พีแอลซี

PLC (PROGRAMMABLE LOGIC CONTROLLER)

โดย

นายชนาเนตร์ ชวรางกูร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอบเกล้าเจ้าคุณทหารลากกระบัง
ปีการศึกษา 2542

พีแอลซี

PLC (PROGRAMMABLE LOGIC CONTROLLER)

โดย

นายชนาเนตร์ ชวรางกูร เลขประจำตัว 40013166

อาจารย์ที่ปรึกษา

อาจารย์สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2542

ปริญญานิพนธ์ ปีการศึกษา 2542

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง พีแอลซี

ผู้จัดทำ

นายชนาเนตร์ ชวรางกูร เลขประจำตัว 40013166

.....อาจารย์ที่ปรึกษา

(.....)

โครงงานเรื่อง พีแอลซี

PROGRAMMABLE LOGIC CONTROLLER

จัดทำโดย

นายชนาเนตร์ ชวรางกูร เลขประจำตัว 40013166

โครงงานนี้ได้รับการตรวจสอบแล้วพร้อมที่จะทำการสอบได้

.....

(.....)

อาจารย์ที่ปรึกษา

พีแอลซี

นายชนาเนตร์ ชวรางกูร
อาจารย์สุรพันธ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

บทคัดย่อ

พีแอลซี ใช้แทนวงจรรีเลย์ในการควบคุมอุปกรณ์หรือเครื่องจักรในโรงงานอุตสาหกรรม โดยได้ทำการจำลองพีแอลซีขึ้นมาโดยใช้ไมโครคอนโทรลเลอร์ MCS-51 เป็นตัวประมวลผลและใช้โปรแกรมภาษาเดลไฟล์ (Delphi) เป็นตัวแปลงภาษาบูลลีน ที่เขียนจากผู้ใช้มาเป็นไฟล์ออปเจ้คโค้ดของไมโครคอนโทลเลอร์แล้วให้ไมโครคอนโทรเลอร์ประมวลผลและส่งผลที่ประมวลได้ไปที่เอาท์พุต โดยแต่ละบิตของเอาท์พุตจะแสดงผลโดยใช้ LED โดยในภาคเรียนนี้ทำการพัฒนาโปรแกรมภาษาแลดเดอร์ ให้สามารถใช้งานได้บนระบบปฏิบัติการวินโดวส์ ซึ่งทำให้โปรแกรมที่สร้างสามารถใช้งานได้ง่ายขึ้น

PLC
(PROGRAMMABLE LOGIC CONTROLLER)

Mr. Chananate Chawarangkul
Dr. Surapan Ouapriboon Advisor
1999

ABSTRACT

PLC (PROGRAMMABLE LOGIC CONTROLLER) is used instead of relays for controlling equipment or machines in the factories. This project uses microcontroller MCS – 51 to be CPU (Central Processing Unit) and uses Delphi language program to translate the symbols of Boolean language from users to objectcode file of microcontroller. Then users will send the Program to microcontroller by RS – 232. The microcontroller will store and process the program. Finally it will send the result to output port. Each bit of output shows the result of program by using LEDs. This semester users can use PLC with Windows operation system that users can use it easily.

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

พีแอลซี (PLC) ย่อมาจากคำว่า "Programmable Logic Controller" เป็นอุปกรณ์ควบคุมอิเล็กทรอนิกส์ที่มีหน่วยความจำในการเก็บโปรแกรมสำหรับควบคุมการทงานของอุปกรณ์ต่าง ๆ ที่ต่อกับขั้วเข้า และขั้วออกของมัน พีแอลซี นี้ยังมีชื่อเรียกอย่างอื่นอีก เช่น PC ซึ่งย่อมาจาก "Programmable Controller" และ SC ซึ่งย่อมาจาก Sequence Controller PC ขนาดเล็กอาจเรียกเป็นตัวซีเควนซ์เซอร์ (Sequencer) ก็มี

พีแอลซี ได้รับการพัฒนาเพื่อใช้ทดแทนระบบควบคุมแบบเก่าที่ใช้รีเลย์ซึ่งติดตั้งดัดแปลงและแก้ไขลำบากมาเป็นระบบควบคุมใหม่ที่ใช้วงจรอิเล็กทรอนิกส์แทนรีเลย์และใช้การเขียนโปรแกรมทำนองเดียวกับการเขียนโปรแกรมคอมพิวเตอร์กำหนดเงื่อนไขการควบคุมแทนการเดินสายเชื่อมต่อวงจรไฟฟ้าแบบเก่าเพื่อเพิ่มความสะดวกระยะแรก

ระยะแรกพีแอลซีถูกใช้แทนวงจรรีเลย์ในการควบคุมเครื่องจักรที่มีการควบคุมแบบ ปิด/เปิด เท่านั้น หลังจากนั้นเนื่องจากพีแอลซีมีขนาดเล็ก นำหนักเบา ติดตั้งบำรุงรักษาง่ายใช้ไฟฟ้าน้อยกว่ารีเลย์ทำให้มีผู้นำ พีแอลซี มาใช้ในโรงงานอุตสาหกรรมต่าง ๆ เพิ่มขึ้น

วัตถุประสงค์

1. สามารถสร้าง พีแอลซี ขึ้นได้เองทั้ง ซอฟต์แวร์ และ ฮาร์ดแวร์
2. สามารถสร้างฮาร์ดแวร์ของ พีแอลซี โดยใช้ตัวประมวลผลที่มีประสิทธิภาพดี, ใช้งานง่าย และราคาถูก ซึ่งก็คือการใช้ชิพตระกูล เอ็มซีเอส-51 (MCS-51)
3. สามารถสร้างซอฟต์แวร์ของ พีแอลซี ที่สามรถเขียนโปรแกรมในรูปของภาษาบูลลีน ในขณะเดียวกันสามารถแสดงภาพโปรแกรมภาษาแลดเดอร์ได้ ซึ่งเมื่อเขียนโปรแกรมเสร็จแปลงเป็นภาษาแอสแซมบลีของไมโครคอนโทรลเลอร์ ตระกูล เอ็มซีเอส-51 ในขณะแปลงภาษาสามารถตรวจสอบความถูกต้องได้ และ เมื่อแปลงเสร็จโปรแกรมสามารถส่งโปรแกรมไปเก็บข้อมูลบนหน่วยความจำของเครื่อง พีแอลซี ได้
4. การใช้งานได้จริงในระดับที่น่าพอใจ
5. สามารเสร้าง พีแอลซี ที่ใช้ต้นทุนต่ำ
6. เป็นกรณีศึกษาเพื่อการพัฒนาสิ่งที่ดีกว่า

ขอบเขตของงานที่ทำ

1. สามารถทำการเขียนโปรแกรมได้ง่าย, สามารถใช้งานได้ง่าย และ ใช้งานได้จริง
2. การใช้งานเป็นการอินเทอร์เฟส (Interface) โดยการเขียนโปรแกรมควบคุมบนเครื่องคอมพิวเตอร์ (PC) และส่งผ่านโปรแกรมโดยผ่านการสื่อของ RS-232 ไปอยู่ภายใต้การควบคุมของไมโครคอนโทรลเลอร์
3. พีแอลซี สามารถมีอินพุตและเอาท์พุตหลักบนตัวเครื่อง อย่างละ 8 ชุด
4. พีแอลซี สามารถเพิ่มจำนวนอินพุตและเอาท์พุตได้ โดยทั้งซอฟต์แวร์และฮาร์ดแวร์สามารถรองรับได้
5. ซอฟต์แวร์บนเครื่องคอมพิวเตอร์ สามารถทำการแปลภาษาจาก ภาษาบูลลีนของ พีแอลซี เป็นภาษาของ ไมโครคอนโทรลเลอร์ ในรูปแบบของ Hex File ได้
6. มีความสามารถทำคำสั่งหลัก ๆ ของภาษา พีแอลซี ได้

ประโยชน์หรือผลที่คาดว่าจะได้รับ

1. ได้พีแอลซีใช้ทดแทนระบบควบคุมแบบเก่าที่ใช้รีเลย์ซึ่งติดตั้งดัดแปลงและแก้ไขลำบากมาเป็น ระบบควบคุมใหม่ที่ใช้วงจรอิเล็กทรอนิกส์แทนรีเลย์และใช้การเขียนโปรแกรมทำนองเดียวกับการเขียนโปรแกรมคอมพิวเตอร์กำหนดเงื่อนไขการควบคุมแทนการเดินสายเชื่อมต่อวงจรไฟฟ้าแบบเก่าเพื่อเพิ่มความสะดวก
2. สร้างความเข้าใจในเรื่องของ พีแอลซี แก่ผู้ทำการศึกษานี้
3. เป็นแนวทางการศึกษา และ พัฒนา แก่ผู้สนใจทำการพัฒนาต่อไปเพื่อส่งที่ดีกว่า
4. สามารถสร้าง พีแอลซี ได้เอง โดยมีค่าใช้จ่ายน้อย

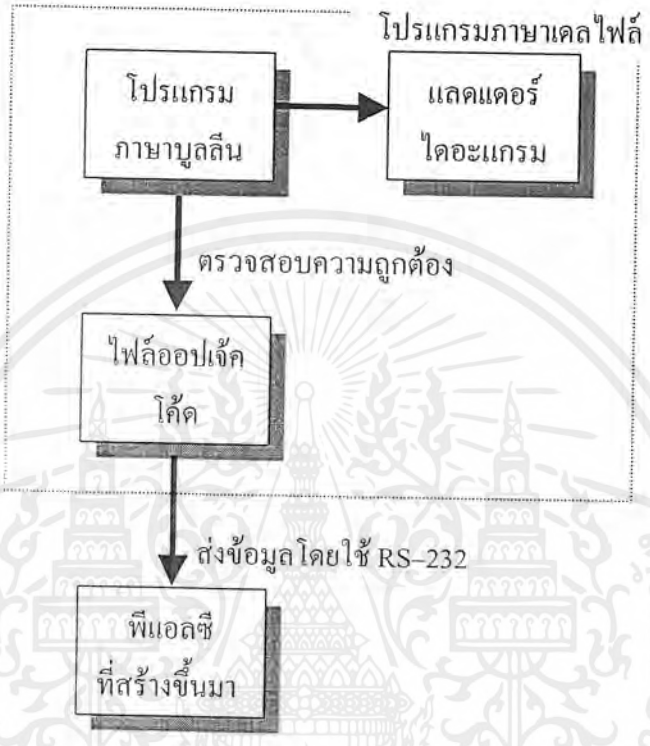
ขั้นตอนการทำโครงงาน

1. ศึกษาโครงสร้างและการทำงานของ พีแอลซี
2. ศึกษาภาษาบูลลีนและแลดเดอร์ไดอะแกรมของ พีแอลซี, ภาษาแอสแซมบลีของ เอ็มซีเอส-51 และการเขียนโปรแกรมเดลไฟล์ เพื่อสร้างซอฟต์แวร์
3. ศึกษาการทำงานของไมโครคอนโทรลเลอร์ ตระกูล เอ็มซีเอส-51 เพื่อสร้างฮาร์ดแวร์

การทำงานของโครงงานพีแอลซีนี้แบ่งเป็นขั้นตอนการทำงานได้ดังนี้

1. เริ่มจากผู้เขียนโปรแกรมภาษาบูลลีน ในเครื่องคอมพิวเตอร์
2. ในระหว่างการเขียนโปรแกรมภาษาบูลลีน สามรถเรียกดูแลดเดอร์ไดอะแกรมได้
3. จากนี้ส่วนของโปรแกรมแปลงภาษาบูลลีน แล้วแปลงเป็นภาษาแอสแซมบลี

4. แล้วส่งโปรแกรมภาษาแลดเดอร์ที่ถูกแปลงเป็นไฟล์ออปเจ้คโค้ดแล้วจากเครื่องคอมพิวเตอร์โดยใช้การส่งแบบ RS – 232 ไปยังหน่วยประมวลผลของพีแอลซี
5. หน่วยประมวลผลจะทำการประมวลผลแล้วส่งค่าออกทางเอาต์พุตที่แสดงผลโดยใช้แอลอีดี (LED) โครงสร้างการทำงานของพีแอลซีเป็นดังรูปที่ 1.1



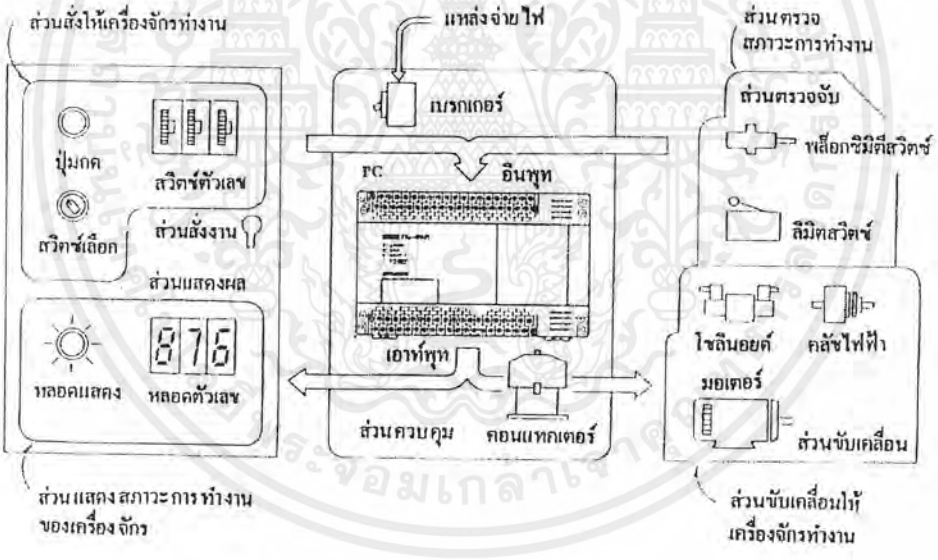
รูปที่ 1.1 แสดงขั้นตอนการใช้งานพีแอลซีที่สร้างขึ้นมา

บทที่ 2
โครงสร้างการทำงานของพีแอลซี

พีแอลซี นี้ถือเป็นอุปกรณ์ควบคุมตัวสำคัญตัวหนึ่งในการควบคุมเครื่องจักรในโรงงานอุตสาหกรรมให้ทำงานอย่างอัตโนมัต ในระบบ FA (Factor Automation) พีแอลซี จะถูกใช้ในการปรับปรุงประสิทธิภาพการทำงานของเครื่องจักร ลดของเสีย ทำให้เครื่องจักรทำงานได้เองโดยอัตโนมัติ เป็นการลดงานของคนงาน พีแอลซี ที่ใช้ในโรงงานมีตั้งแต่ พีแอลซี ที่มีขนาดใหญ่หรือเป็นระบบควบคุม จนกระทั่งถึง พีแอลซี ขนาดเล็กซึ่งใช้ตัวเดียวโดด ๆ

ในบทนี้ จะเน้น พีแอลซี ขนาดเล็กที่ใช้งานตัวเดียวโดย ๆ จะแนะนำให้รู้จักโครงสร้างการทำงาน และวิธีนำมาใช้งานในโรงงานอุตสาหกรรม

2.1 การทำงานของพีแอลซี



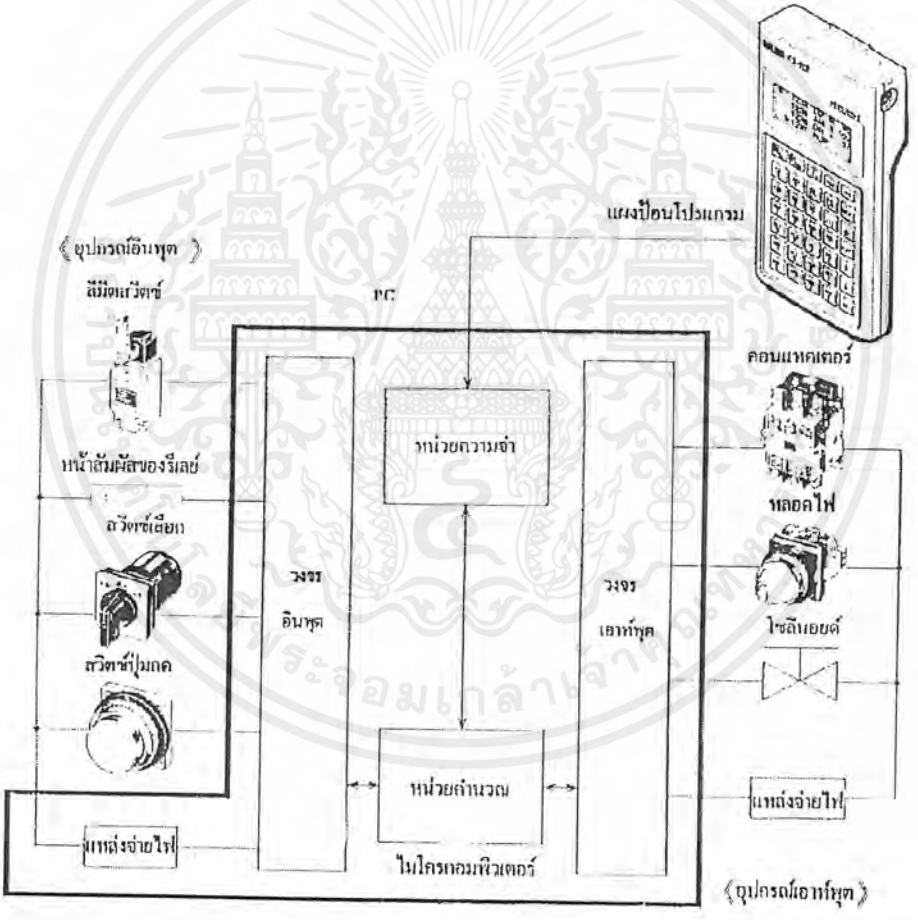
รูปที่ 2.1 แสดงการทำงานของพีแอลซี

พีแอลซีจะรับสัญญาณที่เป็นคำสั่งจากสวิตช์ปุ่มกด สวิตช์เลือก และสวิตช์ตัวเลข ซึ่งอยู่ที่แผงควบคุมเครื่องจักร นอกจากนั้น ยังรับสัญญาณที่มาจากอุปกรณ์ตัววัดสภาพการทำงานของเครื่องจักร เช่น ลิมิตสวิตช์ (Limit switch) พล็อกซิมิตีสวิตช์ (Proximity switch) สวิตช์แสง และสวิตช์ตรวจจับชนิดต่าง ๆ จากนั้น พีแอลซี จะส่งสัญญาณออกไปที่ขั้วออกเพื่อขับเคลื่อนอุปกรณ์ต่าง ๆ เช่น มอเตอร์โซลินอยด์ และคลัตช์แม่เหล็กไฟฟ้า หรือขับพวกหลอดแสง หลอดตัวเลข

พีแอลซีจะรับสัญญาณเข้ามาทางขั้วเข้า และให้สัญญาณออกทางขั้วออก การให้สัญญาณออกนี้จะเป็นไปตามโปรแกรมที่เก็บไว้ในเครื่อง พีแอลซี การขับอุปกรณ์ซึ่งเป็นโหลดขนาดเล็ก เช่น โซ-ลินอยด์ตัวเล็ก หรือหลอดแสดงนั้น พีแอลซี สามารถขับโดยตรงจากขั้วออกได้ แต่ถ้าเป็นอุปกรณ์ที่เป็นโหลดขนาดใหญ่ ใช้ไฟมาก เช่น มอเตอร์สามเฟส หรือ โซลินอยด์ตัวใหญ่ จำเป็นต้องต่อผ่านคอนแทคเตอร์ และรีเลย์เพื่อช่วยขยายกำลังขับ พวกคอนแทคเตอร์ รีเลย์ขับ เบรกเกอร์ เหล่านี้จะถูกติดตั้งอยู่ภายในตู้ควบคุมเดียวกันกับตัว พีแอลซี

2.2 โครงสร้างพีแอลซี

พีแอลซีเป็นอุปกรณ์คอมพิวเตอร์สำหรับใช้งานในอุตสาหกรรม

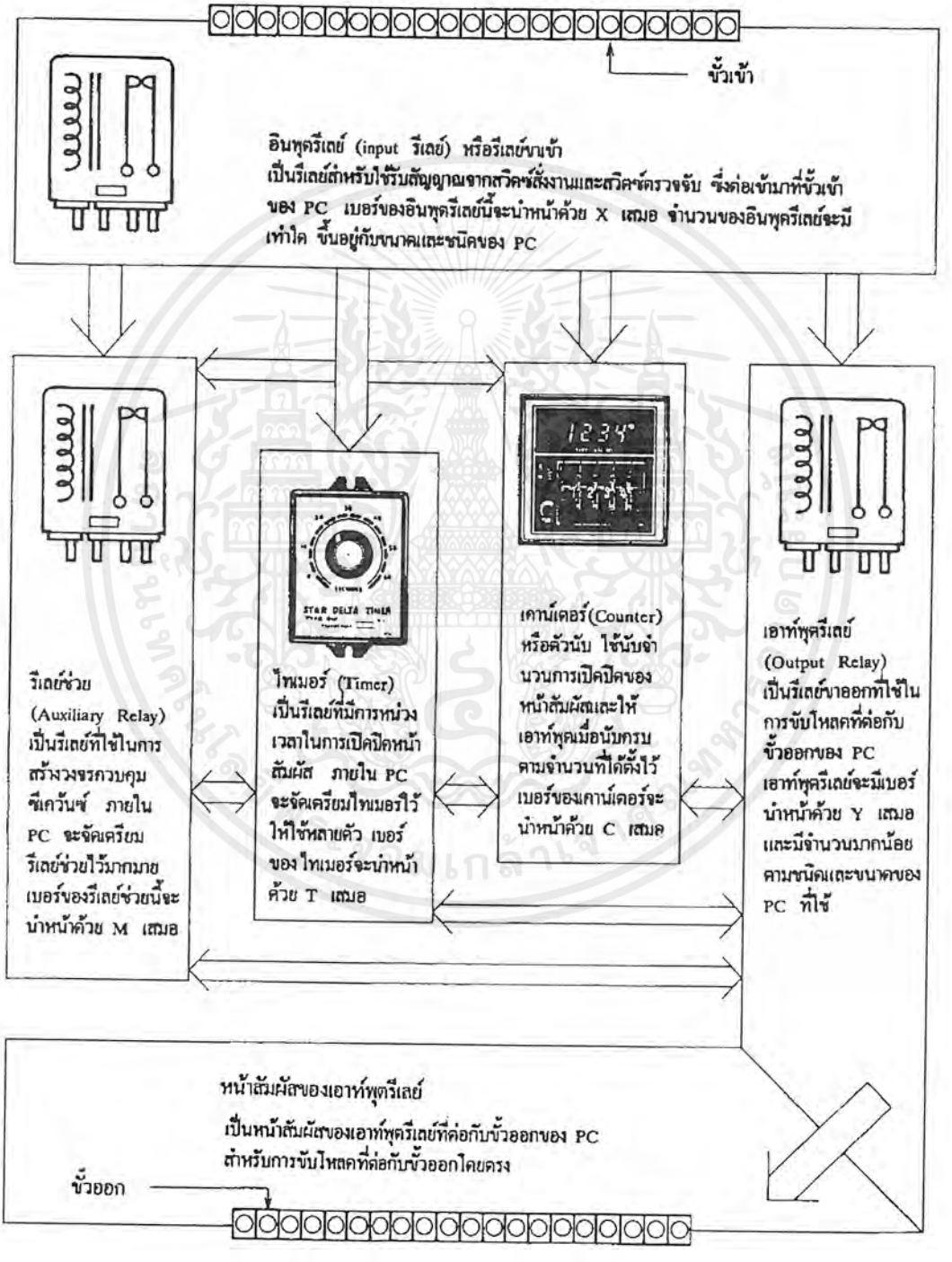


รูปที่ 2.2 แสดงโครงสร้างพีแอลซี

ภายใน พีแอลซี จะมีไมโครคอมพิวเตอร์ และหน่วยความจำเป็นองค์ประกอบวงจรที่สำคัญ ยังมีวงจรอินพุตและวงจรเอาท์พุตสำหรับการต่อกับอุปกรณ์ภายนอก นอกจากนั้น ยังใช้แผงป้อนโปรแกรมในการอ่านและเขียนโปรแกรมเข้าไปในหน่วยความจำของพีแอลซี

2.3 อุปกรณ์เสมือนภายในพีแอลซี

ภายในพีแอลซีจะมีอุปกรณ์ที่ใช้ในการควบคุมซีเควันซ์ เช่น รีเลย์ ไทเมอร์ เคาน์เตอร์ และอุปกรณ์อื่น ๆ อีก อุปกรณ์แต่ละตัวจะมีหน้าสัมผัสทั้งชนิด ปรกติปิด(NC : Normal Close) และปรกติเปิด(NO : Normal Open) และมีหน้าสัมผัสมากมายนับไม่ถ้วน การโปรแกรมให้พีแอลซีทำงานก็คือ การสร้างวงจรรีเลย์โดยการนำองค์ประกอบเหล่านี้มาต่อเป็นวงจรนั่นเอง



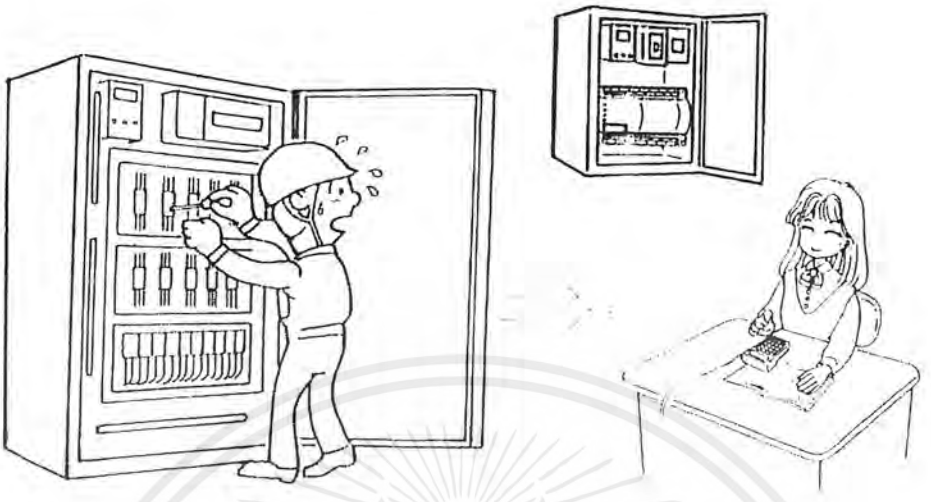
รูปที่ 2.3 แสดงอุปกรณ์ภายในพีแอลซี

2.4 สัญลักษณ์ในวงจรแลดเดอร์

อุปกรณ์ \ หน้าสัมผัส	ชนิด NO		ชนิด NC		แหล่งขับเคลื่อน
	แนวราบ	แนวตั้ง	แนวราบ	แนวตั้ง	
ปุ่มกด (กดติดปล่อยดับ)					
ปุ่มกดแบบล็อก					
รีเลย์ความร้อน (OCR)					ธีทเคอร์ ทำงาน หยุดทำงาน
สวิตช์ (ทั่วไป)					
สวิตช์ (มีกลไก)	*		*		คอยล์
รีเลย์					คอยล์
คอนแทกเตอร์					คอยล์
ไทมอร์ (on - delay)					คอยล์
ไทเมอร์ (off - delay)					คอยล์
อินพุตเอาท์พุตรีเลย์ ไทเมอร์ เคาน์เตอร์ รีเลย์ช่วย	*	*	*	*	หรือ
หน้าสัมผัสของเอาท์พุตรีเลย์ PC					หรือ

รูปที่ 2.4 สัญลักษณ์ในวงจรแลดเดอร์

2.5 ข้อดีของพีแอลซี



1. ราคาถูก

พีแอลซีจะมีราคาถูกกว่าวงจรรีเลย์ ในกรณีที่ใช้รีเลย์มากกว่า 10 ตัวในตู้ควบคุม

2. ออกแบบได้ง่าย

ออกแบบตู้ควบคุมทำได้ง่าย ออกแบบวงจรควบคุมทำได้ง่าย การติดตั้ง และ ทดสอบระบบก็สามารถทำได้ง่ายลงมาก

3. ทำได้เร็วขึ้น

การประกอบตู้ควบคุมทำได้เร็วขึ้นมากเนื่องจากใช้อุปกรณ์น้อยลง การเดินสายน้อยลง วงจรซีเคว้นซ์ใช้การโปรแกรม สามารถเปลี่ยนแปลงแก้ไขได้รวดเร็วโดยไม่ต้องเดินสายใหม่ ตู้ควบคุมกับเครื่องจักรสามารถผลิตแยกกันได้

4. มีขนาดเล็กและเป็นมาตรฐาน

มีขนาดเล็กลงมากเมื่อเทียบกับตู้ควบคุมวงจรรีเลย์ ขนาดของเครื่องเป็นมาตรฐานทำให้การทำตู้ควบคุมสะดวกขึ้น โปรแกรมที่ทำขึ้นโหลดลงในพีแอลซีได้รวดเร็ว และง่ายทำให้สามารถผลิตตู้ควบคุมที่เหมือนกันได้เร็วมาก

5. ความเชื่อถือได้สูง

ไม่มีปัญหาเรื่องรีเลย์ ไทเมอร์ เสียอีกแล้ว ทำโปรแกรมครั้งเดียวใช้ได้ตลอดกาล

6. บำรุงรักษางาน

ชิ้นส่วนน้อยทำให้บำรุงรักษาง่าย และในพีแอลซียังมีความสามารถในการตรวจเช็คตัวเองทำให้การตรวจซ่อมทำได้ง่ายมาก

อุปกรณ์ควบคุม หัวข้อ	รีเลย์	พีแอลซี
ฟังก์ชัน	ใช้ในการควบคุมซับซ้อนได้ ถ้าใช้รีเลย์จำนวนมาก	การควบคุมซับซ้อนเพียงใดก็สามารถโปรแกรมได้
การเปลี่ยนแปลงการควบคุม	เปลี่ยนได้โดยการเดินสายใหม่	เปลี่ยนได้โดยการเปลี่ยนโปรแกรม
ความเชื่อถือได้	ปรกติเชื่อถือได้ แต่มีปัญหาเรื่องจุดต่อสายหลวม และอายุการใช้งานของรีเลย์	องค์ประกอบหลักคือ สารกึ่งตัวนำจึงไม่มีปัญหาเรื่องจุดต่อสายหลวม มีความเชื่อถือได้
ใช้งานได้เอนกประสงค์	ใช้ได้กับงานที่ออกแบบมาเฉพาะเท่านั้น	ใช้งานได้เอนกประสงค์ โดยการโปรแกรม
การขยายระบบ	ทำได้ยากต้องเพิ่มอุปกรณ์หรือเปลี่ยนแปลงแก้ไขใหม่	ขยายได้เรื่อย ๆ จนเต็มขีดความสามารถ
บำรุงรักษาง่าย	ต้องตรวจเช็คบ่อย ๆ และต้องเปลี่ยนอุปกรณ์มีอายุจำกัด	ซ่อมโดยการเปลี่ยนส่วนประกอบ
การเข้าใจผิด	มีใช้กันแพร่หลาย คนส่วนมากเข้าใจเทคนิคการใช้รีเลย์	ยังไม่แพร่หลาย คนส่วนใหญ่ยังไม่เข้าใจเทคนิคการใช้งาน
ขนาด	ใหญ่	เล็ก และไม่ใหญ่ตามความซับซ้อนของการควบคุม
เวลาในการออกแบบ	ต้องเขียนแบบจำนวนมาก และต้องใช้เวลาในการประกอบ และการทดสอบ	ออกแบบได้ง่ายแม้จะเป็นควบคุมที่ซับซ้อน การประกอบวงจรควบคุมทำได้ง่ายและเร็ว
จุดคุ้ม	รีเลย์น้อยกว่า 10 ตัว	รีเลย์ 10 ตัวขึ้นไป

ตาราง 2.1 เปรียบเทียบวงจรรีเลย์กับพีแอลซี

บทที่ 3
ภาษาของพีแอลซี

ภายในพีแอลซีจะเป็นที่รวมของ รีเลย์ ไทเมอร์ เคาน์เตอร์ และอุปกรณ์อื่น ๆ อีกหลาย ๆ ตัว การต่อวงจรของอุปกรณ์เหล่านี้ทำได้โดยการป้อนโปรแกรมผ่านทางแผงป้อนโปรแกรม การต่อวงจรของอุปกรณ์ต่าง ๆ เหล่านี้ นอกจากการต่อหน้าสัมผัส และคอยล์แล้ว ยังมีกฎเกณฑ์ต่าง ๆ ขึ้นอยู่กับชนิดของอุปกรณ์ กฎเกณฑ์เหล่านี้ก็คือ คำสั่งนั่นเอง

คำสั่งประกอบด้วยรหัสคำสั่ง และเบอร์อุปกรณ์ บางคำสั่งอาจไม่ต้องมีเบอร์อุปกรณ์ก็ได้ เราจำเป็นต้องรู้เกี่ยวกับเบอร์ของอุปกรณ์ต่าง ๆ ที่เราสามารถใช้ได้

ในบทภาษาของพีแอลซีจะกล่าวถึงคำสั่งบูลลีน และคำสั่งแลดเดอร์ ที่ใช้งานจริงสำหรับ PLC ที่ได้จัดทำขึ้นนี้ พร้อมทั้งขอบเขตของอุปกรณ์ในการงาน และ การโปรแกรมลงในโปรแกรมที่ได้สร้างขึ้น

3.1 อุปกรณ์และเบอร์อุปกรณ์ ในพีแอลซีที่ทำจะมีอุปกรณ์ต่าง ๆ ดังนี้

X : อินพุทออปโต้คอปเปลอร์ (Input Optocoupled)

เป็นอุปกรณ์ออปโต้สำหรับใช้รับสัญญาณจากสวิตซ์ตรวจจับ ซึ่งต่อเข้ามาที่ขั้วเข้าของ PLC เบอร์ของอินพุทออปโต้คอปเปลอร์นี้จะนำหน้าด้วย X เสมอ

Y : เอาท์พุทรีเลย์ (Output Relay)

เป็นรีเลย์ขาออกที่ใช้ในการขับโหลดที่ต่อกับขั้วออกของพีแอลซีเอาท์พุตรีเลย์จะมีเบอร์นำหน้าด้วย Y เสมอ

M : รีเลย์ช่วย (Auxiliary Relay)

เป็นรีเลย์ที่ใช้ในการสร้างวงจรควบคุมซีเคว้นซ์ ภายในพีแอลซีจะจัดเตรียมรีเลย์ช่วยไว้มากมายเบอร์ของรีเลย์ช่วยนี้จะนำหน้าด้วย M เสมอ

T : ไทเมอร์ (Timer)

เป็นรีเลย์ที่มีการหน่วงเวลาในการปิดเปิดหน้าสัมผัส ภายในพีแอลซีจะเตรียมไทเมอร์ไว้ให้ใช้หลายตัว เบอร์ของไทเมอร์จะนำหน้าด้วย T เสมอ

C : เคาน์เตอร์ (Counter)

หรือตัวนับ ใช้นับจำนวนการเปิดปิดของหน้าสัมผัสและให้เอาท์พุตเมื่อครบจำนวนที่ตั้งไว้ เบอร์ของเคาน์เตอร์จะนำหน้าด้วย C เสมอ

อุปกรณ์	เบอร์	จำนวน
X	0 – 7	8
Y	0 – 7	8
M	0 – 31	32
C	0 - 15 (สามารถนับตั้งแต่ 1 ถึง 255)	16
T	0 - 7 (แบบปกติ)	8
	10 - 17 (แบบสะสมค่า)	8
	* K1 - 32767 หรือ 0.1 - 3276.7Sec	

ตารางที่ 3.1 อุปกรณ์หลักของเครื่องพีแอลซี

อุปกรณ์ / เบอร์	การตั้งค่า
X0 - X7 และ Y0 - Y7	F000H (บอร์ดหลัก)
X8 - X15 และ Y8 - Y15	F001H
X16 - X23 และ Y16 - Y23	F002H
X24 - X31 และ Y24 - Y31	F003H
X32 - X39 และ Y32 - Y39	F004H
X40 - X47 และ Y40 - Y47	F005H
X48 - X55 และ Y48 - Y55	F006H
X56 - X63 และ Y56 - Y63	F007H

ตารางที่ 3.2 การตั้งค่าบนบอร์ดอินพุต/เอาท์พุตหลัก และ บอร์ดขยาย

3.2 ชุดคำสั่ง

1. ภาษาแลดเดอร์ : 

   ภาษาบูลลีน : LD (Load)

   ความหมาย : เป็นการนำค่าสภาวะที่กำหนดเข้ามาสู่โปรแกรม

   วิธีการใช้งาน : ต้องกำหนดหมายเลขหรือตำแหน่งให้กับอุปกรณ์

2. ภาษาแลดเดอร์ : 

   ภาษาบูลลีน : LDI (Load Inverse)

   ความหมาย : เป็นการนำค่าสภาวะที่กำหนดเข้ามาสู่โปรแกรม ในลักษณะตรงข้าม

   วิธีการใช้งาน : ต้องกำหนดหมายเลขหรือตำแหน่งให้กับอุปกรณ์

3. ภาษาแลดเดอร์ : 

   ภาษาบูลลีน : AND

   ความหมาย : เป็นการนำค่าสภาวะที่กำหนดเข้ามาทำลอจิก AND กัน

   วิธีการใช้งาน : ใช้เมื่อเงื่อนไขที่ต้องการเกิดในลักษณะของการอนุกรมตั้งแต่สอง ขึ้นไป

4. ภาษาแลดเดอร์ : 

   ภาษาบูลลีน : ANI (AND Inverse)

   ความหมาย : เป็นการนำค่าสภาวะที่กำหนดเข้ามาทำลอจิก AND Inverse กัน

   วิธีการใช้งาน : ใช้เมื่อเงื่อนไขที่ต้องการเกิดในลักษณะของการอนุกรมตั้งแต่สอง ขึ้นไป

5. ภาษาแลดเดอร์ : 

   ภาษาบูลลีน : OR

   ความหมาย : เป็นการนำค่าสภาวะที่กำหนดเข้ามาทำลอจิก OR กัน

   วิธีการใช้งาน : ใช้เมื่อเงื่อนไขที่ต้องการเกิดในลักษณะของการอนุกรมตั้งแต่สอง ขึ้นไป

6. ภาษาแลดเดอร์ : 

   ภาษาบูลลีน : ORI (OR Inverse)

   ความหมาย : เป็นการนำค่าสภาวะที่กำหนดเข้ามาทำลอจิก OR Inverse กัน

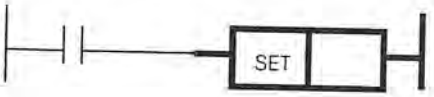
   วิธีการใช้งาน : ใช้เมื่อเงื่อนไขที่ต้องการเกิดในลักษณะของการอนุกรมตั้งแต่สอง ขึ้นไป

7. ภาษาแลดเดอร์ : 

ภาษาบูลลีน : OUT

ความหมาย : ใช้เพื่อควบคุมสถานะของอุปกรณ์ปลายทาง ให้มีการทำงานตามเงื่อนไขข้างหน้า

วิธีการใช้งาน : ใช้เมื่อต้องการนำ ค่าสภาวะ ออกมาทางอุปกรณ์ปลายทางต่าง ๆ

8. ภาษาแลดเดอร์ : 

ภาษาบูลลีน : SET

ความหมาย : ใช้เพื่อควบคุมสถานะของอุปกรณ์ปลายทาง ให้มีการทำงาน ON ค้าง

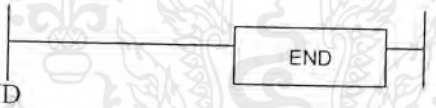
วิธีการใช้งาน : ใช้เมื่อต้องการนำ ค่าสภาวะ ON ออกมาทางอุปกรณ์ปลาย (ใช้ได้กับรีเลย์ช่วย M และ เอาท์พุตรีเลย์ Y เท่านั้น)

9. ภาษาแลดเดอร์ : 

ภาษาบูลลีน : RST (Reset)

ความหมาย : ใช้เพื่อควบคุมสถานะของอุปกรณ์ปลายทาง ให้มีการทำงาน OFF

วิธีการใช้งาน : ใช้เมื่อต้องการนำ ค่าสภาวะ OFF ออกมาทางอุปกรณ์ปลาย (ใช้ได้กับรีเลย์ช่วย M ,เคาน์เตอร์ C และ เอาท์พุตรีเลย์ Y เท่านั้น)

10. ภาษาแลดเดอร์ : 

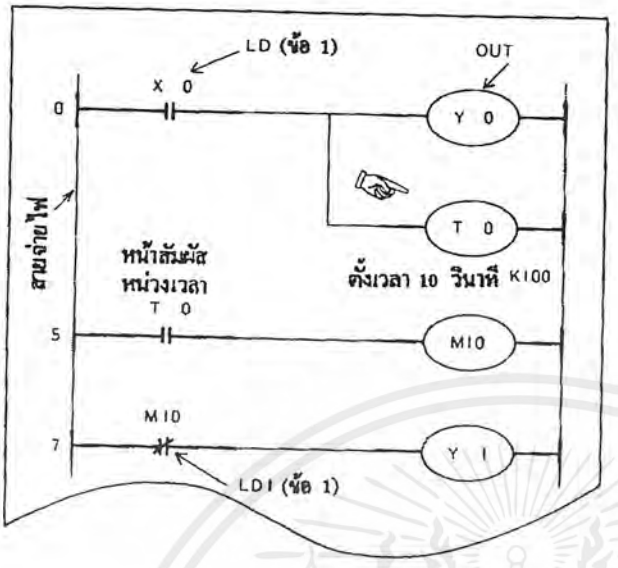
ภาษาบูลลีน : END

ความหมาย : คำสั่งบอกจุดจบของโปรแกรม

วิธีการใช้งาน : พีแอลซี ทั่วไปจะทำงาน 3 จังหวะใหญ่ คือ อ่านอินพุต ทำงานตามโปรแกรม แล้วจึงส่งเอาท์พุตออกไป ถ้าเราเขียนคำสั่ง END ลงในโปรแกรม พีแอลซี จะหยุดทำงานตามโปรแกรม ณ จุดนั้น แล้วจัดส่งเอาท์พุตออกไป โดยไม่ไปสนใจโปรแกรมที่อยู่ลำดับถัดมาเลย

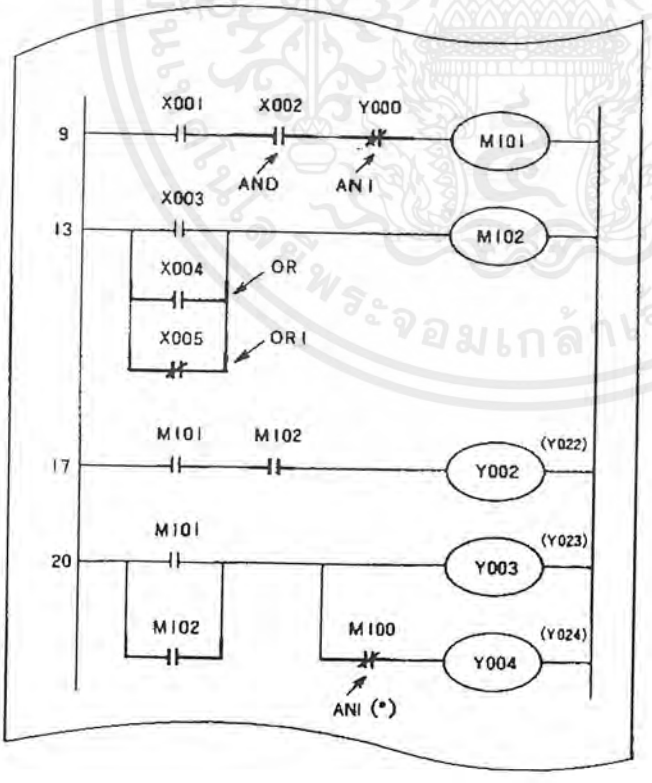
เราสามารถนำคำสั่ง END มาใช้ในการทดสอบโปรแกรมได้ โดยแทรกคำสั่ง END ไว้ตรงท้ายโปรแกรมแต่ละช่วง และลองทดสอบเฉพาะโปรแกรมท่อนนั้น เมื่อทดสอบว่าโปรแกรมทำงานถูกต้องก็ลบคำสั่ง END ออกและไปทดสอบโปรแกรมท่อนถัดไป ทำเช่นนี้เรื่อย ๆ จนสามารถทดสอบทั้งโปรแกรม

3.2.1 ตัวอย่างการใช้คำสั่ง



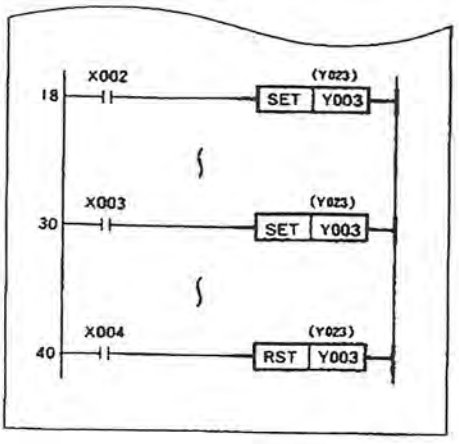
0	LD	X 0
1	OUT	Y 0
2	OUT	T 0
	SP	K100
5	LD	T 0
6	OUT	M10
7	LDI	M10
8	OUT	Y 1

รูปที่ 3.1 คำสั่ง LD, LDI และ OUT



9	LD	X001
10	AND	X002
11	ANI	Y000(20)
12	OUT	M101
13	LD	X003
14	OR	X004
15	ORI	X005
16	OUT	M102
17	LD	M101
18	AND	M102
19	OUT	Y002(22)
20	LD	M101
21	OR	M102
22	OUT	Y003(23)
23	ANI	M100
24	OUT	Y004(24)

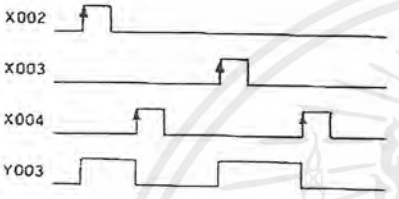
รูปที่ 3.2 คำสั่ง AND, ANI, OR และ ORI



18	LD	X 002
19	SET	Y003(23)
	∫	
30	LD	X 003
31	SET	Y003(23)
	∫	
40	LD	X 004
41	RST	Y003(23)
43	END	

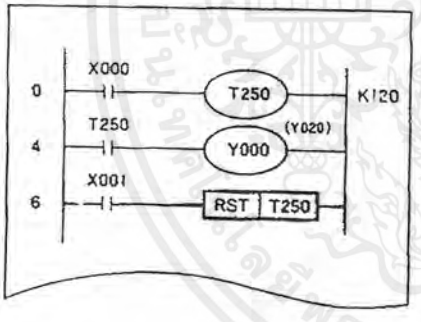
คำสั่ง SET และ RST สามารถใช้กับคอยล์ของรีเลย์เบอร์เดียวกันซ้ำ ๆ หลายครั้งได้

<การทำงาน>



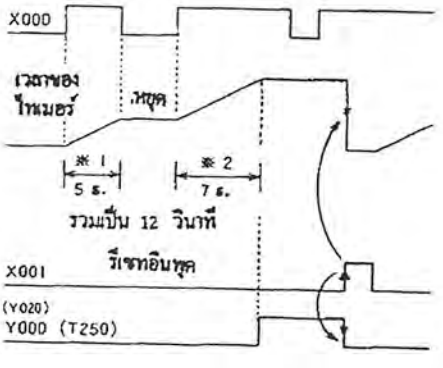
- เมื่อ X002 ON จะทำให้ Y003 ทำงาน และทำงานค้าง แม้ X002 จะ OFF ก็ตาม
- เมื่อ X004 ON จะทำให้ Y003 ทำงาน และทำงานค้าง แม้ X004 จะ OFF ก็ตาม
- X003 ก็จะทำงานเหมือนหน้าสัมผัสทั้งสองด้วย

รูปที่ 3.3 คำสั่ง SET และ RST



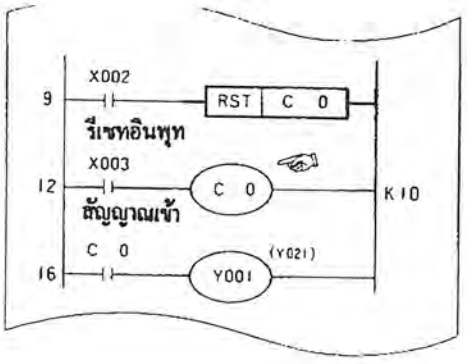
0	LD	X 000
1	OUT	T 250
	SP	K 120
4	LD	T 250
5	OUT	Y 000 (20)
6	LD	X 001
7	RST	T250

< การทำงาน >



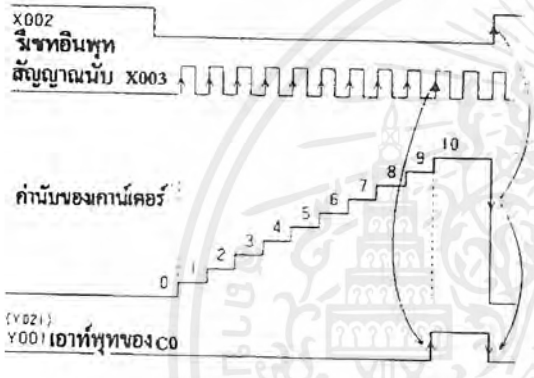
- ขณะที่ X000 ON ไทเมอร์ T250 จะ นับเวลาไปเรื่อย ๆ เมื่อ X000 OFF ไทเมอร์จะหยุดนับเวลา แต่จะเก็บค่าเวลานั้นไว้
- ถ้า X000 ON อีกไทเมอร์ก็จะนับต่อไปอีก เช่นนี้จนครบตามค่าเวลาที่ตั้งไว้ หน้าสัมผัสของไทเมอร์จะ ON
- ถ้ามีสัญญาณรีเซท X001 ON จะทำให้ ไทเมอร์ถูกรีเซท ค่าเวลาในไทเมอร์จะเป็น 0 ใหม่

รูปที่ 3.4 คำสั่ง RST ของไทเมอร์ชนิดสะสมเวลา



9	LD	X 002
10	RST	C 0
12	LD	X 003
13	OUT	C 0
	SP	K 10
16	LD	C 0
17	OUT	Y 001 (21)

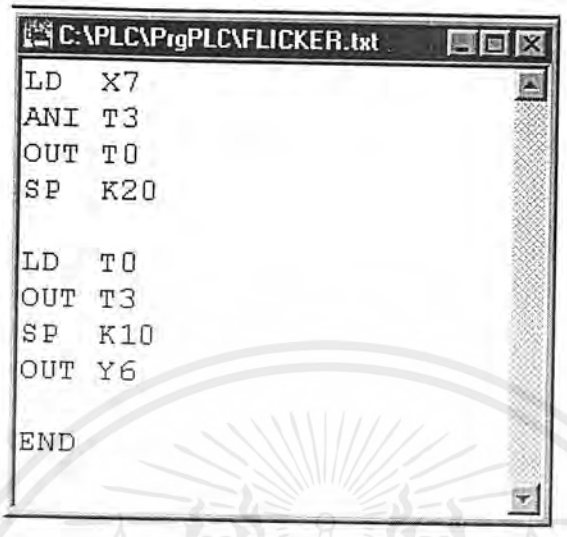
<การทำงาน>



- เคาน์เตอร์จะนับเมื่อสัญญาณเข้าที่ใช้นับ X003 เปลี่ยนจาก OFF เป็น ON ค่า นับจะเพิ่มขึ้นเรื่อย ๆ จนถึงค่าที่ตั้งไว้ K ตั้งได้ตั้งแต่ 1-255 หน้าสัมผัสของเคาน์เตอร์ก็จะ ON
- เมื่อนับครบจำนวนที่ตั้งไว้ค่านับจะไม่เปลี่ยนแปลง และหน้าสัมผัสของเคาน์เตอร์จะ ON ค้างแม้จะมีสัญญาณนับเข้ามาอีกก็ตาม
- ถ้ารีเซทอินพุท X002 ON จะ ทำ การ รีเซต เคาน์เตอร์ C0 ค่านับจะกลายเป็น 0 และหน้าสัมผัสของเคาน์เตอร์ก็จะ OFF

รูปที่ 3.5 คำสั่ง RST ของเคาน์เตอร์

3.3 คำสั่งและการโปรแกรม



รูปที่ 3.6 การเขียนโปรแกรมลงในซอฟต์แวร์

การโปรแกรมภาษาบูลลีนลงในซอฟต์แวร์ (Software) โดยมีรายละเอียดดังต่อไปนี้

1. โปรแกรมเกิดจากการนำคำสั่งต่าง ๆ มาเขียนเรียงกัน การเขียนเรียงลำดับของคำสั่ง ก็คือหมายเลขของบรรทัด ซึ่งการเขียนโปรแกรมบูลลีนไม่จำเป็นต้องใส่ค่า

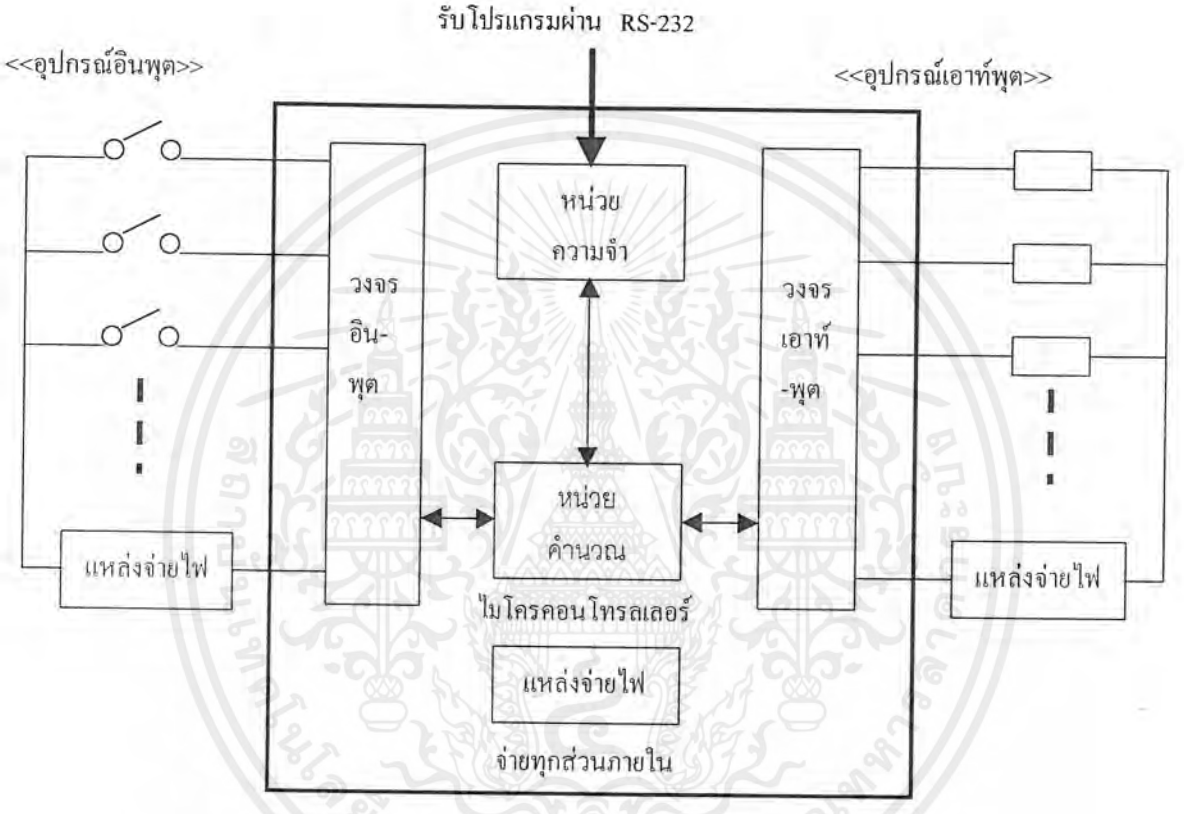
2. คำสั่งจะประกอบด้วย รหัสคำสั่ง (opcode) และเบอร์อุปกรณ์ (operand) บางคำสั่งอาจจะไม่ต้องมีเบอร์อุปกรณ์ก็มี รหัสคำสั่งอาจเรียกย่อว่าคำสั่งก็ได้

ในกรณีของ เคาน์เตอร์ และ ไทเมอร์ จะมีคำสั่งเพิ่มอีกบรรทัดเป็นการระบุค่าขอบเขตการทำงานของอุปกรณ์นั้น ๆ โดย opcode คือ SP และ operand ขึ้นต้นด้วย K และตามด้วยค่าที่ต้องการ

3. PLC จะอ่านคำสั่งในโปรแกรมเริ่มจากลำดับ 0 จนถึง ลำดับที่เป็นคำสั่ง END อ่านคำสั่งและทำงานตามคำสั่งนั้น เมื่อถึงคำสั่ง END ก็จะเริ่มมาอ่านคำสั่งที่ลำดับ 0 ใหม่ วนเป็นสงรอบเช่นนี้อยู่ตลอดเวลา การทำงานหนึ่งวงรอบเรียกว่า เวลาสแกน (scan time) บางครั้งเรียกว่า เวลาหนึ่งไซเกิล (cycle time) เวลานี้ไม่ควรเกิน 10 msec.

บทที่ 4
ฮาร์ดแวร์ของโครงงาน

ภายในพีแอลซีที่สร้างขึ้นมา จะมีโครงสร้างดังต่อไปนี้ ซึ่งฮาร์ดแวร์ที่ทำขึ้นคือในกรอบสี่เหลี่ยมเข้ม



หมายเหตุ - อุปกรณ์อินพุต ในโครงงานนี้จะใช้สวิตช์เพื่อสะดวกในการทดสอบการเขียนคำสั่ง

- อุปกรณ์เอาท์พุต เป็นอุปกรณ์ที่ต่อเสริมเองตามต้องการ โดยในวงจรเอาท์พุตจะเป็นหน้าสัมผัสของรีเลย์ รายละเอียดในบทที่ 2

รูปที่ 4.1 โครงสร้างของพีแอลซีสร้างขึ้น

ส่วนของฮาร์ดแวร์ประกอบด้วย 3 บอร์ด สำคัญ

4.1 บอร์ดหลัก ตามวงจรรูปที่ 4.3

เป็นส่วนของบอร์ดของไมโครคอนโทรลเลอร์ ซึ่งบอร์ดนี้ได้ใช้ไมโครคอนโทรลเลอร์ตระกูล เอ็มซีเอส-51 เบอร์ AT89S8252 ของบริษัท Atmel ที่มี Flash Memory 8Kbytes , EEPROM 2Kbytes และมี RAM 256 Bytes

เหตุผลของการใช้ ไมโครคอนโทรลเลอร์ เบอร์นี้คือ ในแนวทางการพัฒนาโปรแกรมนั้นถือว่าโปรแกรมเป็นหัวใจสำคัญที่สุดสำหรับงานควบคุม และการพัฒนาโปรแกรมควบคุม PLC ที่มีตำแหน่งโปรแกรมอยู่บนหน่วยความจำแบบ Flash Memory 8Kbytes เช่นกัน โดยทั่วไปแล้วการพัฒนาโปรแกรมบนชิพเดี่ยวจะมีอยู่ด้วยกันหลายวิธี แต่ที่คุ้นเคยกันส่วนใหญ่จะเป็นวิธีการพัฒนาผ่านเครื่องโปรแกรมทำให้ต้นทุนในการพัฒนาโปรแกรมสูง และมีขั้นตอนที่ยุ่งยากซับซ้อน

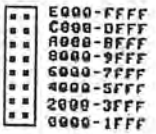
เครื่อง PLC ที่สร้างขึ้นจึงอาศัยคุณสมบัติของชิพเบอร์ AT89S8252 ที่มี PORT SPI ในตัวเพียงแต่การสร้างบอร์ดก็ทำขั้วต่อลักษณะ 10 PIN ให้ตรงกับเครื่องโปรแกรม การพัฒนาโปรแกรมสามารถทำได้ในรูปแบบที่ไม่ต้องถอดชิพมาทำการโปรแกรม หรือที่เรียกว่า "IN SYSTEM PROGRAMMING" คือทำการโหลดข้อมูลจากเครื่องคอมพิวเตอร์ลงสู่ตัวชิพโดยตรง ทำให้การพัฒนาง่ายขึ้นและสะดวกยิ่งขึ้น โดยเครื่องโปรแกรมที่ใช้ในการพัฒนาคือ SPI-LOAD ของบริษัท ศิลารีเสิร์ช จำกัด

หน่วยความจำ ที่ใช้ในการเก็บตัวโปรแกรม ขนาด 32 Kbytes ที่มีให้เลือก 2 แบบคือ

- โวลาไทล์(Volatile) หรือ แรมธรรมดา เบอร์ 62256
  ข้อดี คือ ราคาถูก ส่วนข้อเสีย คือ ข้อมูลจะสูญหายทั้งหมดเมื่อไม่แหล่งจ่ายไฟ
- นอนโวลาไทล์ (Nonvolatile) หรือ NVRAM เบอร์ DS1230Y/AB
  ข้อดี คือ ข้อมูลจะไม่สูญหายเมื่อไม่แหล่งจ่ายไฟและอายุการใช้งานยาวนาน
  ส่วนข้อเสีย คือ ราคาแพง

ไอซีเบอร์ 74LS373 เป็นไอซีพักข้อมูล(Latch) เนื่องจากระบบบัสแอดเดรสและบัสข้อมูลของไมโครคอนโทรลเลอร์ เป็นลักษณะแบบใช้การมัลติเพล็กซ์จากพอร์ตเดียวกัน กล่าวคือ ในระยะเวลาเริ่มต้น เส้นสัญญาณเหล่านี้พอร์ตจะใช้ในการส่งค่าแอดเดรสของตำแหน่งที่ต้องการติดต่อด้วย ในช่วงเวลาต่อมาจึงเปลี่ยนไปเป็นสภาวะอิมพีแดนซ์สูงเพื่อใช้งานในฐานะของบัสข้อมูล แต่เนื่องจากหน่วยความจำที่ใช้กันทั่วไปนั้นไม่ใช้การมัลติเพล็กซ์ และมีขาสัญญาณบัสแอดเดรสและบัสข้อมูลแยกจากกันโดยชัดเจนดังนั้นการเชื่อมต่อหน่วยความจำ จึงจำเป็นต้องมีวงจรประเภทแลตซ์ (Latch) ประกอบเพิ่มเติมขึ้น เพื่อค้างค่าของแอดเดรสที่ส่งจากไมโครคอนโทรลเลอร์ ในช่วงเวลาแรกให้กับขาสัญญาณแอดเดรสของหน่วยความจำต่อไป

ไอซีเบอร์ 74LS156 เป็นไอซีถอดรหัสเพื่อเลือกตำแหน่งเริ่มต้นและขนาดของหน่วยความจำแต่ละจุดของ Jumper ที่ใส่ลงไปจะมีขนาดของ หน่วยความจำ 8K เพิ่มจากจุดเริ่มต้น



รูปที่ 4.2 ตำแหน่งการเลือกค่าเริ่มต้นและขนาดของหน่วยความจำ

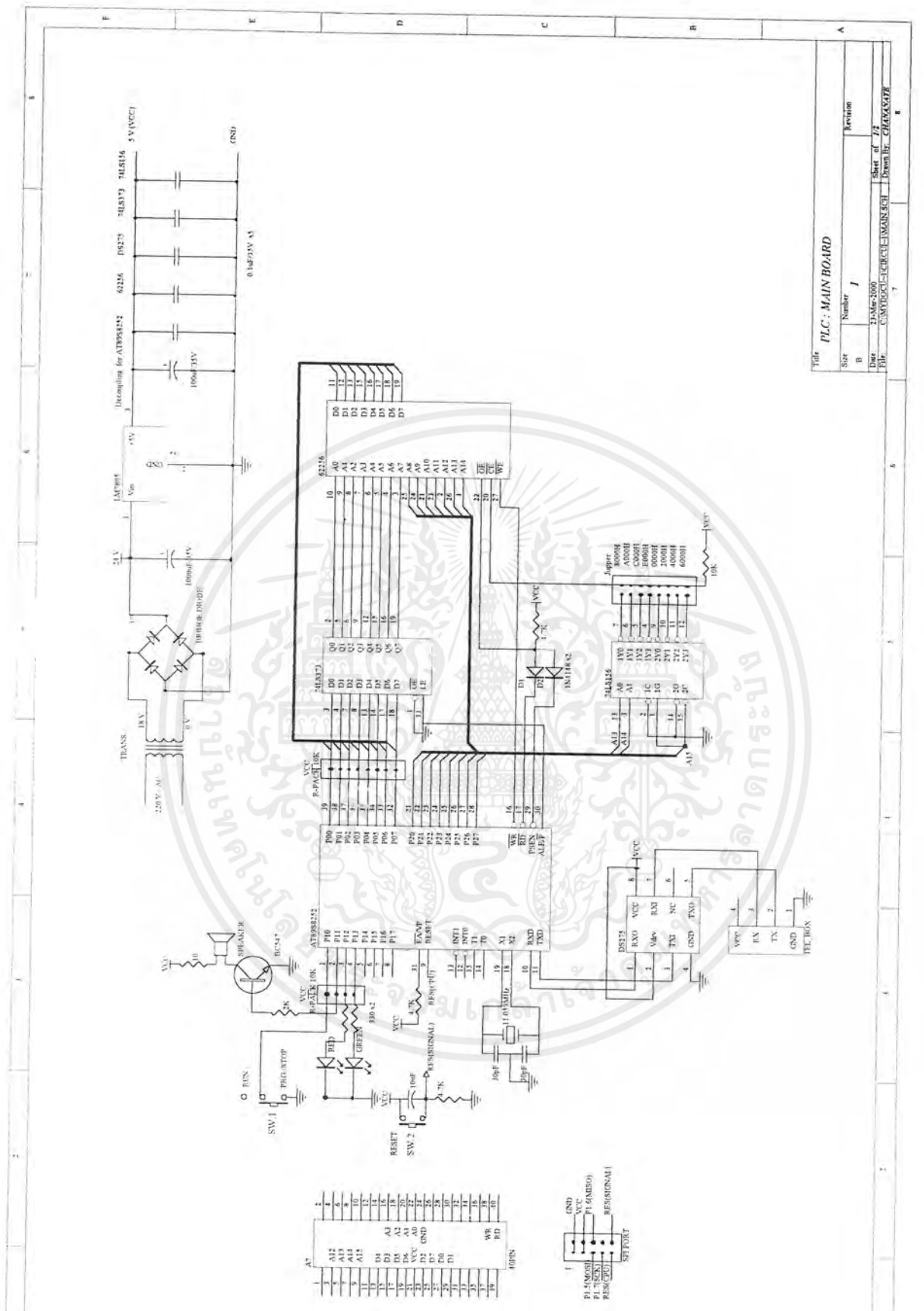
ไดโอด D1 , D2 และ ความต้านทาน 4.7KΩ เป็นการต่อวงของ AND Gate แบบ 2 อินพุต คือ RD และ PSEN ของไมโครคอนโทรลเลอร์ ส่งเอาท์พุตไปที่ OE ของหน่วยความจำ

ไอซีเบอร์ DS275 เป็นตัวเปลี่ยนระดับแรงดันของการสื่อสารแบบ RS-232 เป็นแรงดันแบบระดับ Digital Logic ที่ "0" คือ 0 โวลท์ และ "1" คือ +5 โวลท์ มีความสามารถแบบเดียวกับไอซี MAX232CPE แต่รับช่องการสื่อสารได้หนึ่งคู่คือ การส่งและการับอย่างละชุด และการใช้งานง่ายกว่าที่ไม่ต้องเพิ่มอุปกรณ์เสริมก็สามารถใช้งานได้

อุปกรณ์ต่อ Tel Box เป็นขั้วต่อแบบเดียวกับโทรศัทพ์ เพื่อความสะดวกแก่การต่อเชื่อม

บอร์ดได้ออกแบบวงจรเสียง เพื่อตอบรับการโปรแกรม และสวิตช์แบบเลื่อน SW1 ทำการออกแบบไว้เพื่อทำงานสัมพันธ์กับ LED สีเขียวและสีแดง ดังนั้นการติดตั้งบนบอร์ดจริงมีอยู่ใกล้เคียงกัน ทั้งหมดเป็นการใช้งานในพอร์ตที่ 1 ของไมโครคอนโทรลเลอร์ จะขออธิบายรายละเอียดในส่วนของซอฟต์แวร์ ที่เป็นโปรแกรมมอนิเตอร์ใน Flash Memory ของชิพ

ส่วนพอร์ตแบบ 40 PIN เป็นส่วนในการติดต่อกับบอร์ดอื่น ๆ ในเครื่องพีแอลซีเหตุที่ใช้การวางตำแหน่งข้อสัญญาณดังนี้ เพื่อสามารถนำบอร์ดนี้ไปประยุกต์ใช้กับบอร์ดใช้งานอื่น ๆ ที่มีขายตามท้องตลาด ที่นอกเหนือจากการใช้งานในเรื่องของพีแอลซี



รูปที่ 4.3 บอร์ดหลัก

4.2 บอร์ดอินพุตและเอาต์พุตหลัก ตามวงจรรูปที่ 4.4

เป็นบอร์ดที่รับส่วนของวงจรอินพุตและวงจรเอาท์พุต โดยแยกหน้าที่การทำงานโดยวงจรถอดรหัส ซึ่งไอซีที่ทำหน้าที่นี้คือ 74LS138 #1 และ 74LS138 #2 โดยที่ไอซี 74LS138 #1 จะทำตั้งค่าของตำแหน่งการถอดรหัสของสองไบต์แรก และไอซี 74LS138 #2 จะตั้งค่าของตำแหน่งการถอดรหัสของสองไบต์หลัง รวมเป็น 4 ไบต์ หรือ 16 บิต โดยมีรายละเอียดการตั้งค่าตามที่เขียนไว้ในวงจรแล้ว ซึ่งตำแหน่งที่เลือกนั้นจะประกอบด้วย 8 อินพุต และ 8 เอาท์พุต

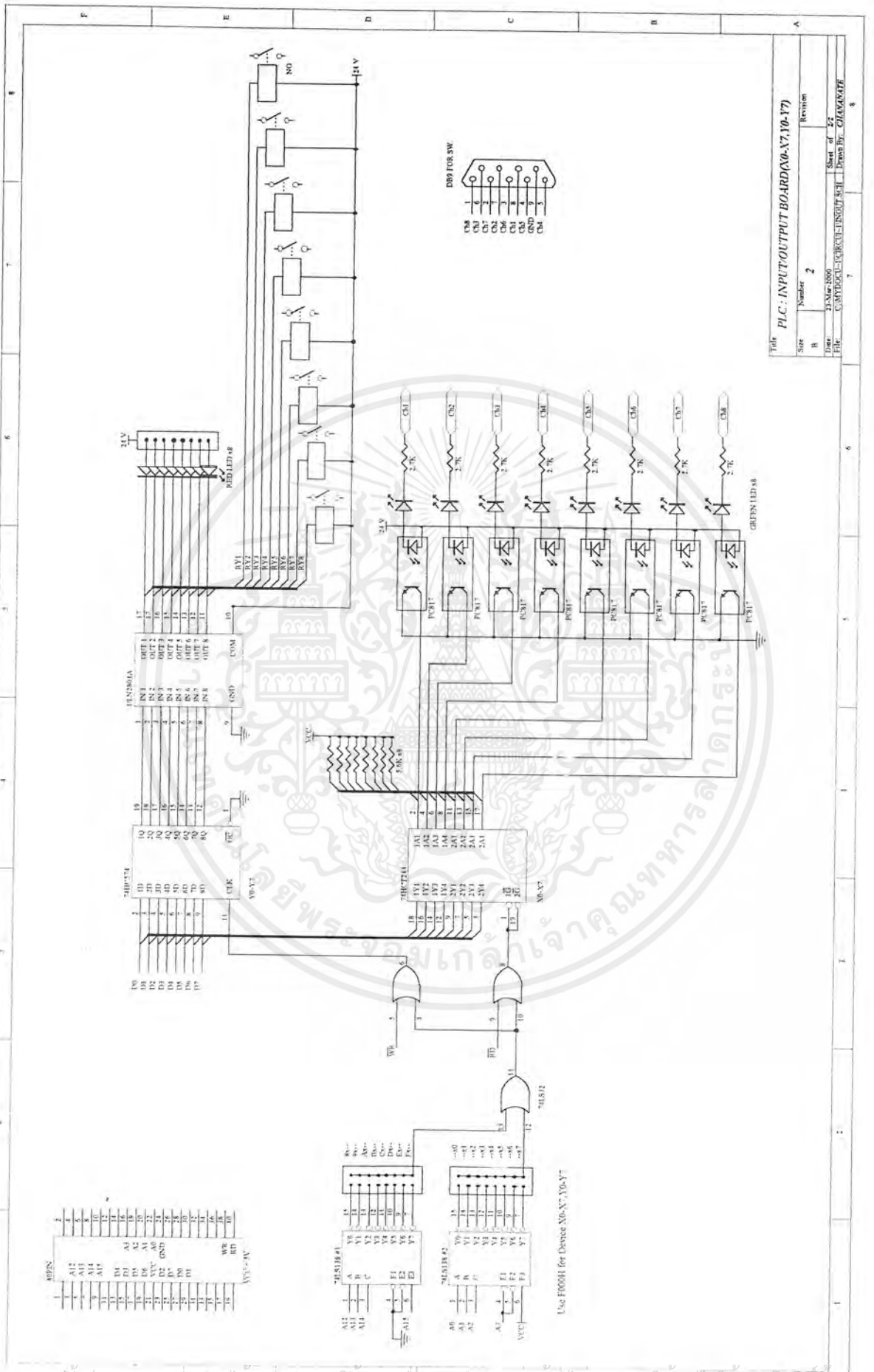
ส่วนการแยกว่าจะรับค่าสัญญาณอินพุต หรือ จะส่งค่าสัญญาณเอาท์พุต โดยการใช้สัญญาณของ WR(ขา16) และ RD(ขา17) ของไมโครคอนโทรลเลอร์เป็นตัวเลือกโดยตรง

ส่วนของวงจรอินพุต คือ วงจรออปโต้คอปเปลอร์ 8 วงจร ออปโต้คอปเปลอร์ เป็นอุปกรณ์ที่มีการแบ่งส่วนอินพุตและเอาต์พุตออกจากกันทางไฟฟ้าโดยสิ้นเชิงการถ่ายทอดสัญญาณระหว่างส่วนอินพุตและเอาต์พุตจะใช้การเชื่อมโยงทางแสงเท่านั้น ทำให้กราวด์ของอินพุตและเอาต์พุตไม่เชื่อมต่อกัน ในการใช้งานจริงก็จะใช้วงจรแหล่งจ่ายไฟแยกต่างหากกัน แต่ในการทำโครงงานนี้ได้มีการใช้แหล่งจ่ายไฟชุดเดียวกันเพื่อเป็นการประหยัดพื้นที่ของวงจร และลดการใช้แหล่งจ่ายไฟเพิ่มสะดวกแก่การเคลื่อนย้าย ในการทำการทำลองก็ง่ายขึ้นด้วย

ส่วนของวงจรเอาท์พุต คือ วงจรรีเลย์ 8 วงจร เป็นการขับรีเลย์โดยใช้ไอซีขับ เบอร์ ULN2803A ไอซีไดรเวอร์เบอร์นี้มีความสามารถสูงกว่า 74LS06 และ 74LS07 โดยภายในบรรจุอินเวอร์เตอร์เกตแบบคอลเล็กเตอร์เปิด 8 ตัว สามารถใช้กับแรงดันได้สูงสุด 50V และกระแสสูงสุด 500mA ซึ่งเบอร์นี้ออกแบบมาเพื่อรับอินพุต 5V แบบ TTL หรือ CMOS นอกจากนั้นภายในไอซีเองยังต่อไดโอดป้องกันไว้ทุกขาเอาต์พุต ทำให้สามารถต่อโหลดที่เป็นขดลวดได้ทันที อีกทั้งยังลดความยุ่งยากของการต่อวงจรขับที่เป็นแบบทรานซิสเตอร์ลงได้มาก

จากวงจรอินพุตและวงจรเอาท์พุต ของพีแอลซีได้ออกแบบอุปกรณ์เชื่อมต่อกับอุปกรณ์ภายนอกตัวพีแอลซีตามรูปที่ 4.1 การรับค่าเป็นออปโต้คอปเปลอร์ ที่เชื่อมโยงด้วยแสง และ การส่งค่ารีเลย์ที่ตัดต่อหน้าสัมผัส โดยอำนาจแม่เหล็กนั้น จะเป็นส่วนป้องกันการเสียหายแก่ตัวพีแอลซีได้เป็นอย่างดี อีกทั้งสามารถทำให้กราวด์ของอินพุตและเอาต์พุตไม่เชื่อมต่อกัน

วงจรทั้งอินพุตและเอาต์พุต มีการแสดงผลด้วย LED ร่วมด้วย และการป้อนอินพุตก็สามารถทำป้อนค่าด้วยชุดสวิทช์โย้ก 8 ตัว ผ่านพอร์ต DB9

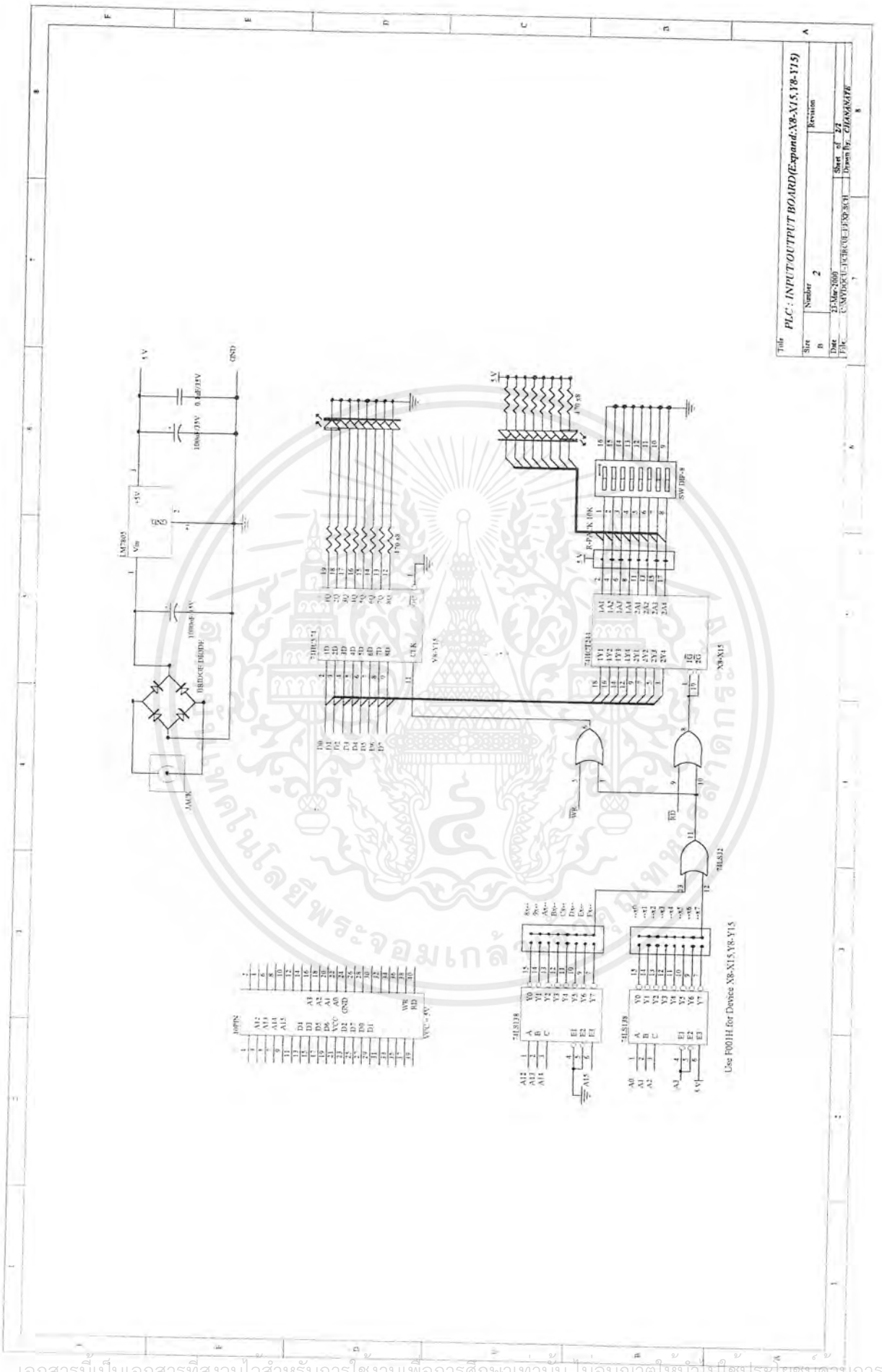


รูปที่ 4.4 บอร์ดอินพุตและเอาท์พุตหลัก

4.3 บอร์ดขยายอินพุต/เอาท์พุต ตามวงจรรูปที่ 4.5

บอร์ดขยายอินพุต/เอาท์พุต เป็นการออกแบบลักษณะเดียวกับบอร์ดอินพุตและเอาท์พุตหลัก เป็นบอร์ดสำหรับทำการทดสอบการเพิ่มจำนวนอินพุตหรือเอาท์พุต ซึ่งจะตัดชุดของออปโต้คอปเปลอร์ และ ชุดขับหน้าสัมผัสรีเลย์ ออก

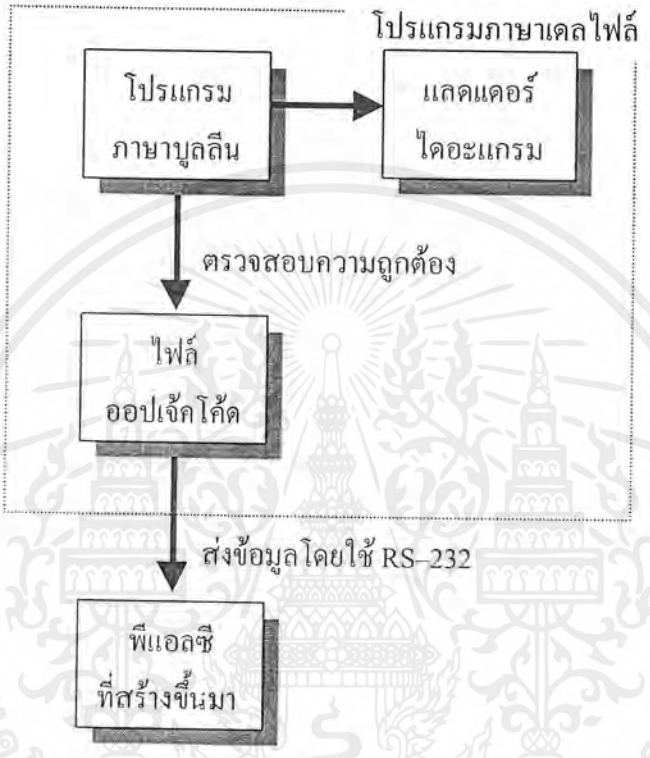
แต่บอร์ดก็สามารถแสดงผลของอินพุตและเอาท์พุตได้ครบสมบูรณ์ ทั้งยังสามารถป้อนทดสอบอินพุตได้ด้วย ดิฟสวิทช์ (Dip SW.)



รูปที่ 4.5 บอร์ดขยายอินพุต/เอาท์พุต

บทที่ 5
ซอฟต์แวร์ของโครงงาน

5.1 ขั้นตอนการทำงานของพีแอลซีที่ได้สร้างขึ้น



รูปที่ 5.1 แสดงขั้นตอนการทำงานของพีแอลซีที่ได้สร้างขึ้น

ส่วนขอซอฟแวร์ทั้งหมดบนคอมพิวเตอร์จะอยู่ภายใต้การควบคุมของโปรแกรมที่สร้างขึ้นหรือซอฟต์แวร์ ที่สร้างจากโปรแกรมทางคอมพิวเตอร์ภาษาเดลไพล์ (Delphi)

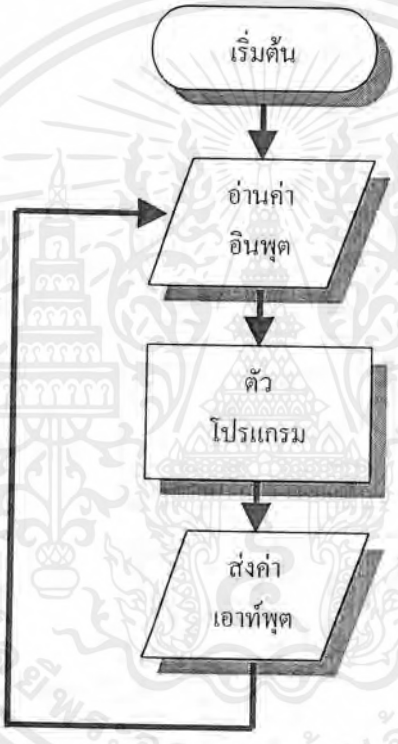
การเขียนโปรแกรมภาษาบูลลีนเป็นขั้นตอนการออกแบบที่จะเขียนลงบนโปรแกรมที่สร้างขึ้น (Editor) มานี้ ซึ่งจะมีพื้นที่ของการเขียนและเมื่อเขียนเสร็จ การจัดเก็บข้อมูลนี้ในรูปแบบของเท็กซ์ไฟล์ คือ เป็นจัดเก็บสกุล ".txt"

ในระหว่างทำการเขียนโปรแกรมของพีแอลซีด้วยภาษาบูลลีนนั้นสามารถทำการเรียกดูลักษณะการต่อวงจรในรูปแบบของ แลดเดอร์ไดอะแกรม ได้เพื่อง่ายแก่ต่อตรวจสอบหรือทำความเข้าใจ

เมื่อการเขียนโปรแกรมภาษาบูลลีนเสร็จไม่จำเป็นต้องเก็บข้อมูลก็ได้ สามารถทำการแปลง (Compile) ภาษาโดยโปรแกรมภาษาเดลไฟล์ที่เขียนขึ้นสามารถแปลจากบูลลีนเป็นไฟล์ออปเจ้คโค้ดได้

เมื่อเสร็จการแปลภาษาจากบูลลีนเป็นไฟล์ออปเจ้คโค้ด แล้วซอฟต์แวร์ก็ยังสามารถจัดส่งข้อมูลที่เป็นตัวโปรแกรมของพีแอลซีไปบนหน่วยความจำบนเครื่องพีแอลซีได้โดยใช้มาตราฐานการสื่อสารของ RS-232

5.2 กระบวนการของไมโครคอนโทรลเลอร์ที่ดำเนินการภาษาพีแอลซี



รูปที่ 5.2 กระบวนการของไมโครคอนโทรลเลอร์ ที่ดำเนินการภาษาพีแอลซี

กระบวนการในรูปที่ 5.2 เริ่มเมื่อสิ้นสุดกระบวนการในแผนผังรูปที่ 5.1 หรือก็คือเมื่อซอฟต์แวร์เสร็จสิ้นการส่งโปรแกรมลงบนพีแอลซีและ พีแอลซีตั้งอยู่ในโหมดของการทำงานแล้ว

กระบวนการของไมโครคอนโทรลเลอร์ ที่ดำเนินการภาษาพีแอลซีโดยการเริ่มต้นตั้งแต่อ่านค่าอินพุตจนถึงการส่งค่าเอาท์พุตและวนกลับมาที่จุดเริ่มต้นใหม่ การทำงานหนึ่งวงรอบเรียกว่า เวลาสแกน(scan time) บางครั้งเรียกว่า เวลาหนึ่งไซเกิล(cycle time)

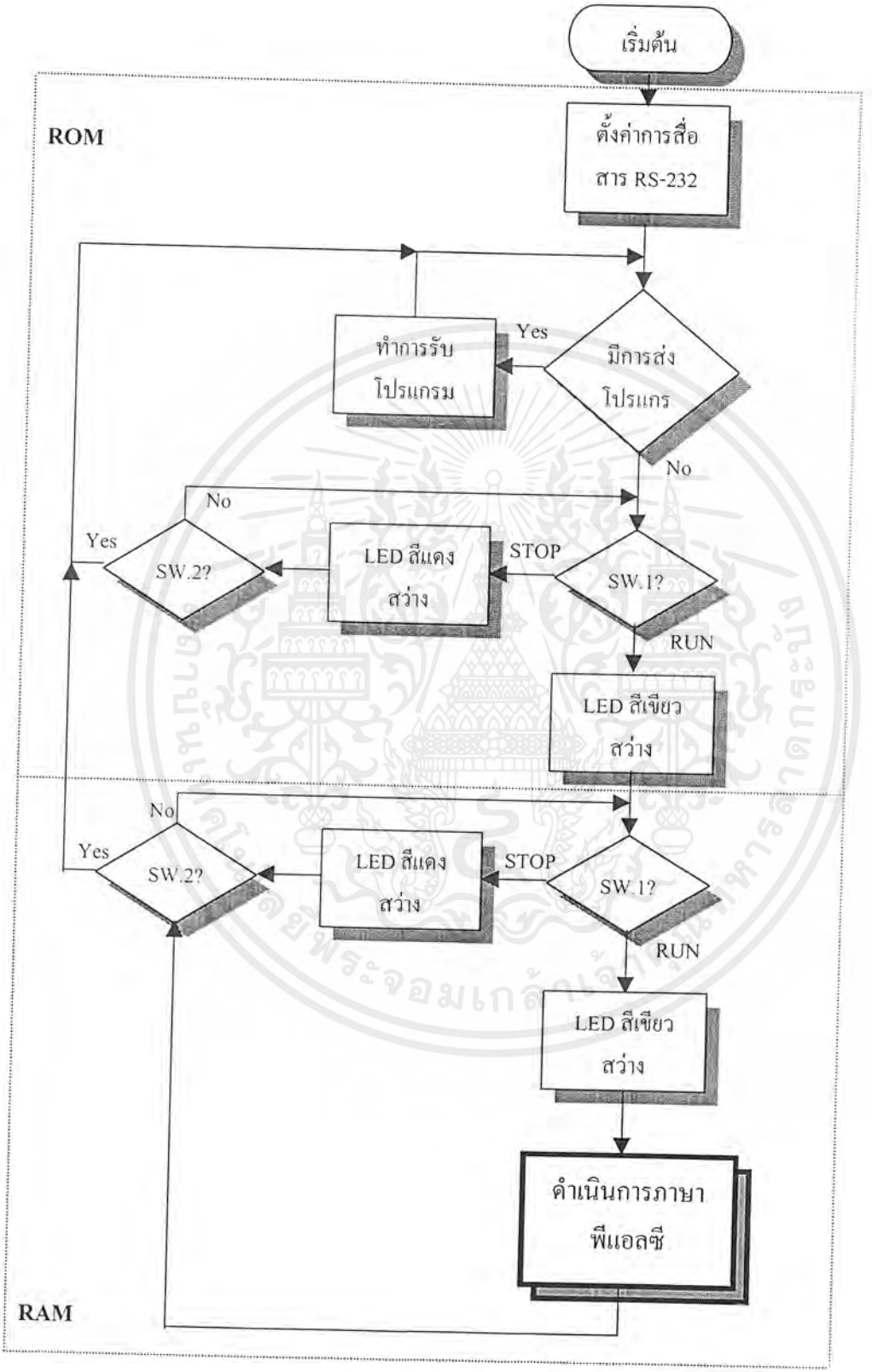
พีแอลซีทั่วไปจะทำงาน 3 จังหวะใหญ่ คือ อ่านอินพุต ทำงานตามโปรแกรม แล้วจึงส่งเอาท์พุตออกไป ถ้าเราเขียนคำสั่ง END ลงในโปรแกรมพีแอลซีจะหยุดทำงานตามโปรแกรม ณ จุดนั้น แล้วจัดส่งเอาท์พุทออกไป โดยไม่ไปสนใจโปรแกรมที่อยู่ลำดับถัดมาเลย

แผนผังข้อมูล อ่านค่าอินพุต จะเป็นการเก็บค่าจากพอร์ทอินพุตมาเก็บไว้ในหน่วยความจำแบบชั่วคราว (RAM) เพื่อให้คำสั่งของพีแอลซีในตัวโปรแกรมสามารถอ่านค่าได้สะดวก

แผนผังกระบวนการของ ตัวโปรแกรมพีแอลซีจะเริ่มอ่านคำสั่งในโปรแกรมเริ่มจากบรรทัดที่ 1 ของภาษาบูลลีนจนถึงบรรทัดที่เป็นคำสั่ง END ก็จะออกไปที่กระบวรการของการส่งค่าเอาท์พุท ซึ่ง ในแต่ละบรรทัดของคำสั่งจะมีการอ่านค่าอินพุทหรือส่งค่าเอาท์พุท จะใช้การพักไว้ในหน่วยความจำแบบชั่วคราว(RAM) ที่กำหนดไว้ตามที่แบ่งส่วนของความจำหรือ Map memory

แผนผังข้อมูล ส่งค่าเอ้าท์พุท คือเมื่อถึงขั้นตอนนี้ไม่ว่าจะมีการเปลี่ยนแปลงข้อมูลหรือไม่ ภายในหน่วยความจำแบบชั่วคราว(RAM) ตรงตำแหน่งที่เป็นที่พักข้อมูลออกเอาท์พุต ก็จะทำการดึงข้อมูลที่พักไว้นี้ ส่งออกพอร์ทเอาท์พุต

5.3 โปรแกรมมอนิเตอร์ที่เขียนลงบนหน่วยความจำหลักของไมโครคอนโทรลเลอร์ มีรายละเอียดดังนี้



รูปที่ 5.3 แผนผังการทำงานของโปรแกรมมอนิเตอร์

ส่วนการดำเนินการภาษาพีแอลซีจะเป็นลักษณะของการทำงานเช่นเดียวกับแผนผังในรูปที่ 5.2 ประกอบด้วย อ่านค่าอินพุต, ตัวโปรแกรม และ การส่งค่าเอาท์พุต ส่วนการทำงานในบล็อกที่เหลือจะเป็นการทำงานที่จัดการโปรแกรมที่ใส่ในหน่วยความจำหลักโดยตัวไมโครคอนโทรลเลอร์

5.3.1 การใช้สวิทช์ควบคุมการทำงานของเครื่องพีแอลซี

SW.1 เป็นแบบเลื่อนคือเมื่อ เลื่อนมาทางซ้ายแอลอีดี(LED) จะติดเฉพาะสีเขียว คือ "RUN" คือการดำเนินการภาษาพีแอลซีและเมื่อเลื่อนมาทางขวา แอลอีดีจะติดเฉพาะสีแดง คือ "STOP" เป็นการหยุดการทำโปรแกรมภาษาพีแอลซีโดยไม่ลบข้อมูลใด ๆ ในการทำโปรแกรมก่อนหน้านี้ และเมื่อเลื่อนกลับมาที่ตำแหน่งที่ "RUN" ก็จะดำเนินการต่อไป

เมื่อต้องการโปรแกรมหรือรับข้อมูลที่เป็นตัวโปรแกรมมาไว้ในแรมของพีแอลซีจะต้องทำการเลื่อน SW.1 มาที่ตำแหน่ง "STOP" และ กด SW.2(สวิตช์กด) หนึ่งครั้ง จะเสียงจากลำโพงตอบรับ 1 ครั้ง ก็เท่ากับว่าสามารถส่งข้อมูลจากคอมพิวเตอร์มาที่พีแอลซีได้

และเมื่อเสร็จสิ้นการส่งข้อมูล ก็จะมีเสียงตอบรับจากลำโพง 2 ครั้ง ก็เท่ากับว่าสามารถใช้โปรแกรมที่พึ่งโหลดมาได้แล้ว โดยเลื่อน SW.1 ไปที่ตำแหน่ง "RUN"

5.4 รูปแบบของไฟล์ในการสื่อสาร

ในการส่งโปรแกรมผ่านการสื่อสารมาตราฐาน RS-232 นั้นโปรแกรมที่เขียนไว้ในตัวของไมโครคอนโทรลเลอร์ จะจัดการเฉพาะรูปแบบของ INTEL-HEX FILE ซึ่งเป็นไฟล์มาตราฐานอันหนึ่งที่นิยมใช้กันมาก สังเกตได้จากจากโปรแกรมคอมไพล์เลอร์ หรือ แอสแซมบลี ต่าง ๆ มักจะกำหนดให้เอาท์พุตที่ต้องการเป็น INTEL-HEX FILE ได้ ทั้งนี้เนื่องจากไฟล์ มีรูปแบบที่เหมาะสมหลายประการ กล่าวคือมีระบบการตรวจสอบผลรวม ซึ่งสามารถตรวจสอบความถูกต้องของข้อมูลได้ในแต่ละบรรทัด รวมทั้งการกำหนดตำแหน่ง (Address) ของข้อมูลได้ และที่สำคัญเป็นไฟล์แบบ แอสกี (ASCII) คือสามารถใช้ในการส่งออกทางพอร์ทสื่อสารต่าง ๆ ได้ และยังสามารถใช้กับโปรแกมอิดิตเตอร์ (Editor) ทั่ว ๆ ไปเพื่อแก้ไขได้ (ไฟล์ที่จะใช้ส่งออกทางพอร์ทสื่อสารจะต้องเป็นไฟล์แบบแอสกี เท่านั้นทั้งนี้เนื่องจากการสื่อสาร จำเป็นจะต้องมีรหัสในการสื่อสารเองอยู่แล้ว จึงไม่สามารถใช้ไฟล์ของชิ้นงาน (Object) ได้ เพราะข้อมูลอาจจะไปซ้ำกับรหัสของการสื่อสารได้) รายละเอียดของไฟล์ในแต่ละบรรทัดเป็นดังนี้

:BCAAAATTHH.....HHCC

: คือ ตัวอักษรของการเริ่มบรรทัด

BC คือ จำนวนไบต์ของข้อมูลในบรรทัด มีค่าเป็นเลขฐาน 16 (HEX)
ถ้า BC = 00 จะเป็น บรรทัดสุดท้ายของการบันทึกไฟล์

AAAA คือ ตำแหน่ง (Address) ของข้อมูลในไบต์แรก

TT คือ ชนิดของข้อมูลในบรรทัดนั้น ๆ
ถ้า TT = 00 เป็น การบันทึกข้อมูล
ถ้า TT = 01 เป็น บรรทัดสุดท้ายของการบันทึกไฟล์

HH คือ ข้อมูลในแต่ละไบต์

CC คือ ค่าในการตรวจสอบผลรวมของบรรทัดนั้น ๆ โดยจะเป็นค่า 2'S COMPLEMENT ของผลบวกของข้อมูลทุก ๆ ไบต์ในบรรทัด ซึ่งรวม BC, AAAA และ TT ด้วย

สำหรับเอาท์พุตของโปรแกรมบนคอมพิวเตอร์ จะให้เอาท์พุตเป็นลักษณะนี้ทั้งหมด ยกเว้น CC ซึ่งไม่จำเป็น สามารถตัดออกได้ เพื่อความสะดวกและลดการเสียเวลาแก่การจัดรูปแบบของไฟล์ในการทำโปรแกรม

ในส่วนของโปรแกรมมอนิเตอร์จะทำการตรวจสอบข้อมูลที่รับเข้าเป็น " : " ก่อนเท่านั้น จากนั้นจึงจะเริ่มตรวจสอบตำแหน่งของหน่วยความจำที่โปรแกรมต้องการจะจัดเก็บ โดยตรวจสอบ

ที่ไบต์ของ AAAA เป็นอันดับต่อมา จากนั้นจะทำการส่งหรือเคลื่อนย้ายข้อมูลที่ HH ลงหน่วยความจำ (RAM)

โดยก่อนตรวจสอบรหัสหลัง ":" และ การย้ายข้อมูล จะทำการคำนวณโดยการแปลงรหัส ASCII เป็น HEX Code ก่อน

5.5 การจัดแบ่งพื้นที่ของหน่วยความจำของเครื่อง PLC

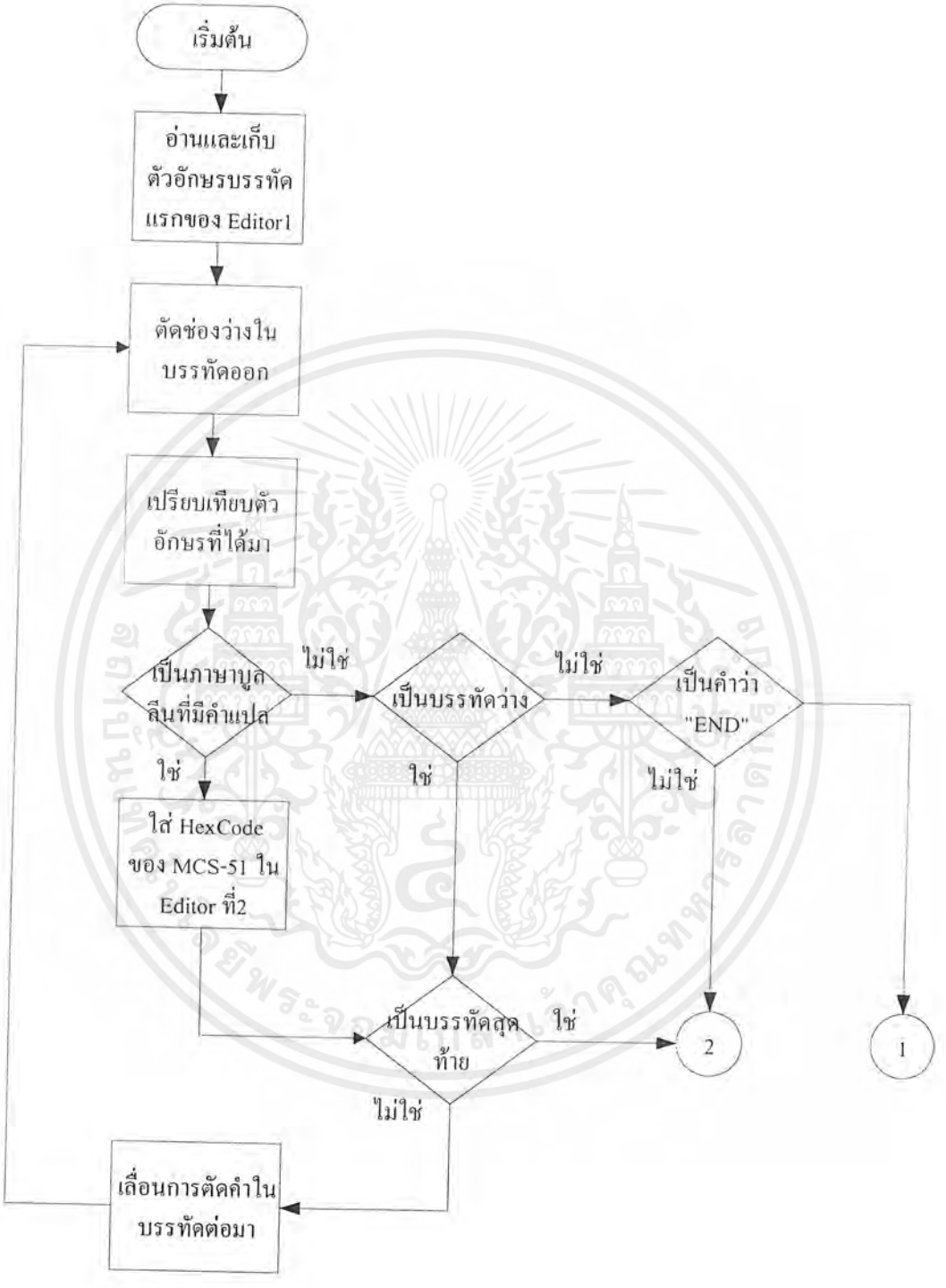
0000-2800	หน่วยความจำหลัก (Flash Memory) เก็บ**โปรแกรมมอนิเตอร์** สำหรับควบคุมการทำงานของฮาร์ดแวร์
4000-BDFF	**ตัวโปรแกรมของพีแอลซี**
BE00-BE03	เก็บสถานะของ X/Y เบอร์ 0-7 พอร์ทที่ 0
BE00	เก็บสถานะปัจจุบันของ X0-7
BE01	เก็บสถานะปัจจุบันของ Y0-7
BE02	เก็บสถานะ Loop ก่อนหน้าของ Y0-7
BE03	เก็บสถานะเริ่มต้น Loop ของ Y0-7
BE04-BE07	เก็บสถานะของ X/Y เบอร์ 8-15 พอร์ทที่ 1
BE08-BE0B	เก็บสถานะของ X/Y เบอร์ 16-23 พอร์ทที่ 2
BE0C-BE0F	เก็บสถานะของ X/Y เบอร์ 24-31 พอร์ทที่ 3
BE10-BE13	เก็บสถานะของ X/Y เบอร์ 32-39 พอร์ทที่ 4
BE14-BE17	เก็บสถานะของ X/Y เบอร์ 40-47 พอร์ทที่ 5
BE18-BE1B	เก็บสถานะของ X/Y เบอร์ 48-55 พอร์ทที่ 6
BE1C-BE1F	เก็บสถานะของ X/Y เบอร์ 56-63 พอร์ทที่ 7
BE30	เก็บสถานะปัจจุบันของ M0-7
BE31	เก็บสถานะ Loop ก่อนหน้าของ M0-7
BE32	เก็บสถานะเริ่มต้น Loop ของ M0-7
BE33	เก็บสถานะปัจจุบันของ M8-15
BE34	เก็บสถานะ Loop ก่อนหน้าของ M8-15
BE35	เก็บสถานะเริ่มต้น Loop ของ M8-15
BE36	เก็บสถานะปัจจุบันของ M16-23
BE37	เก็บสถานะ Loop ก่อนหน้าของ M16-23
BE38	เก็บสถานะเริ่มต้น Loop ของ M16-23
BE39	เก็บสถานะปัจจุบันของ M24-31
BE3A	เก็บสถานะ Loop ก่อนหน้าของ M24-31
BE3B	เก็บสถานะเริ่มต้น Loop ของ M24-31

ตารางที่ 5.1 การจัดพื้นที่หน่วยความจำ(Memory Map)

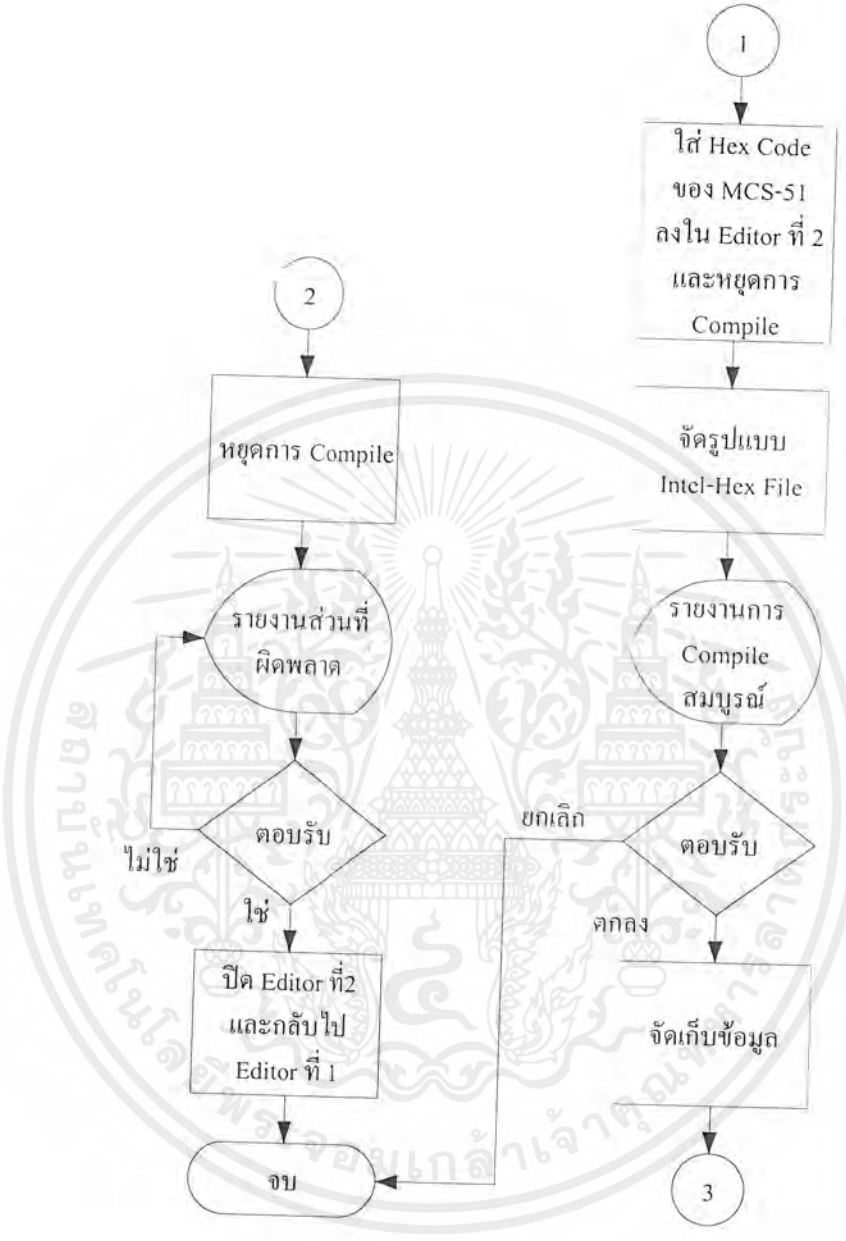
BE40	เก็บสถานะของ C เบอร์ 0 – 7
BE41	เก็บสถานะ การกดสวิตช์ก่อนหน้าของ C0 - 7
BE41-BE49	เก็บค่าของการนับ C0 – 7 ตามลำดับ
BE4A	เก็บสถานะของ C เบอร์ 8 – 15
BE4B	เก็บสถานะ การกดสวิตช์ก่อนหน้าของ C8 –15
BE4C-BE53	เก็บค่าของการนับ C8 – 15 ตามลำดับ
BE60	เก็บสถานะของ T เบอร์ 0 – 7
BE61,BE63,BE65, BE67,BE69,BE6B BE6D,BE6F	เก็บค่าของการเวลาหลักบน C0 – 7 ตามลำดับ
BE62,BE64,BE66, BE68,BE6A,BE6C BE6E,BE70	เก็บค่าของการเวลาหลักล่าง C0 – 7 ตามลำดับ
BE71	เก็บสถานะของ T เบอร์ 10 – 17
BE72,BE74,BE76, BE78,BE7A,BE7C BE7E,BE80	เก็บค่าของการเวลาหลักบน C10 –17 ตามลำดับ
BE73,BE75,BE77, BE79,BE7B,BE7D BE7F,BE81	เก็บค่าของการเวลาหลักล่าง C10 – 17 ตามลำดับ
F000-F007	พอร์ทของบอร์ดอินพุต/เอาท์พุตที่ 0 - 7

ตารางที่ 5.1(ต่อ) การจัดพื้นที่หน่วยความจำ(Memory Map)

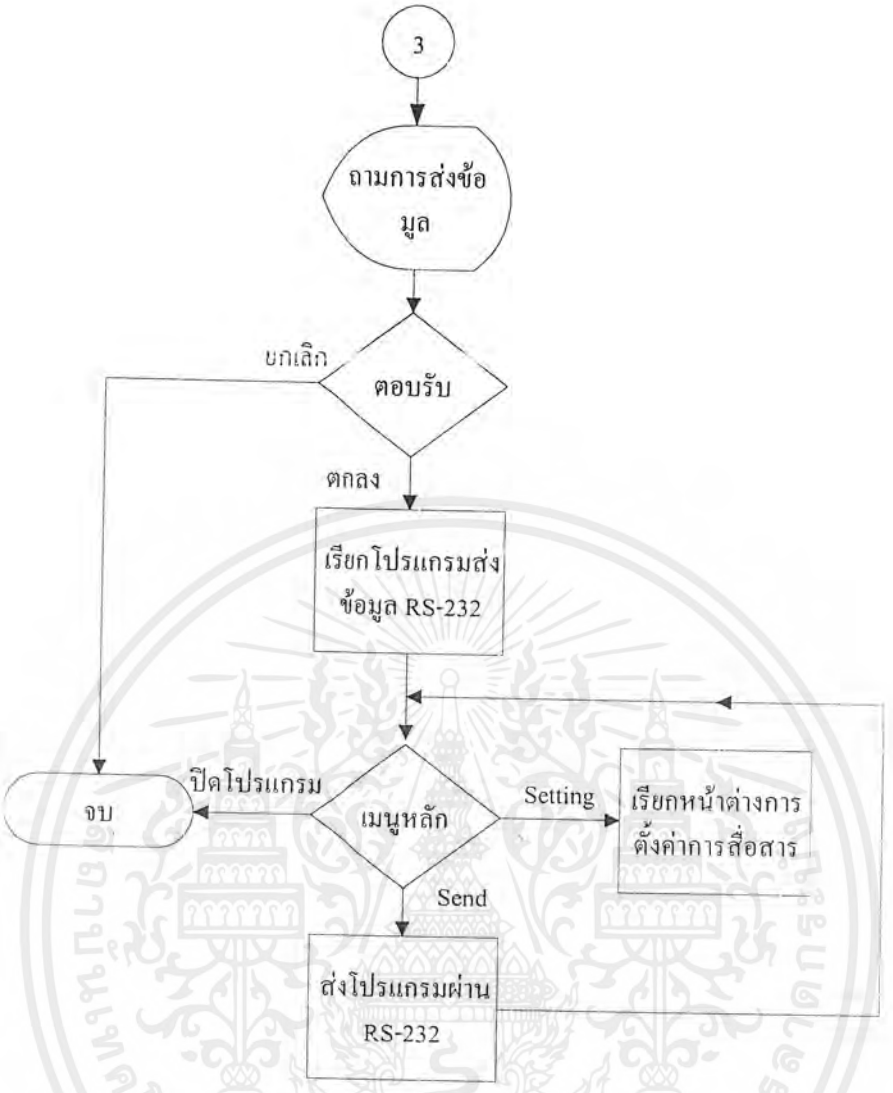
5.6 ซอฟต์แวร์บนเครื่องคอมพิวเตอร์



รูปที่ 5.4 การทำงานของส่วน Compile "F9"



รูปที่ 5.5 การทำงานของส่วน Compile "F9" (ต่อ)



รูปที่ 5.6 การทำงานของส่วน Compile "F9" (ต่อ)

บทที่ 6
การทดลอง

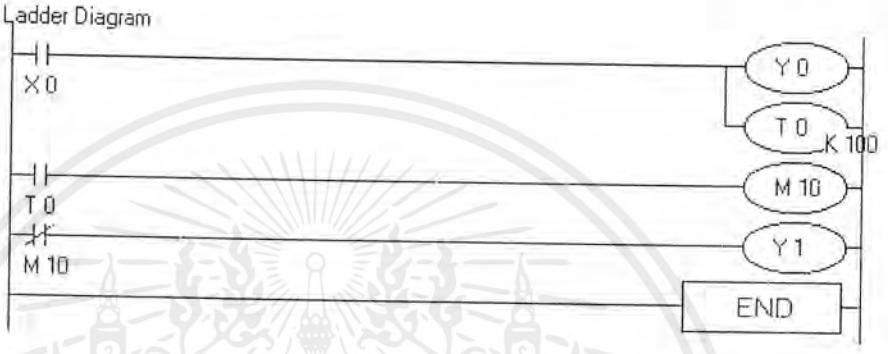
6.1 การทดลองการใช้งานพีแอลซี

6.1.1. วงจรทดสอบการใช้คำสั่ง LD, LDI และ OUT

ภาษาบูลลีน

```
LD   X0
OUT  Y0
OUT  T0
SP   K100

LD   T0
OUT  M10

LDI  M10
OUT  Y1

END
```

ภาษาแลดเดอร์



รูปที่ 6.1 วงจรทดสอบการใช้คำสั่ง LD, LDI และ OUT

การทำงาน

- LD และ LDI เป็นคำสั่งอ่านหน้าสัมผัส ใช้อ่านหน้าสัมผัสของอินพุตออบโต้ X , เอาท์พุตรีเลย์ Y , ไทเมอร์ T , เคาน์เตอร์ C และรีเลย์ช่วย M
- คำสั่ง OUT เป็นคำสั่งขับคอยล์ ใช้ขับคอยล์ของอุปกรณ์ได้ทุกชนิด ยกเว้น X
- สำหรับไทเมอร์และเคาน์เตอร์หลังจากคำสั่ง OUT แล้วยังต้องมีการตั้งค่า K พิจารณาตามคุณสมบัติของเครื่อง
- หน้าสัมผัสของไทเมอร์จะหน่วงเวลาการทำงานหลังจากคอยล์ทำงานแล้ว เวลาที่หน่วงจะเท่ากับค่าที่ตั้งไว้ ไทเมอร์ชนิดนี้เป็น On-delay Timer

ไฟล์ออปเจ้คโค้ด

```
:10400000740078909 0BE00F0A3D8FCF5F0C28DC2
:104010008C758901758C4B758AF9D28CA2905007
:10402000C293D29202402EC292D29302401C90F0
:1040300000E090BE00F090BE03E090BE01F090BE
:1040400035E090BE33F090BE00E0A2E0B3500790
:10405000BE01E0D2E0F04013740090BE62F090BE
```

:1040600061F090BE60E0C2E0F0502630F02390BE

:1040700062E004F0700690BE61E004F090BE61E0

:10408000B4000E90BE62E0B4C80790BE60E0D2E0

:10409000F090BE60E0A2E0500790BE33E0D2E2F0

:1040A00090BE34E0A2E2B3500790BE01E0D2E1F0

:1040B00090BE33E090BE34F0C2F0308D0C758C4B

:1040C000758AF9C28DD28CD2F090BE01E090BE02

:0840D000F090F000F002401C

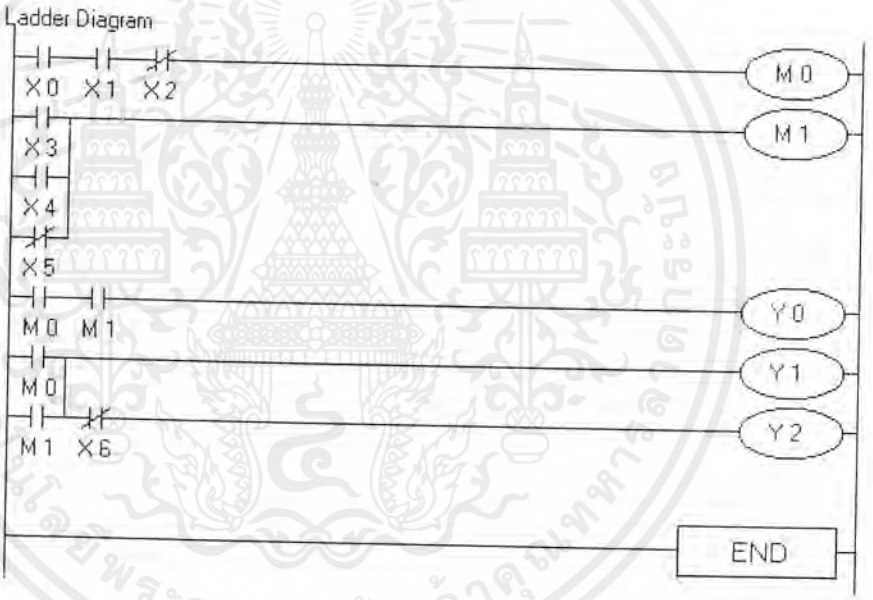
:00000001FF

6.1.2. วงจรทดสอบการใช้คำสั่ง AND, ANI, OR และ ORI

ภาษาบูลลีน

```
LD   X0
AND  X1
ANI  X2
OUT  M0

LD   X3
OR   X4
ORI  X5
OUT  M1

LD   M0
AND  M1
OUT  Y0

LD   M0
OR   M1
OUT  Y1
ANI  X6
OUT  Y2

END
```

ภาษาแลดเดอร์



รูปที่ 6.2 วงจรทดสอบการใช้คำสั่ง AND, ANI, OR และ ORI

การทำงาน

- รีเลย์ช่วย M0 ถูกขับด้วย 3 หน้าสัมผัสที่ต่ออนุกรมกัน M0 จะทำงานเมื่อทั้ง 3 หน้าสัมผัสปิด รีเลย์ช่วย M1 ถูกขับด้วย 3 หน้าสัมผัสที่ต่อขนานกัน M1 จะทำงานถ้า 1 ใน 3 ของหน้าสัมผัสนั้นปิด

- หน้าสัมผัสของ M0 และ M1 ต่อขับเอาท์พุตรีเลย์ Y0 และ Y1 เราอาจไม่จำเป็นต้องใช้รีเลย์ช่วย M0 และ M1 แต่ใช้ Y0 และ Y1 โดยตรงก็ได้แต่จะทำให้วงจรดูซับซ้อนขึ้น

ไฟล์ออปเจ้คโค้ด

```
:10400000740078909 0BE00F0A3D8FCF5F0C28DC2
:10401000 8C758901758C4B758AF9D28CA2905007
:10402000C293D29202402EC292D29302401C90F0
:1040300000E090BE00F090BE03E090BE01F090BE
:1040400032E090BE30F090BE00E0A2E0B390BE00
:10405000E0B0E190BE00E082E2500790BE30E0D2
:10406000E0F090BE00E0A2E3B390BE00E0A0E490
:10407000BE00E072E5500790BE30E0D2E1F090BE
:1040800031E0A2E090BE31E082E1500790BE01E0
:10409000D2E0F090BE31E0A2E090BE31E072E150
:1040A0000790BE01E0D2E1F090BE00E082E65007
:1040B00090BE01E0D2E2F090BE30E090BE31F090
:0E40C000BE01E090BE02F090F000F002401C
:00000001FF
```

6.1.3. วงจรทดสอบการใช้คำสั่ง SET และ RST

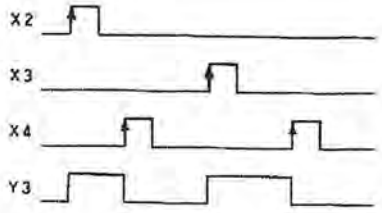
ภาษาบูลลีน

```
LD   X2
SET  Y3

LD   X3
SET  Y3

LD   X4
RST  Y3

END
```

ภาษาแลดเดอร์

รูปที่ 6.3 วงจรทดสอบการใช้คำสั่ง SET และ RST

การทำงาน



รูปที่ 6.4 ไทเมอร์ไดอะแกรมการใช้งาน

- คำสั่ง SET และ RST สามารถใช้กับคอยล์ของรีเลย์เบอร์เดียวกันซ้ำ ๆ หลายครั้งได้
- เมื่อ X2 ON จะทำให้ Y3 ทำงาน และทำงานค้างแม้ X2 จะ OFF ก็ตาม
- เมื่อ X3 ON จะทำให้ Y3 ทำงาน และทำงานค้างแม้ X3 จะ OFF ก็ตาม
- เมื่อ X4 ON จะทำให้ Y3 ไม่ทำงาน และไม่ทำงานค้างแม้ X4 จะ OFF ก็ตาม

ไฟล์ออปเจ้คโค้ด

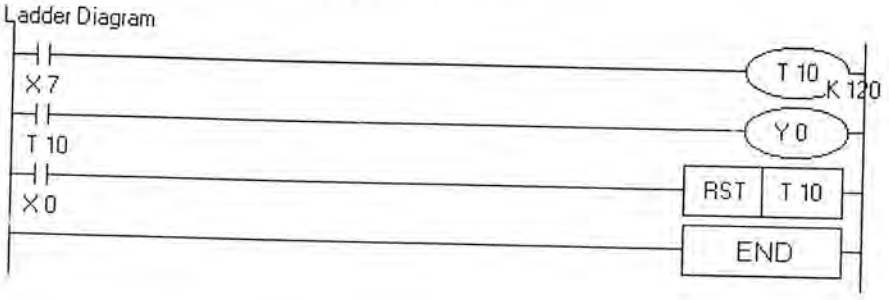
```
:10400000740078909 0BE00F0A3D8FCF5F0C28DC2
:104010008C758901758C4B758AF9D28CA2905007
:10402000C293D29202402EC292D29302401C90F0
:1040300000E090BE00F090BE03E090BE01F090BE
:1040400000E0A2E2B3500790BE03E0D2E3F090BE
:1040500000E0A2E3B3500790BE03E0D2E3F090BE
:1040600000E0A2E4B3500790BE03E0C2E3F090BE
:0D40700001E090BE02F090F000F002401C
:00000001FF
```

6.1.4 วงจรทดสอบการใช้คำสั่ง RST กับ ไทแมอร์ชนิดสะสมเวลา

ภาษาบูลลีน

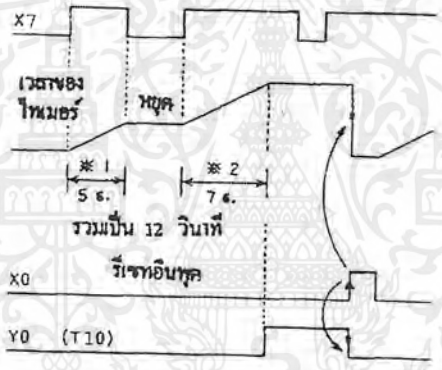
```
LD   X7
OUT  T10
SP   K120

LD   T10
OUT  Y0

LD   X0
RST  T10

END
```

ภาษาแลดเดอร์



รูปที่ 6.5 วงจรทดสอบการใช้คำสั่ง RST กับ ไทแมอร์ชนิดสะสมเวลา

การทำงาน



รูปที่ 6.6 ไทเมอร์ไดอะแกรมการทำงานของไทเมอร์สะสมเวลา

- ขณะที่ X7 ON ไทเมอร์ T10 จะนับเวลาไปเรื่อย ๆ เมื่อ X7 OFF ไทเมอร์จะหยุดนับเวลา แต่จะเก็บค่าเวลานั้นไว้
- ถ้า X7 ON อีกไทเมอร์จะนับต่อไปอีกเช่นนี้จนครอบตามเวลาที่ตั้งไว้ หน้าสัมผัสของไทเมอร์จะ ON
- ถ้ามีสัญญาณรีเซต X0 ON จะทำให้ไทเมอร์ถูกรีเซท ค่าเวลาในไทเมอร์จะเป็นศูนย์ใหม่

ไฟล์ออปเจ้คโค้ด

```
:10400000740078909 0BE00F0A3D8FCF5F0C28DC2
:104010008C758901758C4B758AF9D28CA2905007
:10402000C293D29202402EC292D29302401C90F0
```

:1040300000E090BE00F090BE03E090BE01F090BE

:1040400000E0A2E7B3502630F02390BE73E004F0

:10405000700690BE72E004F090BE72E0B4000E90

:10406000BE73E0B4F00790BE71E0D2E0F090BE71

:10407000E0A2E0500790BE01E0D2E0F090BE00E0

:10408000A2E0B35011740090BE73F090BE72F090

:10409000BE71E0C2E0F0C2F0308D0C758C4B758A

:1040A000F9C28DD28CD2F090BE01E090BE02F090

:0640B000F000F002401C

:00000001FF

6.1.5 วงจรทดสอบการใช้คำสั่ง RST กับ เคาน์เตอร์

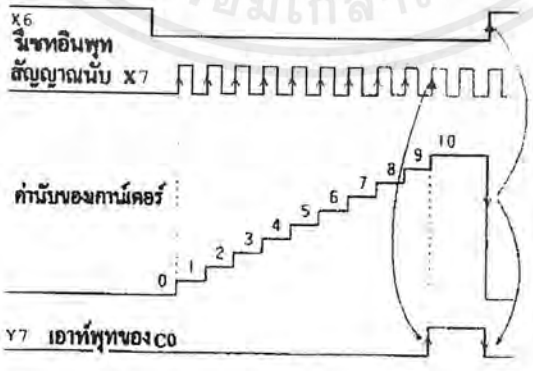
ภาษาบูลลีน

```
LD   X6
RST  C0

LD   X7
OUT  C0
SP   K10

LD   C0
OUT  Y7

END
```

ภาษาแลดเดอร์

รูปที่ 6.7 วงจรทดสอบการใช้คำสั่ง RST กับ เคาน์เตอร์

การทำงาน



รูปที่ 6.8 ไทเมอร์ไดอะแกรมการทำงานของเคาน์เตอร์

- เคาน์เตอร์จะนับเมื่อสัญญาณเข้าที่ใช้นับ X7 เปลี่ยนจาก OFF เป็น ON ค่าที่นับเพิ่มขึ้นเรื่อย ๆ จนถึงค่าที่ตั้งไว้ K ตั้งได้ตั้งแต่ 1-255 หน้าสัมผัสของเคาน์เตอร์ก็จะ ON
- เมื่อนับครบจำนวนที่ตั้งไว้ค่านับจะไม่เปลี่ยนแปลงและหน้าสัมผัสของเคาน์เตอร์จะ ON ค้างแม้จะมีสัญญาณนับเข้ามาอีกก็ตาม
- ถ้ารีเซทอินพุต X6 ON จะทำการรีเซตเคาน์เตอร์ C0 ค่านับจะกลายเป็นศูนย์และหน้าสัมผัสของเคาน์เตอร์ก็จะ OFF

ไฟล์ออปเจ้คโค้ด

```
:10400000740078909OBE00F0A3D8FCF5F0C28DC2
:104010008C758901758C4B758AF9D28CA2905007
:10402000C293D29202402EC292D29302401C90F0
:1040300000E090BE00F090BE03E090BE01F090BE
:1040400000E0A2E6B3500D90BE427400F090BE40
:10405000E0C2E0F090BE00E0A2E7B3501D90BE41
:10406000E092F1B0E0501190BE42E005E0F0B40A
:104070000790BE40E0D2E0F0A2F190BE41E092E0
:10408000F090BE40E0A2E0500790BE01E0D2E7F0
:1040900079FB7A04DAFED9FA90BE01E090BE02F0
:0740A00090F000F002401C
:00000001FF
```

6.1.6 วงจรไฟกระพริบ (Flicker circuit)

ภาษาบูลลีน

```
LD   X7
ANI  T3
OUT  T0
SP   K20

LD   T0
OUT  T3
SP   K10
OUT  Y6

END
```

ภาษาแลดเดอร์

รูปที่ 6.9 วงจรไฟกระพริบ (Flicker circuit)

การทำงาน

รูปที่ 6.10 ไทเมอร์ไดอะแกรมการทำงานของวงจรไฟกระพริบ

ไฟล์ออปเจ้คโค้ด

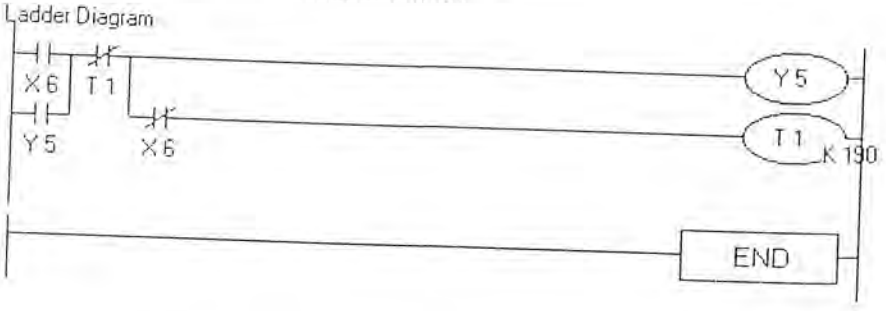
```
:1040000074007890 90BE00F0A3D8FCF5F0C28DC2
:104010008C758901758C4B758AF9D28CA2905007
:10402000C293D29202402EC292D29302401C90F0
:1040300000E090BE00F090BE03E090BE01F090BE
:1040400000E0A2E7B390BE60E0B0E34013740090
:10405000BE62F090BE61F090BE60E0C2E0F05026
:1040600030F02390BE62E004F0700690BE61E004
:10407000F090BE61E0B4000E90BE62E0B4280790
:10408000BE60E0D2E0F090BE60E0A2E040137400
:1040900090BE68F090BE67F090BE60E0C2E3F050
:1040A0002630F02390BE68E004F0700690BE67E0
:1040B00004F090BE67E0B4000E90BE68E0B41407
:1040C00090BE60E0D2E3F0500790BE01E0D2E6F0
:1040D000C2F0308D0C758C4B758AF9C28DD28CD2
:1040E000F090BE01E090BE02F090F000F002401C
:0040F000
:00000001FF
```

6.1.7. วงจรตั้งเวลา (OFF Delay Timer)

ภาษาบูลลีน

```
LD      X6
OR      Y5
ANI     T1
OUT     Y5
ANI     X6
OUT     T1
SP      K190

END
```

ภาษาแลดเดอร์



รูปที่ 6.11 วงจรตั้งเวลา (OFF Delay Timer)

การทำงาน

รูปที่ 6.12 ไทเมอร์ไดอะแกรมการทำงานของวงตั้งเวลา

X6 เป็นอินพุตของการเริ่มการตั้งเวลา ถ้า X6 เปลี่ยนจากปิดเป็นเปิด จากนั้น 19 วินาที Y6 จะหยุดทำงาน วงจรตั้งเวลานับจากการ OFF ของหน้าสัมผัส เรียกว่า "OFF Delay Timer"

ไฟล์ออปเจ็คโค้ด

```
:10400000740078909 0BE00F0A3D8FCF5F0C28DC2
:104010008C758901758C4B758AF9D28CA2905007
:10402000C293D29202402EC292D29302401C90F0
:1040300000E090BE00F090BE03E090BE01F090BE
:1040400000E0A2E6B390BE02E072E590BE60E0B0
:10405000E1500790BE01E0D2E5F090BE00E082E6
:1040600040137400 90BE64F090BE63F090BE60E0
:10407000C2E1F0502630F02390BE64E004F07006
:1040800090BE63E004F090BE63E0B4010E90BE64
:10409000E0B47C0790BE60E0D2E1F0C2F0308D0C
:1040A000758C4B758AF9C28DD28CD2F090BE01E0
```

:0B40B00090BE02F090F000F002401C
:00000001FF

6.2 สรุปการทดลองใช้งานพีแอลซี

การทดลองในวงจรทั้ง 7 นี้เป็นตัวอย่างการใช้งานจริง ที่เครื่องพีแอลซีสามารถทำงานได้อย่างถูกต้องแน่นอน ผู้ใช้สามารถประยุกต์ใช้ได้อีกมากมาย

จากโปรแกรมทั้งหมดนี้สามารถทำการทดลองกับอุปกรณ์ที่ติดต่อกับอินพุต/เอาท์พุต ได้โดยการเพิ่มเบอร์ของอุปกรณ์ ที่เป็นหมายเลขที่นอกเหนือจากหมายเลขของบอร์ดอินพุต/เอาท์พุตหลัก เในการเลือกหมายเลขในพอร์ทขยายนั้นต้องพิจารณาตามตำแหน่งของการตั้งค่าจั้มเปอร์ของบอร์ดด้วย

การทำงานของพีแอลซีที่ทำขึ้นนี้ อาศัยการทำงานร่วมกันระหว่างวงจรภายนอกซึ่งได้แก่ วงจรอินพุต, วงจรเอาท์พุต และวงจรคอนโทรเลอร์ ทำงานร่วมกับตัวโปรแกรมบนเครื่องคอมพิวเตอร์ ในที่นี้ขอเรียกว่า "โปรแกรมคอมไพล์เลอร์" ซึ่งการทำงานก็คล้ายกับที่วางขายอยู่ แต่มีฟังก์ชั่นการทำงานน้อยกว่า เหตุผลที่ต้องเขียนโปรแกรมบูลลีนขึ้นเอง เนื่องมาจากตัวไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์ที่ใช้มีชุดคำสั่งต่างกัน และการซิงค์กันระหว่างวงจรภายนอกกับตัวโปรแกรมของวงจรที่วางขายอยู่เราไม่สามารถรู้ได้ ซึ่งเราจะต้องกำหนดเองทำให้มีความจำเป็นต้องเขียนโปรแกรมบูลลีนขึ้นมาเอง

ในการทดลองเราได้แบ่งการทดลองออกเป็นส่วนหลักได้ 3 ส่วนคือ ส่วนวงจรภายนอก, ส่วนโปรแกรมภาษาคอมไพล์เลอร์ และการทำงานร่วมกันระหว่างวงจรภายนอกกับโปรแกรมภาษาคอมไพล์เลอร์

6.3 การทดสอบการทำงานของส่วนฮาร์ดแวร์

เราจะแบ่งการทำลองออกเป็นส่วนย่อยอีก ซึ่งจะแบ่งได้ประมาณ 4 ส่วน คือ

6.3.1 การทดสอบการทำงานของไมโครคอนโทรลเลอร์กับหน่วยความจำรอม

ทำการทดสอบโดยใช้ AT89S8252 ที่โปรแกรมแล้ว เมื่อเปิดเครื่องโปรแกรมจะทำการตรวจสวิตช์ และทำการเปิดปิด แอลอีดี ตามสถานะของสวิตช์ โปรแกรมสำคัญที่บันทึกในหน่วยความจำส่วนนี้คือ การสื่อสารข้อมูลผ่าน RS-232 โดยโปรแกรมจะทำการเปลี่ยนรหัสแอสกี เป็น เลขฐาน 16 และทำการเคลื่อนย้ายลงหน่วยความจำแรม ก่อนการทดสอบการโปรแกรมในรอมส่วนการเปลี่ยนรหัสข้อมูลได้นั้น ต้องผ่านการทดสอบในหัวข้อการทดสอบการเชื่อมต่อข้อมูลระหว่างไมโครลคอนเลอร์และเครื่องคอมพิวเตอร์ผ่านการสื่อสาร RS-232 ก่อน

เมื่อผ่านการทดสอบแล้ว ในการทดสอบโดยการเขียนโปรแกรมในอิดิตเตอร์ของดอสโหมด เป็นตัวเขียนโดยเลือกโปรแกรมง่าย ๆ ในการทดสอบ ในการเขียนในอิดิตเตอร์จะเขียนเป็นภาษาแอสแซมบลีของ เอ็มซีเอส-51 และ ใช้คอมไพล์เลอร์ชื่อ SXA51.EXE เป็นตัวเปลี่ยนเป็นไฟล์ออปเจ้คโค้ด

เนื่องจากการตั้งค่าของหน่วยความจำในบอร์ดหลัก มีหน่วยความจำเริ่มต้นที่ 4000H เพราะฉะนั้น ที่หัวของการเขียนโปรแกรมแอสแซมบลี เขียน " ORG 4000H "

เมื่อทำการคอมไพล์เสร็จแล้ว จะใช้การส่งโปรแกรมบนดอส โดยเริ่มจากพิมพ์การตั้งค่าการสื่อสารของ RS-232

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

เป็นการตั้งค่าการสื่อสารโดยใช้ขั้วต่อ COM1 , Baud Rate:9600 , Parity:None , Data Bits:8 , Stop Bits:1 ซึ่งเป็นรูปแบบเดียวกับที่ตั้งไว้ที่เครื่องพีแอสซี จากนั้นทำการส่งไฟล์ที่ต้องการส่ง

C:\COPY ชื่อไฟล์.HEX COM1

เมื่อกดเอ็นเตอร์แล้วหากมีการตอบรับจากเครื่องด้วยสัญญาณเสียง ก็หมายความว่าการส่งข้อมูลครบสมบูรณ์ และลองใช้งานโปรแกรม โดยการเลื่อน SW.1 มาที่ "RUN" ไฟสีเขียวก็จะติด

พิจารณาการทำงานของโปรแกรมว่าถูกต้องหรือไม่ ข้อสำคัญของการเลือกโปรแกรมทดลอง คือ เป็นโปรแกรมที่สั้นและมีการส่งค่าออกพอร์ทที่แสดงผลได้ชัดเจน เพื่อการตรวจสอบที่ง่ายขึ้น

6.3.2 การทดสอบการทำงานของไมโครคอนโทรลเลอร์กับหน่วยความจำแรม

การทดสอบในหัวข้อนี้คือทดสอบการใช้งานแรมภายใต้การทำงานของ ไมโครคอนโทรลเลอร์สามารถทดสอบได้คือ การนำเอา NVRAM ไปใส่ในบอร์ดอเนกประสงค์ที่ควบคุมแรมด้วยไมโครคอนโทรลเลอร์ แล้วโหลดโปรแกรมใช้งานจริง พิจารณาการเขียนโปรแกรมลงในแรมด้วยโครงสร้างของฮาร์แวร์พีแอสซี ตัวบอร์ดอเนกประสงค์ เป็นของบริษัทใดก็ได้ บอร์ดจะเป็นเพียงตัวช่วยโหลดโปรแกรม แล้วถอด NVRAM มาใส่ในเครื่องพีแอลซี

หรืออีกวิธีคือการใช้อีพรอมอีมูเลเตอร์ ตั้งค่าให้เป็นแรม ขนาด 32K ทั้งสองวิธีเลือกใช้ตามเครื่องมือที่มีอยู่ การทดสอบส่วนนี้หากการต่อวงจรไม่มีความผิดพลาด หรือแรมไม่เสียแล้วการทำงานตามโปรแกรมก็จะไม่มีปัญหา

6.3.3 การทดสอบการเชื่อมต่อข้อมูลระหว่างไมโครลคอนเลอร์และเครื่องคอมพิวเตอร์ผ่านการสื่อสาร RS-232

ไอซีที่เป็นตัวกลางระหว่างไมโครคอนโทรลเลอร์และเครื่องคอมพิวเตอร์ คือ เบอร์ DS275 สามารถทดการทดสอบได้คือ ทำการก๊อปไฟลออปเจ้คโค้ด ในโหมดการทำงานของดอส เมื่อการทด

งานถูกต้องไอซีต่อถูกต้องตามวงจร สัญญาณที่ออกจากไอซีจะอยู่ที่ระดับ 0-5V เข้าขา RXD ของไมโครคอนโทรลเลอร์

6.3.4 การทดสอบการติดต่อกับพอร์ทอินพุต/เอาท์พุต

สามารถทดการทดสอบโดยการเขียนโปรแกรมไฟวิ่ง การทำงานของโปรแกรมนี้คือวิ่งลักษณะของ การนับเลขไบนารี่ หากสวิตช์ปิดทั้งหมดจะเป็นการนับขึ้น และถ้าสวิตช์ตัวใดเปิดการนับเลขก็จะตรงกันข้าม

สามารถใช้ทดสอบอินพุตสวิตช์ , เอาท์พุตรีเลย์ และ ยังสามารถทำการทดสอบไปในส่วนของบอร์ดขยายได้ โดยการเปลี่ยนคำสั่งที่เบอร์ของพอร์ท

```
                ;Program test Input/Output
PORTA           EQU   0F000H     ;OUT
PORTB           EQU   0F000H     ;IN
                ORG    4000H
                ;
                MOV    R7,#0H
                MOV    DPTR,#PORTA
                MOV    A,#0H
START_A:        MOV    C,P1.0
                JNC    LOOPA
                clr    p1.3
                setb   p1.2
                JMP    LOOPB
LOOPA:          clr    p1.2
                setb   p1.3
                LJMP   START_A ;0040H
LOOPB:          MOVX   @DPTR,A
                MOV    DPTR,#PORTB
                MOVX   A,@DPTR
                JNZ    GO
                ;
                MOV    DPTR,#PORTA
                MOV    A,R7
                MOVX   @DPTR,A     :OUT PORT A
                INC    R7
                LCALL  DELAY
                LJMP   START_A
                ;
```

```
GO:     MOV     DPTR,#PORTA
        MOV     A,R7
        MOVX    @DPTR,A     :OUT PORT A
        DEC     R7
        LCALL   DELAY
        LJMP    START_A
        ;***** Subrountine *****
        ;
DELAY:  PUSH    ACC
        PUSH    PSW
        MOV     R1,#0FFH
DLAY2:  MOV     R2,#0FFH
        DJNZ    R2,$
        DJNZ    R1,DLAY2
        POP     PSW
        POP     ACC
        RET
        ;
        END
```

6.4 สรุปคุณสมบัติของพีแอลซีที่ทำขึ้น

- จำนวนอินพุต 8 อัน
- จำนวนเอาท์พุต 8 อัน
- สามารถขยายอินพุต/เอาท์พุตได้อีกอย่างละ 56 อัน
- หน่วยความจำส่วนรอม (Flash Memory) 8Kx8 บิต
- หน่วยความจำส่วนแรม (64256 หรือ NVRAM) 32Kx8 บิต
- ตัวประมวลผลของวงจรภายนอก AT89S8252
- ส่วนอินพุตเป็นแบบโต้คอปเปลอร์ มีแอลอีดีบอกสถานะ
- ส่วนเอาท์พุตเป็นรีเลย์ มีแอลอีดีบอกสถานะ
- มีซอฟต์แวร์บนเครื่องคอมพิวเตอร์ที่ทำขึ้น รองรับการเขียนโปรแกรมบูลลีน ในระหว่างการเขียนสามารถดูแลดเดอร์ไดอะแกรมได้ ซอฟต์แวร์สามารถเปลี่ยนโปรแกรมบูลลีนที่เขียนขึ้นเป็นไฟล์ออปเจ้คโค้ดของไมโครคอนโทรลเลอร์ ตระกูลเอ็มซีเอส-51 และ ซอฟต์แวร์สามารถทำการโหลดไฟล์ลงเครื่องพีแอลซีได้

6.5 ปัญหาที่พบในการทำงาน

ปัญหาการเกิดบาวนซ์ (Bounce)

ในการกดสวิตช์โดยทั่ว ๆ ไปจะปรากฏพัลซ์ที่ไม่ต้องการเกิดขึ้นมา ซึ่งเกิดผลเนื่องจากการกระเด้งของหน้าสัมผัสปรากฏการณ์นี้เรียกว่า บาวนซ์ (Bounce) วิธีการแก้บาวนซ์ (Debounce) สามารถทำได้โดยวิธีการทางฮาร์ดแวร์ หรือ ซอฟต์แวร์ก็ได้ วิธีการทางฮาร์ดแวร์ โดยทั่ว ๆ ไป จะใช้วงจรแนนด์แลทซ์ หรือ วงจรโมโนสเตเบิล เพื่อแก้สัญญาณบาวนซ์นี้

สัญญาณในลักษณะบาวนซ์นี้จะมีผลต่อวงจรลอจิกอย่างมาก อย่างในพีแอลซีเอาสัญญาณที่จุดนี้ไปเป็นอินพุตของวงจรนับ จะทำให้การนับผิดพลาด คือ เมื่อกดสวิตซ์ 1 ครั้งควรได้ 1 พัลส์ แต่ถ้ามีบาวนซ์เกิดขึ้นจะทำให้ได้พัลส์มากกว่า 1 พัลส์ ค่าที่ได้จึงมากกว่าความต้องการ การที่ใช้ซีพียูอ่านข้อมูลจากสวิตซ์ในลักษณะเช่นนี้ข้อมูลที่ได้อาจจะไม่แน่นอน คืออาจจุได้เป็น 1 หรือ 0 ก็ได้ ดังนั้นจึงต้องมีการกำจัดสัญญาณส่วนที่ไม่ต้องการนี้ออกไป ซึ่งอาจทำได้โดยวิธีการทางด้านฮาร์ดแวร์หรือซอฟต์แวร์ แต่วิธีที่นิยมใช้ในระบบคอมพิวเตอร์คือใช้ซอฟต์แวร์ เพราะไม่ต้องทำการเพิ่มเติมอุปกรณ์อื่น ๆ อีกเลย

การแก้สัญญาณบาวนซ์โดยใช้ซอฟต์แวร์ โปรแกรมจะทำงานโดยการตรวจสอบการกดสวิตซ์ครั้งแรก แล้วสักระยะเวลาอีกช่วงหนึ่งเพื่อให้ผ่านช่วงของสัญญาณบาวนซ์ จากนั้นจึงอ่านสถานะของสวิตซ์อีกครั้งหนึ่ง แล้วทดสอบว่าสวิตซ์ยังถูกกดอยู่หรือไม่ ถ้าสวิตซ์ยังคงถูกกดอยู่ จึงนำข้อมูลที่ได้นี้เข้าไป วิธีการนี้เป็นวิธีที่นิยมใช้โดยทั่ว ๆ ไป และจะใช้เวลาที่หน่วงประมาณ 10 mS

กรณีของเครื่องพีแอลซีนี้จะมีปัญหาในเรื่องของการให้คำสั่งของเคาน์เตอร์ที่จะให้ในการนับการแก้ไขดังที่กล่าวไว้แล้ว คือ สร้างโปรแกรมเดลไฟล์ให้มีการแทรกส่วนการหน่วงเวลาขึ้นเพื่อแก้ไขบาวนซ์ โดยตรวจการตรวจสอบการเขียนโปรแกรมภาษาบูลลีนหากมีการใช้คำสั่งเคาน์เตอร์จะมีการเพิ่มคำสั่งหน่วงเวลาเพิ่มเข้าไปในไฟล์ออปเจ้คโค้ด ในการเขียนโปรแกรมใช้เวลาหน่วงเพิ่ม 3 mS แต่หากไฟล์ออปเจ้คโค้ดมีรอบของสแกนไทม์มากกว่าหรือเท่ากับ 10 mS แล้วจะไม่มีการเพิ่มคำสั่งหน่วงลงไปอีก

การแก้ปัญหาโดยใช้ซอฟต์แวร์นี้ก็มีข้อเสีย ในเครื่องพีแอลซี กล่าวคือ ในเรื่องของการดำเนินการทางโปรแกรมของตัวพีแอลซีนั้นจะทำงานเป็นลูปรอบ เมื่อมีมีการทำการหน่วงแล้ว รอบของการสแกนไทม์ก็จะเพิ่มขึ้น ซึ่งจะส่งผลแก่การใช้คำสั่งไทเมอร์ ทำให้ฐานเวลาการนับคาดเคลื่อนได้ แต่ก็เป็นการคาดเคลื่อนที่น้อยมาก ๆ

บทที่ 7
บทสรุปและวิจารณ์

จากการทำโครงงานพีแอลซีในส่วนของซอฟต์แวร์ต้องทำการเขียนโปรแกรมภาษาเดลไฟล์ (DELPHI) เพื่อแปลงภาษาบูลลีนเป็น ไฟล์ในรูปแบบของ Intel-Hex File ซึ่งมีขั้นตอนในการแปลงภาษามากมาย และต้องใช้วิธีการตัดคำออกเป็นส่วน ๆ ซึ่งทางผู้จัดทำต้องใช้เวลาในการศึกษาโปรแกรมภาษาต่าง ๆ และพบปัญหาในการคอมไพล์ (Compile) ,ความไม่สอดคล้องกันระหว่างภาษาต่าง ๆ , ปัญหาในการแปลงภาษาแอสเซมบลี้เป็นออพโคด (Opcode) โดยใช้ภาษาเดลไฟล์

ในส่วนของฮาร์ดแวร์ การทดสอบวงจรไมโครคอนโทรลเลอร์ซึ่งต้องแยกส่วนในการทดสอบ เช่น

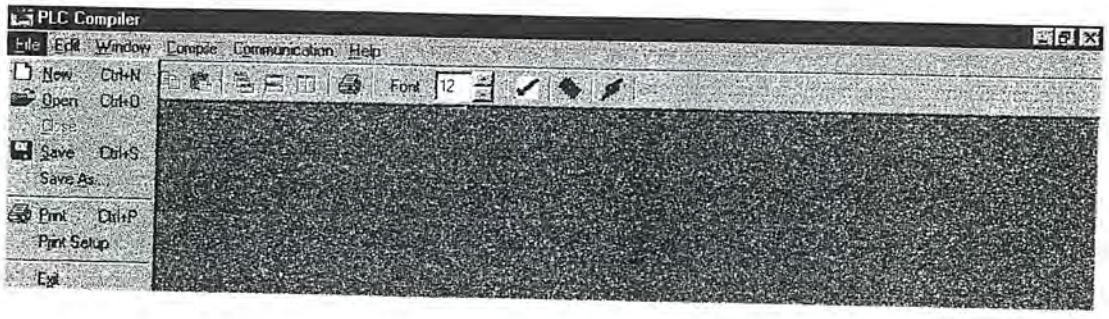
- การเก็บข้อมูลลงการใช้รอมและแรม
- การส่งข้อมูลแบบอนุกรมผ่านพอร์ท RS-232
- ทดสอบการทำงานของพอร์ต

ส่วนวงจรอินพุตและวงจรเอาท์พุตจะไม่ค่อยมีปัญหามากนักเนื่องจาก แต่ละส่วนเป็นวงจรเล็ก ๆ แต่เมื่อนำส่วนของฮาร์ดแวร์ทั้งหมดมารวมกันแล้ว ก็จะพบปัญหาในการรันโปรแกรม

จุดประสงค์ของการทำปริญญานิพนธ์นี้คือสามารถนำพีแอลซีไปใช้ในการควบคุมสวิตซ์ของเครื่องจักรต่าง ๆ ได้จริง แล้วยังจะทำให้ประหยัดเวลาในการปิด / เปิดสวิตช์ ประหยัดพลังงานไฟฟ้า และยังช่วยควบคุมกระบวนการผลิตถูกต้องตามที่ได้โปรแกรมไว้ แต่พีแอลซีนี้ ยังมีข้อเสียอยู่บ้าง เช่น ขนาดของบอร์ดของส่วนประมวลผล ส่วนอินพุตและส่วนเอาท์พุตมีขนาดใหญ่เกินไปซึ่งไม่เหมาะสมกับการใช้งานจริง ซึ่งถ้าทำการออกแบบลายวงจรใหม่ โดยที่ใช้อุปกรณ์เหมือนเดิมทุกอย่างก็จะทำให้พีแอลซีมีขนาดเล็กลง และสามารถนำไปใช้งานได้จริง โดยนำส่วน NO ของรีเลย์ทั้ง 8 ตัว ไปต่อกับสวิตช์ของเครื่องจักรต่าง ๆ

ผลการทดลองที่ได้ออกมานั้นเป็นที่น่าพอใจพีแอลซีสามารถทำงานตามคำสั่งของผู้ใช้ได้จริงและมีความรวดเร็วด้วย

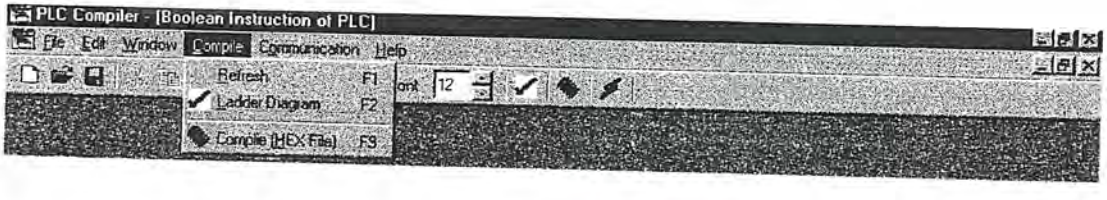
ภาคผนวก



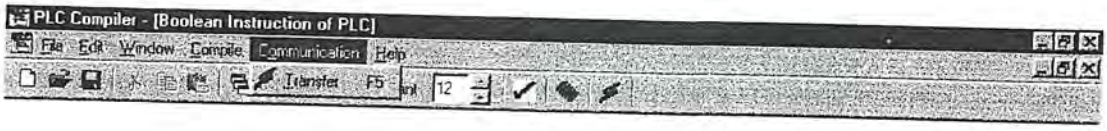
รูปที่ 1 เมนูหลักในการจัดการไฟล์

รูปที่ 2 เมนูในการจัดการในกรอบของ Editor

รูปที่ 3 เมนูในการจัดรูปของหน้าต่าง

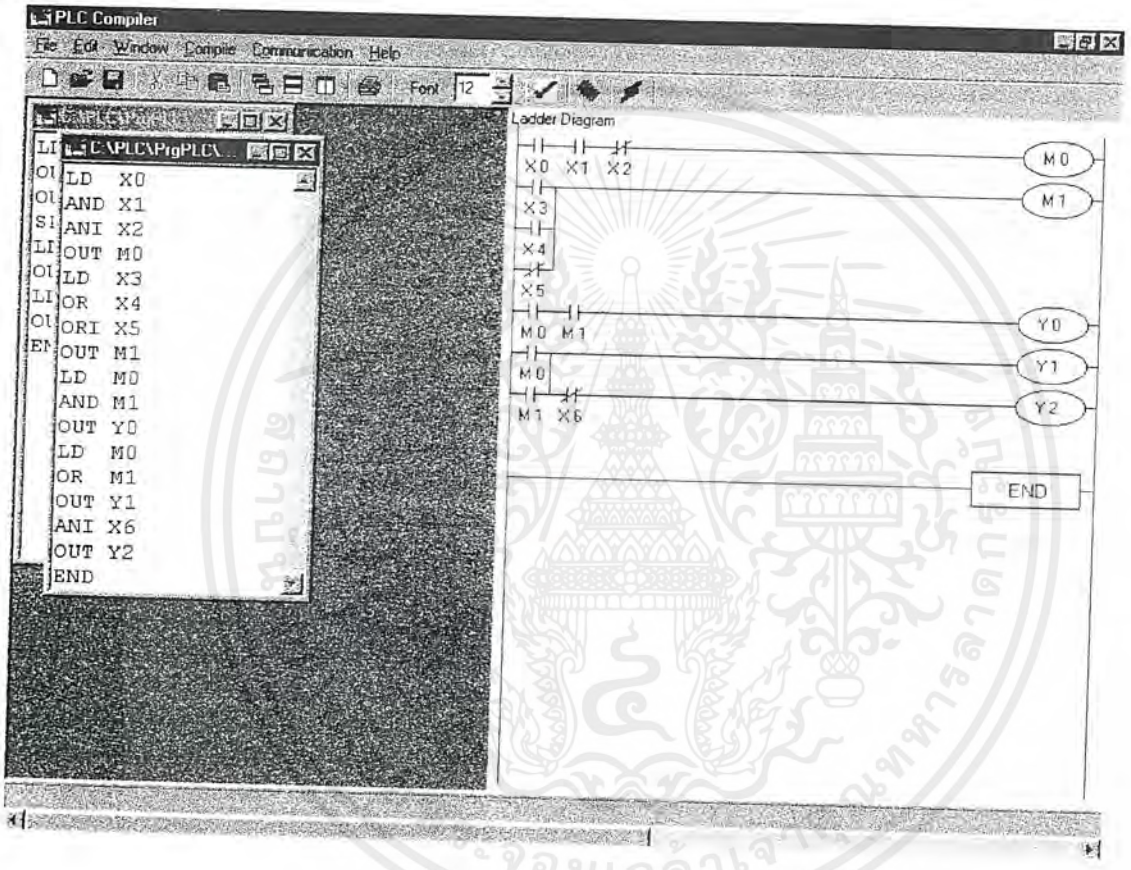


รูปที่ 4 เมนูในการ Compile จากภาษาบูลลีนเป็นแบบต่าง ๆ



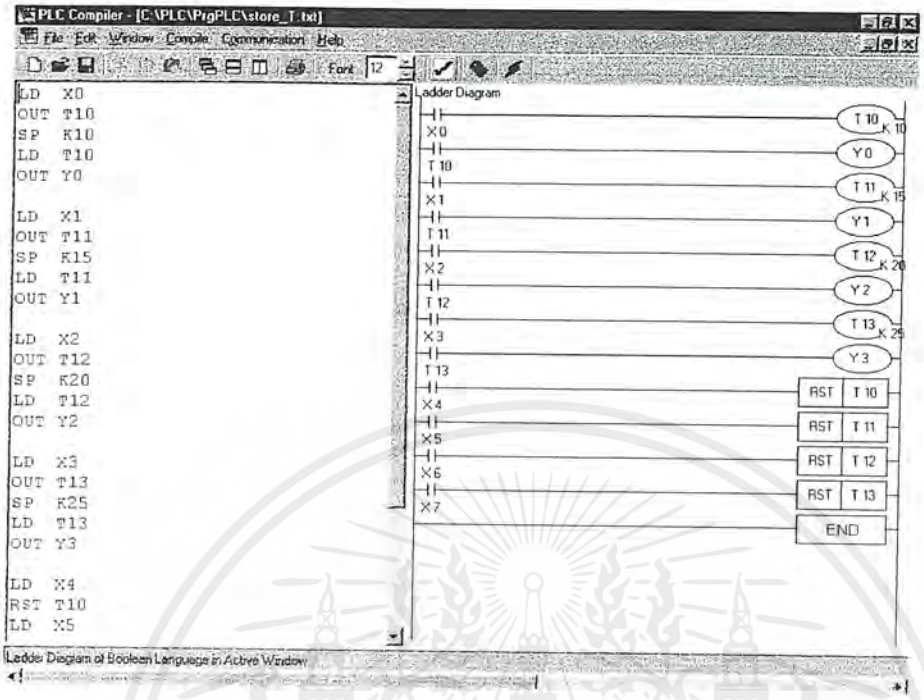
รูปที่ 5 เมนูในการเลือกโปรแกรมในการส่งโปรแกรม

การใช้งาน

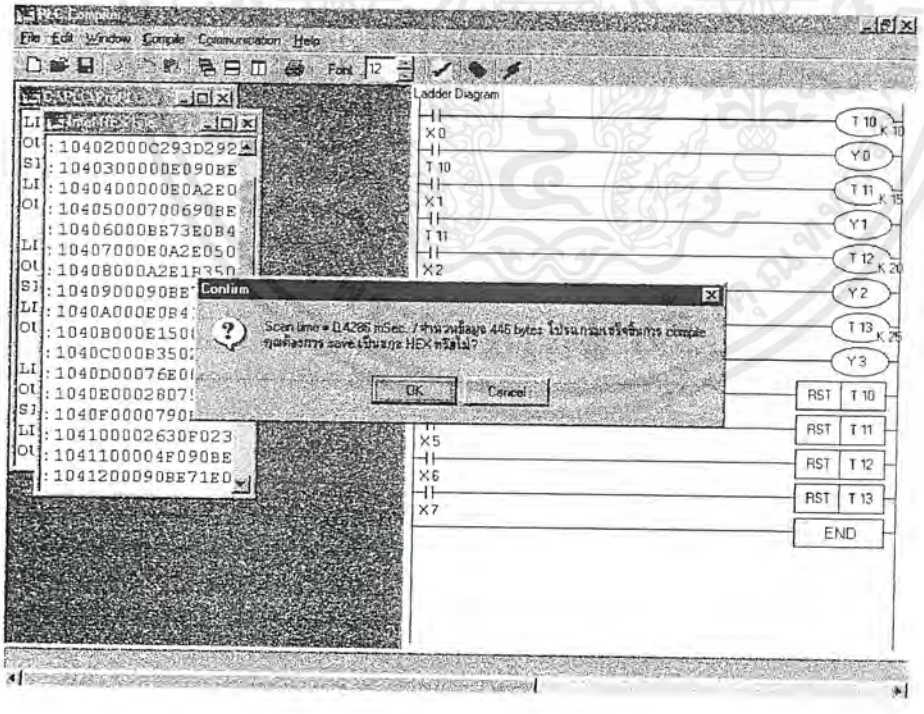


รูปที่ 6 แสดงการใช้งาน

- สามารถเริ่มจากการเลือกที่จะสร้างโปรแกรมใหม่ หรือ การ เปิดโปรแกรมที่สร้างไว้แล้ว โดยเป็นไฟล์สกุล txt หรือที่เป็นไฟล์สำรองสกุล bak (ซอฟต์แวร์จะสำรองไฟล์กรณีของการ Save As)
- จากนั้นสามารถเรียกดูแลดเดอร์ไดอะแกรม ได้โดยการกด F2 และทำการเปลี่ยนภาษาบูลลีนเป็นรูปแลดเดอร์ไดอะแกรม โดยการกด F1

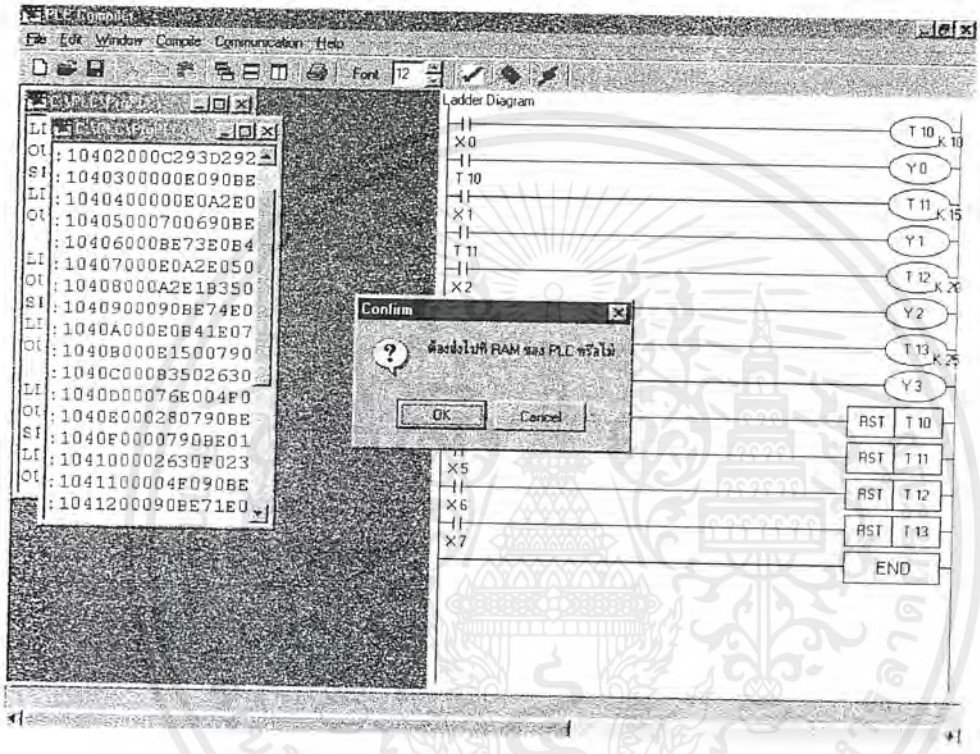


รูปที่ 7 แสดงการเขียนโปรแกรม



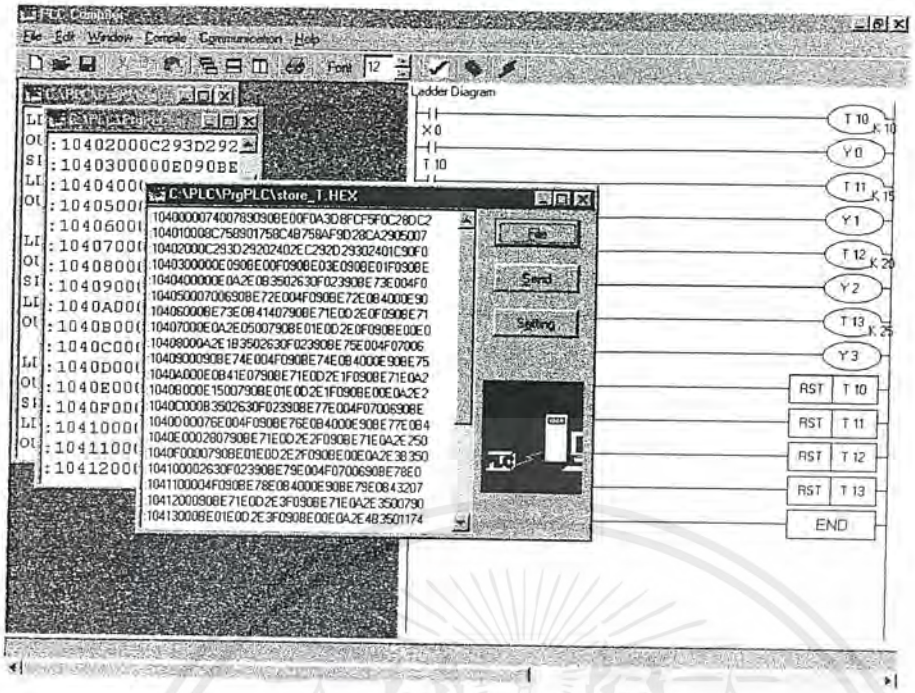
รูปที่ 8 แสดงการแปลภาษา (Compile)

- เมื่อทำการเขียนโปรแกรมที่ต้องการเสร็จแล้วทำการ Compile โดยการเปลี่ยนเป็นไฟล์ ออปเจ้คโค้ด โดยการกด F9 หรือ จะเลือกบน Mainmenu หรือ Speedbar
- เมื่อการเขียนโปรแกรมถูกต้อง โปรแกรมก็จะแสดงการรายงานจำนวนของ Scan time ของการดำเนินการของโปรแกรมใน 1 รอบ และจำนวนการใช้เนื้อที่ใน RAM ของเครื่องพีแอลซี
- จากนั้นโปรแกรมจะถามการจัดเก็บข้อมูล หากจัดเก็บ โปรแกรมจะถามต่อในรูปที่ 9



รูปที่ 9 การสอบถามหลังการจัดเก็บไฟล์ที่ Compile สมบูรณ์

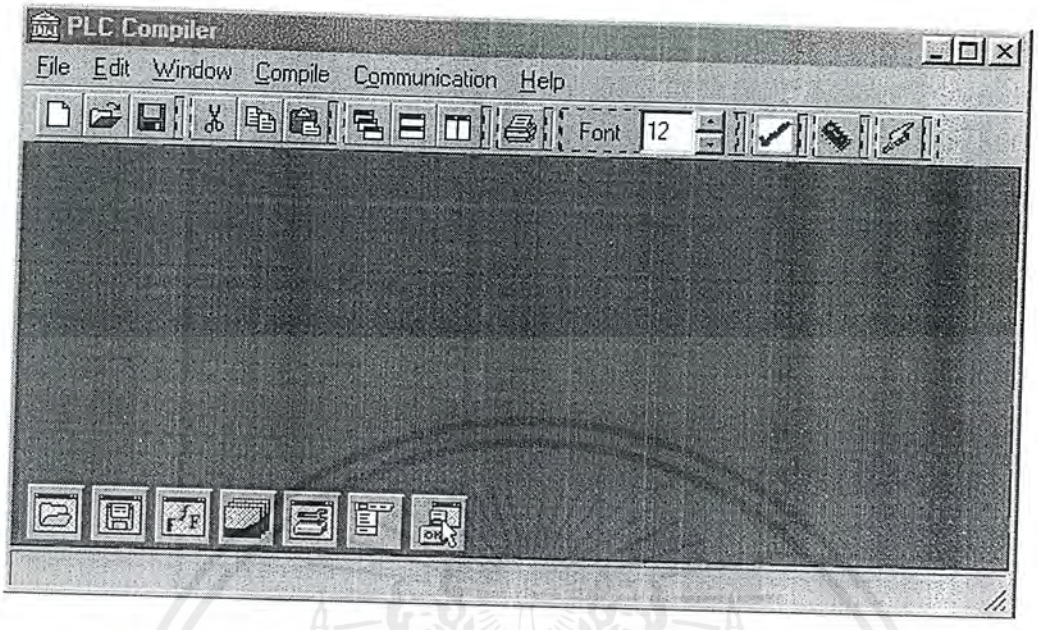
- เมื่อโปรแกรมผู้ต้องการส่งโปรแกรมก็ให้ตอบตกลง โปรแกรมก็จะทำการเรียกโปรแกมย่อยในการสื่อสารข้อมูลกับ RS-232 ต่อไป แสดงในรูปที่ 10



รูปที่ 10 แสดงโปรแกรมย่อยในการสื่อสาร

- โปรแกรมย่อยตัวนี้สามารถเลือกใช้โดยเรียกจาก Mainmenu หรือกดจากปุ่มบน Speedbar
- เมื่อโปรแกรมเปิดออกมาหลังจากการตอบรับ โปรแกรมก็จะทำการย้ายข้อมูลที่ต้องการส่งลงบนโปรแกมย่อยเอง
- เมื่อต่อสายการสื่อสารของ ฮาร์ดแวร์ของ พีแอลซี แล้วเลือกเมนู Send เพื่อส่งโปรแกรม เมื่อข้อมูลส่งลงเครื่อง พีแอลซี เรียบร้อยแล้ว ฮาร์ดแวร์จะทำการตอบรับการรับข้อมูลที่สมบูรณ์ โดยการส่งเสียงตอบรับ 2 ครั้ง เป็นอันสมบูรณ์

โปรแกรมหลัก




```
unit Main;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, Menus,
 StdCtrls, Dialogs, Buttons, Messages, ExtCtrls, ComCtrls, StdActns,
 ActnList, ToolWin, ImgList, Db, DBTables;

type
 TMainForm = class(TForm)
  MainMenu1: TMainMenu;
  File1: TMenuItem;
  FileNewItem: TMenuItem;
  FileOpenItem: TMenuItem;
  FileCloseItem: TMenuItem;
  Window1: TMenuItem;
  Help1: TMenuItem;
  N1: TMenuItem;
  FileExitItem: TMenuItem;
  WindowCascadeItem: TMenuItem;
  WindowTileItem: TMenuItem;
  WindowArrangeItem: TMenuItem;
  HelpAboutItem: TMenuItem;
  OpenDialog: TOpenDialog;
  FileSaveItem: TMenuItem;
  FileSaveAsItem: TMenuItem;
  Edit1: TMenuItem;
  CutItem: TMenuItem;
  CopyItem: TMenuItem;
```

```
    PasteItem: TMenuItem;
    WindowMinimizeItem: TMenuItem;
    StatusBar: TStatusBar;
    ActionList1: TActionList;
    EditCut1: TEditCut;
    EditCopy1: TEditCopy;
    EditPaste1: TEditPaste;
    FileNew1: TAction;
    FileSave1: TAction;
    FileOpen1: TAction;
    FileSaveAs1: TAction;
    WindowCascade1: TWindowCascade;
    WindowTileHorizontal1: TWindowTileHorizontal;
    WindowArrangeAll1: TWindowArrange;
    WindowMinimizeAll1: TWindowMinimizeAll;
    HelpAbout1: TAction;
    FileClose1: TWindowClose;
    WindowTileVertical1: TWindowTileVertical;
    WindowTileItem2: TMenuItem;
    ToolBar2: TToolBar;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    ToolButton4: TToolButton;
    ToolButton5: TToolButton;
    ToolButton6: TToolButton;
    ToolButton9: TToolButton;
    ToolButton7: TToolButton;
    ToolButton8: TToolButton;
    ToolButton10: TToolButton;
    ToolButton11: TToolButton;
    ImageList1: TImageList;
    SaveDialog1: TSaveDialog;
    Compile1: TMenuItem;
    Communication1: TMenuItem;
    Transfer1: TMenuItem;
    Compile3: TMenuItem;
    FontDialog1: TFontDialog;
    ToolButton12: TToolButton;
    ToolButton13: TToolButton;
    ToolButton14: TToolButton;
    Panel1: TPanel;
    PrinterSetupDialog1: TPrinterSetupDialog;
    FontSize: TEdit;
    FSize: TUpDown;
    N3: TMenuItem;
    Print1: TMenuItem;
    PrintSetup1: TMenuItem;
    FilePrint1: TAction;
    ToolButton15: TToolButton;
```

```
    ToolButton16: TToolButton;
    ToolButton17: TToolButton;
    ToolButton18: TToolButton;
    ToolButton19: TToolButton;
    N4: TMenuItem;
    ToolButton20: TToolButton;
    LadderDiagram1: TMenuItem;
    Image1: TImage;
    ToolButton21: TToolButton;
    ToolButton22: TToolButton;
    Refresh1: TMenuItem;
    procedure FileNew1Execute(Sender: TObject);
    procedure FileOpen1Execute(Sender: TObject);
    procedure HelpAbout1Execute(Sender: TObject);
    procedure FileExit1Execute(Sender: TObject);
    procedure FileSaveAs1Execute(Sender: TObject);
    procedure FileSave1Execute(Sender: TObject);
    procedure FileClose1Execute(Sender: TObject);
    procedure Transfer1Click(Sender: TObject);
    procedure Compile3Click(Sender: TObject);
    procedure FontSizeChange(Sender: TObject);
    procedure Panel1Click(Sender: TObject);
    procedure PrintSetup1Click(Sender: TObject);
    procedure FilePrint1Execute(Sender: TObject);
    procedure ToolButton16Click(Sender: TObject);
    procedure ToolButton19Click(Sender: TObject);
    procedure LadderDiagram1Click(Sender: TObject);
    procedure ToolButton21Click(Sender: TObject);
    procedure Refresh1Click(Sender: TObject);
  private
    { Private declarations }
    procedure CreateMDIChild(const Name: string);
  public
    { Public declarations }
  end;

var
  MainForm: TMainForm;

implementation

{$R *.DFM}

uses ChildWin, About, RS232, Printers;
////////////////////////////////////////////////////////////////////

procedure TMainForm.CreateMDIChild(const Name: string);
var
  Child: TMDIChild;
begin      // START
```

```
  { Create a new MDI child window }
  Child := TMDIChild.Create(Application);
  Child.Caption := Name;
  if FileExists(Name) then Child.Memo1.Lines.LoadFromFile(Name);
end;

procedure TMainForm.FileNew1Execute(Sender: TObject);
begin      // NEW
 if MDIChildCount<=1 then  begin
  CreateMDIChild('Boolean Instruction of PLC');
  SaveDialog1.Tag :=0;
  MainForm.ActiveMDIChild.WindowState:=WsMaximized;
  with  ActiveMDIChild as TMDIChild do
  begin FontDialog1.Font := Memo1.Font;
     FontSize.text:    = IntToStr(Memo1.Font.Size);
     Panel1.Hint:      = Memo1.Font.Name;
  end;
 end
end;

procedure TMainForm.FileOpen1Execute(Sender: TObject);
begin      // OPEN
 if OpenDialog.Execute then
 begin
  CreateMDIChild(OpenDialog.FileName);
  with  ActiveMDIChild as TMDIChild do
  begin Memo1.Lines.LoadFromFile(OpenDialog.FileName);
     Memo1.Font:=FontDialog1.Font; end;
  if MDIChildCount>1 then MainForm.Cascade;
  if MDIChildCount=1 then MainForm.ActiveMDIChild.WindowState:=WsMaximized;
  SaveDialog1.Tag :=2;
 end
end;

procedure TMainForm.HelpAbout1Execute(Sender: TObject);
begin      // HELP ABOUT
 AboutBox.ShowModal;
end;

procedure TMainForm.FileExit1Execute(Sender: TObject);
begin      // EXIT
 if SaveDialog1.Tag = 0 then
   FileSaveAs1Execute(Sender)
 else
   Close;
end;

procedure TMainForm.FileSaveAs1Execute(Sender: TObject);
begin      // SAVE AS
if MDIChildCount>=1 then
```

```
    begin
    if SaveDialog1.Execute then
      begin
      with  ActiveMDIChild as TMDIChild do
      begin Memo1.Lines.SaveToFile(ChangeFileExt(SaveDialog1.FileName,'.bak'));
          Memo1.Lines.SaveToFile(ChangeFileExt(SaveDialog1.FileName,'.txt'));
          Caption := ChangeFileExt(SaveDialog1.FileName,'.txt');
          end;
          SaveDialog1.Tag:=1;
      end
    end
    end;

    procedure TMainForm.FileSave1Execute(Sender: TObject);
    begin     // SAVE
    IF MDIChildCount>=1 then
    begin   If SaveDialog1.Tag = 0 then
        FileSaveAs1Execute(Sender)
      else If SaveDialog1.Tag = 1 then
        with ActiveMDIChild as TMDIChild do
        Memo1.Lines.SaveToFile(ChangeFileExt(SaveDialog1.FileName,'.txt'))
      else If SaveDialog1.Tag = 2 then
        with ActiveMDIChild as TMDIChild do
        Memo1.Lines.SaveToFile(Caption);
    end
    end;

    procedure TMainForm.FileClose1Execute(Sender: TObject);
    begin     // CLOSE
      if SaveDialog1.Tag = 0 then  begin
        FileSaveAs1Execute(Sender);
        MDIChildren[0].close;     end
      else
        MDIChildren[0].close;
    end;

    procedure TMainForm.PrintSetup1Click(Sender: TObject);
    begin     // PrinterSetup
      PrinterSetupDialog1.Execute;
    end;

    procedure TMainForm.FilePrint1Execute(Sender: TObject);
    var  Ptxt:TextFile;  Plong:LongInt;
    begin     // Print
      AssignPrn(Ptxt);
      Rewrite(Ptxt);
      with  MainForm.ActiveMDIChild as TMDIChild do
      begin Printer.Canvas.Font:=Memo1.Font;
          For Plong:=0 to Memo1.Lines.Count-1 do
          WriteLn (Ptxt,Memo1.Lines[plong]);  end;
```

```
    CloseFile(Ptxt);
  end;

  procedure TMainForm.FontSizeChange(Sender: TObject);
  begin    // Font.Size
    IF MDIChildCount>=1 then
    begin
      with MainForm.ActiveMDIChild as TMDIChild do
      Memo1.Font.Size:=FSize.Position;
      FontDialog1.Font.Size:=FSize.Position;
    end
    else
      FontSize.Text:=inttostr(FontDialog1.Font.Size);
  end;

  procedure TMainForm.Panel1Click(Sender: TObject);
  begin    // FontDialog
    IF MDIChildCount>=1 then
    begin
      if FontDialog1.Execute then
      begin
      with MainForm.ActiveMDIChild as TMDIChild do
      Memo1.Font:=FontDialog1.Font;
      FontSize.Text:=inttostr(FontDialog1.Font.Size);
      Panel1.Hint:=FontDialog1.Font.Name;
      end
    end
  end;


////////////////////////////////////////////////////////////////
//////////////  COMPILE F9  ////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
procedure TMainForm.Compile3Click(Sender: TObject);
const  PORT0    = 61440; {F000H}
       BUFF_IN  = 48640; {BE00H}
       BUFF_OUT = 48641;
       OUT_Y    = 48642;
       IMAGE_Y  = 48643;
       BUFF_M   = 48688; {BE30H}
       OUT_M    = 48689;
       IMAGE_M  = 48690;
       OUT_C    = 48704; {BE40H}
       IMAGE_C  = 48705;
       COUN_C0  = 48706;
       OUT_T    = 48736; {BE60H}
       COUN_TH0 = 48737;
       COUN_TL0 = 48738;
       OUT_T1   = 48753; {BE71H}
```

```
        COUN_TH10  = 48754;
        COUN_TL10  = 48755;
  var runs    :LongInt;
     Ch1, Ch2, Ch3, Ch4, Sp_word, Sp_KC, Num_C, Sp_KT, Num_T, Hex_Code,
     HIGH, LOW                              :String;
     posi, error_lines, TT, LineAll,
     Number, D_Text, M_Text, Addr, Positions, Reng        :Integer;
     Scan_Time  : Real;
     X : array[1..8] of Integer;
     Y : array[1..8] of Integer;
     M : array[0..3] of Integer;
  BEGIN
  IF MDIChildCount>=1 then
              begin
  runs     :=-1;
  error_lines:= 0;  TT:=0;  Scan_Time:=0;  M_Text:=0;  Addr:=0;
  Sp_word   :='downite';
  Sp_KC     :='space';  Sp_KT:='space';
  CreateMDIChild('Intel HEX File');
  with MainForm.MDIChildren[1] as TMDIChild do  LineAll:=Memo1.Lines.Count;
  with MainForm.MDIChildren[1] as TMDIChild do
    BEGIN
    REPEAT
     runs := runs+1;
     Ch1  := Memo1.Lines[runs];
     Posi := Pos(' ',ch1);
     If Posi=1 then
      begin
      Repeat
      Delete(Ch1,1,1);
      Posi:=Pos(' ',Ch1);
      Until (Posi>1) or (posi=0)
      end;

    If (Posi>1)or(Posi=0) then
     begin//***
     Ch4   := Copy(Ch1,1,3);
     Ch2   := Copy(Ch1,1,Posi-1);
     Delete(Ch1,1,Posi-1);
     Posi  := Pos(' ',Ch1);
     If Posi<>0 then
      begin
      Repeat
      Delete(Ch1,Posi,1);
      Posi:=Pos(' ',Ch1);     // ***End of Cut Word*** //
      Until Posi=0            // Ch4=Check Single word //
     end;                  // Ch2=Instruction SET.. //
    Ch3   :=Copy(Ch1,1,1);    // Ch3=Device..       //
    Delete(Ch1,1,1);          // Ch1=Number of Device. //
```

```
//_____ SCAN WORD _____
If Ch4='' then
// Skip to New line
Else
// END
If (Ch4='END')or(Ch4='end') Then
  begin
  with MainForm.MDIChildren[0] as TMDIChild do
     begin
     Hex_Code  := '';
  if Sp_KC='ok' then       begin  if Scan_Time<2762 then Begin
  Hex_Code := Hex_Code+'79FB7A04DAFED9FA';
  Scan_Time:=Scan_Time+2762; end;                    end;

  if Sp_KT='ok' then  begin
  Hex_Code := Hex_Code+'C2F0308D0C758C4B758AF9C28DD28CD2F0';
  Scan_Time:=Scan_Time+10;   end;

  For TT:=1 to 8 do
  If X[TT]=1 then       begin
  Memo1.Lines.Text := '90'+IntToHex(PORT0+TT,4)+'E090'+IntToHex(BUFF_IN+(4*TT),4)+
  'F0'+Memo1.Lines.Text;
  Scan_Time:=Scan_Time+8;
  X[TT]:=0; end;

  For TT:=1 to 8 do
  If Y[TT]=1 then       begin
  Memo1.Lines.Text := '90'+IntToHex(IMAGE_Y+(4*TT),4)+'E090'+IntToHex(BUFF_OUT+(4*TT),4)+
  'F0'+Memo1.Lines.Text+'90'+IntToHex(BUFF_OUT+(4*TT),4)+'E090'+IntToHex(OUT_Y+(4*TT),4)+
  'F090'+IntToHex(PORT0+TT,4)+'F0';
  Scan_Time:=Scan_Time+20;
  Y[TT]:=0; end;

  For TT:=0 to 3 do
  if M[TT]=1   then     begin
  Memo1.Lines.Text := '90'+IntToHex(IMAGE_M+(3*TT),4)+'E090'+IntToHex(BUFF_M+(3*TT),4)+
  'F0'+Memo1.Lines.Text+'90'+IntToHex(BUFF_M+(3*TT),4)+'E090'+IntToHex(OUT_M+(3*TT),4)+'F0';
  Scan_Time:=Scan_Time+16;
  M[TT]:=0; end;
  TT:=0;

     Memo1.Lines.Text := '7400789090BE00F0A3D8FCF5F0C28DC28C7589017'+
     '58C4B758AF9D28CA2905007C293D29202402EC292D29302401C90F000E090'+
     'BE00F090BE03E090BE01F0'+Memo1.Lines.Text;              {Head.}
     Hex_Code := Hex_Code+'90BE01E090BE02F090F000F002401C';    {Tail.}
     Memo1.Lines.Text := Memo1.Lines.Text + Hex_Code;
     Scan_Time:=(Scan_Time+41)*0.001085089;
//**********Intel HEX File**********
Number := Length(Memo1.Lines.Text);
D_Text := Number DIV 32;
```

```
  M_Text := Number MOD 32;
  Addr   := 16384;  {4000 H}
  Positions := 0;
   IF D_Text<>0 then
   Repeat
   Memo1.Lines.Add(':10'+IntToHex(Addr,4)+'00'+
              Copy(Memo1.Lines.Text,Positions+1,32));
   Positions := Positions + 32;
   Addr      := Addr + 16;
   D_Text    := D_Text-1;
   Until D_Text = 0;

   If M_Text<> 0 then
   M_Text := M_Text div 2;
   Memo1.Lines.Add(':'+IntToHex(M_Text,2)+IntToHex(Addr,4)+'00'+
              Copy(Memo1.Lines.Text,Positions+1,M_Text*2));
   Reng   := Pos(':',Memo1.Lines.Text);
   Number := Length(Memo1.Lines.Text);
   Memo1.Lines.Text := Copy(Memo1.Lines.Text,Reng,Number);
   Memo1.Lines.Add(':00000001FF');//********************Trl Hex File********
  end;
   runs   := LineAll;
   Sp_word := 'right';
   end
 Else
 // LD LDI AND ANI OR ORI , X Y M C T
 If (Ch2='LD')or(Ch2='ld')or(Ch2='LDI')or(Ch2='ldi')or(Ch2='AND')or(Ch2='and')or
   (Ch2='ANI')or(Ch2='ani')or(Ch2='OR')or(Ch2='or')or(Ch2='ORI')or(Ch2='ori')then Begin

 If (Ch3='X')or(Ch3='x')or(Ch3='Y')or(Ch3='y')or(Ch3='M')or(Ch3='m')or
   (Ch3='C')or(Ch3='c')or(Ch3='T')or(Ch3='t')                    then Begin

 If (((Ch3='X')or(Ch3='x'))and(StrToInt(Ch1)>=0)and(StrToInt(Ch1)<64)) or
   (((Ch3='Y')or(Ch3='y'))and(StrToInt(Ch1)>=0)and(StrToInt(Ch1)<64)) or
   (((Ch3='M')or(Ch3='m'))and(StrToInt(Ch1)>=0)and(StrToInt(Ch1)<32)) or
   (((Ch3='C')or(Ch3='c'))and(StrToInt(Ch1)>=0)and(StrToInt(Ch1)<16)) or
   (((Ch3='T')or(Ch3='t'))and(((StrToInt(Ch1)>=0)and(StrToInt(Ch1)<8))or
   ((StrToInt(Ch1)>=10)and(StrToInt(Ch1)<18))))                  then

 with MainForm.MDIChildren[0] as TMDIChild do
 Begin
   Hex_Code := '';
   Sp_word  := 'not_end';
   If ((Ch2='X')or(Ch3='x')) then
    Begin
    TT    := StrToInt(Ch1) DIV 8;
    Hex_Code:='90'+IntToHex(BUFF_IN+(4*TT),4);    {MOV DPTR,#BUFF_IN}
    Ch1   :=IntToStr(StrToInt(Ch1) MOD 8);
    If TT<>0 then X[TT]:=1;
   End else
```

```
If ((Ch3='Y')or(Ch3='y')) then
 Begin
 TT    := 4*(StrToInt(Ch1) DIV 8);
 Hex_Code:='90'+IntToHex(OUT_Y+TT,4);      {MOV DPTR,#OUT_Y}
 Ch1   :=IntToStr(StrToInt(Ch1) MOD 8);
 End else
If ((Ch3='M')or(Ch3='m')) then
 Begin
 TT    := 3*(StrToInt(Ch1) DIV 8);
 Hex_Code:='90'+IntToHex(OUT_M+TT,4);      {MOV DPTR,#OUT_M}
 Ch1   :=IntToStr(StrToInt(Ch1) MOD 8);
 End else
If ((Ch3='C')or(Ch3='c')) then
 Begin
 TT    :=10*(StrToInt(Ch1) DIV 8);
 Hex_Code:='90'+IntToHex(OUT_C+TT,4);      {MOV DPTR,#OUT_C}
 Ch1   :=IntToStr(StrToInt(Ch1) MOD 8);
 End else
If ((Ch3='T')or(Ch3='t'))and(StrToInt(Ch1)>=0)and(StrToInt(Ch1)<8) then
 Begin
 Hex_Code:='90'+IntToHex(OUT_T,4);         {MOV DPTR,#OUT_T}
 End else
If ((Ch3='T')or(Ch3='t'))and(StrToInt(Ch1)>=10)and(StrToInt(Ch1)<18) then
 Begin
 Hex_Code:='90'+IntToHex(OUT_T1,4);        {MOV DPTR,#OUT_T1}
 Ch1   :=IntToStr(StrToInt(Ch1)-10);
 End;

 Hex_Code:=Hex_Code+'E0';                  {MOVX A,@DPTR}

If (Ch2='LD')or(Ch2='ld')or(Ch2='LDI')or(Ch2='ldi')      then
 begin Hex_Code:=Hex_Code+'A2E';           {MOV C,ACC.} end
 else
If (((Ch2='AND')or(Ch2='and'))and(not((Ch3='X')or(Ch3='x')))) or
 (((Ch2='ANI')or(Ch2='ani'))and   ((Ch3='X')or(Ch3='x'))) then
 begin Hex_Code:=Hex_Code+'82E';           {ANL C,ACC.} end
 else
If (((Ch2='ANI')or(Ch2='ani'))and(not((Ch3='X')or(Ch3='x')))) or
 (((Ch2='AND')or(Ch2='and'))and   ((Ch3='X')or(Ch3='x'))) then
 begin Hex_Code:=Hex_Code+'B0E';           {ANL C, ACC.} end
 else
If ( (((Ch2='OR')or(Ch2='or')) and(not((Ch3='X')or(Ch3='x')))) or
 (((Ch2='ORI')or(Ch2='ori'))and   ((Ch3='X')or(Ch3='x'))) then
 begin Hex_Code:=Hex_Code+'72E';           {ORL C,ACC.} end
 else
If (((Ch2='ORI')or(Ch2='ori'))and(not((Ch3='X')or(Ch3='x')))) or
 (((Ch2='OR') or (Ch2='or'))and   ((Ch3='X')or(Ch3='x'))) then
 begin Hex_Code:=Hex_Code+'A0E';           {ORL C,/ACC.} end

 Hex_Code:=Hex_Code+Ch1;                   {bit}
```

```
    If ( ((Ch2='LDI')or(Ch2='ldi'))and(not((Ch3='X')or(Ch3='x'))) ) or
      ( ((Ch2='LD')or(Ch2='ld'))and((Ch3='X')or(Ch3='x')) )      then
    Hex_Code:=Hex_Code+'B3';                    {CPL C}
    Memo1.Lines.Text := Memo1.Lines.Text+Hex_Code;
    Scan_Time:=Scan_Time+6;
   End
   Else
    Begin
    error_lines := runs;              //Display Error in NUMBER
    runs      := LineAll;             //
    Sp_word   := 'nonumber';          //
    End;                                          End
    Else
     Begin
     error_lines := runs;             //Display Error in DEVICE
     runs      := LineAll;            //
     Sp_word   := 'nodevice';         //
     End;                                         End
 Else
 // OUT SET RST for Y M
 If ((Ch2='OUT')or(Ch2='out')or(Ch2='SET')or(Ch2='set')or(Ch2='RST')   or
   (Ch2='rst'))and((Ch3='Y')or(Ch3='y')or(Ch3='M')or(Ch3='m'))      then Begin
 If (((Ch3='Y')or(Ch3='y'))and(StrToInt(Ch1)>=0)and(StrToInt(Ch1)<64)) or
  (((Ch3='M')or(Ch3='m'))and(StrToInt(Ch1)>=0)and(StrToInt(Ch1)<32)) then
  with MainForm.MDIChildren[0] as TMDIChild do
  Begin
  Hex_Code  := '';
  If ((Ch2='OUT')or(Ch2='out'))and((Ch3='Y')or(Ch3='y')) then  Begin
    TT := StrToInt(Ch1) Div 8;
    Hex_Code := '500790'+IntToHex(BUFF_OUT+(4*TT),4)+'E0' {MOV DPTR,#BUFF_OUT}
   End  else
  If ((Ch2='OUT')or(Ch2='out'))and((Ch3='M')or(Ch3='m')) then  Begin
   TT := StrToInt(Ch1) Div 8;
   Hex_Code := '500790'+IntToHex(BUFF_M+(3*TT),4)+'E0';  {MOV DPTR,#BUFF_M}
   End  else
 If ((Ch2='SET')or(Ch2='set')or(Ch2='RST')or(Ch2='rst'))
   and((Ch3='Y')or(Ch3='y')) then                    Begin
   TT := StrToInt(Ch1) Div 8;
   Hex_Code := '500790'+IntToHex(IMAGE_Y+(4*TT),4)+'E0'  {MOV DPTR,#IMAGE_Y}
   End  else
 If ((Ch2='SET')or(Ch2='set')or(Ch2='RST')or(Ch2='rst'))
   and((Ch3='M')or(Ch3='m')) then                    Begin
   TT := StrToInt(Ch1) Div 8;
   Hex_Code := '500790'+IntToHex(IMAGE_M+(3*TT),4)+'E0'; {MOV DPTR,#IMAGE_M}
   End;

 If ((Ch2='OUT')or(Ch2='out')or(Ch2='SET')or(Ch2='set'))
   and((Ch3='Y')or(Ch3='y')) then   If TT<>0 then Y[TT]:=1;

 If ((Ch2='OUT')or(Ch2='out')or(Ch2='SET')or(Ch2='set'))
```

```
      and((Ch3='M')or(Ch3='m')) then          M[TT]:=1;

     If ((Ch2='OUT')or(Ch2='out')or(Ch2='SET')or(Ch2='set'))
      and((Ch3='Y')or(Ch3='y')or(Ch3='M')or(Ch3='m')) then Begin
      Ch1 := IntToStr(StrToInt(Ch1) Mod 8);
      Hex_Code := Hex_Code+'D2E'+Ch1+'F0';
      End                                {SETB ACC.}
      else
    If ((Ch2='RST')or(Ch2='rst'))and((Ch3='Y')or(Ch3='y')or(Ch3='M')or(Ch3='m'))
      then Begin
      Ch1 := IntToStr(StrToInt(Ch1) Mod 8);
      Hex_Code := Hex_Code+'C2E'+Ch1+'F0';
      End;                               {CLR ACC.}

    Memo1.Lines.Text := Memo1.Lines.Text+Hex_Code;
    Scan_Time:=Scan_Time+9;
    Sp_word       := 'not_end';
    End
    Else
    Begin
     error_lines := runs;          //Display Error in NUMBER
     runs       := LineAll;        //
     Sp_word    := 'nonumber';     //
    End;                                End
  Else
  // OUT C(0-15)
  If ((Ch2='OUT')or(Ch2='out'))and((Ch3='C')or(Ch3='c')) then  Begin
  If (StrToInt(Ch1)>=0)and(StrToInt(Ch1)<16) then
    Begin
    Sp_KC      := 'count';
    Num_C      := Ch1;
    Sp_word    := 'XXX';
    error_lines := runs;
    End
    Else
     Begin
     error_lines := runs;          //Display Error in NUMBER
     runs       := LineAll;        //
     Sp_word    := 'nonumber';     //
     End;                               End
Else
// SP K(for COUNTER 0-15)
If ((Ch2='SP')or(Ch2='sp'))and(Sp_KC='count') then
  if (Ch3='K')or(Ch3='k') then
 begin
   if ch1='' then
     begin
     error_lines := runs;
     runs       := LineAll;
     Sp_word    := 'nosp';
```

```
        end
       else
       if (StrToInt(ch1)<=255)and(StrToInt(ch1)>0) then
        with MainForm.MDIChildren[0] as TMDIChild do
        Begin
        Hex_Code  := '';
        TT        := 10*(StrToInt(Num_C) DIV 8);
        Num_C     := IntToStr(StrToInt(Num_C) MOD 8);
        Hex_Code  := '501D90'+IntToHex((IMAGE_C+TT),4)+'E092F1B0E'+ Num_C +
    '501190'+IntToHex((Coun_C0+TT+StrToInt(Num_C)),4)+'E005E0F0B4'+
    IntToHex(StrToInt(Ch1),2)+'0790'+IntToHex((OUT_C+TT),4)+'E0D2E'+Num_C+
    'F0A2F190'+IntToHex((IMAGE_C+TT),4)+'E092E'+Num_C+'F0';
        Memo1.Lines.Text := Memo1.Lines.Text+Hex_Code;
        Scan_Time:=Scan_Time+37;
        Sp_KC     := 'ok';
        Sp_word   := 'not_end';
        End
       else
       begin
       error_lines := runs;
       runs       := LineAll;
       Sp_word    := 'nosp';
       end
      end
      else
      begin
      error_lines := runs;
      runs       := LineAll;
      Sp_word    := 'noinstruc';
    End
    Else
    // RST C(0-15)
    If ((Ch2='RST')or(Ch2='rst'))and((Ch3='C')or(Ch3='c')) then  Begin
    If (StrToInt(Ch1)>=0)and(StrToInt(Ch1)<16) then
    with MainForm.MDIChildren[0] as TMDIChild do
      Begin
      Hex_Code  := '';
      TT        := 10*(StrToInt(Ch1) DIV 8);
      Ch1       := IntToStr(StrToInt(Ch1) MOD 8);
      Hex_Code  := '500D90'+IntToHex((COUN_C0+TT+StrToInt(Ch1)),4)+'7400F090'+
              IntToHex(OUT_C+TT,4)+'E0C2E'+Ch1+'F0';
      Sp_word   := 'not_end';
      Memo1.Lines.Text := Memo1.Lines.Text+Hex_Code;
      Scan_Time:=Scan_Time+14;
      End
      Else
       Begin
      error_lines := runs;            //Display Error in NUMBER
      runs       := LineAll;          //
      Sp_word    := 'nonumber';       //
```

```
        End;                                    End
      Else
      // OUT T(0-7)    //OUT T(10-17)
      If ((Ch2='OUT')or(Ch2='out'))and((Ch3='T')or(Ch3='t')) then  Begin
      If ((StrToInt(Ch1)>=0)and(StrToInt(Ch1)<8))  or
        ((StrToInt(Ch1)>=10)and(StrToInt(Ch1)<18)) then
        Begin
        If (StrToInt(Ch1)>=0) and(StrToInt(Ch1)<8)  then Sp_KT:='time';
        If (StrToInt(Ch1)>=10)and(StrToInt(Ch1)<18) then Sp_KT:='store';
        Num_T     := Ch1;
        Sp_word    := 'XXX';
        error_lines := runs;
        End
        Else
        Begin
        error_lines := runs;            //Display Error in NUMBER
        runs       := LineAll;          //
        Sp_word    := 'nonumber';       //
        End;                                    End
    Else
    //  SP K(for TIMER 0-7)
    If ((Ch2='SP')or(Ch2='sp'))and(Sp_KT='time') then
     if (Ch3='K')or(Ch3='k') then
     begin
      if ch1='' then
        begin
        error_lines := runs;
        runs       := LineAll;
        Sp_word    := 'nosp';
        end
      else
      if (StrToInt(ch1)<=32767)and(StrToInt(ch1)>0) then  //0.1-3276.7 Sec.
       with MainForm.MDIChildren[0] as TMDIChild do      //Max 54H 36.7Sec.
       Begin
       TT      := StrToInt(Ch1)*2;
       Hex_Code  := '';
       LOW      := IntToHex(COUN_TL0+(2*StrToInt(Num_T)),4);
       HIGH     := IntToHex(COUN_TH0+(2*StrToInt(Num_T)),4);
      Hex_Code  := '4013740090'+LOW+'F090'+HIGH+'F090'+IntToHex(OUT_T,4)+
'E0C2E'+Num_T+'F0502630F02390'+LOW+'E004F0700690'+HIGH+'E004F090'+HIGH+'E0B4'+
IntToHex(TT DIV 256,2)+'0E90'+LOW+'E0B4'+IntToHex(TT MOD 256,2)+'0790'+
IntToHex(OUT_T,4)+'E0D2E'+Num_T+'F0';
      Memo1.Lines.Text := Memo1.Lines.Text + Hex_Code;
      Scan_Time:=Scan_Time+57;
      Sp_KT     := 'ok';
      Sp_word   := 'not_end';
      End
    else
    begin
    error_lines := runs;
```

```
          runs      := LineAll;
          Sp_word   := 'nosp';
          end
         end
         else
         begin
         error_lines := runs;
         runs      := LineAll;
         Sp_word   := 'noinstruc';
       End
       Else
       // SP K (for TIMER 10-17)
       If ((Ch2='SP')or(Ch2='sp'))and(Sp_KT='store') then
        if (Ch3='K')or(Ch3='k') then
        begin
         if ch1='' then
          begin
          error_lines := runs;
          runs      := LineAll;
          Sp_word   := 'nosp';
          end
        else
         if (StrToInt(ch1)<=32767)and(StrToInt(ch1)>0) then  //0.1-3276.7 Sec.
          with MainForm.MDIChildren[0] as TMDIChild do     //Max 54H 36.7Sec
          Begin
          TT      := StrToInt(Ch1)*2;                 //OUT T
          Hex_Code := '';                             ///(for TIMER 10-17)
          LOW     := IntToHex(COUN_TL10+(2*(StrToInt(Num_T)-10)),4);
          HIGH    := IntToHex(COUN_TH10+(2*(StrToInt(Num_T)-10)),4);
          Hex_Code := '502630F02390'+LOW+'E004F0700690'+HIGH+'E004F090'+HIGH+
'E0B4'+IntToHex(TT DIV 256,2)+'0E90'+LOW+'E0B4'+IntToHex(TT MOD 256,2)+'0790'+
IntToHex(OUT_T1,4)+'E0D2E'+IntToStr(StrToInt(Num_T)-10)+'F0';
          Memo1.Lines.Text := Memo1.Lines.Text + Hex_Code;
          Scan_Time:=Scan_Time+39;
          Sp_KT    := 'ok';
          Sp_word  := 'not_end';
          End
        else
        begin
        error_lines := runs;
        runs      := LineAll;
        Sp_word   := 'nosp';
        end
      end
      else
      begin
      error_lines := runs;
      runs      := LineAll;
      Sp_word   := 'noinstruc';
    End
```

```
  Else
  // RST T (10-17:Store)
  if ((Ch2='RST')or(Ch2='rst'))and((Ch3='T')or(Ch3='t')) then  Begin
  If (StrToInt(Ch1)>=10)and(StrToInt(Ch1)<18) then
  with MainForm.MDIChildren[0] as TMDIChild do
    Begin                                  //RST T
    Hex_Code := '';                        //(for TIMER 10-17)
    LOW     := IntToHex(COUN_TL10+(2*(StrToInt(Ch1)-10)),4);
    HIGH    := IntToHex(COUN_TH10+(2*(StrToInt(Ch1)-10)),4);
    Hex_Code := '5011740090'+LOW+'F090'+HIGH+'F090'+IntToHex(OUT_T1,4)+
'E0C2E'+IntToStr(StrToInt(Ch1)-10)-'F0';
    Memo1.Lines.Text := Memo1.Lines.Text + Hex_Code;
    Scan_Time:=Scan_Time+20;
    Sp_word  := 'not_end';
    End
    Else
     Begin
     error_lines := runs;          //Display Error in NUMBER
     runs      = LineAll;          //
     Sp_word   = 'nonumber';
     End;                                  End
  Else
  // < NEW INSTRUCTION  if ... end else
  Begin
  error_lines := runs;          //Display Error in INSTRUCTION SET
  runs      := LineAll;
  Sp_word   := 'noinstruc';
  End;
  //_____ SCAN WORD _____
       end;//***
    UNTIL runs>=LineAll
    END;
////////////////////////////////////////////////////////////////////
if Sp_word='not_end' then
   begin
   beep;
   MessageDlg('โปรดตรวจสอบการใช้ END',mtWarning,[mbok],0);
   MDIChildren[0].close;
   end
   else
if Sp_word='noinstruc' then
   begin
   beep;
   MessageDlg('โปรดตรวจสอบคำสั่งที่ Line '+inttostr(error_lines+1),
    mtWarning,[mbok],0);
   MDIChildren[0].close;
   end
   else
if Sp_word='nodevice' then
   begin
```

```
          beep;
          MessageDlg('โปรดตรวจสอบการใช้อุปกรณ์ที่ Line '+inttostr(error_lines+1),
           mtWarning,[mbok],0);
          MDIChildren[0].close;
         end
         else
       if Sp_word='nosp' then
         begin
         beep;
         MessageDlg('ตรวจสอบค่า K ที่ Line '+inttostr(error_lines+1)
          mtWarning,[mbok],0);
         MDIChildren[0].close;
         end
         else
     if Sp_word='nonumber' then
       begin
       beep;
       MessageDlg('หมายเลขอุปกรณ์ไม่ถูกต้อง โปรดแก้ไขที่ Line '+inttostr(error_lines+1),
        mtWarning,[mbok],0);
       MDIChildren[0].close;
       end
       else
   if Sp_word='downte' then
     begin
     beep;
     MessageDlg('คุณยังไม่ได้เขียนโปรแกรมเลย!',mtInformation,[mbok],0);
     MDIChildren[0].close;
     end
     else
  if (Sp_KC='count')or(Sp_KT='time')or(Sp_KT='store') then
    begin
    beep;
    MessageDlg(' SP Line '+inttostr(error_lines+1),mtWarning,[mbok],0);
    MDIChildren[0].close;
    end
    else
if Sp_word='right' then    begin
  MainForm.Cascade;
CASE MessageDlg(
   'Scan time = '+CurrToStr(Scan_Time)+' mSec. '+
   '/ จำนวนข้อมูล '+IntToStr((Addr+M_Text)-16384)+' bytes '+
   'โปรแกรมเสร็จสิ้นการ compile คุณต้องการ save เป็นสกุล HEX หรือไม่?'
   ,mtConfirmation,mbOkCancel,0) Of
mrOk:
if SaveDialog1.Execute then
with ActiveMDIChild as TMDIChild do
begin
Memo1.Lines.SaveToFile(ChangeFileExt(SaveDialog1.FileName,'.HEX'));
Caption := ChangeFileExt(SaveDialog1.FileName,'.HEX');
Case MessageDlg('ต้องส่งไปที่ RAM ของ PLC หรือไม่'
```

```
        ,mtConfirmation,mbOkCancel,0) Of
       mrOk:
       begin
       //MainForm.MDIChildren[0].Close;
       Form1.Memo1.Lines.LoadFromFile(ChangeFileExt(SaveDialog1.FileName,'.HEX'));
       Form1.Caption:=ChangeFileExt(SaveDialog1.FileName,'.HEX');
       Form1.Show;
       end; end; {end of case}
      end;
      END; {end of CASE}      end
              END
  END;//THE END procedure TMainForm.Compile3Click(Sender: TObject);
  ////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////

  procedure TMainForm.Transfer1Click(Sender: TObject);
  begin   Form1.Show;
  end;

  procedure TMainForm.ToolButton16Click(Sender: TObject);
  begin
     if Form1.Visible=False then
     Form1.Visible:=True  else
     Form1.Visible:=False
  end;

  procedure TMainForm.ToolButton19Click(Sender: TObject);
  begin   Compile3Click(Sender);
  end;

  procedure TMainForm.LadderDiagram1Click(Sender: TObject);
  begin
     If Image1.Visible=False then
     begin
     Image1.Visible:=True;
     Image1.Canvas.TextOut(0,0,'Ladder Diagram');
     end  else
     Image1.Visible:=False;
  end;

  procedure TMainForm.ToolButton21Click(Sender: TObject);
  begin
     LadderDiagram1Click(Sender);
  end;

  ////////////////////////////////////////////////////////////
  //////////////////////          //////////////////////////
```

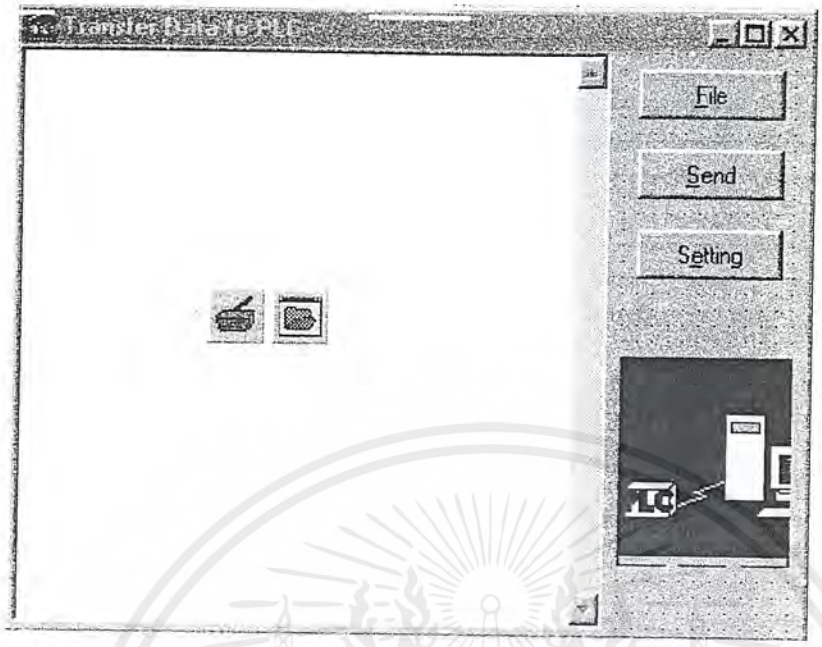
```
//////////////////////// Ladder Diagram ////////////////////////////
//////////////////////                 ////////////////////////////
////////////////////////////////////////////////////////////////////
procedure TMainForm.Refresh1Click(Sender: TObject);
var runs    :LongInt;
    Ch1,Ch2,Ch3,Ch4,Sp,Divi,Num     :String;
    posi, LineAll, cox, coy, Count_or, ory, Count_and, Out_state :Integer;
    Bitmap  : TBitmap;
    Point   : array[0..1] of TPoint;
    Point1  : array[0..1] of TPoint;
BEGIN
IF (MDIChildCount>=1)and(Image1.Visible=True) then
            begin
Bitmap := TBitmap.Create;
runs   := -1;  Out_state:=0;
cox    := 5;   coy := -12;  Count_or := 0;  Count_and := 0;
   Bitmap.LoadFromFile('C:\PLC\P\CLS.bmp');
   Image1.Canvas.Draw(0,15,Bitmap);
with MainForm.ActiveMDIChild as TMDIChild do  LineAll:=Memo1.Lines.Count;
with MainForm.ActiveMDIChild as TMDIChild do
 BEGIN
 REPEAT
    runs := runs+1;
    Ch1  := Memo1.Lines[runs];
    Posi := Pos(' ',ch1);
    If Posi=1 then
     begin
     Repeat
     Delete(Ch1,1,1);
     Posi:=Pos(' ',Ch1);
     Until (Posi>1) or (posi=0)
     end;
   If (Posi>1)or(Posi=0) then
     begin//***
     Ch4   := Copy(Ch1,1,3);
     Ch2   := Copy(Ch1,1,Posi-1);
     Delete(Ch1,1,Posi-1);
     Posi  := Pos(' ',ch1);
     If Posi<>0 then
      begin
      Repeat
      Delete(Ch1,Posi,1);
      Posi:=Pos(' ',Ch1);    // ***End of Cut Word*** //
      Until Posi=0          // Ch4=Check Single word //
      end;                  // Ch2=Instruction SET.. //
    Ch3   :=Copy(ch1,1,1);   // Ch3=Device..        //
    Delete(Ch1,1,1);         // Ch1=Number of Device. //

//_________ SCAN WORD ____________________________________________
```

```
      If Ch4='' then
      // Skip to New line
      Else
      // END
      If (Ch4='END')or(Ch4='end') Then
        begin
        Bitmap.LoadFromFile('C:\PLC\P\END.bmp');
        If (Out_state=0) then  coy:=coy+30;
        Image1.Canvas.Draw(335,coy-5,Bitmap);
        Point1[0].X:=5; Point1[0].Y:=coy+9; Point1[1].X:=335; Point1[1].Y:=coy+9;
        Image1.Canvas.Polyline(Point1);
        runs   := LineAll;
        end
      Else
      // LD LDI AND ANI OR ORI , X Y M C T
      If (Ch2='LD')or(Ch2='ld')  then          begin
        If (Out_state<>0) then  coy:=coy-30;
        cox:=5; coy:=coy+30; Count_or:=1; Count_and:=1; Out_state:=0;
        Bitmap.LoadFromFile('C:\PLC\P\NO.bmp');
        Image1.Canvas.Draw(cox,coy,Bitmap);
        Image1.Canvas.TextOut(cox+7,coy+17,Ch3+' '+Ch1);
        Point[0].X:=cox+29; Point[0].Y:=coy+9; end
      Else
      If (Ch2='LDI')or(Ch2='ldi') then          begin
        If (Out_state<>0) then  coy:=coy-30;
        cox:=5; coy:=coy+30; Count_or:=1; Count_and:=1; Out_state:=0;
        Bitmap.LoadFromFile('C:\PLC\P\NC.bmp');
        Image1.Canvas.Draw(cox,coy,Bitmap);
        Image1.Canvas.TextOut(cox+7,coy+17,Ch3+' '+Ch1);
        Point[0].X:=cox+29; Point[0].Y:=coy+9; end
      Else
      If (Ch2='AND')or(Ch2='and') then          begin
        cox:=cox+30;    Count_and := Count_and+1;
        Bitmap.LoadFromFile('C:\PLC\P\NO.bmp');
        If Count_or>1 then ory :=coy-(30*(Count_or-1)) else ory:=coy;
        Image1.Canvas.Draw(cox,ory,Bitmap);
        Image1.Canvas.TextOut(cox+7,ory+17,Ch3+' '+Ch1);
        Point[0].X:=cox+29; Point[0].Y:=ory+9;
        If Out_state=1 then begin
        Point1[0].X:=cox-1; Point1[0].Y:=ory-21;
        Point1[1].X:=cox-1; Point1[1].Y:=ory+9;
        Image1.Canvas.Polyline(Point1);
        Out_state:=0;
      end  end
    Else
    If (Ch2='ANI')or(Ch2='ani') then          begin
      cox:=cox+30;    Count_and := Count_and+1;
      Bitmap.LoadFromFile('C:\PLC\P\NC.bmp');
      If Count_or>1 then ory:=coy-(30*(Count_or-1)) else ory:=coy;
      Image1.Canvas.Draw(cox,ory,Bitmap);
```

```
        Image1.Canvas.TextOut(cox+7,ory+17,Ch3+' '+Ch1);
        Point[0].X:=cox+29; Point[0].Y:=ory+9;
        If Out_state=1 then begin
         Point1[0].X:=cox-1; Point1[0].Y:=ory-21;
         Point1[1].X:=cox-1; Point1[1].Y:=ory+9;
         Image1.Canvas.Polyline(Point1);
         Out_state:=0;
        end end
      Else
      If (Ch2='OR')or(Ch2='or') then          begin
       coy:=coy+30;   Count_or:=Count_or+1;
       Bitmap.LoadFromFile('C:\PLC\P\NO.bmp');
       Image1.Canvas.Draw(cox,coy,Bitmap);
       Image1.Canvas.TextOut(cox+7,coy+17,Ch3+' '+Ch1);
       Point[1].X:=cox+29; Point[1].Y:=coy+10;
       Image1.Canvas.Polyline(Point);
       If Count_and>1 then          begin
       Point1[0].X:=5; Point1[0].Y:=coy-9; Point1[1].X:=cox; Point1[1].Y:=coy+9;
       Image1.Canvas.Polyline(Point1); end end
     Else
     If (Ch2='ORI')or(Ch2='ori') then          begin
      coy:=coy+30;   Count_or:=Count_or+1;
      Bitmap.LoadFromFile('C:\PLC\P\NC.bmp');
      Image1.Canvas.Draw(cox,coy,Bitmap);
      Image1.Canvas.TextOut(cox+7,coy+17,Ch3+' '+Ch1);
      Point[1].X:=cox+29; Point[1].Y:=coy+10;
      Image1.Canvas.Polyline(Point);
      If Count_and>1 then          begin
      Point1[0].X:=5; Point1[0].Y:=coy-9; Point1[1].X:=cox; Point1[1].Y:=coy+9;
      Image1.Canvas.Polyline(Point1); end end
    Else
      If ((Ch2='OUT')or(Ch2='out')) and ((Ch3='Y')or(Ch3='y')or
       (Ch3='M')or(Ch3='m')) Then          begin
      If Count_or>1 then ory:=coy-(30*(Count_or-1)) else ory:=coy;
      Bitmap.LoadFromFile('C:\PLC\P\OUT.bmp');
      Image1.Canvas.Draw(365,ory-5,Bitmap);
      Image1.Canvas.TextOut(390,ory-2,Ch3+' '+Ch1);
      Point[1].X:=365; Point[1].Y:=Point[0].Y;
      If Out_state=1 then  begin Point[0].X:=365; Point[0].Y:=Point[0].Y+30; end;
      Image1.Canvas.Polyline(Point);
      Out_state:=1;
      coy:=coy+30;       end
   Else
   If ((Ch2='OUT')or(Ch2='out'))and((Ch3='C')or(Ch3='c')or(Ch3='T')or(Ch3='t')) then
     Begin
     Sp    := 'ok';
     Divi   := Ch3;
     Num    := Ch1;
     End
  Else
```

```
    If ((Ch2='SP')or(Ch2='sp'))and((Ch3='K')or(Ch3='k'))and(Sp='ok') then
      begin
      If Count_or>1 then ory:=coy-(30*(Count_or-1)) else ory:=coy;
      Bitmap.LoadFromFile('C:\PLC\P\OUT.bmp');
      Image1.Canvas.Draw(365,ory-5,Bitmap);
      Image1.Canvas.TextOut(390,ory+2,Divi+' '+Num);
      Image1.Canvas.TextOut(415,ory+10,Ch3+' '+Ch1);
      Point[1].X:=365; Point[1].Y:=Point[0].Y;
      If Out_state=1 then  begin Point[0].X:=365; Point[0].Y:=Point[0].Y+30; end;
      Image1.Canvas.Polyline(Point);
      Out_state:=1;
      coy:=coy-30;
      Sp:=''          end
    Else
    If ( ((Ch2='SET')or(Ch2='set')or(Ch2='RST')or(Ch2='rst'))  and
       ((Ch3='Y')or(Ch3='y')or(Ch3='M')or(Ch3='m')) )      or
       ( ((Ch2='RST')or(Ch2='rst'))and( (Ch3='T')or(Ch3='t')
       or(Ch3='C')or(Ch3='c') ) ) then
       begin
       If Count_or>1 then ory:=coy-(30*(Count_or-1)) else ory:=coy;
       Bitmap.LoadFromFile('C:\PLC\P\Block.bmp');
       Image1.Canvas.Draw(335,ory-5,Bitmap);
       Image1.Canvas.TextOut(355,ory+2,Ch2);
       Image1.Canvas.TextOut(393,ory+2,Ch3+' '+Ch1);
       Point[1].X:=335; Point[1].Y:=Point[0].Y;
       If Out_state=1 then  begin Point[0].X:=335; Point[0].Y:=Point[0].Y+30; end;
       Image1.Canvas.Polyline(Point);
       Out_state:=1;
       coy:=coy-30;
       end;


       end
     UNTIL runs>=LineAll;
  Point[0].X:=5; Point[0].Y:=12; Point[1].X:=5; Point[1].Y:=490;
  Image1.Canvas.Polyline(Point);
  Point[0].X:=434; Point[0].Y:=12; Point[1].X:=434; Point[1].Y:=490;
  Image1.Canvas.Polyline(Point);
   end;
  end
 end;
 ////////////////////////////////////////////////////////////
 ////////////////////  THE END  /////////////////////////////
 ////////////////////  Ladder Diagram ///////////////////////
 ////////////////////  ////////////////////////////////////
 ////////////////////  ////////////////////////////////////

 end.  //THE END Prg. PLC Compile
 ////////////////////  ////////////////////////////////////
```

โปรแกรมย่อยการสื่อสาร RS-232




```
unit RS232;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Comport, Comms;

type
  TForm1 = class(TForm)
    ButtonFile: TButton;
    ButtonSend: TButton;
    ButtonSetting: TButton;
    ComPort1: TComPort;
    OpenDialog1: TOpenDialog;
    Memo1: TMemo;
    Image1: TImage;
    procedure ButtonFileClick(Sender: TObject);
    procedure ButtonSettingClick(Sender: TObject);
    procedure ButtonSendClick(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
```

```
implementation

{$R *.DFM}

procedure TForm1.ButtonFileClick(Sender: TObject);
begin
   If OpenDialog1.Execute then      begin
   Form1.Memo1.Lines.LoadFromFile(OpenDialog1.FileName);
   Form1.Caption:=OpenDialog1.FileName; end
end;

procedure TForm1.ButtonSettingClick(Sender: TObject);
begin
   ComPort1.ShowPropForm;
end;

procedure TForm1.ButtonSendClick(Sender: TObject);
begin
   ComPort1.Open;
   If ComPort1.Connected then
   begin
   Form1.ButtonSend.Caption:='Connect..';
   ComPort1.WriteString(Form1.Memo1.Text,true);
   end;
   ComPort1.Close;
   Form1.ButtonSend.Caption:='&Send';
end;

procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
   Form1.Memo1.Clear;
   Form1.Caption:='Transfer Data to PLC';
end;

end.
```

โปรแกรมมอนิเตอร์ในหน่วยความจำหลักของไมโครคอนโทรลเลอร์

```
RAM     EQU   4000H
BAUD96  EQU   0FDH
        ORG   0000H
        ACALL BEEP
        SJMP  START
        ORG   23H
        ACALL HEAD
        ORG   0040H
START:  MOV   SCON,#01010000B
        MOV   IE,#10010000B
        MOV   TMOD,#00101100B
        MOV   TH1,#BAUD96
        SETB  TR1
LOOPA:  MOV   C,P1.0
        JNC   LOOPB
        CLR   P1.3
        SETB  P1.2
        JMP   LOOPC
LOOPB:  CLR   P1.2
        SETB  P1.3
        SJMP  LOOPA
LOOPC:  LJMP  RAM
        RET
HEAD:   JNB   RI,$
        CLR   RI
        MOV   A,SBUF
        CJNE  A,#':',HEAD
        ACALL READ_HEX
        MOV   R0,A
        CJNE  R0,#0,ADDR_TO
        ACALL READ_HEX
        ACALL READ_HEX
        ACALL READ_HEX
        ACALL READ_HEX
```

```
                ACALL  BEEP   ;***
                ACALL  DELAY  ;***
                ACALL  BEEP   ;***
                ACALL  DELAY  ;***
                SJMP   START
                RETI
ADDR_TO:        ACALL  READ_HEX
                MOV    DPH,A
                ACALL  READ_HEX
                MOV    DPL,A
                ACALL  READ_HEX
D_MOVE:         ACALL  READ_HEX
                MOVX   @DPTR,A
                INC    DPTR
                DJNZ   R0,D_MOVE
                SJMP   HEAD
READ_HEX:       MOV    R7,#1H
AGAIN:          JNB    RI,$
                CLR    RI
                MOV    A,SBUF
                CLR    C
                SUBB   A,#40H
                JC     NA_F
                ADD    A,#9H
NA_F:           ANL    A,#0FH
                CJNE   R7,#0,SWAPP
                ORL    A,R6
                RET
SWAPP:          SWAP   A
                MOV    R6,A
                DEC    R7
                AJMP   AGAIN
                ;
BEEP:           PUSH   ACC
                PUSH   PSW
```

```
        MOV   TMOD,#01H
        MOV   R0,0FFH
BEEP_2: MOV   TH0,#0FFH ;2KHz
        MOV   TL0,#01AH ;
        SETB  TR0
        JNB   TF0,$
        CLR   TR0
        CLR   TF0
        CPL   P1.1
        DJNZ  R0,BEEP_2
        CLR   P1.1
        POP   PSW
        POP   PSW
        RET
        ;
DELAY:  PUSH  ACC
        PUSH  PSW
        MOV   R1,#0FFH
DLAY2:  MOV   R2,#0F0H
        DJNZ  R2,$
        DJNZ  R1,DLAY2
        POP   PSW
        POP   ACC
        RET
        ;
        END
```

กิตติกรรมประกาศ

แด่ พ่อ และ แม่ ที่เป็นทุกอย่างสำหรับลูก การสนับสนุนที่ดีในทิศทางที่ผมเป็นคนเลือกเอง และ ครอบครัวของผม น้าภา และ พี่คาว

ขอขอบคุณ อาจารย์ชินภัทร นันทจิวากรชัย อาจารย์ที่ปรึกษาในเทอมที่ 1 ของการทำโครงงานและ เป็นที่ปรึกษาตลอดทั้งหมด แม้ว่าทานจะเดนทางไปทำงานวิจัยที่ประเทศญี่ปุ่นในเทอมที่ 2 ก็ยังมีคำแนะนำผ่านระบบอินเตอร์เนต ประสบการณ์และความรู้ด้านพีแอลซีของท่าน เป็นส่วนสิ่งที่ขาดไม่ได้เลยในการทำปริญญานิพนธ์ และอุปกรณ์เครื่องมือที่ท่านให้ยืม ช่วยให้การทำงานประสบผลสำเร็จได้รวดเร็ว

ขอขอบคุณ อาจารย์สุรพันธ์ เอื้อไพบูลย์ อาจารย์ที่ปรึกษาในเทอมที่ 2 ท่านดูแลแทนอาจารย์ชินภัทร ทุกอย่างที่ท่านให้ช่วยให้ปริญญานิพนธ์นี้สำเร็จลุล่วงไปด้วยดี

ขอขอบใจ น้องกาญ สำหรับกำลังใจ ความห่วงใย ที่มีให้เสมอมา

ชนาเนตร์ ชวรางกูร

(นายชนาเนตร์ ชวรางกูร)

บรรณานุกรม

1. รองศาสตราจารย์กฤษดา วิศวธีรานนท์, "PCตัวควบคุมซีเควิ้น หลักการทำงานและการประยุกต์" พิมพ์ครั้งที่ 5, กรุงเทพมหานคร : โรงพิมพ์จุฬาลงกรณ์มหาวิทยาลัย, 2539.
2. กนก กุสุมาลย์นุกูล, "คู่มือการเขียนโปรแกม Delphi4 ฉบับใช้งานได้จริง",พิมพ์ครั้งที่ 1, กรุงเทพมหานคร : บริษัท ซัคเซส มีเดีย จำกัด, 2542.
3. สุนทร วิสูสุรพจน์, " การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051", พิมพ์ครั้งที่ 1, กรุงเทพมหานคร :บริษัท ซีเอ็ดยูเคชั่น จำกัด(มหาชน), 2537.
4. รองศาสตราสมยส จุณณะปิยะ, "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ตระกูล MCS-51", พิมพ์ครั้งที่ 2, กรุงเทพมหานคร : สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาด-กระบัง, 2541.