การแปลงโปรแกรมจากภาษา ASSEMBLY เป็นภาษา LADDER

CONVERT ASSEMBLY PROGRAM TO LADDER PROGRAM

นางสาวภาสินี พุทธศร

โครงงานสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีไทย – ญี่ปุ่น
พ.ศ. 2554

การแปลงโปรแกรมจากภาษา ASSEMBLY เป็นภาษา LADDER

CONVERT ASSEMBLY PROGRAM TO LADDER PROGRAM

นางสาวภาสินี พุทธศร

โครงงานสหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีไทย – ญี่ปุ่น
พ.ศ. 2554

คณะกรรมการสอบ

.....ประธานกรรมการสอบ
(อาจารย์ ต่อเกียรติ ใต้ธงชัย)

.....กรรมการสอบและอาจารย์ที่ปรึกษา
(อาจารย์บัญชา บริคุต)

.....กรรมการ
(อาจารย์ ศิษฏ์ชฎา อ่ำเทศ)

บทสรุป

ชื่อโครงงาน	การแปลงโปรแกรมจากภาษา Assembly เป็นภาษา Ladder
หน่วยกิต	6
ผู้เขียน	นางสาวภาสินี พุทธศร
อาจารย์ที่ปรึกษา	อาจารย์บัญชา บริคุต
หลักสูตร	วิศวกรรมศาสตรบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
คณะ	วิศวกรรมศาสตร์
พ.ศ.	2554

บทสรุป

งานที่ปฏิบัติ

1. เขียนโปรโตคอลเชื่อมต่อการรับ-ส่งข้อมูลระหว่าง PLC รุ่น FX กับ PLC รุ่น A3A
2.พัฒนาโปรแกรมภาษา Assembly เป็นภาษา Ladder ใน PLC รุ่น Q
3.เขียนโปรแกรมเชื่อมต่อการทำงานของ Inverter ทั้งหมด 9 ตัว
4.ปฏิบัติงานอื่นๆตามที่ได้รับมอบหมายจากพี่วิศวกรในบริษัท

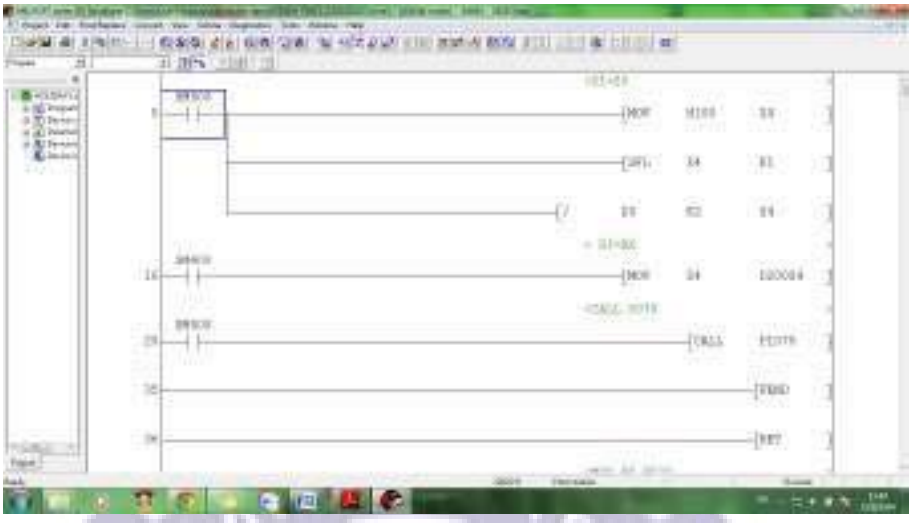
ผลที่ได้รับจากการดำเนินงานและประโยชน์ที่ได้รับ

1.ได้โปรแกรมที่นำไปใช้เชื่อมต่อการรับ-ส่งข้อมูลระหว่าง PLC ทั้งสองรุ่นได้และมีความรู้เกี่ยวกับทางด้านการเขียนโปรแกรมเพิ่มขึ้น

2.ได้ Ladder Diagram ที่เอาไว้ใช้สำหรับ PLC รุ่น Q และเป็นแนวทางในการพัฒนาโปรแกรมต่อไป และยังได้รับทักษะการเขียนโปรแกรมและใช้คำสั่งต่างๆเพิ่มขึ้นด้วย

3.หลังจากที่เขียนโปรแกรมแล้วสามารถสั่งให้ Inverter ทำงานตามคำสั่งได้ และมีทักษะในการใช้งาน PLC เพิ่มขึ้น

4.ได้รับความรู้และทักษะในการเขียนโปรแกรมและการใช้งานอุปกรณ์ต่างๆที่เกี่ยวข้องกับPLCเพิ่มมากขึ้น



รูปโปรแกรมที่ได้จากการแปลงภาษา Assembly มาเป็น Ladder Diagram จำนวน 2425 Step

รูปการเชื่อมต่อการทำงานของ Inverter 9 ตัว

กิตติกรรมประกาศ

ในการจัดทำโครงงานชิ้นนี้สำเร็จด้วยดีเพราะได้รับความอนุเคราะห์จากหลายท่านด้วยกันซึ่งอาจจะนำมากล่าวได้ไม่หมด โดยท่านแรกคือ คุณเสกสัณห์ วัฒนะโชติ ซึ่งท่านเป็นผู้จัดการแผนกวิศวกรรม และเป็นพี่เลี้ยงที่ปรึกษา ท่านได้ให้ความรู้และแนวทางเกี่ยวกับ PLC , ภาษาแอสแซมบลี และรวมถึงการเขียนโปรแกรม ที่ต้องนำมาใช้งาน เทคนิคและทฤษฎีต่าง ๆ ด้วย และยังช่วยในการตรวจสอบรายละเอียด ความถูกต้องของรายละเอียดโครงงานชิ้นนี้ด้วยเพื่อให้ สมบูรณ์ที่สุด ท่านที่สองคือ อาจารย์บัญชา บริคุต เป็นอาจารย์ที่ปรึกษาโครงงานสหกิจศึกษา ท่านได้ช่วยตรวจสอบความถูกต้องของโครงงานและให้คำแนะนำในการปฏิบัติงานด้วยเช่นกัน และนอกจากท่านทั้งสองนี้ยังมีบุคคลอื่น ๆ คือ พี่ ๆ วิศวกรภายในบริษัท เอฟ เอ เทค ทั้งฝ่าย Service, ฝ่าย Repair และฝ่าย Training ที่ให้ความช่วยเหลือในการทำงานมาตลอดเป็นอย่างดี ผู้ปฏิบัติงานจึงถือโอกาสนี้ขอขอบพระคุณทุกท่านไว้เป็นอย่างสูงมา ณ ที่นี้ด้วย

บทที่ 1
บทนำ

1.1 ชื่อและที่ตั้งของสถานประกอบการ

รูปที่ 1.1 Logo บริษัท เอฟ.เอ.เทค จำกัด

บริษัท เอฟ.เอ.เทค จำกัด 896/19,20,21,22 อาคารชุด เอสวี ซิตี้ อาคารที่ 1 ชั้น 12,14 ถนนพระราม 3 แขวงบางโพงพาง เขตยานนาวา กรุงเทพฯ 10120 โทรศัพท์ 02 682 6522 (อัตโนมัติ 20 สาย) แฟกซ์ 02 682 6020

รูปที่ 1.2 แผนที่บริษัท เอฟ.เอ.เทค

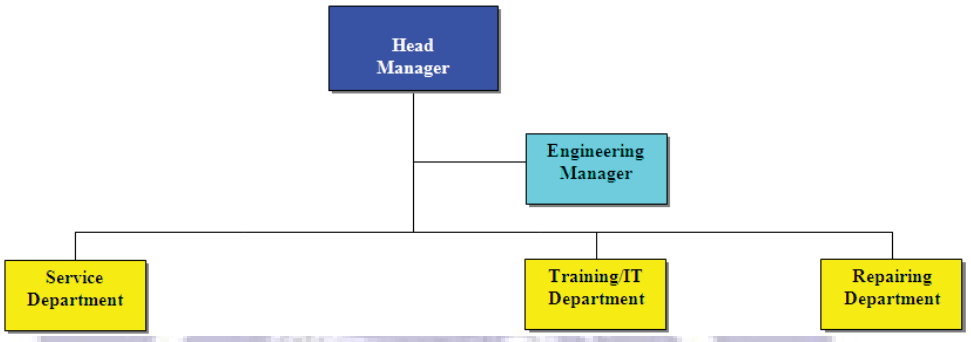
บริษัท เอฟ.เอ.เทค จำกัด เป็นบริษัทชั้นนำในการจำหน่าย และให้บริการ อุปกรณ์การควบคุมแบบอัตโนมัติภายใต้แบรนด์สินค้า Mitsubishi ที่ได้รับการแต่งตั้งอย่างเป็นทางการแต่เพียงผู้เดียวในประเทศไทย อาทิ Programmable Logic Controller (PLC), Inverter, Robot,Servo motor, Graphic Operation Terminal (GOT), HMI Software, CNC, เปิดดำเนินการมามากกว่า 19 ปี ปัจจุบันผลิตภัณฑ์ของบริษัทได้รับการยอมรับ และ มียอดขายเติบโตมาโดยตลอด

1.2 ลักษณะธุรกิจของสถานประกอบการ

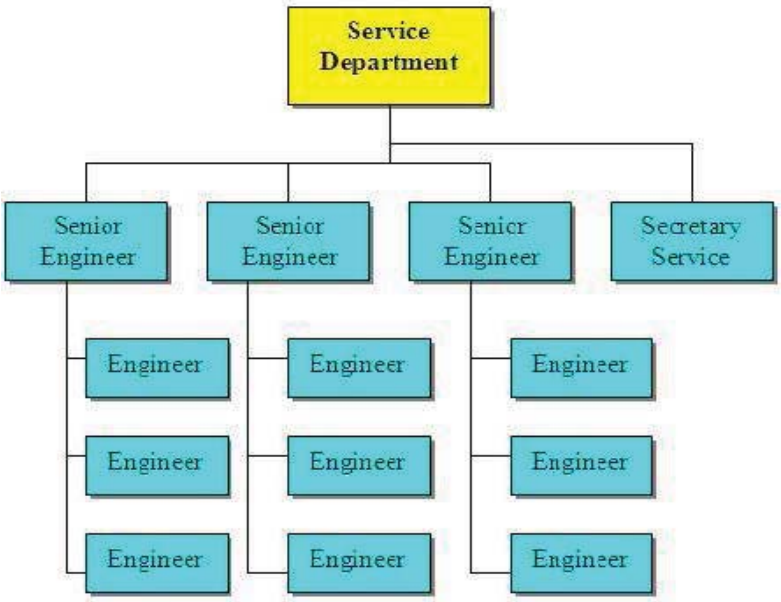
บริษัท เอฟ.เอ.เทค จำกัด ได้จัดตั้งขึ้นเมื่อปี ค.ศ 1991 โดยเริ่มจากการมีพนักงาน 7 คน ซึ่งในปัจจุบันมีมากกว่า 70 คนขึ้นไป ทางบริษัทได้เป็นตัวแทนในการจำหน่ายและให้บริการอุปกรณ์การควบคุมแบบอัตโนมัติภายใต้แบรนด์สินค้า Mitsubishi ในระหว่าง 15 ปีที่ผ่านมา บริษัทได้มีการจัดอบรมให้ความรู้เกี่ยวกับผลิตภัณฑ์ โดยได้มีการจัดอบรมให้แก่วิศวกรมากกว่า 20,000 คน และได้จำหน่าย CPUs มากกว่า 50,000 ชิ้น ให้กับโรงงานอุตสาหกรรมต่าง ๆ และในระหว่างนี้ บริษัทได้มีการดูแลลูกค้าทั้งทางด้านการบริการและการดูแลช่วยเหลือ โดยได้มีการจัดคอร์ส อบรมเกี่ยวกับผลิตภัณฑ์จำนวนทั้งหมด 21 คอร์ส และได้มีการจัดเทรนมามากกว่า 100 คอร์สต่อปี นอกจากนั้นยังได้มีการส่งวิศวกร ของทางบริษัทไปอบรมเกี่ยวกับวิธีการซ่อมแซม บำรุงรักษาและดูแลผลิตภัณฑ์ ที่ประเทศญี่ปุ่น ทำให้ในปัจจุบันวิศวกรของบริษัทสามารถดูแลและแก้ไขปัญหาต่างๆของเครื่อง PLC,Inverter,Servo & CNC ทั้งหมดในประเทศไทยได้

บริษัทมีความเชื่อมั่นในประสิทธิภาพของทีมงาน และหวังที่จะพัฒนาประสิทธิภาพในการทำงานให้ดียิ่งขึ้นไป เพื่อสามารถให้ลูกค้าให้ความไว้วางใจในสินค้าและบริการของทางบริษัทต่อไป

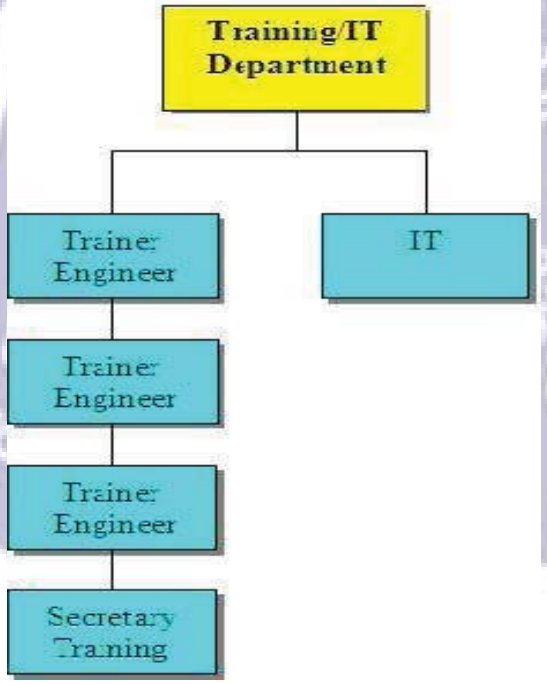
1.3 รูปแบบการจัดองค์กรและการบริหารองค์กร



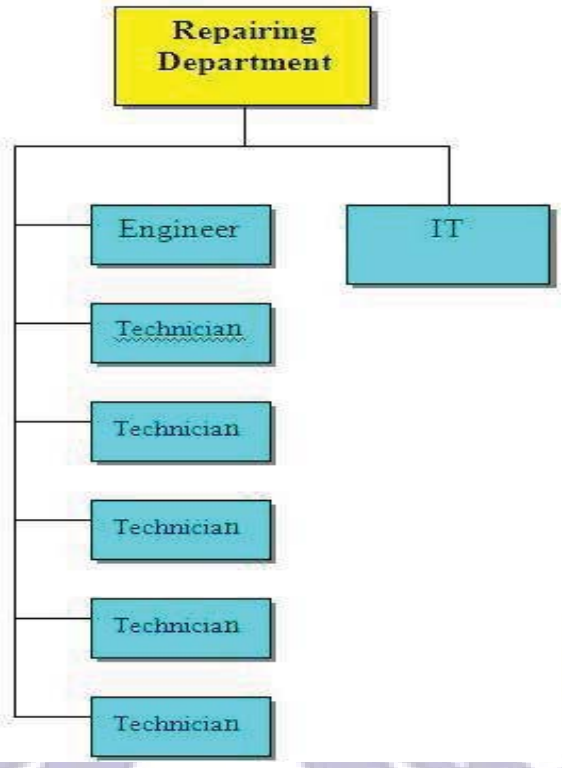
รูปที่ 1.3 รูปแบบการจัดองค์กร(1)



รูปที่ 1.4 รูปแบบการจัดองค์กร(2)



รูปที่ 1.5 รูปแบบการจัดองค์กร(3)



รูปที่ 1.6 รูปแบบการจัดองค์กร(4)

1.4 ตำแหน่งหน้าที่ และงานที่นักศึกษาได้รับมอบหมาย

ตำแหน่ง : ผู้ช่วยวิศวกร(PROGRAM SERVICE)

หน้าที่ : - เขียนโปรโตคอลเชื่อมต่อระหว่าง PLC รุ่น FX กับ PLC รุ่น A3A

- พัฒนาโปรแกรมภาษา Assembly เป็นภาษา Ladder ใน PLC รุ่น Q
- ปฏิบัติงานอื่นๆตามที่ได้รับมอบหมายจากพี่วิศวกรในบริษัท
- ทำคู่มือสำหรับงานที่ได้ปฏิบัติในขณะที่ดำเนินงานสหกิจศึกษา

1.5 พนักงานที่ปรึกษา และตำแหน่งของพนักงานที่ปรึกษา

พนักงานที่ปรึกษา : นายเสกสัณห์ วัฒนะโชติ

ตำแหน่ง : ผู้จัดการแผนกวิศวกรรม

1.6 ระยะเวลาที่ปฏิบัติงาน

เริ่มปฏิบัติงานสหกิจศึกษา ตั้งแต่วันที่ 1 มิถุนายน พ.ศ.2554 ถึง วันที่ 30 กันยายน พ.ศ.2554 รวมระยะเวลาทั้งสิ้น 4 เดือน รวมทั้งสิ้น 728 ชั่วโมง

1.7 วัตถุประสงค์ หรือจุดมุ่งหมายของการปฏิบัติงาน หรือโครงงานที่ได้รับมอบหมายให้ปฏิบัติงานสหกิจศึกษา

1.เขียนโปรโตคอลเชื่อมต่อระหว่าง PLC รุ่น FX กับ PLC รุ่น A3A

- ต้องการทำการ Link ข้อมูลระหว่าง FX 2 รุ่น และส่งข้อมูลไปยัง MODULE อีก 1 ตัว เพื่อทำการแสดงข้อมูลไปยังหน้าจอได้

2.พัฒนาโปรแกรมภาษา Assembly เป็นภาษา Ladder ใน PLC รุ่น Q

- ต้องการพัฒนาภาษา Assembly มาประยุกต์ใช้กับ PLC รุ่น Q
- ต้องการมีแนวทางในการดัดแปลงจากภาษา Assembly มาเป็นภาษา Ladder ได้

3.ต้องการตรวจสอบว่า Inverter สามารถทำงานพร้อมกัน โดยใช้คำสั่งที่มีอยู่

4.ทดลองและแก้ไขปัญหาเกี่ยวกับการใช้งานของ GOT,Inverter ที่ลูกค้าสอบถาม เพื่อให้ตรงตามวัตถุประสงค์ของลูกค้า

1.8 ผลที่คาดว่าจะได้รับจากการปฏิบัติงานหรือโครงงานที่ได้รับมอบหมาย

เขียนโปรโตคอลเชื่อมต่อระหว่าง PLC รุ่น FX กับ PLC รุ่น A3A

- สามารถเชื่อมต่อระหว่าง PLC 2 ตัว ซึ่งตัวแรกทำหน้าที่เป็นตัวแม่ และอีกหนึ่งตัวทำหน้าที่เป็นตัวลูก
- สามารถนำข้อมูลที่อยู่ในเครื่อง PLC ที่ได้จากการทำงานของเครื่องจักรที่อยู่ในโรงงาน มาแสดงผลออกหน้าจอได้
- สามารถนำรูปแบบการเขียนโปรโตคอลไปใช้ในการเชื่อมต่อระหว่าง PLC รุ่นอื่น ๆ กับ MODULE ต่าง ๆ ได้

พัฒนาโปรแกรมภาษา Assembly เป็นภาษา Ladder ใน PLC รุ่น Q

-นำภาษา Ladder ที่ได้จากการดัดแปลงมาจากภาษา Assembly มาพัฒนาต่อได้ โดยสามารถนำไปใช้กับชิ้นงานหรือเครื่องจักรในงานของลูกค้าได้

สามารถเชื่อมต่อ Inverter ให้ทำงานพร้อมกันหมดทั้ง 9 ตัวได้

บทที่ 2
ทฤษฎีและเทคโนโลยีที่ใช้ในการปฏิบัติงาน

ในการปฏิบัติงานครั้งนี้ได้มีการปฏิบัติงานเกี่ยวกับอุปกรณ์การควบคุมแบบอัตโนมัติหรือPLC โดยในการทำงานจำเป็นต้องมีซอฟต์แวร์เป็นตัวช่วยในการทำงาน ซึ่งใช้ในการเขียนโปรแกรมเพื่อสั่งให้เครื่อง PLC ทำงาน โดยซอฟต์แวร์ที่ใช้นั่นประกอบด้วย Gx Developer ซึ่งเป็นซอฟต์แวร์ที่ใช้ในการเขียนโปรแกรมสำหรับเครื่อง PLC และนอกจากนั้นยังต้องทำการศึกษาภาษา Assembly เพิ่มเติมอีกด้วยเพื่อใช้ในการปฏิบัติงาน

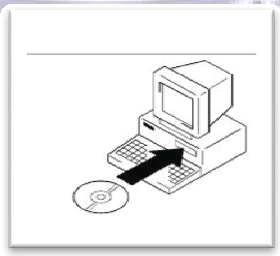
เกี่ยวกับ Software Gx Developer

Gx Developer คือ ซอฟต์แวร์ที่ใช้ในการเขียนโปรแกรม Ladder โดยเราจะต้องทำการเขียนโปรแกรมวงจร Ladder ให้ถูกต้อง โดยใช้คอมพิวเตอร์นี้มีฟังก์ชั่นการใช้งานที่สะดวกมาก ในกรณีนี้โปรแกรมจะเก็บในเครื่องคอมพิวเตอร์ โปรแกรมที่ถูกแปลงเป็นรหัสคำสั่งของ PLC จะถูก Down load ให้กับ PLC โดยผ่านสาย USB หรือสายที่ใช้เชื่อมต่อแบบ RS-232 ที่เชื่อมต่อระหว่าง PLC กับคอมพิวเตอร์ หลังจากที่ Down load แล้ว PLC จะทำงานตามโปรแกรมที่เราเขียนลงไปทันที พร้อมกับทั้งส่งข้อมูลสภาวะการทำงานกลับมายังคอมพิวเตอร์ เพื่อแสดงผลบนหน้าจอให้เห็นการทำงานของหน้าสัมผัส

2.1 โปรแกรม Gx-Developer

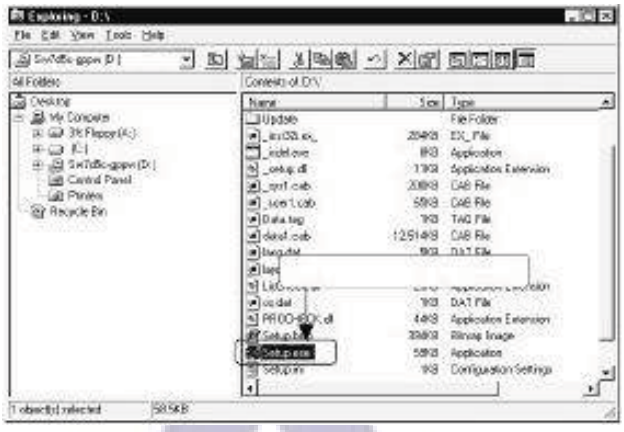
การติดตั้งโปรแกรม Gx-Developer

1.ใส่แผ่น Gx-Developer เพื่อเริ่มขั้นตอนการติดตั้ง



รูปที่ 2.1 การใส่แผ่น Gx-Developer

2. คลิกที่ Start แล้วเลือกที่ Gx-Developer จากนั้นเลือก setup เพื่อทำการติดตั้งโปรแกรม



รูปที่ 2.2 เลือก Set up เพื่อทำการ Install Program

3. ใส่ชื่อผู้ใช้โปรแกรมและชื่อบริษัท

รูปที่ 2.3 ใส่ชื่อผู้สร้างโปรแกรมและชื่อบริษัท

4. ใส่ Product ID ซึ่งดูได้จากเอกสารที่แนบมาพร้อมกับ Software

รูปที่ 2.4 ใส่ Product ID

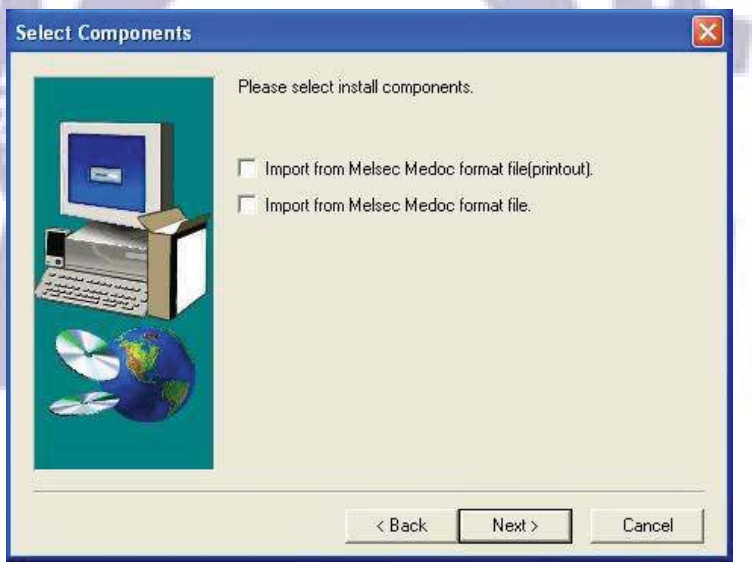
5. หากต้องการให้สามารถเขียนโปรแกรมในรูปแบบ Structure Text (ST) Language ให้ใส่เครื่องหมาย / ในช่อง ☐ (..... (ให้เลือก))

รูปที่ 2.5 เลือก Structure Text (ST) Language

6. หากต้องการให้โปรแกรมทำได้แค่พียง Monitor อย่างเดียว (ไม่สามารถที่จะทำการแก้ไขโปรแกรมได้) ให้ใส่เครื่องหมาย / ในช่อง ☐ ..... (ไม่ต้องเลือก)

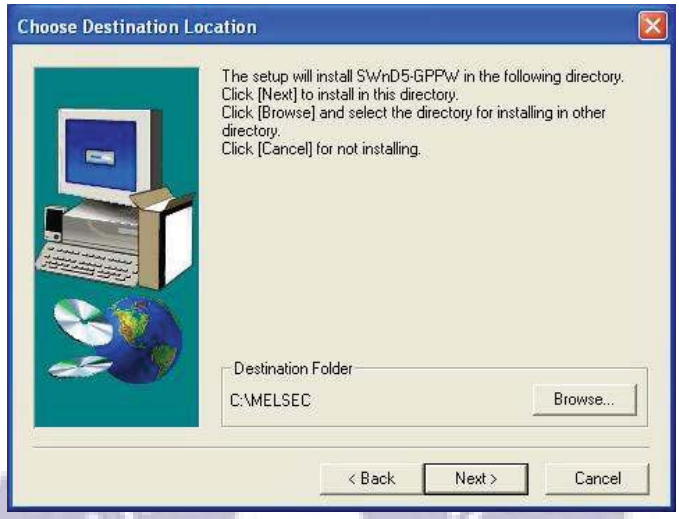
รูปที่ 2.6 เลือกเพื่อ Monitor อย่างเดียว

7. หากต้องการให้ Software Gx-Developer สามารถที่จะ Import Program จาก Software Medoc ได้ให้ใส่เครื่องหมาย/ ในช่อง ☐



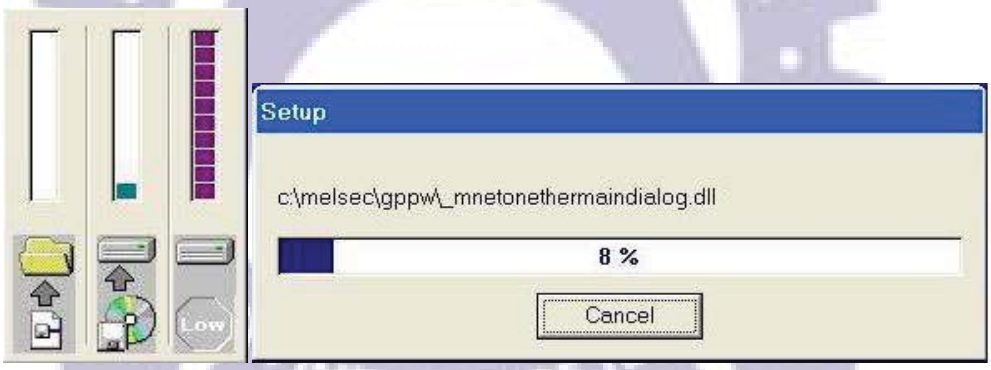
รูปที่ 2.7 เลือกเพื่อ Import Program จาก Software Medoc

8. เลือก Folder ปลายทาง ที่ต้องการเก็บโปรแกรม โดยคลิกที่ Browser หรือใช้ Folder ที่ถูกตั้งไว้อยู่แล้ว จากนั้นเลือก Next



รูปที่ 2.8 เลือกที่จัดเก็บ Program Software

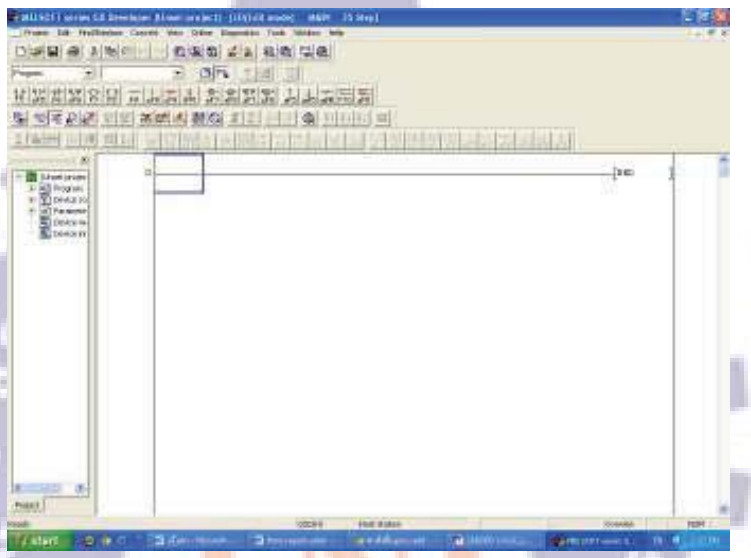
9. จากนั้น โปรแกรมจะทำการติดตั้ง



รูปที่ 2.9 ทำการติดตั้ง Software

10. เมื่อ Install เสร็จหน้าจอจะมีข้อความว่า Complete the installation of this Product เลือก OK

รูปที่ 2.10 การ Install Program เสร็จสิ้น



รูปที่ 2.11 รูป Software Gx-Developer

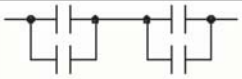
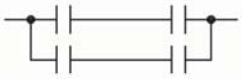
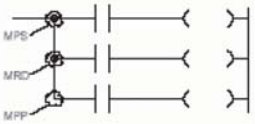
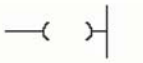
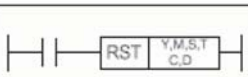
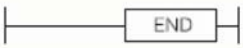
2.2 ภาษาแลดเดอร์

สำหรับภาษาที่ใช้กับตัว Software Gx-Developer นั้น เราจะใช้ภาษา Ladder ในการทำโปรแกรม โดยภาษา Ladder นั้นเป็นภาษาเชิงรูปภาพ ประกอบไปด้วยแลดเดอร์ไดอะแกรมเพื่อไว้ดูและคำสั่งแลดเดอร์เพื่อไว้สั่งงานถูกออกแบบมาเพื่อให้ง่ายต่อการใช้งาน แต่เดิมนั้นออกแบบมาแทนวงจรรีเลย์ ดังนั้นแลดเดอร์ไดอะแกรมก็จะอ้างอิงวงจรซีเคว้นของรีเลย์เป็นส่วนใหญ่ ต่อมามีการพัฒนาฟังชั่นให้สะดวกแก่การใช้งานมากขึ้น แต่จะเป็น PLC ในรุ่นที่สูง ๆ แต่ในการใช้งานจริงนั้น ถ้าไม่ซับซ้อนจนเกินไป ฟังชั่นพื้นฐานก็เพียงพอต่อการใช้งาน

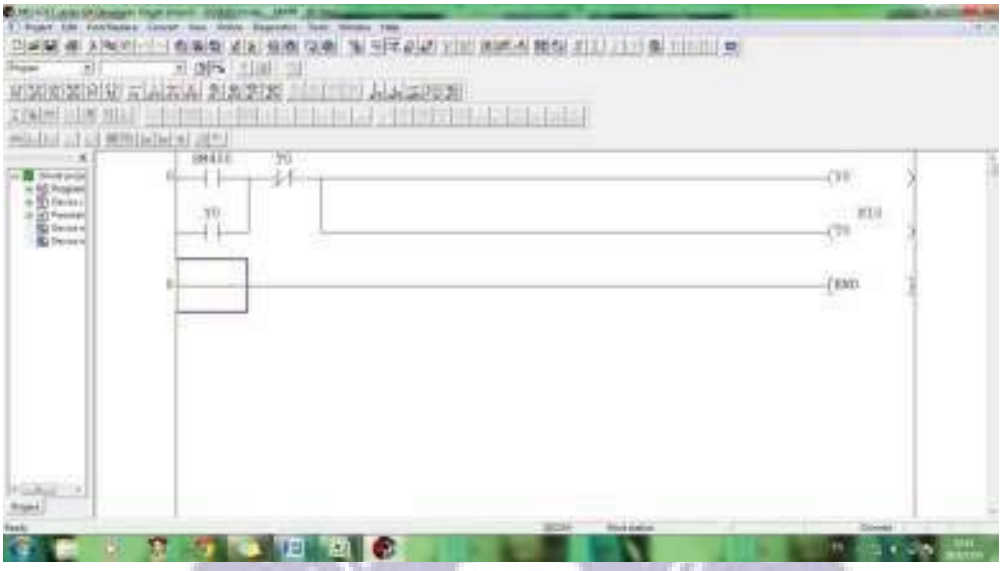
ตารางที่ 2.1 กลุ่มคำสั่งพื้นฐาน(Ladder Instruction & Output Control)

ประเภทของอุปกรณ์	รูปแบบคำสั่ง	สัญญลักษณ์	ความหมาย
หน้าสัมผัส	LD		หน้าสัมผัสแบบปกติเปิด
	LDI		หน้าสัมผัสแบบปกติปิด
	AND		หน้าสัมผัสแบบปกติเปิดมาต่ออนุกรม
	ANI		หน้าสัมผัสแบบปกติปิดมาต่ออนุกรม
	OR		หน้าสัมผัสแบบปกติเปิดมาต่อขนาน
	ORI		หน้าสัมผัสแบบปกติปิดมาต่อขนาน
	LDP		พัลซ์ขอบขาขึ้น
	LDF		พัลซ์ขอบขาลง
	ORP		พัลซ์ขอบขาขึ้นมาต่อขนาน
	ORF		พัลซ์ขอบขาลงมาต่อขนาน
	MEP		ทำงานที่พัลซ์ขอบขาขึ้น
	MEF		ทำงานที่พัลซ์ขอบขาลง
	INV		ทำงานตรงข้ามกับอินพุตที่เข้ามา

ตารางที่ 2.1(ต่อ) กลุ่มคำสั่งพื้นฐาน(Ladder Instruction & Output Control)

ประเภทของอุปกรณ์	รูปแบบคำสั่ง	สัญญลักษณ์	ความหมาย
การเชื่อมต่อ	ANB		การเชื่อมต่อแบบอนุกรมวงจร
	ORB		การเชื่อมต่อแบบขนานวงจร
	MPS MRD MPP		การเชื่อมต่อแบบผสม
เอาท์พุต	OUT		คอยล์เอาท์พุต
	SET		การเซ็ทคอยล์
	RST		การรีเซ็ทคอยล์
	PLS		กำเนิดพัลล์ที่ขอบขาขึ้น
	PLF		กำเนิดพัลล์ที่ขอบขาลง
มาสเตอร์คอนโทรล	MC		คำสั่งเขียนมาสเตอร์คอนโทรล
	MCR		สิ้นสุดการทำงานมาสเตอร์คอนโทรล
,โอน, ย้าย ขอมูลตัวเลข	MOV		ย้ายข้อมูลไปเก็บในรีจีสเตอร์
โปรแกรม END	-		สิ้นสุดการเขียนโปรแกรม

ตัวอย่างการเขียนโปรแกรมภาษา Ladder



รูปที่ 2.12 ตัวอย่างการเขียนโปรแกรมภาษา Ladder

2.3 ภาษาแอสเซมบลี (Assembly Language)

ภาษาแอสเซมบลี (Assembly Language) หมายถึง ภาษาที่ใช้ในการเขียนโปรแกรมภาษาหนึ่ง ซึ่งจะทำงานโดยขึ้นกับรุ่นของไมโครโพรเซสเซอร์ หรือ "หน่วยประมวลผล" (CPU) ของเครื่องคอมพิวเตอร์ การใช้ภาษาแอสเซมบลีจำเป็นต้องผ่านการแปลภาษาด้วยคอมไพเลอร์เฉพาะเรียกว่า แอสเซมเบลอร์ (assembler) ให้อยู่ในรูปของรหัสคำสั่งก่อน (เช่น .OBJ) โดยปกติ ภาษานี้ค่อนข้างมีความยุ่งยากในการใช้งาน และการเขียนโปรแกรมเป็นจำนวนบรรทัดมากมากกว่า เมื่อเปรียบเทียบกับการใช้ภาษาระดับสูง เช่น ภาษา C หรือภาษา BASIC แต่จะทำให้ได้ผลลัพธ์การทำงานของโปรแกรมเร็วกว่าและขนาดของตัวโปรแกรมมีขนาดเนื้อที่น้อยกว่าโปรแกรมที่สร้างจากภาษาอื่นมากจึงนิยมใช้ภาษานี้เมื่อต้องการประหยัดเวลาทำงานของเครื่องคอมพิวเตอร์และเพิ่มประสิทธิภาพของโปรแกรม เนื่องจากตัวคำสั่งภายในภาษาอ้างอิงเฉพาะกับรุ่นของหน่วยประมวลผลดังนั้นถ้ามีการเปลี่ยนแปลงไปใช้กับหน่วยประมวลผลอื่นหรือระบบอื่น (เช่น หน่วยประมวลผล x86 ไม่เหมือนกับ z80) จะต้องมีการปรับแก้ตัวคำสั่งภายในซึ่งบางครั้งอาจไม่สามารถปรับปรุงแก้ไขได้อย่างสมบูรณ์

ภาษาแอสเซมบลี (Assembly Language) เรียกอีกชื่อหนึ่งว่า ภาษาสัญลักษณ์ (Symbolic Programming Language) เนื่องจากภาษาเครื่องยากแก่การเขียนโปรแกรม จึงได้มีการพัฒนาภาษานี้ขึ้นมา จะใช้รหัสและสัญลักษณ์แทน 0 และ 1 ในการเขียนโปรแกรม

ข้อดีของภาษาแอสเซมบลี คือ จะใช้ตัวย่อที่มีความหมาย (นีมอนิก mnemonics) ในคำสั่งเพื่อเขียนโปรแกรม ภาษาแอสเซมบลีเรียนรู้ได้ง่ายและเร็วกว่าภาษาเครื่อง สามารถกำหนดชื่อที่เก็บข้อมูลในหน่วยความจำเป็นค่าภาษาอังกฤษ อย่างไรก็ตาม โปรแกรมที่เขียนขึ้นด้วยภาษาแอสเซมบลี ต้องทำการแปลด้วยโปรแกรมแปลภาษาที่เรียกว่า แอสเซมเบลอร์ (Assembler) โดยจะแปลงโปรแกรมต้นฉบับ (Source Document) ให้เป็นภาษาเครื่องซึ่งคอมพิวเตอร์สามารถเข้าใจได้ ตัวอย่างของสัญลักษณ์ที่ใช้ในภาษาแอสแซมบลี เช่น

A	ย่อมาจาก ADD	หมายถึง	การบวก
S	ย่อมาจาก SUBTRACT	หมายถึง	การลบ
C	ย่อมาจาก COMPARE	หมายถึง	การเปรียบเทียบ
MP	ย่อมาจาก MULTIPLY	หมายถึง	การคูณ
ST	ย่อมาจาก STORE	หมายถึง	การเก็บข้อมูลไว้หน่วยความจำ

ข้อเสีย เป็นภาษาที่ใช้ในยุคแรก ๆ จะมีความยุ่งยากในการเขียนมาก จัดอยู่ในภาษาระดับต่ำ คือภาษาเครื่องและภาษาแอสแซมบลี

ตัวอย่างการเขียนภาษา Assembly

```
0FDC:005D B80000        MOV     AX,0000
0FDC:0060 8ED8          MOV     DS,AX
0FDC:0062 B80000        MOV     AX,0000
0FDC:0065 8EC0          MOV     ES,AX
0FDC:0067 BF0090        MOV     DI,9000
0FDC:006A 26            ES:
0FDC:006B A1FE9F        MOV     AX,[9FFE]
0FDC:006E D1E0          SHL     AX,1
0FDC:0070 03F8          ADD     DI,AX
0FDC:0072 C3            RET
```

รูปที่ 2.13 ตัวอย่างภาษา Assembly

บทที่ 3
การปฏิบัติงานและรายละเอียด

3.1 แผนงานในการปฏิบัติงาน

ตารางที่ 3.1 แผนการปฏิบัติงาน

รายละเอียด	เดือนที่ 1	เดือนที่ 2	เดือนที่ 3	เดือนที่ 4
ศึกษาคำสั่งต่างๆของภาษา Assembly และวิธีการใช้โปรแกรม Debug				
ศึกษาคำสั่งต่างๆของภาษา Ladderและวิธีการใช้งานโปรแกรม Gx-Developer				
เขียนโปรแกรมLadderสำหรับPLC รุ่น Q โดยดัดแปลงจาก ภาษา Assembly				
ทำการตรวจสอบและแก้ไขโปรแกรม				
ทำ report และรายงานกับพี่เลี้ยงที่ปรึกษา				

จากตารางระยะเวลาการปฏิบัติงาน จะเห็นว่าได้แบ่งเป็นหัวข้อละ 2 แถว เป็นการแสดงการเปรียบเทียบระหว่างการวางแผนในการปฏิบัติงานกับการปฏิบัติงานจริงซึ่งในบางครั้งในการปฏิบัติงานจริงอาจไม่เป็นไปตามแผนการปฏิบัติงานที่วางไว้โดยอาจจะเร็วกว่าหรือช้ากว่าซึ่งได้มีการเปรียบเทียบโดยกำหนดดังนี้

สีฟ้า หมายถึง แผนระยะเวลาการปฏิบัติงานที่คาดการณ์ไว้

สีแดง หมายถึง ระยะเวลาการปฏิบัติงานจริง

โดยในขั้นตอนแรกเมื่อเราได้ Code จากภาษา Assembly มาแล้ว เราต้องทำการศึกษาคำสั่งและหาโปรแกรมที่นำมาใช้ทดลองเขียนบางคำสั่งเข้าไปเพื่อดูการทำงานของแต่ละคำสั่งนั้นระยะเวลาในการศึกษาและทำความเข้าใจในตัวคำสั่งและตัวโปรแกรมนั้นคาดว่าใช้เวลาประมาณ 3 สัปดาห์ เพราะได้มีพื้นฐานในตัวภาษา Assembly มาบ้างแล้ว และในทางปฏิบัติก็ได้ตรงกับแผนงานที่ได้ตั้งไว้

ในส่วนที่ 2 เมื่อเราได้ทำการศึกษาในตัวของ Assembly แล้ว ต่อมาเราจำเป็นต้องมีความรู้เกี่ยวกับภาษา Ladder และตัว Software Gx-Developer เพื่อใช้ในการเขียนโปรแกรม ในแผนงานที่ตั้งไว้ ได้ประมาณเวลาในการศึกษาและทำความเข้าใจในตัวโปรแกรมไว้ 3 สัปดาห์ แต่ในทางปฏิบัติแล้วไม่สามารถทำได้ตรงกับแผนที่ตั้งไว้ เพราะตัวโปรแกรมและภาษาที่ใช้ เป็นภาษาที่ยังไม่เคยได้ทดลองใช้มาก่อน จึงจำเป็นต้องใช้เวลานานพอสมควรในการศึกษาทั้งคำสั่ง การใช้งาน Software เพื่อความเข้าใจและนำมาใช้งานได้ถูกต้องในส่วนนี้เองที่ทำให้แผนงานที่ตั้งไว้กับการปฏิบัติงานจริงได้มีการเปลี่ยนไปไม่ตรงตามกับที่ตั้งไว้ มาถึงในส่วนของการเขียนโปรแกรมซึ่งจำเป็นต้องใช้ความรู้และความเข้าใจในภาษาทั้งสองเป็นอย่างมากและด้วยโปรแกรม Assembly มีความยาวและเยอะมากพอสมควรทำให้การเขียนโปรแกรมเป็นไปอย่างล่าช้ากว่ากำหนด 1 สัปดาห์ แต่ก็ไม่ได้เป็นผลกระทบต่องานที่จะทำมากเท่าไรนัก จากนั้นเมื่อเราเขียนโปรแกรมเสร็จแล้วต้องมีการตรวจสอบความถูกต้องและทำการแก้ไขในส่วนที่ผิดซึ่งส่วนนี้ก็ใช้เวลานานกว่าแผนงานที่ตั้งไว้เช่นกันเพราะต้องมีความละเอียดรอบคอบในการทดสอบและตรวจสอบโปรแกรมเป็นอย่างมากเมื่อทำการตรวจสอบและแก้ไขเสร็จแล้วก็ได้ทำ Report เพื่อนำเสนอต่อพี่เลี้ยงที่ปรึกษาในส่วนนี้เองที่เราสามารถย่นระยะเวลาในส่วนที่ขาดไปให้พอดีกับระยะเวลาที่เหลือได้

3.2 รายละเอียดงานที่นักศึกษาปฏิบัติในงานสหกิจศึกษา หรือรายละเอียดโครงงานที่ได้รับมอบหมาย

1. แปลงโปรแกรมจากภาษา Assembly เป็นภาษา Ladder ไว้สำหรับ PLC รุ่น Q โดยได้รับงานจากพี่เลี้ยงที่ปรึกษา จากนั้นพี่เลี้ยงได้ให้เข้าเทรนคอร์สพื้นฐานสำหรับใช้ในการเขียนโปรแกรม

2. เข้าเทรนคอร์ส พื้นฐาน Gx-developer, Advance PLC,Inverter โดยได้เข้าร่วมเทรนพร้อมกับลูกค้าของบริษัท และได้มีการสอบวัดความรู้หลังจากที่ได้รับการเทรน

3. เขียนโปรแกรมเชื่อมต่อและควบคุมการทำงานของ Inverter โดยเป็นการทดลองเพื่อทดสอบดูว่าสามารถเขียนโปรแกรมควบคุม Inverter ให้ทำงานตามคำสั่งพร้อมกันได้ มากกว่า 8 ตัวหรือไม่

4. เขียนโปรแกรมเพื่อเชื่อมต่อการทำงานของ PLC รุ่น FX กับ PLC รุ่น A เพื่อต้องการสั่งให้ PLC ทั้งสองรุ่นทำงานร่วมกันได้

5.ออกไปปฏิบัติงานนอกสถานที่ โดยไปพร้อมกับพี่เลี้ยงที่ปรึกษา

3.3 ขั้นตอนการดำเนินงานที่นักศึกษาปฏิบัติงานหรือโครงงาน

3.3.1 แปลงโปรแกรมจากภาษา Assembly เป็นภาษา Ladder ไว้สำหรับ PLC รุ่น Q

1. เราต้องรู้ว่าคำสั่งแต่ละคำสั่งของภาษา Assembly ทำงานอย่างไร และมีความหมายว่าอย่างไร

2. เมื่อเรารู้ในส่วนของ คำสั่งแล้ว เราต้องรู้ว่า register ที่ใช้ในโปรแกรมนั้น ทำหน้าที่อะไร เพราะใน Ladder แล้ว จะมี register ที่ใช้ในการทำงานที่แตกต่างกันอยู่

3. เมื่อเราได้ทำการศึกษาส่วนของ Assembly แล้ว เราก็ต้องมาศึกษาในส่วนขง Ladder อีกเช่นกันว่า มีการทำงานของแต่ละคำสั่งอย่างไร และ register มีอะไรบ้าง ทำหน้าที่อะไร

4. จากนั้น เราต้องทำความเข้าใจการทำงานของตัวโปรแกรมหลักที่เราจะนำมาแปล ว่ามีการทำงานอย่างไร เปลี่ยนค่าอย่างไร ซึ่งก็เป็นส่วนสำคัญส่วนหนึ่งกเช่นกันในการทำโปรแกรม

5. เมื่อเรารู้ทั้งคำสั่งการทำงาน และหน้าที่ของ register แต่ละตัวแล้ว เราก็ทำการแปลโปรแกรมจาก ภาษา Assembly มาเป็นภาษา Ladder ได้ แต่ในขณะที่ทำการแปล ต้องดูด้วยว่าโปรแกรมที่ใหม่ที่ได้ ให้ผลกรทำงานเหมือนกันกับของเดิมหรือไม่

ตัวอย่างการแปล

ภาษา Assembly	ภาษา Ladder
MOV AX,0000	MOV H0000 D0
SHL AX,1	SHL D0 K1
SUB AX,+02	- H02 D0 D0
ADD AX,+02	+ H02 D0 D0
CALL 078f	CALL P0
MOV DI,9000	W0

จากตัวอย่างอธิบายได้ว่า

1.1 MOV AX,0000 ใน Assembly หมายความว่า นำค่า 0 มาไว้ ใน register ax ซึ่ง register ax ทำหน้าที่เก็บข้อมูลไว้สำหรับนำไปทำการคำนวณ

จะได้

MOV H0000 D0 ใน Ladder หมายความว่า นำค่า 0 ไปเก็บไว้ใน register D0 ซึ่งเป็น data register ทำหน้าที่เก็บข้อมูลของโปรแกรมในภาษา Ladder ซึ่ง สามารถใช้ได้ตั้งแต่ D0-D12287 นำไปใช้ได้กับ register ในภาษา Assembly ที่มีหน้าที่เกี่ยวกับการเก็บข้อมูลหรือนำไปกระทำการต่างๆ

1.2 SHL AX,1 ในภาษา Assembly หมายความว่า shift ข้อมูลที่อยู่ใน AX ไปทางซ้าย 1 ตำแหน่ง แล้วเก็บไว้ที่ AX

จะได้

SHL D0 K1 ใน Ladder หมายความว่า shift ข้อมูลที่อยู่ใน D0 ไปทางซ้าย 1 ตำแหน่งแล้วเก็บไว้ที่ D0

1.3 SUB AX,+02 ในภาษา Assembly หมายความว่า AX=AX- H02

จะได้

H02 D0 D0 ใน Ladder โปรแกรม หมายความว่า D0=D0-H02

1.4 ADD AX,+02 02 ในภาษา Assembly หมายความว่า AX=AX+ H02

จะได้

+ H02 D0 D0 ใน Ladder โปรแกรม หมายความว่า D0=D0+02H

1.5 CALL 078f 02 ในภาษา Assembly หมายความว่า ทำการเรียกโปรแกรมย่อยที่ต้องการมาเพื่อทำงาน โดยในส่วนนี้ถ้ามีการใช้คำสั่ง CALL จะต้องมีการใช้ คำสั่ง RET ปิดท้ายด้วยเสมอ

จะได้

CALL P0 ใน Ladder โปรแกรม คือ เมื่อ ในโปรแกรมต้นฉบับมีการเรียกใช้งานโปรแกรมย่อย ซึ่งในโปรแกรม Ladder จะเทียบได้กับการเรียกใช้ Pointer สามารถใช้ได้ทั้ง CALL และการ JUMP ไป

ที่คำสั่งหรือ Pointer ต่างๆ ซึ่งเราสามารถเรียกใช้ Pointer ได้ตั้งแต่ P0-P4095 สำหรับใช้ในการกระโดดข้ามหรือ เรียกใช้ไปยังฟังก์ชั่นย่อยๆ หรือตำแหน่งที่เราต้องการได้

1.6 MOV DI,9000 ในภาษา Assembly หมายความว่า DI ชี้ไปที่ตำแหน่งที่ 9000 ซึ่ง เราก็ต้องมาดูว่าที่ตำแหน่งที่ 9000 ของ PLC มี Data อะไรอยู่ ถ้าเราเข้าใจในตัวโปรแกรมว่าต้องการอะไร ก็จะสามารถนำข้อมูลหรือ Register ที่อยู่ในตำแหน่งนั้นๆมาใช้งานได้ ซึ่งในกรณีของตัวอย่างนี้ ที่ตำแหน่ง 9000 ของ PLC เป็น register W0 ก็สามารถนำ W0 มาใช้งานในโปรแกรมได้เลย โดยไม่ต้องเขียนโปรแกรมในส่วนของ Code นี้ ส่วนของตัวชี้ ใน Ladder ก็จะมี Z0-Z15 ไว้ใช้สำหรับชี้ตำแหน่งต่างๆเช่นกัน ก็สามารถนำ ตัวชี้เหล่านี้มาใช้งานได้เช่นกัน

จากตัวอย่างด้านบนเป็นแค่ส่วนหนึ่งในการทำการแปลโปรแกรม ซึ่งในส่วนของ Assembly ยังมีคำสั่งอื่นๆที่นอกเหนือจากนี้อยู่อีกมาก และรวมถึงส่วนของ Ladder ด้วยอีกเช่นกัน

3.3.2 เขียนโปรแกรมเชื่อมต่อและควบคุมการทำงานของInverter

1. ทำการศึกษาข้อมูลของ Inverter และคำสั่งที่ใช้ในการเชื่อมต่อการทำงาน โดยได้ศึกษาในคู่มือของบริษัท

2. จากนั้นได้ทดลองเขียนโปรแกรมเพื่อเชื่อมต่อและควบคุมการทำงานของ Inverter โดยได้ให้พี่ที่มีความรู้เกี่ยวกับการตั้งค่าพารามิเตอร์ต่างๆเข้ามาช่วยในการตั้งค่าให้กับ Inverter และการต่อสายเพื่อ Link การทำงาน เนื่องจากการต่อสายมีความยุ่งยากซับซ้อนและตัวนักศึกษาเองยังไม่มีความชำนาญมากพออาจทำให้เกิดความเสียหายต่ออุปกรณ์จึงจำเป็นต้องมีรุ่นพี่เข้ามาช่วยเหลือในงานนี้ด้วย

3. เมื่อได้ทดลองโปรแกรมและแก้ไขแล้ว ผลที่ได้คือสามารถเชื่อมต่อและควบคุมการทำงานของ Inveter ได้ ดังนั้นจึงได้รายงานต่อพี่เลี้ยงที่ปรึกษาเพื่อนำเสนอกับลูกค้าต่อไป

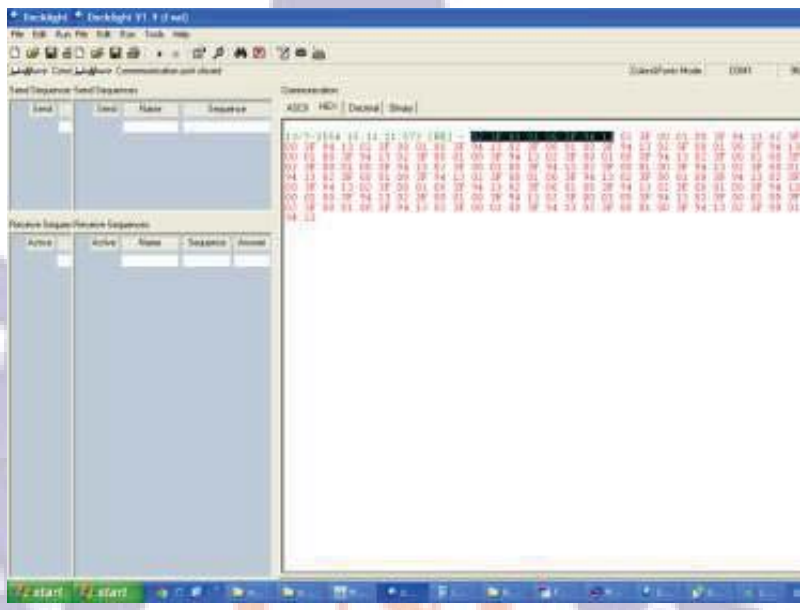
3.3.3 เขียนโปรแกรมเพื่อเชื่อมต่อการทำงานของ PLC รุ่น FX กับ PLC รุ่น A

1. ในขั้นตอนแรกเราต้องศึกษาเกี่ยวกับรายละเอียดของคำสั่งที่ใช้ในการเขียนโปรแกรมเพื่อเชื่อมต่อการทำงานของ PLC A และ FX3U

2. โดยในตอนเริ่มต้น PLC ทั้งสองเป็นการเชื่อมต่อแบบ โปรโตคอล MODBUS แต่เมื่อเราได้เขียนการเชื่อมต่อแบบ MODBUS และทำการรันโปรแกรม ปรากฏว่าไม่สามารถทำให้ PLC ทั้งสองเชื่อมต่อและตอบสนองกันได้ ดังนั้นจึงต้องหาวิธีใหม่ในการทดลอง

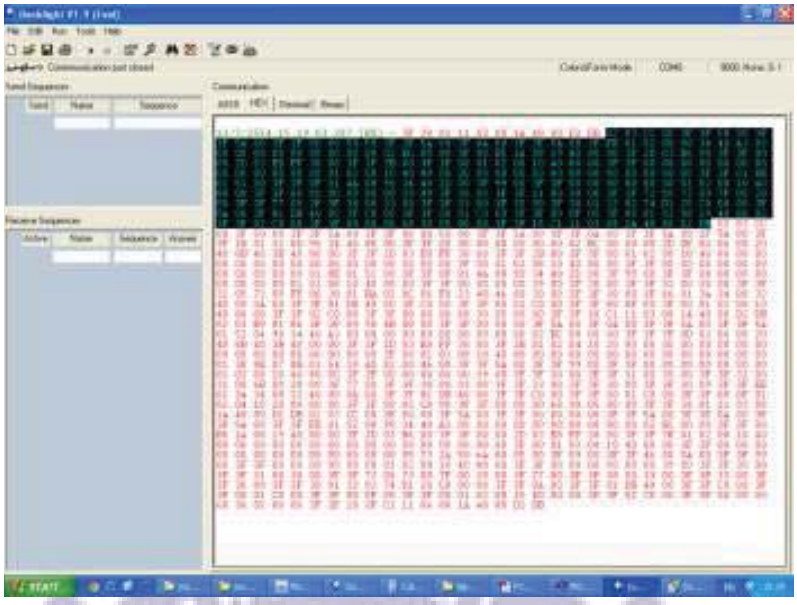
3. จากนั้นเราจึงได้เขียนโปรแกรมแบบ Non-Protocal ขึ้นมาเพื่อทดลองใช้เชื่อมต่อกัน ปรากฏว่าสามารถทำให้ PLC ทั้งสองติดต่อกันได้

ผลที่ได้จากการเขียนโปรแกรมเชื่อมต่อการสื่อสารของ PLC สามารถดักจับข้อมูลได้ดังนี้



รูปที่ 3.1 รูปแสดงการดักจับการส่งข้อมูลของ PLC

จากรูปเราจะนำข้อมูลที่ได้มาเทียบดู และใช้ชุดคำสั่งเขียนโปรแกรม เพื่อดูข้อมูลที่ฝั่งรับตอบกลับข้อมูลที่ตอบกลับจะเป็นดังนี้



รูปที่ 3.2 รูปแสดงการตอบกลับข้อมูลของ PLC

4. จากนั้นเมื่อเราสามารถทำให้อุปกรณ์ทั้งสองสามารถถาม-ตอบข้อมูลกันได้ จึงนำเสนอพี่เลี้ยงที่ปรึกษา เพื่อจะนำไปใช้แก้ไขปัญหาให้กับลูกค้าต่อไป

บทที่ 4

สรุปผลการดำเนินงาน การวิเคราะห์และสรุปผลต่าง ๆ

4.1 สรุปผลการดำเนินงาน

จากการปฏิบัติงานสหกิจมาเป็นเวลา 4 เดือน สามารถสรุปการดำเนินงานได้ดังนี้

1. จากการเขียนโปรแกรมเพื่อเชื่อมต่อการทำงานของ PLC รุ่น FX กับ PLC รุ่น A นั้น ถือเป็นงานแรกที่ได้รับมอบหมายจากพี่เลี้ยงที่ปรึกษา จึงค่อนข้างใช้เวลาในการปฏิบัติงานค่อนข้างจะนาน ใช้เวลาถึงเกือบ ๆ 2 เดือน เพราะต้องมีการทดลองค้นคว้า และเรียนรู้โปรแกรมจากเริ่มต้น จนกระทั่งมาเขียนโปรแกรมเพื่อใช้เชื่อมต่อและนำไปใช้ทดลองกับเครื่องของลูกค้าเนื่องจากเราต้องเรียนรู้โปรแกรมใหม่ และต้องศึกษา ประกอบกับ ไม่มีความชำนาญ และแม่นยำในการทำโปรแกรม จึงทำให้งานเป็นไปอย่างล่าช้าเช่นกันแต่หลังจากจบงานนี้แล้วก็สามารถนำความรู้ที่มีจากการปฏิบัติงานชิ้นแรกมาประยุกต์กับการใช้งานครั้งต่อ ๆ ไปได้

2. ในการดำเนินงานการแปลงโปรแกรมภาษา Assembly เป็นภาษา Ladder นั้น ในขั้นตอนการศึกษาทำความเข้าใจนั้นอาจใช้เวลานานมากเกินไปเนื่องจากการศึกษาในตัวโปรแกรมนั้นไม่ละเอียดรอบคอบเท่าไรนักจึงต้องมีการกลับมาแก้ไขรายละเอียดเพิ่มเติมโดยส่วนมากที่พบจะมีปัญหาเกี่ยวกับ register เพราะในภาษา Assembly จะเป็นการเขียนโปรแกรมเกี่ยวกับ register ทั้งหมด ซึ่งในเมื่อต่างภาษาต่างกันแล้วไม่มีทางที่จะใช้ register แบบเดียวกันได้ แต่ต้องนำ register มาเทียบดูกันว่าแต่ละตัวทำหน้าที่อะไร และมีตัวไหนในภาษา Ladder ทำหน้าที่เหมือน register ในภาษา Assembly และจะเขียนโปรแกรมอย่างไรจึงจะได้ผลที่เหมือนกันมันจึงค่อนข้างยุ่งยากและซับซ้อนในการทำความ

3. ต่อมาในส่วนของการเขียนโปรแกรมเชื่อมต่อและควบคุมการทำงานของ Inverter นั้น ได้ใช้ระยะเวลาถึง 2 สัปดาห์ในการทดลอง ซึ่งความเป็นจริงแล้วน่าจะใช้เวลาไม่เกิน 1สัปดาห์ เพราะได้มีพื้นฐานมาแล้วจากการทำงานในข้อ1. แต่ที่ทำให้ล่าช้าเพราะไม่ชำนาญในการเขียนโปรแกรมประยุกต์สำหรับการเชื่อมต่ออุปกรณ์หลายๆตัวและการต่อสายยังมีปัญหาเนื่องจากการต่อสายเข้ากับอุปกรณ์หลายๆตัวนั้นจำเป็นต้องมีความละเอียดรอบคอบมากเพราะถ้ามีการต่อสายผิดอุปกรณ์อาจไม่ทำงาน

ตามโปรแกรมที่เราเขียนไว้ก็ได้ เช่น เราต้องการให้ Inverter รับ-ส่ง ข้อมูลกับ Computer ต้องต่อ สายที่เป็นสายส่ง ของ Computer เข้ากับ ขาที่เป็นฝั่งรับของ Inverter แต่เราต่อสายที่เป็นสายส่งของ Computer เข้ากับขาส่งของ Inverter การรับ-ส่งข้อมูล อาจไม่เกิดขึ้นเรย หรือไม่เป็นไปตามที่ควร ในกรณีนี้ยังมีการต่อผิดอยู่บ้างจึงเป็นส่วนหนึ่งของปัญหาแต่ก็สามารถแก้ไขปัญหาได้

4.2 วิเคราะห์และสรุปผลต่าง ๆ

เนื่องจากการปฏิบัติงานทั้ง 3 ชิ้นงาน เป็นงานที่ต้องมีการตรวจสอบและแก้ไขการทำงานของ PLC ที่มีการผิดพลาดจากโรงงานแล้วนำไปเสนอให้กับลูกค้าดังนั้นจึงต้องมีการทดลองเพื่อให้บรรลุผลตามต้องการ จึงต้องมีการควบคุมเวลาในการทดลองเพื่อให้เป็นไปตามแผนงานที่วางไว้

4.2.1 การเขียนโปรแกรมเชื่อมต่อ PLC FX และรุ่น Q ผลที่ได้คือ สามารถทำการเขียนโปรแกรมเพื่อเชื่อมต่อการทำงานของอุปกรณ์ทั้งสองชนิดนี้ ได้ และดูข้อมูลที่ใช้รับ-ส่งกันได้หลังจากทำการ load โปรแกรมเข้าไปในตัวเครื่องแล้ว จากการทดลองพบว่า สาเหตุที่ในตอนแรกทำการเชื่อมต่อกันไม่ได้เพราะ ต่างคิดว่าต้องเขียนโปรแกรมแบบการใช้ชุดคำสั่งของโปรโตคอล Modbus แต่หลังจากที่ได้ทดลองแล้วก็ไม่สามารถทำการเชื่อมต่อได้ อาจเป็นเพราะการตั้งค่าต่าง ๆ ไม่ถูกต้องหรือใช้รูปแบบการเขียนเพื่อให้อุปกรณ์ติดต่อกันนั้นไม่ถูกรูปแบบจึงได้มีการประยุกต์มาใช้การเขียนแบบ Non-Protocal ก็สามารถทำให้อุปกรณ์ติดต่อกันได้ จึงได้ยึดแนวทางนี้ในการเขียนและแก้ไขปัญหาต่อ ๆ มา

4.2.2 การเขียนโปรแกรมเพื่อเปลี่ยนจากภาษา Assembly มาเป็นภาษา Ladder สำหรับใช้ใน PLC รุ่น Q ผลคือ ได้โปรแกรมที่เป็น Ladder Diagram ใช้สำหรับ PLC รุ่น Q แต่ในการปฏิบัติงานชิ้นนี้ถือว่าต้องใช้เวลาเป็นอย่างมากเพราะโปรแกรมค่อนข้างซับซ้อนและต้องใช้ความละเอียดในการทำความเข้าใจ ในแต่ละโปรแกรม แต่ละคำสั่งมีการใช้งาน register ที่ค่อนข้างจะเข้าใจยาก ต้องนำมาเทียบกันกับ register ที่มีอยู่ในPLC เพื่อจะได้ข้อมูลและการใช้งานที่ถูกต้อง เพราะถ้าเรานำมาใช้งานผิดหน้าที่ ก็จะทำให้การทำงานและผลที่ได้ผิดเพี้ยนไปเช่นกัน

4.2.3 การเขียนโปรแกรมเชื่อมต่อและควบคุมการทำงานของ Inverter ผลที่ได้คือ สามารถเขียนโปรแกรมควบคุมการทำงานสั่งให้ Inverter ทำงานตามคสั่งที่เราต้องการได้ แต่อาจจะไม่พร้อมกันเลยซะทีเดียว การใช้เวลาในการทดลอง ใช้เวลาประมาณ 1 สัปดาห์ ซึ่งเป็นเวลาที่นานเกินไป เพราะหลักการเขียนโปรแกรมนั้นคล้าย ๆ กันกับ การเขียนเพื่อเชื่อมต่อการทำงานของ PLC สองรุ่น จึงน่าจะใช้เวลาน้อยกว่านี้ แต่อย่างไรก็ตาม งานก็สามารถดำการผ่านไปได้ด้วยดี

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุป

หลังจากได้รับมอบหมายให้ปฏิบัติงานสหกิจศึกษาเป็นเวลา 4 เดือน ปรากฏว่างานที่ได้รับให้ปฏิบัติมีอยู่ 2 งานหลัก ๆ และสามารถสำเร็จลุล่วงไปได้ โดยงานแรกคือ การเขียนโปรโตคอลเชื่อมต่อระหว่าง PLC สองตัวที่ต่างรุ่นกัน ผลที่ได้คือสามารถเชื่อมต่อการสื่อสาร ของ Module สองตัวนี้ได้และได้ข้อมูลที่สามารถนำไปใช้งานต่อได้กับเครื่องมือจริง

ส่วนงานที่สองคือ การแปลงโปรแกรมจากภาษา Assembly เป็นภาษา Ladder Diagram เพื่อนำไปใช้งานต่อกับ PLC รุ่น Q ก็สามารถทำจนเสร็จลุล่วงได้ แต่ยังไม่สามารถนำไปใช้งานได้ เพราะยังต้องมีส่วนที่ต้องแก้ไขอีกนอกเหนือจาก ส่วนที่เป็นของ Assembly และทั้งในส่วนของตัว Assembly เอง ซึ่งในขั้นตอนนี้ยังต้องได้รับความช่วยเหลือจากพี่เลี้ยง เพราะต้องนำไปประยุกต์ใช้งานในขั้นต่อไป

นอกจากนี้แล้วยังมีงานอื่น ๆ ที่ได้รับมอบหมายให้ปฏิบัติ โดยส่วนมากจะเป็นการนำอุปกรณ์มาทดลองก่อนที่จะนำไปใช้และขายให้กับลูกค้านั่นเอง

5.2 สรุปผลการปฏิบัติงาน

5.2.1 การเขียนโปรแกรมเพื่อเชื่อมต่อการทำงานของ PLC รุ่น FX กับ PLC รุ่น A

การปฏิบัติงานชิ้นนี้เพื่อที่จะทำให้ PLC สองรุ่นสามารถเชื่อมต่อกันได้ และสามารถดูข้อมูลที่ Module สองตัวนี้ใช้รับ-ส่งกันได้เพื่อที่จะนำมาพัฒนาในการปฏิบัติงานขั้นต่อไปผลที่ได้ก็เป็นไปตามวัตถุประสงค์ที่ต้องการ แต่อาจจะช้ากว่ากำหนด

5.2.2 การแปลงโปรแกรมภาษา Assembly เป็นภาษา Ladder

ผลที่ได้จากการปฏิบัติงาน คือ ได้ Ladder Diagram ที่เขียนขึ้นโดยใช้ PLC รุ่น Q แต่ยังไม่สามารถนำไปใช้งานได้จริงเพราะยังต้องมีการแก้ไขตัวโปรแกรมอีกเนื่องจากตัวโปรแกรมที่ได้มีความซับซ้อนมาก และต้องใช้ความละเอียดมากในการตรวจสอบความถูกต้อง

5.2.3 เขียนโปรแกรมเชื่อมต่อและควบคุมการทำงานของ Inverter

ผลที่ได้จากการปฏิบัติงานชิ้นนี้ คือ สามารถใช้ชุดคำสั่ง สั่งให้ Inverter ทำงานได้ตามต้องการ โดยสั่งผ่าน PLC แต่ไม่สามารถที่จะให้ตัว Inverter ทำงานพร้อมกันได้หลายตัว ยังมีการหน่วงเวลาของโปรแกรม แต่ก็ไม่มากเท่าไรนัก ทำงานห่างกันประมาน 1-2 วินาที

5.3 ปัญหาและอุปสรรค

1.ปัญหาในการปฏิบัติงานโดยส่วนมากจะเป็นเกี่ยวกับการเขียนโปรแกรมและการใช้งานชุดคำสั่ง เพราะการที่เราจะเขียน code ให้ถูกรูปแบบตามที่ต้องการ จะต้องอาศัยความเข้าใจในการใช้งานตัวคำสั่งและโปรแกรมที่เขียนในหลักการทำงานและผลที่ได้ ซึ่งในส่วนนี้จะมีข้อผิดพลาดบ่อย ทำให้เสียเวลาในการแก้ไขและทำความเข้าใจ

2.ปัญหาอีกข้อหนึ่งในการปฏิบัติงานคือ การต่อสายเพื่อ Link ให้ อุปกรณสามารถทำงานได้ถูกต้อง ในการทำการทดลองต่าง ๆ ต้องมีการต่อสายเข้ากับอุปกรณ์ บางทีก็มีการต่อสายเข้าผิดขั้ว ทำให้อุปกรณ์ไม่ทำงานต้องมีการตรวจสอบทุกครั้งที่มีการต่อเสร็จแล้วหรือเมื่อรันโปรแกรมลงไปแล้วอุปกรณ์ไม่ทำงานนั่นเอง

จากที่กล่าวมาแล้วจะเห็นว่างานที่ปฏิบัติโดยส่วนใหญ่จะเป็นเกี่ยวกับการทดลองโปรแกรมกับอุปกรณ์ ซึ่งส่วนใหญ่แล้วปัญหาที่เกิดขึ้นนั้นจะมีอยู่สองอย่างหลักๆ คือ การเขียนโปรแกรม และการต่อสายซึ่งในการปฏิบัติงานก็เกิดปัญหาแบบนี้ขึ้นเกือบทุกครั้งเพราะแต่ละครั้งที่ทำการทดลองนั้น อุปกรณ์แต่ละชนิดจะมีการใช้งานที่ไม่เหมือนกันเท่าไรนัก จึงจำเป็นต้องเปิดคู่มือทุกครั้งในการใช้งานและการเปิดคู่มือก็เป็นอุปสรรคอีกอย่างหนึ่งเช่นกัน

5.4 ข้อเสนอแนะ

จากการเขียนโปรแกรมเสร็จแล้วในระดับหนึ่งพบว่าโปรแกรมยังไม่สามารถนำไปใช้งานได้ ต้องมีการแก้ไขทั้งในตัวโปรแกรมที่เป็น Assembly ที่ดัดแปลงมาแล้ว หรือ ตัวที่เป็น Ladder เอง เพราะเราได้นำ Code มาจาก PLC รุ่น A จึงต้องมีการปรับปรุงทั้งตัว register และ ตัวแปรแต่ละชนิดข้อมูลต่างๆที่นำมาใช้งาน เพื่อให้สามารถนำมาใช้เป็น PLC รุ่น Q ได้

และในการปฏิบัติงานนั้นสังเกตได้ว่าปัญหาที่เกิดขึ้นนั้นได้เกิดขึ้นเกือบทุกครั้งที่มีการปฏิบัติงานจึงควรที่จะมีความละเอียดรอบคอบมากกว่านี้ และในด้านโปรแกรมนั้น ควรจะมีการศึกษาเพิ่มเติมให้มากกว่านี้ เพื่อจะได้มีความเข้าใจในการใช้งานภาษาและคำสั่งต่าง ๆ ได้ อาจทำให้เกิดปัญหาน้อยลงได้ และในการเปิดคู่มือนั้นควรจะมีการจดบันทึกไว้ด้วยว่าเปิดคู่มืออะไร เพื่อใช้กับงานอะไร

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

รายการตาราง

รายการรูปประกอบ

รายการรูปประกอบ(ต่อ)

เอกสารอ้างอิง

เสกสัณห์ วัฒนะโชติ และคณะ,2550, รู้จัก PLC ใน 14 ชั่วโมง, พิมพ์ครั้งที่ 1,โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย : กรุงเทพฯ

ผศ.ธีรวัฒน์ ประกอบผล,2544,ระบบคอมพิวเตอร์และภาษาแอสแซมบลี,พิมพ์ครั้งที่ 3-6, บริษัท ที เอส บี โปรดักส์ จำกัด : กรุงเทพฯ

ส่งงานเรื่อง ภาษาคอมพิวเตอร์ .[ออนไลน์]
เข้าถึงได้จาก : http://images.noopk.multiply.multiplycontent.com/journal
(วันที่สืบค้นข้อมูล : 4 กรกฎาคม 2554).

รีจิสเตอร์ (Register). [ออนไลน์]
เข้าถึงได้จาก : http://www.thaiall.com/assembly/register.htm
(วันที่สืบค้นข้อมูล : 4 กรกฎาคม 2554).

คำสั่งภาษาแอสแซมบลี.[ออนไลน์]
เข้าถึงได้จาก : http://cs.udru.ac.th/nipon/4122702/instruction.php
(วันที่สืบค้นข้อมูล : 4 กรกฎาคม 2554).

ภาคผนวก

ภาคผนวก ก

อธิบายเพิ่มเติมเกี่ยวกับ PLC

1. ประวัติความเป็นมาของ PLC

เมื่อปี ค.ศ. 1968 บริษัท General Motors แห่งประเทศอเมริกาได้คิดค้นอุปกรณ์ควบคุมแบบใหม่เพื่อใช้ทดแทนวงจรควบคุมแบบเดิมที่ใช้กันอยู่ในโรงงานอุตสาหกรรมของบริษัท และ ในปี ค.ศ. 1969 PLC ก็ถูกผลิตขายในประเทศอเมริกาเป็นแห่งแรก

ค.ศ. 1977 Mitsubishi Electric ได้ผลิต PLC MELSEC PM เพื่อใช้เองและผลิตขายในเวลาต่อมา และผลิตรุ่นอื่นๆ ดังต่อไปนี้

- PLC MELSEC F และ MELSEC K
- MELSEC F พัฒนามาเป็น MELSEC FX Series ในปัจจุบัน
- MELSEC K พัฒนามาเป็น MELSEC A, MELSEC QnA และ MELSEC Q Series ในปัจจุบัน

รูปที่ ก.1 ชนิดของ PLC ของ Mitsubishi

การควบคุมการทำงานในรูปแบบนี้มีใช้กันอย่างหลากหลายตั้งแต่เครื่องใช้ไฟฟ้าในบ้านไปจนถึงฝ่ายการผลิตในงานอุตสาหกรรม เช่น ลิฟต์ บันไดเลื่อน ปั๊มน้ำ จนกระทั่งถึงเครื่องจักรในโรงงานอุตสาหกรรม เช่น เครื่องกลึง สายพานลำเลียง หม้อน้ำ และเครื่องกำเนิดไฟ ส่วนภาคงานอุตสาหกรรมในการควบคุมเครื่องจักรกลทางไฟฟ้ามีการควบคุมด้วยอุปกรณ์หลากหลายรูปแบบดังเช่นในสมัยก่อนใช้รีเลย์ในการควบคุมแต่ในปัจจุบันนิยมใช้โปรแกรมเมเบิ้ลลอจิกคอนโทรลเลอร์ (PLC) ในการควบคุมการทำงานสมัยก่อนประกอบไปด้วยอุปกรณ์ดังต่อไปนี้

1.1 ฟิวส์ (Fuses)

เป็นอุปกรณ์ในการป้องกันวงจรทั้งวงจรควบคุม (Control circuit) และวงจรกำลัง (Power circuit) ฟิวส์ทำงานตัดวงจรโดยการหลอมละลายมีทั้งชนิดตัดวงจรทันที และแบบหน่วงเวลา

รูปที่ ก.2 ฟิวส์ชนิดต่างๆ

1.2 คอนแทกเตอร์ (Contactor)

บางทีเรียก “แมกเนติกคอนแทกเตอร์” เป็นอุปกรณ์ที่ทำงานด้วยแรงดึงดูดแม่เหล็กที่เกิดจากคอล์ยและแกนเหล็กอาร์เมเจอร์ทำให้หน้าสัมผัสเคลื่อนที่มาแตะกันและส่งผ่านกำลังไฟฟ้าเข้าสู่มอเตอร์ นำไปใช้สำหรับงานควบคุมไฟฟ้ากำลัง เช่น มอเตอร์ไฟฟ้า หลอดไฟสัญญาณ เป็นต้น

รูปที่ ก.3 รูปคอนแทกเตอร์

1.3 รีเลย์ (Relay)

อุปกรณ์อิเล็กทรอนิกส์ ที่ทำหน้าที่ ตัด-ต่อวงจร คล้ายกับสวิตซ์ โดยใช้หลักการหน้าสัมผัส และการที่จะให้มันทำงานก็ต้องจ่ายไฟให้มันตามที่กำหนดเพราะเมื่อจ่ายไฟให้กับตัวรีเลย์มันจะทำให้หน้าสัมผัสติดกัน กลายเป็นวงจรปิด และตรงกันข้าม ทันทีที่ไม่ได้จ่ายไฟให้มัน มันก็จะกลายเป็นวงจรเปิด หลักการทำงานคล้ายกับ ขดลวดแม่เหล็กไฟฟ้าหรือโซลินอยด์ (Solenold) รีเลย์ใช้ในการควบคุมวงจรไฟฟ้าได้อย่างหลากหลาย

รูปที่ ก.4 รีเลย์ชนิดต่างๆ

1.4 สวิตช์ปุ่มกด หรือสวิตช์พุชบัทตอน (Push Button)

เป็นสวิตช์ที่เมื่อกดแล้วปล่อยมือ สวิตช์จะเด้งกลับคืนโดยไม่ค้างตำแหน่งไว้ที่เดิม หรือเรียกเป็นภาษาอังกฤษว่า “Momentary Switch” สวิตช์พุชบัทตอนนี้มีทั้งชนิดหน้าสัมผัส ปกติเปิด (Normally Open) และชนิดหน้าสัมผัสปกติปิด (Normally Close)

รูปที่ ก.5 สวิตช์ปุ่มกด หรือสวิตช์พุชบัทตอนชนิดต่างๆ

1.5 ลิมิตสวิตช์ (Limit Switch)

เป็นสวิตช์ควบคุมตำแหน่ง ทำหน้าที่เป็นคอนแทค ในการควบคุมการทำงานของมอเตอร์

- ใช้วงจรคอนโทรล ควบคุมตำแหน่งของเครื่องจักร อุปกรณ์
- เป็นสวิตช์ควบคุมอุปกรณ์
- รับกระแสไฟฟ้าได้ทั้งไฟตรงและไฟสลับ

การนำไปใช้งานมีให้เลือกหลายแบบ หลายขนาด หน้าสัมผัส ใช้กับไฟเอซี หรือ ดีซี มีทั้งแบบ NO และ NC

รูปที่ ก.6 ลิมิตสวิตช์ชนิดต่างๆ

1.6 รีเลย์ตั้งเวลา (Timer Relay)

เป็นนาฬิกาตั้งเวลา ทำหน้าที่ตั้งเวลาการทำงานของวงจร ใช้คอนแทค ในการปิด เปิด ควบคุมการทำงานของวงจร มอเตอร์

- ใช้ในวงจรคอนโทรล ควบคุมเวลาการทำงานของเครื่องจักร, อุปกรณ์
- เป็นสวิตช์ควบคุมอุปกรณ์
- รับกระแสไฟฟ้าได้ทั้ง ไฟตรงและไฟสลับ

รีเลย์ตั้งเวลา ที่ใช้อยู่ทั่วไป มี 2 แบบ

1. แบบ Time On Delay คือ หน่วงเวลาหลังจากจ่ายไฟเข้าคอยล์
2. แบบ Time Off Delay คือ หน่วงเวลาหลังจากหยุดจ่ายไฟเข้าคอยล์

การนำไปใช้งาน มีให้เลือกหลายแบบ หลายขนาด หน้าสัมผัส ใช้กับไฟ AC หรือ DC มีทั้งแบบ NO และ NC

รูปที่ ก.7 รูปรีเลย์ตั้งเวลา

1.7 เซอร์กิตเบรกเกอร์ (Circuit Breaker)

เซอร์กิตเบรกเกอร์ หรือเรียกชื่อย่อว่า ซี บี เป็นตัวตัดต่อวงจรไฟฟ้าภายในบ้าน หรือภายในอาคารสำนักงานได้อย่างฉับพลัน เพราะตัวเซอร์กิตเบรกเกอร์นี้ ประกอบด้วยชุดอุปกรณ์ความร้อน และอุปกรณ์แม่เหล็กไฟฟ้า เป็นตัวควบคุมให้ตัดวงจรโดยอัตโนมัติเมื่อวงจรไฟฟ้านั้น ๆ ลัดวงจร ก่อนจะเกิดอันตราย หรือก่อนทำให้อาคารเกิดไฟไหม้ได้

รูปที่ ก.8 เซอร์กิตเบรกเกอร์ชนิดต่าง ๆ

1.8 เคาน์เตอร์ (Countor)

เคาน์เตอร์ หรือตัวนับเป็นอุปกรณ์ที่มีไว้เพื่อเป็นตัวนับการทำงาน การใช้งานตัวนับในที่นี้เมื่อกำหนดจำนวนค่าในตัวเคาน์เตอร์แล้ว มีสัญญาณเข้า 1 ครั้ง เคาน์เตอร์ก็จะนับ 1 เมื่อครบตามที่เรากำหนดแล้ว หน้าสัมผัสก็จะเปลี่ยนจาก NC เป็น NO และจาก NO เป็น NC

รูปที่ ก.9 เคาน์เตอร์ชนิดต่างๆ

1.9 หลอดไฟสัญญาณ (Pilot Lamp)

เป็นหลอดไฟใช้แสดงสถานะการทำงานของวงจรมีหลายสีให้เลือก เช่น แดง เหลือง น้ำเงิน เขียว โดยการเปลี่ยนฝาครอบพลาสติกด้านหน้า บางชนิดเป็นแบบรวมอยู่กับสวิตช์ปุ่มกด หรือมีหม้อแปลงเล็กในตัวสำหรับแปลงแรงดัน 220V ให้เป็นแรงดันต่ำประมาณ 6V

รูปที่ ก.10 รูปหลอดไฟสัญญาณชนิดต่างๆ

2. ปัญหาของการใช้อุปกรณ์ควบคุมสมัยก่อน

การใช้รีเลย์ในการควบคุม

อุปกรณ์ควบคุมมีขนาดใหญ่ทำให้สิ้นเปลืองพื้นที่มากถ้าติดตั้งในเครื่องจักรจะทำให้มีขนาดใหญ่ การตรวจซ่อมทำด้วยความลำบากและยุ่งยากเนื่องจากมีการเดินสายไฟเป็นจำนวนมากเพราะมีอุปกรณ์จำนวนมาก ทำให้เสียเวลาซ่อมบำรุงมากเมื่อเกิดการขัดข้องและถ้ามีเงื่อนไขเปลี่ยนแปลงต้องรื้อสายไฟแล้วเดินสายไฟใหม่ทั้งหมดทำให้เสียเวลามากการทำงานของอุปกรณ์ควบคุมช้าเพราะมีส่วนของรีเลย์มาก ตัวควบคุมขาดความเสถียร เพราะใช้หน้าคอนแทคเมื่อใช้ไปนานๆ จะมีปัญหาเรื่องเขม่าเนื่องจากการอาร์คตอนเปิด – ปิด ถ้ากระบวนการผลิตใหญ่ความซับซ้อนมากต้องใช้อุปกรณ์มากทำให้ราคาสูง

ยุคต่อมาใช้อุปกรณ์อิเล็กทรอนิกส์ประกอบเป็นอุปกรณ์ควบคุม

ปัญหาที่ไม่สามารถแก้ไขได้

1) เมื่อมีการเปลี่ยนแปลงแก้ไขการควบคุมต้องทำวงจรใหม่
2) กระบวนการผลิตใหญ่ซับซ้อนมากต้องใช้อุปกรณ์มากทำให้ความซับซ้อนสูง
3) การตรวจเช็คยุ่งยาก เพราะหาช่างผู้ชำนาญเฉพาะด้านอิเล็กทีอนิกส์ยาก

ยุคต่อมาใช้ Microprocessor ควบคุม

ปัญหาที่ตามมา

1) ขาดบุคลากรที่มีความชำนาญงานด้าน Microprocessor
2) ใช้ร่วมกับวงจรอิเล็กทรอนิกส์ที่ต้องออกแบบขึ้นมาเฉพาะแต่ละเครื่องจักรทำให้การตรวจเช็คยุ่งยาก หาอุปกรณ์ทดแทนยาก

จุดประสงค์ในการออกแบบ PLC

1) เป็นระบบอิเล็กทรอนิกส์สมัยใหม่
2) มีการโปรแกรมใหม่ได้เหมือนกับคอมพิวเตอร์
3) ทนทานต่อสภาพแวดล้อมในอุตสาหากรรม
4) โปรแกรมใช้ง่าย
5) ดูแลรักษาได้ง่าย ซ่อมบำรุงได้ง่าย
6) ใช้งานใหม่ได้แม้เปลี่ยนรูปแบบการทำงาน
7) ลดเวลาในการหยุดเครื่องจักร
8) สามารถขยายระบบได้ง่าย

3. PLC คืออะไร

PLC ย่อมาจาก Programmable Logic Controller

Programmable Logic Controller หรือ PLC เป็นอุปกรณ์ควบคุมการทำงานของเครื่องจักรในโรงงานอุตสาหกรรมชนิดหนึ่งที่สามารถโปรแกรมการควบคุมให้เป็นไปตามที่ต้องการได้โดยมีหน่วยความจำในการเก็บโปรแกรมสำหรับควบคุมการทำงานของอุปกรณ์ต่าง ๆ ที่ต่อกับขั้วเข้าและขั้วออกขอก PLC การคิดค้น PLC ขึ้นมาเพื่อจุดประสงค์ในการนำมาใช้แทนวงจรควบคุมไฟฟ้าแบบรีเลย์ในโรงงานอุตสาหกรรม ซึ่งวงจรรีเลย์ดังกล่าวมีความยุ่งยากในการออกแบบ มีขนาดของวงจรที่ค่อนข้างใหญ่กินเนื้อที่ในการติดตั้งมาก และต้องเดินสายไฟจำนวนมาก การนำเอา PLC เข้ามาทดแทนสามารถลดปัญหาดังกล่าวลงได้

การควบคุมซีเคว้นซ์ คือ การควบคุมลำดับการทำงานของอุปกรณ์ในระบบให้ทำงานตามเงื่อนไขหรือโปรแกรมที่ได้กำหนดไวแล้ว สัญญาณในการควบคุมส่วนใหญ่จะเป็นสัญญาณ ปิด-เปิด ให้อุปกรณ์ เช่น มอเตอร์ โซลินอยด์วาล์ว หลอดแสดงผล ในการควบคุมการทำงานของอุปกรณ์ นอกจากจะพิจารณาเงื่อนไขการทำงานแล้วยังพิจารณาถึงช่วงเวลาในการทำงานด้วย

ในปัจจุบัน PLC รุ่นใหม่ๆ มีความสามารถที่เพิ่มขึ้นและมีราคาที่ถูกลง ดังนั้นเราจะเห็นว่า PLC ได้รับความนิยมมากขึ้นในโรงงานอุตสาหกรรม โดยสามารถใช้ควบคุมเครื่องจักรในโรงงานอุตสาหกรรมให้ทำงานอย่างอัตโนมัติ, ใช้ปรับปรุงประสิทธิภาพของการทำงานของเครื่องจักร และได้มีการนำมาใช้ประโยชน์กับระบบควบคุมทั่วไปอีกด้วย เช่น ใช้ควบคุมไฟเขียวไฟแดงตามแยกหรือการควบคุมอุปกรณ์ในบ้านเรือนเป็นต้น

3.1 ข้อดีของ PLC

การใช้ PLC แทนวงจรรีเลย์ซีเคว้นซ์มีข้อดีดังนี้

1) ประหยัดค่าใช้จ่าย ถ้าใช้รีเลย์ ไทเมอร์ และคอนแทกเตอร์เกินกว่า 10 ตัวขึ้นไป

2) ลดเวลาในการออกแบบวงจรและประกอบตู้ควบคุม สะดวกในการเปลี่ยนแปลงแก้ไขวงจร

3) มีขนาดเล็กและเป็นมาตรฐาน เมื่อเทียบกับวงจรรีเลย์ซีเคว้นซ์ที่ต้องใช้อุปกรณ์มากกว่า

4) ความเชื่อถือของระบบดีขึ้น ความเชื่อถือของ PLC ดีกว่าวงจรรีเลย์มาก เนื่องมาจากไม่ต้องเป็นห่วงเรื่อง หน้าสัมผัส สายหลุด เหมือนกับวงจรรีเลย์

5) การบำรุงรักษาง่าย ใน PLC จะมีโปรแกรมการตรวจสอบตัวเอง สามารถวิเคราะห์ความผิดปกติของเครื่องได้ง่าย

3.2โครงสร้างของ PLC

โครงสร้างของ PLC มีลักษณะคล้ายกับระบบ Microcomputer หรือ Microcontroller โดยมีองค์ประกอบที่สำคัญดังนี้

1. วงจรอินพุต
2. วงจรเอาท์พุต
3. หน่วยความจำ
4. หน่วยคำนวณและประมวลผล

3.2.1 วงจรอินพุต

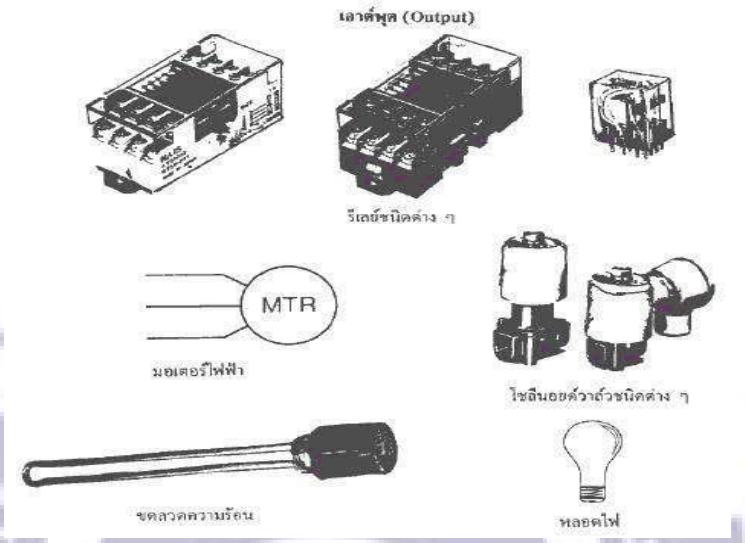
ส่วนอินพุตของ PLC เป็นส่วนที่ใช้ในการเชื่อมต่ออุปกรณ์ภายนอกกับ CPU เพื่อรับสัญญาณจากอุปกรณ์ภายนอกนั้นมาเป็นเงื่อนไขในการควบคุม

อุปกรณ์ควบคุมที่จะนำมาต่อเป็นอินพุต ให้กับ PLC มีหลากหลาย และหนึ่งในจำนวนนั้นมีลักษณะเป็นสวิตช์ เช่น พวก สวิตช์ลำแสง สวิตช์ปุ่มกดแบบต่าง ๆ เป็นต้น

รูปที่ ก.11 รูปตัวอย่างอินพุตของ PLC

3.2.2 วงจรเอาท์พุต

ส่วนเอาท์พุตของ PLC เป็นส่วนที่ใช้ในการเชื่อมต่อกับอุปกรณ์ภายนอกทางด้านขั้วขาออกเพื่อส่งสัญญาณให้กับอุปกรณ์ภายนอก อุปกรณ์เอาท์พุตมีหลากหลายไม่ว่าจะเป็น รีเลย์ หลอดแสดงผลโซลินอยด์วาล์ว คอนแทกเตอร์ ฯลฯ



รูปที่ ก.12 รูปตัวเอาท์พุตของ PLC

3.2.3 หน่วยความจำ (Memory)

ทำหน้าที่เก็บรักษาโปรแกรมและข้อมูลที่ใช้ในการทำงาน ซึ่งแบ่งออกได้ดังนี้

RAM หรือ SRAM

หน่วยความจำประเภทนี้ต้องมีแบตเตอรี่ไว้เลี้ยงข้อมูลเมื่อเกิดไฟดับ เป็นหน่วยความจำมาตรฐานใน PLC ส่วนใหญ่ และมีในรูป Memory cassette หรือ Memory card สำหรับ PLC บางรุ่น การอ่านและการเขียนข้อมูลลงในหน่วยความจำชนิดนี้ทำได้ง่ายจึงเหมาะกับงานที่มีการเปลี่ยนแปลงหรือแก้ไขข้อมูลบ่อยๆ

ROM หน่วยความจำที่ไม่ต้องมีแบตเตอรี่ไว้เลี้ยงข้อมูลเมื่อไฟดับ

- EPROM การเขียนข้อมูลลงในหน่วยความจำชนิดนี้ทำได้ยาก ถ้าต้องการแก้ไขหรือเปลี่ยนแปลงโปรแกรมต้องใช้อุปกรณ์พิเศษในการเขียนและลบโปรแกรม จึงเหมาะกับงานที่ไม่ต้องแก้ไขโปรแกรมบ่อยๆ ปัจจุบันไม่มีการผลิตหน่วยความจำประเภทนี้สำหรับ PLC รุ่นใหม่ ๆ

- EEPROM หรือ E^2PROM การเขียนหรือแก้ไขโปรแกรมลงในหน่วยความจำชนิดนี้ทำได้ง่ายโดยไม่ต้องใช้เครื่องมือพิเศษและไม่ต้องการแบตเตอรี่ไว้คอยเลี้ยงข้อมูลใน PLC มีหลายรุ่นที่มี EEPROM ติดมาเป็นหน่วยความจำมาตรฐานในการเก็บโปรแกรมหรือสำรองโปรแกรม

- FLASH ROM การเขียนหรือแก้ไขโปรแกรมลงในหน่วยความจำชนิดนี้ทำได้ง่ายโดยไม่ต้องใช้เครื่องมือพิเศษ ปัจจุบันมีใน Memory Card ของ PLC รุ่น Q และติดมาเป็นหน่วยความจำมาตรฐานใน PLC บางรุ่น

- ATA ROM มีความจุสูงกว่าหน่วยความจำประเภท ROM แบบอื่นๆ ปัจจุบันมีใน Memory Card ของ PLC รุ่น Q การเขียนหรือแก้ไขข้อมูลในหน่วยความจำชนิดนี้ทำได้โดยเครื่องคอมพิวเตอร์หรือจากคำสั่งในโปรแกรม PLC

3.2.4 หน่วยประมวลผล หรือ CPU

หน่วยประมวลผล หรือ CPU จะทำงานตามโปรแกรมในหน่วยความจำ โดยอ่านข้อมูลจากวงจร อินพุต แล้วทำการประมวลผลข้อมูลตามโปรแกรมควบคุมที่ป้อนเข้าไป เพื่อหาสัญญาควบคุมที่ถูกต้องและส่งสัญญาณควบคุมออกไปที่วงจรเอาท์พุตภายใน PLC เหมือนกับมีรีเลย์ ไทเมอร์ เคาน์เตอร์ และอุปกรณ์ควบคุมอื่นๆ อีกมากมาย เนื่องจาก PLC เป็นอุปกรณ์อิเล็กทรอนิกส์ ซึ่งภายในมีไมโครคอมพิวเตอร์เป็นองค์ประกอบสำคัญ แต่ในแง่ของผู้ใช้ไม่จำเป็นต้องมีความรู้เกี่ยวกับไมโครคอมพิวเตอร์เลยก็ได้ คิดเพียงว่าภายใน PLC เป็นที่รวมของรีเลย์ ไทเมอร์ เคาน์เตอร์ และอุปกรณ์อื่นๆ ก็พอแล้ว แต่เราจะต้องสร้างโปรแกรมป้อนเข้าไปใน PLC โดยใช้หน้าสัมผัสขาเข้า รีเลย์ช่วย ไทเมอร์ และอุปกรณ์ต่าง ๆ ภายในประกอบเป็นวงจรควบคุมที่ต้องการ เมื่อป้อนโปรแกรมเข้าไปในหน่วยความจำของ PLC แล้ว กดปุ่มให้ PLC เริ่มทำงานตามโปรแกรมนั้น PLC จะทำงานเหมือนวงจรรีเลย์ ที่เราป้อนเข้าไปทุกประการ

4. ตัวอย่างการเลือก PLC เพื่อนำไปใช้งาน

เนื่องจากในปัจจุบันนี้ผู้ผลิตสินค้าได้ทำการผลิต PLC ออกมาจำหน่ายในท้องตลาดมากมายหลายประเภท โดยแต่ละประเภทนั้นก็จะแบ่งย่อยออกเป็นรุ่นได้อีกหลายรุ่น ซึ่ง PLC แต่ละรุ่นนั้นจะมีคุณสมบัติการทำงาน และความสามารถของหน่วยประมวลผลกลาง (CPU) ซึ่งแตกต่างกันออกไป ดังนั้นการที่เราจะเลือก PLC เพื่อนำไปใช้งานนั้นจำเป็นที่จะต้องศึกษาถึงข้อมูล และคุณสมบัติของ PLC แต่ละรุ่น แต่ละแบบก่อนการนำเอา PLC นั้นไปใช้งานเพื่อจะให้เกิดประโยชน์สูงสุดต่อการทำงานและคุ้มค่ากับราคาที่ต้องจ่ายออกไป

จากที่กล่าวไปข้างต้นว่าก่อนที่เราจะเลือก PLC เพื่อนำเอาไปใช้งานนั้นจำเป็นที่จะต้องศึกษาถึงข้อมูล และคุณสมบัติพื้นฐานโดยทั่วไปของ PLC แต่ละประเภทก่อน โดยที่เราจะสามารถหาข้อมูลได้จากผู้ขายสินค้านั่นเอง ในที่นี้จะยกตัวอย่างข้อมูลพื้นฐานที่เราจะนำมาวิเคราะห์ก่อนที่จะเลือก PLC เพื่อนำไปใช้งานได้ดังต่อไปนี้

4.1 PLC FX-SERIES

PLC ประเภทนี้เป็นแบบขนาดเล็ก และง่ายต่อการนำไปใช้งาน เนื่องจากเป็น PLC ที่มีทั้งภาคอินพุต เอาท์พุต เพาเวอร์ซัพพลาย และหน่วยความจำ ตลอดจนหน่วยประมวลผลกลาง ซึ่งอยู่ภายในตัวเดียวกันทั้งหมด เราเรียกชนิดของ PLC รุ่น FX SERIES นี้ว่า PLC แบบ MICRO TYPE โดยที่หน่วยประมวลผลกลาง (CPU) ของ PLC ชนิดนี้สามารถจะรองรับอินพุต และเอาท์พุตที่นำมาต่อใช้งานบนตัวมันได้สูงสุดไม่เกิน 256 จุด

การแบ่งรุ่นของ FX-SERIES

ในปัจจุบัน PLC FX-SERIES สามารถแบ่งออกเป็นรุ่นใหญ่ ๆ ได้ทั้งหมด 3 รุ่นด้วยกันโดยแต่ละรุ่นนั้นจะมีขีดความสามารถของการรองรับอินพุต เอาท์พุต จำนวนความจุของโปรแกรม และจำนวนของอุปกรณ์ภายในที่นำมาใช้ในการสร้างโปรแกรมที่ไม่เท่ากันโดยจะแสดงได้ดังต่อไปนี้

ตารางที่ ก.1 แสดงขีดความสามารถของ PLC FX-SERIES

PLC FX-SERIES	I/O max (Points)	Program (steps)	Data register (Points)
1. FX1S	30	2K	256
2. FX1N	128	8K	8000
3. FX3U	256	64K	8000

จากตาราง จะพบว่าปัจจุบัน PLC FX-SERIES สามารถแบ่งออกเป็นรุ่นใหญ่ ๆ ได้ทั้งหมด 3 รุ่นด้วยกัน โดยแต่ละรุ่นจะมีขีดความสามารถของหน่วยประมวลผลกลาง หรือ CPU ไม่เท่ากัน

ยกตัวอย่างเช่น ในเครื่องจักรเครื่องหนึ่ง มีการใช้งาน PLC รุ่น FX1N อยู่ ซึ่งใช้งานในการรองรับอินพุต และเอาท์พุตอยู่ที่ 110 Point เขียนโปรแกรมไปได้ 7500 Step ต่อมามีความต้องการที่จะปรับปรุงเครื่องจักรให้มีประสิทธิภาพดีมากยิ่งขึ้น โดยต้องเขียนโปรแกรม PLC เพิ่มเติมโดยอาจจะเพิ่มจำนวนโปรแกรมมากกว่า 8K Step และเพิ่มจำนวนอินพุตและเอาท์พุตจากเดิมที่ใช้อยู่เพิ่มเป็นคือ 200 Point ดังนั้นจะเลือก PLC รุ่น FX1N มาใช้งานต่ออีกไม่ได้เพราะขีดความสามารถไม่พอกับความต้องการใช้งาน จึงจำเป็นต้องเลือกใช้งาน PLC รุ่นที่มีขีดความสามารถที่สูงกว่ามาใช้งานแทน คือ เลือกเป็นรุ่น FX3U ซึ่งจะสามารถรองรับอินพุต และเอาท์พุตได้สูงสุดถึง 256 Point เขียนโปรแกรมได้ 64K Step เป็นต้น

นอกจากจะพิจารณาความสามารถในการรองรับอินพุตและเอาท์พุตแล้วควรจะพิจารณาขีดความสามารถในเรื่องอื่นๆเพิ่มเติมด้วย เช่น จำนวนของแหล่งจัดเก็บข้อมูลที่เป็นตัวเลข (Data Register) ว่ามีเพียงพอสำหรับจัดเก็บข้อมูลหรือไม่ หรือจำนวนของอุปกรณ์ภายในที่จะนำเอามาช่วยเขียนโปรแกรมเช่น รีเลย์ช่วย,ไทเมอร์ เพียงพอสำหรับนำมาเขียนโปรแกรมหรือไม่ ดังนั้นการที่เราจะนำ PLC FX-SERIES ไป ใช้งานนั้นจะต้องคำนึงถึงสิ่งที่กล่าวมาข้างต้นด้วยว่าเหมาะสมกับงานนั้นๆ มากน้อยเพียงใด แต่ถ้าเราต้องการใช้งานอินพุตและเอาท์พุตรวมกันทั้งหมด มากกว่า 256 จุดขึ้นไป เราจะต้องเลือก PLC ที่มีขนาดใหญ่กว่า PLC FX-SERIES ซึ่งจะกล่าวถึงในหัวข้อถัดไป

4.2 PLC Q-SERIES

PLC รุ่น Q-SERIES ในปัจจุบันเป็นที่นิยมใช้อย่างมากในหมู่นักออกแบบเครื่องจักร (MAKER MACHINE) เนื่องจาก PLC รุ่นนี้มีความสามารถในการรองรับการต่อใช้งานในส่วนของภาคอินพุตและภาคเอาท์พุตมากกว่า PLC รุ่น FX-SERIES อีกทั้งมีความเร็วสูงในการประมวลผลโปรแกรมในหน่วยประมวลผลกลาง (CPU) อีกด้วย เราเรียกชนิดของ PLC รุ่น Q-SERIES นี้ว่า PLC แบบ MODULAR TYPE เนื่องจาก PLC รุ่นนี้จะมีส่วนประกอบของโครงสร้าง PLC ที่แยกออกจากกันได้ โดยเป็นลักษณะของ MODULE เพื่อให้ ง่ายต่อการบำรุงรักษา เช่น CPU MODULE, INPUT MODULE, OUTPUT MODULE, POWER SUPPLY MODULE และอื่นๆ ดังนั้นเมื่อ ต้องการนำ PLC รุ่นนี้ไปใช้งานจะต้องนำองค์ประกอบแต่ละส่วนมาผสมผสานกันก่อนนำไป ติดตั้งและใช้งาน

การแบ่งรุ่นของ CPU Q-SERIES

ในปัจจุบันหน่วยประมวลผลกลางหรือที่เรานิยมเรียกกันว่า CPU นั้นแบ่งออกได้เป็นหลายรุ่นด้วยกัน ซึ่งในแต่ละรุ่นนั้นก็จะมีความแตกต่าง และมีความสามารถของ CPU ที่ไม่เท่ากันอีกด้วย ดังนั้นก่อนที่เราจะเลือก CPU เพื่อนำไปใช้งานนั้นเราจำเป็นจะต้องศึกษาถึงคุณสมบัติและข้อมูลพื้นฐานของ CPU ในแต่ละรุ่นเสียก่อน โดยข้อมูลพื้นฐานดังกล่าวนี้สามารถสอบถามได้จากผู้จำหน่ายโดยตรง

หัวข้อนี้จะแสดงถึงข้อมูลพื้นฐานทั่วๆไป และขีดความสามารถของ CPU รุ่น Q-SERIES เบื้องต้น เพื่อเป็นข้อมูลสำหรับการใช้พิจารณาการเลือกใช้งานของ PLC รุ่น Q-SERIES ต่อไป

ตารางที่ ก.2 ตารางแสดงขีดความสามารถของ PLC Q Basic และ High Performance

C PU Type Q-SERIES	I/O max (Points)	Program (Steps)	Data Register (Points)	Device Memory (Words)
BASIC MODEL				
1.Q00JCPU	256	8K	11136	18K
2.Q00CPU	1024	8K	11136	18K
3.Q01CPU	1024	14K	11136	18K
HIGH PERFORMANCE MODEL				
1.Q02CPU	4096	28K	12288	29K
2.Q02HCPU	4096	28K	12288	29K
3.Q06HCPU	4096	60K	12288	29K
4.Q12HCPU	4096	124K	12288	29K
5.Q25HCPU	4096	252K	12288	29K

CPU Type Q-SERIES	I/O max (Points)	Program (Steps)	Data Register (Points)	Device Memory (Words)

UNIVERSAL MODEL Q CPU

1.Q02UCPU	2048	20K	12288	29K
2.Q03UDCPU	4096	30K	12288	29K
3.Q04UDHCPU	4096	40K	12288	29K
4.Q06UDHCPU	4096	60K	12288	29K

จากตาราง จะพบว่าปัจจุบัน CPU Q-SERIES สามารถแบ่งออกเป็น Model ใหญ่ๆ ได้ทั้งหมด 2 Model ด้วยกันคือ แบบ BASIC MODEL และแบบ HIGH PERFORMANCE MODEL ซึ่งในแต่ละ MODEL ก็จะแบ่งออกเป็นรุ่นย่อยๆ ได้อีก หลายรุ่นด้วยกัน ซึ่งในที่นี้จะพบว่า CPU ของ PLC รุ่น Q-SERIES แบบ HIGH PERFORMANCE MODEL จะมีความสามารถในการรองรับอินพุตและเอาท์พุต ตลอดจนความสามารถในการรองรับกานเขียนโปรแกรม (Program Capacity) และการเขียนข้อมูลลงในตัวที่ใช้สำหรับเก็บข้อมูล(Data Register)มากกว่าCPUของPLCรุ่น
Q-SERIES แบบ BASIC MODEL

ยกตัวอย่างเช่น เครื่องจักรเครื่องหนึ่ง มี PLC รุ่น Q-SERIES แบบ BASIC MODEL รุ่น Q00 CPU ใช้ในการรองรับอินพุตและเอาท์พุตอยู่ที่ 950 Point เขียนโปรแกรมไปได้ 7500 Step ต่อมามีการพัฒนาเครื่องจักรให้มีประสิทธิภาพเพิ่มขึ้นให้สามารถรองรับอินพุตและเอาท์พุต เพิ่มเป็น 1000 Point และต้องการเขียนโปรแกรมให้ได้มากกว่า 8K Step ดังนั้นจะเลือก PLC รุ่น Q00 CPU มาใช้งานต่ออีกไม่ได้จึงจำเป็นต้องเลือกใช้งาน PLC รุ่นที่มีขีดความสามารถที่สูงกว่า คือเลือกเป็นรุ่น Q01CPU ซึ่งจะสามารถรองรับอินพุตและเอาท์พุตได้เท่ากันคือ ไม่เกิน 1024 Point แต่จะเขียนโปรแกรมได้สูงสุดถึง 14K Step เป็นต้น

หรือ มี PLC รุ่น Q-SERIES แบบ HIGH PERFORMANCE MODEL รุ่น Q02HCPU อยู่ซึ่งจะสามารถรองรับอินพุต และเอาท์พุตได้สูงสุดไม่เกิน4096 Point เขียนโปรแกรมได้ 28K Step แต่ในขณะที่ความต้องการใช้งาน PLC จริง ต้องการรุ่นที่สามารถรองรับอินพุตและเอาท์พุตได้สูงสุดไม่เกิน 4096 Point แต่ต้องการเขียนโปรแกรมให้ได้มากกว่า 28K Step ดังนั้นจะเลือก PLC รุ่น Q02HCPU มาใช้งานไม่ได้จึงจำเป็นต้องเลือกใช้งาน PLC รุ่นที่มีขีดความสามารถที่สูงกว่า คือเลือกเป็นรุ่น Q06HCPU หรือรุ่นที่สูงกว่ามาใช้งานซึ่งจะสามารถรองรับอินพุต และเอาท์พุตได้เท่ากันคือไม่เกิน 4096 Point แต่จะเขียนโปรแกรมได้มากกว่า 28K Step เป็นต้น

ดังนั้นจะสังเกตุได้ว่า PLC รุ่น Q-SERIES แบบ HIGH PERFORMANCE MODEL จะสามารถรองรับอินพุตและเอาท์พุตได้เท่ากันคือไม่เกิน 4096 Point และมีจำนวนของอุปกรณ์ภายใน ที่จะนำเอามาช่วยเขียนโปรแกรม เช่น รีเลย์ช่วย, ไทเมอร์ ที่เท่ากันแต่จะมีความสามารถในการเขียนโปรแกรม (จำนวน Step Program) ที่ไม่เท่ากัน ดังนั้นการที่เราจะเลือกเอา CPU แต่ละรุ่นแต่ละแบบไปใช้งานนั้นจะต้องคำนึงถึงสิ่งที่กล่าวมาข้างต้นด้วยว่า เหมาะสมกับงานนั้นๆ มากน้อยเพียงใด ทั้งนี้เพื่อความเหมาะสมและทำให้เกิดประโยชน์สูงสุดกับผู้ใช้ต่อไป

จากที่ได้กล่าวมาข้างต้นว่า CPU Q-SERIES มีขีดความสามารถที่อยู่เหนือกว่า PLC FX-SERIES อยู่มาก ทั้งในด้านการรองรับอินพุต และเอาท์พุต ตลอดจนความเร็วในการประมวลผลโปรแกรมและการเก็บค่าข้อมูลแบบตัวเลขซึ่งมีการเก็บที่มากกว่า นอกจากนั้นแล้ว PLC MITSUBISHI ยังถูกออกแบบมาเพื่อใช้รองรับการเชื่อมต่อ PLC แบบ โครงข่าย (NETWORK) อีกด้วยในปัจจุบันนี้โรงงานอุตสาหกรรม ในประเทศไทย หลายโรงงานได้เลือกการต่อใช้งาน PLC เป็นแบบโครงข่าย (NETWORK) เพื่อใช้สำหรับโอนถ่ายข้อมูลระหว่างตัว CPU ตัวหนึ่งไปสู่ CPU อีกตัวหนึ่ง และยังสามารถใช้อุปกรณ์พิเศษภายนอกมาต่อเชื่อมการทำงานในระบบโครงข่ายของ PLC ได้อีกด้วย ซึ่งการต่อใช้งานในระบบโครงข่าย (NETWORK) ของ PLC MITSUBISHI นั้นสามารถทำได้หลายแบบด้วยกันดังที่จะกล่าวต่อไป

ภาคผนวก ข

วิธีการใช้โปรแกรม Gx-Developer และการเขียน Ladder

การนำซอฟต์แวร์ Gx-Developer มาใช้งานและการเขียยนโปรแกรม Ladder

หลังจากที่เราได้รู้วิธีการติดตั้ง ซอฟต์แวร์ กันไปแล้ว มาส่วนนี้เราจะอธิบายถึงวิธีการใช้งานเพื่อนำไปใช้เขียนโปรแกรม ในขั้นแรกเราควรจะมาเรียนรู้ส่วนประกอบและเครื่องมือที่ช่วยสำหรับใช้งานในโปรแกรมกันก่อน เพื่อที่จะมีความเข้าใจในการทำโปรแกรมต่อไป

1. การใช้งานโปรแกรม Gx-Developer

การใช้งานโปรแกรม Gx-Developer ให้เลือกที่ Program => MELSOFT Application => เลือกที่ Gx-Developer

รูปที่ ข.1 การเข้าสู่โปรแกรม Gx-Developer

และเมื่อเราเข้าใช้โปรแกรมแล้วจะพบหน้าต่างการใช้งานโปรแกรมดังรูป

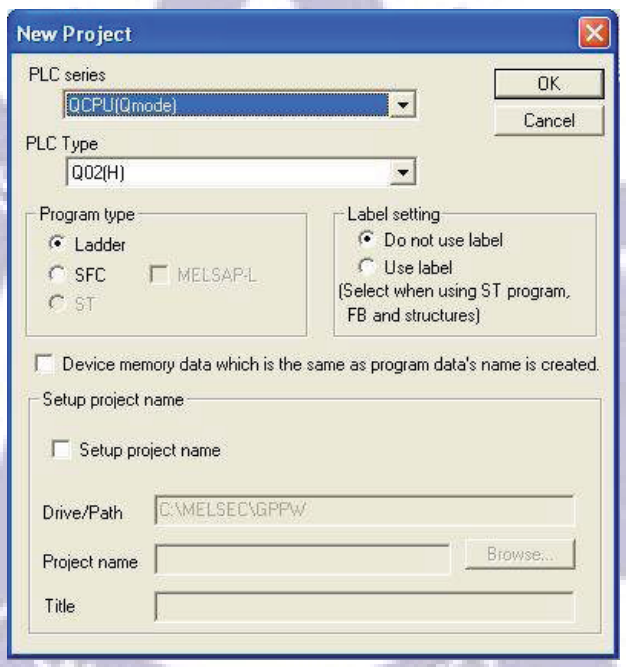
รูปที่ ข.2 โปรแกรม Gx-Developer พร้อมใช้งาน

2. การสร้างโปรแกรมใหม่

เลือก Project => New Project เลือก Series และ Type ของ PLC ที่ใช้ในการติดต่อ จากนั้นเลือก OK

หรือ คลิกที่ toolbar 

ต่อมาจะเป็นรูปการเลือกรุ่น ของ PLC



รูปที่ ข.3 การสร้างโปรเจคใหม่

ซึ่งความหมายของแต่ละฟังก์ชัน มีดังนี้

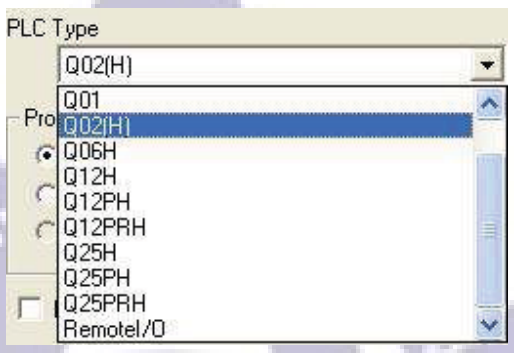
1)PLC Series ไว้สำหรับเลือกรุ่นของ PLC มิตซูบิชิ ที่จะใช้เขียนโปรแกรมซึ่งมีดังนี้

-QCPU (Q-mode) -A Series

-QnA (Series) -MOTION(SCPU)

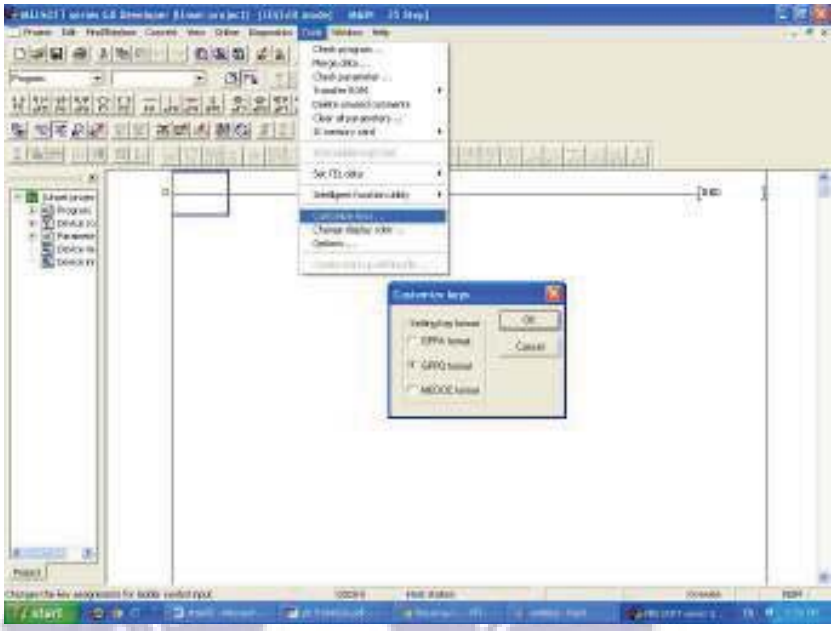
- QCPU (A-mode) -FX Series

2) PLC Type ไว้สำหรับเลือกชนิดของ PLC แต่ละรุ่น



รูปที่ ข.4 การเลือกรุ่นของ PLC

เมื่อเราทำการเลือก Series และ Type ได้แล้ว จากนั้นจะมาเริ่มต้นการเขียนโปรแกรมและการใช้คำสั่งพื้นฐาน ในการใช้งานซอฟต์แวร์ Gx Developer นั้นผู้ใช้สามารถเลือกที่จะเลือก Function key ได้เอง โดยเลือกได้ 3 รูปแบบ คือ GPPA, GPPQ และแบบ MEDOC ตัวอย่างที่จะได้อธิบายต่อไปนี้ จะใช้ Function key แบบ GPPQ ซึ่งเป็นฟังก์ชั่นมาตรฐาน และเป็นค่า Default หลังจากทำการติดตั้งโปรแกรมเป็นที่เรียบร้อยแล้ว สำหรับผู้ที่ต้องการเปลี่ยนรูปแบบ กรใช้ฟังก์ชั่นการคีย์ โปรแกรมก็สามารถทำได้โดยเข้าไปที่เมนู Tool => Customize keys => เลือกฟังก์ชั่นคีย์ตามต้องการ



รูปที่ ข.5 การเลือกรูปแบบของ Function key

สัญลักษณ์ที่ใช้ในการเขียนโปรแกรม

รูปที่ ข.6 สัญลักษณ์ที่ใช้ในการเขียนโปรแกรม

3. วิธีการเขียนคำสั่ง

Example (LD and OUT instructions)

Ladder Diagram

Instruction List

```
0   LD    X000
1   OUT   Y020
```

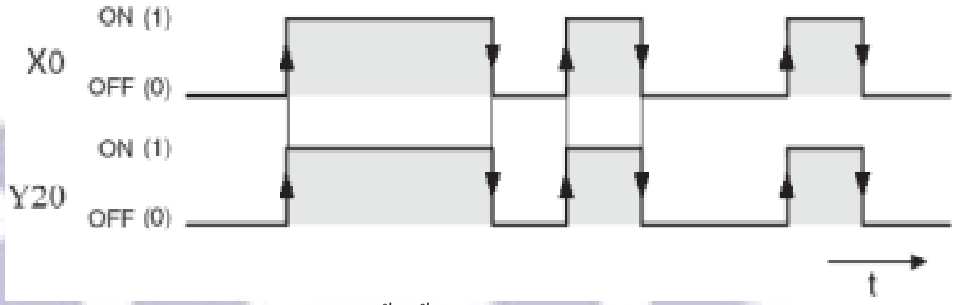
วิธีการเขียน

1.กด F5 => ใส่เบอร์อุปกรณ์อินพุต X000 => กดปุ่ม OK

2.กด F7 => ใส่เบอร์อุปกรณ์เอาท์พุต Y20 => กดปุ่ม OK

3.ทำการ Convert Program หรือกดปุ่ม F4

Timing Diagram



จากรูปจะเห็นว่า Y20 จะทำงานหรือไม่นั้นขึ้นอยู่กับ การ ON/OFF ของอุปกรณ์อินพุต X0 ยกตัวอย่างเช่น เมื่อ X0 มีค่าเป็น 1 หรือว่า ON นั้น เอาท์พุต Y20 ก็จะมีค่าเป็น 1 หรือว่า ON ด้วย และในทางกลับกัน เมื่อ X0 มีค่าเป็น 0 หรือว่า OFF เอาท์พุต Y20 จะมีค่าเป็น 0 หรือ OFF ด้วยเช่นกัน

Example (LDI and OUT instructions)

Ladder Diagram

Instruction List

0	LDI	X000
1	OUT	Y020

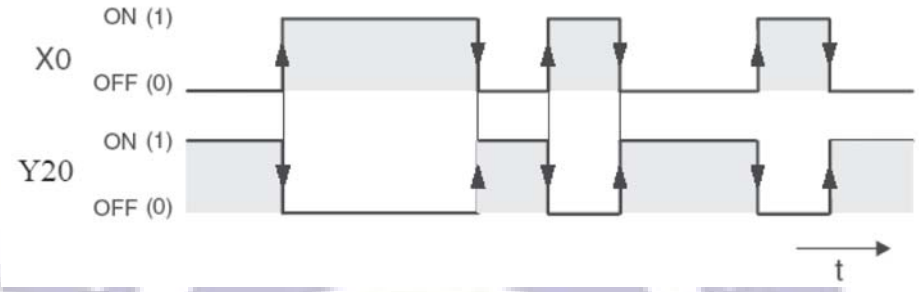
วิธีการเขียนโปรแกรม

1.กด F6 => ใส่เบอร์อุปกรณ์อินพุต X000 => กดปุ่ม OK

2.กด F7 => ใส่เบอร์อุปกรณ์อินพุต Y20 => กดปุ่ม OK

3.ทำการConvert Program หรือกดปุ่ม F4

Timing Diagram



จากรูป จะเห็นว่าเอาท์พุต Y20 จะทำงานตรงข้ามกับการทำงาน ON/OFF ของอุปกรณ์อินพุต X0 กล่าวคือ ขณะที่อุปกรณ์อินพุต X0 มีค่าเป็น 1 หรือว่า ON นั้น เอาท์พุต Y20 ก็จะมีค่าเป็น 0 หรือว่า OFF และในทางกลับกันเมื่อ X0 มีค่าเป็น 0 หรือ OFF นั้น เอาท์พุต Y20 จะมีค่าเป็น 1 หรือ ON ด้วยเช่นกัน กล่าวโดยสรุปก็คือ คำสั่ง LDI นั้นจะตรงกันข้ามกับ LD นั่นเอง

Example of an AND instruction

Ladder Diagram

Instruction List

```
0   LD    X000
1   AND   X001
2   OUT   Y020
```

วิธีการเขียนโปรแกรม

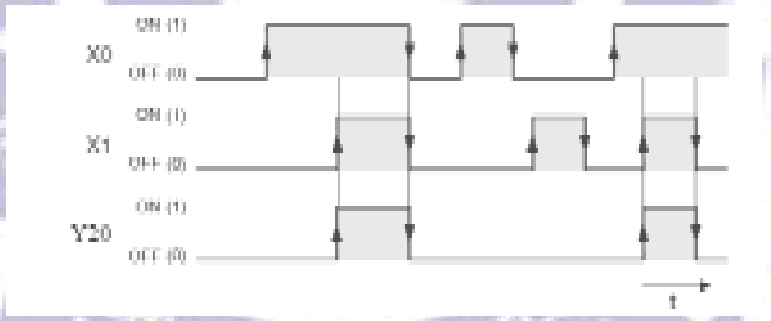
1.กด F5 => ใส่เบอร์อุปกรณ์อินพุต X000 => กดปุ่ม OK

2.กด F5 => ใส่เบอร์อุปกรณ์อินพุต X001 => กดปุ่ม OK

3.กด F7 => ใส่เบอร์อุปกรณ์เอาท์พุต Y20 => กดปุ่ม OK

4.ทำการ Convert Program หรือกดปุ่ม F4

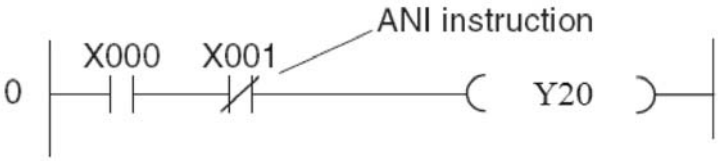
Timing Diagram



จากรูปจะเห็นว่า เอาท์พุต Y20 จะทำงานได้ก็ต่อเมื่ออินพุต X0 และ X1 ทำงาน หรือ ON พร้อมกัน 2 ตัว

Example of an ANI instruction

Ladder Diagram



Instruction List

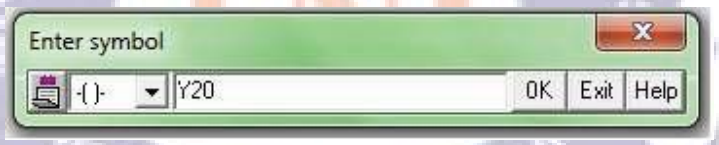
```
0   LD    X000
1   ANI   X001
2   OUT   Y020
```

วิธีการเขียนโปรแกรม

1.กด F5 => ใส่เบอร์อุปกรณ์อินพุต X000 => กดปุ่ม OK

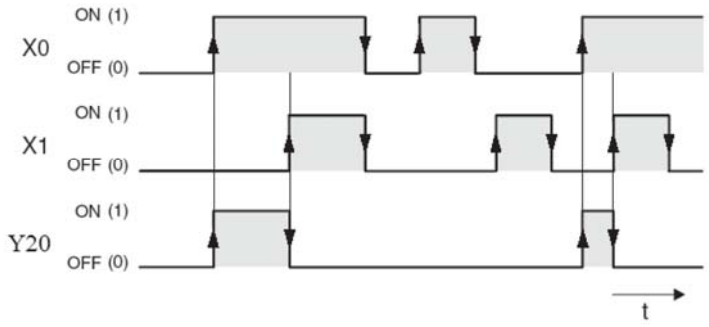
2.กด F6 => ใส่เบอร์อุปกรณ์อินพุต X001 => กดปุ่ม OK

3.กด F7 => ใส่เบอร์อุปกรณ์เอาท์พุต Y20 => กดปุ่ม OK

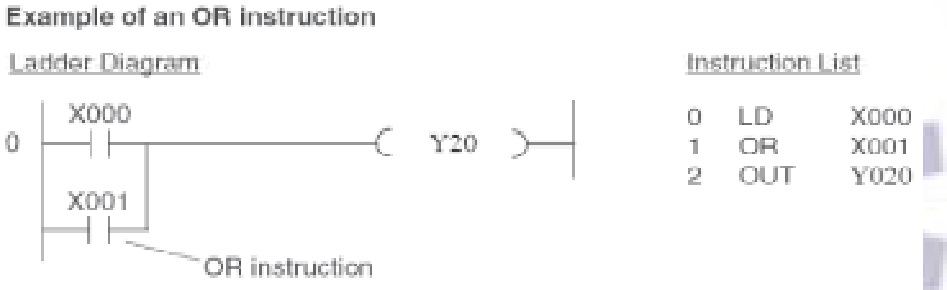


4.ทำการ Convert Program หรือกดปุ่ม F4

Timing Diagram



จากรูปจะเห็นว่า เอาท์พุต Y20 จะทำงานได้ก็ต่อเมื่ออินพุต X0 และ X1 ทำงาน หรือ ON ไม่พร้อมกัน



วิธีการเขียนโปรแกรม

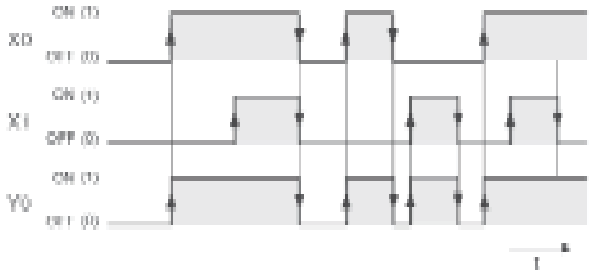
1.กด F5 => ใส่เบอร์อุปกรณ์อินพุต X000 => กดปุ่ม OK

2.กด F7 => ใส่เบอร์อุปกรณ์เอาท์พุต Y20 => กดปุ่ม OK

3.กด Shift + F5 => ใส่เบอร์อุปกรณ์อินพุต X001 => กดปุ่ม OK

4.ทำการ Convert Program หรือกดปุ่ม F4

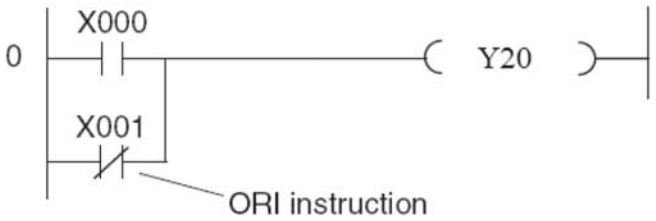
Timing Diagram



จากรูปจะเห็นว่า เอาท์พุต Y20 จะทำงานได้ก็ต่อเมื่ออินพุต X0 และ X1 ตัวใดตัวหนึ่งมีสถานะเป็น 1 หรือ ON นั่นเอง

Example of an ORI instruction

Ladder Diagram



Instruction List

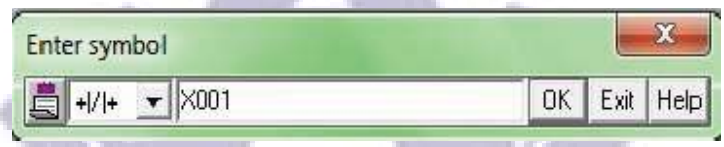
```
0   LD    X000
1   ORI   X001
2   OUT   Y020
```

วิธีการเขียนโปรแกรม

1.กด F5 => ใส่เบอร์อุปกรณ์อินพุต X000 => กดปุ่ม OK

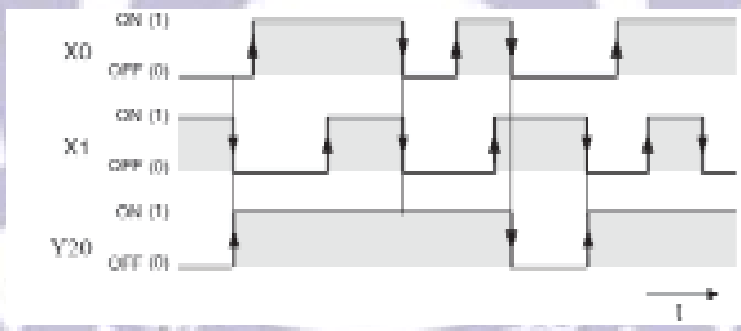
2.กด F7 => ใส่เบอร์อุปกรณ์เอาท์พุต Y20 => กดปุ่ม OK

3.กด Shift + F6 => ใส่เบอร์อุปกรณ์อินพุต X001 => กดปุ่ม OK



4.ทำการ Convert Program หรือกดปุ่ม F4

Timing Diagram

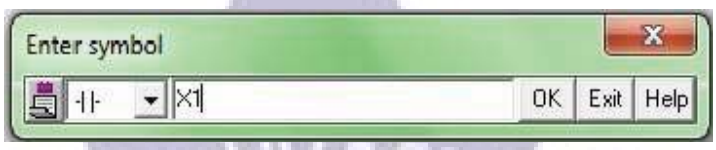


3.1 ตัวอย่างการเขียนโปรแกรมไฟกระพริบ

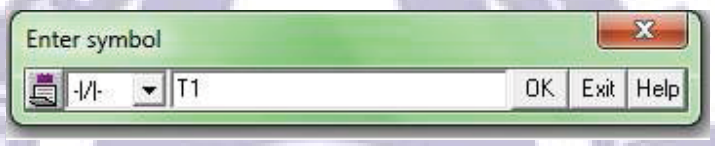
การเขียนโปรแกรม Ladder ตามตัวอย่างข้างล่างนี้ จะใช้ Function Key ของ Software GPPQ และการเขียนโปรแกรมLadder นี้จะต้องอยู่ในโหมด write (กด F2) เสมอ

วิธีการเขียน

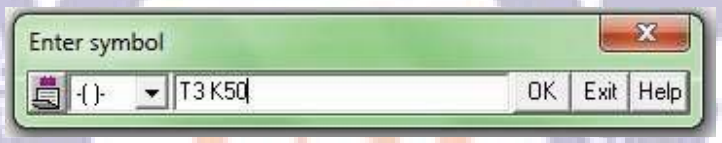
1.กด F5 => ใส่เบอร์อุปกรณ์อินพุต X001 => กดปุ่ม OK



2.กด F6 => ใส่เบอร์อุปกรณ์ Timer T1 => กดปุ่ม OK

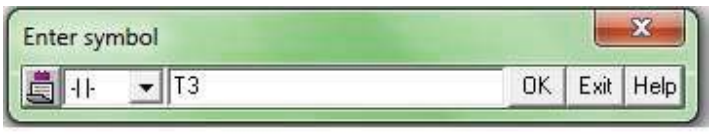


3.กด F7=> ใส่เบอร์อุปกรณ์ Timer T3 และใส่ค่าคงที่ K เท่ากับ 50 => กดปุ่ม OK



4.กด Shift + F5 ใส่เบอร์อุปกรณ์ Timer T3 => กดปุ่ม OK

5.กด F5 => ใส่เบอร์อุปกรณ์ Timer T3 => กดปุ่ม OK

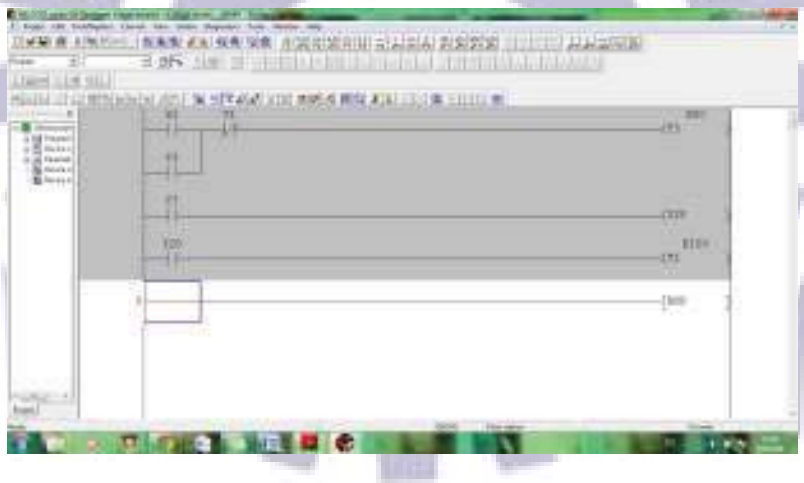


6.กด F7 => ใส่เบอร์อุปกรณ์เอาท์พุต Y20 => กดปุ่ม OK

7.กด F5=> ใส่เบอร์อุปกรณ์ Y20 => กดปุ่ม OK

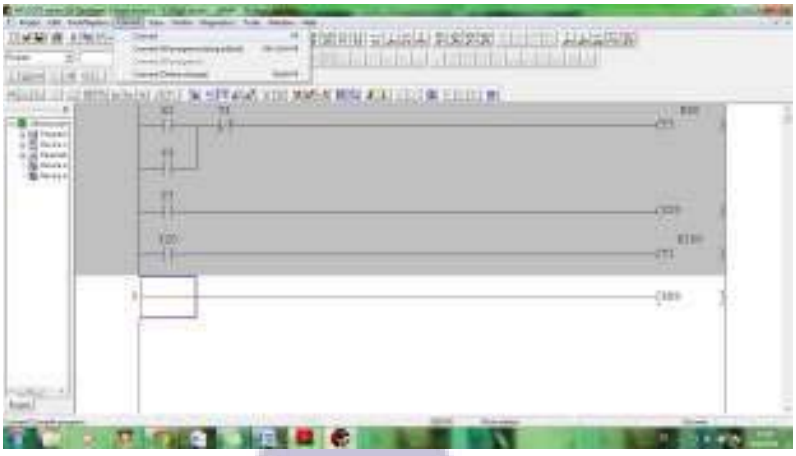
8.กด F7 => ใส่เบอร์อุปกรณ์ Timer T1 และใส่ค่าคงที่ K เท่ากับ 100 => กดปุ่ม OK

เมื่อเขียนเสร็จแล้วจะได้ Ladder Diagram ตามภาพด้านล่าง



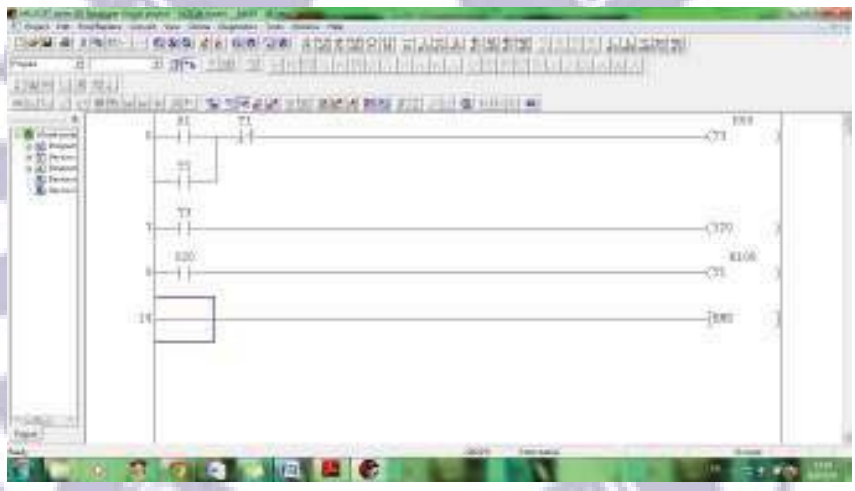
รูปที่ ข.7 โปรแกรมที่เขียนเสร็จแล้วแต่ยังไม่มีการ Convert

หลังจากเขียนโปรแกรมเสร็จเรียบร้อยแล้วจะต้องทำการ Convert Program โดยการเลือกที่ Convert => Convert หรือกด F4



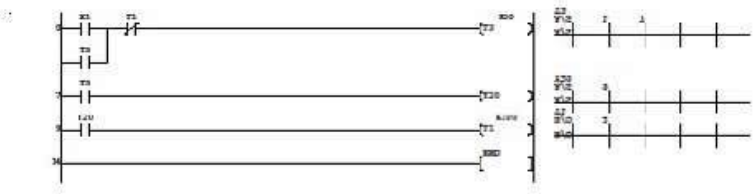
รูปที่ ข.8 ขั้นตอนการ Convert Program

จะได้โปรแกรมตามตัวอย่างด้านล่าง ซึ่งเป็นโปรแกรมที่เสร็จสมบูรณ์



รูปที่ ข.9 โปรแกรมแบบเสร็จสมบูรณ์ สามารถนำไปใช้ Load ลงเครื่อง PLC ได้

การทำงานของโปรแกรมนี้ คือ เมื่อเรากด Switch X1 นานเป็นเวลาครบ 5 วินาที จะทำให้หลอดไฟ Y20 ติดค้างนาน เป็นเวลา 10 วินาที เมื่อครบ 10 วินาที หลอดไฟ Y20 จะดับ จากนั้น การทำงานจะเริ่มทำงานใหม่ตั้งแต่แรก

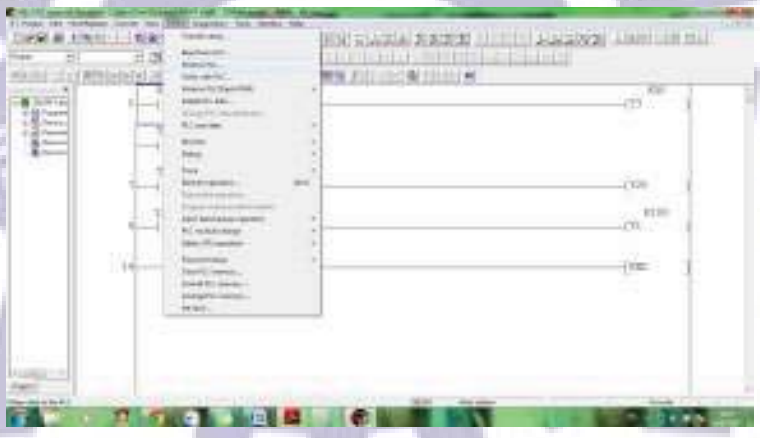


รูปที่ ข.10 Ladder Diagram

ข้อสำคัญสำหรับการเขียนโปรแกรม คือก่อนที่จะเริ่มเขียนโปรแกรมแลดเดอร์ หมวดการทำงานจะต้องอยู่ใน Write Mode เสมอ

สำหรับการนำโปรแกรมต่างๆที่เราเขียนนั้นไปใช้งาน สามารถทำได้โดย การ Load Program ลงใน PLC โดยผ่านทางสาย USB หรือสาย RS-232 ส่วนวิธีการ Load นั้น ทำได้โดย

1.ไปที่ Online => Write to PLC => OK



รูปที่ ข.11 การ Write Program ลง PLC

2.เลือก Param + Prog

3.จากนั้นกด Execute

และเมื่อเรา Load Program ลงใน PLC แล้ว ตัวเครื่องนั้น จะสามารถทำงานได้ตามคำสั่งที่เรา

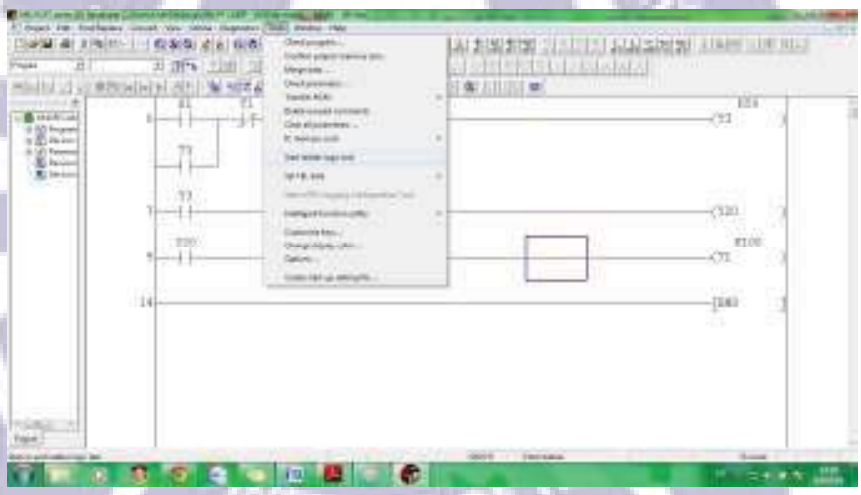
เขียนไว้ทันที แต่ในกรณีที่เราไม่มีเครื่อง PLC เราก็สามารถตรวจดูการทำงานของโปรแกรมได้โดยใช้การแสดงภาพจำลองการทำงานของโปรแกรมแลดเดอร์(Gx Simulator)

4. การแสดงภาพจำลองการทำงานของโปรแกรมแลดเดอร์ (Gx Simulator)

Software Gx Developer มีฟังก์ชั่นที่สามารถจำลองการทำงานของโปรแกรมแลดเดอร์ที่เสมือนจริงได้ โดยฟังก์ชั่นการทำงานแบบนี้เราเรียกว่า Simulator นอกจากจะดูการทำงานของโปรแกรมแลดเดอร์แล้ว ในฟังก์ชั่น Simulator นี้ยังสามารถที่จะทำการ Monitor ดูการทำงานของอุปกรณ์ชนิดต่างๆ ที่เขียนในโปรแกรมแลดเดอร์โดยที่ไม่จำเป็นต้องต่อกับ PLC เช่นกัน

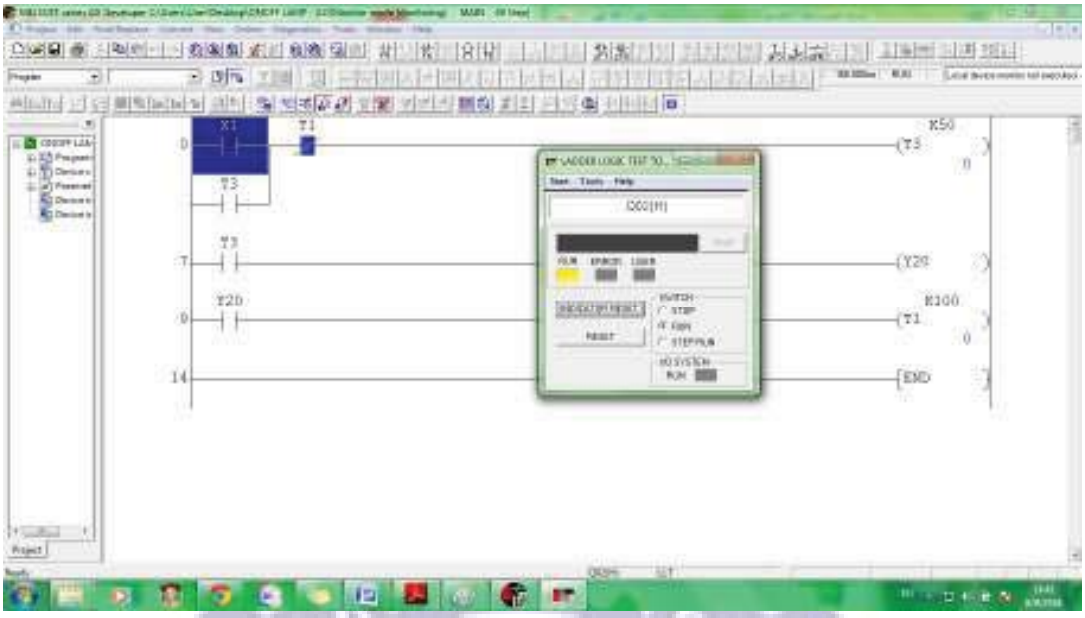
ขั้นตอนการ Simulator

เมื่อเราทำการ Convert Program (F4) แล้ว เลือกที่เมนู Tools => Start ladder logic test (ต้องทำการ install โปรแกรม Simulator นี้ด้วย เพราะถ้าไม่มีตัวโปรแกรมนี้จะไม่สามารถทำการ Simulator ได้)



รูปที่ ข.12 รูปการ Simulator

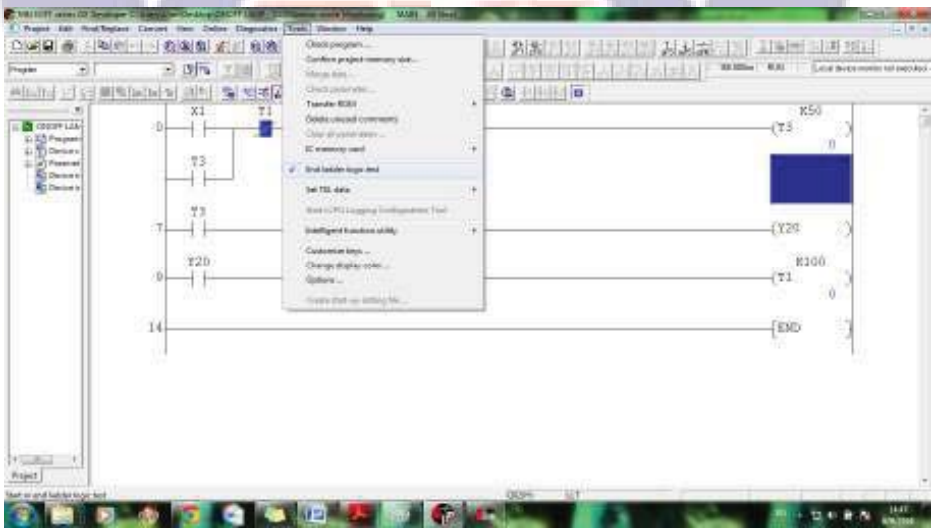
โปรแกรมจะทำการจำลองการทำงานของคอมพิวเตอร์ให้เสมือนให้เสมือนเป็น PLC และการเขียนโปรแกรมแลดเดอร์เข้าไปในหน่วยความจำของคอมพิวเตอร์ เพื่อทำการประมวลผลการทำงานของโปรแกรมแลดเดอร์ใน CPU ภายในตัวคอมพิวเตอร์เอง



รูปที่ ข.13 หน้าจออยู่ใน Mode Monitor

โปรแกรมแลดเดอร์จะอยู่ในโหมดการ Monitor ทันทีหลังจากเลือกฟังก์ชั่นการ Simulator จากนั้นเราก็สามารถดูการทำงานของโปรแกรมได้ทันที

การยกเลิกการใช้งานฟังก์ชั่น Simulator เลือกเมนู Tools => End ladder logic test



รูปที่ ข.14 แสดงการยกเลิกการใช้งานฟังก์ชั่น Simulator

ภาคผนวก ค

ตัวอย่าง Code โปรแกรมที่นำไปใช้งาน

1.โปรแกรม Assembly ที่นำมาปฏิบัติงาน

ส่วน Main Micro Program

```
0FDC:0000 9C            PUSHF
0FDC:0001 50            PUSH    AX
0FDC:0002 51            PUSH    CX
0FDC:0003 52            PUSH    DX
0FDC:0004 53            PUSH    BX
0FDC:0005 55            PUSH    BP
0FDC:0006 56            PUSH    SI
0FDC:0007 57            PUSH    DI
0FDC:0008 06            PUSH    ES
0FDC:0009 1E            PUSH    DS
0FDC:000A B80000        MOV     AX,0000
0FDC:000D 8ED8          MOV     DS,AX
0FDC:000F 8B3EFE9F      MOV     DI,[9FFE]
0FDC:0013 BB0088        MOV     BX,8800
0FDC:0016 D1E7          SHL     DI,1
0FDC:0018 03FB          ADD     DI,BX
0FDC:001A 8B1EFA9F      MOV     BX,[9FFA]
0FDC:001E D1E3          SHL     BX,1
0FDC:0020 BE3100        MOV     SI,0031
0FDC:0023 2E            CS:
0FDC:0024 FF10          CALL    [BX+SI]
0FDC:0026 1F            POP     DS
0FDC:0027 07            POP     ES
0FDC:0028 5F            POP     DI
0FDC:0029 5E            POP     SI
0FDC:002A 5D            POP     BP
0FDC:002B 5B            POP     BX
0FDC:002C 5A            POP     DX
0FDC:002D 59            POP     CX
0FDC:002E 58            POP     AX
0FDC:002F 9D            POPF
0FDC:0030 CB            RETF
```

0075

```
0FDC:0075 50            PUSH    AX
0FDC:0076 53            PUSH    BX
0FDC:0077 51            PUSH    CX
0FDC:0078 52            PUSH    DX
0FDC:0079 57            PUSH    DI
0FDC:007A 56            PUSH    SI
0FDC:007B 1E            PUSH    DS
0FDC:007C 06            PUSH    ES
0FDC:007D E8DDFF        CALL    005D
0FDC:0080 8B1D          MOV     BX,[DI]
0FDC:0082 D1EB          SHR     BX,1
0FDC:0084 D1EB          SHR     BX,1
0FDC:0086 8DB70084      LEA     SI,[BX+8400]
0FDC:008A 804D3604      OR      BYTE PTR [DI+36],04
0FDC:008E 26            ES:
0FDC:008F F60401        TEST    BYTE PTR [SI],01
0FDC:0092 7504          JNZ     0098
0FDC:0094 806536FB      AND     BYTE PTR [DI+36],FB
0FDC:0098 26            ES:
0FDC:0099 8024EF        AND     BYTE PTR [SI],EF
0FDC:009C 8B4502        MOV     AX,[DI+02]
0FDC:009F 3B4504        CMP     AX,[DI+04]
0FDC:00A2 7204          JB      00A8
0FDC:00A4 26            ES:
0FDC:00A5 800C10        OR      BYTE PTR [SI],10
0FDC:00A8 26            ES:
0FDC:00A9 F60480        TEST    BYTE PTR [SI],80
0FDC:00AC 7504          JNZ     00B2
0FDC:00AE 26            ES:
0FDC:00AF 803410        XOR     BYTE PTR [SI],10
0FDC:00B2 26            ES:
0FDC:00B3 F60480        TEST    BYTE PTR [SI],80
0FDC:00B6 7407          JZ      00BF
```

0075(ต่อ)

```
0FDC:00B8 3B4514        CMP     AX,[DI+14]
0FDC:00BB 7D0B          JGE     00C8
0FDC:00BD EB05          JMP     00C4
0FDC:00BF 3B4514        CMP     AX,[DI+14]
0FDC:00C2 7E04          JLE     00C8
0FDC:00C4 26            ES:
0FDC:00C5 8024DF        AND     BYTE PTR [SI],DF
0FDC:00C8 26            ES:
0FDC:00C9 F60401        TEST    BYTE PTR [SI],01
0FDC:00CC 7503          JNZ     00D1
0FDC:00CE E98A00        JMP     015B
0FDC:00D1 26            ES:
0FDC:00D2 F60404        TEST    BYTE PTR [SI],04
0FDC:00D5 7503          JNZ     00DA
0FDC:00D7 E98100        JMP     015B
0FDC:00DA 8B4D08        MOV     CX,[DI+08]
0FDC:00DD 3B4D0A        CMP     CX,[DI+0A]
0FDC:00E0 7279          JB      015B
0FDC:00E2 8B4502        MOV     AX,[DI+02]
0FDC:00E5 2B452E        SUB     AX,[DI+2E]
0FDC:00E8 E8A406        CALL    078F
0FDC:00EB 8B5D22        MOV     BX,[DI+22]
0FDC:00EE F7EB          IMUL    BX
0FDC:00F0 83FAFF        CMP     DX,-01
0FDC:00F3 7C66          JL      015B
0FDC:00F5 7F07          JG      00FE
0FDC:00F7 3D18FC        CMP     AX,FC18
0FDC:00FA 770C          JA      0108
0FDC:00FC EB5D          JMP     015B
0FDC:00FE 83FA00        CMP     DX,+00
0FDC:0101 7F58          JG      015B
0FDC:0103 3DE803        CMP     AX,03E8
0FDC:0106 7753          JA      015B
0FDC:0108 03C1          ADD     AX,CX
0FDC:010A 7E4F          JLE     015B
0FDC:010C 8BD0          MOV     DX,AX
0FDC:010E 33C0          XOR     AX,AX
0FDC:0110 8B5D06        MOV     BX,[DI+06]
0FDC:0113 83FB05        CMP     BX,+05
0FDC:0116 7E43          JLE     015B
0FDC:0118 50            PUSH    AX
0FDC:0119 52            PUSH    DX
```

0075(ต่อ)

```
0FDC:011A 8BC3          MOV     AX,BX
0FDC:011C 8B5D2A        MOV     BX,[DI+2A]
0FDC:011F F7EB          IMUL    BX
0FDC:0121 BB0040        MOV     BX,4000
0FDC:0124 F7FB          IDIV    BX
0FDC:0126 8BD8          MOV     BX,AX
0FDC:0128 5A            POP     DX
0FDC:0129 58            POP     AX
0FDC:012A E87B06        CALL    07A8
0FDC:012D 034510        ADD     AX,[DI+10]
0FDC:0130 135512        ADC     DX,[DI+12]
0FDC:0133 2B450C        SUB     AX,[DI+0C]
0FDC:0136 1B550E        SBB     DX,[DI+0E]
0FDC:0139 3B5526        CMP     DX,[DI+26]
0FDC:013C 7E03          JLE     0141
0FDC:013E 8B5526        MOV     DX,[DI+26]
0FDC:0141 3B5524        CMP     DX,[DI+24]
0FDC:0144 7D03          JGE     0149
0FDC:0146 8B5524        MOV     DX,[DI+24]
0FDC:0149 895512        MOV     [DI+12],DX
0FDC:014C 894510        MOV     [DI+10],AX
0FDC:014F BB5000        MOV     BX,0050
0FDC:0152 E85306        CALL    07A8
0FDC:0155 89450C        MOV     [DI+0C],AX
0FDC:0158 89550E        MOV     [DI+0E],DX
0FDC:015B 8B5D02        MOV     BX,[DI+02]
0FDC:015E 895D2E        MOV     [DI+2E],BX
0FDC:0161 8B4502        MOV     AX,[DI+02]
0FDC:0164 2B4504        SUB     AX,[DI+04]
0FDC:0167 50            PUSH    AX
0FDC:0168 7D02          JGE     016C
0FDC:016A F7D8          NEG     AX
0FDC:016C 05F401        ADD     AX,01F4
0FDC:016F F7E8          IMUL    AX
0FDC:0171 BB6504        MOV     BX,0465
0FDC:0174 F7FB          IDIV    BX
0FDC:0176 2DDE00        SUB     AX,00DE
0FDC:0179 5B            POP     BX
0FDC:017A 83FB00        CMP     BX,+00
0FDC:017D 7D02          JGE     0181
```

0075(ต่อ)

```
0FDC:017F F7D8          NEG     AX
0FDC:0181 034504        ADD     AX,[DI+04]
0FDC:0184 894530        MOV     [DI+30],AX
0FDC:0187 B80001        MOV     AX,0100
0FDC:018A 894538        MOV     [DI+38],AX
0FDC:018D 26            ES:
0FDC:018E F60408        TEST    BYTE PTR [SI],08
0FDC:0191 7419          JZ      01AC
0FDC:0193 8B4512        MOV     AX,[DI+12]
0FDC:0196 F7E8          IMUL    AX
0FDC:0198 8B5D26        MOV     BX,[DI+26]
0FDC:019B F7FB          IDIV    BX
0FDC:019D 8B5D1A        MOV     BX,[DI+1A]
0FDC:01A0 F7EB          IMUL    BX
0FDC:01A2 8B5D26        MOV     BX,[DI+26]
0FDC:01A5 F7FB          IDIV    BX
0FDC:01A7 89453E        MOV     [DI+3E],AX
0FDC:01AA EB06          JMP     01B2
0FDC:01AC 8B451A        MOV     AX,[DI+1A]
0FDC:01AF 89453E        MOV     [DI+3E],AX
0FDC:01B2 8B4508        MOV     AX,[DI+08]
0FDC:01B5 894534        MOV     [DI+34],AX
0FDC:01B8 8B4504        MOV     AX,[DI+04]
0FDC:01BB 894532        MOV     [DI+32],AX
0FDC:01BE 8B4516        MOV     AX,[DI+16]
0FDC:01C1 89453A        MOV     [DI+3A],AX
0FDC:01C4 8B4518        MOV     AX,[DI+18]
0FDC:01C7 89453C        MOV     [DI+3C],AX
0FDC:01CA 8B451C        MOV     AX,[DI+1C]
0FDC:01CD 894540        MOV     [DI+40],AX
0FDC:01D0 EB00          JMP     01D2
0FDC:01D2 57            PUSH    DI
0FDC:01D3 83C730        ADD     DI,+30
0FDC:01D6 E89603        CALL    056F
0FDC:01D9 5F            POP     DI
0FDC:01DA B80000        MOV     AX,0000
0FDC:01DD 26            ES:
0FDC:01DE F60401        TEST    BYTE PTR [SI],01
```

```
0FDC:01E1 740F          JZ      01F2
0FDC:01E3 8B4508        MOV     AX,[DI+08]
0FDC:01E6 8B5D2C        MOV     BX,[DI+2C]
0FDC:01E9 F7EB          IMUL    BX
0FDC:01EB 8B5D12        MOV     BX,[DI+12]
0FDC:01EE F7FB          IDIV    BX
0FDC:01F0 8BD8          MOV     BX,AX
0FDC:01F2 03454E        ADD     AX,[DI+4E]
0FDC:01F5 89451E        MOV     [DI+1E],AX
0FDC:01F8 26            ES:
0FDC:01F9 F60402        TEST    BYTE PTR [SI],02
0FDC:01FC 7503          JNZ     0201
0FDC:01FE 894506        MOV     [DI+06],AX
```

0075(ต่อ)

```
0FDC:0201 07            POP     ES
0FDC:0202 1F            POP     DS
0FDC:0203 5E            POP     SI
0FDC:0204 5F            POP     DI
0FDC:0205 5A            POP     DX
0FDC:0206 59            POP     CX
0FDC:0207 5B            POP     BX
0FDC:0208 58            POP     AX
0FDC:0209 C3            RET
```

005D

```
0FDC:005D B80000        MOV     AX,0000
0FDC:0060 8ED8          MOV     DS,AX
0FDC:0062 B80000        MOV     AX,0000
0FDC:0065 8EC0          MOV     ES,AX
0FDC:0067 BF0090        MOV     DI,9000
0FDC:006A 26            ES:
0FDC:006B A1FE9F        MOV     AX,[9FFE]
0FDC:006E D1E0          SHL     AX,1
0FDC:0070 03F8          ADD     DI,AX
0FDC:0072 C3            RET
```

078F

```
0FDC:078F 83F800        CMP     AX,+00
0FDC:0792 7D0A          JGE     079E
0FDC:0794 83C002        ADD     AX,+02
0FDC:0797 7E0D          JLE     07A6
0FDC:0799 B80000        MOV     AX,0000
0FDC:079C EB08          JMP     07A6
0FDC:079E 83E802        SUB     AX,+02
0FDC:07A1 7D03          JGE     07A6
0FDC:07A3 B80000        MOV     AX,0000
0FDC:07A6 C3            RET
```

07A8

```
0FDC:07A8 52            PUSH    DX
0FDC:07A9 50            PUSH    AX
0FDC:07AA 53            PUSH    BX
0FDC:07AB 51            PUSH    CX
0FDC:07AC 55            PUSH    BP
0FDC:07AD 8BEC          MOV     BP,SP
0FDC:07AF 33DB          XOR     BX,BX
0FDC:07B1 B120          MOV     CL,20
0FDC:07B3 C746060000    MOV     WORD PTR [BP+06],0000
0FDC:07B8 D16606        SHL     WORD PTR [BP+06],1
0FDC:07BB D15608        RCL     WORD PTR [BP+08],1
0FDC:07BE D1E0          SHL     AX,1
0FDC:07C0 D1D2          RCL     DX,1
0FDC:07C2 D1D3          RCL     BX,1
0FDC:07C4 7205          JB      07CB
0FDC:07C6 3B5E04        CMP     BX,[BP+04]
0FDC:07C9 7206          JB      07D1
0FDC:07CB 2B5E04        SUB     BX,[BP+04]
0FDC:07CE FF4606        INC     WORD PTR [BP+06]
0FDC:07D1 FEC9          DEC     CL
0FDC:07D3 75E3          JNZ     07B8
0FDC:07D5 895E04        MOV     [BP+04],BX
0FDC:07D8 5D            POP     BP
0FDC:07D9 59            POP     CX
0FDC:07DA 5B            POP     BX
0FDC:07DB 58            POP     AX
0FDC:07DC 5A            POP     DX
0FDC:07DD C3            RET
```

056F

```
0FDC:056F 50            PUSH    AX
0FDC:0570 52            PUSH    DX
0FDC:0571 53            PUSH    BX
0FDC:0572 56            PUSH    SI
0FDC:0573 57            PUSH    DI
0FDC:0574 F745060400    TEST    WORD PTR [DI+06],0004
0FDC:0579 7557          JNZ     05D2
0FDC:057B 8B05          MOV     AX,[DI]
0FDC:057D 89452C        MOV     [DI+2C],AX
0FDC:0580 8B5D12        MOV     BX,[DI+12]
0FDC:0583 F7E3          MUL     BX
0FDC:0585 894528        MOV     [DI+28],AX
0FDC:0588 89552A        MOV     [DI+2A],DX
0FDC:058B 8B4502        MOV     AX,[DI+02]
0FDC:058E 894536        MOV     [DI+36],AX
```

056F(ต่อ)

```
0FDC:0591 8B5D14        MOV     BX,[DI+14]
0FDC:0594 F7E3          MUL     BX
0FDC:0596 894532        MOV     [DI+32],AX
0FDC:0599 895534        MOV     [DI+34],DX
0FDC:059C 8B4504        MOV     AX,[DI+04]
0FDC:059F 894540        MOV     [DI+40],AX
0FDC:05A2 8B5D16        MOV     BX,[DI+16]
0FDC:05A5 F7E3          MUL     BX
0FDC:05A7 89453C        MOV     [DI+3C],AX
0FDC:05AA 89553E        MOV     [DI+3E],DX
0FDC:05AD 33C0          XOR     AX,AX
0FDC:05AF 894548        MOV     [DI+48],AX
0FDC:05B2 89454A        MOV     [DI+4A],AX
0FDC:05B5 89454C        MOV     [DI+4C],AX
0FDC:05B8 89454E        MOV     [DI+4E],AX
0FDC:05BB 8B4536        MOV     AX,[DI+36]
0FDC:05BE 2B452C        SUB     AX,[DI+2C]
0FDC:05C1 894542        MOV     [DI+42],AX
0FDC:05C4 894550        MOV     [DI+50],AX
0FDC:05C7 8B4540        MOV     AX,[DI+40]
0FDC:05CA 89455A        MOV     [DI+5A],AX
0FDC:05CD 33DB          XOR     BX,BX
0FDC:05CF E97901        JMP     074B
```

```
0FDC:05D2 8B05          MOV     AX,[DI]
0FDC:05D4 894524        MOV     [DI+24],AX
0FDC:05D7 8B4512        MOV     AX,[DI+12]
0FDC:05DA 894526        MOV     [DI+26],AX
0FDC:05DD 83C724        ADD     DI,+24
0FDC:05E0 E87101        CALL    0754
0FDC:05E3 83EF24        SUB     DI,+24
0FDC:05E6 8B4502        MOV     AX,[DI+02]
0FDC:05E9 89452E        MOV     [DI+2E],AX
0FDC:05EC 8B4514        MOV     AX,[DI+14]
0FDC:05EF 894530        MOV     [DI+30],AX
0FDC:05F2 83C72E        ADD     DI,+2E
0FDC:05F5 E85C01        CALL    0754
0FDC:05F8 83EF2E        SUB     DI,+2E
0FDC:05FB 8B4540        MOV     AX,[DI+40]
0FDC:05FE 89455A        MOV     [DI+5A],AX
0FDC:0601 8B4504        MOV     AX,[DI+04]
0FDC:0604 894538        MOV     [DI+38],AX
0FDC:0607 8B4516        MOV     AX,[DI+16]
0FDC:060A 89453A        MOV     [DI+3A],AX
0FDC:060D 83C738        ADD     DI,+38
0FDC:0610 E84101        CALL    0754
```

056F(ต่อ)

```
0FDC:0613 83EF38        SUB     DI,+38
0FDC:0616 8B4542        MOV     AX,[DI+42]
0FDC:0619 894550        MOV     [DI+50],AX
0FDC:061C 8B4536        MOV     AX,[DI+36]
0FDC:061F 2B452C        SUB     AX,[DI+2C]
0FDC:0622 894542        MOV     [DI+42],AX
0FDC:0625 8B5D0A        MOV     BX,[DI+0A]
0FDC:0628 F7EB          IMUL    BX
0FDC:062A 894544        MOV     [DI+44],AX
0FDC:062D 895546        MOV     [DI+46],DX
0FDC:0630 8B4548        MOV     AX,[DI+48]
0FDC:0633 89454C        MOV     [DI+4C],AX
0FDC:0636 8B454A        MOV     AX,[DI+4A]
0FDC:0639 89454E        MOV     [DI+4E],AX
0FDC:063C 8B4542        MOV     AX,[DI+42]
0FDC:063F 8B5D0C        MOV     BX,[DI+0C]
0FDC:0642 F7EB          IMUL    BX
0FDC:0644 034548        ADD     AX,[DI+48]
0FDC:0647 13554A        ADC     DX,[DI+4A]
0FDC:064A 7106          JNO     0652
0FDC:064C 8B454C        MOV     AX,[DI+4C]
0FDC:064F 8B554E        MOV     DX,[DI+4E]
0FDC:0652 894548        MOV     [DI+48],AX
```

```
0FDC:0655 89554A        MOV     [DI+4A],DX
0FDC:0658 D1FA          SAR     DX,1
0FDC:065A D1D8          RCR     AX,1
0FDC:065C D1FA          SAR     DX,1
0FDC:065E D1D8          RCR     AX,1
0FDC:0660 D1FA          SAR     DX,1
0FDC:0662 D1D8          RCR     AX,1
0FDC:0664 D1FA          SAR     DX,1
0FDC:0666 D1D8          RCR     AX,1
0FDC:0668 D1FA          SAR     DX,1
0FDC:066A D1D8          RCR     AX,1
0FDC:066C 894556        MOV     [DI+56],AX
0FDC:066F 895558        MOV     [DI+58],DX
0FDC:0672 8B4542        MOV     AX,[DI+42]
0FDC:0675 2B4550        SUB     AX,[DI+50]
0FDC:0678 E81401        CALL    078F
0FDC:067B 8B5D0E        MOV     BX,[DI+0E]
0FDC:067E F7EB          IMUL    BX
0FDC:0680 894552        MOV     [DI+52],AX
0FDC:0683 895554        MOV     [DI+54],DX
0FDC:0686 8B4540        MOV     AX,[DI+40]
0FDC:0689 2B455A        SUB     AX,[DI+5A]
0FDC:068C E80001        CALL    078F
0FDC:068F 8B5D10        MOV     BX,[DI+10]
0FDC:0692 F7EB          IMUL    BX
0FDC:0694 89455C        MOV     [DI+5C],AX
```

056F(ต่อ)

```
0FDC:0697 89555E        MOV     [DI+5E],DX
0FDC:069A 8B5D44        MOV     BX,[DI+44]
0FDC:069D 8B4546        MOV     AX,[DI+46]
0FDC:06A0 99            CWD
0FDC:06A1 8BC8          MOV     CX,AX
0FDC:06A3 8BF2          MOV     SI,DX
0FDC:06A5 8B4558        MOV     AX,[DI+58]
0FDC:06A8 99            CWD
0FDC:06A9 035D56        ADD     BX,[DI+56]
0FDC:06AC 13C8          ADC     CX,AX
0FDC:06AE 13F2          ADC     SI,DX
0FDC:06B0 8B4554        MOV     AX,[DI+54]
0FDC:06B3 99            CWD
0FDC:06B4 035D52        ADD     BX,[DI+52]
```

```
0FDC:06B7 13C8          ADC     CX,AX
0FDC:06B9 13F2          ADC     SI,DX
0FDC:06BB 8B455E        MOV     AX,[DI+5E]
0FDC:06BE 99            CWD
0FDC:06BF 035D5C        ADD     BX,[DI+5C]
0FDC:06C2 13C8          ADC     CX,AX
0FDC:06C4 13F2          ADC     SI,DX
0FDC:06C6 895D60        MOV     [DI+60],BX
0FDC:06C9 894D62        MOV     [DI+62],CX
0FDC:06CC 897564        MOV     [DI+64],SI
0FDC:06CF 8B5D08        MOV     BX,[DI+08]
0FDC:06D2 8B4560        MOV     AX,[DI+60]
0FDC:06D5 F7E3          MUL     BX
0FDC:06D7 894566        MOV     [DI+66],AX
0FDC:06DA 895568        MOV     [DI+68],DX
0FDC:06DD 8B4562        MOV     AX,[DI+62]
0FDC:06E0 F7E3          MUL     BX
0FDC:06E2 034568        ADD     AX,[DI+68]
0FDC:06E5 83D200        ADC     DX,+00
0FDC:06E8 894568        MOV     [DI+68],AX
0FDC:06EB 89556A        MOV     [DI+6A],DX
0FDC:06EE 8B4564        MOV     AX,[DI+64]
0FDC:06F1 F7EB          IMUL    BX
0FDC:06F3 03456A        ADD     AX,[DI+6A]
0FDC:06F6 83D200        ADC     DX,+00
0FDC:06F9 89456A        MOV     [DI+6A],AX
0FDC:06FC 89556C        MOV     [DI+6C],DX
0FDC:06FF BB0000        MOV     BX,0000
0FDC:0702 3B5D6C        CMP     BX,[DI+6C]
0FDC:0705 7F11          JG      0718
0FDC:0707 7C33          JL      073C
0FDC:0709 3B5D6A        CMP     BX,[DI+6A]
0FDC:070C 7C2E          JL      073C
0FDC:070E 8B5D68        MOV     BX,[DI+68]
0FDC:0711 83FB00        CMP     BX,+00
0FDC:0714 7C26          JL      073C
0FDC:0716 EB15          JMP     072D
0FDC:0718 BBFFFF        MOV     BX,FFFF
```

056F(ต่อ)

```
0FDC:071B 3B5D6C        CMP     BX,[DI+6C]
0FDC:071E 7F17          JG      0737
0FDC:0720 3B5D6A        CMP     BX,[DI+6A]
0FDC:0723 7F12          JG      0737
0FDC:0725 8B5D68        MOV     BX,[DI+68]
0FDC:0728 83FB00        CMP     BX,+00
0FDC:072B 7D0A          JGE     0737
0FDC:072D 3B5D18        CMP     BX,[DI+18]
0FDC:0730 7F0A          JG      073C
0FDC:0732 3B5D1A        CMP     BX,[DI+1A]
0FDC:0735 7D14          JGE     074B
0FDC:0737 8B5D1A        MOV     BX,[DI+1A]
0FDC:073A EB03          JMP     073F

0FDC:073C 8B5D18        MOV     BX,[DI+18]
0FDC:073F 8B454C        MOV     AX,[DI+4C]
0FDC:0742 894548        MOV     [DI+48],AX
0FDC:0745 8B454E        MOV     AX,[DI+4E]
0FDC:0748 89454A        MOV     [DI+4A],AX
0FDC:074B 895D1E        MOV     [DI+1E],BX
0FDC:074E 5F            POP     DI
0FDC:074F 5E            POP     SI
0FDC:0750 5B            POP     BX
0FDC:0751 5A            POP     DX
0FDC:0752 58            POP     AX
0FDC:0753 C3            RET
```

0754(1)

```
0FDC:0754 9C            PUSHF
0FDC:0755 52            PUSH    DX
0FDC:0756 50            PUSH    AX
0FDC:0757 53            PUSH    BX
0FDC:0758 BB0000        MOV     BX,0000
0FDC:075B 395D02        CMP     [DI+02],BX
0FDC:075E 7425          JZ      0785
0FDC:0760 8B4504        MOV     AX,[DI+04]
0FDC:0763 8B5506        MOV     DX,[DI+06]
0FDC:0766 8B5D02        MOV     BX,[DI+02]
0FDC:0769 F7FB          IDIV    BX
0FDC:076B 8B1D          MOV     BX,[DI]
0FDC:076D 2BD8          SUB     BX,AX
0FDC:076F 8BC3          MOV     AX,BX
0FDC:0771 99            CWD
0FDC:0772 034504        ADD     AX,[DI+04]
```

0754(1)ต่อ

```
0FDC:0775 135506        ADC     DX,[DI+06]
0FDC:0778 894504        MOV     [DI+04],AX
0FDC:077B 895506        MOV     [DI+06],DX
0FDC:077E 8B5D02        MOV     BX,[DI+02]
0FDC:0781 F7FB          IDIV    BX
0FDC:0783 EB02          JMP     0787
0FDC:0785 8B05          MOV     AX,[DI]
0FDC:0787 894508        MOV     [DI+08],AX
0FDC:078A 5B            POP     BX
0FDC:078B 58            POP     AX
0FDC:078C 5A            POP     DX
0FDC:078D 9D            POPF
0FDC:078E C3            RET
```

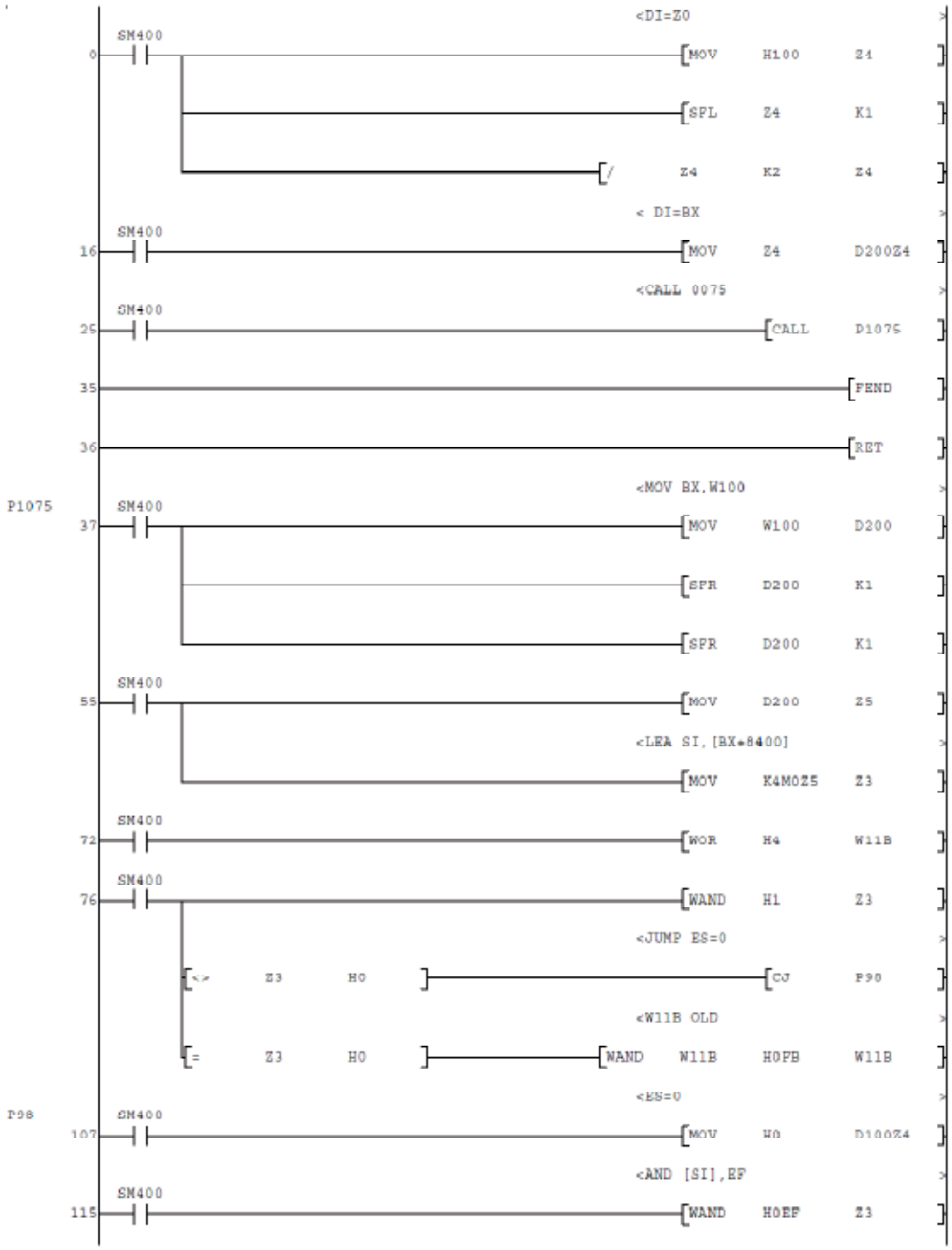
0754(2)

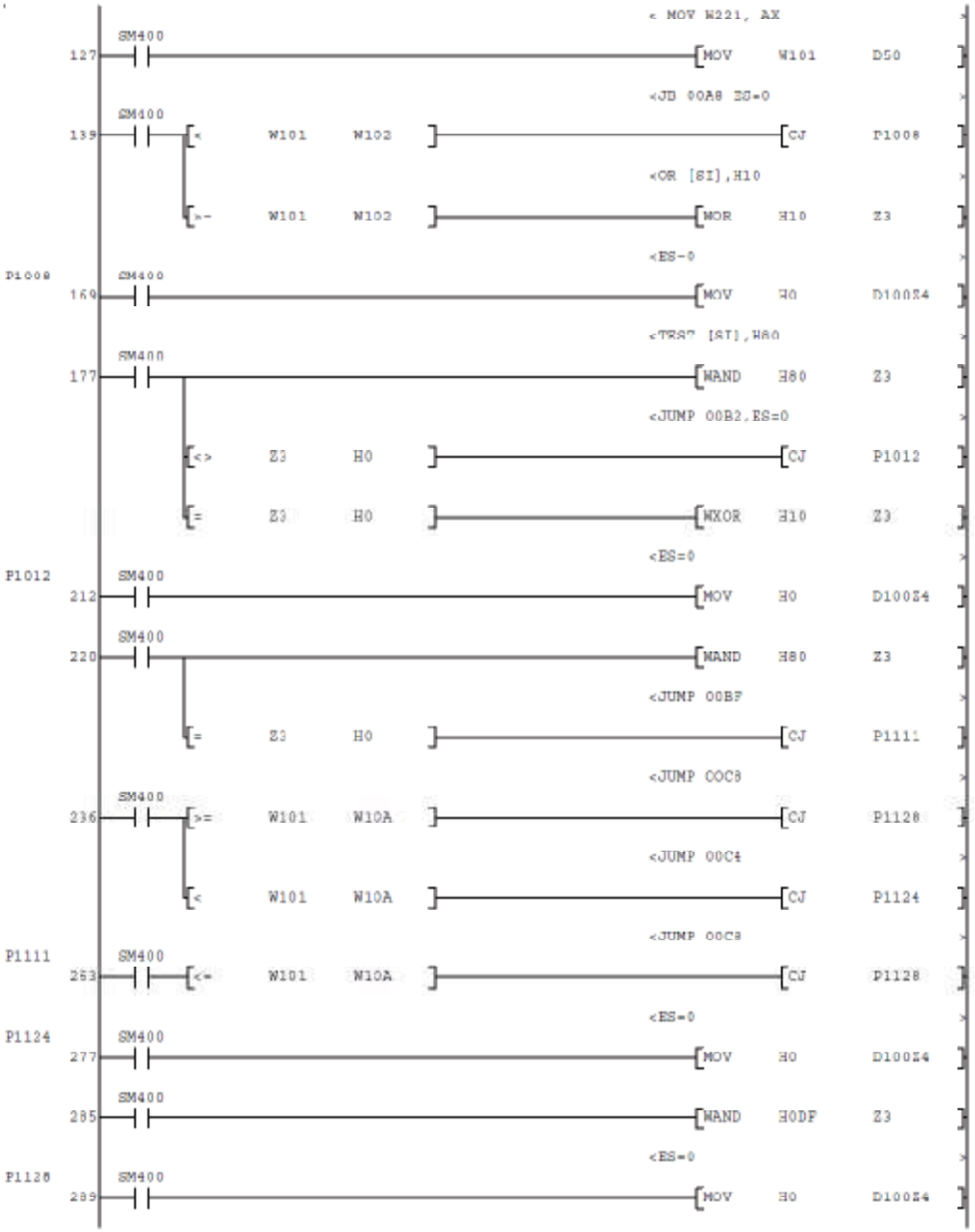
```
0FDC:0754 9C            PUSHF
0FDC:0755 52            PUSH    DX
0FDC:0756 50            PUSH    AX
0FDC:0757 53            PUSH    BX
0FDC:0758 BB0000        MOV     BX,0000
0FDC:075B 395D02        CMP     [DI+02],BX
0FDC:075E 7425          JZ      0785
0FDC:0760 8B4504        MOV     AX,[DI+04]
0FDC:0763 8B5506        MOV     DX,[DI+06]
0FDC:0766 8B5D02        MOV     BX,[DI+02]
0FDC:0769 F7FB          IDIV    BX
0FDC:076B 8B1D          MOV     BX,[DI]
0FDC:076D 2BD8          SUB     BX,AX
0FDC:076F 8BC3          MOV     AX,BX
0FDC:0771 99            CWD
0FDC:0772 034504        ADD     AX,[DI+04]
0FDC:0775 135506        ADC     DX,[DI+06]
0FDC:0778 894504        MOV     [DI+04],AX
0FDC:077B 895506        MOV     [DI+06],DX
0FDC:077E 8B5D02        MOV     BX,[DI+02]
0FDC:0781 F7FB          IDIV    BX
0FDC:0783 EB02          JMP     0787
0FDC:0785 8B05          MOV     AX,[DI]
0FDC:0787 894508        MOV     [DI+08],AX
0FDC:078A 5B            POP     BX
0FDC:078B 58            POP     AX
0FDC:078C 5A            POP     DX
0FDC:078D 9D            POPF
0FDC:078E C3            RET
```

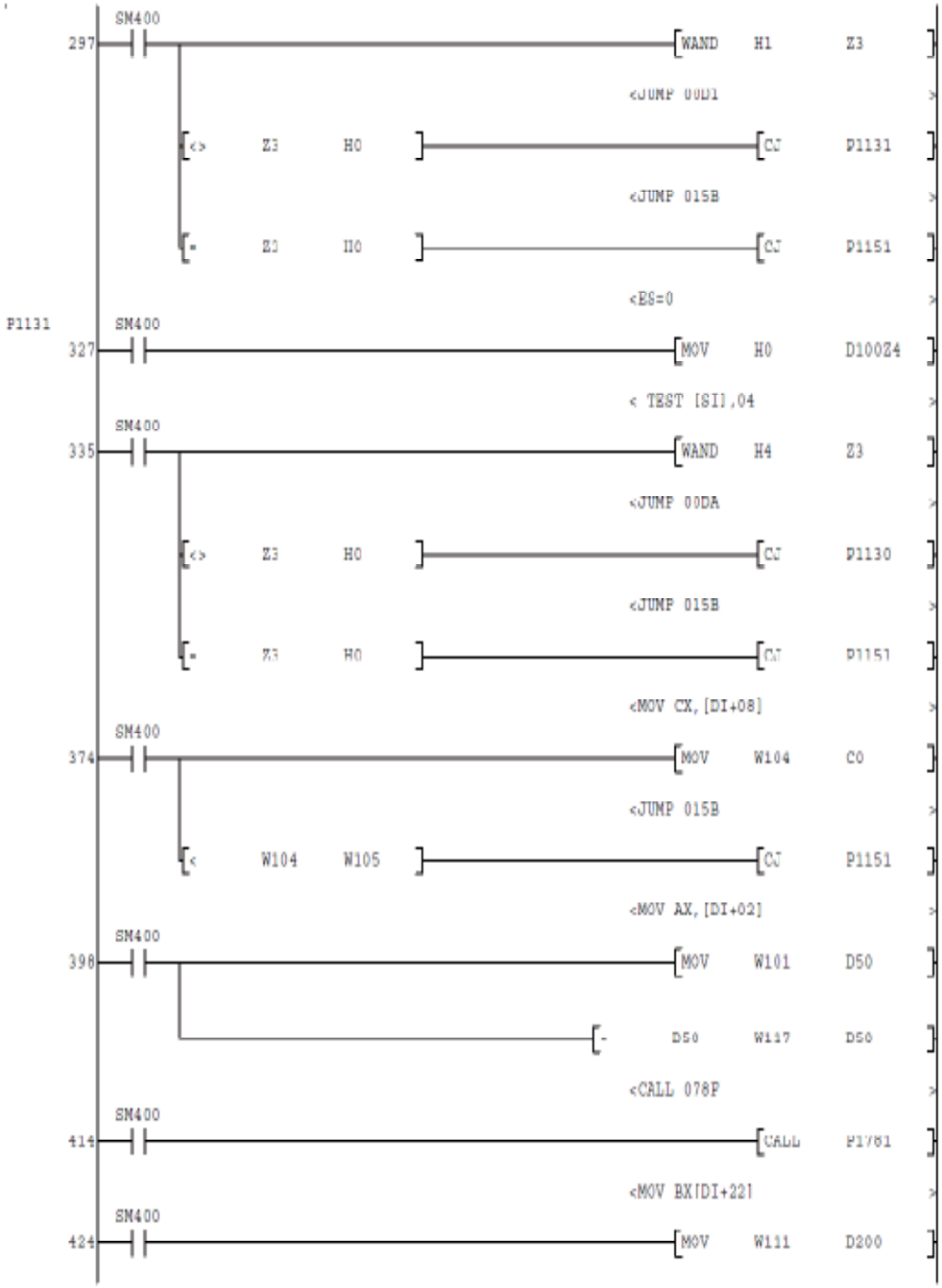
0754(3)

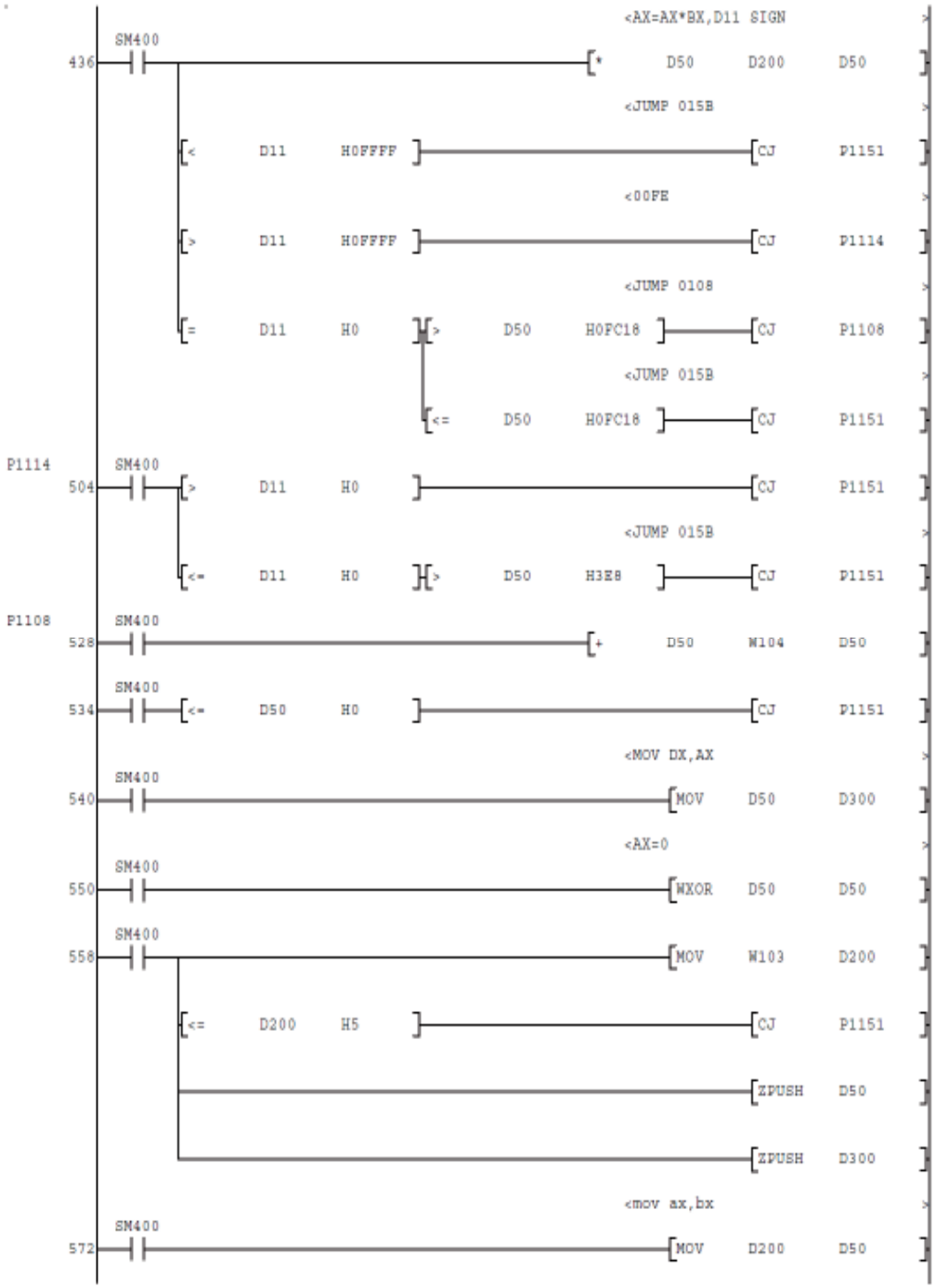
```
0FDC:0754 9C            PUSHF
0FDC:0755 52            PUSH    DX
0FDC:0756 50            PUSH    AX
0FDC:0757 53            PUSH    BX
0FDC:0758 BB0000        MOV     BX,0000
0FDC:075B 395D02        CMP     [DI+02],BX
0FDC:075E 7425          JZ      0785
0FDC:0760 8B4504        MOV     AX,[DI+04]
0FDC:0763 8B5506        MOV     DX,[DI+06]
0FDC:0766 8B5D02        MOV     BX,[DI+02]
0FDC:0769 F7FB          IDIV    BX
0FDC:076B 8B1D          MOV     BX,[DI]
0FDC:076D 2BD8          SUB     BX,AX
0FDC:076F 8BC3          MOV     AX,BX
0FDC:0771 99            CWD
0FDC:0772 034504        ADD     AX,[DI+04]
-------------------------------------------
0FDC:0775 135506        ADC     DX,[DI+06]
0FDC:0778 894504        MOV     [DI+04],AX
0FDC:077B 895506        MOV     [DI+06],DX
0FDC:077E 8B5D02        MOV     BX,[DI+02]
0FDC:0781 F7FB          IDIV    BX
0FDC:0783 EB02          JMP     0787
0FDC:0785 8B05          MOV     AX,[DI]
0FDC:0787 894508        MOV     [DI+08],AX
0FDC:078A 5B            POP     BX
0FDC:078B 58            POP     AX
0FDC:078C 5A            POP     DX
0FDC:078D 9D            POPF
0FDC:078E C3            RET
0FDC:0775 135506        ADC     DX,[DI+06]
0FDC:0778 894504        MOV     [DI+04],AX
0FDC:077B 895506        MOV     [DI+06],DX
0FDC:077E 8B5D02        MOV     BX,[DI+02]
0FDC:0781 F7FB          IDIV    BX
0FDC:0783 EB02          JMP     0787
0FDC:0785 8B05          MOV     AX,[DI]
0FDC:0787 894508        MOV     [DI+08],AX
0FDC:078A 5B            POP     BX
0FDC:078B 58            POP     AX
0FDC:078C 5A            POP     DX
0FDC:078D 9D            POPF
0FDC:078E C3            RET
```

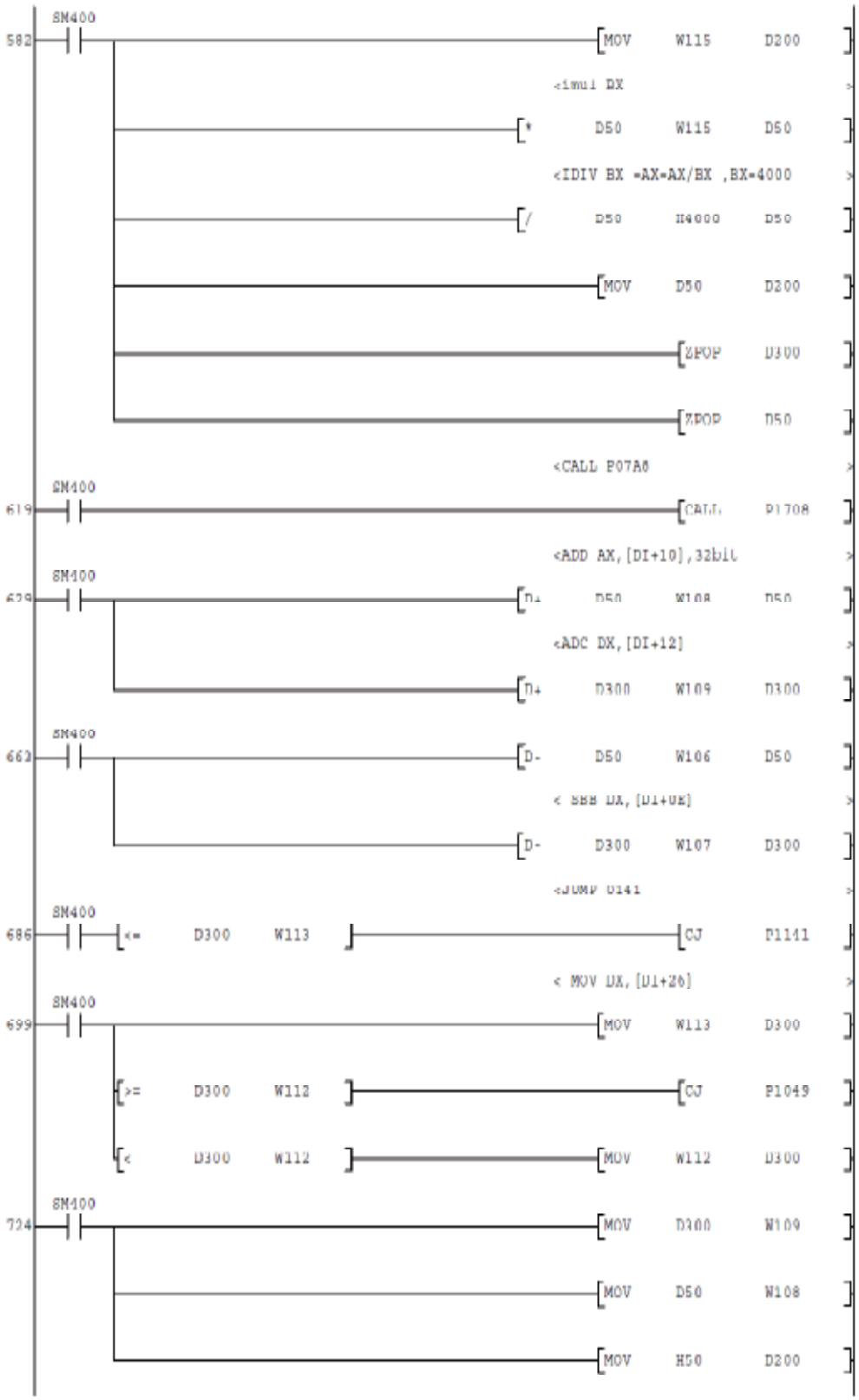
2.ส่วนที่เป็น Ladder Program

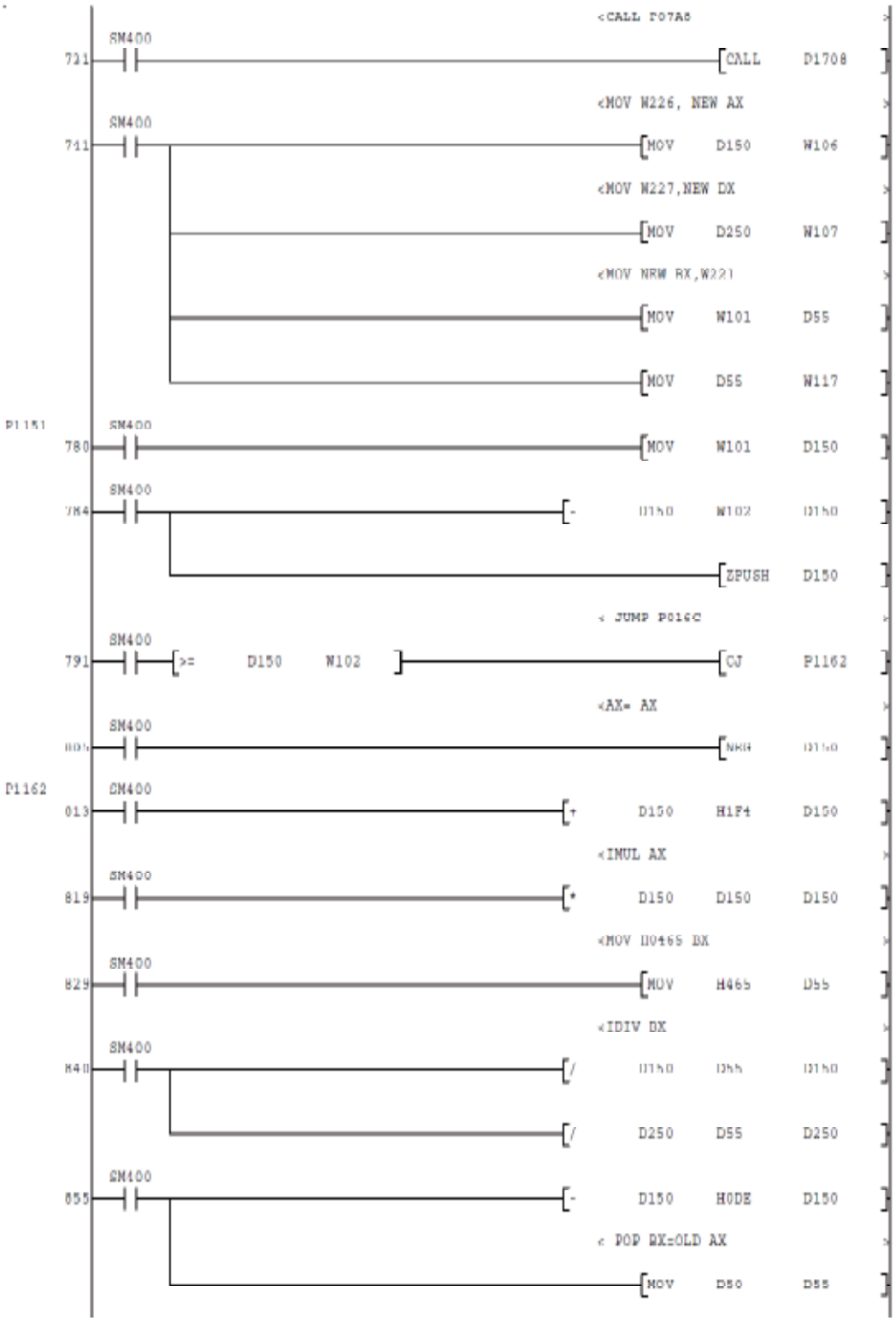


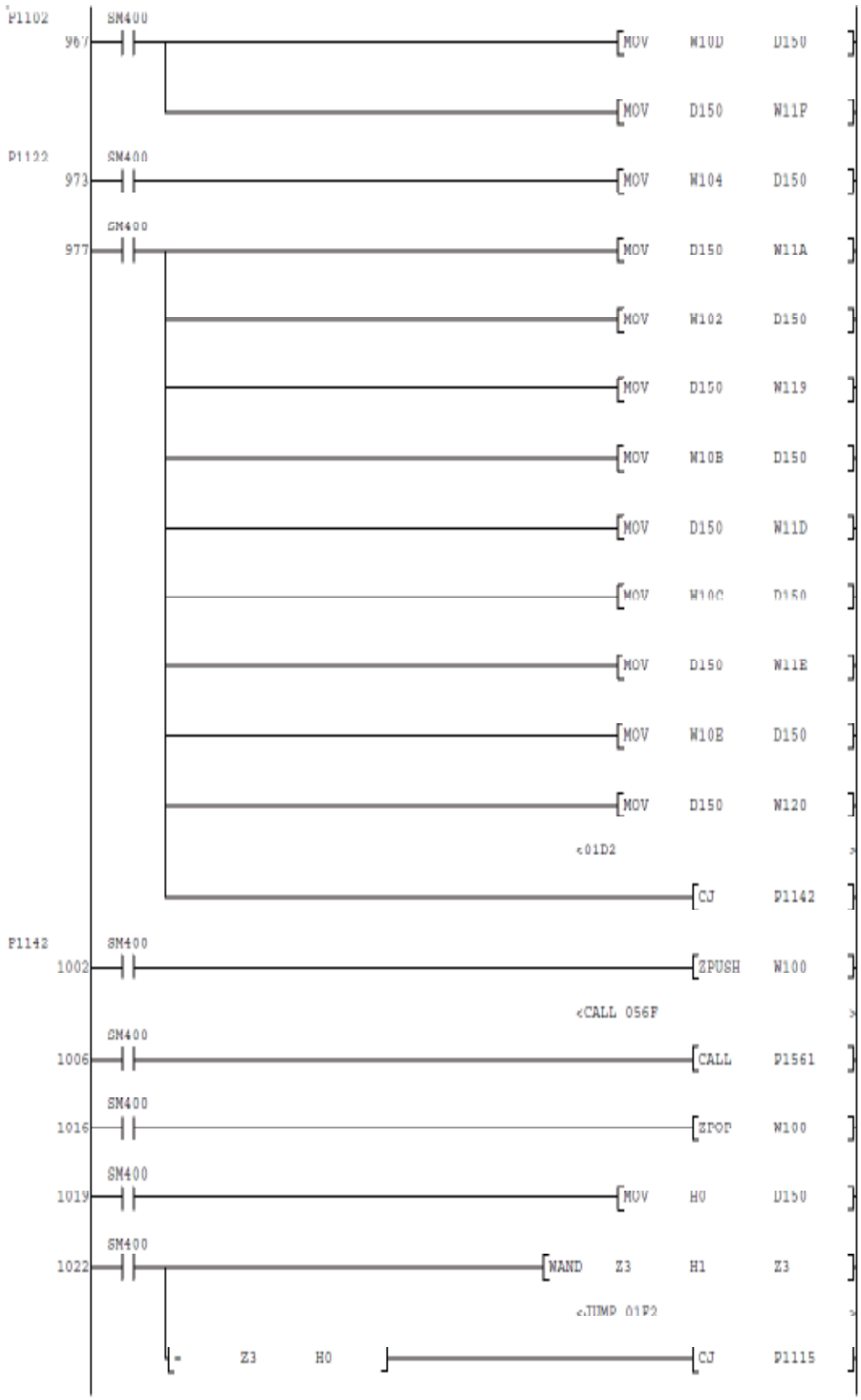

< MOV W221, AX
SM400
127
MOV W101 D50
<JD 00A8 ZS=0
SM400
139
< W101 W102
CJ P1008
<OR [SI],H10
>- W101 W102
MOR H10 Z3
<ES=0
P1008
SM400
169
MOV H0 D100Z4
<TRS? [ST],H80
SM400
177
WAND H80 Z3
<JUMP 00B2,ES=0
<> Z3 H0
CJ P1012
= Z3 H0
WXOR H10 Z3
<ES=0
P1012
SM400
212
MOV H0 D100Z4
SM400
220
WAND H80 Z3
<JUMP 00BF
= Z3 H0
CJ P1111
<JUMP 00C8
SM400
236
>= W101 W10A
CJ P1128
<JUMP 00C4
< W101 W10A
CJ P1124
<JUMP 00C8
P1111
SM400
253
<= W101 W10A
CJ P1128
<ES=0
P1124
SM400
277
MOV H0 D100Z4
SM400
285
WAND H0DF Z3
<ES=0
P1128
SM400
289
MOV H0 D100Z4

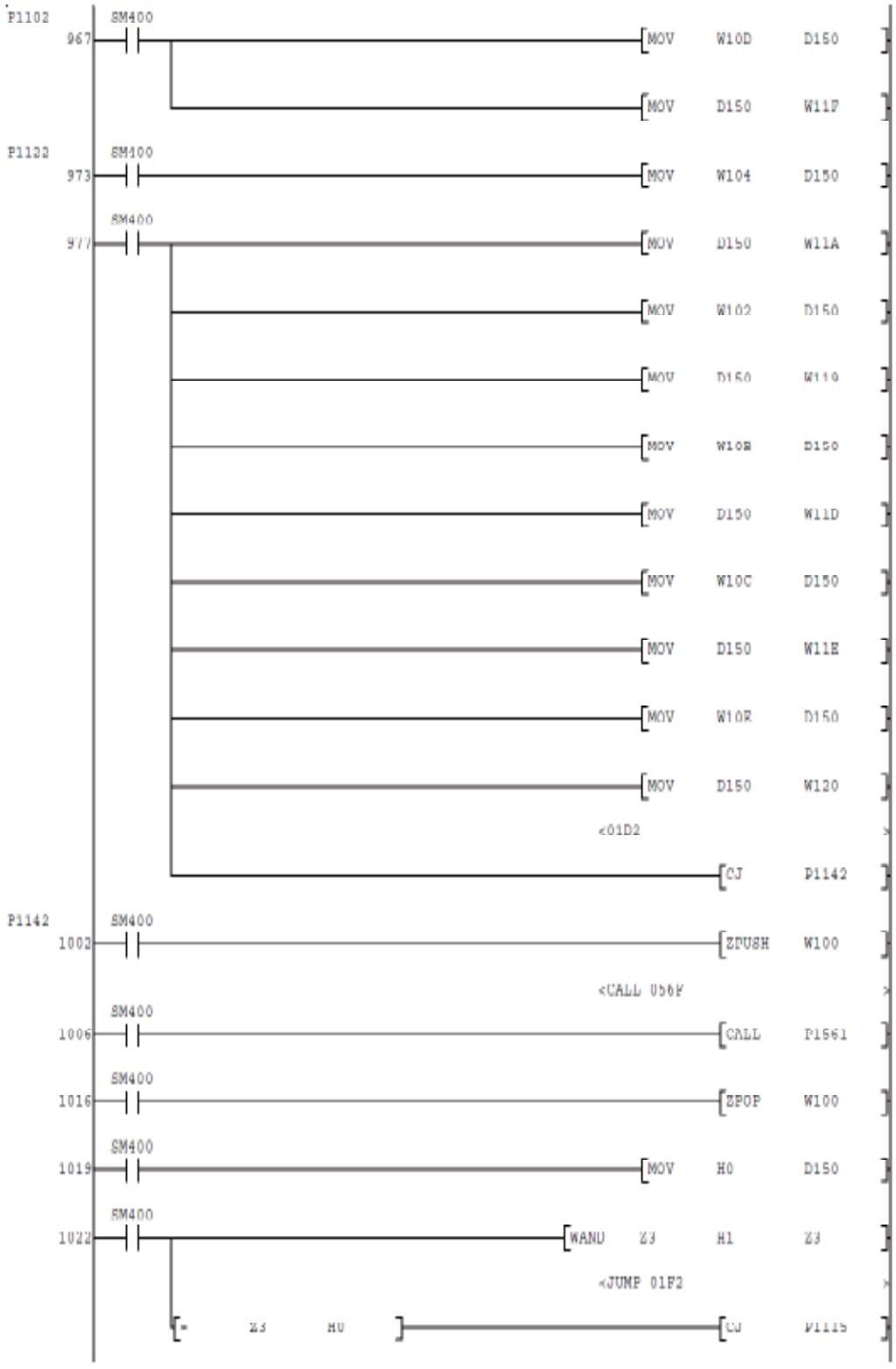

SM400
297
WAND H1 Z3
<JUMP 00D1
<> Z3 H0
CJ P1131
<JUMP 015B
= Z3 H0
CJ P1151
<ES=0
P1131
SM400
327
MOV H0 D100Z4
< TEST [SI],04
SM400
335
WAND H4 Z3
<JUMP 00DA
<> Z3 H0
CJ P1130
<JUMP 015B
= Z3 H0
CJ P1151
<MOV CX,[DI+08]
SM400
374
MOV W104 C0
<JUMP 015B
< W104 W105
CJ P1151
<MOV AX,[DI+02]
SM400
398
MOV W101 D50
- D50 W117 D50
<CALL 078F
SM400
414
CALL P1761
<MOV BX[DI+22]
SM400
424
MOV W111 D200

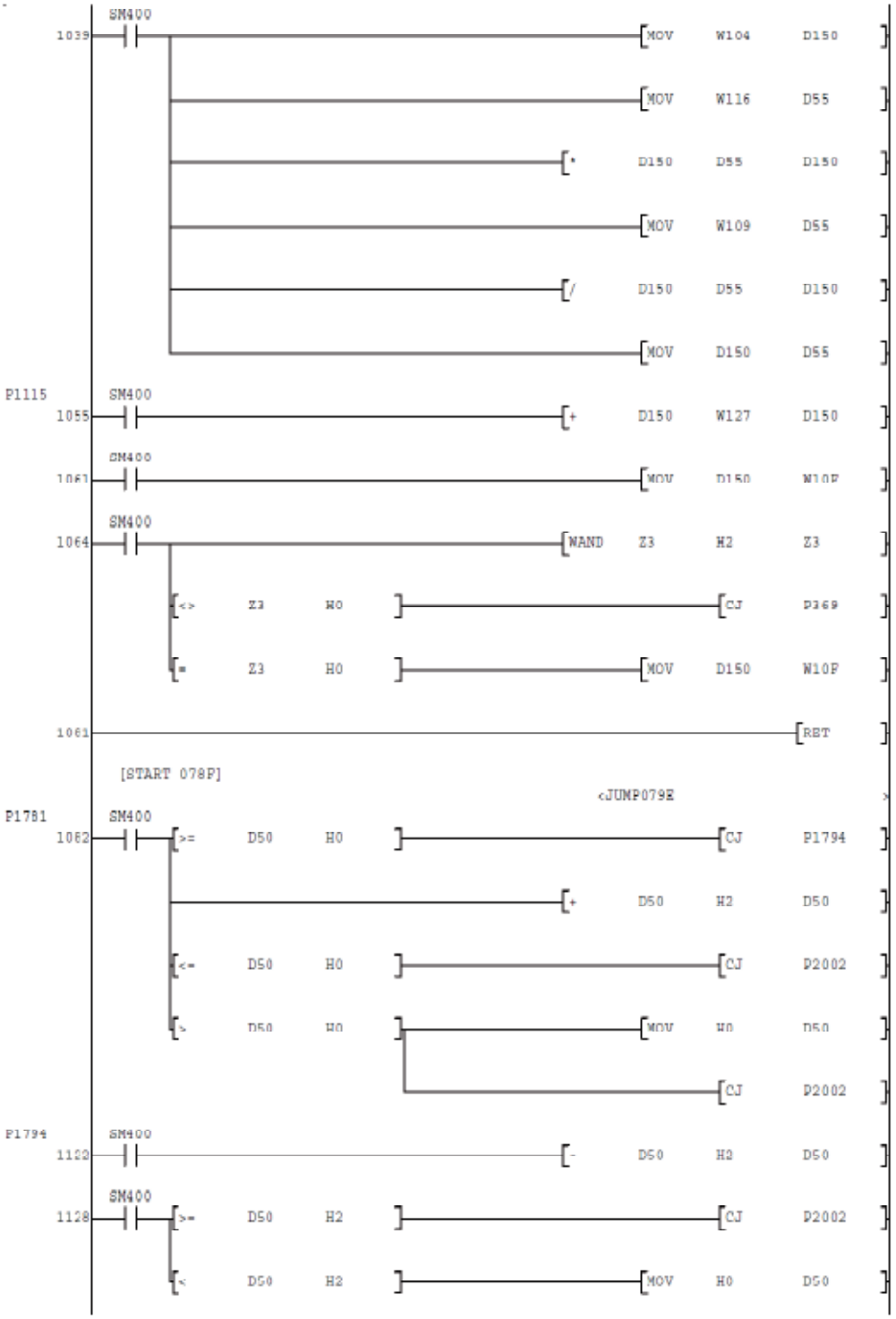

<AX=AX*BX,D11 SIGN
SM400
436
*
D50
D200
D50
<JUMP 015B
<
D11
H0FFFF
CJ
P1151
<00FE
>
D11
H0FFFF
CJ
P1114
<JUMP 0108
=
D11
H0
>
D50
H0FC18
CJ
P1108
<JUMP 015B
<=
D50
H0FC18
CJ
P1151
P1114
SM400
504
>
D11
H0
CJ
P1151
<JUMP 015B
<=
D11
H0
>
D50
H3E8
CJ
P1151
P1108
SM400
528
+
D50
W104
D50
SM400
534
<=
D50
H0
CJ
P1151
<MOV DX,AX
SM400
540
MOV
D50
D300
<AX=0
SM400
550
WXOR
D50
D50
SM400
558
MOV
W103
D200
<=
D200
H5
CJ
P1151
ZPUSH
D50
ZPUSH
D300
<mov ax,bx
SM400
572
MOV
D200
D50

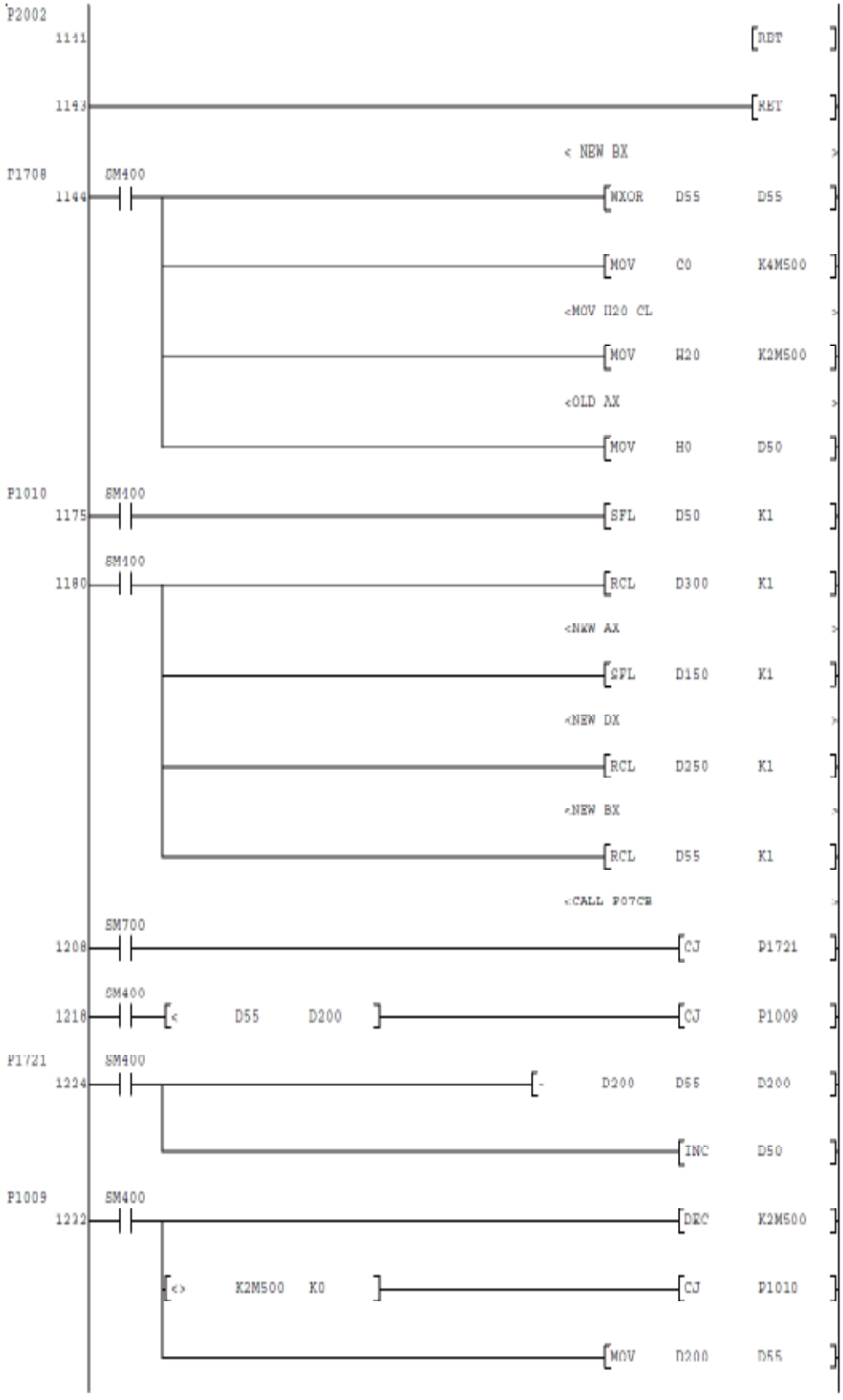

582
SM400
MOV W115 D200
<imul BX
* D50 W115 D50
<IDIV BX =AX=AX/BX ,BX=4000
/ D50 H4000 D50
MOV D50 D200
&POP D300
%POP D50
<CALL P07A8
619
SM400
CALL P1708
<ADD AX,[DI+10],32bit
SM400
<ADC DX,[DI+12]
D+ D300 W109 D300
662
SM400
D- D50 W106 D50
< SBB DX,[DI+0E]
D- D300 W107 D300
<JUMP P141
686
SM400
<= D300 W113
CJ P1141
< MOV DX,[DI+26]
699
SM400
MOV W113 D300
>= D300 W112
CJ P1049
< D300 W112
MOV W112 D300
724
SM400
MOV D300 W109
MOV D50 W108
MOV H50 D200


<CALL F07A6
SM400
721
CALL D1708
<MOV W226, NEW AX
SM400
741
MOV D150 W106
<MOV W227,NEW DX
MOV D250 W107
<MOV NEW BX,W221
MOV W101 D55
MOV D55 W117
P1151
SM400
780
MOV W101 D150
SM400
784
ZPUSH D150
< JUMP P016C
SM400
791
>= D150 W102
CJ P1162
<AX= AX
SM400
P1162
SM400
813
D150 H1F4 D150
<IMUL AX
SM400
819
D150 D150 D150
<MOV H0465 BX
SM400
829
MOV H465 D55
<IDIV BX
SM400
840
D150 D55 D150
D250 D55 D250
SM400
855
D150 H0DE D150
< POP BX=OLD AX
MOV D50 D55

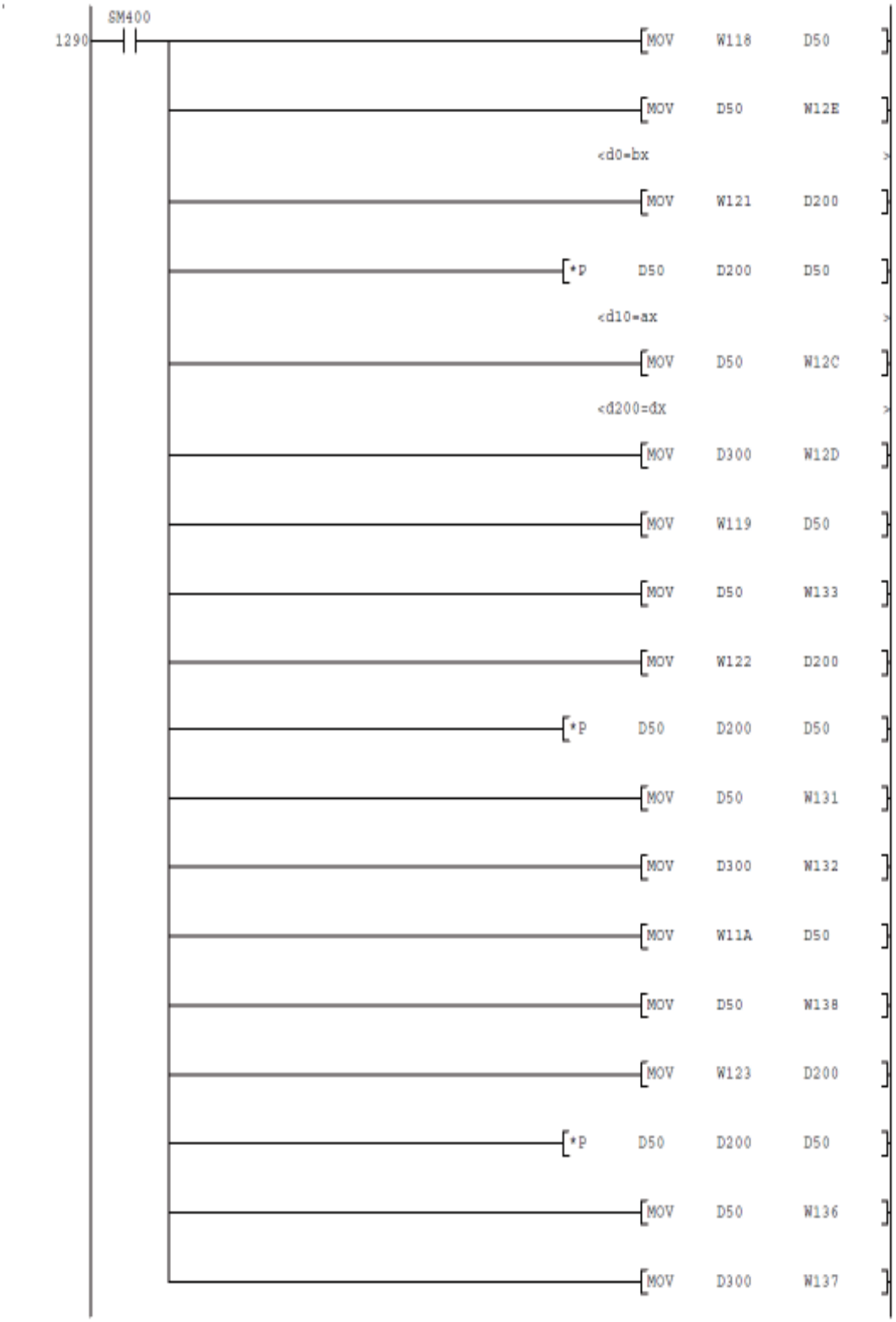

P1102
967
SM400
MOV W10D D150
MOV D150 W11F
P1122
973
SM400
MOV W104 D150
977
SM400
MOV D150 W11A
MOV W102 D150
MOV D150 W119
MOV W10B D150
MOV D150 W11D
MOV W10C D150
MOV D150 W11E
MOV W10E D150
MOV D150 W120
<01D2
CJ P1142
P1142
1002
SM400
ZPUSH W100
<CALL 056F
1006
SM400
CALL P1561
1016
SM400
ZPOP W100
1019
SM400
MOV H0 D150
1022
SM400
NAND Z3 H1 Z3
<JUMP 01F2
= Z3 H0
CJ P1115

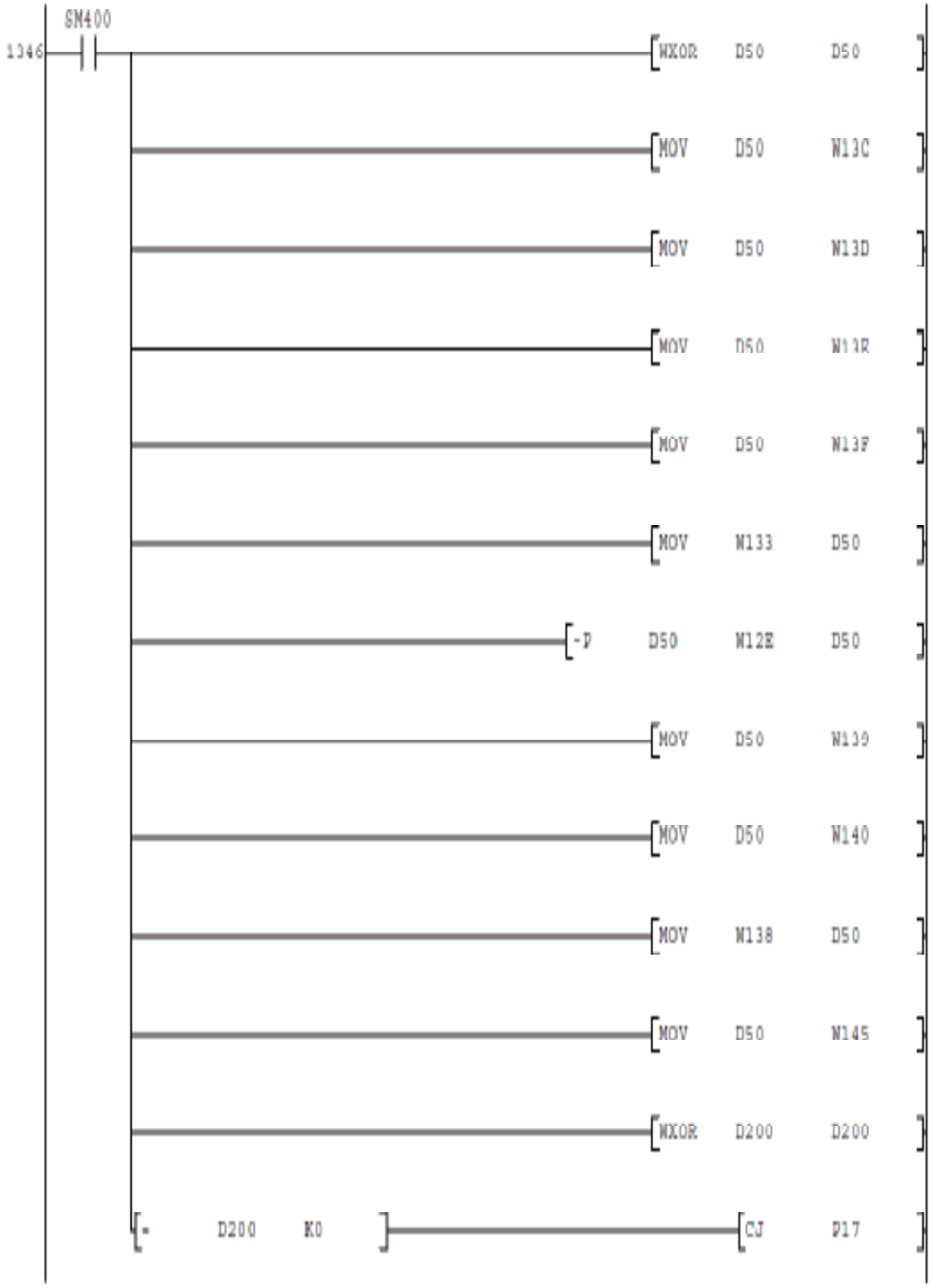

P1102
967
SM400
MOV W10D D150
MOV D150 W11F
P1122
973
SM400
MOV W104 D150
977
SM400
MOV D150 W11A
MOV W102 D150
MOV D150 W110
MOV W10B D150
MOV D150 W11D
MOV W10C D150
MOV D150 W11E
MOV W10R D150
MOV D150 W120
<01D2
CJ P1142
P1142
1002
SM400
ZPUSH W100
<CALL U56F
1006
SM400
CALL P1561
1016
SM400
ZPOP W100
1019
SM400
MOV H0 D150
1022
SM400
WANU Z3 H1 Z3
<JUMP 01F2
Z3 H0
CJ P1115

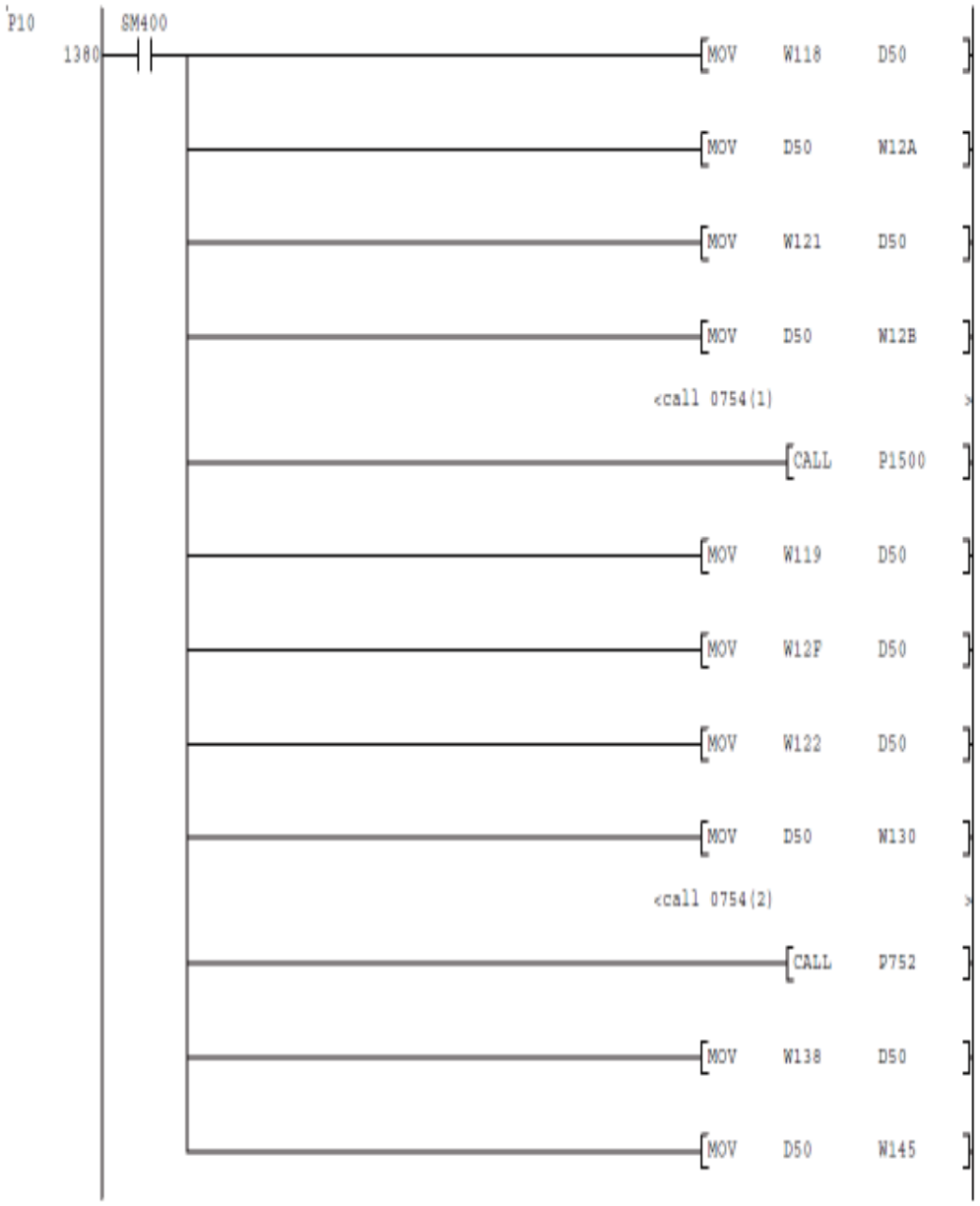

SM400
1039
MOV W104 D150
MOV W116 D55
* D150 D55 D150
MOV W109 D55
/ D150 D55 D150
MOV D150 D55
P1115
SM400
1055
+ D150 W127 D150
SM400
1061
MOV D150 W10F
SM400
1064
WAND Z3 H2 Z3
<> Z3 W0 CJ P369
= Z3 H0 MOV D150 W10F
1081
RET
[START 078P]
<JUMP079E
P1781
SM400
1082
>= D50 H0 CJ P1794
+ D50 H2 D50
<= D50 H0 CJ P2002
< D50 W0 MOV W0 D50
CJ P2002
P1794
SM400
1122
- D50 H2 D50
SM400
1128
>= D50 H2 CJ P2002
< D50 H2 MOV H0 D50

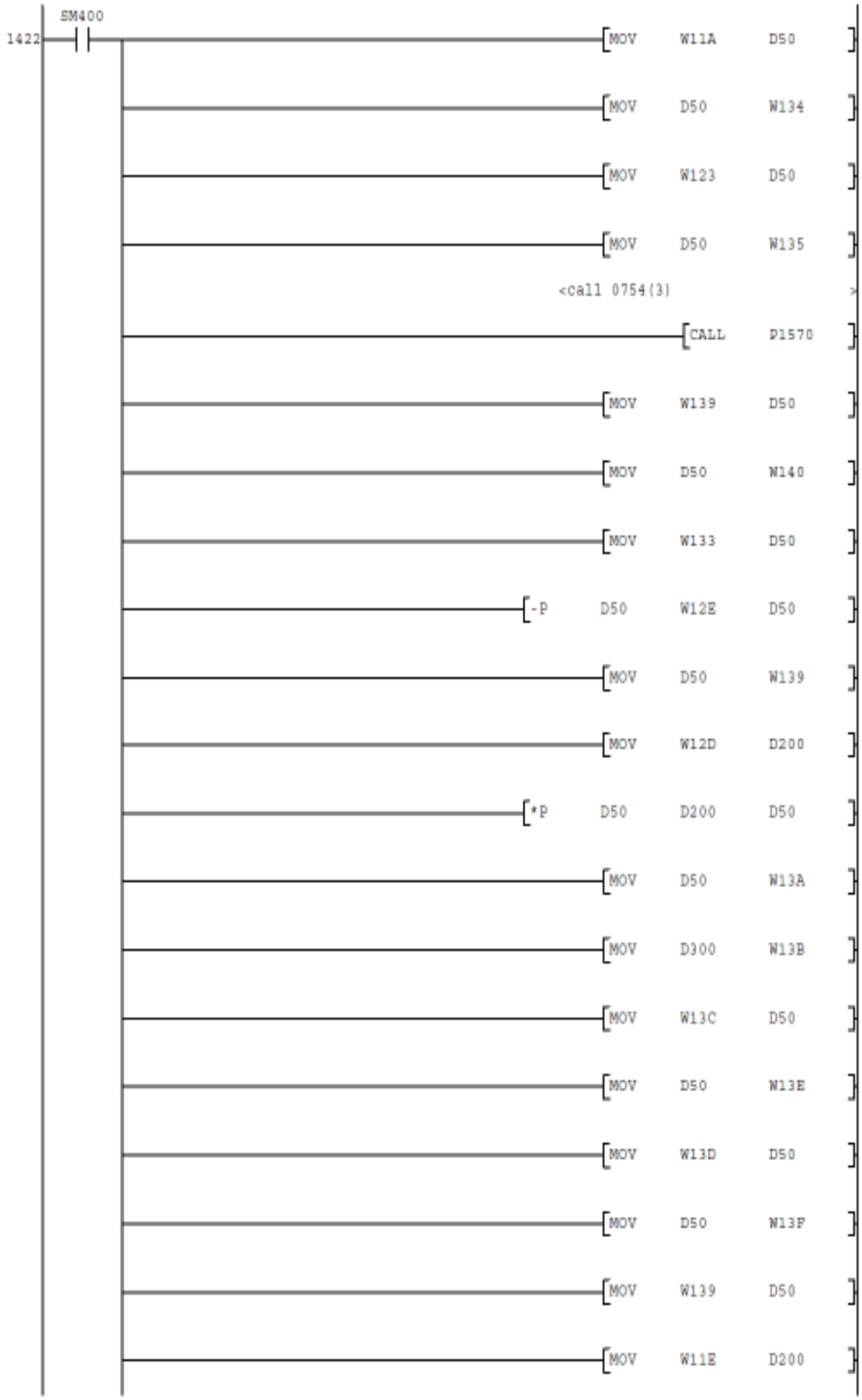

P2002
1141
RBT
1143
RET
< NEW BX
T1708
SM400
1144
WXOR D55 D55
MOV C0 K4M500
<MOV H20 CL
MOV H20 K2M500
<OLD AX
MOV H0 D50
P1010
SM400
1175
SFL D50 K1
SM400
1180
RCL D300 K1
<NEW AX
SFL D150 K1
<NEW DX
RCL D250 K1
<NEW BX
RCL D55 K1
<CALL P07CB
SM700
1208
CJ P1721
SM400
1218
< D55 D200
CJ P1009
P1721
SM400
1224
- D200 D55 D200
INC D50
P1009
SM400
1222
DEC K2M500
<> K2M500 K0
CJ P1010
MOV D200 D55

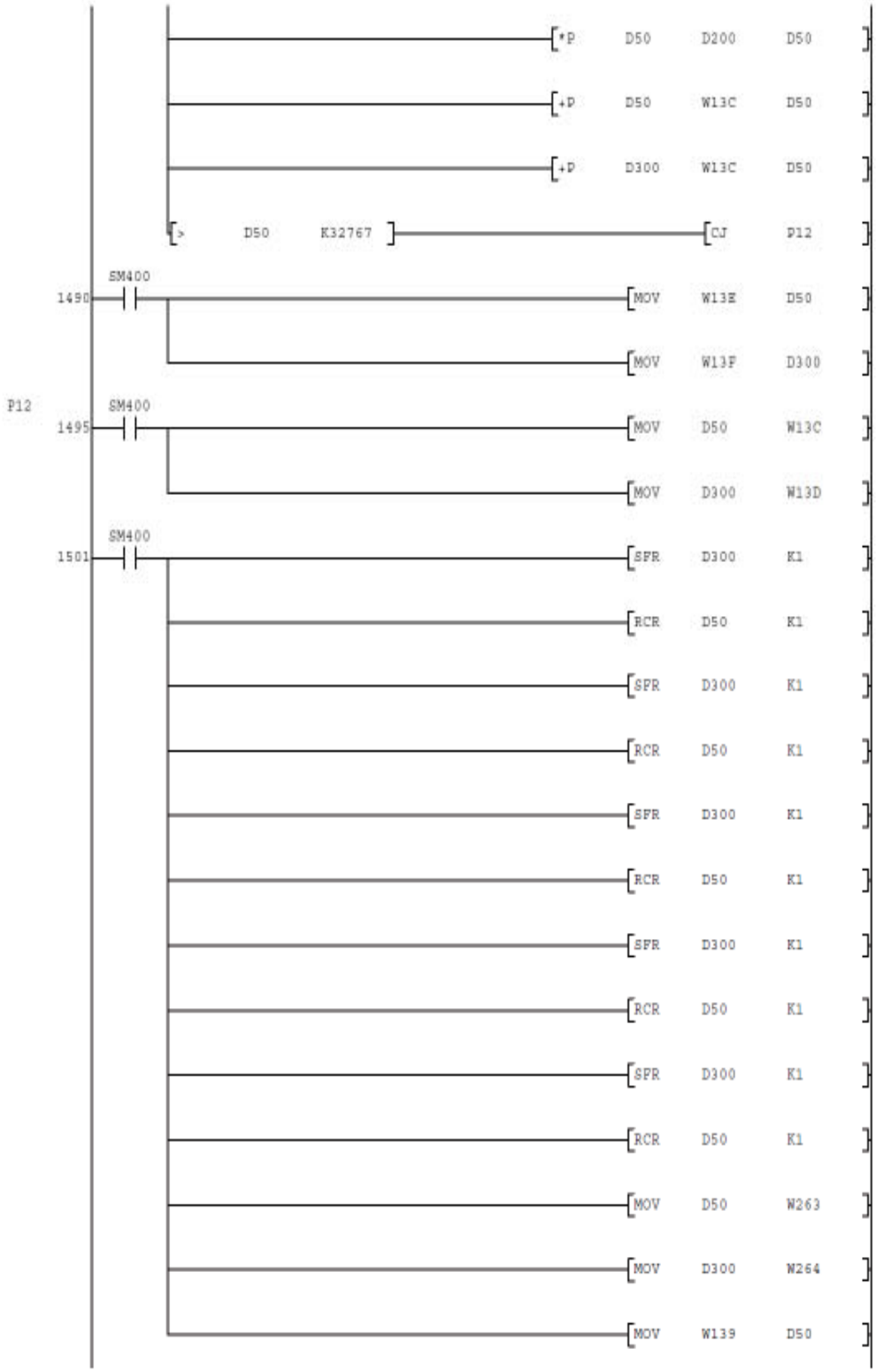

<NEW CX
SM400
1245
MOV K4M500 C3
1254
RET
[START 056F]
<di=W0
P1561
SM400
1255
MOV H0 W118
<cmp bit2<>0 to jump
SM400
1271
MOV W11B K4M100
M102
1287
CJ P10

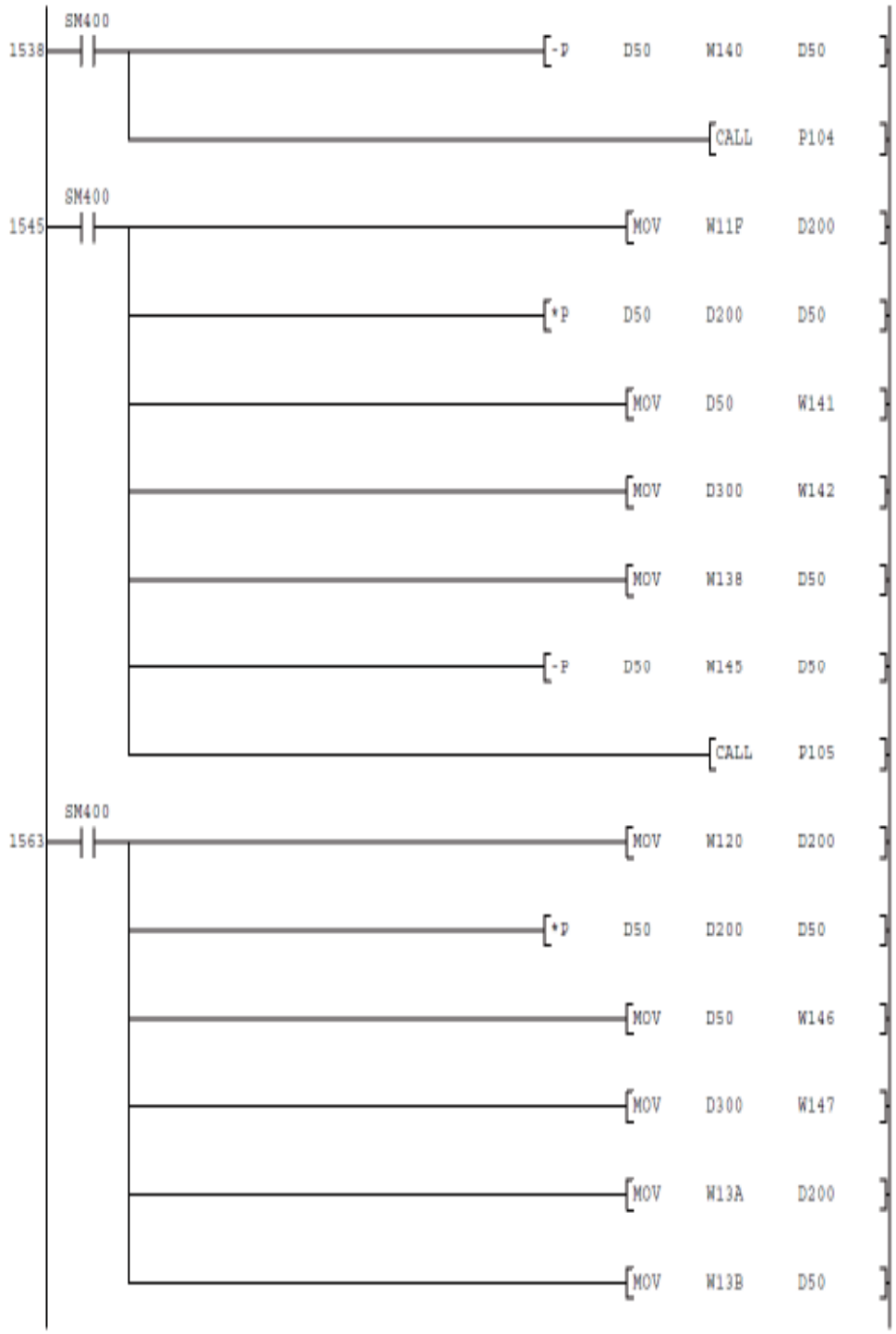

SM400
1290
MOV W118 D50
MOV D50 W12E
<d0=bx
MOV W121 D200
*P D50 D200 D50
<d10=ax
MOV D50 W12C
<d200=dx
MOV D300 W12D
MOV W119 D50
MOV D50 W133
MOV W122 D200
*P D50 D200 D50
MOV D50 W131
MOV D300 W132
MOV W11A D50
MOV D50 W138
MOV W123 D200
*P D50 D200 D50
MOV D50 W136
MOV D300 W137

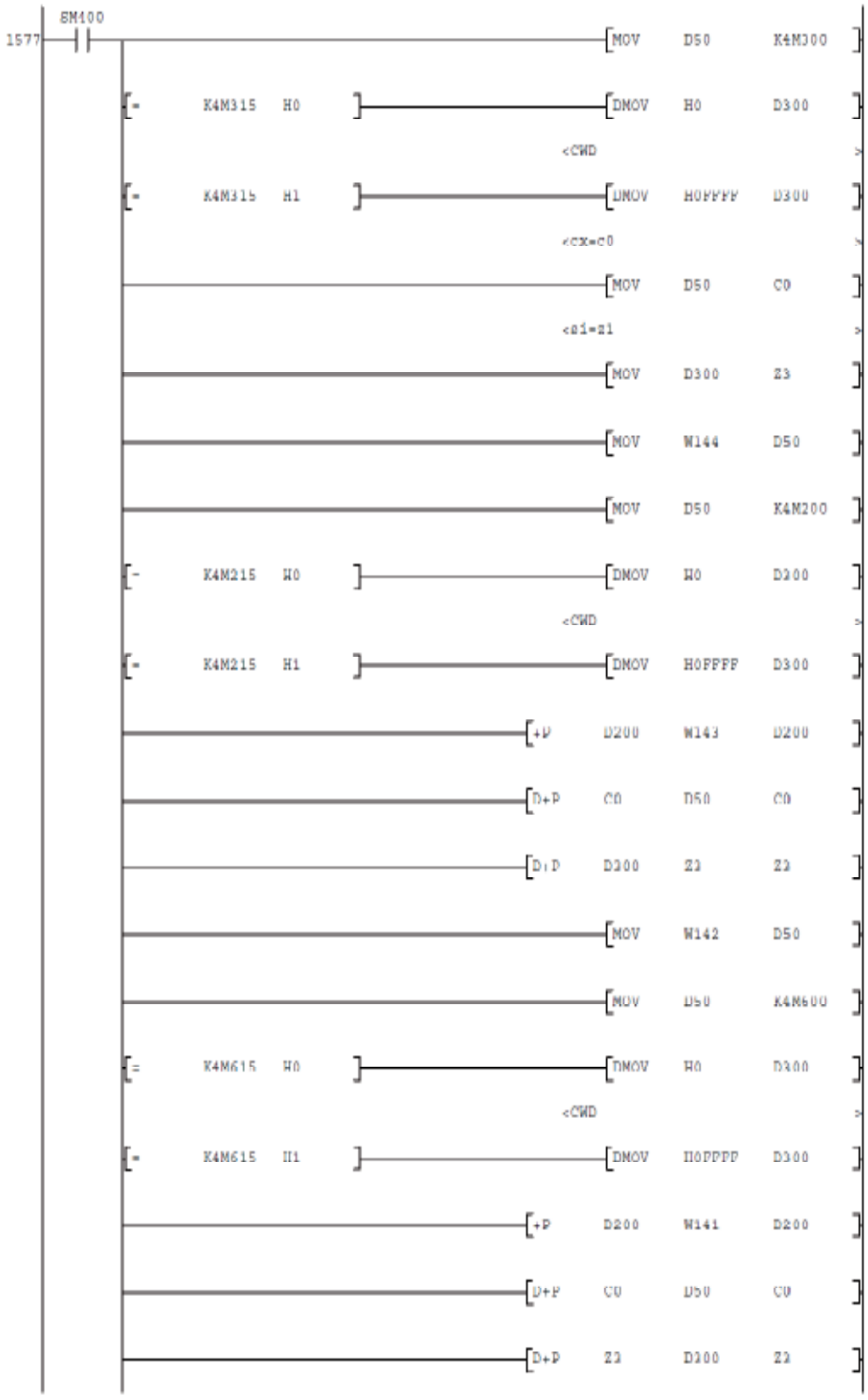

SM400
1346
WXOR D50 D50
MOV D50 W13C
MOV D50 W13D
MOV D50 W13E
MOV D50 W13F
MOV W133 D50
-P D50 W12E D50
MOV D50 W139
MOV D50 W140
MOV W138 D50
MOV D50 W145
WXOR D200 D200
D200 K0
CJ P17

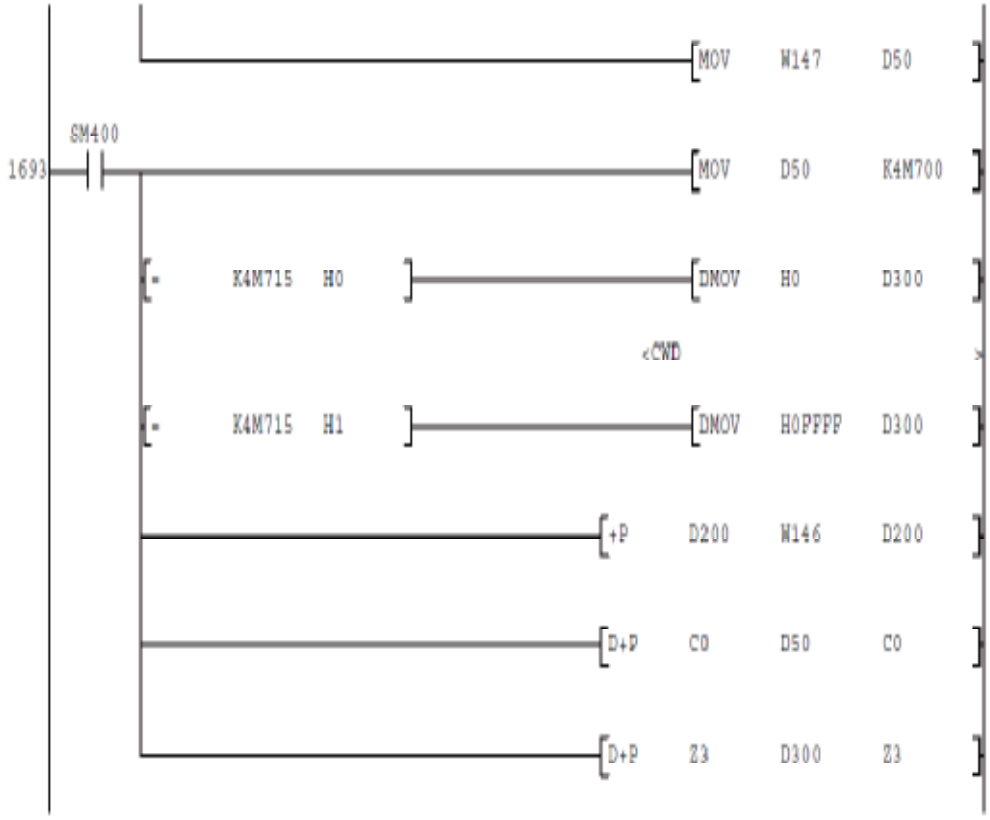

P10
1380
SM400
MOV W118 D50
MOV D50 W12A
MOV W121 D50
MOV D50 W12B
<call 0754(1)
CALL P1500
MOV W119 D50
MOV W12F D50
MOV W122 D50
MOV D50 W130
<call 0754(2)
CALL P752
MOV W138 D50
MOV D50 W145

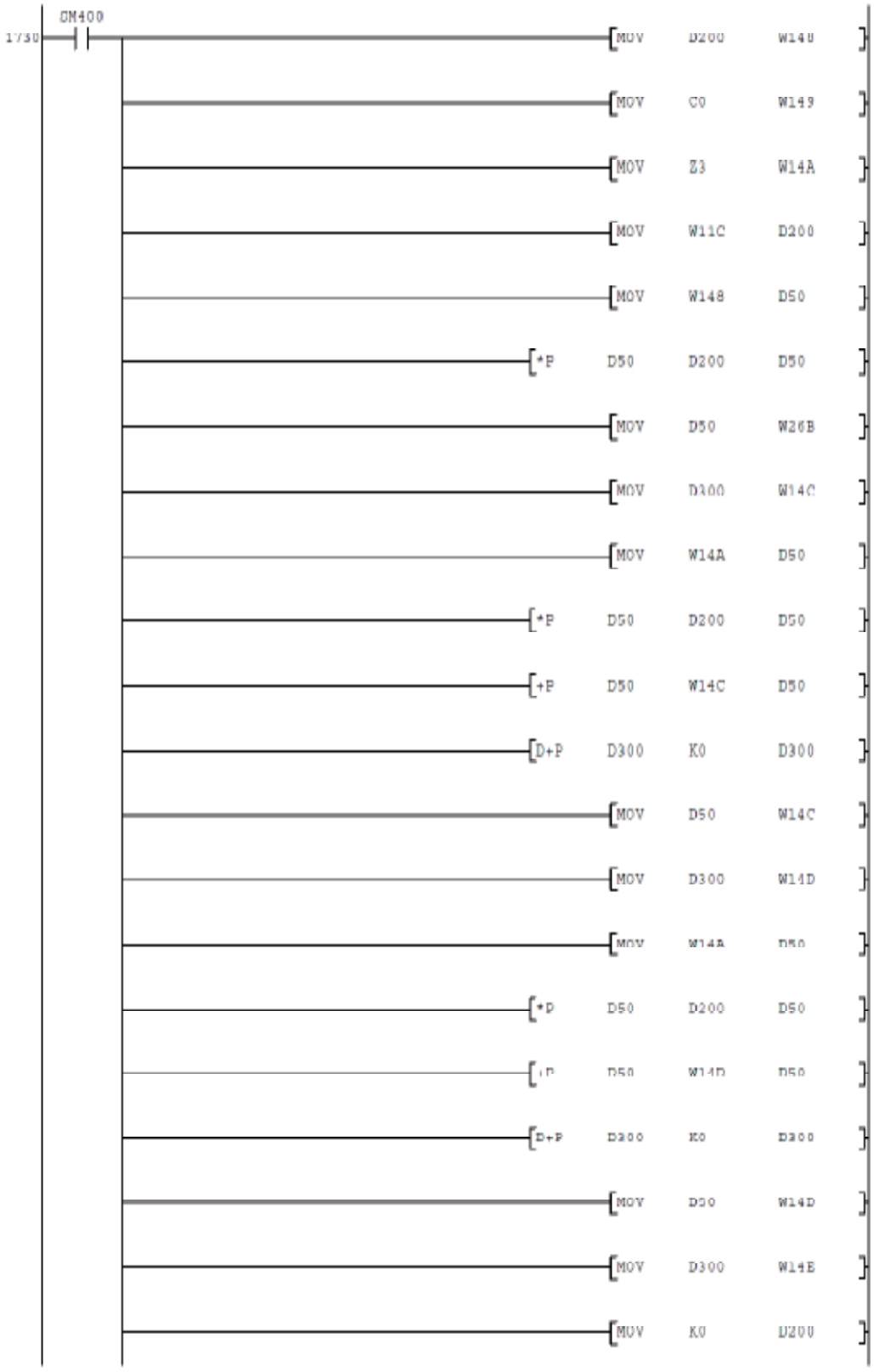

SM400
1422
MOV W11A D50
MOV D50 W134
MOV W123 D50
MOV D50 W135
<call 0754(3)
CALL P1570
MOV W139 D50
MOV D50 W140
MOV W133 D50
-P D50 W12E D50
MOV D50 W139
MOV W12D D200
*P D50 D200 D50
MOV D50 W13A
MOV D300 W13B
MOV W13C D50
MOV D50 W13E
MOV W13D D50
MOV D50 W13F
MOV W139 D50
MOV W11E D200


*P D50 D200 D50
+P D50 W13C D50
+P D300 W13C D50
> D50 K32767
CJ P12
SM400
1490
MOV W13E D50
MOV W13F D300
P12
SM400
1495
MOV D50 W13C
MOV D300 W13D
SM400
1501
SFR D300 K1
RCR D50 K1
SFR D300 K1
RCR D50 K1
SFR D300 K1
RCR D50 K1
SFR D300 K1
RCR D50 K1
SFR D300 K1
RCR D50 K1
MOV D50 W263
MOV D300 W264
MOV W139 D50

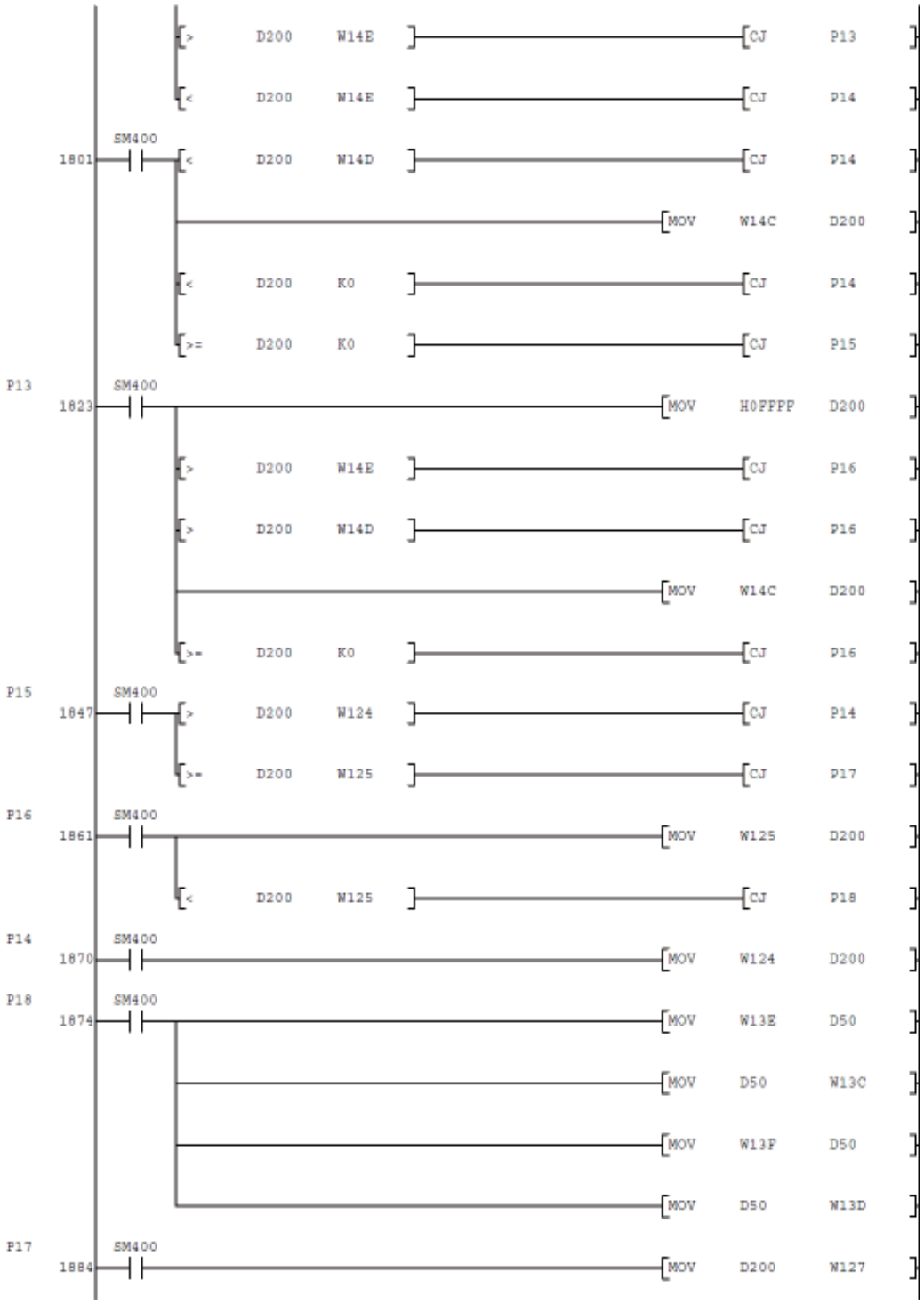

SM400
1538
-P D50 W140 D50
CALL P104
SM400
1545
MOV W11F D200
*P D50 D200 D50
MOV D50 W141
MOV D300 W142
MOV W138 D50
-P D50 W145 D50
CALL P105
SM400
1563
MOV W120 D200
*P D50 D200 D50
MOV D50 W146
MOV D300 W147
MOV W13A D200
MOV W13B D50

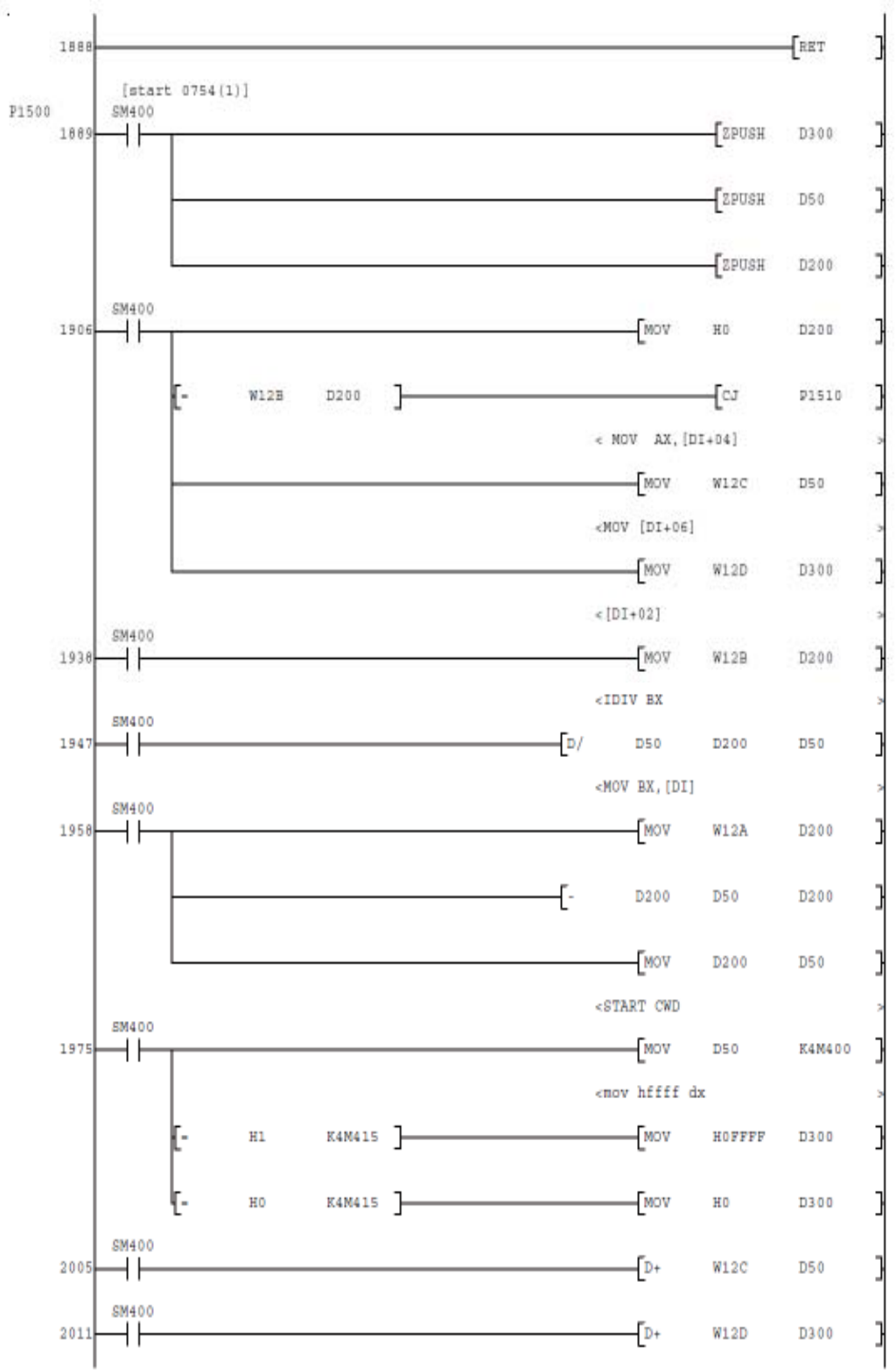

1577
SM400
MOV D50 K4M300
= K4M315 H0
DMOV H0 D300
<CWD
= K4M315 H1
DMOV H0FFFF D300
<CX=C0
MOV D50 C0
<Z1=Z1
MOV D300 Z3
MOV W144 D50
MOV D50 K4M200
= K4M215 H0
DMOV H0 D200
<CWD
= K4M215 H1
DMOV H0FFFF D300
+P D200 W143 D200
D+P C0 D50 C0
D+P D200 Z3 Z3
MOV W142 D50
MOV D50 K4M600
= K4M615 H0
DMOV H0 D300
<CWD
= K4M615 H1
DMOV H0FFFF D300
+P D200 W141 D200
D+P C0 D50 C0
D+P Z3 D300 Z3

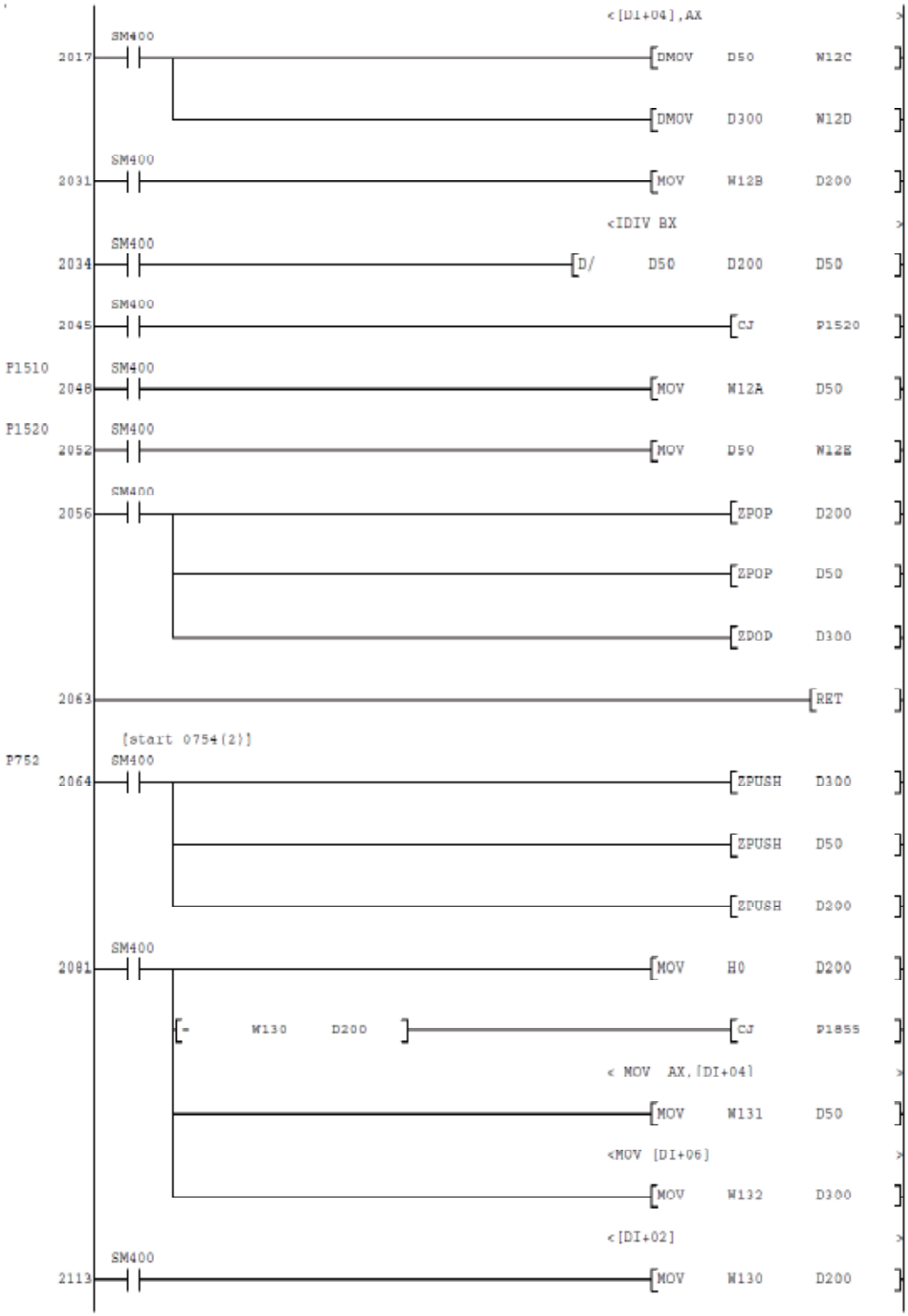


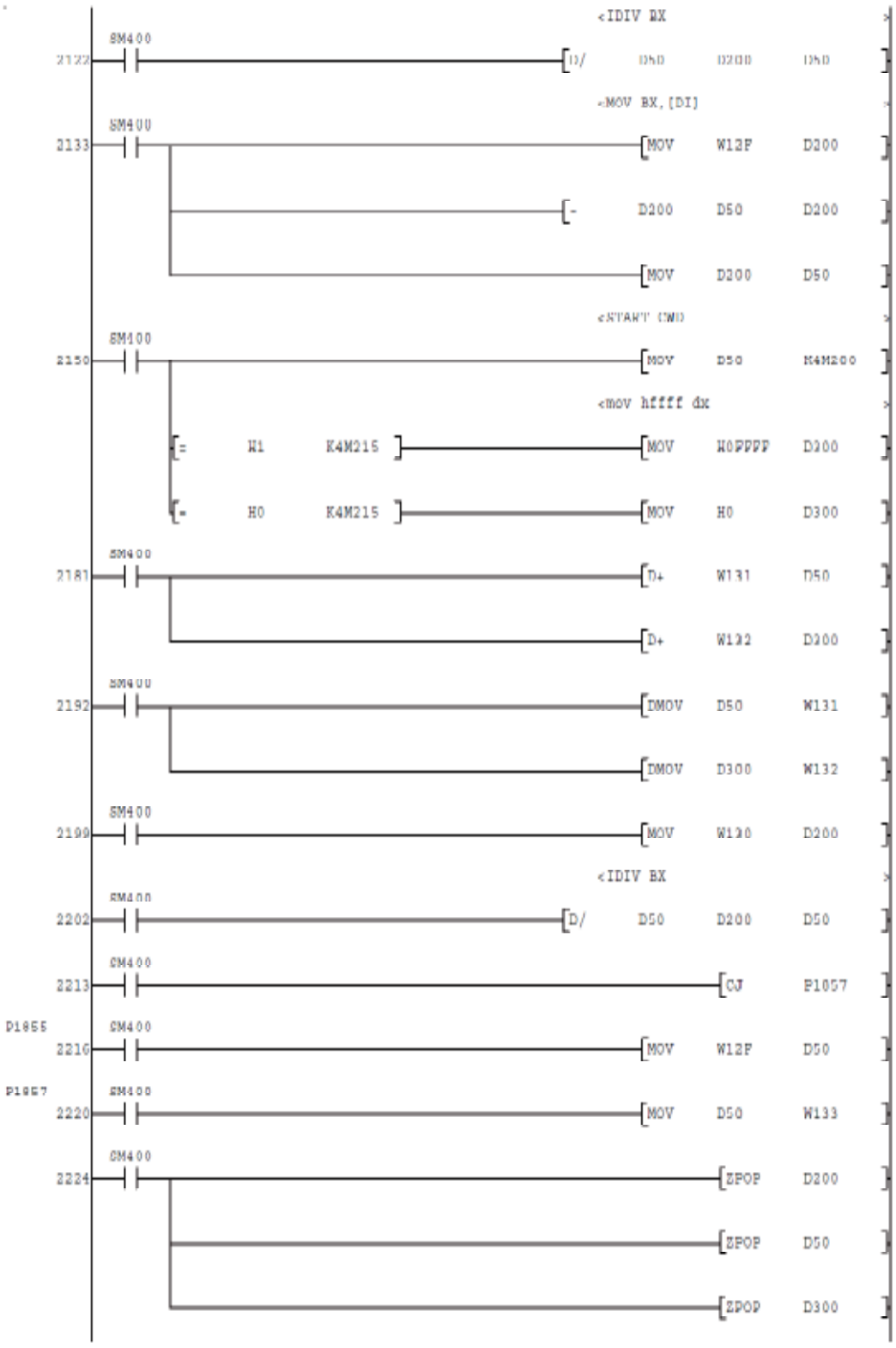

SM400
MOV D200 W140
MOV C0 W149
MOV Z3 W14A
MOV W11C D200
MOV W148 D50
*P D50 D200 D50
MOV D50 W26B
MOV D300 W14C
MOV W14A D50
*P D50 D200 D50
+P D50 W14C D50
D+P D300 K0 D300
MOV D50 W14C
MOV D300 W14D
*P D50 D200 D50
D+P D300 K0 D300
MOV D50 W14D
MOV D300 W14E
MOV K0 D200

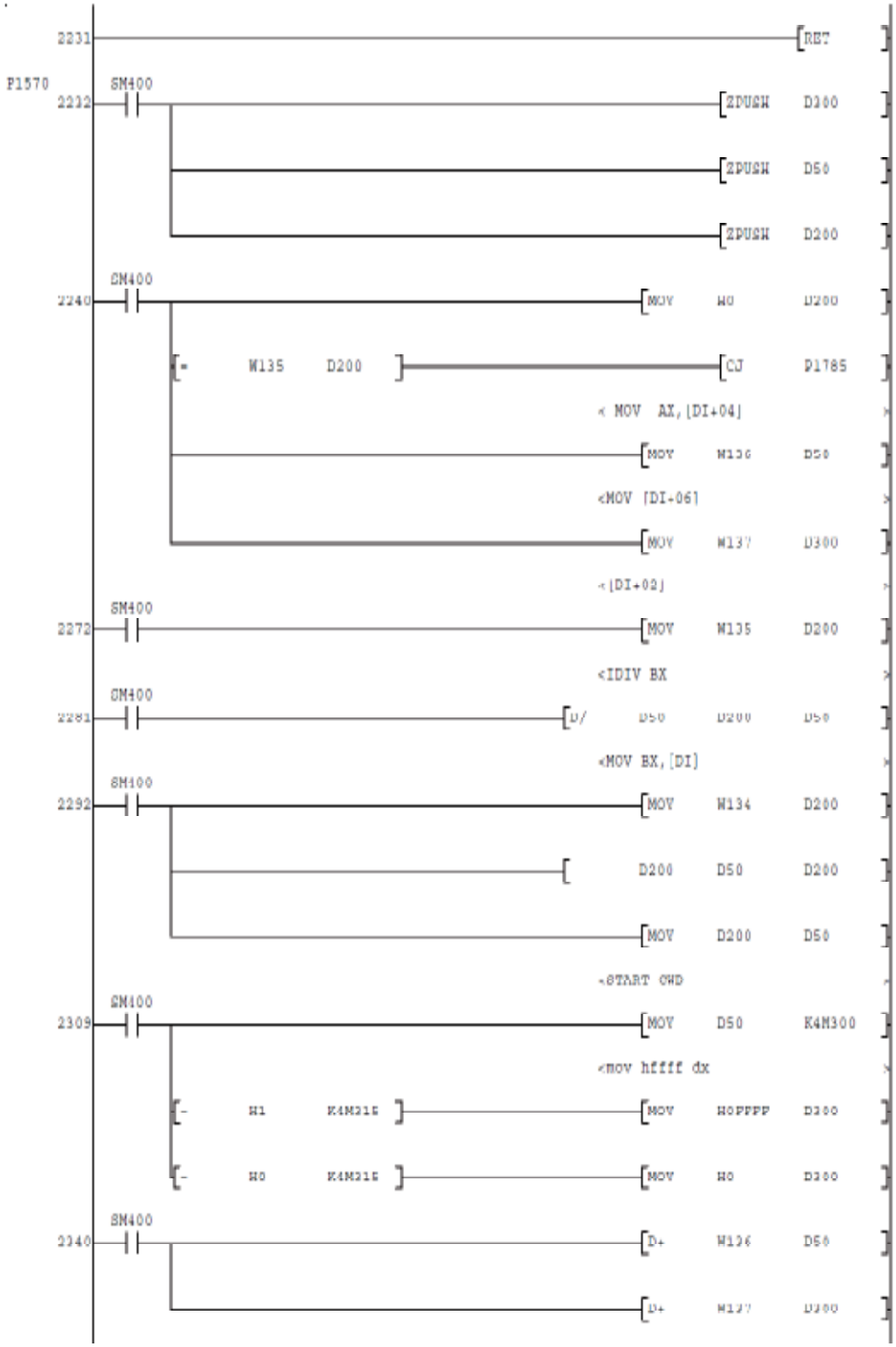

TNI
THAI - NICHI INSTITUTE OF TECHNOLOGY

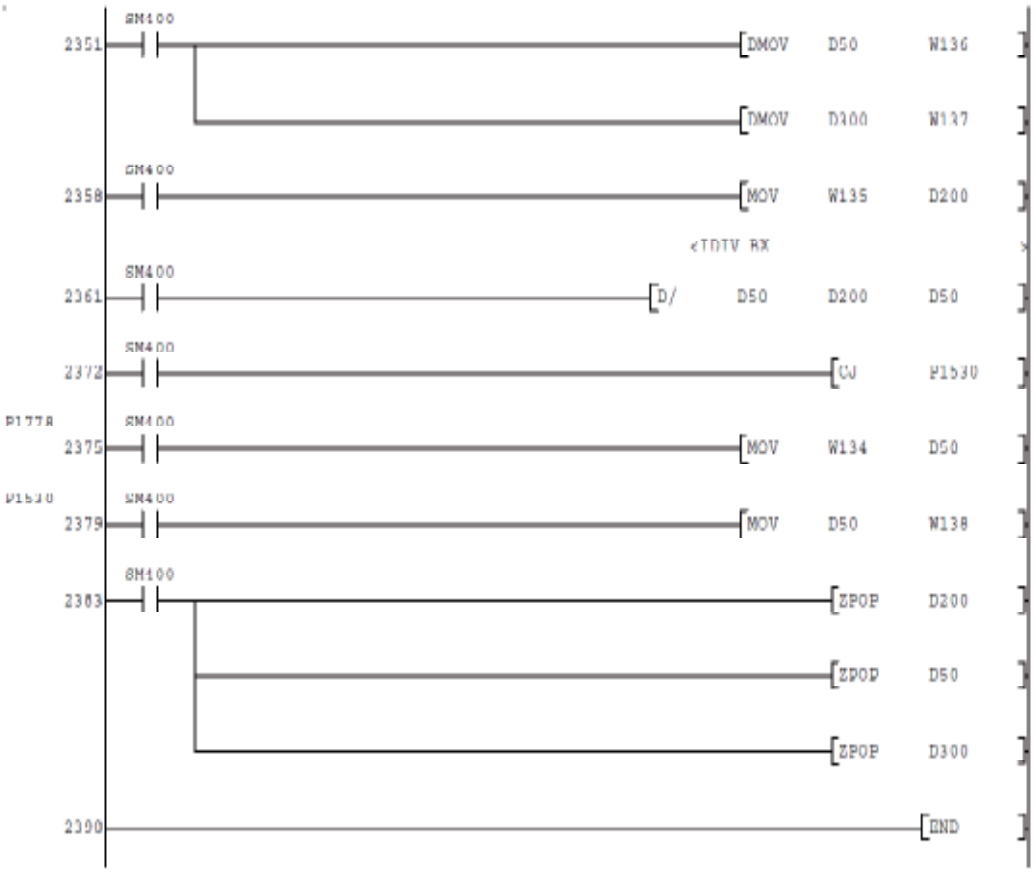

> D200 W14E CJ P13
< D200 W14E CJ P14
SM400
1801 < D200 W14D CJ P14
MOV W14C D200
< D200 K0 CJ P14
>= D200 K0 CJ P15
P13
SM400
1823 MOV H0FFFF D200
> D200 W14E CJ P16
> D200 W14D CJ P16
MOV W14C D200
>= D200 K0 CJ P16
P15
SM400
1847 > D200 W124 CJ P14
>= D200 W125 CJ P17
P16
SM400
1861 MOV W125 D200
< D200 W125 CJ P18
P14
SM400
1870 MOV W124 D200
P18
SM400
1874 MOV W13E D50
MOV D50 W13C
MOV W13F D50
MOV D50 W13D
P17
SM400
1884 MOV D200 W127


1888
RET
[start 0754(1)]
P1500
SM400
1889
ZPUSH D300
ZPUSH D50
ZPUSH D200
SM400
1906
MOV H0 D200
= W12B D200
CJ P1510
< MOV AX,[DI+04]
MOV W12C D50
<MOV [DI+06]
MOV W12D D300
<[DI+02]
SM400
1938
MOV W12B D200
<IDIV BX
SM400
1947
D/ D50 D200 D50
<MOV BX,[DI]
SM400
1958
MOV W12A D200
- D200 D50 D200
MOV D200 D50
<START CWD
SM400
1975
MOV D50 K4M400
<mov hffff dx
= H1 K4M415
MOV H0FFFF D300
= H0 K4M415
MOV H0 D300
SM400
2005
D+ W12C D50
SM400
2011
D+ W12D D300


<[DI+04],AX
SM400
2017
DMOV D50 W12C
DMOV D300 W12D
SM400
2031
MOV W12B D200
<IDIV BX
SM400
2034
D/ D50 D200 D50
SM400
2045
CJ P1520
P1510
SM400
2048
MOV W12A D50
P1520
SM400
2052
MOV D50 W12E
SM400
2056
ZPOP D200
ZPOP D50
ZPOP D300
2063
RET
{start 0754(2)}
P752
SM400
2064
ZPUSH D300
ZPUSH D50
ZPUSH D200
SM400
2081
MOV H0 D200
= W130 D200
CJ P1855
< MOV AX,[DI+04]
MOV W131 D50
<MOV [DI+06]
MOV W132 D300
<[DI+02]
SM400
2113
MOV W130 D200


<IDIV BX
SM400
D/ D50 D200 D50
<MOV BX,[DI]
SM400
MOV W12F D200
- D200 D50 D200
MOV D200 D50
<START CMD
SM400
MOV D50 K4M200
<mov hffff dx
= H1 K4M215
MOV H0FFFF D200
= H0 K4M215
MOV H0 D300
SM400
D+ W131 D50
D+ W132 D200
SM400
DMOV D50 W131
DMOV D300 W132
SM400
MOV W130 D200
<IDIV BX
D/ D50 D200 D50
SM400
CJ P1057
P1055
SM400
MOV W12F D50
P1057
SM400
MOV D50 W133
SM400
ZPOP D200
ZPOP D50
ZPOP D300


2231
RET
P1570
SM400
2232
ZPUSH D100
ZPUSH D50
ZPUSH D200
SM400
2240
MOV W0 D200
= W135 D200
CJ P1785
< MOV AX,[DI+04]
MOV W136 D50
<MOV [DI+06]
MOV W137 D300
<[DI+02]
SM400
2272
MOV W135 D200
<IDIV BX
SM400
2281
D/ D50 D200 D50
<MOV BX,[DI]
SM400
2292
MOV W134 D200
D200 D50 D200
MOV D200 D50
<START CWD
SM400
2309
MOV D50 K4M300
<mov hffff dx
- W1 K4M216
MOV H0FFFF D300
- W0 K4M216
MOV H0 D300
SM400
2340
D+ W136 D50
D+ W137 D300


SM400
2351
DMOV D50 W136
DMOV D100 W137
SM400
2358
MOV W135 D200
<TDTV RX
SM400
2361
D/ D50 D200 D50
SM400
2372
CJ P1530
P1778
SM400
2375
MOV W134 D50
P1530
SM400
2379
MOV D50 W138
SM100
2383
ZPOP D200
ZPOP D50
ZPOP D300
2390
END

ประวัติผู้วิจัย

ชื่อ-สกุล	นางสาวภาสินี พุทธศร
วัน เดือน ปีเกิด	30 พฤศจิกายน 2532
ประวัติการศึกษา	
ระดับประถมศึกษา	ประถมศึกษาตอนต้นและตอนปลาย พ.ศ. 2539
	โรงเรียนวัดประเสริฐราษฎร์รังสรรค์วิทยา
ระดับมัธยมศึกษา	มัธยมศึกษาตอนต้นและตอนปลาย พ.ศ. 2545
	โรงเรียนเบญจมราชูทิศ ราชบุรี
ระดับอุดมศึกษา	คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมคอมพิวเตอร์ พ.ศ. 2550
	สถาบันเทคโนโลยีไทย – ญี่ปุ่น
ทุนการศึกษา	-
ประวัติการฝึกอบรม	-ไม่มี -
ผลงานที่ได้รับการตีพิมพ์	-ไม่มี -