไฟของอุปกรณ์ภายนอกเข้ามาทำความเสียหายให้กับหน่วยประมวลผลกลาง สวิตช์อิเล็กทรอนิกส์ในหน่วยเอาท์พุทนี้มักใช้ไตรแอค (Triac) หรือเอสซีอาร์ (SCR) ซึ่งเปิดและปิดโดยไม่เกิดประกายไฟเหมือนหน้าสัมผัสรีเลย์ จึงไม่รบกวนการทำงานของอุปกรณ์ภายนอกและยืดอายุการใช้งานของ PLC



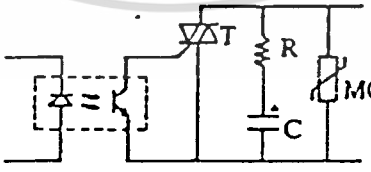
รูปที่ 2.6 ก) การเชื่อมต่อหน่วยอินพุทแบบAC



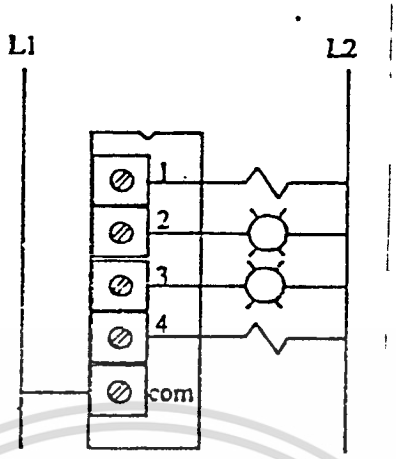
รูปที่ 2.6 ข) การเชื่อมต่อหน่วยอินพุทแบบDC



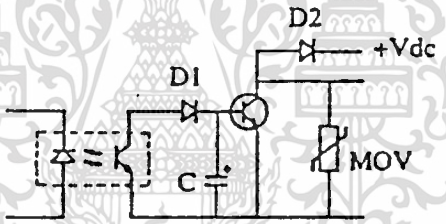
รูปที่ 2.7 การเชื่อมต่อหน่วยเอาท์พุทแบบ TTL



รูปที่ 2.8 วงจรหน่วยเอาท์พุทแบบ AC



รูปที่ 2.9 การเชื่อมต่อหน่วยเอาท์พุทแบบ AC



รูปที่ 2.10 วงจรหน่วยอินพุทแบบ DC

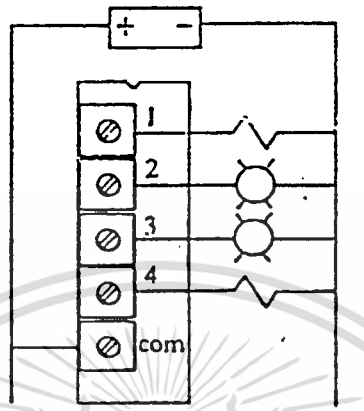
หน่วยเอาท์พุทแบบ DC ทำหน้าที่ควบคุมอุปกรณ์เอาท์พุท ที่ทำงานด้วยสัญญาณไฟฟ้ากระแสตรง ลักษณะการทำงานของวงจรคล้ายกับหน่วยเอาท์พุทแบบ AC แต่ใช้ทรานซิสเตอร์เป็นตัวเปิด/ปิดวงจรไฟฟ้าแทนไทรแอคและเอสซีอาร์ดังรูปที่ 2.10

2.3.3 หน่วยป้อนข้อมูล

ทำหน้าที่ป้อน ตรวจสอบและแก้ไขโปรแกรมหน่วยความจำ ซึ่งโปรแกรมบริหารระบบจะนำไปประมวลเพื่อควบคุมอุปกรณ์ภายนอกอีกทีหนึ่ง หน่วยป้อนข้อมูลของ PLC แบ่งออกเป็น 3 ประเภทคือ

เครื่องป้อนโปรแกรมจอภาพ (CRT Programer) เครื่องป้อนโปรแกรมจอภาพจะประกอบด้วย จอภาพ แป้นพิมพ์ และวงจรอิเล็กทรอนิกส์ ที่ทำหน้าที่ติดต่อกับ PLC มีลักษณะ

คล้ายกับคอมพิวเตอร์ทั่วไป ผู้ใช้สามารถป้อนโปรแกรมโดยใช้ภาษาแลดเดอร์หรือสัญญลักษณ์รีเลย์โดยตรง



รูปที่ 2.11 การเชื่อมต่อหน่วยเอาท์พุทแบบ DC

เครื่องป้อนโปรแกรมขนาดเล็ก (Mini Programer) มีลักษณะคล้ายเครื่องคำนวณเลข ประกอบด้วยแป้นพิมพ์และส่วนแสดงผลชนิดแอลอีดี หรือ แอลซีดี แสดงผลได้ครั้งละ 1-2 บรรทัดแป้นพิมพ์ประกอบด้วยแป้นพิมพ์ตัวเลขคำสั่งและฟังก์ชันพิเศษต่าง ๆ

คอมพิวเตอร์ (Computer) บริษัทบางแห่งได้จัดทำโปรแกรมสำเร็จรูปเพื่อใช้คอมพิวเตอร์ส่วนบุคคลทำหน้าที่ป้อนโปรแกรม ติดต่อกับเครื่อง PLC และจัดทำรายการต่าง ๆ

คำสั่งที่ใช้เขียนโปรแกรม PLC มี 4 ภาษา คือ ภาษาแลดเดอร์ (Ladder Diagram) ภาษาบูลลีน ภาษาบล็อก และคำสั่งข้อความภาษาอังกฤษ (English Statememt Luage) ซึ่งแต่ละภาษามีวิธีใช้ที่แตกต่างกัน ภาษาแลดเดอร์และภาษาบูลลีนเป็นภาษาพื้นฐานที่ใช้กับ PLC ขนาดเล็กแทนอุปกรณ์รีเลย์ อุปกรณ์หน่วงเวลาและอุปกรณ์นับจำนวน ในการควบคุมแบบ ON/OFF ภาษาบล็อกและคำสั่งข้อความภาษาอังกฤษระดับสูง มักใช้กับการควบคุมที่ซับซ้อนหรือมีการคำนวณทางคณิตศาสตร์เกี่ยวข้องเช่น การควบคุมแบบอนาล็อกและการควบคุมตำแหน่ง

ภาษาแลดเดอร์ประกอบด้วยสัญลักษณ์หน้าสัมผัส และขดลวด มีลักษณะคล้ายวงจรรีเลย์ การเขียนโปรแกรมภาษาแลดเดอร์จากวงจรรีเลย์จึงทำได้ง่าย โปรแกรมภาษาแลดเดอร์ที่ประกอบด้วยหน้าสัมผัสต่างๆ ที่ทำงานร่วมกัน เพื่อส่งสภาวะการควบคุมไปยังอุปกรณ์เอาท์พุท 1 จุด เราเรียกว่า รังก์ (Rung) บางครั้งโปรแกรมภาษาแลดเดอร์ 1 รังก์ อาจมีอุปกรณ์เอาท์พุทมากกว่า 1 จุด แต่อุปกรณ์เอาท์พุทเหล่านี้ต้องได้รับสภาวะการควบคุมจากจุดเดียวกันเสมอ

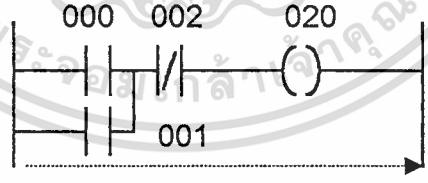
การเขียนโปรแกรมภาษาแลดเดอร์ ต้องระบุตำแหน่ง หรือ หมายเลขของอุปกรณ์ที่ต้องการให้ถูกต้องและตรงกันทุกครั้ง โดยหมายเลขของหน้าสัมผัสและขดลวดนี้จะสัมพันธ์กับตารางข้อมูล

สัญลักษณ์หน้าสัมผัส หมายถึง การรับสภาวะของหน่วยอินพุท/เอาท์พุท และอุปกรณ์ภายใน เพื่อปฏิบัติลอจิกตามเงื่อนไขควบคุม หน้าสัมผัสที่ใช้ในวงจรรีเลย์ หรือวงจรแลดเดอร์ด้วยกัน 2 ชนิด คือ

หน้าสัมผัสแบบปกติเปิด (Normally-Open Contact) (-----| |-----) ถ้าหน้าสัมผัสแบบปกติเปิดมีสภาวะเป็นจริงหรือ ON หมายถึงการปิดหรือต่อวงจรไฟฟ้า ถ้าหน้าสัมผัสแบบปกติเปิดจะมีสภาวะเป็นเท็จหรือ OFF หมายถึง สภาวะเปิดหรือตัดวงจรไฟฟ้า

หน้าสัมผัสแบบปกติปิด (Normally-Closed Contact) (-----|/|-----) ถ้าหน้าสัมผัสแบบปกติปิดมีสภาวะเป็นจริงหรือ ON หมายถึงการเปิดหรือตัดวงจรไฟฟ้า ถ้าหน้าสัมผัสแบบปกติปิดจะมีสภาวะเป็นเท็จหรือ OFF หมายถึง สภาวะปิดหรือต่อวงจรไฟฟ้า

สัญลักษณ์ขดลวด (Coil) -----( )------ หมายถึง การส่งคำสั่งหรือผลการควบคุมไปยังหน่วยเอาท์พุทหรืออุปกรณ์ภายใน รูปแบบการจัดเรียงหน้าสัมผัสทำให้เกิดสภาวะการควบคุมที่เรียกว่า "สภาวะรังก์" อุปกรณ์เอาท์พุทจะทำงาน เมื่อรังก์เป็นจริง หรือมีเส้นผ่านหน้าสัมผัสปิดจากด้านซ้ายของรังก์ มาที่ขดลวดเอาท์พุท และสิ้นสุดที่ปลายด้านขวาของรังก์ เกิดขึ้นอย่างน้อย 1 เส้นทาง



รูปที่ 2.12 โปรแกรมภาษาแลดเดอร์ 1 รังก์

ภาษาบูลลีนเป็นภาษาพื้นฐานของ PLC เช่นเดียวกับภาษาแลดเดอร์ คำสั่งในภาษาบูลลีนมีลักษณะคล้ายสัญลักษณ์ของพีชคณิตบูลลีน ซึ่งประกอบด้วยคำสั่งต่างๆ ที่สามารถทำงานได้เช่นเดียวกับภาษาแลดเดอร์

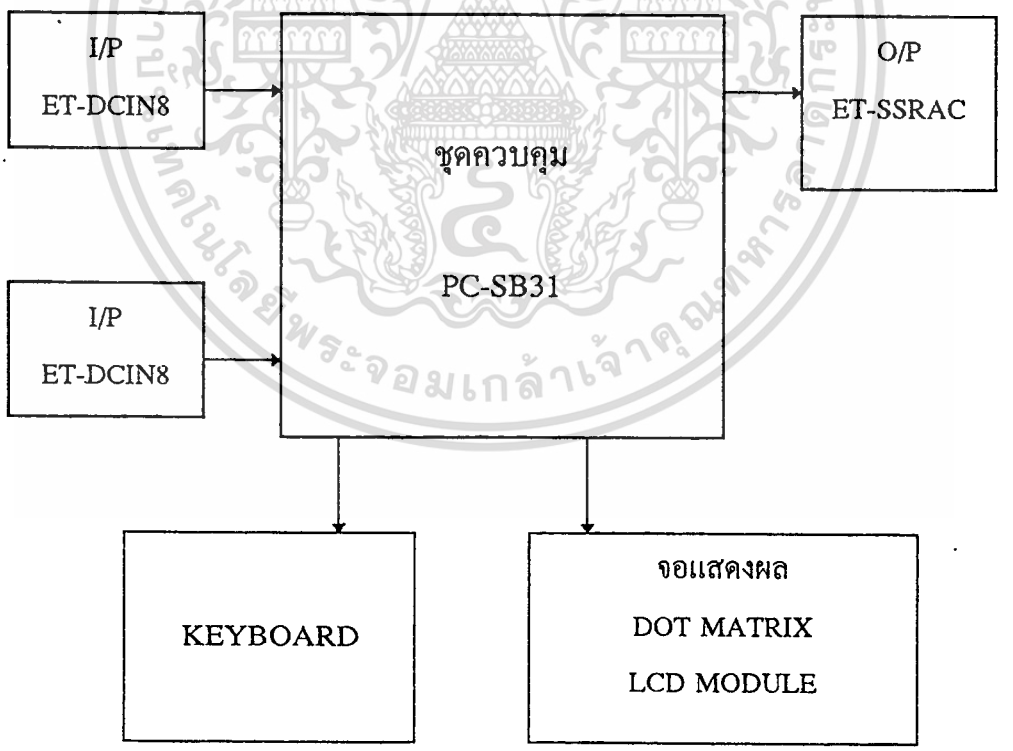
บทที่ 3

การออกแบบวงจร

จากการที่เราได้ทราบถึงรายละเอียดต่างๆ ในบทที่ 2 ไปแล้วนั้น ว่า PLC จะต้องประกอบไปด้วยส่วนประกอบดังนี้ คือ หน่วยอินพุท หน่วยประมวลผลกลาง หน่วยเอาท์พุทซึ่งในโครงงานนี้ได้กำหนดให้อินพุทมี 16 อินพุท และมี 8 เอาท์พุทและมีคีย์บอร์ด (keyboard) จำนวน 4 x 8 คีย์ จอแสดงผลแบบLCD (LCD) แบบ 4 บรรทัด และหน่วยประมวลผลกลางใช้ซีพียูตระกูล MCS-51 (8031 AH)

การออกแบบและการทำงาน

ในการออกแบบนั้นเราได้แบ่งการทำงานออกเป็น 5 ภาคด้วยกัน แสดงได้ดังรูปที่ 3.1



รูปที่ 3.1 บล็อกไดอะแกรมการทำงานของตัวควบคุมลำดับ

3.1 คุณสมบัติของตัวควบคุมลำดับ

ไมโครคอนโทรลเลอร์	: 8031 AH
หน่วยความจำ	: RAM 6264 (8 กิโลไบท์) ใช้งานที่แอดเดรส
	2000 - 3FFF
	EPROM 2764 (8 กิโลไบท์) ใช้งานที่แอดเดรส
	0000 - 1FFF
	RAM ภายใน 128 กิโลไบท์ ใช้เป็นรีจีสเตอร์ต่าง ๆ
พอร์ทอินพุท/เอาท์พุท	: ขนาด 24 บิท โดยใช้ชิพ 8255
	: ขนาด 8 บิท โดยใช้ P.1.ของ 8031
คีย์บอร์ด	: ขนาด 4x8 คีย์ โดยใช้ P.2.ร่วมกับ 74LS138
LCD	: ใช้แบบ Dot Matrix LCD ขนาด 4 แถว x 16 ตัวอักษร
ตัวตั้งเวลา	: มีจำนวน 16 ตัว ตั้งเวลาได้ 0 - 99.9 วินาที
ตัวนับจำนวน	: มีจำนวน 8 ตัว นับได้สูงสุด 0 - 999. ต่อตัว
สัญญาณนาฬิกา	: ใช้คริสตอล 11.0592 MHz

3.2. การทำงานของชุดควบคุม (PC-SB31)

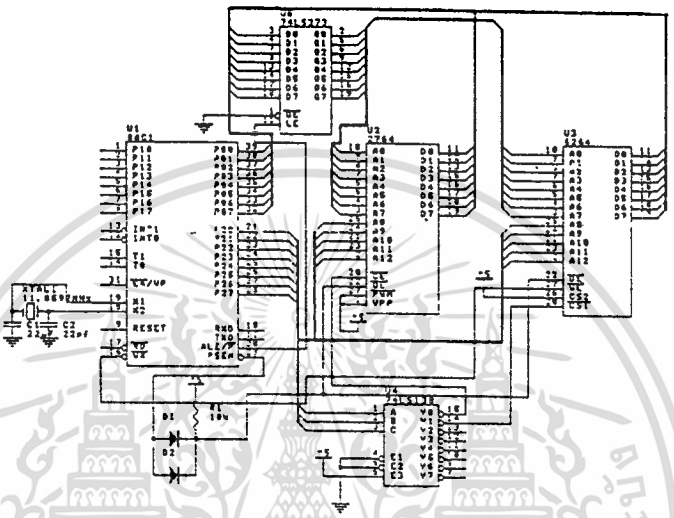
หัวใจของการทำงานอยู่ที่ ไมโครคอนโทรลเลอร์ 8031 AH ซึ่งอยู่ในบอร์ด PC-SB31 ซึ่งเป็นไมโครคอนโทรลเลอร์ที่มีแรมและพอร์ทภายใน สามารถใช้ได้เลย แต่เราต้องต่อเพิ่มขึ้นเพราะมีน้อยเกินไป จึงต่อแบบการใช้หน่วยความจำภายนอกโดยใช้ ไอซี 74LS373 เป็นตัวช่วยดังแสดงในรูปที่ 3.2

3.2.1 การจัดหน่วยความจำ

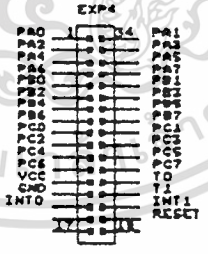
8031 สามารถต่อหน่วยความจำได้ 2 อย่าง คือ โปรแกรมเมโมรีและดาต้าเมโมรีอย่างละ 64 กิโลไบท์ สำหรับในโครงงานนี้จะใช้ 2764 เป็น EPROM ขนาด 8 กิโลไบท์ เป็นโปรแกรมเมโมรีที่ตำแหน่ง 0000-1FFF ส่วนแรมนั้นเราจะใช้เบอร์ 6264 ขนาด 8 กิโลไบท์ทำให้เป็นทั้งโปรแกรมเมโมรีและดาต้าเมโมรีที่ตำแหน่ง 2000 - 3FFF

3.2.2 ระบบบัส

ใช้บัสมาตรฐานของบอร์ด PC-SB31 มีขนาด 34 ขา ซึ่งขานี้จะตรงกับ 72IO พอร์ท ซึ่งเป็นอินพุทและเอาท์พุทที่ต่อจาก 8255 และใช้ P.1 ของ 8031AH อีกพอร์ทหนึ่ง ซึ่งพอร์ทนี้จะใช้ต่อกับ ET-SSRAC เพื่อทำเป็นชุดเอาท์พุทดังรูปที่ 3.3



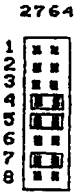
รูปที่ 3.2 การต่อหน่วยความจำภายนอกโดยใช้ ไอซี เบอร์ 74LS373



รูปที่ 3.3 การจัดตำแหน่งของ EXP4 ของ PC-SB31 (72IO)

3.2.3 การเลือกหน่วยความจำอีพรอม

ในบอร์ด PC-SB31 นี้จะมีจัมพ์เปอร์เพื่อเลือกว่าต้องการใช้หน่วยความจำอีพรอมเบอร์ใด ซึ่งในที่นี้เราจะใช้เบอร์ 2764 ขนาด 8 กิโลไบท์ 74LS373 เราก็จะเซ็ทจัมพ์เปอร์หมายเลข 3 (JP3) ดังรูปที่ 3.4



รูปที่ 3.4 การเลือกจัมพ์เปอร์สำหรับหน่วยความจำอีพรอมเบอร์ 2764

3.2.4 การเลือกหน่วยความจำแรม

ในบอร์ด PC-SB31 นี้จะมีจัมพ์เปอร์เพื่อเลือกว่าต้องการใช้หน่วยความจำแรมเบอร์ใด ซึ่งในที่นี้เราจะใช้เบอร์ 6264 เราก็จะเซทจัมพ์เปอร์หมายเลข 5 (JP5) ดังรูปที่ 3.5

รูปที่ 3.5 การเลือกจัมพ์เปอร์สำหรับหน่วยความจำแรมเบอร์ 6264

3.2.5 การเลือกตำแหน่งของแอดเดรสเริ่มต้น

เราต้องทำการเลือกตำแหน่งของแอดเดรสก่อนว่า เราต้องการให้อีพรอมอยู่ที่ตำแหน่งใด เดิ้มอยู่ที่ตำแหน่งใดโดยการเซทค่าจัมพ์เปอร์ ดังรูปที่ 3.6

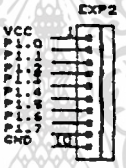
รูปที่ 3.6 การเลือกแอดเดรสเริ่มต้นของหน่วยความจำ

3.2.6 แอดเดรสของพอร์ทอินพุท/เอาท์พุทของ 8255

เราจะใช้ 8255 เป็นตำแหน่งหน่วยความจำที่ตำแหน่ง E000-FFFF แต่จะใช้จริงๆ เพียง 4 ไบท์แรกเท่านั้น ที่นำมาดีโคทพอร์ทแต่ละจุดของจัมพ์เปอร์ที่ใส่ลงไปจะมีขนาดของหน่วยความจำ 8 กิโลไบท์ เพิ่มขึ้นจากจุดเริ่มต้น

3.2.7 การเชื่อมต่อคีย์บอร์ด

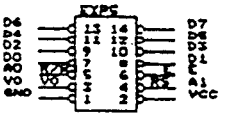
เป็นคอนเน็กเตอร์ 10 ขา ที่ต่อจาก P.1.(EXP2) ของ 8031 จะใช้เพียง 7 บิท เท่านั้น คือ 3 บิทล่าง คือ บิท 0 - บิท 2 จะนำไปต่อกับ 74LS138 ที่บอร์ดคีย์บอร์ด เพื่อเลือกทางหลัก และให้ 4 บิทบนคือ บิท 4 - บิท 7 เพื่อเลือกทางแถวจะได้คีย์บอร์ด จำนวน 32 คีย์ พอดีการจัดขาของพอร์ท P.1 เพื่อนำไปทำเป็นคีย์บอร์ดดังรูปที่ 3.7



รูปที่ 3.7 การจัดขาของพอร์ทอินพุทเอาท์พุท P.1 เพื่อนำไปทำเป็นคีย์บอร์ด

3.2.8 การเชื่อมต่อLCD

เป็นคอนเน็กเตอร์ 14 ขา (EXP5) ต่อออกมาจากบอร์ด PC-SB31 ใช้เชื่อมต่อโดยตรง และมีความต้านทานปรับค่าได้ขนาด 10 กิโลโอห์ม เพื่อใช้ในการปรับความเข้มของจอ LCD ขาสัญญาณที่ใช้ในการติดต่อแสดงในรูปที่ 3.8

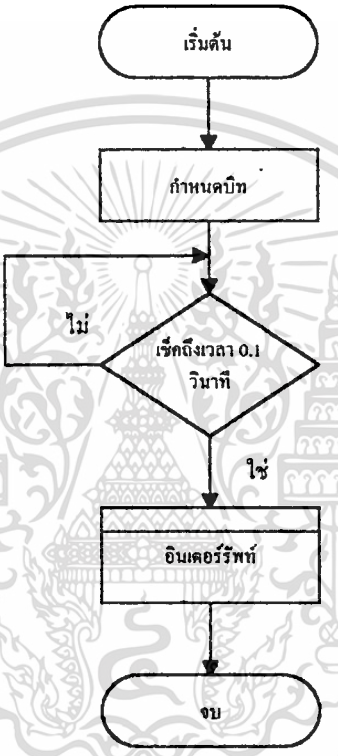


รูปที่ 3.8 การจัดขาของการต่อLCDเพิ่มเติม

3.2.9 ตัวตั้งเวลา

ในการใช้ PLC นั้นตัวตั้งเวลามีความส

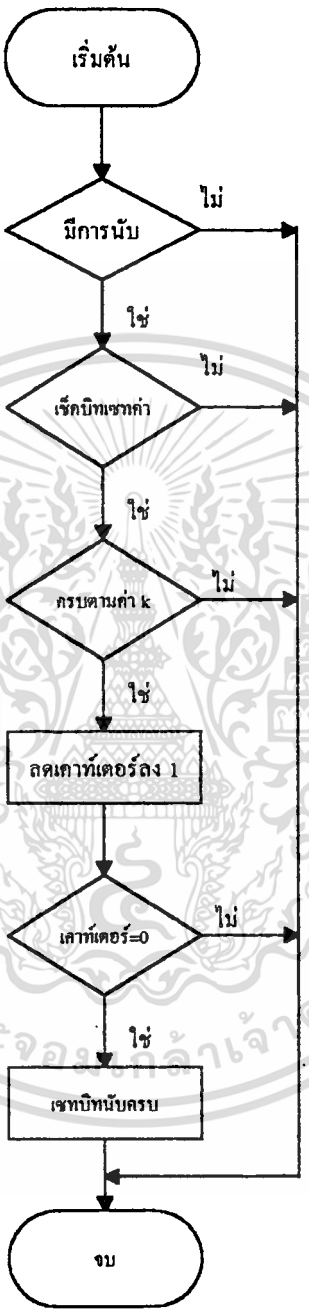
ควบคุมการทำงาน ซึ่งในนี้เราจะใช้ซอฟแวร์ที่เขียนขึ้นเพื่อทำเป็นตัวตั้งเวลา เพื่อให้ได้เวลาตามต้องการได้ โดยใช้ Times 0 ที่มีอยู่ใน 8031AH แล้วใช้เทคนิคการตั้งเวลาโดยใช้ไทม์เมอร์ตัวเดียว แล้วนำไปลดค่าในหน่วยความจำ จะทำให้เราได้ไทม์เมอร์ทั้งหมด 16 ตัว โฟลว์ชาร์ทรูปที่ 3.9



รูปที่ 3.9 โฟล์วชาร์ทของโปรแกรมตัวตั้งเวลา

3.2.10 ตัวนับ

ในการใช้PLC เราก็ต้องมีการนับเข้ามาเกี่ยวข้องด้วยอยู่ตลอดเวลา เราจึงได้ทำตัวนับขึ้นมาโดยใช้ซอฟแวร์ในการทำตัวนับ โดยการใช้ไทม์เมอร์เช่นเดียวกัน แต่เราจะใช้เป็นตัวรับสัญญาณแทนที่พอร์ทอินพุท แล้วนำไปลดค่าในหน่วยความจำแทน การทำงานของโปรแกรมตัวนับขั้นแรกจะตรวจสอบว่ามีการนับหรือไม่ถ้ามีก็จะตรวจสอบเช็คบิทเซทค่าถ้ามีจะตรวจสอบว่าครบตามค่า k หรือไม่ ถ้าครบจะลดเคาท์เตอร์ลง 1 แล้วตรวจสอบเคาท์เตอร์ว่าเท่ากับ 0 ถ้าไม่ จะจบโปรแกรม



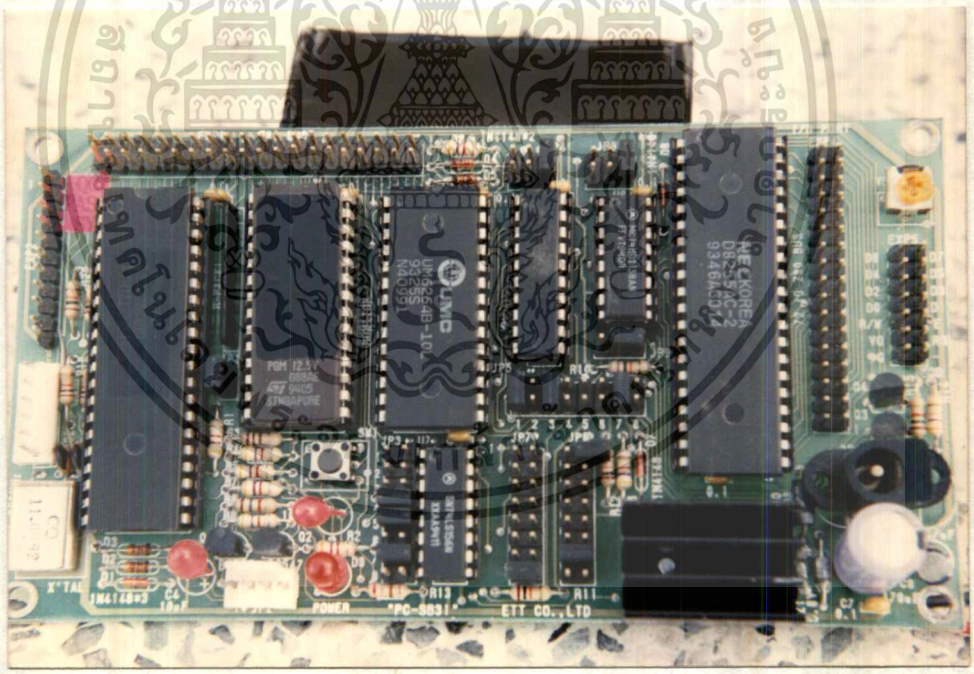
รูปที่ 3.10 โฟล์วชาร์ทของโปรแกรมตัวนับ

3.3 ชุดรับข้อมูล

ในเครื่องPLCนั้น เราจะต้องมีชุดรับข้อมูลจากภายนอกเข้ามาทำการประมวลผล ซึ่งเราจะใช้เป็นโมดูล อีกเช่นกันในการรับข้อมูลเราจะใช้บอร์ด ET-DCIN8 แต่ตามคุณสมบัติของ PLC ที่เราทำขึ้นนั้น กำหนดให้มี 16 อินพุท ดังนั้นเราต้องใช้บอร์ดชนิดนี้ 2 ตัว นำมาต่อกับบอร์ด PC-SB31 ทาง 8255

3.3.1 ลักษณะของบอร์ด

ET-DCIN8 เป็นอินพุทแบบ 8 บิท ในลักษณะของPhoto-coupled isolation ซึ่งจะกันการรบกวนเข้าสู่ระบบที่เราจะต่อร่วมเป็นอย่างดีโดยสามารถใช้กับอินพุทได้ 2 ระดับ คือ 5 V_{DC}และ 24 V_{DC} และนอกจากนี้ ET-DCIN8 จะยังมีส่วนที่เป็นเอาท์พุท 7 บิท แบบ Darlingtons open collector 500 mA สามารถต่อขับวงจรรีเลย์ต่างๆ ได้โดยตรง



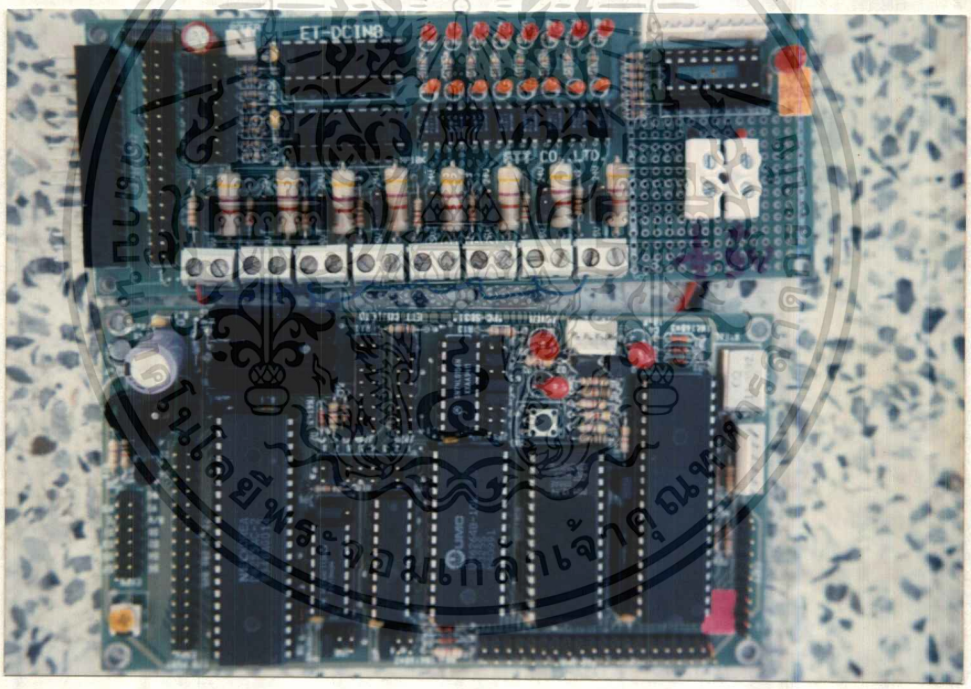
รูปที่ 3.11 บอร์ด PC-SB31

3.3.2 การต่อเข้ากับ PC-SB31

เราจะต่อเข้าทาง 72IO ของ PC-SB31 แล้วอีกด้านหนึ่งก็ต่อกับ72 IO ของ ET-DCIN8 ซึ่งในการต่อใช้งานนั้นเราต้องดูด้วยว่าเราต้องการใช้งานที่พอร์ทใดเนื่องจากชุดอินพุทมีอยู่ด้วยกัน 2 ชุด พอร์ทที่ ET-DCIN8 ต้องการใช้นั้น มี 2 พอร์ท คือ A และ B

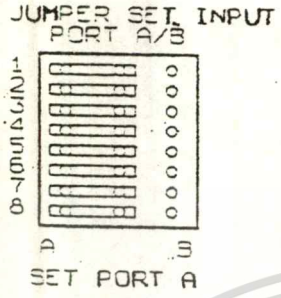
3.3.3 การดีโคดพอร์ท

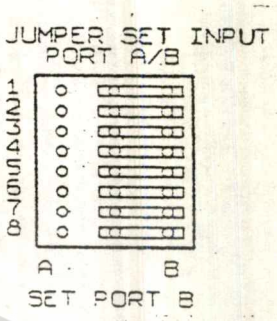
เมื่อเราต่อ ET-DCIN8 เข้ากับ PC-SB31 เรียบร้อยแล้วเราก็ต้องนำมาเซทจัมพ์เปอร์ที่บอร์ดทั้งสองอันโดยใน 8 อินพุทแรกเราจะเซทจัมพ์เปอร์ให้ใช้เป็นพอร์ท Aแล้ว อีก 8 อินพุทหลังเราจะเซทให้ใช้พอร์ท ดังรูปที่ 3.13 และ 3.14



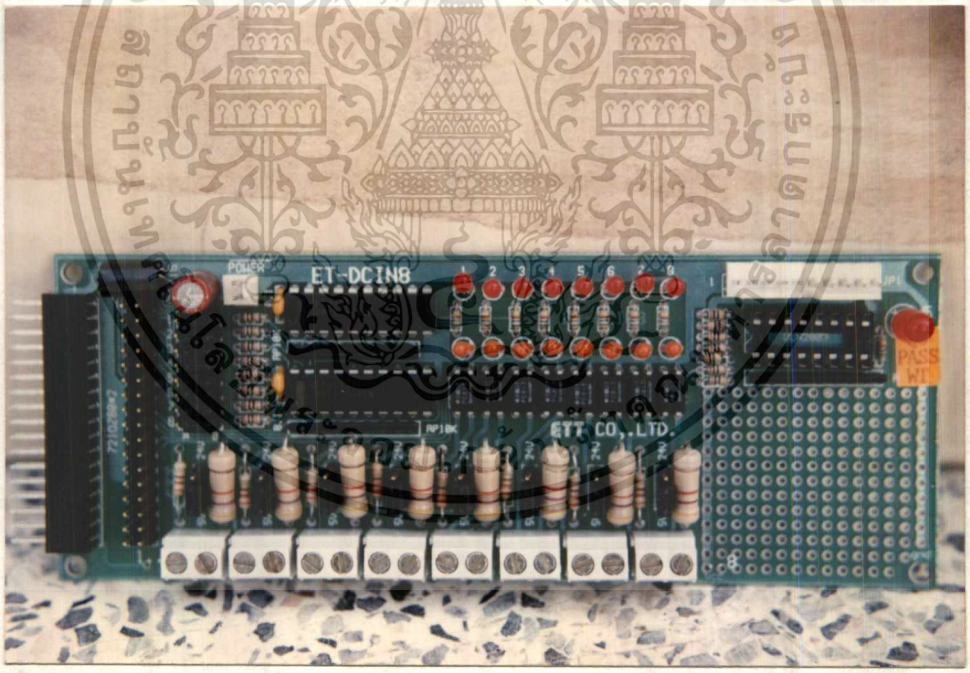
รูปที่ 3.12 การต่อ ET-DCIN8 เข้ากับ PC-SB31

จัมพ์เปอร์ชุดที่ 2 จะมีอยู่จำนวนชุดละ 8 ตัว จะมีไว้เพื่อเลือกว่าอินพุทที่จะต่อเป็นระดับสัญญาณ 5 โวลท์DC หรือ 24 โวลท์DCดังรูปที่ 3.15 และ 3.16

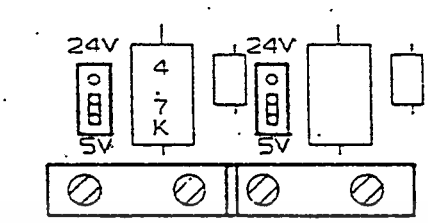




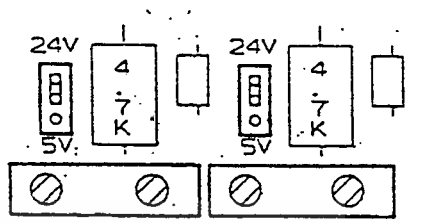
รูปที่ 3.13 การเซทจัมพ์เปอร์ให้ใช้พอร์ท A รูปที่ 3.14 การเซทจัมพ์เปอร์ให้ใช้พอร์ท B



รูปที่ 3.15 บอร์ด ET-DCIN8



รูปที่ 3.16 เซทให้รับอินพุท 5 โวลท์DC



รูปที่ 3.17 เซทให้รับอินพุท 24 โวลท์DC

3.4. ชุดควบคุมเอาท์พุทแบบใช้โซลิทสเตจรีเลย์ (ET-SSRAC Solid-State Relay)

ชุดควบคุมเอาท์พุทนี้จะใช้ควบคุมเอาท์พุทแบบ 8 ช่องสัญญาณ และตัวมันเองสามารถขยายช่องสัญญาณได้ถึง 24 ช่องสัญญาณ โดยการต่อพ่วงกัน แต่ในที่นี้เราใช้เพียง 8 ช่องสัญญาณ ส่วนพอร์ทที่ใช้จะใช้ทางพอร์ท C

เหตุที่ใช้ชุด ET-SSRAC ก็เนื่องมาจากอุปกรณ์ทางด้านเอาท์พุทที่จะต่อนั้นมีลักษณะส่วนใหญ่ใช้กับไฟ AC 220V เพราะฉะนั้นจึงเลือกเอา ET-SSRAC มาทำเป็นชุดเอาท์พุทเพราะว่าใช้อุปกรณ์ของโซลิดสเตจรีเลย์ หรือเอสเอสอาร์ (SSR) ซึ่งจะใช้แทนระบบการเคลื่อนไหวทางกลหรือระบบรีเลย์แบบเดิม ซึ่งจะเกิดประกายไฟที่หน้าสัมผัสได้

3.4.1. ข้อดีของไตรแอค

- การทำงานจะต่อวงจรที่โวลท์เตจขณะเป็นศูนย์หรือใกล้จุดศูนย์
- มีอายุการทำงานที่ยาวนานมาก
- ไม่มีเสียงรบกวนเกิดขึ้นในเวลาทำงาน
- สามารถต่อร่วมกับระบบไมโครโปรเซสเซอร์ได้ง่าย
- สามารถตัดต่อวงจรได้อย่างรวดเร็ว
- ไม่มีส่วนเคลื่อนไหวทางกลในการทำงาน

- ไม่เกิดอาการเบาวส์ (Bounce) ที่หน้าสัมผัส ซึ่งจะทำให้เกิดสัญญาณรบกวน
- ไม่เป็นตัวก่อให้เกิดสัญญาณรบกวนต่อระบบไฟฟ้าที่ตัวมันต่ออยู่

3.4.2 ข้อควรระวังในการใช้ไตรแอค

- ไม่สามารถใช้กับวงจรที่มีโวลท์เตจสูงมาก ๆ ได้
- ไม่สามารถใช้เอสเอสอาร์สำหรับ(AC Line) หรือDC Line) ได้ในตัวเดียวกัน
- เวลาใช้งานจะเกิดความร้อนขึ้นที่ตัวเอสเอสอาร์ จึงจำเป็นต้องติดตั้งฮีทซิงค์ (Hest Sink) ช่วยในการระบายความร้อน
- ตัวเอสเอสอาร์จะเสียโดยง่าย ถ้าเกิดการลัดวงจรขึ้น จึงควรระมัดระวัง

3.4.3 การทำงานของ ET-SSRAC แบ่งเป็น 2 ส่วนใหญ่ๆ คือ

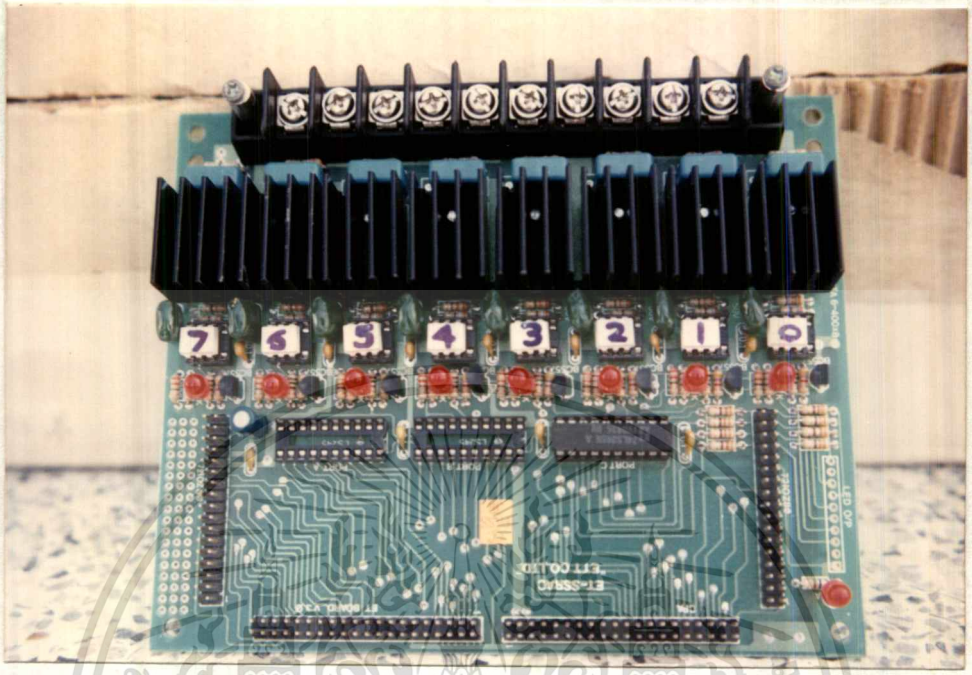
- ภาคอินพุท จะรับสัญญาณจากพอร์ท 8255 ซึ่งส่งมาให้ 74LS245 ซึ่งเป็นบัฟเฟอร์ (Buffer) และมาขับให้ BC557 ทำงานให้ ไอซี Optical photo MOC3082 เปลี่ยนจากสัญญาณไฟฟ้าเป็นสัญญาณแสงส่งไปภาคเอาท์พุทอีกทีหนึ่ง ซึ่งจะเป็นการตัดขาดภาคอินพุทและเอาท์พุทออกจากัน
- ภาคเอาท์พุท จะรับสัญญาณจาก MOC3082 ซึ่งจะเป็นการทำงานในรูปลักษณะZero switching โดยจะต่อวงจรขับTriacต่อเมื่อสัญญาณไฟเอซีใกล้เคียงศูนย์โวลท์

3.4.4 การต่อใช้งานร่วมกับ PC-SB31

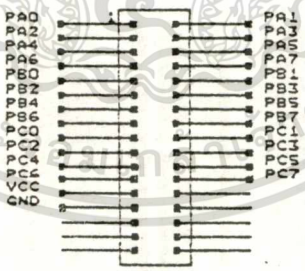
ให้ต่อ ET-SSRAC เข้ากับ 72IO ของบอร์ด ET-DCIN8 ซึ่งบอร์ด ET-DCIN8 ต่อพ่วงอยู่ด้วยกัน 2 ตัวอยู่แล้ว โดยเรานำ ET-SSRAC ต่อพ่วงเข้าไปอีก โดยที่ ET-DCIN8 มีการใช้พอร์ท A และ B ไปแล้วนั้น ใน ET-SSRAC นี้ก็จะใช้พอร์ท C โดยที่เราจะนำ ไอซี 74LS245 ไปใส่ไว้ที่ซอคเก็ตที่เขียนว่าพอร์ท C

3.5 ชุดแสดงผลแบบ Dot Matrix LCD

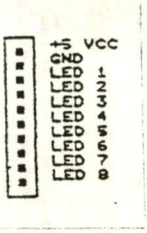
ในภาคแสดงผลนั้น เราจะใช้LCDเป็นตัวแสดงผล LCDที่ใช้เป็นของบริษัท ETT ขนาด 4 แถว 16 หลัก ซึ่งสามารถแสดงผลได้สูงสุด 84 ตัวอักษรต่อการแสดงผลเต็มหน้าจอ



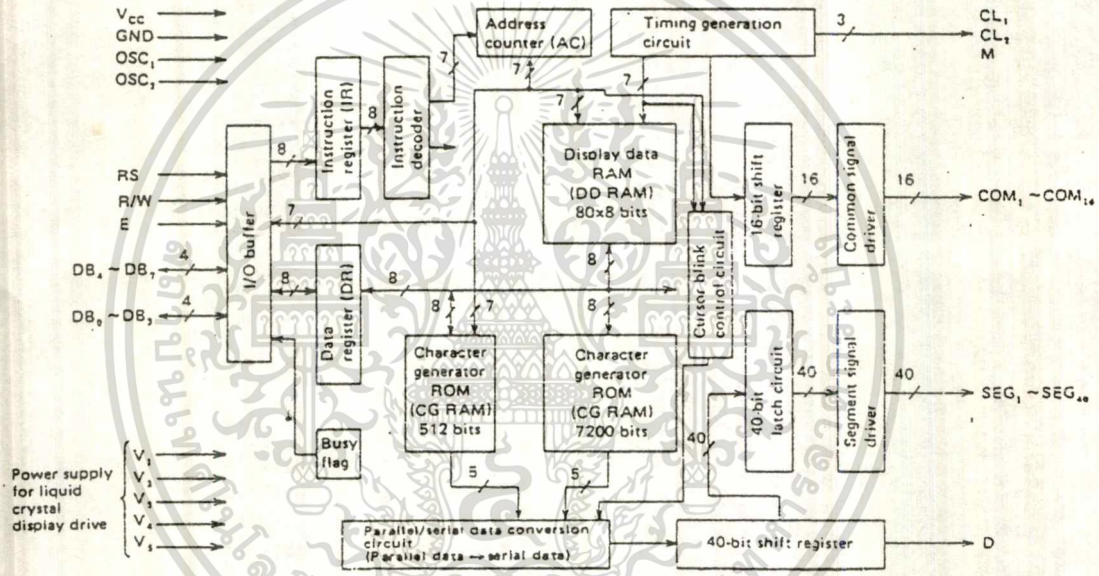
รูปที่ 3.18 บอร์ด ET-SSRACกับการใส่74LS245



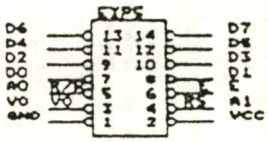
รูปที่ 3.19 การต่อขาสัญญาณแบบ 72IO ขนาด 34 ขาบน ET-SSRAC



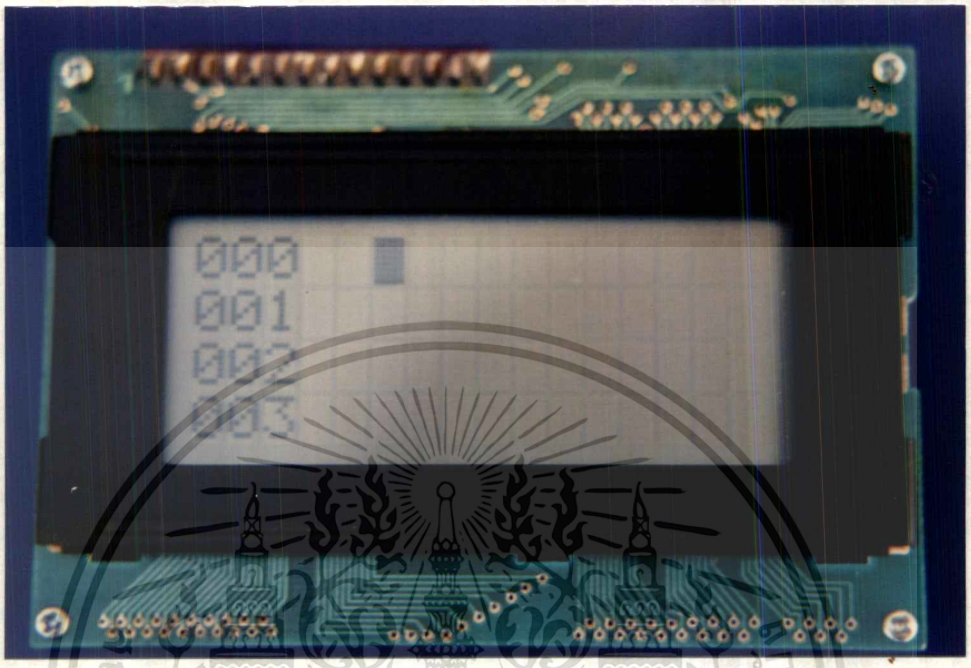
รูปที่ 3.20 การต่อขาสัญญาณของแอลอีดี (LED) เอาท์พุท



รูปที่ 3.21 โครงสร้างภายในของ HD44780



รูปที่ 3.22 ขาสัญญาณสำหรับการต่อ LCD โมดูล



รูปที่ 3.23 ชุดแสดงผลแบบ Dot Matrix LCD

ส่วนประกอบของ LCD โมดูล แบ่งได้ดังนี้

3.5.1 Dot Matrix LCD

เป็นตัวแสดงผลให้เรามองเห็นในลักษณะการปิดและเปิดตัวเองกับแสง ก็คือส่วนของที่เป็นตัวกระจกบรรจุผลึก

3.5.2. ไดร์เวอร์ (Driver)

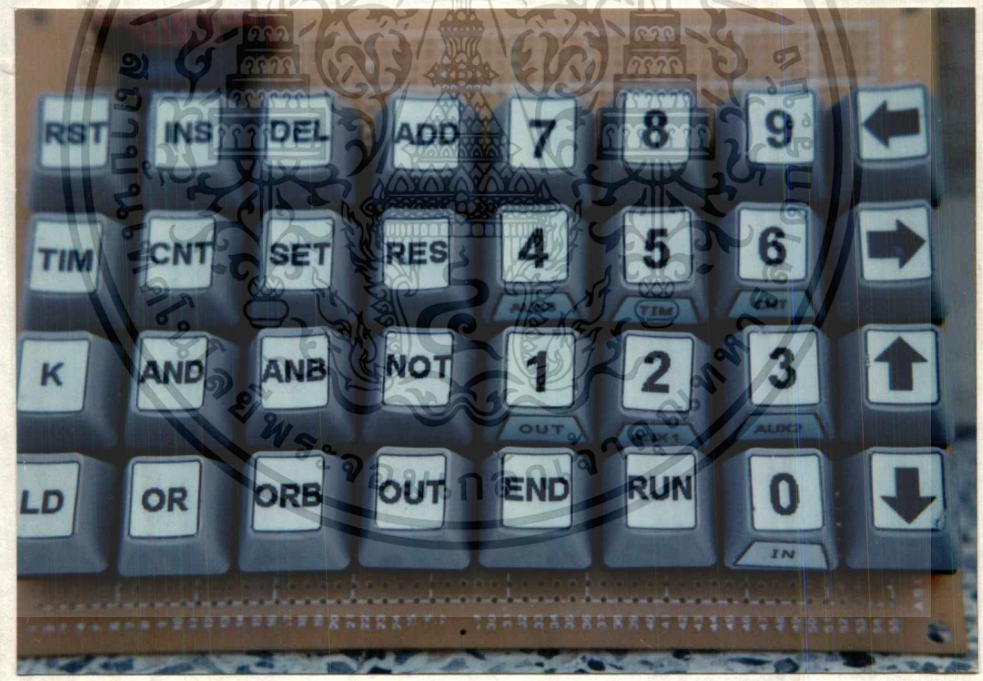
เป็นตัวรับสัญญาณจากตัวควบคุมมาขับ LCD อีกทีหนึ่งโดยมีเบอร์ที่ใช้ควบคุมคือ HD44100H

3.5.3. คอนโทรลเลอร์ (Controller)

เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาจัดการควบคุม LCD โมดูลให้ทำงานแสดงผลต่างๆ เช่น การลบจอภาพ การเกิดตัวอักษร เป็นต้น โดยมีเบอร์ไอซีที่ใช้คือ HD 44780 ซึ่งจะใช้ในแบบคาร์แรคเตอร์ LCDโมดูล (Character LCD Module)

HD44780 เป็นไอซีแบบ แอลเอสไอ (LSI) โดยแสดงผลในรูปตัวอักษรหรือสัญญลักษณ์ต่างๆ สามารถต่อใช้งานแบบ 4 บิท หรือ 8 บิทก็ได้ ถ้าต่อแบบ 4 บิท จะต่อใช้งานที่ DB7-DB4 เท่านั้น โดยข้อมูลครั้งแรกที่ส่งนั้น HD44780 จะถือเป็นข้อมูล 4 บิทบน และข้อมูลที่ส่งต่อมานั้นเป็นข้อมูล 4 บิทล่าง

การต่อร่วมกับ PC-SB31 นั้นจะมีคอนเน็คเตอร์สำหรับต่อ LCD อยู่แล้ว เราเพียงนำคอนเน็คเตอร์ต่อเข้ากับ LCD โมดูล และ PC-SB31 เข้าด้วยกัน



รูปที่ 3.24 ชุดรับข้อมูลทางคีย์บอร์ด

3.6 คีย์บอร์ด(Keyboard)

คีย์บอร์ดที่ได้ทำขึ้นนี้เราจะใช้ไอซีเบอร์ 74LS138 ซึ่งเป็นไอซีทีทีแอล ชนิด 3 อินพุท 8 เอาท์พุทดีโคดเดอร์ โดยเราจะใช้ พอร์ท 1 ของ 8031AH โดยให้ 3 บิทล่าง คือ P1.0-P1.2 เป็นเอาท์พุทสแกนทางด้านโรว์ (Row) จาก Y_0-Y_7 จะได้ทั้งหมด 8 แถว และจะใช้ 4 บิทบน คือ P1.4-P1.7 เป็นอินพุทรับสัญญาณจากโรว์จะได้คีย์ทั้งหมด 8×4=32คีย์ ดังรูป 3.24

บทที่ 4

การทดลองและผลการทดลอง

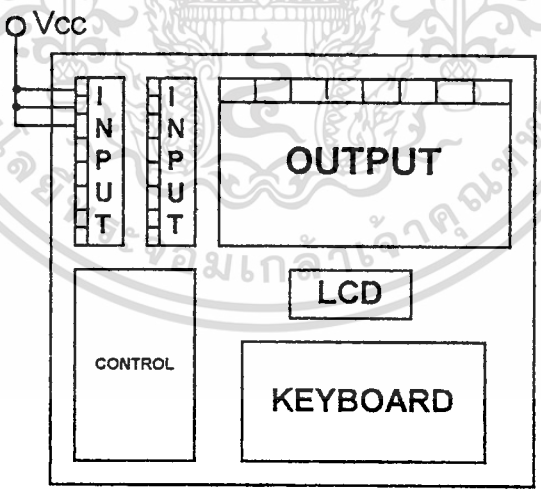
ในการทดลองของ PLC ชุดนี้ จะใช้ชุดทดลองที่เป็นชุดจำลองรถส่งของจำนวน 2 คัน รับของจากที่เดียวกันส่งให้ 2 ที่ โดยชุด PLC นี้จะควบคุมการเดินหน้าและถอยหลังของรถส่งของทั้ง 2 คัน และสับรางการทำงานตลอดจนการนำของใส่รถและการนับจำนวนของที่ต้องการใส่ในรถ ซึ่งชุดทดลองนี้สามารถใช้อุปกรณ์ทุกอย่างที่มีในตัว PLC คือ อินพุท เอาท์พุท ไทม์เมอร์และเคาท์เตอร์ จึงจะเรียกได้ว่าเป็นชุดทดลองที่ค่อนข้างสมบูรณ์

การทดลองโปรแกรมควบคุมรถส่งของ

ลำดับขั้นการทดลอง

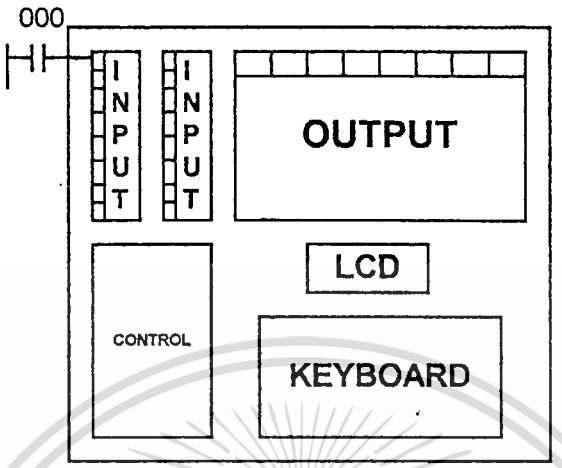
4.1.ต่อสายจากชุด PLC มาที่ชุดทดลองรถส่งของดังนี้

4.1.1.ทำการต่อสายไฟเลี้ยงเข้าที่ขั้วอินพุท 0,1และ2



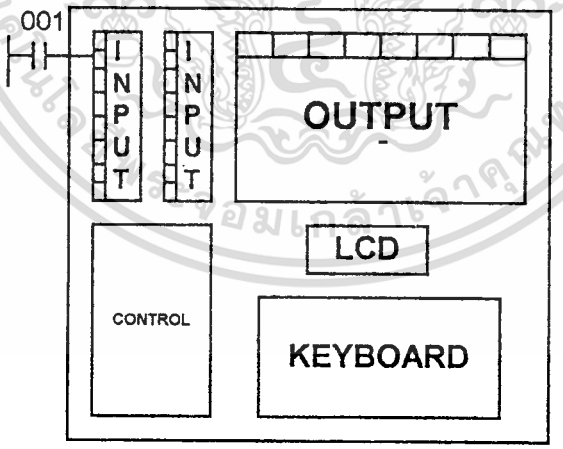
รูปที่ 4.1 การต่อสายไฟเลี้ยง

4.1.2.ต่ออินพุท 0 เข้าที่สวิทช์ 000 บริเวณท่าส่งของ



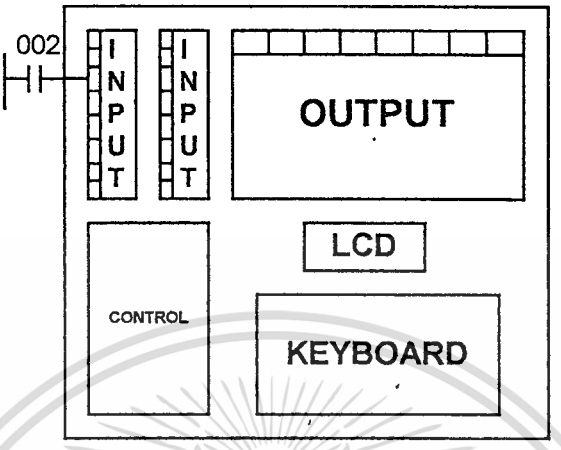
รูปที่ 4.2 การต่ออินพุท 0 เข้าที่ สวิทช์ 000

4.1.3.ต่ออินพุท 1 เข้าที่สวิทช์ 001 บริเวณท่ารับของที่ 1



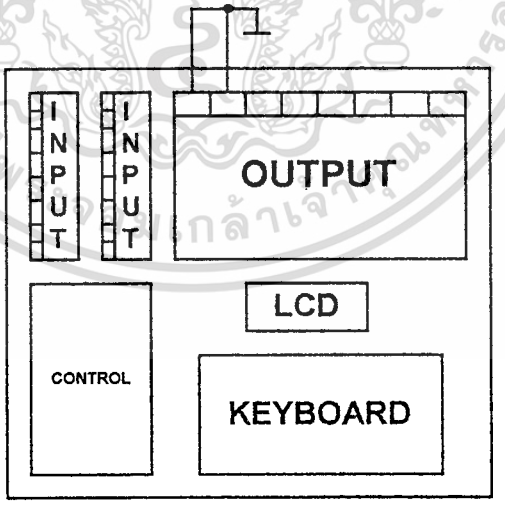
รูปที่ 4.3 การต่ออินพุท 1 เข้าที่ สวิทช์ 001

4.1.4.ต่ออินพุท 2 เข้าที่สวิทช์ 002 บริเวณท่ารับของที่ 2



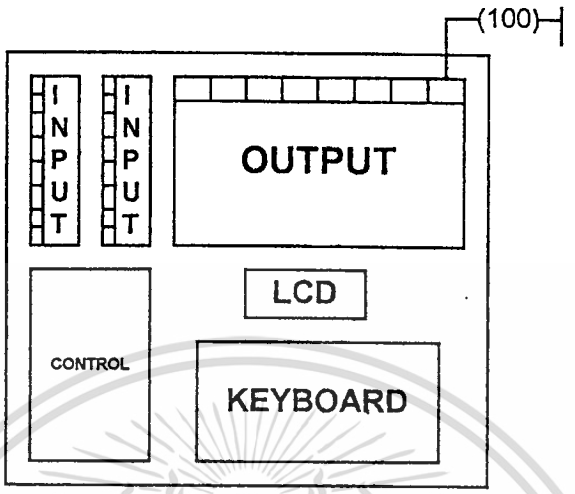
รูปที่ 4.4 การต่ออินพุท 2 เข้าที่ สวิทช์ 002

4.1.5.ต่อสาย จุดร่วม ของชุดเอาท์พุทเข้าที่ จุดร่วม ของชุดรถส่งของ



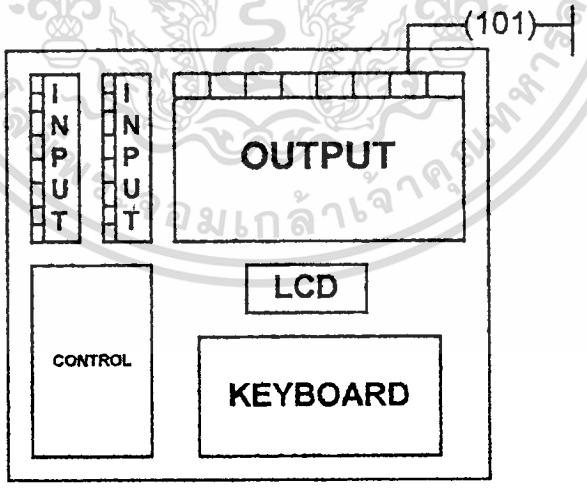
รูปที่ 4.5 การต่อสาย จุดร่วม ของชุด เอาท์พุทเข้าที่รถส่งของ

4.1.6.ต่อสายเอาท์พุท 0 เข้าที่มอเตอร์ 100 เพื่อให้รถสีเขียวเดินหน้า



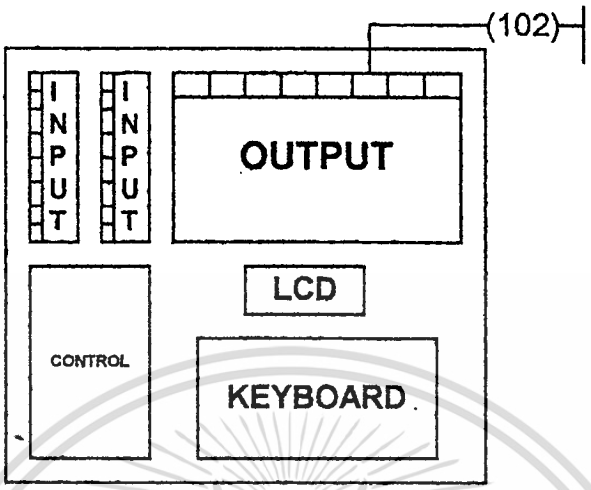
รูปที่ 4.6 การต่อเอาท์พุท 0 เข้าที่ มอเตอร์ 100

4.1.7.ต่อสายเอาท์พุท 1 เข้าที่มอเตอร์ 101 เพื่อให้รถสีเขียวถอยหลัง



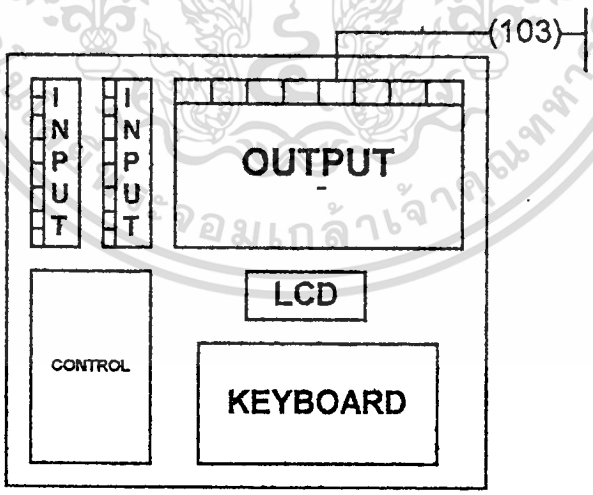
รูปที่ 4.7 การต่อเอาท์พุท 1 เข้าที่ มอเตอร์ 101

4.1.8.ต่อสายเอาท์พุท 2 เข้าที่มอเตอร์ 102 เพื่อให้รถสีฟ้าเดินหน้า



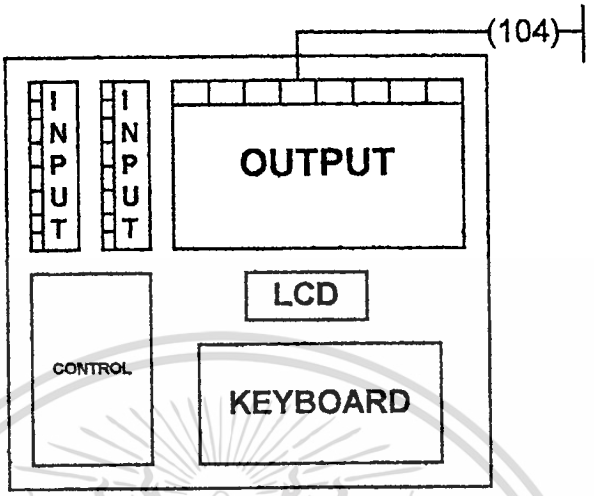
รูปที่ 4.8 การต่อเอาท์พุท 2 เข้าที่ มอเตอร์ 102

4.1.9.ต่อสายเอาท์พุท 3 เข้าที่มอเตอร์ 103 เพื่อให้รถสีฟ้าถอยหลัง



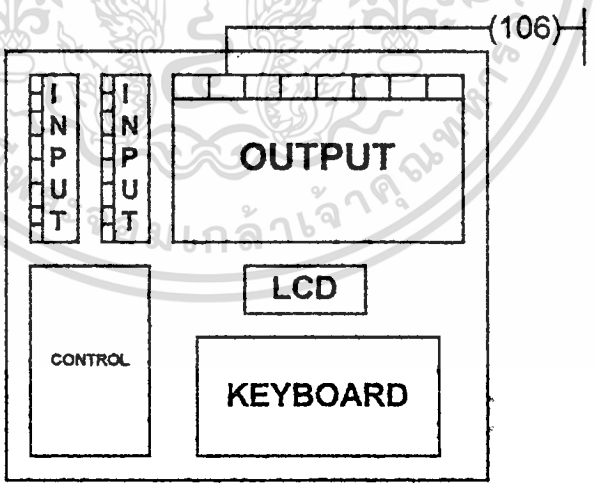
รูปที่ 4.9 การต่อเอาท์พุท 3 เข้าที่ มอเตอร์ 103

4.1.10.ต่อสายเอาท์พุท 4 เข้าที่สวิทช์แม่เหล็ก 104 เพื่อสับราง



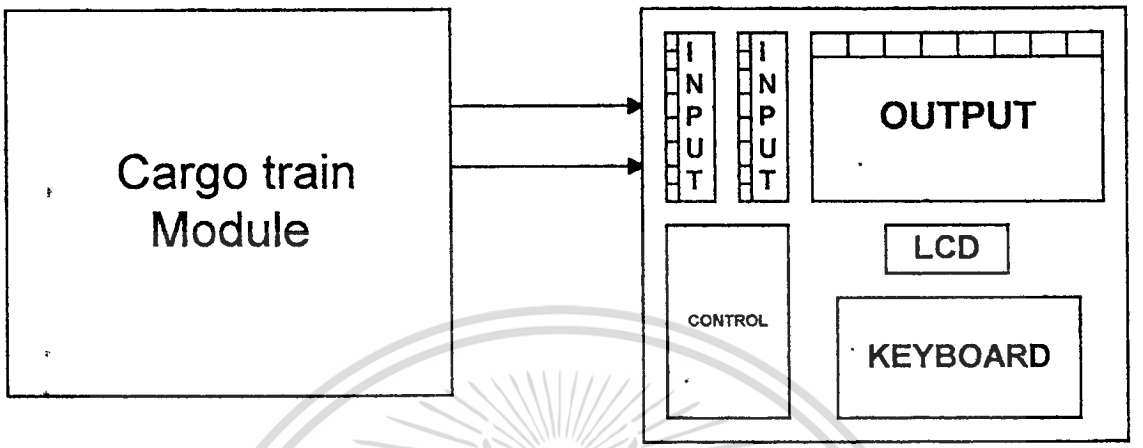
รูปที่ 4.10 การต่อเอาท์พุท 4 เข้าที่ สวิทช์แม่เหล็ก 104

4.1.11.ต่อสายเอาท์พุท 6 เข้าที่สวิทช์แม่เหล็ก 106 เพื่อส่งของ



รูปที่ 4.11 การต่อเอาท์พุท 6 เข้าที่ สวิทช์แม่เหล็ก 106

4.1.12.ต่อสายไฟเลี้ยงเข้าที่บอร์ด PLC



รูปที่ 4.12 การต่อสายไฟเลี้ยงเข้าที่ชุดควบคุม

4.2 การออกแบบโปรแกรมในการทดลอง

ในการทดลอง ได้เขียนแลดเดอร์ไดอะแกรมและภาษาสเต็ป ดังแสดงในภาคผนวก ง

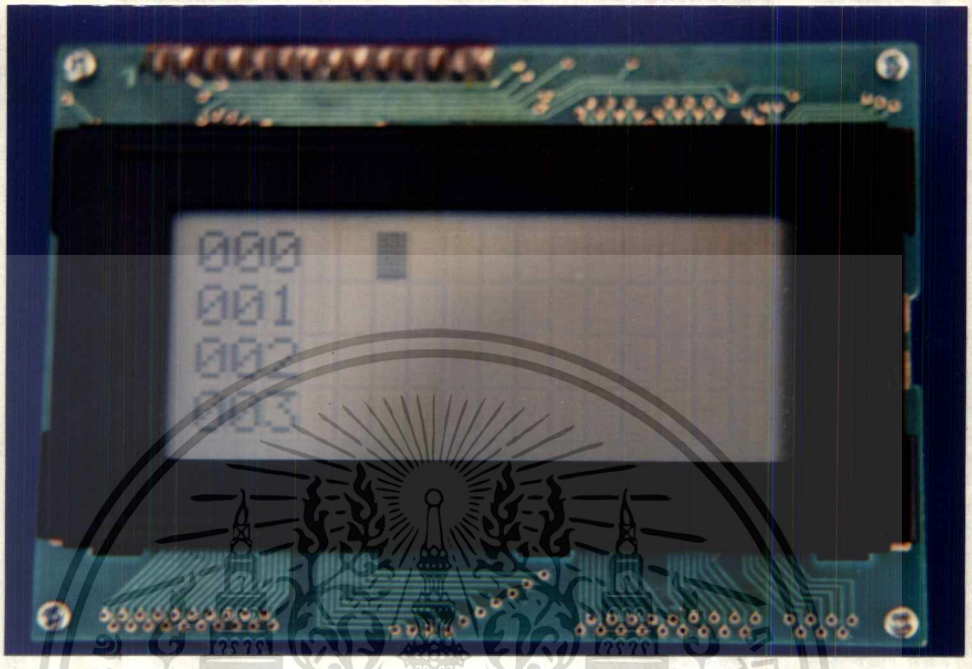
4.3.นำโปรแกรมภาษาสเต็ปที่ได้มาป้อนเข้าที่บอร์ด PLC

4.3.1เปิดสวิทช์ป้อนไฟเข้าเครื่อง

4.3.2ที่จอแสดงผลจะปรากฏดังรูปที่ 4.13

4.4.3.จากนั้นเริ่มป้อนโปรแกรมตามข้อ 4.3 ลงไป

4.4.4.เมื่อป้อนเสร็จแล้วกดปุ่ม RUN

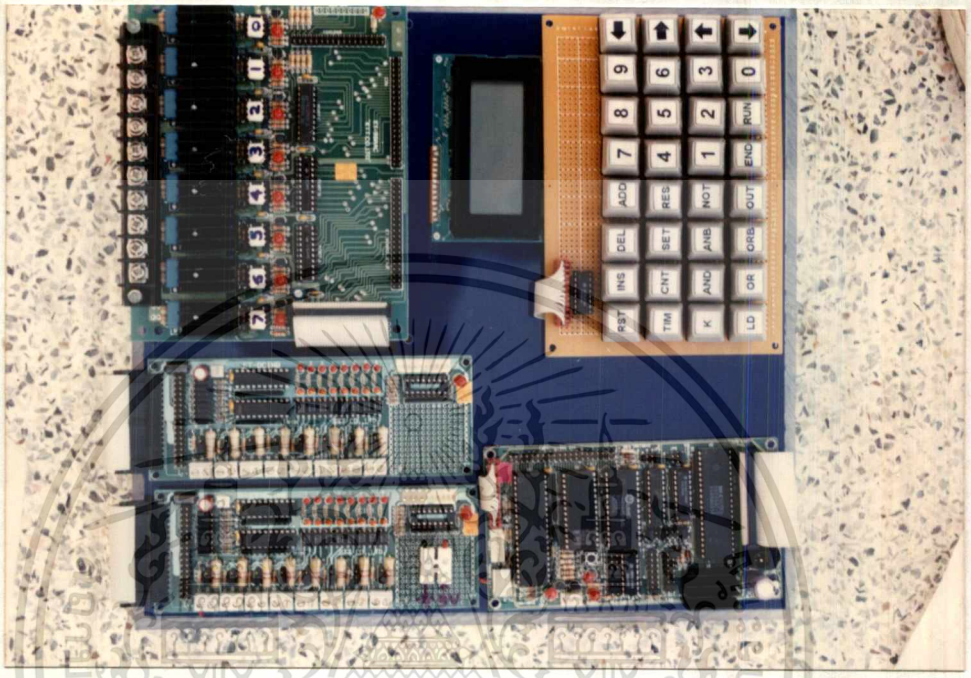


รูปที่ 4.13 การแสดงข้อความพร้อมรับข้อมูล

4.5 สรุปผลการทดลอง

จากชุดลองนี้ได้ทำงานตามที่เราโปรแกรมไว้อย่างถูกต้อง แต่บางครั้งอาจเกิดปัญหาขึ้นได้ เช่น เมื่อให้โปรแกรมทำงานแล้วเกิดผิดพลาดเราสามารถแก้ไขโปรแกรมได้ สำหรับการแก้ไขโปรแกรม โปรดดูที่ภาคผนวก ค

สำหรับชุดทดลองนี้เมื่อทำงานแล้วจะมีปัญหาบางเล็กน้อยคือ รถไฟสีฟ้าจะวิ่งเร็วกว่าสีเขียว มากจนบางครั้งทำให้ตกรางได้ ส่วนรถไฟสีเขียวซึ่งวิ่งช้าอาจจะหยุดวิ่งได้ เราสามารถเปลี่ยนแปลง จำนวนของที่จะใส่ได้โดยแก้โปรแกรมบรรทัดที่ 084 กำหนดค่า K ใหม่ได้



รูปที่ 4.14 ชุดทดลองรับส่งของโดยใช้ PLC ควบคุม

บทที่ 5

บทสรุปและวิจารณ์ผลการทดลอง

โครงงานนี้ได้เสนอการทำชุดทดลอง PLC ขึ้น ซึ่งใช้อุปกรณ์ไมโครคอนโทรลเลอร์เบอร์ 8031 ของอินเทลมาควบคุมการทำงาน โดยการเขียนโปรแกรมควบคุมการทำงานขึ้น เพื่อให้สามารถใช้งานได้เหมือนกับชุด PLC ทั่วไป การนำไมโครคอนโทรลเลอร์มาประยุกต์ใช้งานนั้นสามารถลดอุปกรณ์ทางด้านฮาร์ดแวร์ได้มาก และสามารถลดสัญญาณรบกวนซึ่งเป็นปัญหาอย่างมากลงได้มาก

จากการทดลองการทำงานของชุด PLC นี้ ทำให้ทราบถึงปัญหาที่เกิดขึ้นซึ่งสามารถสรุปได้ดังนี้คือ

5.1 ปัญหาการทดลอง

1. ปัญหาจากชุดวงจรควบคุมที่นำมาใช้ เนื่องจากขั้วที่ต่อมาจากชุดควบคุม มีสัญญาณที่ต้องการไม่ครบจึงต้องดัดแปลงเพิ่มขึ้นอีก 1 ชุด คือสัญญาณรีเซตซึ่งเพิ่มที่ขั้วต่อสายพอร์ท 1

2. ปัญหาจากชุดเอาท์พุท เนื่องจากที่ชุดอินพุทนั้นได้นำสัญญาณของชุดเอาท์พุทไปใช้งานด้วยจึงทำให้เกิดปัญหาบ้างเล็กน้อย แก้ไขโดยการตัดสัญญาณของชุดเอาท์พุทที่พอร์ทอินพุทนำไปใช้งานออก

3. ชุดเอาท์พุทจะใช้ไตรแอค ซึ่งจำกัดสัญญาณรบกวนได้เป็นอย่างดี จึงตัดปัญหาในส่วนนี้ได้เป็นอย่างมาก แต่มีข้อเสียคือไม่สามารถใช้กับอุปกรณ์ที่เป็นไฟตรงได้ จึงแก้ปัญหาโดยการต่อวงจรเปลี่ยนแรงดันไฟสลับให้เป็นแรงดันไฟตรงที่หลังชุดเอาท์พุทนี้

4. ข้อมูลเกี่ยวกับ PLC ที่หาได้นั้น หลาย ๆ เล่มในจุดใหญ่จะเขียนเหมือนกัน แต่ในรายละเอียดของแต่ละเครื่องจะไม่เหมือนกัน จึงได้ตัดสินใจใช้มาตรฐานของเครื่องPLC-1 ของบริษัท เอที รีเซอร์ส ซึ่งมีข้อมูลละเอียดพอสมควร

5.2 ข้อเสนอแนะและแนวทางการพัฒนา

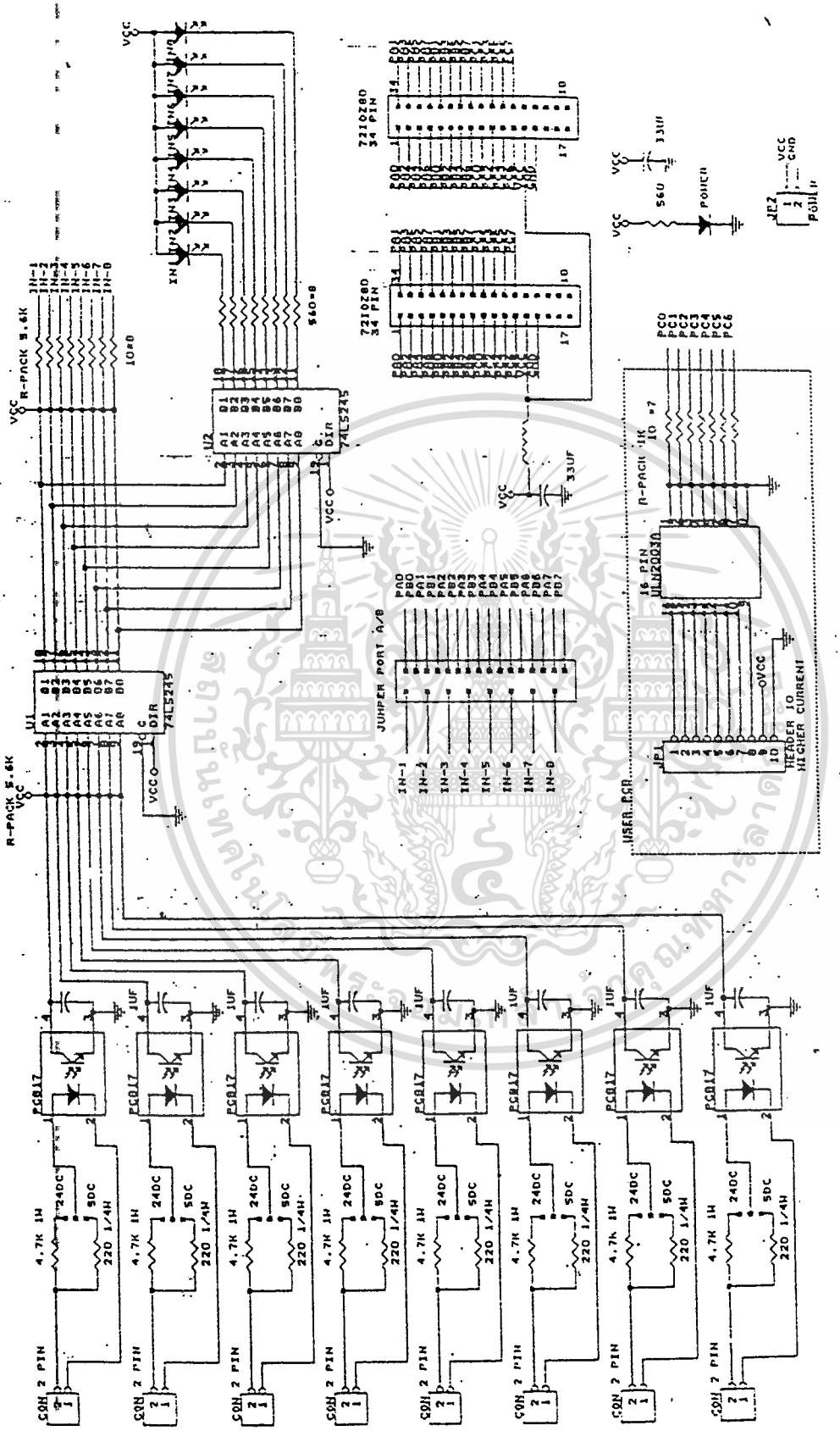
1. เนื่องจากมาตรฐานที่ใช้ ขีดความสามารถยังไม่สูงมากนักจึงควรพัฒนาโดยใช้มาตรฐานของเครื่องอื่น อย่างเช่น ในตระกูลของมิตซูบิชิ ซึ่งมีความสามารถมาก แต่หาข้อมูลได้ยากมาก ผู้ที่จะทำการพัฒนาต่อควรจะหาข้อมูลให้มากกว่านี้

2. ชุดเอาท์พุทสามารถใช้กับไฟสลับได้ดีไม่มีสัญญาณรบกวน แต่ถ้าใช้กับไฟตรงจะใช้ได้ยากมาก จึงควรเพิ่มชุดเอาท์พุทที่ใช้กับไฟตรงขึ้นมาอีก 1 ชุด เพื่อให้ใช้งานได้อย่างสะดวก

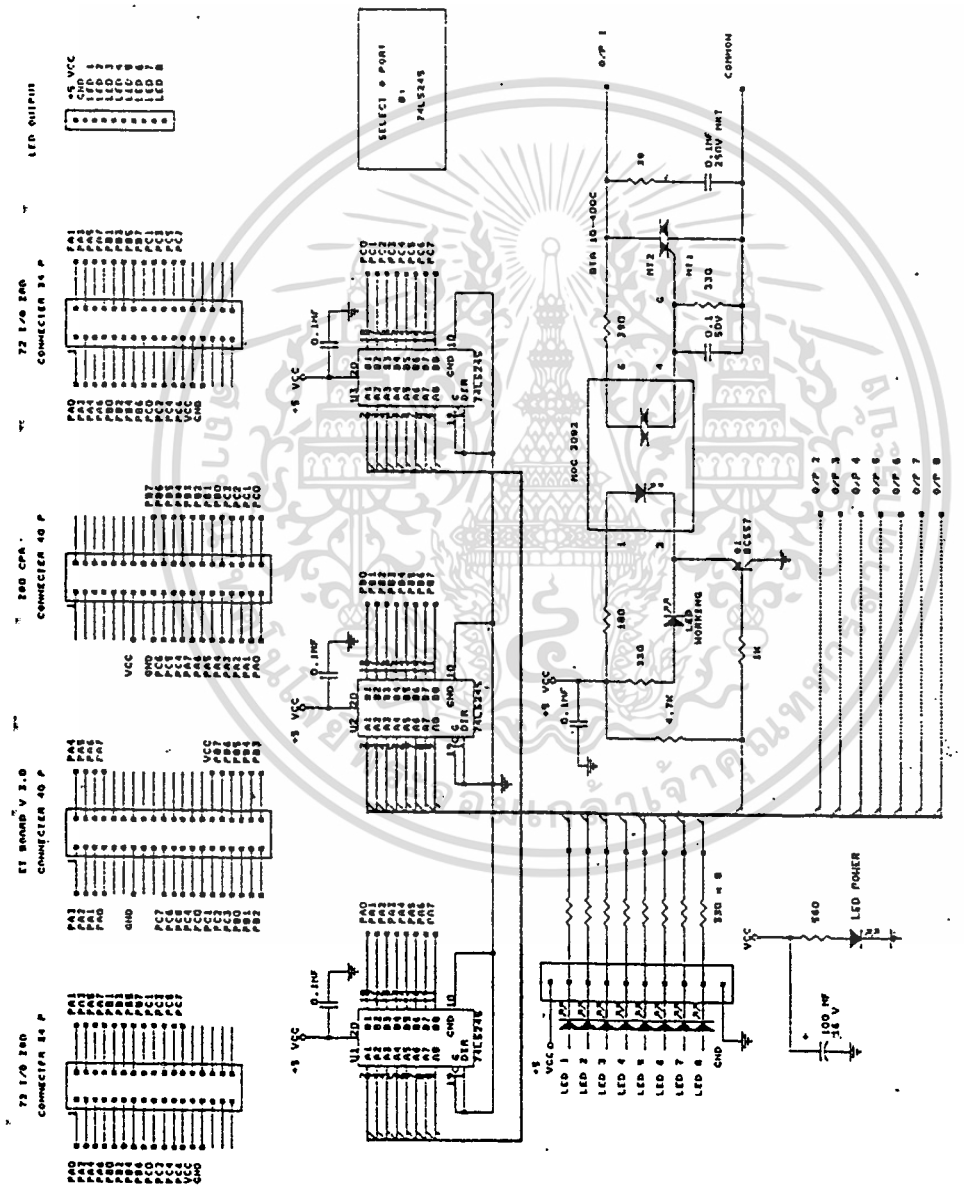
3. ชุด PLC นี้เป็นเครื่องสแตนอโลน (stand alone) คือ ใช้งานได้เพียงตัวเดียว ไม่สามารถต่อเข้ากับเครื่องคอมพิวเตอร์ส่วนบุคคลได้ จึงควรเพิ่มชุดติดต่อระหว่างเครื่อง PLC กับคอมพิวเตอร์ส่วนบุคคล เพื่อสะดวกในการเขียนโปรแกรมใหญ่ ๆ และยังสามารถเพิ่มภาษาวงจรคือเขียนรูป แล้วให้คอมพิวเตอร์ส่วนบุคคลแปลงเป็นภาษาแลดเดอร์ (ladder) ก่อนแล้วส่งข้อมูลมาที่เครื่อง PLC จะทำให้เขียนโปรแกรมได้อย่างสะดวกขึ้นมาก

ภาคผนวก ก

วงจรต่างๆของเครื่อง PLC



วงจรบอร์ด ET-DCIN8



วงจรบอร์ด ET-SSRAC

MAXIMUM RATINGS

Electric maximum ratings

Item	Symbol	Min.	Max.	Unit	Remarks
Power supply for logic	$V_{DD} - V_{SS}$	Refer to individual specification		V	
Power supply for LCD drive	$V_{DD} - V_O$			V	
Input voltage	V_I			V	
Static electricity		—	100	V	See note

Note Electro-static discharge resistance is tested by charging a condenser with a capacity of 200pF and discharging it by contact with an interface connector pin.

Environmental conditions

Item	Operating		Non-operating		Remarks
	Min.	Max.	Min.	Max.	
Ambient temperature	Refer to individual specifications				No dew
Humidity	Note				
Vibration	—	4.9m/s² (0.5G)	—	19.6 m/s² (2G)	
Shock	—	29.4 m/s² (3G)	—	490 m/s²(50G)	XYZ 3 directions
Corrosion gas	No corrosion gas				

Note Humidity conditions are as follows.

Number of dots / Ambient temperature (Ta)	Under 128 × 240	128 × 240 or over
Ta ≤ 40°C	95% RH max.	85% RH max.
Ta > 40°C (Below maximum temperature)	Below maximum absolute humidity of 40°C 95% RH	Below maximum absolute humidity of 40°C 85% RH

RELIABILITY CONDITIONS

LCD MODULE (Consumer Type)

Item	Conditions	Evaluation
High Temperature Operation	Operating 96 – 100 Hrs at 50 ± 2°C surrounding temp.	No change is visiblein appearance nor function
Low Temperature Operation	Operating 96 – 100 Hrs at 0 ± 2°C surrounding temp.	
High Temperature Storage	Storage 96 – 100 Hrs at 60 ± 2°C surrounding temp. then storage 4 Hrs at normal condition (Power Off)	
Low Temperature Storage	Storage 96 – 100 Hrs at −20 ± 2°C surrounding temp. then storage 4 Hrs at normal condition (Power Off) No dew to be found	
Damp Pruof	Storage 96 – 100 Hrs at 40 ± 2°C and 90 – 95% RH surrounding condition, then storage 4 Hrs at normal condition (Power Off) No dew to be found.	

Note The above condition is only representative, and may, differ in case of customized specifications.

OPTICAL DATA

Ta = 25°C

Item	Symbol	Condition	Min.	Typ.	Max.	Unit	Notes to see
Viewing angle	$\phi 2 - \phi 1$	K = 1.4	—	20	—	deg.	—
Contrast ratio	K	$\phi = 25°$ $\theta = 0°$	—	2	—	—	—
Response time (rise)	t_r	$\phi = 25°$ $\theta = 0°$	—	250	400	ms	—
				150	250		—
Response time (fall)	t_f	$\phi = 25°$ $\theta = 0°$	—	250	400	ms	—
				150	250		—

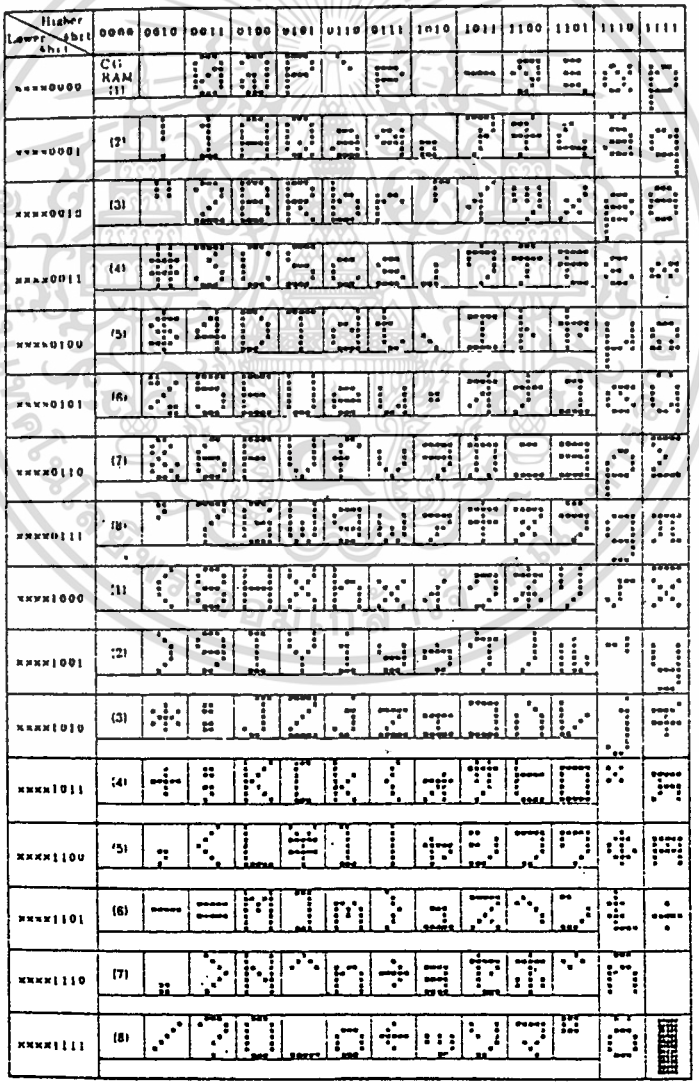
Instruction	Code										Description	Execution time (when fosc is 250 kHz) Note 1	Execution time (when fosc is 160 kHz) Note 2
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0			
Clear display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position (Address 0).	82 μs ~ 1.64 ms	120 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.	1μs	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.

ตารางคำสั่งของ HD44780

CHARACTER FONT TABLE

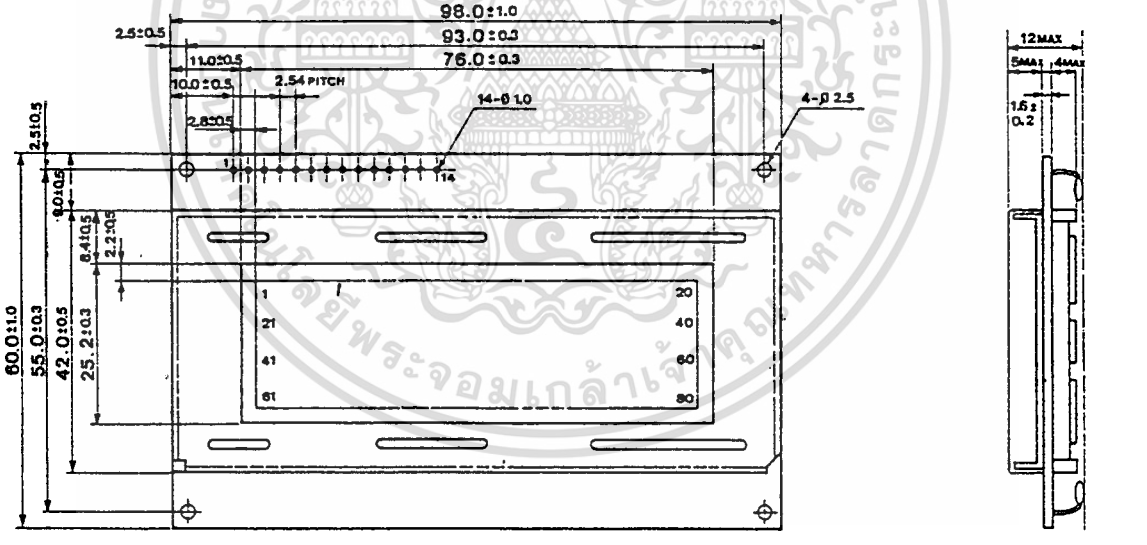


NOTE: CGRAM is a CHARACTER GENERATOR RAM having a storage function of character pattern which enable to change freely by user's program.

ตารางตัวอักษรใน LCD Module

DV-2004

No.	Signal	No.	Signal
1	VSS	2	VDD
3	V0	4	RS
5	R/W	6	E
7	DB 0	8	DB 1
9	DB 2	10	DB 3
11	DB 4	12	DB 5
13	DB 6	14	DB 7



ลักษณะทางกายภาพของจอ LCD

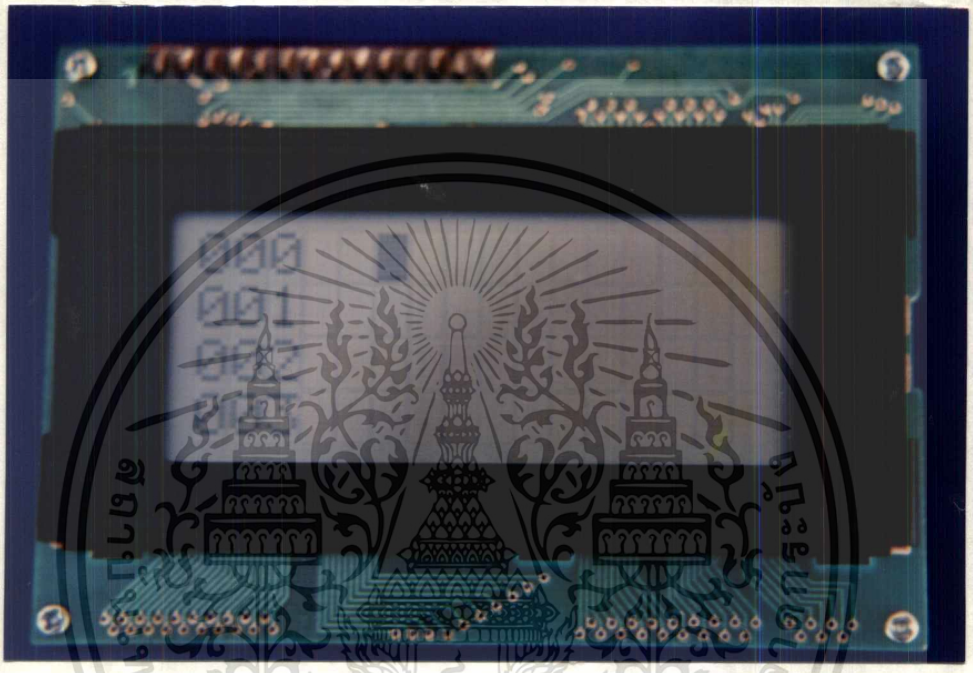
ภาคผนวก ข

การใช้งานเครื่อง PLC

การใช้งาน PLC-8051AH

ขั้นตอนการใช้งาน

เมื่อเปิดเครื่องใหม่ เครื่องจะแสดงข้อความ



รูปแสดงสภาวะรอคำสั่ง

การใช้งานพีแอลซีทำได้โดยสั่งงานทางคีย์บอร์ด ซึ่งมีอยู่ทั้งหมด 32 คีย์ ดังรูปที่ ซึ่งแบ่งออกได้ 4 กลุ่มคือ

กลุ่มที่ 1. คีย์ตัวเลข ใช้ สำหรับป้อนข้อมูลที่เป็นตัวเลข เช่น เลขที่บรรทัดของคำสั่ง หรือข้อมูลของคำสั่งนั้น ๆ

กลุ่มที่ 2. คีย์คำสั่ง ใช้สำหรับป้อนคำสั่งในโปรแกรมมีอยู่ 13 คีย์ คือ OUT, SET, RES, TIM, CNT, K, AND, ANB, LD, OR, ORB, NOT และ END

กลุ่มที่ 3 คีย์แก้ไข มีด้วยกัน 7 คีย์ คือ ADDR, INS, DEL,

กลุ่มที่ 4 คีย์ควบคุม ใช้สำหรับควบคุมการทำงานของเครื่อง มี 2 คีย์ คือ RST และ RUN

การใช้คีย์บอร์ดของพีแอลซี

1. คีย์ RST ใช้สำหรับรีเซทระบบ ในขณะที่เราต้องการให้เครื่องหยุดทำงาน ขณะใดขณะหนึ่ง

2. คีย์ ADDR ใช้สำหรับบอกเครื่องว่า เราจะป้อนโปรแกรมที่บรรทัดที่เท่าใด หรือว่าเราต้องการจะอ่านโปแกรมที่บรรทัดที่เท่าใด

3. คีย์ INS ใช้สำหรับการแทรกบรรทัดในการป้อนโปรแกรม โดยจะเลื่อนคำสั่งในบรรทัดปัจจุบันไปยังบรรทัดถัดไป และคำสั่งในบรรทัดถัดไปก็จะเลื่อนไปเรื่อยๆ ถ้าหากว่าโปรแกรมมีจำนวนมากกว่าเครื่องจะเก็บไว้ได้ คือเกิน 999 บรรทัด คำสั่งในบรรทัดสุดท้ายจะถูกลบหายไป

4. คีย์ DEL ใช้สำหรับลบคำสั่งในบรรทัดปัจจุบัน และเครื่องทำการเลื่อนคำสั่งบรรทัดถัดไปขึ้นมาแทน

5. คีย์ ใช้สำหรับเลื่อนขึ้นไปยังบรรทัดก่อนหน้า 1 บรรทัด

6. คีย์ ใช้สำหรับเลื่อนลงไปยังบรรทัดถัดไป 1 บรรทัด

7. คีย์ ใช้สำหรับเลื่อนเคอร์เซอร์ไปทางซ้าย 1 ตำแหน่ง

8. คีย์ ใช้สำหรับเลื่อนเคอร์เซอร์ไปทางขวา 1 ตำแหน่ง

9. คีย์คำสั่ง (COMMAND KEYS) ใช้สำหรับป้อนคำสั่งต่าง ๆ ในขณะทำการป้อนโปรแกรมให้แก่เครื่อง ซึ่งแบ่งออกเป็น 3 กลุ่มด้วยกันคือ

9.1 คีย์คำสั่งทำหน้าที่แทนคำสั่งต่างๆ ในภาษาบูลลีน ประกอบด้วยคีย์ OUT, SET, RES, K, AND, ANB, NOP, LD, OR, ORB และ END

9.2 คีย์ที่ใช้ร่วมกับคีย์คำสั่งอื่น เพื่อให้เกิดคำสั่งใหม่ในภาษาบูลลีน ซึ่งมีเพียง 1 คีย์คือ NOT เช่นเมื่อกดร่วมกับคีย์ LD จะเปลี่ยนคำสั่ง LD เป็นคำสั่ง LD-NOT แต่ถ้ากดคีย์ NOT อีกครั้ง เครื่องจะเปลี่ยนเป็นคำสั่ง LD-NOT ให้กับมาเป็นคำสั่ง LD อีกครั้ง

9.3 คีย์ที่ใช้ระบุคำสั่งและชนิดอุปกรณ์ ซึ่งจะประกอบด้วยคีย์ TIM และ CNT โดยถ้าเรากดคีย์นี้ให้โหมดคำสั่ง TIM และ CNT แต่ถ้าเรากดคีย์คีย์เหล่านี้ในโหมดของการรับข้อมูล จะทำให้เกิดอักษรตัว "T" และตัว "C" ตามลำดับ

10. คีย์ตัวเลข (NUMERIC KEYS) ใช้สำหรับการป้อนลำดับที่บรรทัดของโปรแกรมของโปรแกรมหรือข้อมูลที่เป็นตัวเลขทั้งหลายของโปรแกรม

11. คีย์ RUN ใช้สำหรับให้เครื่องทำงานตามโปรแกรมบูลลีนที่เราเขียนไว้

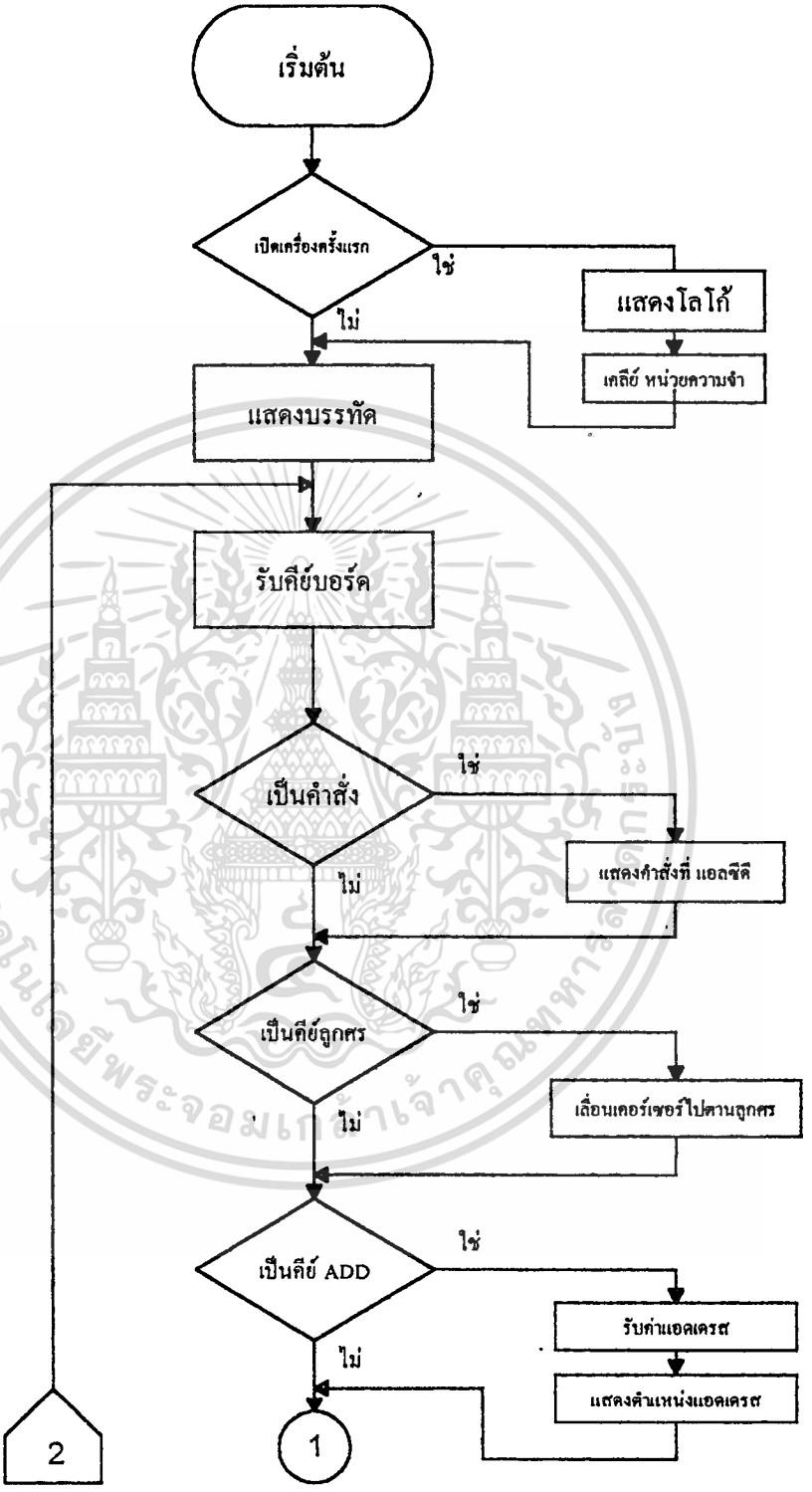
ตัวอย่างการป้อนโปรแกรม

กดคีย์	การแสดงผลบนจอแอลซีดี	หมายเหตุ
	PLC 8051 Press Any Key	1.เมื่อเราเปิดเครื่องครั้งแรกจะแสดงผลดังรูป
	001 _ 002 003 004	2.เมื่อกดปุ่มใดๆ เครื่องจะรอคำสั่งที่บรรทัด 001
LD	001 LD 002 _ 003 004	3. เมื่อเรากดคำสั่งใด เครื่องจะแสดงคำสั่งและจะไปรอรับหมายเลขอุปกรณ์ต่อไป
0 0 1	001 LD 001 002 003 004	4. เมื่อเราป้อนหมายเลขของอุปกรณ์ครบแล้วเครื่องจะไปรอรับคำสั่งใหม่ที่บรรทัดถัดไป
	001 LD 001 002 OUT 102 003 004	5.ถ้าเราป้อนโปรแกรมผิดก็ใช้คีย์ลูกศรไปที่ตำแหน่งนั้นแล้วคีย์ทับลงไปได้เลย
DEL	001 LD 001 002 003 004	6.เมื่อต้องการลบบรรทัดใดก็เลื่อลูกศร

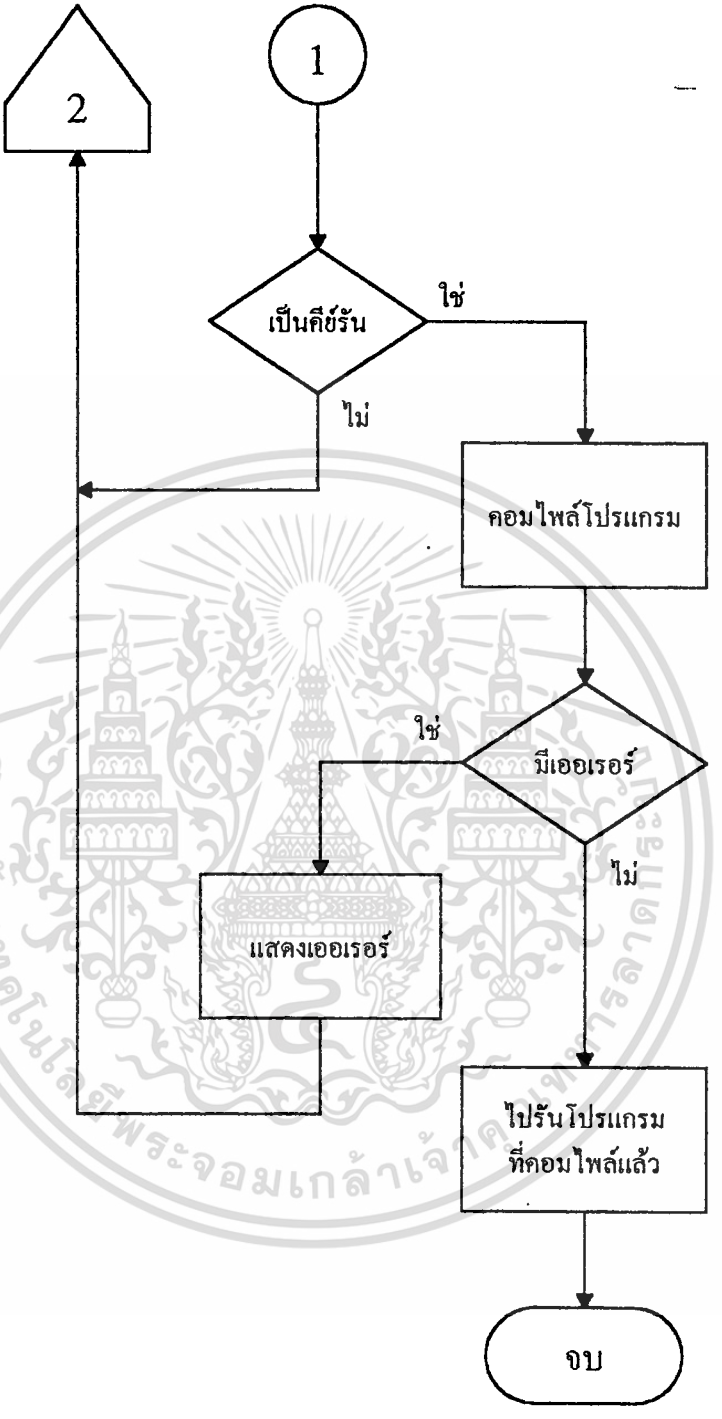
กดคีย์	การแสดงผลบนจอแอลซีดี	หมายเหตุ
INS	001 002 LD 001 003 004	ไปที่บรรทัดนั้นแล้วกดปุ่ม DEL 7.เมื่อต้องการเพิ่มบรรทัดก็ใช้ลูกศรเลื่อนแล้วกดปุ่ม INS จะเลื่อนไปบรรทัดถัดไปแล้วที่บรรทัดนั้นจะว่างใส่คำสั่งได้
ADDR	ENTER LINE NUMBRE 123	8.เมื่อจะไปบรรทัดใดกดปุ่ม ADDR แล้วเครื่องจะรอใส่หมายเลขบรรทัดเมื่อใส่แล้วกดปุ่ม ADDR อีกครั้งก็จะไปที่บรรทัดนั้น
END	001 LD 001 002 AND 002 003 OUT 101 004 END	9.เมื่อจบโปรแกรมให้กดปุ่ม END ทุกครั้ง

ภาคผนวก ค

โฟลว์ชาตท์โปรแกรมหลัก



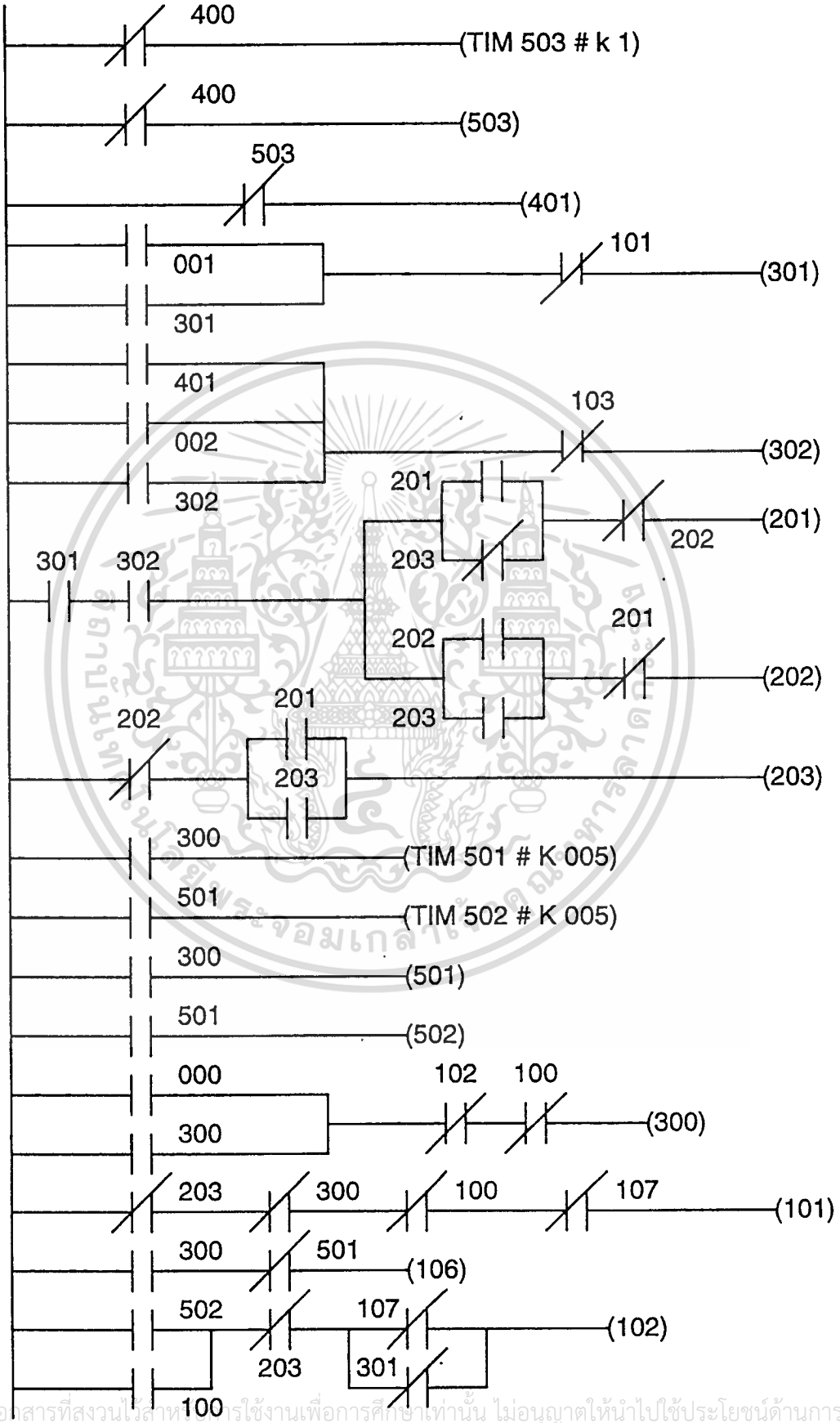
โฟลว์ชาตท์การทำงานของโปรแกรมหลัก

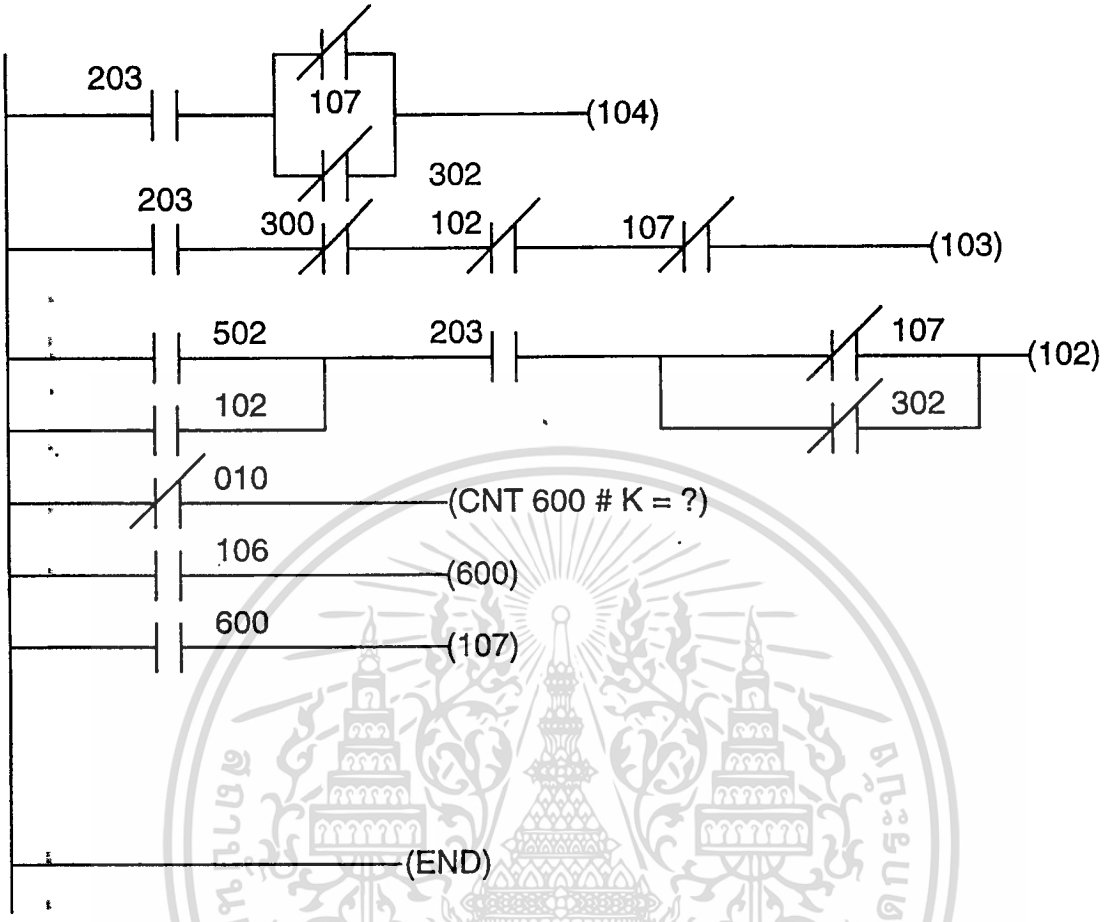


โฟลว์ชาตท์การทำงานของโปรแกรมหลัก (ต่อ)

ภาคผนวก ง

แลดเดอร์ไดอะแกรมและภาษาสเต็ป





โปรแกรมภาษาสเต็ป

000	LD-NOT	400
001	TIM	503
002	K	001
003	LD-NOT	400
004	OUT	503
005	LD-NOT	503
006	OUT	401
007	LD	001
008	OR	301
009	AND-NOT	101
010	OUT	301
011	LD	102
012	OR	302
013	OR	401
014	AND-NOT	103
015	OUT	302
016	LD	301
017	AND	302
018	LD	201
019	OR-NOT	203
020	ANB	
021	AND-NOT	202
022	OUT	201

023	LD	301
024	AND	302
025	LD	202
026	OR	203
027	ANB	
028	AND-NOT	201
029	OUT	202
030	LD-NOT	202
031	LD	201
032	OR	203
033	ANB	
034	OUT	203
035	LD	300
036	TIM	501
037	K	005
038	LD	501
039	TIM	502
040	K	005
041	LD	300
042	OUT	501
043	LD	501
044	OUT	502
045	LD	000
046	OR	300

047	AND-NOT	102
048	AND-NOT	100
049	OUT	300
050	LD-NOT	203
051	AND-NOT	300
052	AND-NOT	100
053	AND-NOT	107
054	OUT	101
055	LD	300
056	AND-NOT	501
057	OUT	106
058	LD	502
059	OR	100
060	AND-NOT	203
061	LD-NOT	107
062	OR-NOT	301
063	ANB	
064	OUT	100
065	LD	203
066	LD-NOT	107
067	OR-NOT	302
068	ANB	
069	OUT	104
070	LD	203

071	AND-NOT	300
072	AND-NOT	102
073	AND-NOT	107
074	OUT	103
075	LD	502
076	OR	102
077	AND	203
078	LD-NOT	107
079	OR-NOT	302
080	ANB	
081	OUT	102
082	LD-NOT	010
083	CNT	600
084	K	012
085	LD	106
086	OUT	600
087	LD	600
088	OUT	107
089	END	

ภาคผนวก จ

โปรแกรม

```
        : PROGRAM MONITOR OF PROGRAMMABLE LOGIC CONTROL BY 8031AH
          ORG    00H
IN0:      EQU    20H
IN1:      EQU    21H
OUTPUT:     EQU    22H
IN0SET:     EQU    23H
IN1SET:     EQU    24H
OUTSET:     EQU    25H


COM_COUNTER:  EQU    40H   ;DW  POINT TO COMPLIER COUNTER USE BIT 0 ONLY
DATA_POINTER:  EQU   42H   ;DW  POINT TO DATA TO COMPLIER
RUN_POINTER:   EQU   44H   ;DW  POINT TO AREA TO RUN
ERR_POINTER:   EQU   46H   ;DW  POINT TO ERROR AREA
BUFFER1:    EQU    48H
BUFFER2:    EQU    49H
COUNTER1:    EQU   4AH
COUNTER2:    EQU   4BH
ERRBUFF:     EQU   4CH
INTBUFF:     EQU   4DH

RUN:        EQU    3000H  ;RUN AREA
ERR_AREA:     EQU   3A00H  ;SAVE ERROR MASSAGE
PORTA:      EQU    E0E0H  :INPUT PORT A
PORTB:      EQU    E0E1H  ;INPUT PORT B
PORTC:      EQU    E0E2H  ;OUTPUT PORT C      1 = IN  0 = OUT
PORT_CONTROL:  EQU   E0E3H  :CONTROL PORT  D7  MODE  A  CU MODE B  CL
DATA_CONTROL:  EQU   92H   ;COTROL DATA   1  0  0  1   0   0  1  0
AUXPOINT:     EQU    3800H
TIMPOINT:     EQU    3820H
CNTPOINT:     EQU    3840H

COMMAND       EQU    0E0C0H  ; Read-Write Register
READBUSY      EQU    0E0C1H  ; Read BF(Busy Flag) and address
```

```
READDATA      EQU    0E0C3H  ; Read Data from DD ram
CLRSCR        EQU    01H     ; CLEAR DISPLAY
HOME          EQU    02H     ; RETURN HOME

                     ; INT RAM
FIRST_START:  EQU    30H     ; DB
DIS_ADDR:     EQU    31H     ; DW
DATA_ADDR:    EQU    33H     ; DW
LINEBUFFER:   EQU    35H     ; DW   35 = MSB   36 = LSB
LINE:         EQU    37H     ; DW
KEY_DATA:     EQU    39H     ; DB   KEY BOARD DATA
POINT:        EQU    3AH     ; DW
FBUFFER:      EQU    3CH     ; DB 4 BYTE


          ORG    1F00H
;*************** DAA FOR SUBTRACTION *****

SUBDAA: PUSH   PSW          ;
     PUSH   PSW
     JNB    AC,SD_CON1
     CLR    C
     SUBB   A,#6
SD_CON1:POP    PSW          ;POP 1
     JNC    SD_CON2
     CLR    C
     SUBB   A,#60H
SD_CON2:POP    PSW          ;POP 2
     RET
```

```
        ORG     00H


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

START:  NOP
        NOP
        DJNZ    A,START
        LJMP    START0


        ORG     000BH
        LJMP    INT_SER



START0:

        MOV     DPTR,#PORT_CONTROL
        MOV     A,#DATA_CONTROL         ;SET 8255
        MOVX    @DPTR,A
        MOV     A,#FFH
        DEC     DPTR                    ;SET PORT C
        MOVX    @DPTR,A
        MOV     SP,#07H                 ;INIT  SP USE +++
        LCALL   STARTUP
        LCALL   INIT
START1: LCALL   DISPAGE
        MOV     A,#85H                  ;CURSOR AT COMMAND LINE 1
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF                  ;Wait busy flag
```

```
ST:   MOV    DPTR,#READBUSY        ;GET CURSOR POSITION IN A
      MOVX   A,@DPTR
      CLR    A.7                  ;CLEAR FIRST BIT
      MOV    R0,A                 ;GET CURSOR POSITION TO BUFFER
      ANL    A,#0FH               ;CHECK LSB NIBBLE

      CJNE   A,#0H,STCON0         ;IF END LINE
      LCALL  ST_LINE              ;THEN   CALL ST_LINE
      LJMP   ST
STCON0: LCALL  KEY                ;IF CURSOR NO END LINE
      MOV    A,KEY_DATA           ;GET KEYBOARD DATA
      JNB    A.5,STCON1           ;IF FUNCTION KEY
      LCALL  ST_FUNC              ;THEN CALL ST_FUNCTION KEY
      LCALL  DELAY
      LJMP   ST
STCON1: MOV    A,R0               ;GET CURSOR POSITION FROM BUFFER
      ANL    A,#0FH               ;CHECK LSB NIBBLE
      CJNE   A,#05H,STCON2        ;IF CURSOR AT COMMAND POSITION
      LCALL  ST_COM               ;THEN CALL ST_COMMAND
      LJMP   ST
STCON2: LCALL  ST_OPER
      LJMP   ST


ST_FUNC:MOV    A,KEY_DATA
      CJNE   A,#28H,FUNC_CON1     ;CHECK KEY  NOT
      LCALL  FUNC_NOT
      LJMP   FUNC_END
FUNC_CON1:
      CJNE   A,#25H,FUNC_CON2     ;UP
      LCALL  FUNC_UP
      LJMP   FUNC_END
FUNC_CON2:
      CJNE   A,#24H,FUNC_CON3     ;DOWN
```

```
        LCALL   FUNC_DOWN
        LJMP    FUNC_END
FUNC_CON3:
;       CJNE    A,#27H,FUNC_CON4        ;ADDRESS
;*********************************************************************************
;******************************** CHANGE TO MON KEY ******************************

        CJNE    A,#22H,FUNC_CON4        ;ADDRESS

        LCALL   FUNC_ADDR
        LJMP    FUNC_END
FUNC_CON4:
        CJNE    A,#20H,FUNC_CON5        ;INS
        LCALL   FUNC_INS
        LJMP    FUNC_END
FUNC_CON5:
        CJNE    A,#21H,FUNC_CON6        ;DEL
        LCALL   FUNC_DEL
        LJMP    FUNC_END

;*********************************************************************************
;********************************* LEFT AND RIGHT ********************************
FUNC_CON6:
        CJNE    A,#27H,FUNC_CON7        ;LEFT
        LCALL   FUNC_LEFT
        LJMP    FUNC_END
FUNC_CON7:
        CJNE    A,#26H,FUNC_CON8        ;RIGHT
        LCALL   FUNC_RIGHT
        LJMP    FUNC_END
;*********************************************************************************


FUNC_CON8:
        CJNE    A,#23H,FUNC_END         ;GO
```

```
        LJMP    COMPLIER



FUNC_END:RET


FUNC_INS:
        PUSH    DPL
        PUSH    DPH
        MOV     POINT,#FBH
        MOV     POINT+1,#3FH
FI:     CLR     C
        MOV     A,POINT
        SUBB    A,DATA_ADDR
        MOV     A,POINT+1
        SUBB    A,DATA_ADDR+1
        JC      FI_CARY
FI_NOCA:MOV     R7,#4
        MOV     DPL,POINT
        MOV     DPH,POINT+1
FI_GET: MOVX    A,@DPTR
        INC     DPTR                ;DPTR + 4
        INC     DPTR
        INC     DPTR
        INC     DPTR
        MOVX    @DPTR,A
        CLR     C                   ;DPTR - 5  NO INSTRUCTION DEC DPTR
        MOV     A,DPL
        SUBB    A,#5
        MOV     DPL,A
```

```
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
        DJNZ    R7,FI_GET
        MOV     POINT,DPL
        MOV     POINT+1,DPH
        LJMP    FI
FI_CARY:MOV     DPL,POINT
        MOV     DPH,POINT+1
        INC     DPTR
        MOV     A,#' '
        MOV     R7,#4
FI_SP:  MOVX    @DPTR,A
        INC     DPTR
        DJNZ    R7,FI_SP


FI_DISPLAY:
        MOV     FBUFFER,DATA_ADDR
        MOV     FBUFFER+1,DATA_ADDR+1
        MOV     FBUFFER+2,LINE
        MOV     FBUFFER+3,LINE+1
        MOV     A,R0
        ANL     A,#F0H                  ;GET LINE OF CURSOR
        CJNE    A,#00H,FI_CON1          ;IF CURSOR AT LINE 1
                                        ;FIRST LINE = CURSOR LINE
                                        ;ROW DATA_ADDR = DATA_ADDR
        LJMP    FI_ROW
FI_CON1:CJNE    A,#40H,FI_CON2          ;IF CURSOR AT LINE 2
        CLR     C                       ;FIRST LINE = CURSOR LINE - 1
        MOV     A,LINE
        SUBB    A,#1
        LCALL   SUBDAA
        MOV     LINE,A
        MOV     A,LINE+1
```

```
    SUBB    A,#0
    LCALL   SUBDAA
    MOV     LINE+1,A
    CLR     C                   ;ROW DATA_ADDR = DATA_ADDR - 4
    MOV     A,DATA_ADDR
    SUBB    A,#4
    MOV     DATA_ADDR,A
    MOV     A,DATA_ADDR+1
    SUBB    A,#0
    MOV     DATA_ADDR+1,A
    LJMP    FI_ROW
FI_CON2:CJNE    A,#10H,FI_CON3          ;IF CURSOR AT LINE 3
    CLR     C                   ;FIRST LINE = CURSOR LINE - 2
    MOV     A,LINE
    SUBB    A,#2
    LCALL   SUBDAA
    MOV     LINE,A
    MOV     A,LINE+1
    SUBB    A,#0
    LCALL   SUBDAA
    MOV     LINE+1,A
    CLR     C                   ;ROW DATA_ADDR = DATA_ADDR - 8
    MOV     A,DATA_ADDR
    SUBB    A,#8
    MOV     DATA_ADDR,A
    MOV     A,DATA_ADDR+1
    SUBB    A,#0
    MOV     DATA_ADDR+1,A
    LJMP    FI_ROW
FI_CON3:CJNE    A,#50H,FI_ROW           ;IF CURSOR AT LINE 4
    CLR     C                   ;FIRST LINE = CURSOR LINE - 3
    MOV     A,LINE
    SUBB    A,#3
    LCALL   SUBDAA
    MOV     LINE,A
```

```
        MOV     A,LINE+1
        SUBB    A,#0
        LCALL   SUBDAA
        MOV     LINE+1,A
        CLR     C                   ;ROW DATA_ADDR = DATA_ADDR - 12d
        MOV     A,DATA_ADDR
        SUBB    A,#0CH
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DATA_ADDR+1,A
FI_ROW: MOV     A,R0
        ANL     A,#0FH              ;GET CURSOR COLUME
        CJNE    A,#0FH,FI_CON4      ;IF COLUME = OPERAND 3
        CLR     C
        MOV     A,DATA_ADDR         ;FIRST DATA_ADDR = ROW DATA ADDR - 3
        SUBB    A,#3
        MOV     DPL,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DPH,A
        LJMP    FI_END
FI_CON4:CJNE    A,#0EH,FI_CON5      ;IF COLUME = OPERAND 2
        CLR     C
        MOV     A,DATA_ADDR         ;FIRST DATA_ADDR = ROW DATA ADDR - 2
        SUBB    A,#2
        MOV     DPL,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DPH,A
        LJMP    FI_END
FI_CON5:CJNE    A,#0DH,FI_CON6      ;IF COLUME = OPERAND 1
        CLR     C
        MOV     A,DATA_ADDR         ;FIRST DATA_ADDR = ROW DATA ADDR - 1
        SUBB    A,#1
```

```
        MOV     DPL,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DPH,A
        LJMP    FI_END
FI_CON6:MOV     DPL,DATA_ADDR           ;IF COLUME = COMMAND
        MOV     DPH,DATA_ADDR+1         ;FIRST DATA_ADDR = ROW DATA_ADDR
FI_END: LCALL   DISPAGE
        MOV     DATA_ADDR,FBUFFER       ;GET DATA FROM BUFFER
        MOV     DATA_ADDR+1,FBUFFER+1
        MOV     LINE,FBUFFER+2
        MOV     LINE+1,FBUFFER+3
        MOV     A,R0                    ;CURSOR AT OLE POSITION
        SETB    A.7
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF                  ;Wait busy flag
        POP     DPH
        POP     DPL
        RET

FUNC_DEL:
        PUSH    DPH
        PUSH    DPL
        MOV     FBUFFER,DATA_ADDR
        MOV     FBUFFER+1,DATA_ADDR+1
        MOV     FBUFFER+2,LINE
        MOV     FBUFFER+3,LINE+1
        MOV     POINT,DATA_ADDR
        MOV     POINT+1,DATA_ADDR+1
        MOV     A,R0                    ;GET CURSOR POSITION
        ANL     A,#0FH                  ;GET COLUME
        CJNE    A,#05H,FDEL_CON1        ;IF COMMAND
        MOV     A,DATA_ADDR             ;
        ADD     A,#4
```

```
    MOV    POINT,A
    MOV    A,DATA_ADDR+1
    ADDC   A,#0
    MOV    POINT+1,A
    LJMP   FDEL_CUR
FDEL_CON1:
    CJNE   A,#0DH,FDEL_CON2      ;IF OPER 1   =   + 3
    MOV    A,DATA_ADDR
    ADD    A,#3
    MOV    POINT,A
    MOV    A,DATA_ADDR+1
    ADDC   A,#0
    MOV    POINT+1,A
    LJMP   FDEL_CUR
FDEL_CON2:
    CJNE   A,#0EH,FDEL_CON3      ;IF OPER 2   =   + 2
    MOV    A,DATA_ADDR
    ADD    A,#2
    MOV    POINT,A
    MOV    A,DATA_ADDR+1
    ADDC   A,#0
    MOV    POINT+1,A
    LJMP   FDEL_CUR
FDEL_CON3:
    CJNE   A,#0FH,FDEL_CUR       ;IF OPER 1   =   + 1
    MOV    A,DATA_ADDR
    ADD    A,#1
    MOV    POINT,A
    MOV    A,DATA_ADDR+1
    ADDC   A,#0
    MOV    POINT+1,A
FDEL_CUR:
    MOV    DPL,POINT
    MOV    DPH,POINT+1
FDEL_LOOP:
```

```
        MOV     R7,#4
FDEL_GET:
        MOVX    A,@DPTR
        PUSH    A                   ;DPTR - 4
        CLR     C
        MOV     A,DPL
        SUBB    A,#4
        MOV     DPL,A
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
        POP     A
        MOVX    @DPTR,A
        MOV     A,DPL               ;DPTR + 5
        ADD     A,#5
        MOV     DPL,A
        MOV     A,DPH
        ADDC    A,#0
        MOV     DPH,A
        DJNZ    R7,FDEL_GET
        CJNE    A,#30H,FDEL_LOOP    ;TO 3000H
;*******************************************************************************
        MOV     DPTR,#2FFFH
        MOV     A,#' '              ;LAST 4 BYTE = BLANK
        MOVX    @DPTR,A
        DEC     DPL
        MOV     A,#' '
        MOVX    @DPTR,A
        DEC     DPL
        MOV     A,#' '
        MOVX    @DPTR,A
        DEC     DPL
        MOV     A,#' '
        MOVX    @DPTR,A
;*******************************************************************************
```

```
        MOV     A,R0
        ANL     A,#F0H                  ;GET LINE OF CURSOR
        CJNE    A,#00H,FDEL_CON4        ;IF CURSOR AT LINE 1
                                        ;FIRST LINE = CURSOR LINE
                                        ;ROW DATA_ADDR = DATA_ADDR
        LJMP    FDEL_ROW
FDEL_CON4:
        CJNE    A,#40H,FDEL_CON5        ;IF CURSOR AT LINE 2
        CLR     C                       ;FIRST LINE = CURSOR LINE - 1
        MOV     A,LINE
        SUBB    A,#1
        LCALL   SUBDAA
        MOV     LINE,A
        MOV     A,LINE+1
        SUBB    A,#0
        LCALL   SUBDAA
        MOV     LINE+1,A
        CLR     A
        MOV     A,DATA_ADDR             ;ROW DATA_ADDR = DATA_ADDR - 4
        SUBB    A,#4
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DATA_ADDR+1,A
        LJMP    FDEL_ROW
FDEL_CON5:
        CJNE    A,#10H,FDEL_CON6        ;IF CURSOR AT LINE 3
        CLR     C                       ;FDELRST LINE = CURSOR LINE - 2
        MOV     A,LINE
        SUBB    A,#2
        LCALL   SUBDAA
        MOV     LINE,A
        MOV     A,LINE+1
        SUBB    A,#0
```

```
        LCALL   SUBDAA
        MOV     LINE+1,A
        CLR     C
        MOV     A,DATA_ADDR             ;ROW DATA_ADDR = DATA_ADDR - 8
        SUBB    A,#8
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DATA_ADDR+1,A
        LJMP    FDEL_ROW
FDEL_CON6:
        CJNE    A,#50H,FDEL_ROW         ;IF CURSOR AT LINE 4
        CLR     C                       ;FDELRST LINE = CURSOR LINE - 3
        MOV     A,LINE
        SUBB    A,#3
        LCALL   SUBDAA
        MOV     LINE,A
        MOV     A,LINE+1
        SUBB    A,#0
        LCALL   SUBDAA
        MOV     LINE+1,A
        CLR     C
        MOV     A,DATA_ADDR             ;ROW DATA_ADDR = DATA_ADDR - 12d
        SUBB    A,#0CH
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DATA_ADDR+1,A
FDEL_ROW:
        MOV     A,R0
        ANL     A,#0FH                  ;GET CURSOR COLUME
        CJNE    A,#0FH,FDEL_CON7        ;IF COLUME = OPERAND 3
        CLR     C
        MOV     A,DATA_ADDR             ;FIRST DATA_ADDR = ROW DATA ADDR - 3
        SUBB    A,#3
```

```
    MOV    DPL,A
    MOV    A,DATA_ADDR+1
    SUBB   A,#0
    MOV    DPH,A
    LJMP   FDEL_END
FDEL_CON7:
    CJNE   A,#0EH,FDEL_CON8      ;IF COLUME = OPERAND 2
    CLR    C
    MOV    A,DATA_ADDR           ;FIRST DATA_ADDR = ROW DATA ADDR - 2
    SUBB   A,#2
    MOV    DPL,A
    MOV    A,DATA_ADDR+1
    SUBB   A,#0
    MOV    DPH,A
    LJMP   FDEL_END
FDEL_CON8:
    CJNE   A,#0DH,FDEL_CON9      ;IF COLUME = OPERAND 1
    CLR    C
    MOV    A,DATA_ADDR           ;FIRST DATA_ADDR = ROW DATA ADDR - 1
    SUBB   A,#1
    MOV    DPL,A
    MOV    A,DATA_ADDR+1
    SUBB   A,#0
    MOV    DPH,A
    LJMP   FDEL_END
FDEL_CON9:
    MOV    DPL,DATA_ADDR         ;IF COLUME = COMMAND
    MOV    DPH,DATA_ADDR+1       ;FIRST DATA_ADDR = ROW DATA_ADDR
FDEL_END:
    LCALL  DISPAGE
    MOV    DATA_ADDR,FBUFFER     ;GET DATA FROM BUFFER
    MOV    DATA_ADDR+1,FBUFFER+1
    MOV    LINE,FBUFFER+2
    MOV    LINE+1,FBUFFER+3
    MOV    A,R0                  ;CURSOR AT OLE POSITION
```

```
        SETB    A.7
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF                  ;Wait busy flag
        POP     DPH
        POP     DPL
        RET


FUNC_NOT:
        MOV     DPL,DATA_ADDR
        MOV     DPH,DATA_ADDR+1
        CLR     C                       ;DPTR - 1  NO INSTRUCTION DEC DPTR
        MOV     A,DPL
        SUBB    A,#1
        MOV     DPL,A
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
        MOVX    A,@DPTR                 ;GET OLD DATA_ADDR
        CJNE    A,#43H,FN_CON1          ;CHECK COMMAND LD
        MOV     A,#44H
        MOVX    @DPTR,A
        LJMP    FN_CON6
FN_CON1:CJNE    A,#44H,FN_CON2          ;CHECK COMMAND LD NOT
        MOV     A,#43H
        MOVX    @DPTR,A
        LJMP    FN_CON6
FN_CON2:CJNE    A,#45H,FN_CON3          ;CHECK COMMAND AND
        MOV     A,#46H
        MOVX    @DPTR,A
        LJMP    FN_CON6
FN_CON3:CJNE    A,#46H,FN_CON4          ;CHECK COMMAND AND NOT
```

```
        MOV     A,#45H
        MOVX    @DPTR,A
        LJMP    FN_CON6
FN_CON4:CJNE    A,#47H,FN_CON5          ;CHECK COMMAND OR
        MOV     A,#48H
        MOVX    @DPTR,A
        LJMP    FN_CON6
FN_CON5:CJNE    A,#48H,FN_C5.1          ;CHECK COMMAND OR NOT
        MOV     A,#47H
        MOVX    @DPTR,A
        LJMP    FN_CON6
FN_C5.1:CJNE    A,#4CH,FN_C5.2          ;CHECK COMMAND TIME
        MOV     A,#4DH
        MOVX    @DPTR,A
        LJMP    FN_CON6
FN_C5.2:CJNE    A,#4DH,FN_C5.3          ;CHECK COMMAND TIME NOT
        MOV     A,#4CH
        MOVX    @DPTR,A
        LJMP    FN_CON6
FN_C5.3:CJNE    A,#4EH,FN_C5.4          ;CHECK COMMAND COUNTER
        MOV     A,#4FH
        MOVX    @DPTR,A
        LJMP    FN_CON6
FN_C5.4:CJNE    A,#4FH,FN_NO            ;CHECK COMMAND COUNTER NOT
        MOV     A,#4EH
        MOVX    @DPTR,A
        LJMP    FN_CON6
FN_CON6:MOV     A,R0                    ;GET CURSOR POSITION
        ANL     A,#0F0H                 ;CLEAR LSB NIBBLE
        ORL     A,#085H                 ;SET LSB NIBBLE = COMMAND AND BIT 7
        PUSH    DPL
        PUSH    DPH
        MOV     DPTR,#COMMAND           ;MOVE CURSOR
        MOVX    @DPTR,A
        LCALL   WAITBF                  ;Wait busy flag
```

```
        POP     DPH
        POP     DPL
        LCALL   DISCOM
FN_NO:  INC     DPTR                ;GET OLE DATA_ADDR
        MOV     DATA_ADDR,DPL
        MOV     DATA_ADDR+1,DPH
        RET


FUNC_UP:PUSH    DPL
        PUSH    DPH
        MOV     A,R0                ;GET CURSOR POSITION
        ANL     A,#F0H              ;CLEAR LSB NIBBLE
        CJNE    A,#00H,FU_NO_FIRST_LINE
        MOV     A,LINE+1            ;IF LINE 001
        CJNE    A,#0,FU_CON0
        MOV     A,LINE
        CJNE    A,#0,FU_CON0
        LJMP    FU_END
FU_CON0:CLR     C                   ;IF FIRST LINE
        MOV     A,LINE              ;LINE - 1
        SUBB    A,#1
        LCALL   SUBDAA
        MOV     LINE,A
        MOV     A,LINE+1
        SUBB    A,#0
        LCALL   SUBDAA
        MOV     LINE+1,A
        CLR     C
        MOV     A,DATA_ADDR         ;DATA_ADDR - 4
        SUBB    A,#4
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DATA_ADDR+1,A
```

```
;*********************************************************************************************
        MOV     DPL,DATA_ADDR
        MOV     DPH,DATA_ADDR+1

        MOV     A,R0
        ANL     A,#0FH                  ;GET CURSOR COLUME

        CJNE    A,#0FH,FU_EXT1          ;IF COLUME = OPERAND 3
        CLR     C
        MOV     A,DPL                   ;DPTR - 3
        SUBB    A,#3
        MOV     DPL,A
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
        LJMP    FU_EXTE
FU_EXT1:CJNE    A,#0EH,FU_EXT2          ;IF COLUME = OPERAND 2
        CLR     C
        MOV     A,DPL                   ;DPTR - 2
        SUBB    A,#2
        MOV     DPL,A
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
        LJMP    FU_EXTE
FU_EXT2:CJNE    A,#0DH,FU_EXTE          ;IF COLUME = OPERAND 1
        CLR     C
        MOV     A,DPL                   ;DPTR - 1
        SUBB    A,#1
        MOV     DPL,A
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
FU_EXTE:                                ;IF COLUME = COMMAND
```

```
;*****************************************************************************************
****

        LCALL   DISPAGE
        MOV     A,R0
        SETB    A.7
        MOV     DPTR,#COMMAND           ;MOV CURSOR
        MOVX    @DPTR,A
        LCALL   WAITBF                  ;Wait busy flag
        LJMP    FU_END
FU_NO_FIRST_LINE:
        MOV     A,R0                    ;GET CURSOR POSITION
        ANL     A,#F0H                  ;CLEAR LSB NIBBLE
        CJNE    A,#50H,FU_CON1          ;IF LINE 4
        MOV     R6,#10H                 ;THEN LINE 3
        LJMP    FU_CON3
FU_CON1:CJNE    A,#10H,FU_CON2          ;IF LINE 3
        MOV     R6,#40H                 ;THEN LINE 2
        LJMP    FU_CON3
FU_CON2:CJNE    A,#40H,FU_CON3          ;IF LINE 2
        MOV     R6,#00H                 ;THEN LINE 1
FU_CON3:CLR     C
        MOV     A,LINE                  ;LINE - 1
        SUBB    A,#1
        LCALL   SUBDAA
        MOV     LINE,A
        MOV     A,LINE+1
        SUBB    A,#0
        LCALL   SUBDAA
        MOV     LINE+1,A
        CLR     C
        MOV     A,DATA_ADDR             ;DATA ADDR - 4
        SUBB    A,#4
```

```
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DATA_ADDR+1,A
        MOV     A,R0                ;GET OLD CURSOR
        ANL     A,#8FH              ;CLEAR D6,D5,D4
        ORL     A,R6                ;MOV NEW LINE
        SETB    A.7
        MOV     DPTR,#COMMAND       ;MOV CURSOR
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
FU_END: POP     DPH
        POP     DPL
        RET

FUNC_DOWN:
        PUSH    DPL
        PUSH    DPH
        MOV     A,R0                ;GET CURSOR POSITION
        ANL     A,#F0H              ;CLEAR LSB NIBBLE
        CJNE    A,#50H,FD_NO_LL1
        LJMP    FD_CON0.5           ;************************************************
FD_NO_LL1:                          ;** RELATIVE TO LONG
        LJMP    FD_NO_LL2
FD_CON0.5:
        MOV     A,LINE+1            ;IF LINE 999
        CJNE    A,#09H,FD_CON0
        MOV     A,LINE
        CJNE    A,#99H,FD_CON0
        LJMP    FD_END
FD_CON0:MOV     A,LINE              ;IF LAST LINE
        ADD     A,#1                ;LINE + 1
        DA      A
        MOV     LINE,A
```

```
        MOV     A,LINE+1
        ADDC    A,#0
        DA      A
        MOV     LINE+1,A
        MOV     A,DATA_ADDR             ;DATA_ADDR + 4
        ADD     A,#4
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        ADDC    A,#0
        MOV     DATA_ADDR+1,A

        MOV     FBUFFER,DATA_ADDR
        MOV     FBUFFER+1,DATA_ADDR+1
        MOV     FBUFFER+2,LINE
        MOV     FBUFFER+3,LINE+1

        MOV     DPL,DATA_ADDR
        MOV     DPH,DATA_ADDR+1

        CLR     C                       ;DATA_ADDR - 12d
        MOV     A,DATA_ADDR
        SUBB    A,#12
        MOV     DPL,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DPH,A

        CLR     C
        MOV     A,LINE                  ;LINE - 3
        SUBB    A,#3
        LCALL   SUBDAA
        MOV     LINE,A
        MOV     A,LINE+1
        SUBB    A,#0
        LCALL   SUBDAA
```

```
        MOV     LINE+1,A

;*****************************************************************************

        MOV     A,R0
        ANL     A,#0FH              ;GET CURSOR COLUME

        CJNE    A,#0FH,FD_EXT1      ;IF COLUME = OPERAND 3
        CLR     C
        MOV     A,DPL               ;DPTR - 3
        SUBB    A,#3
        MOV     DPL,A
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
        LJMP    FD_EXTE
FD_EXT1:CJNE    A,#0EH,FD_EXT2      ;IF COLUME = OPERAND 2
        CLR     C
        MOV     A,DPL               ;DPTR - 2
        SUBB    A,#2
        MOV     DPL,A
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
        LJMP    FD_EXTE
FD_EXT2:CJNE    A,#0DH,FD_EXTE      ;IF COLUME = OPERAND 1
        CLR     C
        MOV     A,DPL               ;DPTR - 1
        SUBB    A,#1
        MOV     DPL,A
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
FD_EXTE:                            ;IF COLUME = COMMAND
```

```
;*****************************************************************************************
****

    LCALL   DISPAGE
    MOV     DATA_ADDR,FBUFFER
    MOV     DATA_ADDR+1,FBUFFER+1
    MOV     LINE,FBUFFER+2
    MOV     LINE+1,FBUFFER+3


    MOV     A,R0
    SETB    A.7
    MOV     DPTR,#COMMAND           ;MOV CURSOR
    MOVX    @DPTR,A
    LCALL   WAITBF                  ;Wait busy flag
    LJMP    FD_END
FD_NO_LL2:
    MOV     A,R0                    ;GET CURSOR POSITION
    ANL     A,#F0H                  ;CLEAR LSB NIBBLE
    CJNE    A,#00H,FD_CON1          ;IF LINE 1
    MOV     R6,#40H                 ;THEN LINE 2
    LJMP    FD_CON3
FD_CON1:CJNE    A,#40H,FD_CON2      ;IF LINE 2
    MOV     R6,#10H                 ;THEN LINE 3
    LJMP    FD_CON3
FD_CON2:CJNE    A,#10H,FD_CON3      ;IF LINE 3
    MOV     R6,#50H                 ;THEN LINE 4
FD_CON3:MOV     A,R0                ;GET OLD CURSOR
    ANL     A,#8FH                  ;CLEAR D6,D5,D4
    ORL     A,R6                    ;MOV NEW LINE
FD_CON4:SETB    A.7
    MOV     DPTR,#COMMAND           ;MOV CURSOR
    MOVX    @DPTR,A
    LCALL   WAITBF                  ;Wait busy flag
```

```
        MOV     A,DATA_ADDR             ;DATA ADDR + 4
        ADD     A,#4
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        ADDC    A,#0
        MOV     DATA_ADDR+1,A
        MOV     A,LINE
        ADD     A,#1                    ;LINE + 1
        DA      A
        MOV     LINE,A
        MOV     A,LINE+1
        ADDC    A,#0
        DA      A
        MOV     LINE+1,A

FD_END: POP     DPH
        POP     DPL
        RET


FUNC_LEFT:
        MOV     A,R0
        ANL     A,#0FH                  ;GET CURSOR COLUME

        CJNE    A,#0FH,FL_CON1          ;IF COLUME = OPERAND 3
        CLR     C
        MOV     A,DATA_ADDR             ;DEC DATA ADDRESS
        SUBB    A,#1
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DATA_ADDR+1,A
```

```
        MOV     A,R0                ;OLD CURSOR
        ANL     A,#F0H              ;CLEAR COLUME
        ORL     A,#8EH              ;MOV CURSOR TO OPER 2
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        LJMP    FL_END
FL_CON1:CJNE    A,#0EH,FL_CON2      ;IF COLUME = OPERAND 2
        CLR     C
        MOV     A,DATA_ADDR         ;DEC DATA ADDRESS
        SUBB    A,#1
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DATA_ADDR+1,A

        MOV     A,R0                ;OLD CURSOR
        ANL     A,#F0H              ;CLEAR COLUME
        ORL     A,#8DH              ;MOV CURSOR TO OPER 1
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        LJMP    FL_END
FL_CON2:CJNE    A,#0DH,FL_END       ;IF COLUME = OPERAND 1
        CLR     C
        MOV     A,DATA_ADDR         ;DEC DATA ADDRESS
        SUBB    A,#1
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DATA_ADDR+1,A

        MOV     A,R0                ;OLD CURSOR
        ANL     A,#F0H              ;CLEAR COLUME
```

```
        ORL     A,#85H              ;MOV CURSOR TO COMMAND
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        LJMP    FL_END

FL_END: LCALL   DELAY               ;IF COLUME = COMMAND
        RET


FUNC_RIGHT:
        MOV     A,R0
        ANL     A,#0FH              ;GET CURSOR COLUME

        CJNE    A,#05H,FR_CON1      ;IF COLUME = COMMAND
        CLR     C
        MOV     A,DATA_ADDR         ;INC DATA ADDRESS
        ADD     A,#1
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        ADDC    A,#0
        MOV     DATA_ADDR+1,A

        MOV     A,R0                ;OLD CURSOR
        ANL     A,#F0H              ;CLEAR COLUME
        ORL     A,#8DH              ;MOV CURSOR TO OPER 1
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        LJMP    FR_END
FR_CON1:CJNE    A,#0DH,FR_CON2      ;IF COLUME = OPERAND 1
        CLR     C
        MOV     A,DATA_ADDR         ;INC DATA ADDRESS
```

```
        ADD     A,#1
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        ADDC    A,#0
        MOV     DATA_ADDR+1,A

        MOV     A,R0                ;OLD CURSOR
        ANL     A,#F0H              ;CLEAR COLUME
        ORL     A,#8EH              ;MOV CURSOR TO OPER 2
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        LJMP    FR_END
FR_CON2:CJNE    A,#0EH,FR_END       ;IF COLUME = OPERAND 2
        CLR     C
        MOV     A,DATA_ADDR         ;DEC DATA ADDRESS
        ADD     A,#1
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        ADDC    A,#0
        MOV     DATA_ADDR+1,A

        MOV     A,R0                ;OLD CURSOR
        ANL     A,#F0H              ;CLEAR COLUME
        ORL     A,#8FH              ;MOV CURSOR TO OPER 3
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        LJMP    FR_END

FR_END: LCALL   DELAY               ;IF COLUME = OPER 3
        RET
```

```
FUNC_ADDR:
      PUSH    DPL
      PUSH    DPH
FA_NEW: MOV     A,#CLRSCR             ;CLEAR DISPLAY
      MOV     DPTR,#COMMAND
      MOVX    @DPTR,A
      LCALL   WAITBF                ;Wait busy flag
      MOV     R6,#22
      MOV     DPTR,#ADDR_FUNC
FA_LOOP:MOV     A,#0                  ;DISPLAY TITLE
      MOVC    A,@A+DPTR
      LCALL   WRITE
      INC     DPTR
      DJNZ    R6,FA_LOOP
;*************************************DELAY KEY
**********************************
      LCALL   DELAY
      LCALL   DELAY

FA_L2:  LCALL   KEY
;       CJNE    A,#27H,FA_CON0        ;CHECK ADDR AGAIN
;       LJMP    FA_NEW
;*****************************************************************************************
;FA_CON0:CJNE    A,#26H,FA_CON1        ;CHECK DATA KEY TO GO


FA_CON0:CJNE    A,#22H,FA_CON1        ;CHANGE TO MON KEY

      LCALL   DELAY
      LCALL   DELAY
      LJMP    FA_GO
FA_CON1:ANL     A,#30H                ;CHECK BIT 3,4 IF SET = NOT NUMBER
      JNZ     FA_L2                 ;IF NOT NUMBER CALL KEY AGAIN
```

```
        MOV     DPTR,#READBUSY          ;GET CURSOR POSITION IN A
        MOVX    A,@DPTR
        CLR     A.7                     ;CLEAR FIRST BIT
        CJNE    A,#19H,FA_CON2          ;IF NO OVER
        MOV     A,#96H                  ;IF OVER 3 NUMBER
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF                  ;Wait busy flag
        MOV     R6,#3                   ;CLEAR NUMBER DISPLAY
FA_CL:  MOV     A,#' '
        LCALL   WRITE
        DJNZ    R6,FA_CL
        MOV     A,#96H                  ;SET CURSOR TO FIRST NUMBER
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF                  ;Wait busy flag
        LCALL   DELAY
        LJMP    FA_L2
FA_CON2:MOV     A,KEY_DATA
        ADD     A,#'0'
        LCALL   WRITE
        LCALL   DELAY
        LJMP    FA_L2
FA_GO:  MOV     R6,#96H                 ;CURSOR POSITION BUFFER
        MOV     R4,#3
        MOV     FBUFFER,#0              ;SET LINE BUFFER = 00
        MOV     FBUFFER+1,#0
FA_GO0: MOV     A,R6                    ;SET CURSOR POSITION
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF                  ;Wait busy flag
        MOV     DPTR,#READDATA
FA_GO1: MOVX    A,@DPTR                 ;READ DATA
        JB      ACC.7,FA_GO1            ;Busy Flag
        CJNE    A,#' ',FA_GO1.1         ;IF LSB NO HAVE MIDDLE = LSB
```

```
        LJMP    FA_GO3
FA_GO1.1:
        SWAP    A
        MOV     R5,A                ;BUFFER
        MOV     R7,#4               ;SET LOOP
FA_GO2: MOV     A,R5
        RLC     A
        MOV     R5,A
        MOV     A,FBUFFER
        RLC     A
        MOV     FBUFFER,A
        MOV     A,FBUFFER+1
        RLC     A
        MOV     FBUFFER+1,A
        DJNZ    R7,FA_GO2
FA_GO3: INC     R6                  ;DEC CURSOR POSITION
        DJNZ    R4,FA_GO0
        MOV     LINE,#0
        MOV     LINE+1,#0
        MOV     DATA_ADDR,#60H
        MOV     DATA_ADDR+1,#20H
FA_GO4: MOV     A,LINE+1
        CJNE    A,FBUFFER+1,FA_GO5  ;CHECK MSB IF LINE <> FBUFFER
        MOV     A,LINE
        CJNE    A,FBUFFER,FA_GO5    ;CHECK LSB
        LJMP    FA_GO6
FA_GO5: MOV     A,DATA_ADDR         ;ADD DATA ADDR + 4
        ADD     A,#4
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        ADDC    A,#0
        MOV     DATA_ADDR+1,A
        MOV     A,LINE              ;INC LINE
        ADD     A,#1
        DA      A
```

```
        MOV     LINE,A
        MOV     A,LINE+1
        ADDC    A,#0
        DA      A
        MOV     LINE+1,A
        LJMP    FA_GO4
FA_GO6: MOV     A,LINE+1            ;CHECK END LINE
        CJNE    A,#09H,FA_GO9
        MOV     A,LINE
        CJNE    A,#99H,FA_GO7       ;IF LINE 99
        MOV     LINE,#96H
        CLR     C                   ;DATA ADDR = 99
        MOV     A,DATA_ADDR         ;DATA TO DISPLAY = 99 - 12
        SUBB    A,#0CH
        MOV     DPL,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DPH,A
        LCALL   DISPAGE
        MOV     A,#D5H              ;MOV CURSOR TO LINE 4
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        MOV     LINE,#99H
        LJMP    FA_END
FA_GO7: CJNE    A,#98H,FA_GO8
        MOV     LINE,#96H
        CLR     C                   ;DATA ADDR - 8
        MOV     A,DATA_ADDR         ;DATA TO DISPLAY = 99 - 8
        SUBB    A,#08H
        MOV     DPL,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DPH,A
        LCALL   DISPAGE
```

```
        MOV     A,#95H              ;MOV CURSOR TO LINE 3
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        MOV     LINE,#98H
        LJMP    FA_END
FA_GO8: CJNE    A,#97H,FA_GO9
        MOV     LINE,#96H
        CLR     C                   ;DATA ADDR - 4
        MOV     A,DATA_ADDR         ;DATA TO DISPLAY = 99 - 4
        SUBB    A,#04H
        MOV     DPL,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0
        MOV     DPH,A
        LCALL   DISPAGE
        MOV     A,#C5H              ;MOV CURSOR TO LINE 2
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        MOV     LINE,#97H
        LJMP    FA_END
FA_GO9: MOV     DPL,DATA_ADDR
        MOV     DPH,DATA_ADDR+1
        LCALL   DISPAGE
        MOV     A,#85H              ;MOV CURSOR TO LINE 1
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
FA_END: MOV     SP,#7
        LJMP    ST
```

```
ST_LINE:PUSH    DPL
        PUSH    DPH
        MOV     A,LINE+1            ;IF END OF LINE
        CJNE    A,#09H,L_CON
        MOV     A,LINE
        CJNE    A,#99H,L_CON
        MOV     DATA_ADDR,#F8H      ;DATA_ADDR = LAST LINE
        MOV     DATA_ADDR+1,#3FH
        MOV     A,#D5H              ;CURSOR AT COMMAND LINE 4
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        LJMP    L_RET
L_CON:  MOV     A,R0                ;GET CURSOR POSITION
        CJNE    A,#60H,L_CON0       ;IF ENDE PAGE
        LCALL   END_PAGE            ;THEN CALL END PAGE
        LJMP    L_ED
L_CON0: CJNE    A,#10H,L_CON1       ;CHECK END OF FIRST LINE
        MOV     A,#0C0H             ;line 2   set DD ram bit 7 mus 1 40=C0
        LJMP    L_END
L_CON1: CJNE    A,#50H,L_CON2       ;CHECK END OF LINE 2
        MOV     A,#90H              ;THEN LINE 3
        LJMP    L_END
L_CON2: MOV     A,#0D0H             ;IF END OF LINE 3 THEN LINE 4
L_END:  MOV     DPTR,#COMMAND       ;MOVE CURSOR
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
L_ED:   MOV     A,LINE
        ADD     A,#1                ;INC LINE
        DA      A
        MOV     LINE,A
        MOV     A,#0
        ADDC    A,LINE+1
        DA      A
        MOV     LINE+1,A
```

```
    LCALL   DISLINE
L_RET:  POP     DPH
        POP     DPL
        RET

END_PAGE:PUSH   DPL
        PUSH    DPH
        MOV     FBUFFER,DATA_ADDR       ;GET DATA ADDR  AND  LINE TO BUFFER
        MOV     FBUFFER+1,DATA_ADDR+1
        MOV     FBUFFER+2,LINE
        MOV     FBUFFER+3,LINE+1

                        ;DATA ADDRESS - 0CH
        CLR     C               ;CLEAR CARRY FLAG
        MOV     A,DATA_ADDR
        SUBB    A,#0CH
        MOV     DATA_ADDR,A
        MOV     A,DATA_ADDR+1
        SUBB    A,#0                ;SUB CARRY FLAG
        MOV     DATA_ADDR+1,A
                        ;LINE - 2
        CLR     C               ;CLEAR CARRY FLAG
        CLR     AC
        MOV     A,LINE
        SUBB    A,#2
        LCALL   SUBDAA
        MOV     LINE,A
        MOV     A,LINE+1
        SUBB    A,#0                ;SUB CARRY FLAG
        LCALL   SUBDAA
        MOV     LINE+1,A

        MOV     DPL,DATA_ADDR
        MOV     DPH,DATA_ADDR+1
        LCALL   DISPAGE
```

```
    MOV     A,#0D0H             ;MOV CURSOR TO COMMAND LINE 4
    MOV     DPTR,#COMMAND       ;MOVE CURSOR
    MOVX    @DPTR,A
    LCALL   WAITBF              ;Wait busy flag
    MOV     DATA_ADDR,FBUFFER   ;GET DATA ADDR  AND  LINE FROM BUFFER
    MOV     DATA_ADDR+1,FBUFFER+1
    MOV     LINE,FBUFFER+2
    MOV     LINE+1,FBUFFER+3
    POP     DPH
    POP     DPL
    RET

ST_COM: PUSH    DPL             ;COMMAND KEY
    PUSH    DPH
    MOV     A,KEY_DATA
    JNB     A.6,NO_COM          ;IF NO COMMAND KEY
    MOV     DPL,DATA_ADDR       ;GET DATA ADDRESS FROM BUFFER
    MOV     DPH,DATA_ADDR+1
    MOVX    @DPTR,A             ;MOV DATA TO DATA ADDRESS
    LCALL   DISCOM              ;DISPLAY
    LCALL   DELAY               ;DELAY KEYBOARD

    MOV     A,KEY_DATA
    CJNE    A,#40H,C_CON0       ;CHECK COMMAND NO OPERAND
    LCALL   COM_NO_OPER
    LJMP    C_CON2
C_CON0: CJNE    A,#41H,C_CON1
    LCALL   COM_NO_OPER
    LJMP    C_CON2
```

```
C_CON1: CJNE    A,#42H,C_CON2
        LCALL   COM_NO_OPER
C_CON2: INC     DPTR                ;DATA_ADDR + 1
        MOV     DATA_ADDR,DPL       ;GET DATA ADDR TO BUFFER
        MOV     DATA_ADDR+1,DPH
NO_COM: POP     DPH
        POP     DPL
        RET

COM_NO_OPER:
        INC     DPTR                ;DATA_ADDR + 3 = GET SPACE TO OPER
        MOV     A,#' '
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#' '
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#' '
        MOVX    @DPTR,A
        LCALL   DISOPER
        LCALL   DISOPER
        LCALL   DISOPER
        RET


ST_OPER:PUSH    DPL
        PUSH    DPH
        MOV     A,KEY_DATA
        JB      A.6,NO_NUM          ;IF NO NUMBER KEY
        MOV     DPL,DATA_ADDR       ;GET DATA ADDRESS FROM BUFFER
        MOV     DPH,DATA_ADDR+1
        MOVX    @DPTR,A             ;MOV DATA TO DATA ADDRESS
        LCALL   DISOPER             ;DISPLAY
```

```
        LCALL   DELAY               ;DELAY KEYBOARD
        INC     DPTR
        MOV     DATA_ADDR,DPL       ;GET DATA ADDR TO BUFFER
        MOV     DATA_ADDR+1,DPH
NO_NUM: POP     DPH
        POP     DPL
        RET


;************** DELAY FOR KEY **************
;******** USE R6,R7*************************
DELAY:  PUSH    PSW
        PUSH    A
        MOV     A,R6
        PUSH    A
        MOV     A,R7
        PUSH    A
        MOV     R7,#090H
DELAY1: MOV     R6,#0FFH
DELAY2: NOP
        NOP
        DJNZ    R6,DELAY2
        DJNZ    R7,DELAY1

        POP     A
        MOV     R7,A
        POP     A
        MOV     R6,A
        POP     A
        POP     PSW
        RET

;*************** START UP ************************************
```

```
STARTUP:
      MOV     A,#F7H                    ;KEY LD TO CLEAR MEM
      MOV     P1,A
      JB      P1.7,ST_CON0.5
      JNB     P1.7,$
      LJMP    FIRST
ST_CON0.5:
      JB      P1.6,ST_CON0              ;KEY K TO TITLE
      JNB     P1.6,$
      LCALL   TITLE
      LJMP    FIRST
ST_CON0:MOV     A,FIRST_START
      CJNE    A,#55H,FIRST              ;IF FIRST
      LJMP    NOFIRST
FIRST:  MOV     FIRST_START,#55H          ;SET FIRST START
      MOV     DPTR,#2060H               ;SET SPACE AT DATA
      MOV     A,#' '                    ;SPACE BAR
      MOV     R0,#10H
STLOOP: MOV     R1,#0FFH
STLOOP1:MOVX    @DPTR,A
      INC     DPTR
      DJNZ    R1,STLOOP1
      DJNZ    R0,STLOOP
NOFIRST:MOV     LINE,#0                   ;SET LINE = 1
      MOV     LINE+1,#0
      MOV     DIS_ADDR,#60H             ;SET DISPLAY ADDR = 2060
      MOV     DIS_ADDR+1,#20H
      MOV     DATA_ADDR,#60H            ;SET DATA ADDR = 2060
      MOV     DATA_ADDR+1,#20H
      MOV     DPTR,#2060H
      RET
;*******************************************************************************
;************************  TITLE DISPLAY  ***************************************
;*******************************************************************************
```

```
TITLE: LCALL   INIT
       MOV     A,#CLRSCR          ;CLEAR DISPLAY
       MOV     DPTR,#COMMAND
       MOVX    @DPTR,A
       LCALL   WAITBF             ;Wait busy flag

       MOV     DPTR,#TITLE_DATA
       MOV     R5,#32
TITLE1: MOV    A,#0               ;DISPLAY LINE 1 AND 3 CONTINUE
       MOVC    A,@A+DPTR
       LCALL   WRITE
       INC     DPTR
       DJNZ    R5,TITLE1

       MOV     A,#C0H             ;MOV CURSOR TO LINE 2
       MOV     DPTR,#COMMAND
       MOVX    @DPTR,A
       LCALL   WAITBF             ;Wait busy flag

       MOV     DPTR,#TITLE_DATA+32
       MOV     R5,#32             ;PRINT LINE 2 AND 4 CONTINUE
TITLE2: MOV    A,#0               ;DISPLAY
       MOVC    A,@A+DPTR
       LCALL   WRITE
       INC     DPTR
       DJNZ    R5,TITLE2


       LCALL   DELAY
       LCALL   KEY
       LCALL   DELAY
       RET
```

```
;********* DISPLAY 1 PAGE *********************************
;**** INPUT LINE NUMBER IN LINE and FIRST ADDR IN @DPTR **
;****USE A,R3,R4*****************************************

DISPAGE:PUSH    DPL
        PUSH    DPH
        MOV     A,LINE              ;GET LINE NUMBER TO BUFFER
        MOV     LINEBUFFER,A
        MOV     A,LINE+1
        MOV     LINEBUFFER+1,A
        PUSH    DPL
        PUSH    DPH
        MOV     A,#80H              ;MOVE CURSOR TO FIRST
        MOV     DPTR,#COMMAND           ;IF END OF LINE 4 THEN LINE 4
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        POP     DPH
        POP     DPL
        MOV     R3,#4               ;DISPLAY 4 LINE
PAGELOOP:LCALL  DISLINE                 ;DISPLAY LINE NUMBER
        LCALL   DISCOM
        INC     DPTR
        MOV     R4,#03H
OPER3:  LCALL   DISOPER
        INC     DPTR
        DJNZ    R4,OPER3
        LCALL   CURLINE
        DJNZ    R3,PAGELOOP
        MOV     A,LINEBUFFER            ;GET LINE BUFFER TO LINE
        MOV     LINE,A
        MOV     A,LINEBUFFER+1
        MOV     LINE+1,A
        POP     DPH
```

```
        POP     DPL
        RET


;***********INC CURSOR TO NEXT LINE **********
;******* USE A  ****************************
;******END   LINE + 1 ************************

CURLINE:PUSH    DPL
        PUSH    DPH
        MOV     DPTR,#READBUSY
        MOVX    A,@DPTR                 ;GET CURSOR POSITION IN A
        CLR     A.7                     ;CLEAR FIRST BIT
        CJNE    A,#10H,ILINECON2        ;CHECK END OF FIRST LINE
        MOV     A,#0C0H                 ;line 2   set DD ram bit 7 mus 1 40=C0
        LJMP    ILINEEND
ILINECON2:CJNE  A,#50H,ILINECON3        ;CHECK END OF LINE 2
        MOV     A,#90H                  ;THEN LINE 3
        LJMP    ILINEEND
ILINECON3:MOV   A,#0D0H                 ;IF END OF LINE 3 THEN LINE 4
ILINEEND:MOV    DPTR,#COMMAND           ;IF END OF LINE 4 THEN LINE 4
        MOVX    @DPTR,A
        LCALL   WAITBF                  ;Wait busy flag
        MOV     A,LINE
        ADD     A,#1                    ;INC LINE
        DA      A
        MOV     LINE,A
        MOV     A,#0
        ADDC    A,LINE+1                ;ADD CARRY FLAG
        DA      A
        MOV     LINE+1,A
        POP     DPH
        POP     DPL
        RET
```

```
; ***Write ASCII to LCD ***

WRITE:  PUSH    DPL
        PUSH    DPH
        MOV     DPTR,#WRITEDATA
        MOVX    @DPTR,A
        LCALL   WAITBF      ;Wait LCD module ready
        POP     DPH
        POP     DPL
        RET

;*** Wait for ready***************
;*** by mean of check busy flag ***

WAITBF: PUSH    DPL
        PUSH    DPH
        MOV     DPTR,#READBUSY
RDY1:   MOVX    A,@DPTR
        JB      ACC.7,RDY1      ;Busy Flag
        POP     DPH
        POP     DPL
        RET

INIT:   PUSH    DPL
        PUSH    DPH
        MOV     DPTR,#COMMAND
        MOV     A,#38H          ;8bit, 2 line, 5x7 dot
        MOVX    @DPTR,A
        LCALL   WAITBF
        MOV     A,#0FH
        MOVX    @DPTR,A
        LCALL   WAITBF
        MOV     A,#6            ;increment cursor
```

```
        MOVX    @DPTR,A
        LCALL   WAITBF
        MOV     A,#1            ;clear and home
        MOVX    @DPTR,A
        LCALL   WAITBF
        POP     DPH
        POP     DPL
        RET


;**********DISPLAY LINE NUMBER************
;***** INPUT = LINE NUMBER IN LINE*********
;*** LSB = ADDR 34H     MSB = ADDR 35H ***
;***** USE A *****************************

DISLINE:MOV     A,LINE+1                ;GET LINE NUMBER
        ANL     A,#0FH                  ;USE 4 BIT
        ADD     A,#'0'                  ;ADD A + ASCII 0
        LCALL   WRITE

        MOV     A,LINE
        SWAP    A                       ;FOR FIRST NIBBLE
        ANL     A,#0FH                  ;USE 4 BIT
        ADD     A,#'0'                  ;ADD A + ASCII 0
        LCALL   WRITE

        MOV     A,LINE
        ANL     A,#0FH                  ;USE 4 BIT
        ADD     A,#'0'                  ;ADD A + ASCII 0
        LCALL   WRITE

        MOV     A,#' '
        LCALL   WRITE                   ;DISPLAY 2 SPACE
        MOV     A,#' '
        LCALL   WRITE
```

```
      RET

;********* DISPLAY COMMAND*******************
;**** INPUT = COMMAND NUMBER IN REG @DPTR****
;**** USE A,B,R1 ****************************
DISCOM: PUSH    DPL
        PUSH    DPH
        MOVX    A,@DPTR             ;GET DATA TO REG A
        CJNE    A,#' ',DISCON       ;IF SPACE
        MOV     A,#11H              ;THEN
DISCON: CLR     A.6                 ;CLEAR COMMAND CODE
        MOV     B,#7                ;NUMBER IN COMBUFF = DISPLAY COMMAND DATA
        MUL     AB                  ;MUL 7 BECOUSE 1 COMMAND = 7 CHAR
        MOV     DPTR,#COMNO
        ADD     A,DPL
        MOV     DPL,A
        MOV     A,#0
        ADDC    A,DPH
        MOV     DPH,A
        MOV     R1,#7               ;DISPLAY COMMAND 7 CHAR
DLOOP:  MOV     A,#0
        MOVC    A,@A+DPTR           ;DISPLAY IN TABLE
        LCALL   WRITE
        INC     DPTR
        DJNZ    R1,DLOOP
        MOV     A,#' '              ;DISPLAY SPACE
        LCALL   WRITE
        POP     DPH
        POP     DPL
        RET

;******* DISPLAY OPERAND **************
;**** INPUT DATA IN @DPTR *************
```

```
;*** USE A ****************************
DISOPER:PUSH    DPL
        PUSH    DPH
        MOVX    A,@DPTR             ;GET DATA
        CJNE    A,#' ',OPERCON      ;IF SPACE
        LJMP    OPERSP              ;THEN
OPERCON:ADD     A,#'0'              ;CHANGE TO ASCII
OPERSP: LCALL   WRITE
        INC     DPTR
        POP     DPH
        POP     DPL
        RET


;******************SCAN KEYBOARD*********************
;***************** USE R2,R3,R4,DPTR ***************
;************* DATA OUT AT REGISTER  A **************
KEY:    PUSH    DPL
        PUSH    DPH
        PUSH    PSW
        PUSH    A
        MOV     A,R2
        PUSH    A
        MOV     A,R3
        PUSH    A
        MOV     A,R4
        PUSH    A
NOKEY:  MOV     R2,#0               ;COL
        MOV     R4,#0               ;COUNTER
COL:    MOV     A,R2
        ORL     A,#0F8H
        MOV     P1,A                ;SCAN COL
        MOV     R3,#80H             ;ROW
ROW:    MOV     A,P1                ;IN ROW
        ANL     A,R3                ;CHECK = 0 = PRESS
```

```
    JZ     PRESS
    INC    R4                ;INC COUNTER
    MOV    A,R3              ;SHIFT ROW
    RR     A
    MOV    R3,A
    JNB    A.3,ROW
    INC    R2
    MOV    A,R2
    JNB    A.3,COL
    LJMP   NOKEY             ;IF NO PRESS SCAN AGAIN
PRESS: MOV   A,R4
    MOV    DPTR,#KEYTBL      ;CHANGE CODE
    MOVC   A,@A+DPTR         ;OPEN TABLE
    MOV    KEY_DATA,A        ;MOVE DATA TO KEY DATA BUFFER
    POP    A
    MOV    R4,A
    POP    A
    MOV    R3,A
    POP    A
    MOV    R2,A
    POP    A
    MOV    A,KEY_DATA
    POP    PSW
    POP    DPH
    POP    DPL
    RET
;
;********** KEY TABLE ************************
KEYTBL: DB     24H         ;DOWN
```

```
        DB      'ANB    '
        DB      'ORB    '

;***********COMMAND 1 OPERAND*****************
COM1OP  DB      'LD     '
        DB      'LD-NOT '
        DB      'AND    '
        DB      'AND-NOT'
        DB      'OR     '
        DB      'OR-NOT '
        DB      'SET    '
        DB      'RESET  '
        DB      'OUT    '
        DB      'TIM    '
        DB      'TIM-NOT'
        DB      'CNT    '
        DB      'CNT-NOT'
        DB      'K      '
        DB      '       '

;********** TITLE OF ADDRESS FUNCTION ********
ADDR_FUNC       DB      ' ADDRESS  NUMBER     '

;*****************************************************************************************
*****************************
;*****************************************************************************************
*****************************
;*****************************************************************************************
*****************************
;*****************************************************************************************
*****************************
;*****************************************************************************************
*****************************
```

```
        DB      25H             ;UP
        DB      26H             ;DATA
        DB      27H             ;ADDR
        DB      00H             ;0
        DB      03H             ;3
        DB      06H             ;6
        DB      09H             ;9
        DB      23H             ;GO
        DB      02H             ;2
        DB      05H             ;5
        DB      08H             ;8
        DB      40H             ;END
        DB      01H             ;2
        DB      04H             ;4
        DB      07H             ;7
        DB      4BH             ;OUT
        DB      28H             ;NOT
        DB      4AH             ;RES
        DB      22H             ;MON
        DB      42H             ;ORB
        DB      41H             ;ANDB
        DB      49H             ;SET
        DB      21H             ;DEL
        DB      47H             ;OR
        DB      45H             ;AND
        DB      4EH             ;CNT
        DB      20H             ;INS
        DB      43H             ;LD
        DB      50H             ;K
        DB      4CH             ;TIM
        DB      0FFH            ;NO PRESS
        DB      0FFH            ;NO PRESS

;***********COMMAND WITH NO OPERAND***********
COMNO   DB      'END    '
```

;***
****
;***
****
;***
****
;***
****
;***
****
;***
****
;***
****
;***
****
;***
****
;***
****

;***
;***************** COMPLIER ********************
;***
COMPLIER:
    LCALL PMON   ;PIINT MONITOR BACK GROUND
    MOV   TMOD,#1   ;T0 MODE 1
    MOV   TH0,#4CH   ;COUNT UP TO 0000
    MOV   TL0,#00H   ;10000-B400=C400
    SETB   EA
    SETB   ET0
    SETB   TR0   ;RUN TIMER

```
        MOV     R2,#50H
        CLR     A
        MOV     DPTR,#AUXPOINT
COMPLI1:MOVX    @DPTR,A             ;CLEAR TIM CNT AUX
        INC     DPTR
        DJNZ    R2,COMPLI1
        MOV     R2,#6
        MOV     R1,#IN0             ;CLEAR IN,SET,OUT
COMPLI2:MOV     @R1,A
        INC     R1
        DJNZ    R2,COMPLI2

INI:    MOV     RUN_POINTER,#00H
        MOV     RUN_POINTER+1,#30H
        MOV     COM_COUNTER,#00H
        MOV     COM_COUNTER+1,#30H
        MOV     DATA_POINTER,#60H
        MOV     DATA_POINTER+1,#20H
        MOV     ERR_POINTER,#00H
        MOV     ERR_POINTER+1,#3AH
        MOV     ERRBUFF,#0          ;SET NO ERROR

        MOV     DPTR,#PORT_CONTROL
        MOV     A,#DATA_CONTROL
        MOVX    @DPTR,A

        MOV     R0,#50H
        MOV     R5,#27              ;DATA NUMBER
        MOV     DPTR,#RUN_DATA
INI_LOOP1:                          ;MOV DATA TO INT RAM
        CLR     A
        MOVC    A,@A+DPTR
        MOV     @R0,A
```

```
    INC     DPTR
    INC     R0
    DJNZ    R5,INI_LOOP1
    MOV     R0,#50H
    MOV     DPTR,#RUN
    MOV     R1,#27              ;DATA NUMBER
INI_LOOP2:                      ;MOV INT RAM TO RUN
    MOV     A,@R0
    MOVX    @DPTR,A
    INC     R0
    INC     DPTR
    DJNZ    R1,INI_LOOP2        ;GET DATA TO POINTER
    MOV     RUN_POINTER;DPL
    MOV     RUN_POINTER+1,DPH

CPST:  MOV     DPL,DATA_POINTER
    MOV     DPH,DATA_POINTER+1
    LCALL   COM_PLI
    MOV     DPL,DATA_POINTER
    MOV     DPH,DATA_POINTER+1
    INC     DPTR
    INC     DPTR
    INC     DPTR
    INC     DPTR
    MOV     DATA_POINTER,DPL
    MOV     DATA_POINTER+1,DPH
    MOVX    A,@DPTR
    CJNE    A,#40H,CPST_CON1    ;END KEY
    LJMP    CPST_END
CPST_CON1:CJNE    A,#20H,CPST       ;SPACE
    LCALL   OPER_ERR12

CPST_END:
    MOV     DPTR,#RUN_DATA1
    MOV     R7,#3
```

```
        MOV     R0,#50H
        MOV     R5,#3
CPST_END1:                      ;MOV  DATA TO INT RAM
        CLR     A
        MOVC    A,@A+DPTR
        MOV     @R0,A
        INC     DPTR
        INC     R0
        DJNZ    R5,CPST_END1
        MOV     R0,#50H
        MOV     DPL,RUN_POINTER
        MOV     DPH,RUN_POINTER+1
        MOV     R1,#3
CPST_END2:                      ;MOV INT RAM TO RUN
        MOV     A,@R0
        MOVX    @DPTR,A
        INC     R0
        INC     DPTR
        DJNZ    R1,CPST_END2        ;GET DATA TO POINTER
        LJMP    ERRCHECK


;************* SUB ROU TINE COMMAND **********************
;******* IN COMMAND USE R0 TO POINT RUN AREA ************
; if out of this sub MUSE get data at 60h - r0  MOVE to run area

COM_PLI:
        MOV     R0,#50H             ;SET RUN AREA
        MOVX    A,@DPTR             ;GET DATA
        INC     DPTR                ;INC DATD TO POINT OPER 1
        CJNE    A,#43H,COM_2        ;LD
        LCALL   LD
        LJMP    COM_ED
COM_2:
```

```
    CJNE   A,#44H,COM_3          ;LD NOT
    LCALL  LDNOT
    LJMP   COM_ED
COM_3:
    CJNE   A,#45H,COM_4          ;AND
    LCALL  AND
    LJMP   COM_ED
COM_4:
    CJNE   A,#46H,COM_5          ;AND NOT
    LCALL  ANDNOT
    LJMP   COM_ED
COM_5:
    CJNE   A,#47H,COM_6          ;OR
    LCALL  OR
    LJMP   COM_ED
COM_6:
    CJNE   A,#48H,COM_7          ;OR NOT
    LCALL  ORNOT
    LJMP   COM_ED
COM_7:
    CJNE   A,#41H,COM_8          ;ANB
    LCALL  ANB
    LJMP   COM_ED
COM_8:
    CJNE   A,#42H,COM_9          ;ORB
    LCALL  ORB
    LJMP   COM_ED
COM_9:
    CJNE   A,#49H,COM_10         ;SET
    LCALL  SET
    LJMP   COM_ED
COM_10:
    CJNE   A,#4AH,COM_11         ;RES
    LCALL  RES
    LJMP   COM_ED
```

```
COM_11:
    CJNE   A,#4CH,COM_12          ;TIM
    LCALL  TIM
    LJMP   COM_ED
COM_12:
    CJNE   A,#4EH,COM_13          ;CNT
    LCALL  CNT
    LJMP   COM_ED
COM_13:
    CJNE   A,#4BH,COM_END         ;OUT
    MOV    R4,#00H
    LCALL  OUT
    MOV    A,R4
    JNZ    COM_END                ;CHECK OUT CNT
    LJMP   COM_ED

COM_ED:
    CLR    C                      ;GET INT POINT
    MOV    A,R0
    SUBB   A,#50H                 ;GET NUMBER OF DATA
    JZ     COM_ED_2               ;IF NO DATA
    MOV    DPL,RUN_POINTER
    MOV    DPH,RUN_POINTER+1
    MOV    R1,A                   ;LOOP = DATA NUMBER
    MOV    R0,#50H                ;INT RAM POINTER
COM_ED_1:
    MOV    A,@R0
    MOVX   @DPTR,A
    INC    R0
    INC    DPTR
    DJNZ   R1,COM_ED_1            ;GET DATA TO POINTER
    MOV    RUN_POINTER,DPL
    MOV    RUN_POINTER+1,DPH
COM_ED_2:
COM_END:
```

```
;
;
    RET
;
;
;
;*********************************************************
;********** SUB ROU TINE LD ******************************
;*********************************************************
;
LD:   MOVX   A,@DPTR              ;GET OPER 1
      CJNE   A,#0,LD_CHK0         ;IF INPUT FROM PORT
      LCALL  LD_PORT
      LJMP   LD_END
LD_CHK0:CJNE   A,#1,LD_CHK1       ;IF INPUT FROM OUT
      LCALL  LD_OUT
      LJMP   LD_END
LD_CHK1:CJNE   A,#2,LD_CHK2
      INC    DPTR                 ;GET OPER 2
      MOVX   A,@DPTR
      ORL    A,#20H               ;
      CLR    C
      SUBB   A,#20H               ;POINTER IN A
      LJMP   LD_AUX
LD_CHK2:CJNE   A,#3,LD_CHK3
      INC    DPTR                 ;GET OPER 2
      MOVX   A,@DPTR
      ORL    A,#30H               ;
      CLR    C
      SUBB   A,#26H               ;POINTER IN A
      LJMP   LD_AUX
LD_CHK3:CJNE   A,#4,LD_CHK4
      INC    DPTR                 ;GET OPER 2
      MOVX   A,@DPTR
      ORL    A,#40H               ;
      CLR    C
      SUBB   A,#2CH               ;POINTER IN A
      LJMP   LD_AUX
```

```
LD_CHK4:CJNE    A,#5,LD_CHK5
        LJMP    LD_TIM
LD_CHK5:CJNE    A,#6,LD_CHK6
        LJMP    LD_CNT
        LJMP    LD_END
LD_CHK6:LCALL   OPER_ERR1
        LJMP    LD_RET

LD_PORT:
        MOV     A,#A2H
        MOV     @R0,A                   ;GET A2 TO RUN AREA
        INC     R0                      ;INC RUN  AREA
        INC     DPTR                    ;INC DATA POINTER
        MOVX    A,@DPTR                 ;GET OPER 2
        CJNE    A,#0,LD_PORT_B
LD_PORT_A:
        INC     DPTR                    ;INC DATA POINTER
        MOVX    A,@DPTR                 ;GET OPER 3
        MOV     R6,A
        SUBB    A,#8                    ;Check if over 7
        JC      LD_PORT_A1
        LCALL   OPER_ERR2
LD_PORT_A1:
        MOV     A,R6
        MOV     @R0,A                   ;get data to runarea
                                ;because data 0 point port a.0
                                ;                       = bit addr 0
                                ;       data 1 point port a.1
                                ;                       = bit addr 1
        INC     R0
        LJMP    LD_P_END
LD_PORT_B:
        CJNE    A,#1,LD_PORT_B1:        ;IF NO 0 1 IS ERROR
        LJMP    LD_PORT_B2
LD_PORT_B1:
```

```
        LCALL   OPER_ERR3
LD_PORT_B2:
        INC     DPTR                ;INC DATA POINTER
        MOVX    A,@DPTR             ;GET OPER 3
        MOV     R6,A
        SUBB    A,#8                ;Check if over 7
        JC      LD_PORT_B3
        LCALL   OPER_ERR2
LD_PORT_B3:
        MOV     A,R6
        ADD     A,#8H
        MOV     @R0,A
        INC     R0
LD_P_END:RET


LD_OUT: MOV     A,#A2H
        MOV     @R0,A               ;GET A2 TO RUN AREA
        INC     R0                  ;INC RUN  AREA
        INC     DPTR                ;INC DATA POINTER
        MOVX    A,@DPTR             ;GET OPER 2
        CJNE    A,#0,LD_OUTA
        LJMP    LD_OUTB
LD_OUTA:LCALL   OPER_ERR4
LD_OUTB:INC     DPTR                ;INC DATA POINTER
        MOVX    A,@DPTR             ;GET OPER 3
        MOV     R6,A
        SUBB    A,#8                ;Check if over 7
        JC      LD_OUTC
        LCALL   OPER_ERR2
LD_OUTC:MOV     A,R6
        ADD     A,#10H              ;OUT = 10 T0 17
        MOV     @R0,A
        INC     R0
        RET
```

```
LD_AUX: PUSH    DPL
        PUSH    DPH
        MOV     DPTR,#AUXPOINT          ;ADD POINTER IN A
        ADD     A,DPL
        MOV     DPL,A
        CLR     A
        ADDC    A,DPH
        MOV     DPH,A
        MOV     A,#90H                  ;90  =  MOV A,@DPTR
        MOV     @R0,A
        INC     R0
        MOV     A,DPH
        MOV     @R0,A
        INC     R0
        MOV     A,DPL
        MOV     @R0,A
        INC     R0

        MOV     A,#E0H
        MOV     @R0,A
        INC     R0

        MOV     A,#A2H                  ; A2 = MOV C,X
        MOV     @R0,A
        INC     R0

        POP     DPH
        POP     DPL
        INC     DPTR
        MOVX    A,@DPTR                 ;GET OPER 3
        MOV     BUFFER2,A
        SUBB    A,#8                    ;Check if over 7
```

```
        JC      LD_AUX1
        LCALL   OPER_ERR2
LD_AUX1:MOV     A,BUFFER2               ;OPER 3
        ORL     A,#E0H                  ;CHANGE TO E0 - E7
        MOV     @R0,A
        INC     R0
        LJMP    LD_END


LD_TIM: MOV     A,#90H                  ;90 XX YY = LD DPTR,#XXYY
        MOV     @R0,A
        INC     R0
        INC     DPTR                    ;GET OPER 2
        MOVX    A,@DPTR
        CJNE    A,#0,LD_TIM2
        INC     DPTR                    ;GET OPER 3
        MOVX    A,@DPTR
        MOV     BUFFER1,A
        CLR     C
        SUBB    A,#8
        JC      LD_TIM1
        LCALL   OPER_ERR2
LD_TIM1:MOV     A,BUFFER1
        ADD     A,BUFFER1               ;A*2
        MOV     DPTR,#TIMPOINT
        ADD     A,DPL
        MOV     DPL,A
        INC     DPTR                    ;GET MSB
        LJMP    LD_TIM_ED


LD_TIM2:CJNE    A,#1,LD_TIM3
        LJMP    LD_TIM4
LD_TIM3:LCALL   OPER_ERR5
LD_TIM4:INC     DPTR                    ;GET OPER 3
        MOVX    A,@DPTR
```

```
        MOV     BUFFER1,A
        CLR     C
        SUBB    A,#8
        JC      LD_TIM5
        LCALL   OPER_ERR2
LD_TIM5:MOV     A,BUFFER1
        ADD     A,BUFFER1               ;A*2
        ORL     A,#10H                  ;
        MOV     DPTR,#TIMPOINT
        ADD     A,DPL
        MOV     DPL,A
        INC     DPTR                    ;GET MSB
LD_TIM_ED:
        MOV     A,DPH
        MOV     @R0,A                   ;PUT DPH , DPL TO RUN
        INC     R0
        MOV     A,DPL
        MOV     @R0,A
        INC     R0
        MOV     A,#E0H                  ;E0 = MOVX A,@DPTR
        MOV     @R0,A
        INC     R0
        MOV     A,#A2H                  ;A2 E5 = MOV C,A.5
        MOV     @R0,A
        INC     R0
        MOV     A,#E5H
        MOV     @R0,A
        INC     R0
        LJMP    LD_END

LD_CNT: MOV     A,#90H                  ;90 XX YY = LD DPTR,#XXYY
        MOV     @R0,A
        INC     R0
        INC     DPTR                    ;GET OPER 2
        MOVX    A,@DPTR
```

```
        JZ      LD_CNT1
        LCALL   OPER_ERR6
LD_CNT1:INC     DPTR                ;GET OPER 3
        MOVX    A,@DPTR
        MOV     BUFFER1,A
        CLR     C
        SUBB    A,#8
        JC      LD_CNT2
        LCALL   OPER_ERR2
LD_CNT2:MOV     A,BUFFER1
        ADD     A,BUFFER1           ;A*2
        MOV     DPTR,#CNTPOINT
        ADD     A,DPL
        MOV     DPL,A
        INC     DPTR                ;GET MSB
        MOV     A,DPH
        MOV     @R0,A               ;PUT DPH , DPL TO RUN
        INC     R0
        MOV     A,DPL
        MOV     @R0,A
        INC     R0
        MOV     A,#E0H              ;E0 = MOVX A,@DPTR
        MOV     @R0,A
        INC     R0
        MOV     A,#A2H              ;A2 E6 = MOV C,A.5
        MOV     @R0,A
        INC     R0
        MOV     A,#E5H
        MOV     @R0,A
        INC     R0
        LJMP    LD_END


LD_END: MOV     R6,#4
        MOV     DPTR,#LD_DATA
```

```
LD_END1:CLR     A
        MOVC    A,@A+DPTR
        MOV     @R0,A
        INC     R0
        INC     DPTR
        DJNZ    R6,LD_END1
LD_RET: RET

;***********************************************************
;********** SUB ROU TINE LDNOT ***************************
;***********************************************************

LDNOT:  LCALL   LD
        MOV     A,#E6H          ;LD A,@R0
        MOV     @R0,A
        INC     R0
        MOV     A,#F4H          ;CPL A
        MOV     @R0,A
        INC     R0
        MOV     A,#F6H          LD @R0,A
        MOV     @R0,A
        INC     R0
        RET

;***********************************************************
;********** SUB ROU TINE AND *****************************
;***********************************************************

AND:    LCALL   LD
        MOV     A,#E6H          ;MOV A,@R0
        MOV     @R0,A
        INC     R0
        MOV     A,#FAH          ;MOV R2,A
        MOV     @R0,A
```

```
    INC    R0
    MOV    A,#18H              ;DEC R0
    MOV    @R0,A
    INC    R0
    MOV    A,#E6H              ;MOV A,@R0
    MOV    @R0,A
    INC    R0
    MOV    A,#5AH              ;ANL A,R2
    MOV    @R0,A
    INC    R0
    MOV    A,#F6H              ;MOV @R0,A
    MOV    @R0,A
    INC    R0
    RET

;*********************************************************
;********** SUB ROU TINE ANDNOT **************************
;*********************************************************

ANDNOT: LCALL   LDNOT
    MOV    A,#E6H              ;MOV A,@R0
    MOV    @R0,A
    INC    R0
    MOV    A,#FAH              ;MOV R2,A
    MOV    @R0,A
    INC    R0
    MOV    A,#18H              ;DEC R0
    MOV    @R0,A
    INC    R0
    MOV    A,#E6H              ;MOV A,@R0
    MOV    @R0,A
    INC    R0
    MOV    A,#5AH              ;ANL A,R2
    MOV    @R0,A
    INC    R0
```

```
        MOV     A,#F6H              ;MOV @R0,A
        MOV     @R0,A
        INC     R0
        RET

;*********************************************************
;********** SUB ROU TINE OR ******************************
;*********************************************************

OR:     LCALL   LD
        MOV     A,#E6H              ;MOV A,@R0
        MOV     @R0,A
        INC     R0
        MOV     A,#FAH              ;MOV R2,A
        MOV     @R0,A
        INC     R0
        MOV     A,#18H              ;DEC R0
        MOV     @R0,A
        INC     R0
        MOV     A,#E6H              ;MOV A,@R0
        MOV     @R0,A
        INC     R0
        MOV     A,#4AH              ;ORL A,R2
        MOV     @R0,A
        INC     R0
        MOV     A,#F6H              ;MOV @R0,A
        MOV     @R0,A
        INC     R0
        RET

;*********************************************************
;********** SUB ROU TINE ORNOT ***************************
;*********************************************************

ORNOT:  LCALL   LDNOT
```

```
    MOV    A,#E6H              ;MOV A,@R0
    MOV    @R0,A
    INC    R0
    MOV    A,#FAH              ;MOV R2,A
    MOV    @R0,A
    INC    R0
    MOV    A,#18H              ;DEC R0
    MOV    @R0,A
    INC    R0
    MOV    A,#E6H              ;MOV A,@R0
    MOV    @R0,A
    INC    R0
    MOV    A,#4AH              ;ORL A,R2
    MOV    @R0,A
    INC    R0
    MOV    A,#F6H              ;MOV @R0,A
    MOV    @R0,A
    INC    R0
    RET

;*********************************************************
;********** SUB ROU TINE ANB *****************************
;*********************************************************

ANB:  MOV    A,#E6H              ;MOV A,@R0
    MOV    @R0,A
    INC    R0
    MOV    A,#FAH              ;MOV R2,A
    MOV    @R0,A
    INC    R0
    MOV    A,#18H              ;DEC R0
    MOV    @R0,A
    INC    R0
    MOV    A,#E6H              ;MOV A,@R0
    MOV    @R0,A
```

```
        INC     R0
        MOV     A,#5AH              ;ANL A,R2
        MOV     @R0,A
        INC     R0
        MOV     A,#F6H              ;MOV @R0,A
        MOV     @R0,A
        INC     R0
        RET


;*************************************************************
;********** SUB ROU TINE ORB ***************************
;*************************************************************

ORB:    MOV     A,#E6H              ;MOV A,@R0
        MOV     @R0,A
        INC     R0
        MOV     A,#FAH              ;MOV R2,A
        MOV     @R0,A
        INC     R0
        MOV     A,#18H              ;DEC R0
        MOV     @R0,A
        INC     R0
        MOV     A,#E6H              ;MOV A,@R0
        MOV     @R0,A
        INC     R0
        MOV     A,#4AH              ;ORL A,R2
        MOV     @R0,A
        INC     R0
        MOV     A,#F6H              ;MOV @R0,A
        MOV     @R0,A
        INC     R0
        RET
```

```
;*************************************************************
;********** SUB ROU TINE OUT *********************************
;*************************************************************

OUT:   MOV     A,#E6H              ;E6 = MOV A,@R0
       MOV     @R0,A
       INC     R0
       MOV     A,#A2H              ;A2 E7 = MOV C,A.7
       MOV     @R0,A
       INC     R0
       MOV     A,#E7H
       MOV     @R0,A
       INC     R0

       MOVX    A,@DPTR             ;GET OPER 1
       CJNE    A,#1,OUT_CHK1       ;IF 1 = INPUT FROM PORT
       LJMP    OUT_PORT
OUT_CHK1:
       CJNE    A,#2,OUT_CHK2       ;2 = AUX
       INC     DPTR                ;GET OPER 2
       MOVX    A,@DPTR
       ORL     A,#20H              ;
       CLR     C
       SUBB    A,#20H              ;POINTER IN A
       LJMP    OUT_AUX
OUT_CHK2:
       CJNE    A,#3,OUT_CHK3       ;3 = AUX
       INC     DPTR                ;GET OPER 2
       MOVX    A,@DPTR
       ORL     A,#30H              ;
       CLR     C
       SUBB    A,#26H              ;POINTER IN A
```

```
        LJMP    OUT_AUX
OUT_CHK3:
        CJNE    A,#4,OUT_CHK4           ;4 = AUX
        INC     DPTR                    ;GET OPER 2
        MOVX    A,@DPTR
        ORL     A,#40H                  ;
        CLR     C
        SUBB    A,#2CH                  ;POINTER IN A
        LJMP    OUT_AUX
OUT_CHK4:
        CJNE    A,#5,OUT_CHK5           ;5 = TIM
        LJMP    OUT_TIM
OUT_CHK5:
        CJNE    A,#6,OUT_CHK6           ;6 = CNT
        LJMP    OUT_CNT
        LJMP    OUT_END
OUT_CHK6:
        LCALL   OPER_ERR1
        LJMP    OUT_RET

OUT_AUX:PUSH    DPL
        PUSH    DPH
        MOV     DPTR,#AUXPOINT          ;ADD POINTER IN A
        ADD     A,DPL
        MOV     DPL,A
        CLR     A
        ADDC    A,DPH
        MOV     DPH,A
        MOV     A,#90H                  ;90  =  MOV A,@DPTR
        MOV     @R0,A
        INC     R0
        MOV     A,DPH
        MOV     @R0,A
        INC     R0
        MOV     A,DPL
```

```
    MOV     @R0,A
    INC     R0

    MOV     A,#E0H
    MOV     @R0,A
    INC     R0

    MOV     A,#92H              ; A2 = MOV X,C
    MOV     @R0,A
    INC     R0

    POP     DPH
    POP     DPL
    INC     DPTR
    MOVX    A,@DPTR             ;GET OPER 3
    MOV     BUFFER2,A
    SUBB    A,#8                ;Check if over 7
    JC      OUT_AUX1
    LCALL   OPER_ERR2
OUT_AUX1:
    MOV     A,BUFFER2           ;OPER 3
    ORL     A,#E0H              ;CHANGE TO E0 - E7
    MOV     @R0,A
    INC     R0
    LJMP    OUT_END

OUT_TIM:MOV     A,#90H          ;90 XX YY = MOV DPTR,#XXYY
    MOV     @R0,A
    INC     R0
    INC     DPTR                ;GET OPER 2
    MOVX    A,@DPTR
    CJNE    A,#0,OUT_TIM2
    INC     DPTR                ;GET OPER 3
    MOVX    A,@DPTR
```

```
    MOVX   A,@DPTR
    MOV    BUFFER1,A
    CLR    C
    SUBB   A,#8
    JC     OUT_TIM1
    LCALL  OPER_ERR2
OUT_TIM1:MOV   A,BUFFER1
    ADD    A,BUFFER1          ;A*2
    MOV    DPTR,#TIMPOINT
    ADD    A,DPL
    MOV    DPL,A
    INC    DPTR               ;GET MSB
    LJMP   OUT_TIM_ED

OUT_TIM2:
    CJNE   A,#1,OUT_TIM3
    LJMP   OUT_TIM4
OUT_TIM3:
    LCALL  OPER_ERR5
OUT_TIM4:
    INC    DPTR               ;GET OPER 3
    MOVX   A,@DPTR
    MOV    BUFFER1,A
    CLR    C
    SUBB   A,#8
    JC     OUT_TIM5
    LCALL  OPER_ERR2
OUT_TIM5:
    MOV    A,BUFFER1
    ADD    A,BUFFER1          ;A*2
    ORL    A,#10H             ;
    MOV    DPTR,#TIMPOINT
    ADD    A,DPL
    MOV    DPL,A
```

```
    INC    DPTR              :GET MSB
OUT_TIM_ED:
    MOV    A,DPH
    MOV    @R0,A             :PUT DPH , DPL TO RUN
    INC    R0
    MOV    A,DPL
    MOV    @R0,A
    INC    R0
    MOV    A,#E0H            :E0 = MOVX A,@DPTR
    MOV    @R0,A
    INC    R0
    MOV    A,#92H            :92 E6 = MOV A.6,C
    MOV    @R0,A
    INC    R0
    MOV    A,#E6H
    MOV    @R0,A
    INC    R0
    LJMP   OUT_END

OUT_CNT:INC   DPTR           :GET OPER 2
    MOVX   A,@DPTR
    JZ     OUT_CNT1
    LCALL  OPER_ERR6
OUT_CNT1:
    INC    DPTR              ;GET OPER 3
    MOVX   A,@DPTR
    MOV    BUFFER1,A
    CLR    C
    SUBB   A,#8
    JC     OUT_CNT2
    LCALL  OPER_ERR2
OUT_CNT2:
    MOV    A,BUFFER1
    ADD    A,BUFFER1         ;A*2
```

```
    MOV     DPTR,#CNTPOINT
    ADD     A,DPL
    MOV     DPL,A
    INC     DPTR                ;GET MSB

    MOV     COUNTER1,DPL
    MOV     COUNTER2,DPH


    MOV     DPTR,#CNT_DATA
    MOV     BUFFER1,DPL         ;CODE POINTER
    MOV     BUFFER2,DPH
    MOV     R1,#4               ;LOOP = DATA NUMBER
OUT_CNT3:
    MOV     DPL,BUFFER1
    MOV     DPH,BUFFER2         ;CODE POINTER
    CLR     A
    MOVC    A,@A+DPTR
    INC     DPTR
    MOV     BUFFER1,DPL
    MOV     BUFFER2,DPH

    MOV     DPL,RUN_POINTER
    MOV     DPH,RUN_POINTER+1   ;RUN AREA POINTER
    MOVX    @DPTR,A
    INC     DPTR
    MOV     RUN_POINTER,DPL
    MOV     RUN_POINTER+1,DPH

    DJNZ    R1,OUT_CNT3         ;GET DATA TO POINTER

    MOV     DPL,RUN_POINTER     ;GET DPTR NUMBER TO RUN AREA
```

```
    MOV    DPH,RUN_POINTER+1      ;RUN AREA POINTER
    MOV    A,COUNTER2
    MOVX   @DPTR,A
    INC    DPTR
    MOV    A,COUNTER1
    MOVX   @DPTR,A
    INC    DPTR
    MOV    RUN_POINTER,DPL
    MOV    RUN_POINTER+1,DPH


    MOV    DPTR,#CNT_DATA1
    MOV    BUFFER1,DPL            :CODE POINTER
    MOV    BUFFER2,DPH
    MOV    R1,#68                 ;LOOP = DATA NUMBER
OUT_CNT4:
    MOV    DPL,BUFFER1
    MOV    DPH,BUFFER2            :CODE POINTER
    CLR    A
    MOVC   A,@A+DPTR
    INC    DPTR
    MOV    BUFFER1,DPL
    MOV    BUFFER2,DPH

    MOV    DPL,RUN_POINTER
    MOV    DPH,RUN_POINTER+1      :RUN AREA POINTER
    MOVX   @DPTR,A
    INC    DPTR
    MOV    RUN_POINTER,DPL
    MOV    RUN_POINTER+1,DPH

    DJNZ   R1,OUT_CNT4            ;GET DATA TO POINTER

    MOV    R4,#FFH                :SET CHECK
```

```
        RET

OUT_END:MOV     A,#F0H              ;F0 = MOVX @DPTR,A
        MOV     @R0,A
        INC     R0
        MOV     A,#E6H
        MOV     @R0,A
        INC     R0
        MOV     A,#78H
        MOV     @R0,A
        INC     R0
        MOV     A,#5FH
        MOV     @R0,A
        INC     R0
        MOV     A,#F6H
        MOV     @R0,A
        INC     R0

OUT_RET:RET


OUT_PORT:
        INC     DPTR                :OPER 2
        MOVX    A,@DPTR
        JZ      OUT_P_2
        LCALL   OPER_ERR4
OUT_P_2:INC     DPTR
        MOVX    A,@DPTR             :OPER 3
        MOV     R6,A
        SUBB    A,#8                ;Check if over 7
```

```
    JC      OUT_P_3
    LCALL   OPER_ERR2
OUT_P_3:MOV     A,#92H
    MOV     @R0,A
    INC     R0
    MOV     A,R6
    ADD     A,#10H
    MOV     @R0,A
    INC     R0
    MOV     R5,#13
    MOV     DPTR,#OUT_P_DATA
OUT_P_L2:
    CLR     A
    MOVC    A,@A+DPTR
    MOV     @R0,A
    INC     DPTR
    INC     R0
    DJNZ    R5,OUT_P_L2
    RET

;*********************************************************
;********** SUB ROU TINE SET ****************************
;*********************************************************
SET:    MOV     A,#E6H              ;MOV A,@R0
    MOV     @R0,A
    INC     R0
    MOV     A,#A2H              ;A2 E7 = MOV C,A.7
    MOV     @R0,A
    INC     R0
    MOV     A,#E7H
    MOV     @R0,A
    INC     R0
```

```
MOV    A,#50H              ;50 02 = JNC CON
MOV    @R0.A
INC    R0
MOV    A.#08H
MOV    @R0,A
INC    R0
MOV    A,#D2H              ;D2 bit = SET bit
MOV    @R0.A
INC    R0
LCALL  SET_SW
MOV    BUFFER1.A
MOV    @R0.A               :CLEAR DATA
INC    R0
MOV    A.#D2H              :D2 bit = SET bit
MOV    @R0,A
INC    R0
MOV    A,BUFFER1
ADD    A.#18H              :SET = DATA A+18H = 24d
MOV    @R0,A
INC    R0
MOV    A,#E6H
MOV    @R0.A
INC    R0
MOV    A.#78H
MOV    @R0,A
INC    R0
MOV    A,#5FH
MOV    @R0.A
INC    R0
MOV    A.#F6H
MOV    @R0,A
INC    R0
RET
```

```
;*********************************************************
;********** SUB ROU TINE RES ****************************
;*********************************************************
RES:   MOV    A,#E6H              ;MOV A,@R0
       MOV    @R0,A
       INC    R0
       MOV    A,#A2H              ;A2 E7 = MOV C,A.7
       MOV    @R0,A
       INC    R0
       MOV    A,#E7H
       MOV    @R0,A
       INC    R0
       MOV    A,#50H              ;50 04 = JNC CON
       MOV    @R0,A
       INC    R0
       MOV    A,#08H
       MOV    @R0,A
       INC    R0
       MOV    A,#C2H              ;C2 bit = RESET bit
       MOV    @R0,A
       INC    R0
       LCALL  SET_SW
       MOV    BUFFER1,A
       MOV    @R0,A               ;CLEAR DATA
       INC    R0
       MOV    A,#C2H              ;C2 bit = RESET bit
       MOV    @R0,A
       INC    R0
       MOV    A,BUFFER1
       ADD    A,#18H              ;SET = DATA A+18H
       MOV    @R0,A
       INC    R0
```

```
    MOV    A,#E6H
    MOV    @R0.A
    INC    R0
    MOV    A.#78H
    MOV    @R0,A
    INC    R0
    MOV    A,#5FH
    MOV    @R0.A
    INC    R0
    MOV    A.#F6H
    MOV    @R0,A
    INC    R0
    RET

;*********************************************************
;********** SUB ROU TINE SET SWITCH **********************
;*********************************************************

SET_SW: MOVX   A,@DPTR          ;GET OPER 1
    CJNE   A.#0.SET_1
    LJMP   SET_IN           ;0 = SET IN
SET_1:  CJNE   A.#1.SET_2
    LJMP   SET_OUT          ;1 = SET OUT
SET_2:  LCALL  OPER_ERR7        ;IF ERROR
    RET
SET_IN: INC    DPTR             ;INC DATA POINTER
    MOVX   A.@DPTR          ;GET OPER 2
    CJNE   A.#0.SET_IN_B
SET_IN_A:
    INC    DPTR             ;INC DATA POINTER
    MOVX   A.@DPTR          ;GET OPER 3
    MOV    R6.A
    SUBB   A.#8             ;Check if over 7
    JC     SET_IN_A1
```

```
    LCALL   OPER_ERR2
SET_IN_A1:
    MOV     A,R6
    LJMP    SET_END
SET_IN_B:
    CJNE    A,#1,SET_IN_B1          :IF NO 1 IS ERROR
    LJMP    SET_IN_B2
SET_IN_B1:
    LCALL   OPER_ERR3
SET_IN_B2:
    INC     DPTR                    :INC DATA POINTER
    MOVX    A,@DPTR                 :GET OPER 3
    MOV     R6,A
    SUBB    A,#8                    ;Check if over 7
    JC      SET_IN_B3
    LCALL   OPER_ERR2
SET_IN_B3:
    MOV     A,R6
    ADD     A,#8H
    LJMP    SET_END


SET_OUT:INC     DPTR                :INC DATA POINTER
    MOVX    A,@DPTR                 ;GET OPER 2
    CJNE    A,#0,SET_OUTA
    LJMP    SET_OUTB
SET_OUTA:
    LCALL   OPER_ERR4
SET_OUTB:
    INC     DPTR                    :INC DATA POINTER
    MOVX    A,@DPTR                 :GET OPER 3
    MOV     R6,A
    SUBB    A,#8                    ;Check if over 7
```

```
;
        JC      SET_OUTC
        LCALL   OPER_ERR2
SET_OUTC:
        MOV     A,R6
;       ADD     A,#10H              ;OUT = 10 T0 17
;
SET_END:
        RET


;
;*************************************************
;****************** TIM ***************************
;*************************************************

TIM:    MOVX    A,@DPTR             ;GET OPER 1
        CJNE    A,#5,TIM1
        LJMP    TIM2
TIM1:   LCALL   OPER_ERR8
TIM2:   MOV     A,#E6H              ;MOV A,@R0
        MOV     @R0,A
        INC     R0
;       MOV     A,#A2H              ;A2 E7 = MOV C,A.7
        MOV     @R0,A
        INC     R0
;       MOV     A,#E7H
        MOV     @R0,A
;       INC     R0
;       MOV     A,#50H              ;50 15 = JNC TIM1
        MOV     @R0,A
        INC     R0
;       MOV     A,#15H
        MOV     @R0,A
        INC     R0
```

```
        MOV     A,#90H              ;90 XX YY =  MOV DPTR,#XXYY
        MOV     @R0,A
        INC     R0

        INC     DPTR                ;GET OPER 2
        MOVX    A,@DPTR
        CJNE    A,#0,TIM4
        INC     DPTR                ;GET OPER 3
        MOVX    A,@DPTR
        MOV     COUNTER1,A
        CLR     C
        SUBB    A,#8
        JC      TIM3
        LCALL   OPER_ERR2
TIM3:   MOV     A,COUNTER1
        ADD     A,COUNTER1          :A*2
        MOV     DPTR,#TIMPOINT
        ADD     A,DPL
        MOV     DPL,A
        INC     DPTR                ;GET MSB
        LJMP    TIM_ED

TIM4:   CJNE    A,#1,TIM5
        LJMP    TIM6
TIM5:   LCALL   OPER_ERR5
TIM6:   INC     DPTR                :GET OPER 3
        MOVX    A,@DPTR
        MOV     COUNTER1,A
        CLR     C
        SUBB    A,#8
        JC      TIM7
        LCALL   OPER_ERR2
TIM7:   MOV     A,COUNTER1
        ADD     A,COUNTER1          :A*2
```

```
    ORL    A,#10H              :
    MOV    DPTR,#TIMPOINT
    ADD    A,DPL
    MOV    DPL,A
    INC    DPTR                :GET MSB

TIM_ED:
    MOV    A,DPH
    MOV    @R0,A               ;PUT DPH , DPL TO RUN
    INC    R0
    MOV    A,DPL
    MOV    @R0,A
    INC    R0
    MOV    BUFFER1,DPL
    MOV    BUFFER2,DPH

    MOV    R6,#5               ;MOV DATA TO INT RAM
    MOV    DPTR,#TIM_DATA1
TIM_LP1:CLR    A
    MOVC   A,@A+DPTR
    MOV    @R0,A
    INC    R0
    INC    DPTR
    DJNZ   R6,TIM_LP1

;********************** GET K ***************
    MOV    DPL,DATA_POINTER
    MOV    DPH,DATA_POINTER+1
    INC    DPTR                :CHANGE TO NEXT COMMAND
    INC    DPTR
    INC    DPTR
    INC    DPTR
    MOV    DATA_POINTER,DPL
    MOV    DATA_POINTER+1,DPH
```

```
        MOVX    A,@DPTR             ;GET NEXT COMMAND
        CJNE    A.#50H.TIM_K1
        LJMP    TIM_K2
TIM_K1: CLR     A
        MOV     A,DPL
        SUBB    A,#4
        MOV     DPL,A
        MOV     A.DPH
        SUBB    A.#0
        MOV     DPH,A
        MOV     DATA_POINTER,DPL        :GET OLD DATA
        MOV     DATA_POINTER+1,DPH
        LCALL   OPER_ERR10
TIM_K2: MOV     @R0.#74H                ;74 data = MOV A,data
        INC     R0
        INC     DPTR
        MOVX    A.@DPTR             ;GET OPER 1
        MOV     @R0.A
        INC     R0
        INC     DPTR
        MOVX    A,@DPTR             ;GET OPER 2
        MOV     R2,A
        INC     DPTR
        MOVX    A.@DPTR             :GET OPER 3   = 0lsb
        SWAP    A                   ; = lsb0
        ORL     A,R2                ; = lsbmsb
        SWAP    A                   ; = msblsb
        MOV     @R0.#7AH
        INC     R0
        MOV     @R0.A
        INC     R0
```

```
        MOV     R6,#11              :MOV DATA TO INT RAM
        MOV     DPTR,#TIM_DATA2
TIM_LP2:CLR     A
        MOVC    A,@A+DPTR
        MOV     @R0,A
        INC     R0
        INC     DPTR
        DJNZ    R6,TIM_LP2



        MOV     DPL,BUFFER1
        MOV     DPH,BUFFER2
        MOV     A,DPH
        MOV     @R0,A
        INC     R0
        MOV     A,DPL
        MOV     @R0,A
        INC     R0
        MOV     A,#F0H
        MOV     @R0,A
        INC     R0
        MOV     A,#E6H
        MOV     @R0,A
        INC     R0
        MOV     A,#78H
        MOV     @R0,A
        INC     R0
        MOV     A,#5FH
        MOV     @R0,A
        INC     R0
        MOV     A,#F6H
        MOV     @R0,A
        INC     R0
```

```
    RET

;*************************************************
;****************** CNT ******************************
;*************************************************

CNT:   MOVX   A,@DPTR
       CJNE   A,#6,CNT1
       LJMP   CNT2
CNT1:  LCALL  OPER_ERR9
CNT2:  INC    DPTR                ;GET OPER 2
       MOVX   A,@DPTR
       CJNE   A,#0,CNT4
       LJMP   CNT4
CNT3:  LCALL  OPER_ERR6
CNT4:  INC    DPTR                ;GET OPER 3
       MOVX   A,@DPTR
       MOV    COUNTER1,A
       CLR    C
       SUBB   A,#8
       JC     CNT5
       LCALL  OPER_ERR2
CNT5:  MOV    A,COUNTER1
       ADD    A,COUNTER1          ;A*2
       MOV    DPTR,#CNTPOINT
       ADD    A,DPL
       MOV    DPL,A
       INC    DPTR                ;GET MSB
       MOV    BUFFER1,DPL
       MOV    BUFFER2,DPH


       MOV    R6,#6               ;MOV DATA TO INT RAM
       MOV    DPTR,#TIM_DATA      ;USE SAME TIM
```

```
CNT_LP1:CLR    A
        MOVC   A,@A+DPTR
        MOV    @R0,A
        INC    R0
        INC    DPTR
        DJNZ   R6,CNT_LP1

        MOV    DPL,BUFFER1
        MOV    DPH,BUFFER2
        MOV    A,DPH
        MOV    @R0,A              ;PUT DPH , DPL TO RUN
        INC    R0
        MOV    A,DPL
        MOV    @R0,A
        INC    R0

        MOV    R6,#5              ;MOV DATA TO INT RAM
        MOV    DPTR,#TIM_DATA1    ;USE SAME TIM
CNT_LP2:CLR    A
        MOVC   A,@A+DPTR
        MOV    @R0,A
        INC    R0
        INC    DPTR
        DJNZ   R6,CNT_LP2

;************************ GET K ****************
        MOV    DPL,DATA_POINTER
        MOV    DPH,DATA_POINTER+1
        INC    DPTR               ;CHANGE TO NEXT COMMAND
        INC    DPTR
        INC    DPTR
        INC    DPTR
        MOV    DATA_POINTER,DPL
        MOV    DATA_POINTER+1,DPH
```

```
        MOVX    A,@DPTR             ;GET NEXT COMMAND
        CJNE    A,#50H,CNT_K1
        LJMP    CNT_K2
CNT_K1: CLR     A
        MOV     A,DPL
        SUBB    A,#4
        MOV     DPL,A
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
        MOV     DATA_POINTER,DPL    ;GET OLD DATA
        MOV     DATA_POINTER+1,DPH
        LCALL   OPER_ERR11
CNT_K2: MOV     @R0,#74H            ;74 data = MOV A,data
        INC     R0
        INC     DPTR
        MOVX    A,@DPTR             ;GET OPER 1
        MOV     @R0,A
        INC     R0
        INC     DPTR
        MOVX    A,@DPTR             ;GET OPER 2
        MOV     R2,A
        INC     DPTR
        MOVX    A,@DPTR             ;GET OPER 3  = 0lsb
        SWAP    A                   ; = lsb0
        ORL     A,R2                ; = lsbmsb
        SWAP    A                   ; = msblsb
        MOV     @R0,#7AH
        INC     R0
        MOV     @R0,A
        INC     R0

        MOV     R6,#11              ;MOV DATA TO INT RAM
        MOV     DPTR,#TIM_DATA2     ;USE SAME TIM
```

```
CNT_LP3:CLR    A
        MOVC   A.@A+DPTR
        MOV    @R0,A
        INC    R0
        INC    DPTR
        DJNZ   R6,CNT_LP3

        MOV    DPL.BUFFER1
        MOV    DPH.BUFFER2
        MOV    A.DPH
        MOV    @R0,A
        INC    R0
        MOV    A,DPL
        MOV    @R0.A
        INC    R0
        MOV    A,#F0H
        MOV    @R0,A
        INC    R0
        MOV    A,#E6H
        MOV    @R0.A
        INC    R0
        MOV    A.#78H
        MOV    @R0,A
        INC    R0
        MOV    A,#5FH
        MOV    @R0.A
        INC    R0
        MOV    A,#F6H
        MOV    @R0,A
        INC    R0
        RET

;*************************************************************************
;********************* DISPLAY MONNITOR ***************************
```

```
;*************************************************************************

PMON:  MOV    A,#80H              ;CURSOR AT COMMAND LINE 1
       MOV    DPTR,#COMMAND
       MOVX   @DPTR,A
       LCALL  WAITBF              ;Wait busy flag
       MOV    DPTR,#MON_D         :GET DATA
       MOV    R6,#16
PMON_C1:CLR    A                  ;PRINTE LINE 1
       MOVC   A,@A+DPTR
       LCALL  WRITE
       INC    DPTR
       DJNZ   R6,PMON_C1

       MOV    A,#C0H              ;CURSOR AT COMMAND LINE 2
       MOV    DPTR,#COMMAND
       MOVX   @DPTR,A
       LCALL  WAITBF              ;Wait busy flag
       MOV    DPTR,#MON_D1        :GET DATA
       MOV    R6,#16
PMON_C2:CLR    A                  ;PRINTE LINE 2
       MOVC   A,@A+DPTR
       LCALL  WRITE
       INC    DPTR
       DJNZ   R6,PMON_C2

       MOV    A,#90H              ;CURSOR AT COMMAND LINE 3
       MOV    DPTR,#COMMAND
       MOVX   @DPTR,A
       LCALL  WAITBF              ;Wait busy flag
       MOV    DPTR,#MON_D2        :GET DATA
       MOV    R6,#16
PMON_C3:CLR    A                  ;PRINTE LINE 3
       MOVC   A,@A+DPTR
```

```
        LCALL   WRITE
        INC     DPTR
        DJNZ    R6,PMON_C3

        MOV     A,#D0H              ;CURSOR AT COMMAND LINE 4
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag
        MOV     DPTR,#MON_D3        ;GET DATA
        MOV     R6,#16
PMON_C4:CLR     A                   ;PRINTE LINE 4
        MOVC    A,@A+DPTR
        LCALL   WRITE
        INC     DPTR
        DJNZ    R6,PMON_C4

        RET

;*****************************************************************************
;*****************************************************************************
;*****************************************************************************
;*********          INTERUPT  SERVICE  ROUTINE          **********
;*****************************************************************************
;*****************************************************************************
;*****************************************************************************


INT_SER:MOV     TH0,#4CH            ;INT 2 TIME = 0.1ms
        MOV     TL0,#00H
        SETB    TR0
        PUSH    PSW
        PUSH    DPL
        PUSH    DPH
        PUSH    A
```

```
        MOV     INTBUFF+1,R6
        MOV     INTBUFF+2,R7

        MOV     A,INTBUFF
        JNZ     INT_SER1
        LCALL   TIM_SER             ;IF 0  = FIRST 0.05ms
        INC     A
        SJMP    INT_SER_ED
INT_SER1:
        LCALL   MON                 ;IF 1  = SECOUND 0.5 ms
        CLR     A
INT_SER_ED:
        MOV     INTBUFF,A
        MOV     R6,INTBUFF+1
        MOV     R7,INTBUFF+2
        POP     A
        POP     DPH
        POP     DPL
        POP     PSW
        RETI

;*************************************************************************
;************** MONITOR SERVICE ROUTINE **************************
;*************************************************************************
MON:    MOV     A,#C8H              ;CURSOR AT COMMAND LINE 2
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag

        MOV     R7,20H              ;IN0
        LCALL   MON_OUT

        MOV     A,#98H              ;CURSOR AT COMMAND LINE 3
        MOV     DPTR,#COMMAND
```

```
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag

        MOV     R7,21H              ;IN1
        LCALL   MON_OUT

        MOV     A,#D8H              ;CURSOR AT COMMAND LINE 4
        MOV     DPTR,#COMMAND
        MOVX    @DPTR,A
        LCALL   WAITBF              ;Wait busy flag

        MOV     R7,22H              ;OUT
        LCALL   MON_OUT
        RET

MON_OUT:MOV     R6,#8
MON_O_1:MOV     A,R7
        CLR     C
        RRC     A
        MOV     R7,A
        MOV     A,#'0'
        JNC     MON_O_2
        MOV     A,#'1'
MON_O_2:LCALL   WRITE
        DJNZ    R6,MON_O_1
        RET

;*****************************************************************************
;**************  TIMER  SERVICE  ROUTINE  ****************************
;*****************************************************************************

TIM_SER:MOV     A,OUTPUT            ;OUT PORT
        ORL     A,OUTSET
        CPL     A
```

```
        MOV     DPTR,#PORTC
        MOVX    @DPTR,A

        MOV     R6,#16              :TIMER 16 PEACE
        MOV     DPTR,#TIMPOINT
        INC     DPTR                ;TO POINT  BIT & MSB
TSER_C1:MOVX    A,@DPTR
        MOV     C,A.7               ;SET
        JNC     TSER_END
        MOV     C,A.6               ;RUN
        JNC     TSER_END
        MOV     C,A.5               ;ENT COUNT = 0
        JC      TSER_END
        DEC     DPL                 :LSB
        MOVX    A,@DPTR
        CLR     C
        SUBB    A,#1
        JNZ     TSER_C2
        SETB    6DH
TSER_C2:LCALL   SUBDAA              ;CALL SUBDAA
        MOVX    @DPTR,A
        INC     DPTR                ;TO MSB
        MOVX    A,@DPTR
        ANL     A,#0FH              ;CLEAR BIT
        SUBB    A,#0
        JNZ     TSER_C3
        SETB    6CH
TSER_C3:LCALL   SUBDAA              ;CALL SUBDAA

        MOV     C,6DH
        JNC     TSER_C4             :NO ZERO
        MOV     C,6CH
        JNC     TSER_C4             :NO ZERO    SET RUN END  X   OR MSB
        ORL     A,#AFH              ;IF ZERO    1   0   1   0   1111
```

```
        SJMP    TSER_C5
TSER_C4:ORL     A,#C0H              :       1  1  0  0   1111
TSER_C5:MOVX    @DPTR,A

TSER_END:
        CLR     6CH
        CLR     6DH

        INC     DPTR                ;TO NEXT TIMER
        INC     DPTR                ; 1 TIMER = 2 BYTE
        DJNZ    R6,TSER_C1          ;COUNTER TIMER 20 PEACE
        RET

;*************************************************************************
;************************* DATA ***************************************
;*************************************************************************
RUN_DATA:
        DB      78H,5FH,90H,E0H,E0H,E0H,F4H,45H,23H,F5H
        DB      20H,A3H,E0H,F4H,45H,24H,F5H,21H,E5H,22H
        DB      45H,25H,F5H,22H,A3H,F4H,F0H
RUN_DATA1:
        DB      02H,30H,00H

LD_DATA:DB      92H,E7H,08H,F6H

OUT_P_DATA:
        DB      E5H,22H,45H,25H,F4H,90H,E0H,E2H,F0H,E6H
        DB      78H,5FH,F6H

TIM_DATA:
        DB      E6H,A2H,E7H,50H,15H,90H
TIM_DATA1:
        DB      E0H,A2H,E7H,40H,12H
TIM_DATA2:
```

```
        DB      44H,80H,F0H,15H,82H,EAH,F0H,80H,05H,E4H,90H

CNT_DATA:
        DB      E6H,A2H,E7H,90H
CNT_DATA1:                                      ;68
        DB      E0H,40H,05H,C2H,E6H,F0H,80H,34H,A2H,E7H
        DB      50H,30H,A2H,E5H,40H,2CH,A2H,E6H,40H,28H
        DB      15H,82H,E0H,C3H,94H,01H,70H,02H,D2H,6FH
        DB      12H,1FH,00H                     ;CALL SUBDAA
        DB      F0H,A3H,E0H,54H,0FH,94H,00H,70H,02H,D2H,6EH
        DB      12H,1FH,00H                     ;CALL SUBDAA
        DB      A2H,6FH,50H,06H,A2H,6EH,50H,02H,44H,F0H
        DB      44H,C0H,F0H,C2H,6FH,C2H,6EH,E6H,78H,5FH,F6H


MON_D:  DB      'MONITOR 01234567'
MON_D1: DB      'in 0 => 00000000'
MON_D2: DB      'in 1 => 00000000'
MON_D3: DB      'out  => 00000000'


TITLE_DATA:
        DB      ' PLC - 8051    '
        DB      '               '
        DB      '    V 1.0      '
        DB      ' press any key '

;*************************************************
```

```
;****************** ERROR ***************************
;****************************************************
ERRCHECK:
      MOV    A,ERRBUFF
      JNZ    DISERR              ;IF ERROR
      LJMP   RUN


DISERR:
      CLR    TR0
      MOV    DPTR,#ERR_AREA

DISERCON:
      MOV    R1,DPL              ;ERR AREA
      MOV    R2,DPH

      MOVX   A,@DPTR
      MOV    R7,A                ;CODE,MSB IN R7
      INC    DPTR
      MOVX   A,@DPTR             ;LSB IN R6
      MOV    R6,A


      MOV    A,#CLRSCR           ;CLEAR DISPLAY
      MOV    DPTR,#COMMAND
      MOVX   @DPTR,A
      LCALL  WAITBF              ;Wait busy flag

      MOV    DPTR,#ERR_L_DATA
      MOV    R5,#5
DISERRL:MOV  A,#0                ;DISPLAY
      MOVC   A,@A+DPTR
      LCALL  WRITE
```

```
    INC    DPTR
    DJNZ   R5,DISERRL

    MOV    A,R7              :CODE
    ANL    A,#0FH            ;MSB
    ADD    A,#'0'
    LCALL  WRITE
    MOV    A,R6              :LSB
    SWAP   A
    ANL    A,#0FH
    ADD    A,#'0'
    LCALL  WRITE
    MOV    A,R6
    ANL    A,#0FH
    ADD    A,#'0'
    LCALL  WRITE

    MOV    A,R7
    ANL    A,#F0H            :CODE MSB
    SWAP   A                 ;0000 CODE
    CJNE   A,#1,DISERR1
    MOV    DPTR,#ERR_DATA1
    LJMP   DISERR_END
DISERR1:CJNE   A,#2,DISERR2
    MOV    DPTR,#ERR_DATA2
    LJMP   DISERR_END
DISERR2:CJNE   A,#3,DISERR3
    MOV    DPTR,#ERR_DATA3
    LJMP   DISERR_END
DISERR3:CJNE   A,#4,DISERR4
    MOV    DPTR,#ERR_DATA4
    LJMP   DISERR_END
DISERR4:CJNE   A,#5,DISERR5
```

```
        MOV     DPTR,#ERR_DATA5
        LJMP    DISERR_END
DISERR5:CJNE    A,#6,DISERR6
        MOV     DPTR,#ERR_DATA6
        LJMP    DISERR_END
DISERR6:CJNE    A,#7,DISERR7
        MOV     DPTR,#ERR_DATA7
        LJMP    DISERR_END
DISERR7:CJNE    A,#8,DISERR8
        MOV     DPTR,#ERR_DATA8
        LJMP    DISERR_END
DISERR8:CJNE    A,#9,DISERR9
        MOV     DPTR,#ERR_DATA9
        LJMP    DISERR_END
DISERR9:CJNE    A,#0AH,DISERR10
        MOV     DPTR,#ERR_DATA10
        LJMP    DISERR_END
DISERR10:CJNE   A,#0BH,DISERR11     ;IF NO CODE THEN END
        MOV     DPTR,#ERR_DATA11
        LJMP    DISERR_END
DISERR11:CJNE   A,#0CH,DISERR10
        MOV     DPTR,#ERR_DATA12

DISERR_END:
        MOV     R3,DPL
        MOV     R4,DPH


        MOV     DPL,R3
        MOV     DPH,R4
        MOV     R5,#24              ;PRINT LINE 1,8 AND 3 CONTINUE
DISERRL0:
        MOV     A,#0                ;DISPLAY
```

```
    MOVC   A,@A+DPTR
    LCALL  WRITE
    INC    DPTR
    DJNZ   R5,DISERRL0
    MOV    R3,DPL
    MOV    R4,DPH

    MOV    A,#C0H              :MOV CURSOR TO LINE 2
    MOV    DPTR,#COMMAND
    MOVX   @DPTR,A
    LCALL  WAITBF              :Wait busy flag

    MOV    DPL,R3
    MOV    DPH,R4
    MOV    R5,#31              ;PRINT LINE 2 AND 4 DISPLAY CURSOR
DISERL1:MOV    A,#0            ;DISPLAY
    MOVC   A,@A+DPTR
    LCALL  WRITE
    INC    DPTR
    DJNZ   R5,DISERL1

DISERLKEY
    LCALL  DELAY
    LCALL  KEY
    MOV    A,KEY_DATA
    CJNE   A,#25H,DISERL2
    LJMP   DISERLUP            ;UP
DISERL2:CJNE   A,#24H,DISERLKEY
                               :DOWN
    MOV    DPL,R1              :ERROR COUNTER
    MOV    DPH,R2
    INC    DPTR                ;TO NEXT ERR
    INC    DPTR
    MOV    R1,DPL
```

```
        MOV     R2,DPH
        CLR     C
        MOV     A,ERR_POINTER           :LSB = DPL
        SUBB    A,DPL
        JNZ     DISERL3
        MOV     A,ERR_POINTER+1
        SUBB    A,DPH                   ;MSB = DPH
        JZ      DISED                   ;IF END

DISERL3:LJMP    DISERCON                ;NO END

DISED:  CLR     C                       ;IF END DISPLAY OLD
        MOV     A,DPL
        SUBB    A,#2                    ;ERR COUNT - 2
        MOV     DPL,A
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
        LJMP    DISERCON

DISERLUP:
        MOV     DPL,R1                  ;ERROR COUNTER
        MOV     DPH,R2
        MOV     A,R1                    ;LOWEST = 3A00H
        CJNE    A,#0,DISERLUP1
        MOV     A,R2
        CJNE    A,#3AH,DISERLUP1
        LJMP    DISERCON                ;IF LOWEST NO - 2
DISERLUP1:
        CLR     C                       ;ERR COUNT - 2
        MOV     A,DPL
        SUBB    A,#2
        MOV     DPL,A
```

```
        MOV     A,DPH
        SUBB    A,#0
        MOV     DPH,A
        LJMP    DISERCON

ERR_L_DATA:
        DB      'LINE '

ERR_DATA1:
        DB              ' ERR 1 '
        DB      'invalide number '      ;line 3
        DB      'operand 1       '      ;line 2
        DB      'up dn or reset  '      ;line 4

ERR_DATA2:
        DB              ' ERR 2 '
        DB      'muse use 0 - 7  '      ;line 3
        DB      'operand 3       '      ;line 2
        DB      'up dn or reset  '      ;line 4
ERR_DATA3:
        DB              ' ERR 3 '
        DB      'input use 0,1   '
        DB      'operand 1       '
        DB      'up dn or reset  '
ERR_DATA4:
        DB              ' ERR 4 '
        DB      'output must 0   '
        DB      'operand 2       '
        DB      'up dn or reset  '
ERR_DATA5:
        DB              ' ERR 5 '
        DB      'timer use 0,1   '
        DB      'operand 2       '
```

```
        DB      'up dn or reset  '
ERR_DATA6:
        DB              ' ERR 6 '
        DB      'counter must 0  '
        DB      'operand 2       '
        DB      'up dn or reset  '
ERR_DATA7:
        DB              ' ERR 7 '
        DB      'set use  0,1    '
        DB      'operand 1       '
        DB      'up dn or reset  '
ERR_DATA8:
        DB              ' ERR 8 '
        DB      'timer must 5    '
        DB      'operand 1       '
        DB      'up dn or reset  '
ERR_DATA9:
        DB              ' ERR 9 '
        DB      'counter must 6  '
        DB      'operand 1       '
        DB      'up dn or reset  '
ERR_DATA10:
        DB              ' ERR 10'
        DB      'timer no K      '
        DB      'operand 1       '
        DB      'up dn or reset  '
ERR_DATA11:
        DB              ' ERR 11'
        DB      'counter no K    '
        DB      'operand 1       '
        DB      'up dn or reset  '
ERR_DATA12:
        DB              ' ERR 12'
        DB      'space in program'
```

```
        DB      'no have end or '
        DB      'up dn or reset '

OPER_ERR1:
        PUSH    A
        PUSH    PSW
        MOV     A,#10H
        LJMP    OPER_ERRE
OPER_ERR2:
        PUSH    A
        PUSH    PSW
        MOV     A,#20H
        LJMP    OPER_ERRE
OPER_ERR3:
        PUSH    A
        PUSH    PSW
        MOV     A,#30H
        LJMP    OPER_ERRE
OPER_ERR4:
        PUSH    A
        PUSH    PSW
        MOV     A,#40H
        LJMP    OPER_ERRE
OPER_ERR5:
        PUSH    A
        PUSH    PSW
        MOV     A,#50H
        LJMP    OPER_ERRE
OPER_ERR6:
        PUSH    A
```

```
        PUSH    PSW
        MOV     A.#60H
        LJMP    OPER_ERRE
OPER_ERR7:
        PUSH    A
        PUSH    PSW
        MOV     A,#70H
        LJMP    OPER_ERRE
OPER_ERR8:
        PUSH    A
        PUSH    PSW
        MOV     A.#80H
        LJMP    OPER_ERRE
OPER_ERR9:
        PUSH    A
        PUSH    PSW
        MOV     A,#90H
        LJMP    OPER_ERRE
OPER_ERR10:
        PUSH    A
        PUSH    PSW
        MOV     A.#A0H
        LJMP    OPER_ERRE
OPER_ERR11:
        PUSH    A
        PUSH    PSW
        MOV     A,#B0H
        LJMP    OPER_ERRE
OPER_ERR12:
        PUSH    A
        PUSH    PSW
        MOV     A.#C0H

OPER_ERRE:
```

```
        PUSH    DPL
        PUSH    DPH

        MOV     INTBUFF,R2
        MOV     INTBUFF+1,R3        ;COUNT  LSB
        MOV     INTBUFF+2,R4            MSB
        MOV     INTBUFF+3,R5        ;LINE   LSB
        MOV     INTBUFF+4,R6            MSB

        MOV         R2,A            :BUFFER ERR CODE

        MOV     R3,#60H
        MOV     R4,#20H
        MOV     R5,#0
        MOV     R6,#0
OP_ER_CMP:                      :COMPAIR
        CLR     C
        MOV     A,DATA_POINTER      ;LSB
        SUBB    A,R3
        JZ      OP_ER_CMPZ          ;ZERO MUST ZERO 2 PEACE
        JC      OP_ER_CMPC          :CARRY MUST CARRY 2 PEACE
        LJMP    OP_ER_EX

OP_ER_CMPZ:
        CLR     C
        MOV     A,DATA_POINTER+1    :MSB
        SUBB    A,R4
        JNZ     OP_ER_EX
        LJMP    OP_ER_CON           ;IF EQUARE

OP_ER_CMPC:
        MOV     A,DATA_POINTER+1    :MSB
        SUBB    A,R4
        JNC     OP_ER_EX
```

```
        LJMP    OP_ER_CON           ;IF CARRY 2 PEACE

OP_ER_EX:
        MOV     A,R3                :COUNT + 4
        ADD     A,#4
        MOV     R3,A
        MOV     A,R4
        ADDC    A,#0
        MOV     R4,A

        MOV     A,R5                ;LINE + 1 DAA
        ADD     A,#1
        DA      A
        MOV     R5,A
        MOV     A,R6
        ADDC    A,#0
        DA      A
        MOV     R6,A

        LJMP    OP_ER_CMP


OP_ER_CON:
        MOV     A,R2                ;  CODE 0000 + 0000 MSB
        ORL     A,R6                ;     = CODE MSB
        MOV     ERRBUFF,#FFH
        MOV     DPL,ERR_POINTER
        MOV     DPH,ERR_POINTER+1
        MOVX    @DPTR,A             ;GET CODE MSB TO ERR POINTER
        INC     DPTR
        MOV     A,R5                ;LSB
        MOVX    @DPTR,A
        INC     DPTR                :CHANGE ERR BUFFER
        MOV     ERR_POINTER,DPL
```

```
        MOV     ERR_POINTER+1,DPH
        MOV     R2,INTBUFF
        MOV     R3,INTBUFF+1
        MOV     R4,INTBUFF+2
        MOV     R5,INTBUFF+3
        MOV     R6,INTBUFF+4
        POP     DPH
        POP     DPL
        POP     PSW
        POP     A
        RET


        END
```

บรรณานุกรม

1. กฤษดา วิศวธีรานนท์. "PC ตัวควบคุมซีเควนซ์ หลักการทำงานและการประยุกต์": โรงพิมพ์จุฬาลงกรณ์มหาวิทยาลัย,2534.

2. บริษัท อีทีทีจำกัด. DOT MATRIX LCD MODULE. กรุงเทพมหานคร: โรงพิมพ์ทิพยวิสุทธ์, 2535

3. บริษัท อีทีทีจำกัด. ET-DCIN8 OPTOISOLATE INPUT BOARD. กรุงเทพมหานคร: โรงพิมพ์ทิพยวิสุทธ์, 2535

4. บริษัท อีทีทีจำกัด. ET-SSRAC SOLID-STATE RELAY. กรุงเทพมหานคร: โรงพิมพ์ทิพยวิสุทธ์, 2535

5. บริษัท อีที จำกัด. PC-SB31 USER MANUAL. กรุงเทพมหานคร: โรงพิมพ์ทิพยวิสุทธ์, 2535