คำนำ

คู่มือเล่มนี้ใช้ประกอบการฝึกอบรมหลักสูตรการใช้งาน PLC ระดับที่ 1 ซึ่งตามหลักสูตรนั้นจะมุ่งเน้นไปที่ทักษะการใช้งาน PLC ขั้นพื้นฐานเช่น แนะนำโครงสร้างของ PLC, การกำหนดแอสเดรสอินพุต/เอาต์พุต, การเขียนโปรแกรมให้ PLC ด้วยคำสั่งพื้นฐานที่ใช้งานบ่อย รวมทั้งการประยุกต์ใช้งานควบคุมเซอร์โวมอเตอร์เบื้องต้น

ดังนั้นคู่มือเล่มนี้จึงเหมาะสำหรับผู้เริ่มต้นศึกษาการทำงานของ PLC ได้เป็นอย่างดี

ทางบริษัทฯ ยินดีรับฟังความคิดเห็นและข้อเสนอแนะ เพื่อจะได้นำมาปรับปรุงให้ดียิ่งขึ้น

บริษัท ออมรอน อีเลคทรอนิคส์ จำกัด

พิมพ์ครั้งที่ 1 ตุลาคม 2550 จำนวนพิมพ์ 2,000 เล่ม

บริษัท ออมรอน อีเลคทรอนิคส์ จำกัด โทรศัพท์ 0-2942-6700 โทรสาร 0-2937-0501
เว็บไซต์ www.omron-ap.co.th

สารบัญ

APPENDIX

บทที่ 1
แนะนำให้รู้จัก PLC

PLC (Programmable Logic Controller) หรือปัจจุบันใช้คำว่า **PC (Programmable Controller)** ในที่นี้จะใช้คำว่า PLC แทน PC เพื่อป้องกันความสับสนกับคำว่า PC (Personal Computer)

PLC เป็นอุปกรณ์ที่คิดค้นขึ้นมา เพื่อใช้ควบคุมการทำงานของเครื่องจักรหรือระบบต่างๆ แทนวงจรรีเลย์แบบเก่า ซึ่งวงจรรีเลย์มีข้อเสียคือ การเดินสายและการเปลี่ยนแปลงเงื่อนไขในการควบคุมมีความยุ่งยาก และเมื่อใช้งานไปนานๆ หน้าสัมผัสของรีเลย์จะเสื่อม ทำให้ขาดเสถียรภาพในการควบคุม ดังนั้นปัจจุบัน PLC จึงเข้ามาทดแทนวงจรรีเลย์ เพราะ PLC ใช้งานได้ง่ายกว่า สามารถต่อเข้ากับอุปกรณ์อินพุต/เอาต์พุตได้โดยตรง นอกจากนั้นเพียงแค่เขียนโปรแกรมควบคุมก็สามารถใช้งานได้ทันที ถ้าต้องการจะเปลี่ยนเงื่อนไขใหม่สามารถทำได้โดยการเปลี่ยนแปลงโปรแกรมเท่านั้น นอกจากนี้ PLC ยังสามารถใช้งานร่วมกับอุปกรณ์อื่นๆ เช่น เครื่องอ่านบาร์โค้ด, เครื่องพิมพ์ (Printer) และระบบ RFID เป็นต้น

ในปัจจุบันนอกจาก PLC จะใช้งานแบบเดี่ยว (Stand alone) แล้ว ยังสามารถต่อ PLC หลายๆ ตัวเข้าด้วยกันเป็นเครือข่าย (Network) เพื่อควบคุมการทำงานของระบบให้มีประสิทธิภาพมากยิ่งขึ้นอีกด้วย จะเห็นได้ว่าการใช้งาน PLC มีความยืดหยุ่นมากกว่าการใช้งานวงจรรีเลย์แบบเก่า ดังนั้นในปัจจุบันโรงงานอุตสาหกรรมต่างๆ จึงใช้ PLC เป็นหัวใจหลักในการควบคุมการทำงานของเครื่องจักร

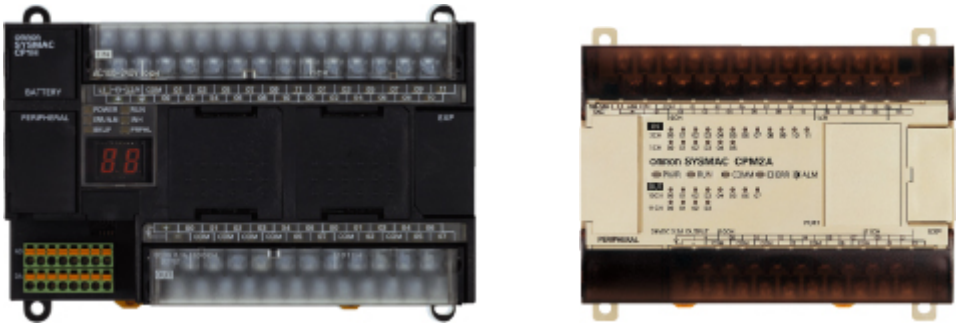
เราสามารถจำแนกประเภทของ PLC ตามลักษณะภายนอกได้เป็น 2 ชนิด คือ

1. 1 ชนิดของ PLC

เราสามารถจำแนก PLC ตามโครงสร้างหรือลักษณะภายนอกได้เป็น 2 ชนิด คือ

1.1.1 PLC ชนิดบล็อก (Block Type PLCs)

PLC ประเภทนี้ จะรวมส่วนประกอบทั้งหมดของ PLC อยู่ในบล็อกเดียวกัน ไม่ว่าจะเป็นตัวประมวลผล หน่วยความจำ ภาคอินพุต/เอาต์พุต และแหล่งจ่ายไฟ สามารถแสดงตัวอย่าง PLC แบบ Block Type ให้เห็นดังรูปที่ 1.1

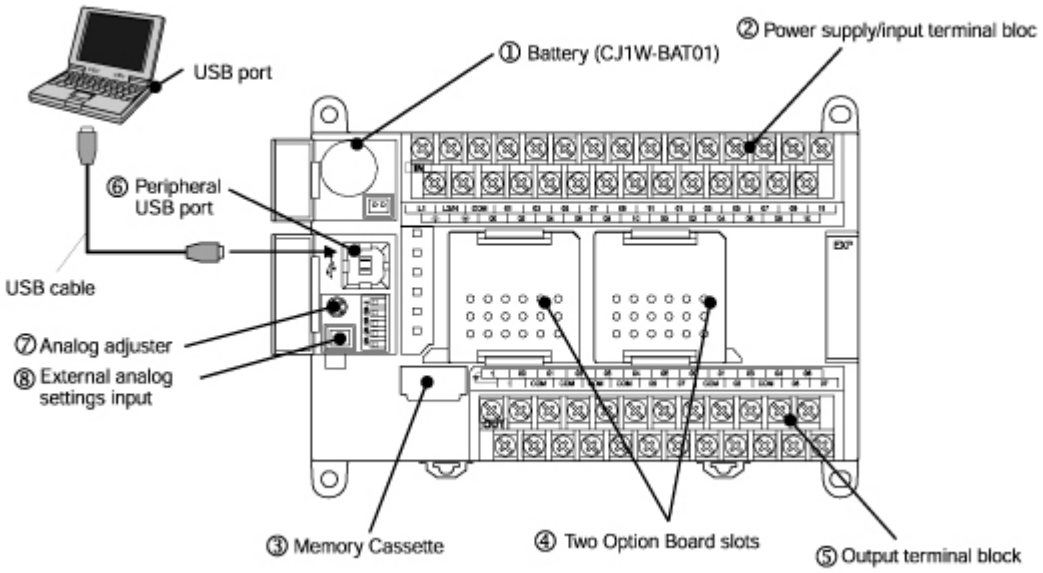


CP1L/H CPM2A

รูปที่ 1.1 *แสดงรูปร่าง PLC ชนิด Block Type*

- **ส่วนประกอบของ PLC แบบ Block Type**

ในที่นี้จะยกตัวอย่าง PLC แบบ Block Type ของ OMRON รุ่น CP1L และ CP1H

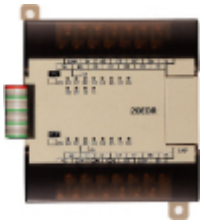


รูปที่ 1.2 *โครงสร้างภายนอก ของ PLC*

จากรูปที่ 1.2 สามารถอธิบายความหมายของแต่ละส่วนได้ดังนี้

① คือ แบตเตอรี่ (Battery)
② คือ ขั้วต่อแหล่งไฟและอินพุต (Power Supply/Input Terminal)
③ คือ ช่องเสียบหน่วยความจำ (Memory Cassette)
④ คือ ช่องเสียบเพื่อเพิ่มพอร์ตติดต่อสื่อสาร (Option Board slots)
⑤ คือ ขั้วต่อเอาต์พุต (Output Terminal)
⑥ คือ พอร์ตเชื่อมต่อกับอุปกรณ์ป้อนโปรแกรม (USB Port)
⑦ คือ ปุ่มปรับอนาลอก (Analog Adjuster)
⑧ คือ ขั้วต่ออินพุตสำหรับอนาลอก setting (External analog setting input)
⑨ คือ พอร์ตขยายอินพุต/เอาต์พุต (Expansion I/O Unit Connector)

ในกรณีที่ท่านต้องการเพิ่มจำนวนอินพุต/เอาต์พุต สามารถใช้หน่วยขยายอินพุต/เอาต์พุต (Expansion I/O Units) เพื่อเพิ่มจำนวนอินพุต/เอาต์พุตได้โดยการต่อเข้าที่พอร์ตขยายอินพุต เอาต์พุต (Expansion I/O Unit Connector) สามารถแสดงโครงสร้างของหน่วยขยายอินพุต/เอาต์พุต ได้ดังรูปที่ 1.3



***รูปที่ 1.3** แสดงหน่วยขยายอินพุต/เอาต์พุต (Expansion I/O Units)*

- **ข้อดีและข้อเสีย ของ PLC แบบ Block Type**

  สามารถยกตัวอย่างข้อดีข้อเสียของ PLC แบบ Block Type ดังนี้

ข้อดี	ข้อเสีย
1. มีขนาดเล็กสามารถติดตั้งได้ง่ายจึงเหมาะกับงานควบคุมขนาดเล็กๆ 2. สามารถใช้งานแทนวงจรรีเลย์ได้ 3. มีฟังก์ชันพิเศษ เช่นฟังก์ชันทางคณิตศาสตร์และฟังก์ชันอื่นๆ 4. มีราคาถูกกว่าแบบแร็คหรือโมดูลในจำนวนอินพุต/เอาต์พุตที่เท่ากัน	1. การเพิ่มจำนวนอินพุต/เอาต์พุตสามารถเพิ่มได้น้อยกว่า PLC ชนิดโมดูล 2. เมื่ออินพุต/เอาต์พุตเสียจุดใดจุดหนึ่งต้องนำ PLC ออกไปทั้งชุดทำให้ระบบต้องหยุดทำงานชั่วระยะเวลาหนึ่ง 3. มีฟังก์ชันให้เลือกใช้งานน้อยกว่า PLC ชนิดโมดูล

เนื้อหาในหัวข้อต่อไปจะกล่าวถึง PLC อีกชนิดหนึ่งซึ่งแยกส่วนประกอบต่างๆ ออกจากกัน เรียกว่า PLC ชนิดโมดูล (Modular Type PLCs)

1.1.2 PLC ชนิดโมดูล (Modular Type PLCs) หรือแร็ค (Rack Type PLCs)

PLC ชนิดนี้ส่วนประกอบแต่ละส่วนสามารถแยกออกจากกันเป็นมดูล (Modules) เช่นภาคอินพุต/เอาต์พุต จะอยู่ในส่วนของโมดูลอินพุต/เอาต์พุต (Input/Output Units) ซึ่งสามารถเลือกใช้งานได้ว่าจะใช้โมดูลขนาดกี่อินพุต/เอาต์พุต ซึ่งมีให้เลือกใช้งานหลายรูปแบบอาจจะใช้เป็นอินพุตอย่างเดียวขนาด 8 /16 จุด หรือเป็นเอาต์พุตอย่างเดียวขนาด 4/8/12/16 จุด ขึ้นอยู่กับรุ่นของ PLC ด้วย

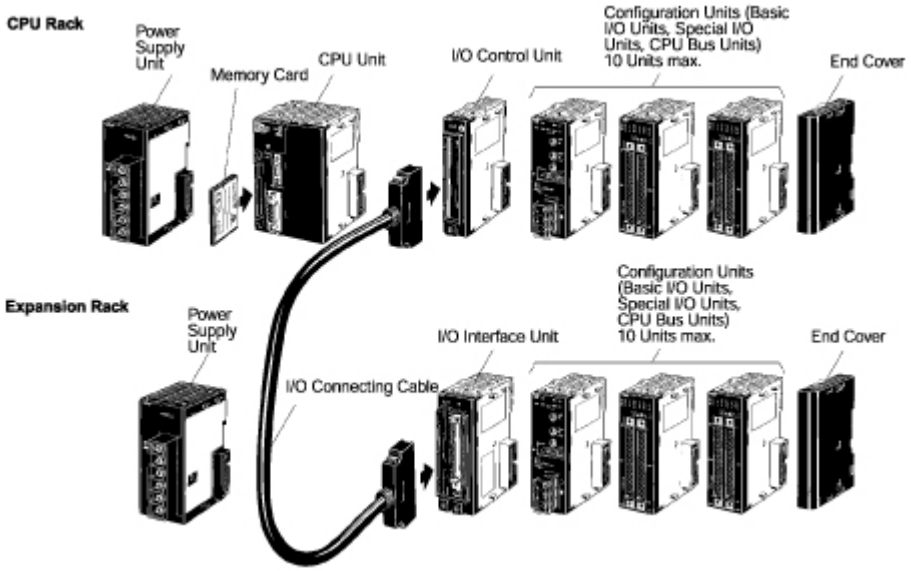
ในส่วนของตัวประมวลผลและหน่วยความจำจะรวมอยู่ในซีพียูโมดูล (CPU Unit) เราสามารถเปลี่ยนขนาดของ CPU Unit ให้เหมาะสมตามความต้องการใช้งาน เช่น PLC รุ่น CS1 จะมี CPU ให้เลือกใช้งานหลายรุ่นเช่น รุ่น CS1G-CPU42H จะมีความแตกต่างกับ PLC รุ่น CS1H-CPU65H (ทั้งสองรุ่นเป็น PLC ตระกูล CS1 เหมือนกัน) ตรงขนาดความจุของโปรแกรมและการรองรับจำนวนอินพุต/เอาต์พุต เป็นต้น

ส่วนประกอบต่างๆ ของ PLC ชนิดโมดูลที่กล่าวมาทั้งหมดนั้น เมื่อต้องการใช้งานจะถูกนำมาต่อร่วมกัน บางรุ่นใช้เป็นคอนเนคเตอร์ในการเชื่อมต่อกันระหว่างยูนิต เช่น รุ่น CQM1 / CQM1H หรือ CJ1M/H/G แต่บางรุ่นใช้ Backplane ในการรวมยูนิตต่างๆ เข้าด้วยกัน เพื่อให้สามารถใช้งานร่วมกันได้สามารถยกตัวอย่าง PLC ชนิดโมดูลได้ดังแสดงรูปที่ 1.4

CJ1 CS1

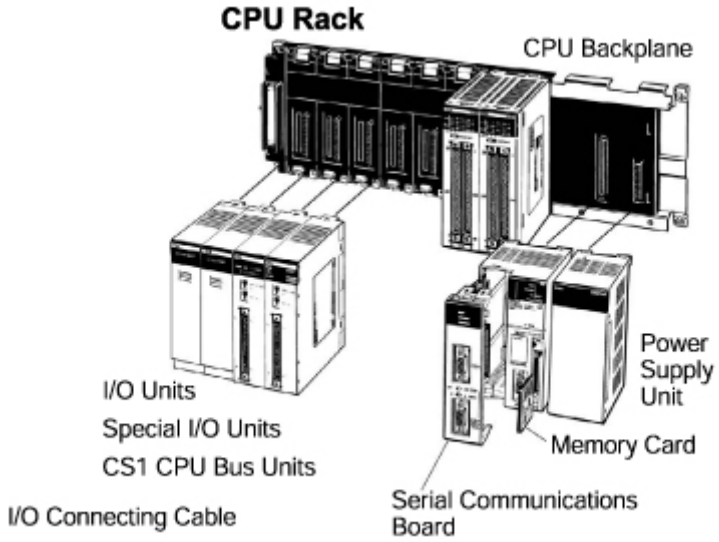
***รูปที่ 1.4** แสดงรูปร่างของ PLC ชนิดโมดูล*

ยกตัวอย่าง PLC รุ่น CJ1 จะใช้คอนเนคเตอร์ในการเชื่อมต่อแต่ละโมดูลเข้าด้วยกัน เพื่อให้สามารถทำงานร่วมกันได้ ดังแสดงในรูปที่ 1.5



***รูปที่ 1.5** แสดงชนิดของ PLC ชนิดโมดูล ที่ใช้คอนเนคเตอร์ในการเชื่อมต่อ*

ส่วน PLC รุ่น CS1 จะใช้ Backplane ในการเชื่อมต่อแต่ละโมดูลเข้าด้วยกันเพื่อให้ทำงานร่วมกัน ดังแสดงในรูปที่ 1.6



***รูปที่ 1.6** แสดงชนิดของ PLC ชนิดโมดูล ที่ใช้ Backplane ในการเชื่อมต่อ*

- **ข้อดีและข้อเสียของ PLC ชนิดโมดูล**

ข้อดี	ข้อเสีย
1. เพิ่มขยายระบบได้ง่ายเพียงแค่ติดตั้งโมดูลต่างๆ ที่ต้องการใช้งานลงไปบน Back plane 2. สามารถขยายจำนวนอินพุต/เอาต์พุตได้มากกว่าแบบ Block Type 3. อุปกรณ์อินพุต/เอาต์พุตเสียจุดใดจุดหนึ่ง สามารถถอดเฉพาะโมดูลนั้นไปซ่อม ทำให้ระบบสามารถทำการต่อได้ 4. มียูนิต และรูปแบบการติดต่อสื่อสารให้เลือกใช้งานมากกว่าแบบ Block Type	1. ราคาแพงเมื่อเทียบกับ PLC แบบ Block Type ที่มีจำนวนI/O เท่ากัน

จะเห็นว่า PLC แต่ละชนิดจะมีคุณสมบัติแตกต่างกัน PLC รุ่นที่ใหญ่ขึ้น จะมีคุณสมบัติและฟังก์ชันพิเศษอื่นๆ มากกว่า PLC รุ่นเล็กซึ่งสามารถเปรียบเทียบให้เห็นความแตกต่างดังตารางต่อไปนี้

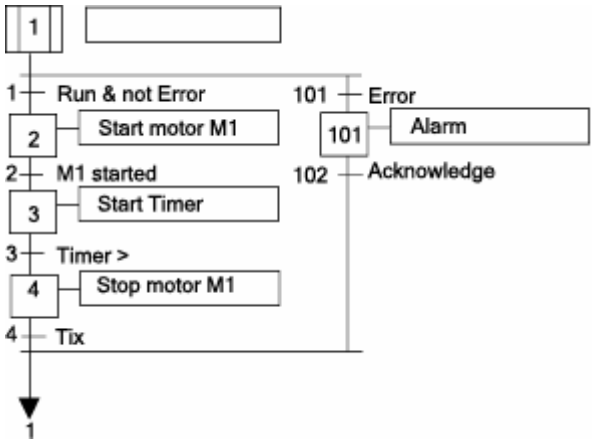
ตารางที่ 1.1 เปรียบเทียบคุณสมบัติของ PLC

คุณสมบัติ	รุ่น			
	CPM1A	**CP1L**	**CP1H**	**CS1**
จำนวนอินพุต/เอาต์พุต (Max.)	100 จุด	160 จุด	256 จุด	5,120 จุด
ความจุโปรแกรม(Max.)	2 KWords	10 KSteps	20 KSteps	250 KSteps
ความเร็วในการประมวลผล	0.72 μS	0.55 μS	0.1μS	0.04 μS
ไทม์เมอร์/เคาน์เตอร์	128	4,096/4,096	4,096/4,096	4,096/4,096
หน่วยความจำในส่วนของ DM	1,024 Words	32K Words	32KWords	32KWords
ระบบสื่อสาร	● CompoBus/S ● Host Link ● NT Link ● 1:1 Link	● CompoBus/S ● Host Link ● NT Link ● 1:1 Link ● Modbus-RTU ● Componet	● Controller Link ● CompoBus/D ● Ethernet ● Protocol Macro ● รวมทั้งระบบสื่อสารที่มีใน PLC รุ่นต่ำกว่า	● Ethernet ● Sysmac Link ● Profibus-DP ● Modbus ● รวมทั้งระบบสื่อสารที่มีในรุ่นต่ำกว่า

1.2 ภาษาที่ใช้ในการเขียนโปรแกรมให้กับ PLC

PLC แต่ละยี่ห้อจะใช้ภาษาในการเขียนโปรแกรมเพื่อสั่งให้ PLC ทำงานตามความต้องการแตกต่างกัน ซึ่งตามมาตรฐาน IEC1131-3 ได้แบ่งมาตรฐานภาษาต่างๆ ออกเป็น 5 แบบตามรูปที่แสดงข้างล่างนี้ ภาษาที่นิยมใช้มากที่สุดคือ Ladder Diagram เพราะเป็นภาษาที่ง่ายมีลักษณะคล้ายวงจรควบคุมแบบรีเลย์ ส่วนภาษาที่นิยมเป็นอันดับสองคือ Function Block

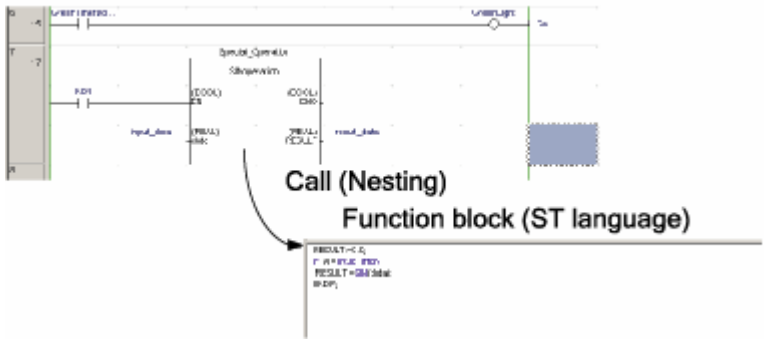
1) Sequential Flow Chart Language



2) Structure Text Language

```
D: = B*B - 4*A*C;
IF D < 0.0 THEN Nroots := 0;
ELSIF D  = 0.0 THEN
          Nroots:= 1;
          X1 := -B/(2.0*A);
ELSE Nroots := 2;
          X1 := (-B+sqrt(D)) / (2.0*A) ;
          X2 := (-B-sqrt(D)) / (2.0*A) ;
END_IF
```

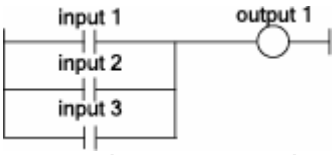
3) Function Block Diagram Language



4) Instruction List Language

```
Label:  LD      a1      (*result := a1*)
        ADD(    a2      (*delayed ADD, result := a2*)
        MUL(    a3      (*delayed MUL, result := a3*)
        SUB     a4      (*result := a3 - a4*)
        )               (*execute delayed MUL,*)
                        (*result := a1 + (a2*(a3 - a4) *a5)*)
        ADD     a6      (*a1 + (a2*(a3 - a4)*a5) + a6*)
        ST      res     (*store current result in res*)
```

5) Ladder Diagram



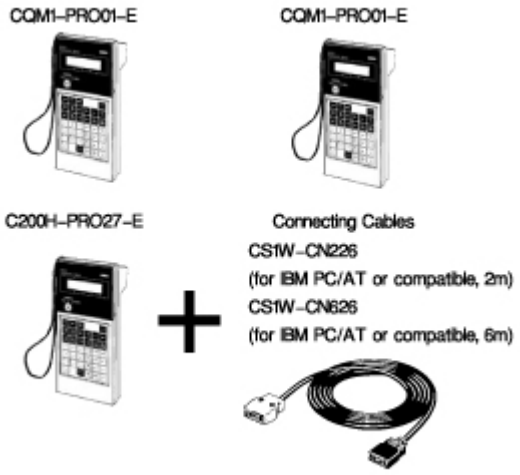
หลังจากที่ได้เรียนรู้ภาษาที่ใช้ในการเขียนโปรแกรมให้กับ PLC แล้ว ในหัวข้อต่อไปจะกล่าวถึงอุปกรณ์ที่ใช้ในการป้อนโปรแกรมให้กับ PLC ซึ่งจะกล่าวถึงรายละเอียดดังนี้

1.3 อุปกรณ์สำหรับการโปรแกรม

การสั่งให้ PLC ทำงาน จะต้องป้อนโปรแกรมให้กับ PLC ก่อน ซึ่งอุปกรณ์ที่ใช้ในการป้อนโปรแกรมให้กับ PLC นั้น สามารถแบ่งได้เป็น 2 ประเภท

1.3.1 ตัวป้อนโปรแกรมแบบมือถือ (Hand Held Programmer)

แต่ละยี่ห้อจะมีชื่อเรียกแตกต่างกัน เช่น OMRON จะเรียกว่า Programming Console เป็นต้น สามารถยกตัวอย่างให้เห็นดังรูปที่ 1.7 ในปัจจุบัน PLC รุ่นใหม่ๆ ของออมรอนไม่ได้ใช้ Programming Console ในการเขียนโปรแกรมแล้วเพราะใช้งานยาก



รูปที่ 1.7 แสดงตัวป้อนโปรแกรมแบบมือถือ (Programming Console)

การเขียนโปรแกรมให้กับ PLC โดยการใช้ Programming Console จะป้อนเป็นภาษา Statement List หรือ Mnemonic เช่น คำสั่ง LD, AND, OR ซึ่งเป็นคำสั่งพื้นฐาน สามารถเรียกใช้งานโดยการกดปุ่มที่อยู่บนตัว Programming Console นั้น แต่เมื่อต้องการใช้งานฟังก์ชันอื่นๆ ที่มีอยู่ใน PLC สามารถเรียกใช้งานได้โดยการกดปุ่มเรียกใช้คำสั่งพิเศษ

การใช้ Programming Console มีข้อดีตรงที่มีความสะดวกในการเคลื่อนย้าย และสามารถพกพาได้สะดวกเนื่องจากมีขนาดเล็ก แต่ก็มีข้อเสียคือในการใช้งานผู้ใช้ต้องศึกษาวิธีการใช้งานของอุปกรณ์เหล่านี้ว่ามีวิธีการกดอย่างไร ถึงจะสั่งงาน PLC ได้

ในหัวข้อต่อไปจะกล่าวถึงอุปกรณ์ที่ใช้ในการป้อนโปรแกรมให้กับ PLC อีกชนิดหนึ่งคือ คอมพิวเตอร์ส่วนบุคคล (Personal Computer)

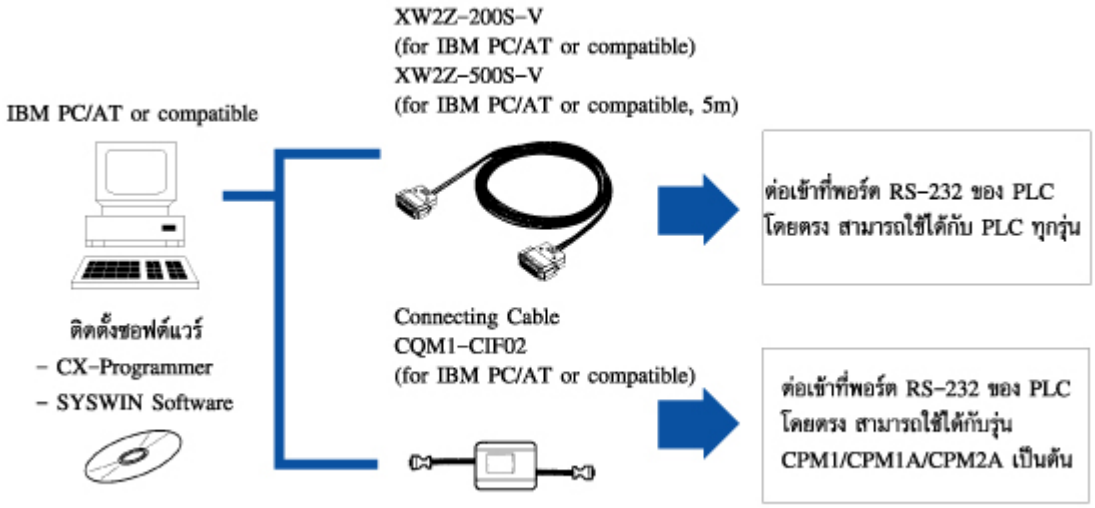
1.3.2 คอมพิวเตอร์

สามารถใช้ในการเขียนโปรแกรมให้กับ PLC ได้ โดยใช้งานร่วมกับซอฟต์แวร์ (Software) เฉพาะของ PLC ยี่ห้อนั้น เช่น PLC ของ OMRON จะใช้ซอฟต์แวร์ที่มีชื่อเรียกแตกต่างกันไป สามารถยกตัวอย่างได้เช่น

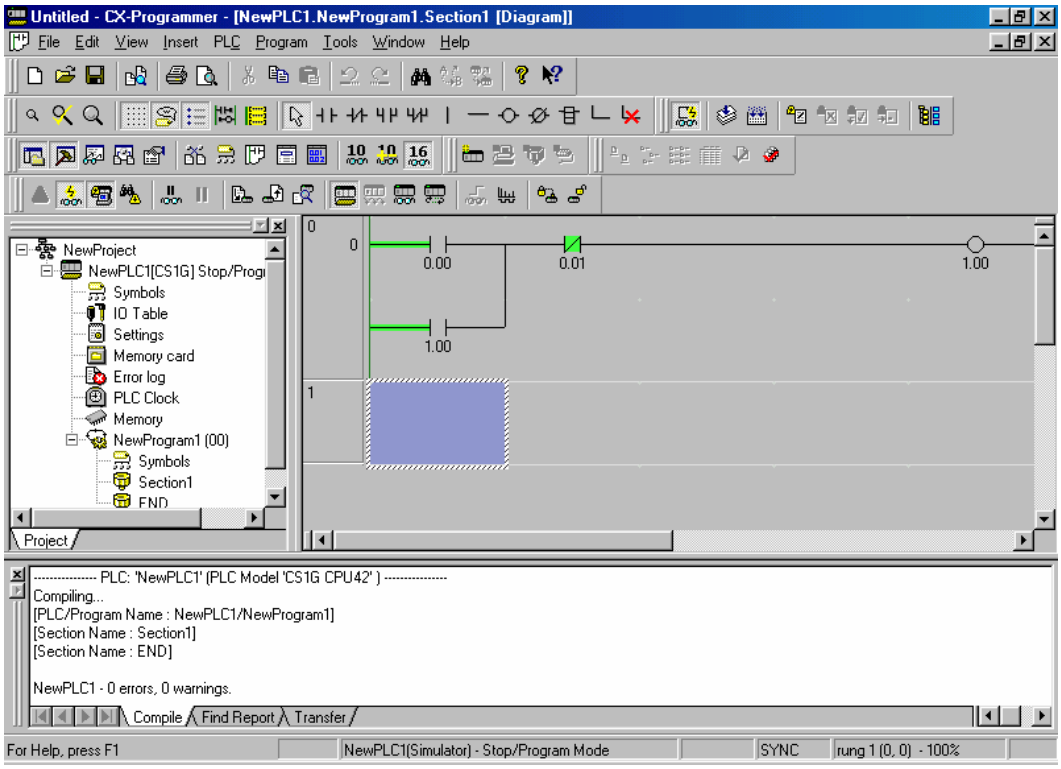
- Syswin Support Software
- CX-Programmer

ใช้ได้กับระบบปฏิบัติการตั้งแต่ Window XP ขึ้นไป หรือ Window NT ซึ่งซอฟต์แวร์ต่างๆ เหล่านี้ได้ถูกพัฒนาขึ้นเพื่อใช้กับ PLC รุ่นใหม่ที่ผลิตขึ้นมาอย่างเช่น CX-Programmer มีการพัฒนาเป็นเวอร์ชันที่สูงขึ้นเรื่อยๆ เพื่อรองรับกับ PLC รุ่นใหม่ๆ และฟังก์ชันใหม่ๆ ของ PLC

วิธีการต่อคอมพิวเตอร์กับ PLC สามารถแสดงให้เห็นดังนี้



***รูปที่ 1.8** แสดงวิธีการต่อใช้งานคอมพิวเตอร์กับ PLC*



***รูปที่ 1.9** ตัวอย่างซอฟต์แวร์ (CX-Programmer)*

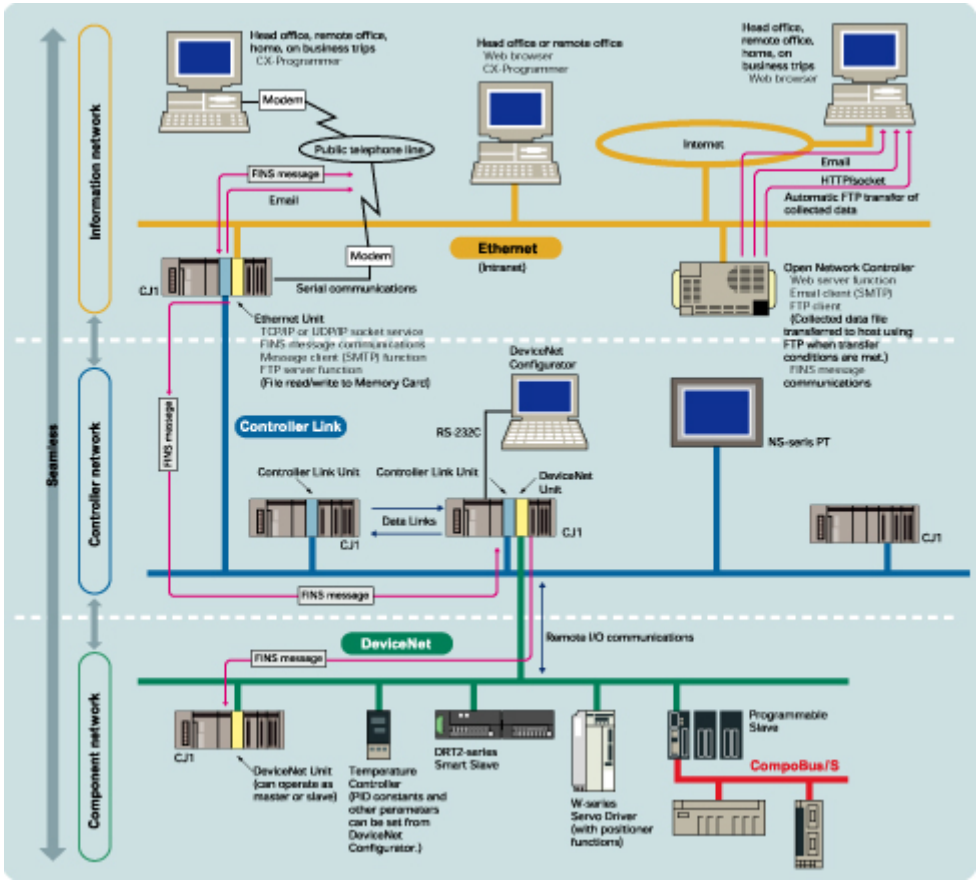
ข้อดีของการใช้เครื่องคอมพิวเตอร์ในการป้อนโปรแกรมให้กับ PLC คือ ใช้งานง่าย เช่นในกรณีใช้ CX-Programmer ร่วมกับระบบปฏิบัติการ Window จากรูปที่ 1.9 ท่านจะเห็นว่าการเขียนโปรแกรมเป็นภาษา Ladder Diagram จะเป็นการนำสัญลักษณ์ต่างๆ เข้ามาใช้แทนการเขียนคำสั่ง ทำให้เข้าใจง่ายเพียงแค่คลิกเลือกสัญลักษณ์ต่างๆ จากส่วนของ Toolbar นอกจากนั้นยังมี Toolbar อื่นๆ ให้เลือกใช้งานซึ่งง่ายกว่าการใช้ Programming Console

ดังนั้นสามารถสรุปได้ว่าการป้อนโปรแกรมให้กับ PLC สามารถทำได้ 2 วิธีคือ การใช้ Programming Console และการใช้เครื่องคอมพิวเตอร์ ซึ่งขึ้นอยู่กับความสะดวกของผู้ใช้ ในหัวข้อต่อไปจะกล่าวถึงระบบการติดต่อสื่อสารของ PLC

1.4 ระบบสื่อสาร (Communications)

ระบบสื่อสารของ PLC คือการนำ PLC ไปต่อใช้งานร่วมกับอุปกรณ์อื่นๆ เพื่อให้อุปกรณ์อื่นควบคุมการทำงานของ PLC หรือ ให้ PLC ไปควบคุมการทำงานของอุปกรณ์อื่น หรือ เป็นระบบที่ใช้ในการแลกเปลี่ยนข้อมูลระหว่าง PLC กับ PLC ก็ได้ ซึ่งปัจจุบัน PLC สามารถนำไปต่อร่วมกับอุปกรณ์ของยี่ห้อเดียวกัน หรืออุปกรณ์ภายนอกต่างยี่ห้อกัน เพื่อควบคุมการทำงานของระบบให้ใช้งานได้อย่างกว้างขวางมากขึ้น สำหรับระบบสื่อสารของแต่ละยี่ห้อจะมีชื่อเรียกไม่เหมือนกัน

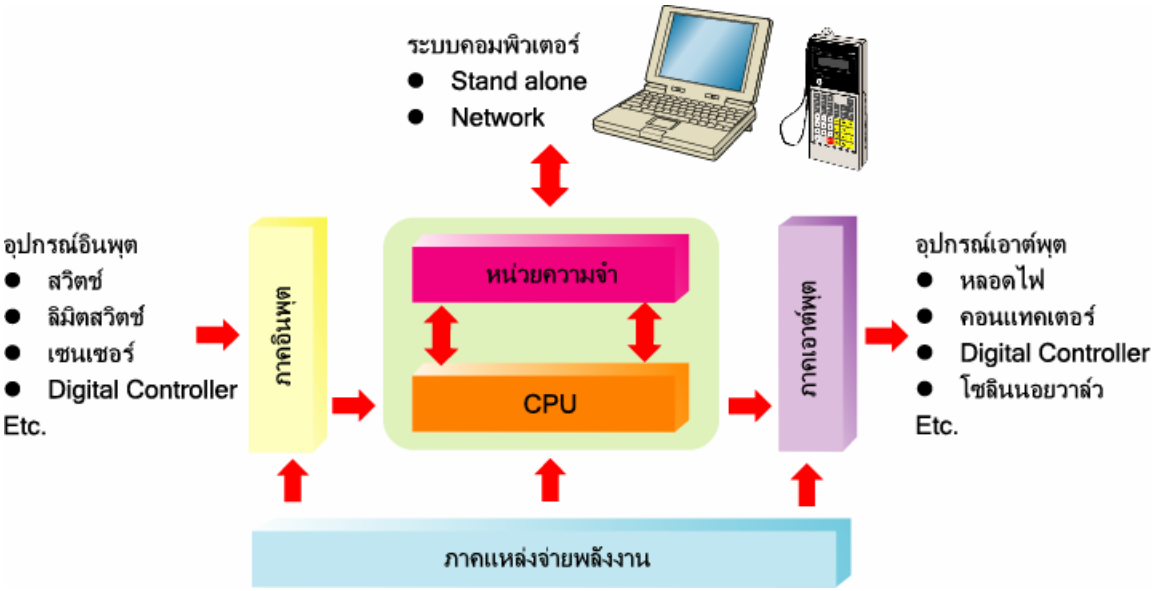
นอกจากนี้ PLC แต่ละรุ่นยังมีระบบการติดต่อสื่อสารบางรูปแบบแตกต่างกันด้วย เช่น PLC รุ่นเล็กจะมีความสามารถในการติดต่อสื่อสารได้น้อยกว่า PLC รุ่นใหญ่ เช่น PLC รุ่น CP1 สามารถใช้ระบบสื่อสารได้เฉพาะ Compobus/S, Host link, 1:1 link, NT link ส่วนรุ่นที่สูงขึ้นมาเช่น CJ1 หรือ CS1 นอกจากจะใช้ระบบที่เป็นระบบการติดต่อสื่อสารพื้นฐานที่มีใน PLC รุ่นเล็กแล้ว ยังสามารถติดต่อสื่อสารในลักษณะของ Ethernet ได้อีกด้วย สามารถแสดงตัวอย่างรูปแบบการติดต่อสื่อสารของ PLC ได้ดังรูปที่ 1.10



รูปที่ 1.10 *ระบบการติดต่อสื่อสารของ PLC ในโรงงานอุตสาหกรรม (PLC Network)*

1.5 โครงสร้างของ PLC

โครงสร้างภายในของ PLC แต่ละส่วนจะประกอบกันทำงานเป็นระบบควบคุมที่เราเรียกว่า PLC ซึ่งประกอบไปด้วยส่วนสำคัญดังรูปที่ 1.11



รูปที่ 1.11 *ไดอะแกรมภายใน PLC*

จากไดอะแกรมดังรูปที่ 1.11 PLC จะมีส่วนประกอบสำคัญด้วยกันทั้งหมด 5 ส่วนดังนี้

1. ซีพียู (CPU; Central Processing Unit)
2. หน่วยความจำ (Memory Unit)
3. ภาคอินพุต (Input Unit)
4. ภาคเอาต์พุต (Output Unit)
5. ภาคแหล่งจ่ายพลังงาน (Power Supply Unit)

ยูนิตทั้ง 5 ส่วนเมื่อประกอบเข้าด้วยกันแล้วก็จะกลายเป็น PLC ชุดหนึ่งที่สามารถทำงานได้ แต่ละยูนิตจะมีหน้าที่และคุณสมบัติดังนี้

1.5.1 ซีพียู (CPU; Central Process Unit)

ซีพียูหรือหน่วยประมวลผลกลาง ทำหน้าที่ประมวลผลการทำงานตามคำสั่งของส่วนต่างๆ ตามที่ได้รับมา ผลจากการประมวลผลก็จะถูกส่งออกไปส่วนต่างๆ ตามที่ระบุไว้ด้วยคำสั่งนั่นเอง ซีพียูจะใช้เวลาในการประมวลช้าหรือเร็ว ขึ้นอยู่กับการเลือกขนาดของซีพียู และความยาวของโปรแกรมที่เขียนด้วย

ปกติแล้วซีพียูจะใช้ไมโครโปรเซสเซอร์ขนาดตั้งแต่ 4 บิต, 8 บิต, 16 บิต, 32 บิต, 64 บิต หรือ 128 บิตมาทำงาน โดยที่ซีพียูแต่ละขนาดก็จะมีความสามารถจำกัดไม่เท่ากันจึงทำให้ PLC ในแต่ละรุ่นมีความสามารถต่างกันนั่นเอง หรือแม้กระทั่งว่าภายใน PLC บางรุ่นจะใช้ไมโครโปรเซสเซอร์ถึง 2 ตัวช่วยกันทำงาน เวลาการประมวลผลก็จะเร็วกว่า PLC ที่ใช้ไมโครโปรเซสเซอร์เพียงแค่ตัวเดียว

โดยปกติแล้วการเลือกใช้งาน PLC จะเลือกจากการประยุกต์ใช้งานจึงทำให้ผู้ใช้งาน (User) ไม่รู้ว่าผู้ผลิตใช้ไมโครโปรเซสเซอร์รุ่น หรือเบอร์อะไรในการสร้างเครื่อง PLC ดังนั้นเวลาพิจารณาเลือกใช้ PLC ซึ่งไม่มีการระบุเบอร์หรือรุ่นของไมโครโปรเซสเซอร์ผู้ใช้งานสามารถเลือกจากคุณสมบัติอื่น เช่น จำนวนอินพุต/เอาต์พุต, ความเร็วในการประมวลผลของคำสั่ง, ขนาดความจุโปรแกรม และข้อมูล เป็นต้น

1.5.2 หน่วยความจำ (Memory Unit)

หน่วยความจำเป็นอุปกรณ์ที่ใช้เก็บโปรแกรมและข้อมูลต่างๆ ของ PLC กรณีที่สั่งให้ PLC ทำงาน (RUN) มันจะนำเอาโปรแกรมและข้อมูลในหน่วยความจำมาประมวลผลการทำงาน สำหรับหน่วยความจำที่ใช้งานมีด้วยกัน 2 ชนิด คือ

- หน่วยความจำชั่วคราว (RAM: Random Access Memory)
- หน่วยความจำถาวร (ROM: Read Only Memory)

- ***หน่วยความจำชั่วคราว*** *(RAM: Random Access Memory)*

โปรแกรมและข้อมูลที่สร้างขึ้นโดยผู้ใช้จะถูกจัดเก็บในส่วนนี้ คุณสมบัติของ RAM เมื่อไม่มีไฟเลี้ยงจะทำให้โปรแกรมและข้อมูลหายไปทันที ดังนั้นภายใน PLC จะพบว่าจะมีแบตเตอรี่สำรองข้อมูล (Backup Battery) เอาไว้สำรองข้อมูล (Backup Data) กรณีที่ไฟหลัก (Main Power Supply) ไม่จ่ายไฟให้กับ PLC ข้อควรระวังคือ ไม่ควรที่จะถอดแบตเตอรี่สำรอง (Backup Battery) กรณีที่ไม่มีไฟจ่ายให้ PLC

- ***หน่วยความจำถาวร*** *(ROM: Read Only Memory)*

เป็นหน่วยความจำอีกชนิดหนึ่ง โดยที่ข้อมูลใน ROM ไม่จำเป็นต้องมีแบตเตอรี่สำรองข้อมูล แต่ก็มีปัญหาเรื่องเวลาในการเข้าถึงข้อมูล (Time Access) ช้ากว่า RAM จึงปรากฏให้ผู้ใช้เห็นว่า PLC จะมีหน่วยความจำใช้งานทั้ง RAM และ ROM ร่วมกันอยู่

ROM แบ่งออกเป็น 3 ชนิด ดังนี้

1) PROM (Programmable ROM)

2) EPROM (Erasable Programmable ROM)

3) EEPROM (Electrical Erasable Programmable ROM)

PROM จัดเป็น ROM รุ่นแรก เขียนข้อมูลลงชิปได้เพียงครั้งเดียว ถ้าเขียนข้อมูลไม่สมบูรณ์ชิปก็จะเสียทันที ไม่สามารถนำกลับมาเขียนใหม่ได้อีก จึงได้มีการพัฒนามาเป็นรุ่น EPROM ซึ่งสามารถเขียนข้อมูลลงชิปได้หลายครั้ง เพียงแค่นำชิปไปฉายแสงอุลตร้าไวโอเลตก็จะเป็นการลบข้อมูลในชิปด้วยสัญญาณทางไฟฟ้าได้เลย จึงทำให้เกิดความสะดวกสบายมากขึ้น แต่เรื่องเวลาในการเข้าถึงข้อมูลก็ยังช้ากว่า RAM อยู่

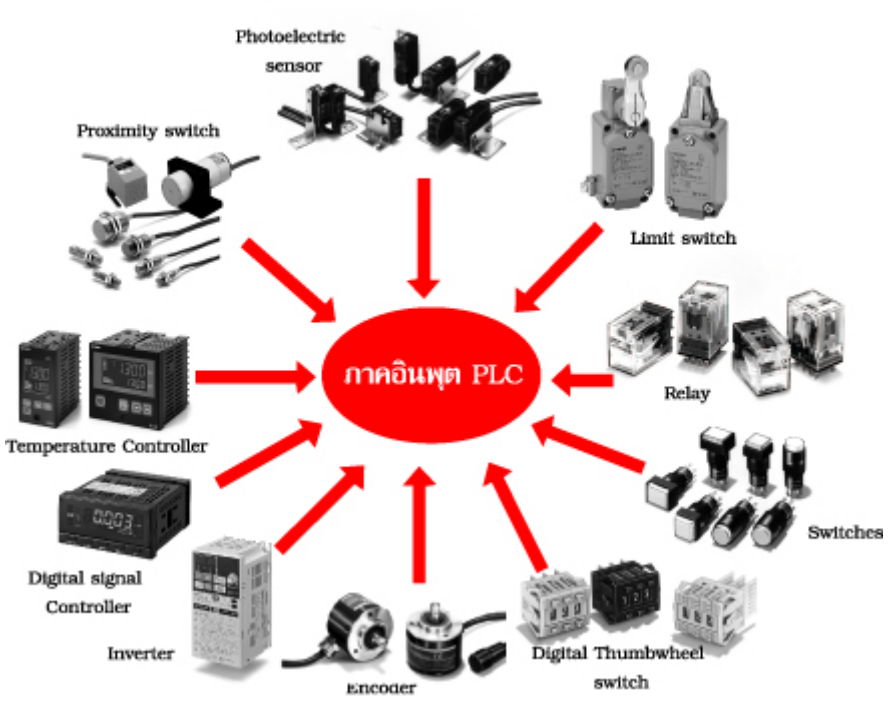
การใช้งานหน่วยความจำใน PLC

- RAM จะใช้เก็บโปรแกรมและข้อมูลที่ทำงานจากการสั่ง RUN/STOP PLC

- ROM จะใช้จัดเก็บซอฟต์แวร์ระบบ (System Software) และเป็นชุดสำรองโปรแกรมและข้อมูล (Backup Program and Data) เพื่อป้องกันกรณีที่โปรแกรมและข้อมูลใน RAM หายไป ผู้ใช้สามารถที่จะถ่ายโปรแกรมและข้อมูลเข้าไปที่ RAM ใหม่ได้

1.5.3 ภาคอินพุต (Input Unit)

ภาคอินพุตของ PLC ทำหน้าที่รับสัญญาณอินพุตเข้ามาแปลงสัญญาณ ส่งเข้าไปภายใน PLC อุปกรณ์อินพุต (Input Device) ต่างๆ ที่นำมาต่อกับภาคอินพุตได้นั้น สามารถแสดงตัวอย่างได้ตามรูปที่แสดงดังนี้



รูปที่ 1.12 *แสดงอุปกรณ์อินพุตต่างๆ*

อุปกรณ์ที่สามารถนำมาต่อกับภาคอินพุต PLC ได้จัดออกเป็นกลุ่มๆ ดังรูปที่ 1.12 โดยกลุ่มอุปกรณ์แต่ละกลุ่มจะมีวิธีต่อวงจรเข้าภาคอินพุต PLC แตกต่างกันออกไป เวลาใช้งานอุปกรณ์แต่ละกลุ่ม จำเป็นต้องศึกษาข้อมูลเพิ่มเติมของอุปกรณ์แต่ละชนิดก่อน เพื่อความเข้าใจขั้นตอนการทำงาน และสามารถต่อวงจรได้ถูกต้อง

อุปกรณ์ที่นำมาต่อกับภาคอินพุตของ PLC อุปกรณ์บางกลุ่มจะมีสัญญาณทั้งอินพุต/เอาต์พุต เช่น Inverter, Digital Signal, Controller, ตัวควบคุมอุณหภูมิ, เซนเซอร์รุ่นพิเศษ เป็นต้น จำเป็นต้องต่อใช้งานให้ถูกต้อง ซึ่งสามารถแนะนำได้ในขั้นต้นคือ ต่อวงจรภาคเอาต์พุตของอุปกรณ์นั้นๆ เข้ากับภาคอินพุต PLC

ภาคเอาต์พุตของอุปกรณ์จะมีเอาต์พุตให้เลือกใช้งานหลายแบบ ซึ่งภาคอินพุต PLC มีวงจรภาคอินพุตอยู่หลายแบบเช่นกัน เพื่อรองรับอุปกรณ์อินพุตในแต่ละแบบให้เหมาะสม

- **วงจรภาคอินพุต (Input Circuit PLC)**

  วงจรภาคอินพุตแบ่งออกเป็น 2 ประเภทใหญ่ๆ คือ

  1) ดิจิตอลอินพุต (Digital Input)
  2) อนาลอกอินพุต (Analog Input)

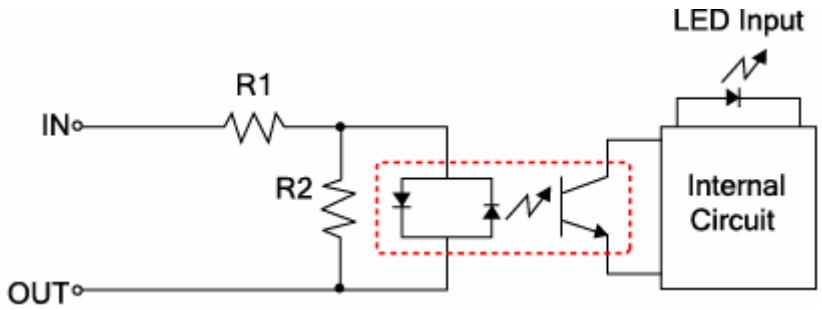
1) ดิจิตอลอินพุต (Digital Input Type)

ดิจิตอลอินพุต หมายถึง อินพุตที่รับรู้สัญญาณได้เพียงแค่ “ON” หรือ “OFF” เท่านั้น ตามโครงสร้างจะมีดิจิตอลอินพุต 2 แบบคือ

1.1) วงจรอินพุตไฟตรง (DC Input)

1.2) วงจรอินพุตไฟสลับ (AC Input)

1.1) ***วงจรอินพุตไฟตรง (DC Input)*** จะใช้อุปกรณ์ที่ทำงานด้วยแรงดันไฟฟ้ากระแสตรงตัวอย่างวงจรอินพุตไฟตรงแสดงดังรูปที่ 1.13



รูปที่ 1.13 *วงจรอินพุตแบบ DC*

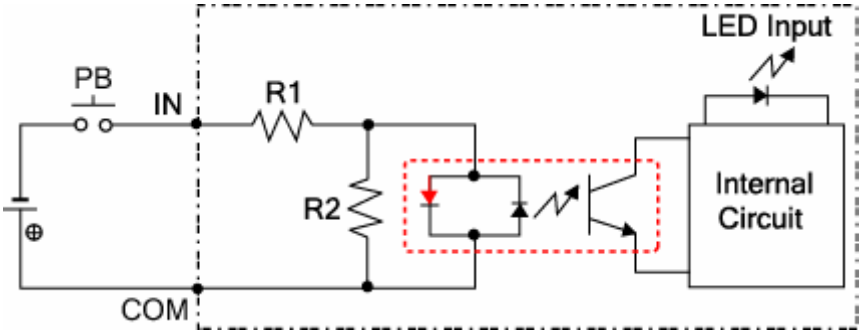
หมายเหตุ **:** ค่าความต้านทาน R1 และ R2 ดูได้จากคู่มือของ PLC รุ่นนั้นๆ

จากรูปที่ 1.13 ภาคอินพุตจะใช้วงจรลดทอนแรงดันแล้วขับออปโตทรานซิสเตอร์ จากออปโตทรานซิสเตอร์ก็จะไปขับภาคอินพุตของ IC เพื่อส่งสัญญาณไปให้ CPU อีกทีหนึ่ง ซึ่งการใช้อุปกรณ์ประเภทออปโต (Opto) ทำให้ระบบ PLC สามารถแยกสัญญาณกราวด์ (Ground) ของภาคอินพุตออกจากวงจรภายในได้ สำหรับวงจรภาคอินพุตดังรูปที่ 1.13 สามารถสรุปคุณสมบัติได้ดังตารางที่ 1.2

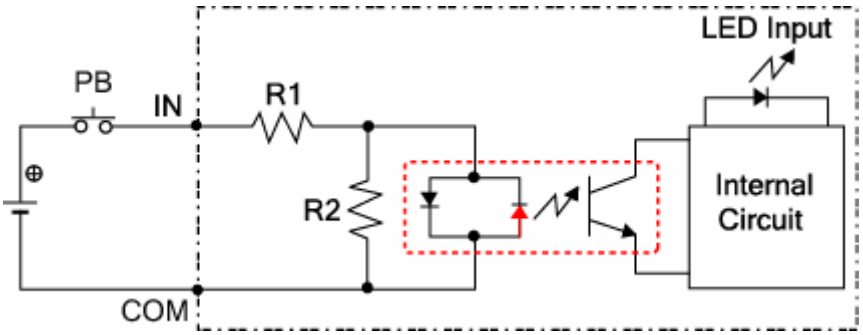
ตารางที่ 1.2 ตัวอย่างคุณสมบัติภาคอินพุต (DC)

	คุณสมบัติ
แรงดันอินพุต	24 VDC $^{+10\%}/_{+15\%}$ (26.4V-18V)
อินพุตอิมซีแดนซ์	2 kΩ
กระแสอินพุต	12 mA
แรงดันอินพุตขณะทำงาน	"ON" 14.4 VDC min. "OFF" 5.0 VDC max.
เวลาตอบสนองอินพุต	"ON Delay": 8 mS max. "OFF Delay": 8 mS max. สามารถปรับค่าได้ตั้งแต่ 1,2,4,8,16,32,64,128 mS โดยใช้โหมด PC Setup

สำหรับวงจรภาคอินพุตดังรูปที่ 1.13 จะพบว่า ภาคอินพุตของออปโตทรานซิส- เตอร์มีไดโอด (Diode) ต่อสลับขั้วกันอยู่ เพื่อเวลาใช้งานสามารถเลือกต่อวงจรได้ 2 แบบ ดังรูปที่ 1.14



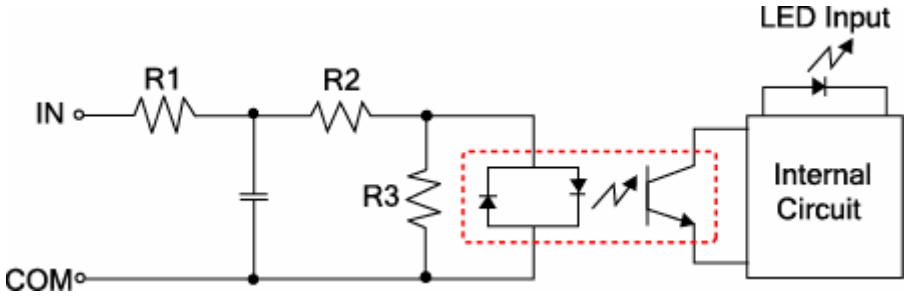
ก . การต่ออินพุตแบบ Source



ข. การต่ออินพุตแบบ Sink

รูปที่ 1.14 *การต่อวงจรอินพุตแบบ DC Source/Sink*

1.2) ***วงจรอินพุตไฟสลับ (AC Input)*** ใช้ไฟสลับผ่านแรงดันทำให้ไม่มีปัญหาเรื่องแรงดันตกคร่อมในสายมากเกินไปเหมือนวงจรอินพุตไฟตรงโดยที่ผ่านแรงดันอินพุตตั้งแต่ 100-220 VAC สำหรับ PLC บางรุ่นก็จะแบ่งอินพุตแบบนี้ออกเป็น 2 ย่านคือ 100-120 และ 200-240 VAC ลักษณะวงจรอินพุตแสดงดังรูปที่ 1.15



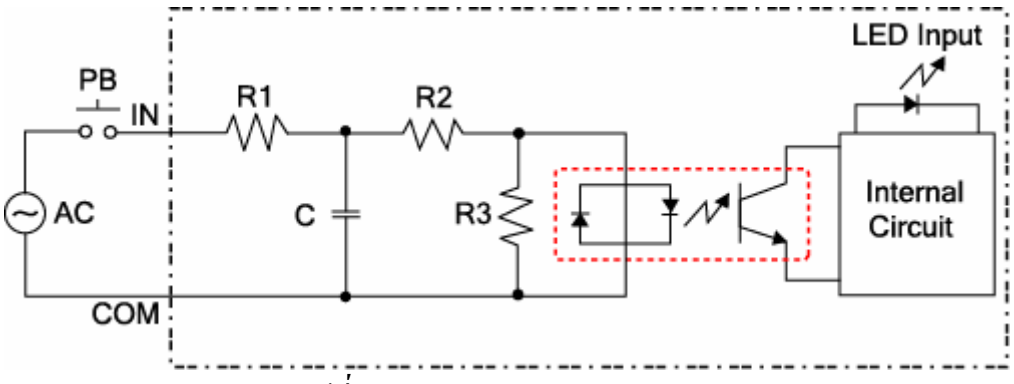
รูปที่ 1.15 *วงจรอินพุตแบบ AC*

คุณสมบัติของวงจรอินพุตไฟสลับทั้งแรงดันอินพุตระบบไฟ 110V หรือ 220V ดังแสดงตารางที่ 1.3

ตารางที่ 1.3 คุณสมบัติภาคอินพุต (AC)

	คุณสมบัติ	
แรงดันอินพุต	100-120 VAC$^{+10\%}/_{+15\%}$ 50/60Hz	200-240 VAC$^{+10\%}/_{+15\%}$ 50/60Hz
อินพุตอิมซีแดนซ์	2 kΩ (50Hz), 17 kΩ (60 Hz)	38 kΩ (50Hz), 32 kΩ (60 Hz)
กระแสอินพุต	5 mA (at 100 VAC)	6 mA (at 200 VAC)
แรงดันอินพุตขณะทำงาน	“ON” 60 VAC min. “OFF” 20 VAC max.	“ON” 150 VAC min. “OFF” 40 VAC max.
เวลาตอบสนองอินพุต	“ON Delay”: 35 mS max. “OFF Delay”: 55 mS max.	

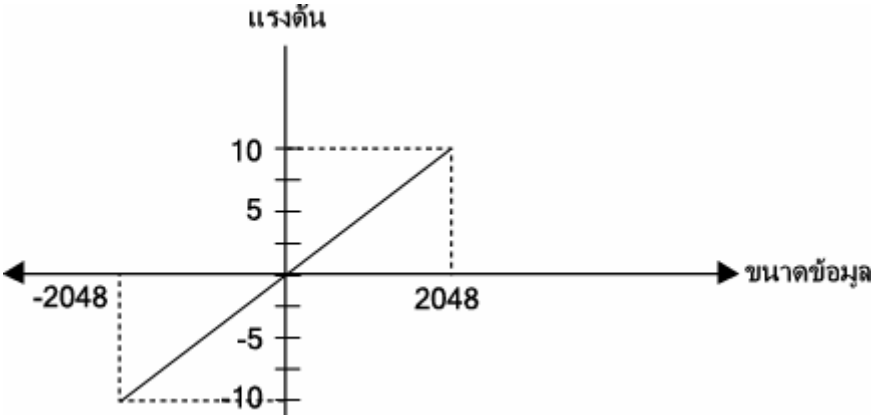
ลักษณะการต่อวงจรใช้งานสำหรับภาคอินพุตแบบ AC จะมีลักษณะการต่อดังรูปที่ 1.16



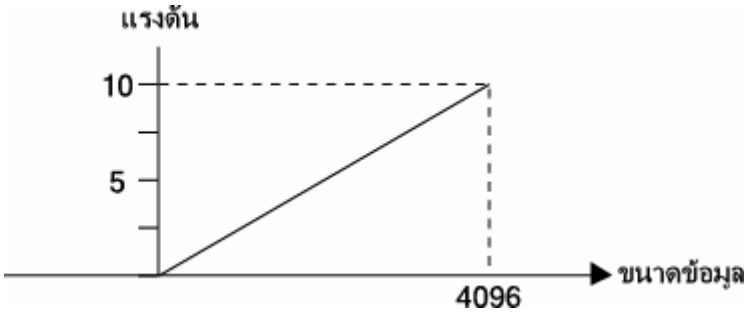
รูปที่ 1.16 *การต่อวงจรอินพุตแบบ AC*

2) อนาลอกอินพุต (Analog Input Type)

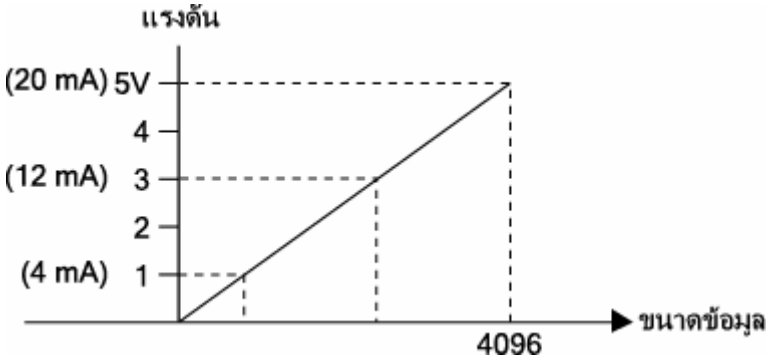
อนาลอกอินพุตจัดเป็นอินพุตที่สามารถรับสัญญาณที่บอกเป็นปริมาณที่เปลี่ยนแปลงค่าได้เช่น 0-10 VDC, ±10 VDC 1-5 V และ 4-20 mA ดังรูปที่ 1.17



ก. สัญญาณขนาด ±10 VDC



ข. สัญญาณขนาด 0-10 VDC



ค. สัญญาณขนาด 1-5 V (4-20 mA)

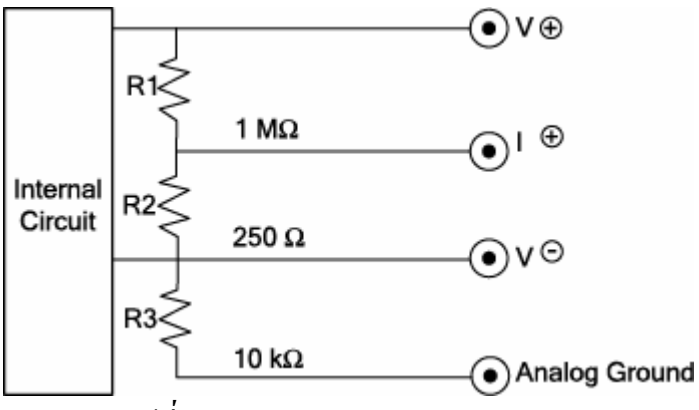
***รูปที่ 1.17** สัญญาณแบบต่างๆ ที่ส่งให้อนาลอกอินพุต*

สัญญาณอนาลอกทั้ง 3 แบบ จัดเป็นขนาดสัญญาณมาตรฐานที่กำหนดไว้ใช้ในอุตสาหกรรม ดังนั้นอุปกรณ์ที่มีภาคเอาต์พุตเป็นแบบอนาลอกเช่น อนาลอกเซนเซอร์, ภาคอนาลอกเอาต์พุตของ Digital Signal Controller, Temperature Controller เป็นต้น ก็จะมีขนาดของสัญญาณตามมาตรฐานเช่นกัน ซึ่งตัวอุปกรณ์อาจจะมีเอาต์พุตแบบใดแบบหนึ่งหรือทั้ง 3 แบบเลยก็ได้ ดังนั้นภาคอนาลอกอินพุตของ PLC ก็ต้องสามารถเลือกตรวจสอบได้ทั้ง 3แบบเช่นกัน

หลักการทำงานของอนาลอกอินพุตของ PLC นำค่าที่วัดได้แปลงเป็นสัญญาณดิจิตอล สามารถแสดงได้ดังไดอะแกรมรูปที่ 1.18

รูปที่ 1.18 *ไดอะแกรมการส่งข้อมูลอนาลอกให้ PLC*

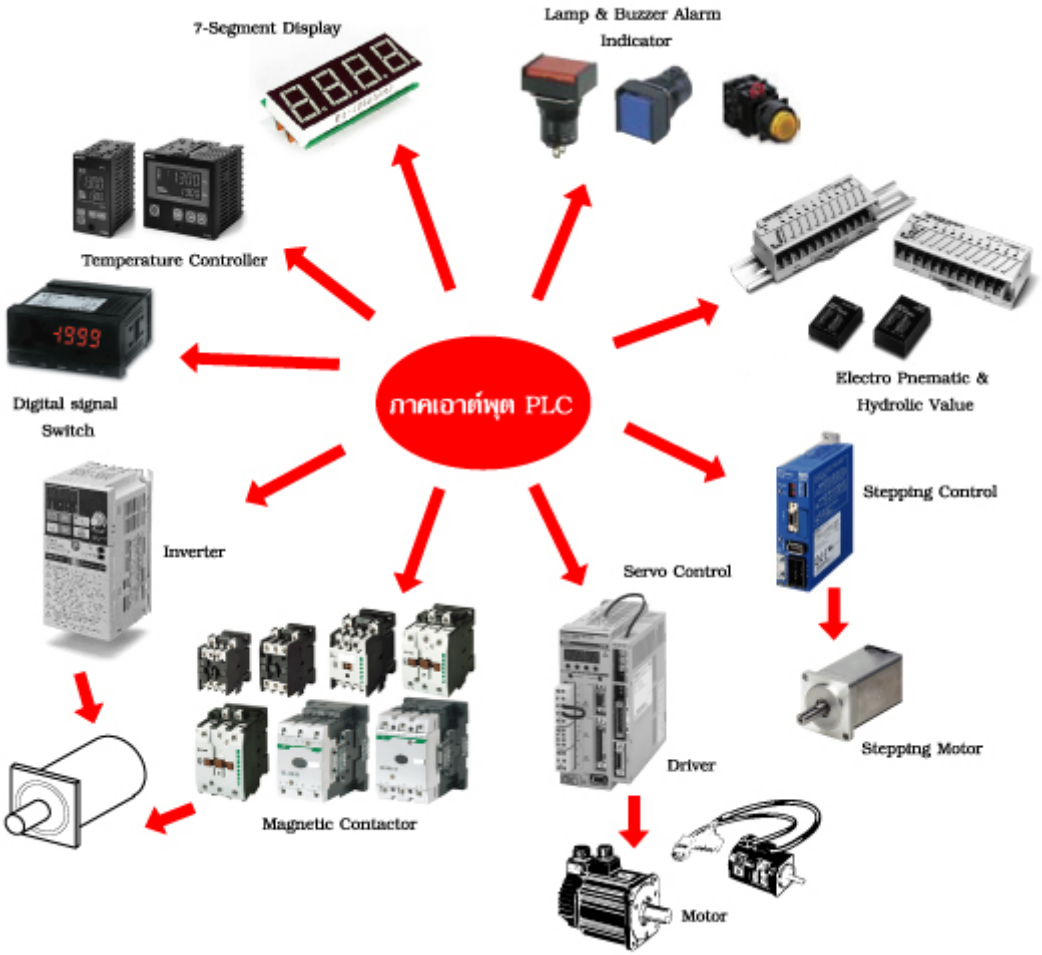
อุปกรณ์ที่วัดค่าออกมาเป็นปริมาณอนาลอกส่วนมากเป็นการวัดระยะทาง, วัดความเร็ว, วัดอุณหภูมิ, วัดปริมาณแสง, วัดความดันเป็นต้น แล้วแปลงค่าเป็นสัญญาณทางไฟฟ้าออกมา ดังนั้นเวลาที่อุปกรณ์เหล่านี้วัดค่าออกมาเป็นอนาลอกค่าใดๆ ผู้ใช้จำเป็นต้องทำตารางเปรียบเทียบค่าด้วย เพื่อที่จะกำหนดขนาดข้อมูลให้กับ PLC ให้ควบคุมตามที่ต้องการ วงจรภาคอินพุตแบบอนาลอกของ PLC จะมีลักษณะวงจรตามรูปที่ 1.19



รูปที่ 1.19 *วงจรอนาลอกอินพุตของ PLC*

1.5.4 ภาคเอาต์พุต (Output Unit)

ภาคเอาต์พุตของ PLC ทำหน้าที่ส่งสัญญาณออกไปขับโหลดชนิดต่างๆ ตามเงื่อนไขที่ได้โปรแกรมเอาไว้ ชนิดของโหลดที่สามารถนำมาต่อกับภาคเอาต์พุต สามารถแยกออกเป็นกลุ่มได้ดังนี้



รูปที่ 1.20 *กลุ่มอุปกรณ์ที่ต่อกับภาคเอาต์พุต PLC*

จากรูปที่ 1.20 กลุ่มอุปกรณ์ต่างๆ ที่ต่อกับภาคเอาต์พุต PLC นั้น ในแต่ละกลุ่มก็จะควบคุมลักษณะของงานแตกต่างกันไปตามคุณสมบัติของอุปกรณ์นั้นๆ การต่อวงจรเข้าภาคเอาต์พุต PLC จะมีมาตรฐานทางอุตสาหกรรมกำกับอยู่เช่นกัน จึงทำให้ผู้ใช้ไม่ต้องใช้อุปกรณ์เสริมมาก เพียงแต่ดูรายละเอียดการต่อให้เข้าใจก็เพียงพอแล้ว

ชนิดเอาต์พุตของ PLC จะมีให้เลือกใช้อยู่ 2 ลักษณะเช่นเดียวกับภาคอินพุตคือ

1) ดิจิตอลเอาต์พุต (Digital Output)

2) อนาลอกเอาต์พุต (Analog Output)

1) ดิจิตอลเอาต์พุต (Digital Output)

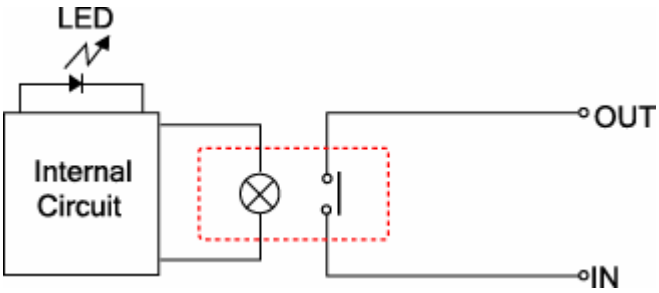
อุปกรณ์ที่สามารถสั่งการทำงานได้เพียง "ON" หรือ "OFF" จัดว่าเป็นการควบคุมแบบดิจิตอลเอาต์พุตโดยมีชนิดของเอาต์พุตให้เลือกใช้ 3 แบบคือ

1-1) เอาต์พุตชนิด "Relay Contact Output"

1-2) เอาต์พุตชนิด "Transistor Output"

1-3) เอาต์พุตชนิด "Solid State Relay: SSR Output"

1-1) ***เอาต์พุตชนิดรีเลย์ "Relay Contact Output"*** เอาต์พุตชนิดรีเลย์สามารถนำเอาต์พุตไปขับโหลด ACหรือ DC ก็ได้ ลักษณะวงจรดังรูปที่ 1.21



รูปที่ 1.21 *วงจรเอาต์พุตแบบรีเลย์*

การเปิด/ปิดหน้าสัมผัสของรีเลย์จะอาศัยหลักการทำงานของสนามแม่เหล็ก ดังนั้นเวลาที่นำหน้าสัมผัสรีเลย์ไปใช้งานจึงเปรียบได้เสมือนสวิตช์ควบคุมแบบ NO หรือ NC จึงสามารถที่จะใช้หน้าสัมผัสไปควบคุมโหลดได้ทั้งชนิด AC หรือ DC ซึ่งข้อพิจารณาในการเลือกใช้ต้องพิจารณาความสามารถทนกระแสและแรงดันได้สูงสุดเท่าไร ปกติแล้วภาคเอาต์พุตของ PLC ที่ เลือกเป็นชนิดรีเลย์เอาต์พุตทนกระแสใช้งานตามปกติได้ 2A จึงไม่เหมาะที่จะนำไปขับโหลด AC หรือ DC ที่มีกระแสสูงกว่า 2A คุณสมบัติต่างๆ ของภาคเอาต์พุตชนิดรีเลย์ แสดงไว้ในตารางที่ 1.4

กรณีโหลดที่ใช้งานมีกระแสกระชากสูงกว่า 2A มากๆ ไม่ควรใช้เอาต์พุตรีเลย์ต่อกับโหลดนั้นๆ โดยตรง ควรต่อผ่านรีเลย์บัพเฟอร์ที่สามารถทนกระแสได้ดีกว่า

ตารางที่ 1.4 คุณสมบัติภาคเอาต์พุตชนิดรีเลย์

รายละเอียด			คุณสมบัติ
อัตราทนการทำงานสูงสุด (Max. switching capacity)			2 A/250 VAC (COSϕ = 1) 2 A/24 VDC
อัตราทนการทำงานต่ำสุด (Min. switching capacity)			10 mA/5 VDC
อายุการใช้งาน (Relay Service Life)	ระบบไฟฟ้า	Resistance Load	300,000 ครั้ง
		Inductive Load	100,000 ครั้ง
	ระบบกลไก (Mechanical)		10 ล้านครั้ง
	Switching Rate		30 ครั้งต่อนาที
เวลาตอบสนอง	OFF Delay		15 mS (max)
	ON Delay		15 mS (max)

อายุการใช้งานจะขึ้นอยู่กับขนาดโหลดที่ใช้ต่อกับเอาต์พุตชนิดรีเลย์ไป ควบคุม จากตารางโหลดที่เป็นขดลวด (Inductive Load) จะทำให้อายุการใช้งานรีเลย์สั้นกว่าโหลดจำพวกหลอดไฟถึง 3 เท่า ส่วนในเรื่องเวลาตอบสนองตามคุณสมบัติภาคเอาต์พุตแบบรีเลย์ จะตอบสนองคำสั่งช้าที่สุด เมื่อเปรียบเทียบกับภาคเอาต์พุตแบบอื่นๆ

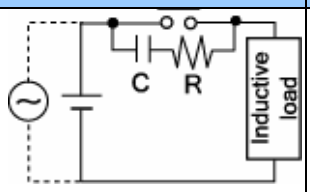
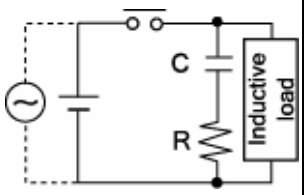
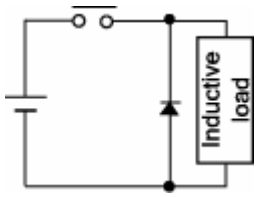
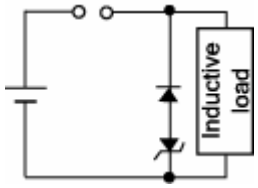
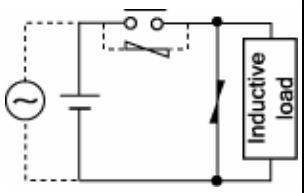
พิกัดการเปิด/ปิดวงจร(Switching Rate)

นอกจากอายุการใช้งานของเอาต์พุตแบบรีเลย์จะขึ้นอยู่กับขนาดของโหลดแล้ว ความถี่ในการเปิด/ปิดวงจรโหลดเป็นพิกัดอีกตัวหนึ่งที่ส่งผลต่ออายุการใช้งาน โดยปกติแล้วไม่ควรเปิด/ปิดวงจรโหลดเกินกว่า 30 ครั้งต่อนาที ถ้าจำเป็นต้องเปิด/ปิดวงจรบ่อยครั้ง ควรใช้เอาต์พุตทรานซิสเตอร์จะเหมาะสมกว่า

วงจรป้องกันหน้าคอนแทค

ในการใช้งานเอาต์พุตรีเลย์ให้มีอายุการใช้งานที่ยาวนานขึ้นควรต่อวงจรป้องกันหน้าคอนแทคเข้ากับรีเลย์ เพื่อลด Noise และป้องกันการสร้างกรด Nitric และ Carbide ซึ่งจะเกิดขึ้นขณะที่หน้าคอนแทคเปิดวงจร การใช้วงจรป้องกันจะช่วยลดผลกระทบดังกล่าวได้ ตารางข้างล่างแสดงตัวอย่างการต่อวงจรป้องกัน เวลาเราใช้เอาต์พุตรีเลย์ตัด/ต่อโหลดประเภท Inductive เช่น Solenoid valve จะทำให้เกิดการอาร์ค (Are) ขึ้นที่หน้าคอนแทค ถ้าสภาพแวดล้อมมีความชื้นสูงจะส่งผลทำให้เกิดกรด Nitric ซึ่งจะทำให้รีเลย์ทำงานผิดปกติได้ ดังนั้นควรใช้อุปกรณ์ลด Surge เพื่อลดปัญหาดังกล่าว

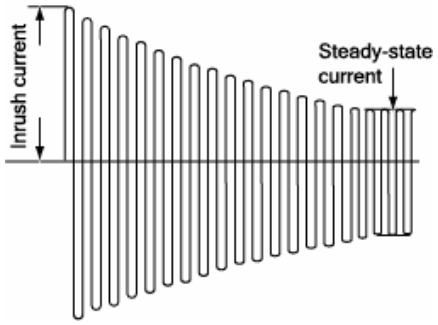
- **ตัวอย่าง Surge Suppressors**

ตัวอย่างวงจร		ไฟที่ใช้ได้		คุณลักษณะ	การเลือกอุปกรณ์
		AC	DC		
CR type		* (OK)	OK	อิมพีแดนซ์ของโหลดต้องน้อยกว่าวงจร RC เมื่อใช้รีเลย์กับแรงดันไฟ AC	ค่า C และ R ที่เหมาะสมคือ C = 0.1-0.5 uF ต่อกระแส Switching 1A และ R=0.5-1 โอมห์ ต่อแรงดัน Switching 1V อย่างไรก็ตามค่านี้อาจไม่เป็นค่าคงที่เหมาะสมเสมอไป ทั้งนี้ขึ้นอยู่กับโหลดและคุณสมบัติของรีเลย์ การเลือก C ควรให้มีค่า dielectric strength 200-300 V ถ้าใช้กับแรงดันไฟ AC และเป็น Capacitor ชนิด AC
		OK	OK	เวลาในการตัดวงจรจะช้าลงถ้าโหลดเป็น Inductive เช่น Solenoid Valve วงจรนี้จะใช้งานได้ผลดีถ้าต่อคร่อมโหลดเมื่อใช้แรงดันเป็น 24-48 V และต่อคร่อมหน้าคอนแทค ถ้าแรงดันเป็น 100-240 V	
Diode type		NG	OK	พลังงานสะสมอยู่ที่ Coil ของโหลด Inductive จะสร้างเป็นกระแสไฟฟ้าไหลผ่าน Diode ที่ต่อคร่อมอยู่กับ Coil นั้น วงจรนี้จะมีผลทำให้เวลาการตัดวงจรนานกว่าแบบ RC	ควรใช้ Diode ที่มี Reverse breakdown voltage เป็น 10 เท่าของแรงดันใช้งาน
Diode + Zener diode type		NG	OK	วงจรนี้จะทำงานดีกว่าแบบ Diode ในงานบางประเภทแต่เวลาตัดวงจรจะนานมาก	Breakdown voltage ของ Zener diode ควรเท่ากับแรงดันใช้งาน
Varistor type		OK	OK	วงจรนี้จะป้องกันแรงดันสูงที่เกิดขึ้นที่หน้าคอนแทคเนื่องจาก Varistor มีคุณสมบัติรักษาแรงดันให้คงที่ วงจรนี้จะใช้งานได้ผลดีถ้าต่อคร่อมโหลดเมื่อใช้แรงดันเป็น 24-48 V และต่อคร่อมหน้าคอนแทคถ้าแรงดันเป็น100-240 V	Cutoff voltage (Vc) ต้องเป็นไปตามเงื่อนไขต่อไปนี้ Contact dielectric strength > Vc > Supply voltage (กรณีไฟ AC ให้คูณ 2 ของค่าที่ได้)

โหลดความต้านทานและโหลดอินดักทีฟ

ความสามารถในการเปิด/ปิด (Switching power) ของโหลด Inductive จะต่ำกว่าโหลดความต้านทาน เนื่องจากมีพลังงานแม่เหล็กไฟฟ้าสะสมในคอล์ยของโหลดอินดักทีฟ ตารางข้างล่างแสดงกระแสกระชาก (Inrush) ที่เกิดจากโหลดประเภทต่าง ๆ

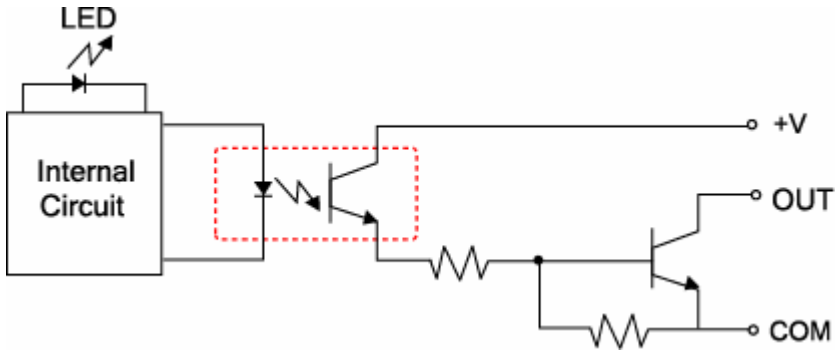
โหลด AC กับ Inrush Current

ประเภทโหลด	อัตรากระแสกระชากต่อกระแสไฟในภาวะปกติ	รูปคลื่น
Solenoid 	ประมาณ 10	
Incandescent bulb 	ประมาณ 10 - 15	
Motor 	ประมาณ 5 - 10	
Relay 	ประมาณ 2 - 3	
Capacitor 	ประมาณ 20 - 50	
Resistive load 	1	

1-2) ***เอาต์พุตชนิดทรานซิสเตอร์ (Transistor Output)*** เอาต์พุตแบบทรานซิสเตอร์ มีให้เลือกใช้อยู่ 2 ประเภทคือ

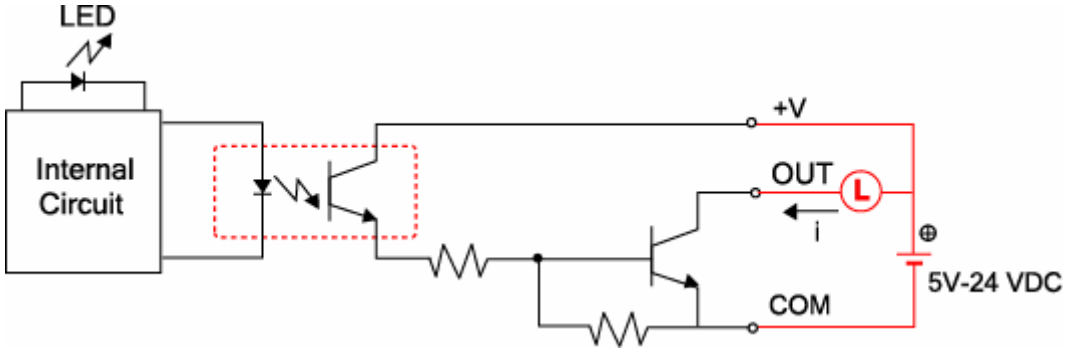
- เอาต์พุตทรานซิสเตอร์แบบ NPN
- เอาต์พุตทรานซิสเตอร์แบบ PNP

- **เอาต์พุตทรานซิสเตอร์แบบ NPN** มีลักษณะวงจรดังรูปที่ 1.22



รูปที่ 1.22 *วงจรภายในเอาต์พุตทรานซิสเตอร์แบบ NPN*

จากวงจรภายในจะใช้ออปโตทรานซิสเตอร์ผลิตสัญญาณขับทรานซิสเตอร์Q1 โดย Q1 จะทำหน้าที่ขับโหลดอีกที วงจรลักษณะนี้ทำให้วงจรภายในแยกสัญญาณกราวนด์ออกจากวงจรภาคเอาต์พุตได้ส่วนลักษณะการต่อวงจรใช้งานนั้นสามารถต่อใช้งานขับโหลดได้เฉพาะ DC เท่านั้น ดังรูปที่ 1.23



รูปที่ 1.23 *การต่อใช้งานเอาต์พุตทรานซิสเตอร์แบบ NPN*

การต่อขับโหลดดังรูปที่ 1.23 เป็นการต่อแบบซิงค์ (Sink type) คือดึงกระแสเข้าสู่ภาคเอาต์พุต ดังนั้นทรานซิสเตอร์ต้องทนกระแสซิงค์ได้ เพื่อป้องกันไม่ให้ทรานซิสเตอร์พังที่ขาอิมิเตอร์ Q1 เขียนว่า COM (COMMON) เนื่องจากว่าเวลานำภาคเอาต์พุตแบบนี้ไปใช้งานจริงจะมีวงจรลักษณะนี้ต่ออยู่หลายชุดเช่น 8, 16, 32 ชุดเป็นต้น วงจรใช้งานจริงก็จะต่อขาอิมิเตอร์ร่วมกันแล้วดึงออกมาเป็นขาที่เขียนว่า “COM” นั่นเองและที่ขั้ว +V ก็ต่อร่วมเช่นกัน

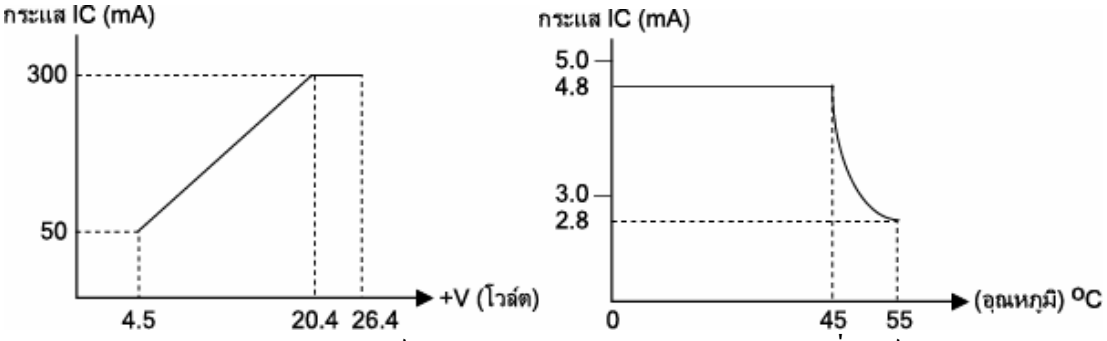
คุณสมบัติส่วนต่างๆ ของภาคเอาต์พุตทรานซิสเตอร์แบบ NPN นี้ สามารถดูรายละเอียดได้ดังตารางที่ 1.4

ตารางที่ 1.4 คุณสมบัติภาคเอาต์พุตทรานซิสเตอร์แบบ NPN

รายละเอียด		คุณสมบัติ
แหล่งจ่ายไฟ +V		5-24 VDC (40mA min) ±10% (2.5 mA X จำนวนบิตที่ “ON”)
อัตราทนการทำงานสูงสุด (Max. switching capacity)		50 mA ที่แรงดัน 4.5 V - 300 mA ที่แรงดัน 26.4 V
กระแสรั่วไหล (Leakage Current)		0.1 mA (สูงสุด)
แรงดันไฟฟ้า (Residual Voltage)		0.8 VDC (สูงสุด)
เวลาตอบสนอง	OFF Delay	0.1 mS (สูงสุด)
	ON Delay	0.4 mS (สูงสุด)

อัตราทนการทำงานสูงสุด (Max. Switching Capacity)

จัดเป็นตัวแปรที่ต้องคำนึงถึงเวลานำไปใช้งาน เพราะว่าภาคเอาต์พุต PLC เวลาที่ผลิตออกมาใช้งาน จะมีวงจรทรานซิสเตอร์มากกว่า 1 ชุดเสมอ เช่น 8, 16 ชุด ทำให้ต้องพิจารณากระแสที่สามารถจะขับโหลดได้พร้อมกันทุกชุดของเอาต์พุตด้วย ดังรูปที่ 1.24

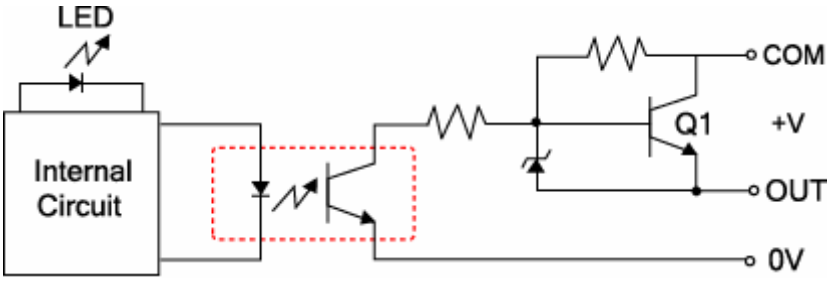


ก. กระแสขับโหลดต่อเอาต์พุต 1 ชุด ข. กระแสขับโหลดที่เอาต์พุต 16 ชุด

***รูปที่ 1.24** กราฟกระแส (IC) ขับโหลด*

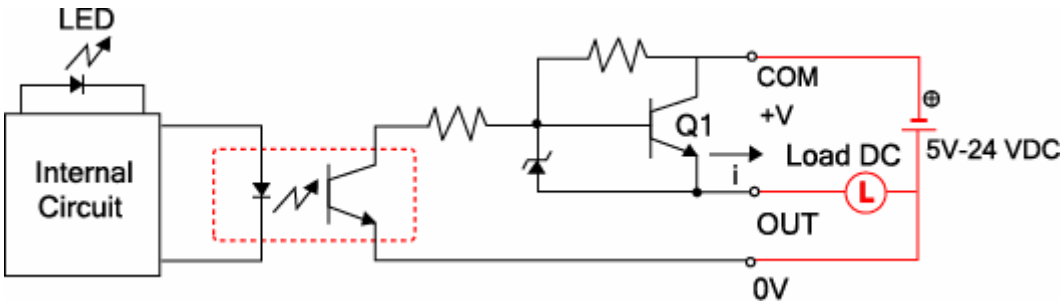
จากกราฟจะพบว่าถ้าขับโหลดทีละชุดไม่พร้อมกัน สามารถที่จะขับโหลดได้ถึง 300 mA ที่แรงดัน 24 VDC ได้ แต่เมื่อขับโหลดพร้อมกันทั้งหมด 16 ชุด ก็จะทำให้จ่ายกระแส (IC) ได้เพียง 4.8 mA ต่อ 1 โหลด ดังนั้นเวลาใช้ภาคเอาต์พุตแบบทรานซิสเตอร์ ถึงแม้ว่าสามารถตอบสนองโหลดได้เร็วกว่ารีเลย์แต่มีข้อจำกัดในเรื่องกระแส ส่วนใหญ่จะใช้ภาคเอาต์พุตทรานซิสเตอร์ขับโหลดวงจรอิเล็กทรอนิคส์แบบต่างๆ เช่น 7-Seg Display, Digital Controller, Servo Driver เป็นต้น

- **ภาคเอาต์พุตทรานซิสเตอร์แบบ PNP** มีลักษณะวงจรดังรูปที่ 1.25



***รูปที่ 1.25** วงจรภายในเอาต์พุตทรานซิสเตอร์แบบ PNP*

ลักษณะวงจรคล้ายวงจรของเอาต์พุตทรานซิสเตอร์แบบ NPN เพียงแต่เปลี่ยนวงจรส่วน Q1 เท่านั้น ลักษณะการต่อวงจรสามารถต่อได้ดังรูปที่ 1.26



***รูปที่ 1.26** การต่อใช้งานเอาต์พุตทรานซิสเตอร์แบบ PNP*

ต่อวงจรโดยขั้วที่เขียนว่า COM ของภาคเอาต์พุต ให้ต่อไฟบวก (+V) ขา 0V ต่อกับไฟ 0V และขา OUT ต่อกับโหลด

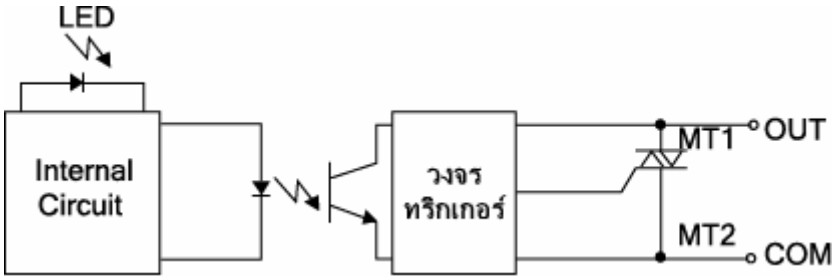
การต่อวงจรลักษณะแบบนี้เป็นการต่อแบบซอร์ส (Source type) โดยที่ทรานซิสเตอร์ Q1 ต้องทนกระแสที่จะจ่ายให้โหลดได้ เราอาจจะเรียกว่า กระแสซอร์ส (I source) คุณสมบัติของวงจรเอาต์พุตแบบนี้แสดงไว้ดังตารางที่ 1.6

ตารางที่ 1.6 คุณสมบัติภาคเอาต์พุตทรานซิสเตอร์แบบ PNP

รายละเอียด		คุณสมบัติ
แหล่งจ่ายไฟขา +V (COM)		5-24 VDC (60mA min) ±10% (3.5 mA X จำนวนบิตที่ “ON”)
อัตราทนการทำงานสูงสุด (Max. switching capacity)		50 mA ที่แรงดัน 4.5 V - 300 mA ที่แรงดัน 26.4 V
กระแสรั่วไหล (Leakage Current)		0.1 mA (สูงสุด)
แรงดันไฟฟ้า (Residual Voltage)		0.8 V (สูงสุด)
เวลาตอบสนอง	OFF Delay	0.1 mS (สูงสุด)
	ON Delay	0.4 mS (สูงสุด)

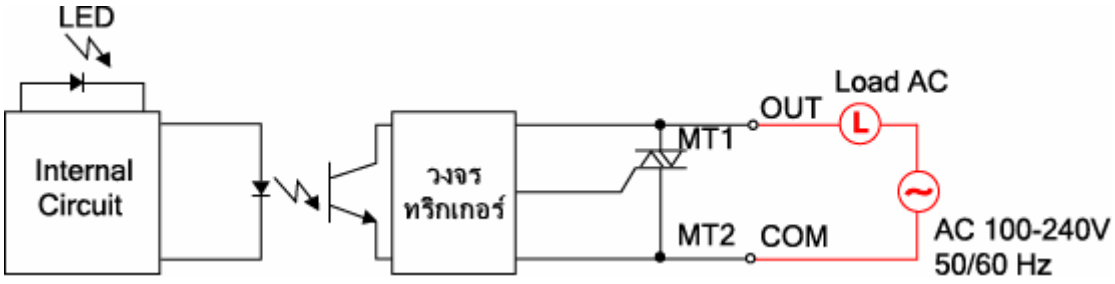
ภาคเอาต์พุตทรานซิสเตอร์แบบ PNP จะมีคุณสมบัติในเรื่องอัตราทนการทำงานสูงสุด (Max switching capacity) เหมือนกับภาคเอาต์พุตทรานซิสเตอร์แบบ NPN ซึ่งดูได้จากรูปที่ 1.24 เช่นกัน

1-3) ***เอาต์พุตชนิดโซลิตสเตรทรีเลย์ (Solid State Relay: SSR)*** เอาต์พุตประเภทนี้จะนำมาใช้ควบคุมโหลด AC ที่ต้องการควบคุมความเร็วในการตอบสนองที่ดีกว่าใช้เอาต์พุตแบบรีเลย์ อุปกรณ์ภาคเอาต์พุตที่ใช้จะใช้ไตรแอดเป็นสวิตช์ควบคุมโหลด ลักษณะวงจรเอาต์พุตแบบ SSR นี้ แสดงไว้ดังรูปที่ 1.27



รูปที่ 1.27 *วงจรภายในเอาต์พุตโซลิตสเตรทรีเลย์*

คุณสมบัติของไตรแอดจะทำให้สามารถควบคุมโหลด AC ได้ทั้งซีกบวกและซีกลบของรูปคลื่นไซน์ (Sine wave) ส่วนวงจรทริกเกอร์ทำหน้าที่กระตุ้นไตรแอดให้ทำงานสอดคล้องกับรูปคลื่นไซน์ อย่างน้อยก็เป็นการป้องกันไตรแอดได้ระดับหนึ่ง การต่อวงจรเอาต์พุตแบบ SSR สามารถต่อใช้งานได้ดังรูปที่ 1.28



รูปที่ 1.28 *การต่อใช้งานเอาต์พุต SSR*

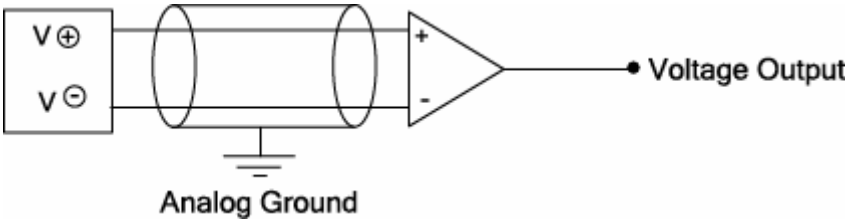
ลักษณะการต่อวงจรโหลดกับภาคเอาต์พุต SSR จะต่อในลักษณะอนุกรมกันโดยขาข้างหนึ่งของโหลดต่อกับขา OUT อีกข้างต่อเข้ากับแหล่งจ่ายไฟสลับ ส่วนขาอีกข้างหนึ่งคือขา COM นำไปต่อกับขั้วแหล่งจ่ายไฟสลับอีกข้าง คุณสมบัติของเอาต์พุต SSR ดูได้จากตารางที่ 1.7

ตารางที่ 1.7 คุณสมบัติภาคเอาต์พุตแบบโซลิตสเตรทรีเลย์ (SSR)

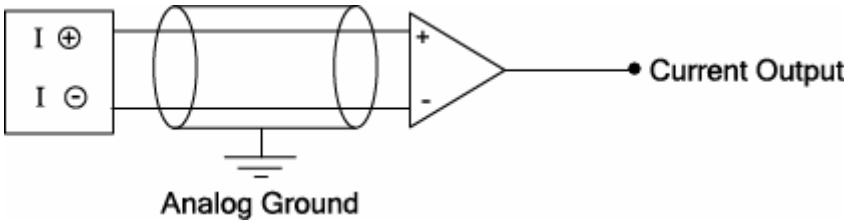
รายละเอียด		คุณสมบัติ
อัตราทนการทำงานสูงสุด (Max. switching capacity)		100-240 VAC (0.4A)
กระแสรั่วไหล (Leakage Current)		1 mA (สูงสุด) ที่ 100 VAC 2 mA (สูงสุด) ที่ 200 VAC
แรงดันไฟฟ้า (Residual Voltage)		1.5 V (สูงสุด) (0.4A)
เวลาตอบสนอง	OFF Delay	6 mS (สูงสุด)
	ON Delay	½ cycle + 5 mS (สูงสุด)

- **อนาลอกเอาต์พุต (Analog Output)**

ภาคเอาต์พุตของ PLC แบบอนาลอกเป็นการเพิ่มความสามารถให้ PLC ส่งสัญญาณควบคุมเชิงปริมาณได้ ค่าที่จะส่งออกไปก็จัดเป็นค่าสัญญาณมาตรฐานเหมือนภาคอินพุตแบบอนาลอกคือ สัญญาณ 0-10 VDC, ±10 VDC และ 1-5 V (4-20mA) ลักษณะกราฟภาคเอาต์พุตที่จะส่งสัญญาณออกไปเหมือนกับกราฟอนาลอกอินพุตดังรูปที่ 1.17 การส่งสัญญาณของอนาลอกเอาต์พุตจะส่งสัญญาณ 2 แบบคือ แรงดันและกระแส การต่อสายสัญญาณเพื่อเลือกสัญญาณเป็นกระแสหรือแรงดันของภาคเอาต์พุตอนาลอกจะมีสัญญาณกำกับขั้วไว้ สามารถแยกการต่อได้ 2 ลักษณะดังรูปที่ 1.29



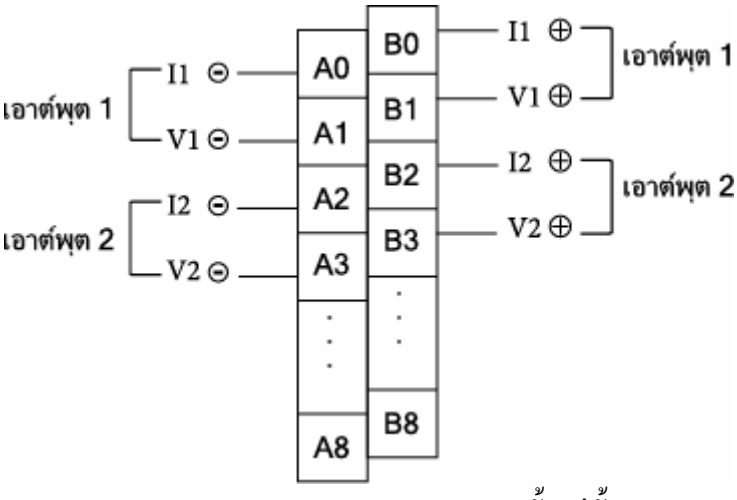
ก. ส่งสัญญาณแบบแรงดัน (Voltage Output)



ข. ส่งสัญญาณแบบกระแส (Current Output)

***รูปที่ 1.29** ส่งสัญญาณแบบกระแส/แรงดันของอนาลอกเอาต์พุต*

วิธีการสังเกตขั้วต่อสายของอนาลอกเอาต์พุต จะมีสัญลักษณ์แยกไว้ว่าเป็นของอนาลอกเอาต์พุตชนิดใด ดังรูปที่ 1.30

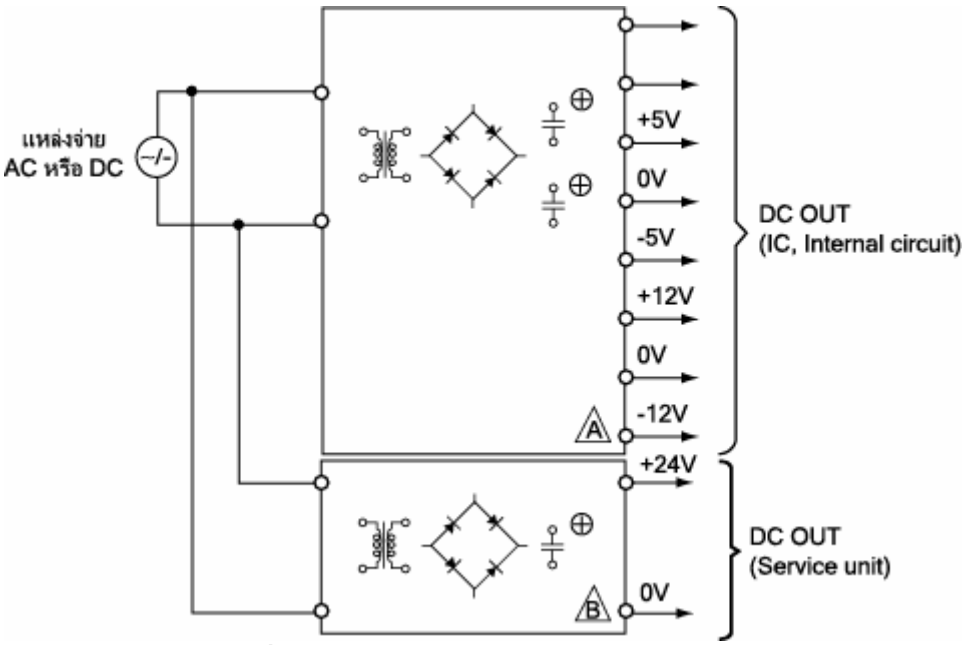


หมายเหตุ : ตั้งแต่ขั้ว A4-A8/B4-B8 ว่าง

รูปที่ 1.30 *ตำแหน่งขั้วอนาลอกเอาต์พุต*

1.5.5 ภาคแหล่งจ่ายพลังงาน (Power Supply Unit)

ภาคแหล่งจ่ายพลังงาน จะทำหน้าที่จ่ายพลังงานให้กับอุปกรณ์ภายใน PLC ได้แก่ อุปกรณ์ไอซี, ไฟเลี้ยงวงจรกำหนดการทำงานแบบต่างๆ เป็นต้น นอกจากนี้ยังจ่ายพลังงานเลี้ยงวงจรที่จะนำมาต่อกับ PLC ทั้งภาคอินพุต/เอาต์พุต ไดอะแกรมของแหล่งจ่ายพลังงานเขียนไดอะแกรมได้ดังรูปที่ 1.31



รูปที่ 1.31 *ไดอะแกรมภาคแหล่งจ่ายไฟ PLC*

แหล่งจ่ายพลังงานของ PLC จะแบ่งออกเป็น 2 ชุด ชุดหนึ่งสำหรับอุปกรณ์และวงจรภายในแต่ละโมดูลต่างๆ ของ PLC อีกชุดหนึ่งเป็นตัวจ่ายพลังงาน (Service Unit 24VDC) 24VDC สำหรับการต่อวงจรภาคอินพุตหรือเอาต์พุตก็ได้ โดยปกติแล้วชุดบริการ 24VDC ชุดนี้จะจ่ายกระแสได้ค่อนข้างต่ำ ไม่เหมาะสำหรับนำไปจ่ายโหลดที่ดึงกระแสสูง ส่วนมากจะนำไปต่อใช้งานเฉพาะวงจรภาคอินพุต PLC เท่านั้น แต่ถ้าจะนำไปต่อสำหรับทดสอบเครื่อง PLC หรือชุดฝึกทดลอง ก็ไม่จำเป็นต้องใช้แหล่งจ่ายภายนอกเพิ่ม ยกตัวอย่างเช่น ชุดฝึกทดลอง PLC ของออมรอน เป็นต้น

สำหรับการใช้งานจริง แหล่งจ่ายจะถูกออกแบบมา 2 ลักษณะ ตามโครงสร้างภายนอก PLC คือ แหล่งจ่ายชนิดที่รวมอยู่ในตัว PLC เลย เช่น CP1L จะมีชุดจ่ายพลังงานในตัวเพียงแค่ป้อนไฟให้กับ CP1L มันจะจัดสรรพลังงานให้กับอุปกรณ์ต่างๆ บนตัว PLC อีกชนิดหนึ่งจะแยกออกมาเป็นโมดูล (Module) ลักษณะดังรูปที่ 1.32

***รูปที่ 1.32** แหล่งจ่ายไฟชนิดโมดูล*

โดยปกติแล้วแหล่งจ่ายพลังงานที่ผลิตออกมาสำหรับขายทั่วโลก จะออกแบบให้ใช้ระบบไฟได้หลายแบบ เพื่อที่จะทำให้ PLC ใช้ควบคุมระบบไฟฟ้าได้หลายแบบนั่นเอง คุณสมบัติของแหล่งจ่ายไฟของ PLC จะมีคุณสมบัติดังนี้

แหล่งจ่ายไฟ: 100-240 VAC 50/60 Hz หรือ 24 VDC
ชุดบริการ 24 VDC: 24 V (0.5A)

ส่วนการเลือกขนาดวัตต์จะคำนวณจากโมดูลต่างๆ ของ PLC ที่ใช้งานซึ่งผู้ผลิตได้ออกแบบเผื่อไว้ให้เรียบร้อยแล้ว

บทที่ 2
การติดตั้งและออกแบบระบบ

ก่อนที่จะเริ่มต้นเขียนโปรแกรมกันคงหนีไม่พ้นการติดตั้งระบบ PLC ในบทนี้จะแนะนำวิธีการและข้อควรระวังต่างๆ ในการติดตั้งระบบ PLC ที่ถูกต้องเพื่อให้ระบบมีเสถียรภาพในการทำงานและยืดอายุการใช้งาน นอกจากนั้นจะกล่าวถึงขั้นตอนการออกแบบระบบควบคุมอีกด้วย

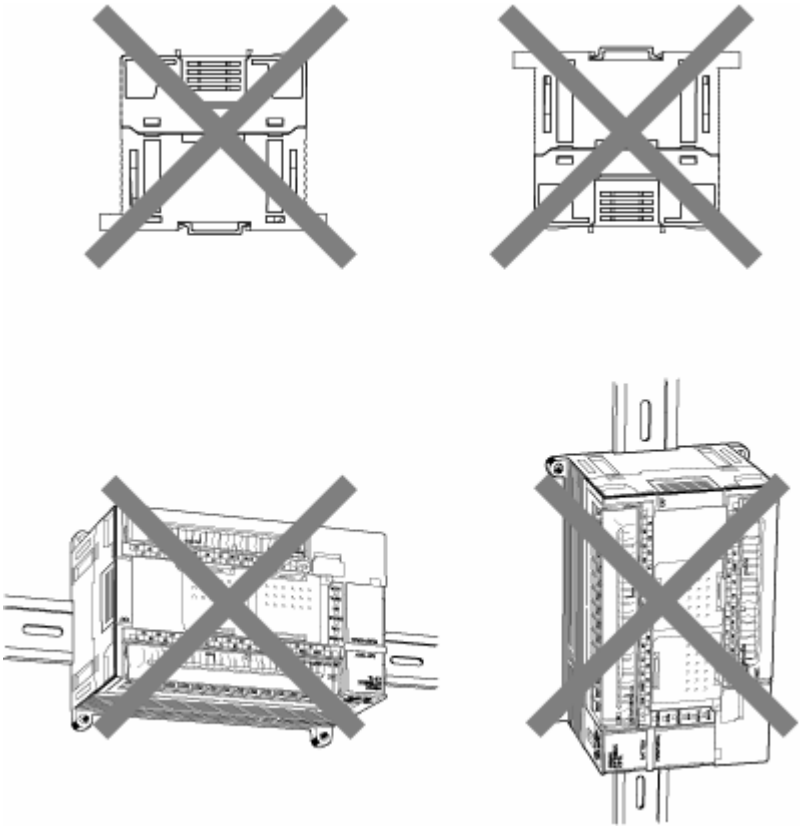
2.1 การติดตั้ง PLC

ผู้ติดตั้งควรอ่านคู่มือของ PLC แต่ละรุ่นเพื่อการติดตั้งที่ถูกต้อง ในหนังสือเล่มนี้จะกล่าวถึงหลักการทั่วๆ ไปที่ควรระวัง โดยยกตัวอย่าง PLC รุ่น CP1 เป็นอุปกรณ์อ้างอิง

2.1.1 สภาพแวดล้อมในการติดตั้ง

ไม่ควรติดตั้ง PLC ในสภาวะแวดล้อมดังต่อไปนี้

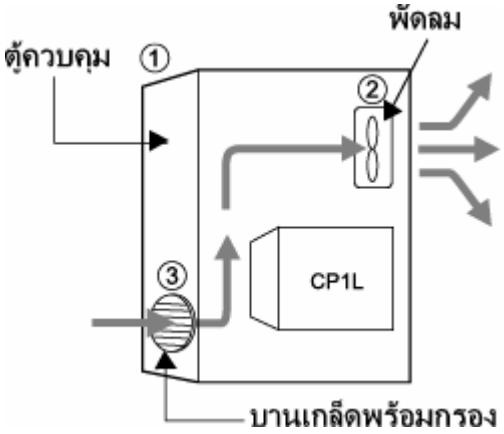
- ไม่ควรติดตั้ง PLC ในสถานที่ที่มีอุณหภูมิและความชื้นเกินกว่าค่าพิกัดที่กำหนดไว้
- หลีกเลี่ยงบริเวณที่มีฝุ่นมากและมีกรดเกลือ หรือสารเคมี เช่น คอลไลด์
- ไม่ควรติดตั้งใกล้สายส่งไฟกำลังและบริเวณที่มีสนามไฟฟ้าและคลื่นวิทยุที่มีกำลังแรง เช่น อินเวอร์เตอร์ เป็นต้น
- ไม่ควรติดตั้ง PLC ในลักษณะตามรูปที่ 2.1



รูปที่ 2.1 แสดงทิศทางในการติดตั้งที่ไม่ถูกต้อง

2.1.2 การติดตั้งในตู้ควบคุม

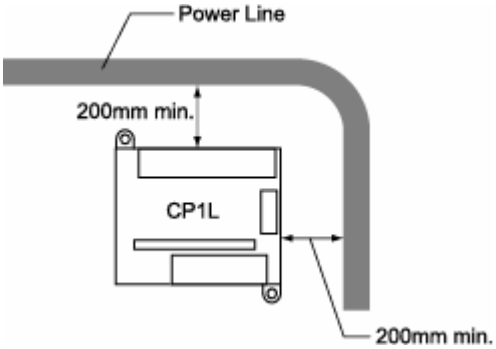
ควรจัดให้มีการไหลเวียนของอากาศที่เพียงพอในการระบายความร้อน และอย่าติดตั้งอุปกรณ์ที่กำเนิดความร้อนใกล้ PLC (เช่น Heater และ Transformer) ถ้าอุณหภูมิแวดล้อมสูงกว่า 55°C ควรติดตั้งพัดลมหรือเครื่องปรับอากาศเพื่อระบายความร้อน ดังแสดงตัวอย่างในรูปข้างล่างนี้



2.1.3 การลดปัญหาจาก Noise และกระแส Spike

2.1.3.1 Noise จากอุปกรณ์แรงดันสูง

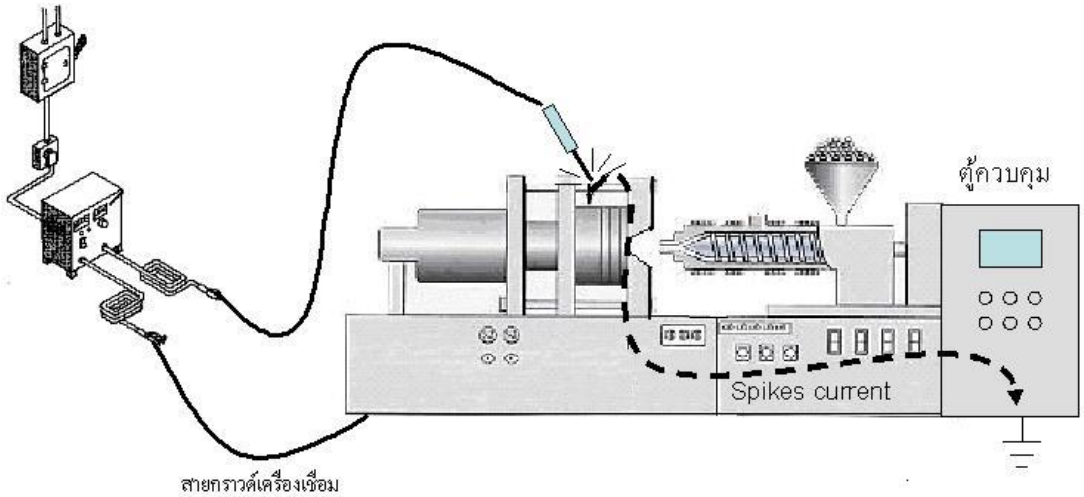
ควรหลีกเลี่ยงการติดตั้ง PLC ในตู้ควบคุมที่มีอุปกรณ์แรงดันสูงติดตั้งอยู่ และพยายามรักษาระยะห่างของ PLC จากสายส่งกำลังอย่างน้อย 200 มม.



2.1.3.2 กระแสไปก์ (Spike) จากเครื่องเชื่อมไฟฟ้า

สไปก์ (Spike) คือ แรงดันหรือกระแสไฟฟ้าที่สูงอย่างมหาศาลและเกิดขึ้นอย่างเฉียบพลัน ซึ่งอาจเกิดจากฟ้าผ่า แต่มีอุปกรณ์ชนิดหนึ่งที่สามารถสร้างความเสียหายให้อุปกรณ์ควบคุมอีเล็กทรอนิกส์ เช่น พีแอลซีและเซนเซอร์ อุปกรณ์นั้นก็คือ เครื่องเชื่อมไฟฟ้า

โดยปกติอุปกรณ์ควบคุมอิเล็กทรอนิกส์จะต่อกราวด์เข้ากับโครงของเครื่องจักรหรือตู้ควบคุม เมื่อเราใช้เครื่องเชื่อมไฟฟ้ากับส่วนที่เป็นโลหะซึ่งเป็นทางเดินของระบบกราวด์นี้อาจทำให้กระแสสไปก์ (Spike current) ที่เกิดจากเครื่องเชื่อมไหลผ่านโลหะต่างๆ เหล่านี้เข้าสู่อุปกรณ์ควบคุมอิเล็กทรอนิกส์ซึ่งส่งผลให้อุปกรณ์ดังกล่าวเสียหายได้ บ่อยครั้งที่เราพบว่า PLC เกิดความเสียหายจากกรณีดังกล่าว เพราะมีพนักงานบางท่านใช้เชื่อมไฟฟ้าที่หน้างาน แต่อาจไม่เกิดขึ้นทุกครั้งเสมอไป เพราะอาจมีปัจจัยอื่นมาเกี่ยวข้องด้วย เช่นกระแสสไปก์อาจไหลสู่กราวด์ก่อนมาถึงอุปกรณ์ควบคุมอีเล็กทรอนิกส์



วิธีการลดปัญหาดังกล่าว อาจทำได้โดย

- ถอดระบบกราวด์ของ PLC หรืออุปกรณ์ควบคุมเป็นการชั่วคราว หรือติดตั้งตู้ควบคุมหลังจากติดตั้งอุปกรณ์ทางกลและเครื่องจักรแล้ว

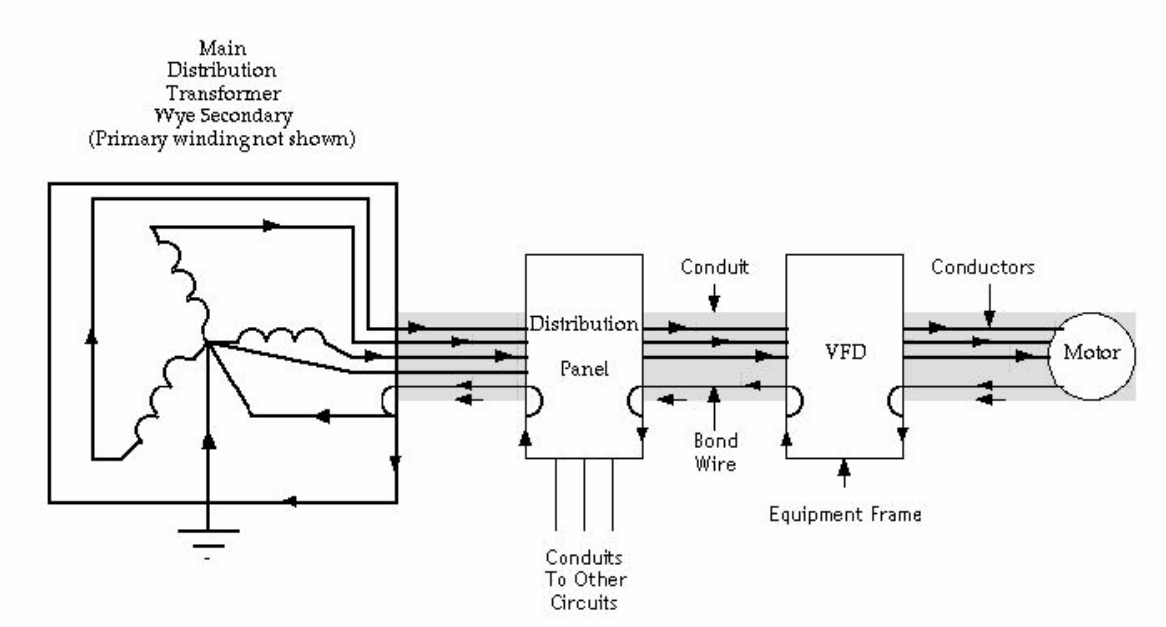
- ควรหลีกเลี่ยงการเชื่อมใกล้อุปกรณ์ควบคุมและพยายามวางสายกราวด์ของเครื่องเชื่อมให้ใกล้จุดเชื่อมให้มากที่สุดและยึดให้แน่น เพื่อให้กระแสไปก์กลับสู่กราวด์ได้สะดวก

2.1.3.3 Noise จาก AC Drive

Noise จาก AC Drive จะก่อให้เกิดสัญญาณรบกวนความถี่วิทยุ Radio Frequency Interference (RFI) ในช่วง 0.5 MHz ถึง 1.7 MHz และสร้าง Electromagnetic Interference Frequencies (EMI) ในช่วง 1.7 MHz ถึง 30 MHz ความถี่สูงเหล่านี้เกิดจากการทำงานของชุด PWM ซึ่งเกิดจากการลัดวงจรชั่วขณะของ IGBT นอกจากนั้น EMI ยังเกิดได้จาก Harmonics ซึ่งเกิดขึ้นจาก “Reflected wave” ที่มีเหตุมาจาก Capacitive ของสายมอเตอร์ที่ยาวและมีผลต่ออิมพีแดนซ์ที่ไม่สอดคล้องกันของสายมอเตอร์กับขดลวดมอเตอร์ รวมๆ แล้วเราเรียก EMI/RFI นี้ว่า Electrical Noise

Noise ที่เกิดขึ้นอาจย้อนกลับจาก AC Drive สู่แหล่งจ่ายไฟ (Power line) และส่งผลกระทบกับอุปกรณ์อื่นๆ เช่น พีแอลซีและเซนเซอร์ เป็นต้น

EMI/RFI จะแพร่กระจายไปตามตัวมอเตอร์สู่สายมอเตอร์และอาจกระจายสู่กราวด์ จากนั้น EMI/RFI จะพยายามแผ่กระจายกลับไปยังแหล่งจายไฟต้นกำลังที่จ่ายให้ AC Drive หรือ Inverter ซึ่งเส้นทางการย้อนกลับนี้อาจผ่านทางระบบกราวด์เข้าไปถึงจุดต่อ WYE ที่ขดลวดทุติยภูมิของหม้อแปลง ดังแสดงได้ในรูป



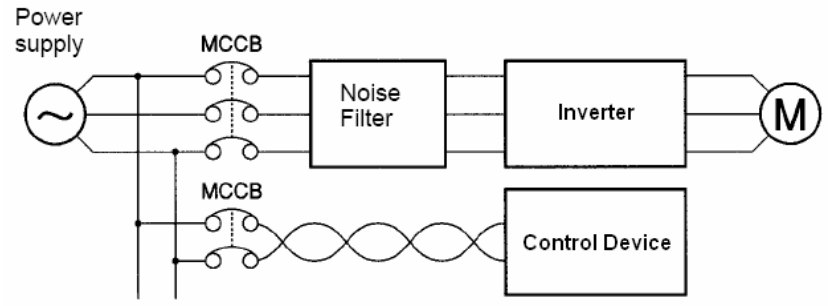
เส้นทางการไหลกลับเข้าสู่แหล่งจ่ายไฟกำลังของ EMI/RFI ตามท่อร้อยสายและอุปกรณ์ต่างๆ ของระบบกราวด์ ทำให้เกิด “Voltage Gradient” ซึ่งส่งผลกระทบกับอุปกรณ์ควบคุมต่างๆ โดยเราจะเห็นได้ว่าการกราวด์ของระบบอาจประกอบด้วยมอเตอร์ โครงตู้ต่างๆ ท่อร้อยสาย เหล็กโครงสร้าง เช่น I-beam ท่อน้ำ ดังนั้น EMI/RFI และ “Voltage Gradient” ที่เกิดขึ้นจะแพร่กระจายไปตามอุปกรณ์ต่างๆ เหล่านั้น ทำให้เกิดปัญหากับอุปกรณ์ควบคุมต่างๆ ที่อยู่ตามเส้นทางของมัน

การลดผลกระทบของ Noise

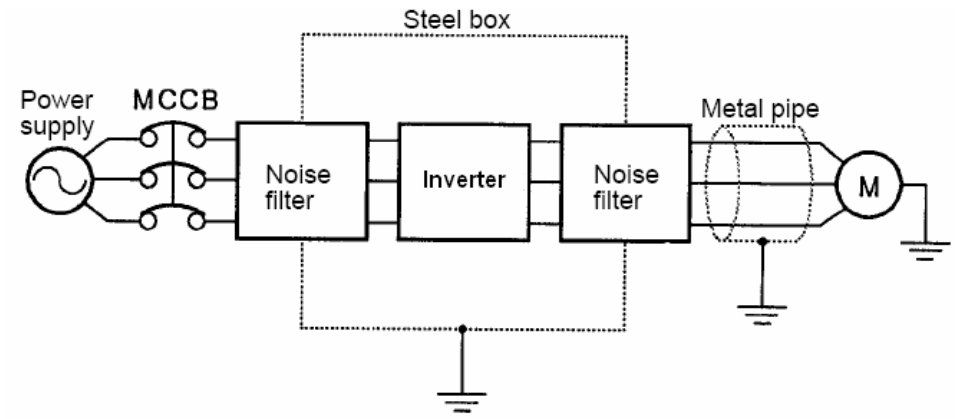
จากที่ได้กล่าวมาข้างต้น Noise จะแพร่กระจายไปทั่วกับอุปกรณ์ที่เป็นโลหะของระบบกราวด์ ซึ่งการแก้ไขปัญหาทำได้ค่อนข้างยาก เราขอแนะนำวิธีการลดปัญหาของ Noise ที่เกิดจาก Inverter ดังนี้

1. การใช้ Noise Filter

การใส่ Noise Filter ที่ด้านอินพุตของ Inverter จะช่วยลดผลกระทบของ Noise ที่จะถูกส่งย้อนกลับไปที่แหล่งจ่ายไฟ แต่ควรเลือก Noise Filter ที่ออกแบบหรือขนาดที่เหมาะสมกับ Inverter นั้นๆ ดังแสดงได้ในรูป



นอกจากนั้นเรายังสามารถลด Noise ที่เกิดขึ้นจาก “Reflecter Waves” จากมอเตอร์โดยการใส่ Noise Filter ที่ด้านเอาต์พุตของ Inverter ถ้าจำเป็น ดังแสดงได้ในรูป แต่ในกรณีจะใช้เมื่อสายมอเตอร์มีความยาวมาก



2. การลดความถี่ Carrier Frequency

ใน Inverter จะมีชุด PWM ทำหน้าที่ควบคุม IGBT เพื่อจ่ายไฟให้มอเตอร์ ซึ่งเราสามารถปรับเปลี่ยนความถี่ของชุด PWM ได้และเรียกความถี่นี้ว่า Carrier Frequency การลดความถี่นี้ลงอาจช่วยลด Noise ที่เกิดขึ้นได้

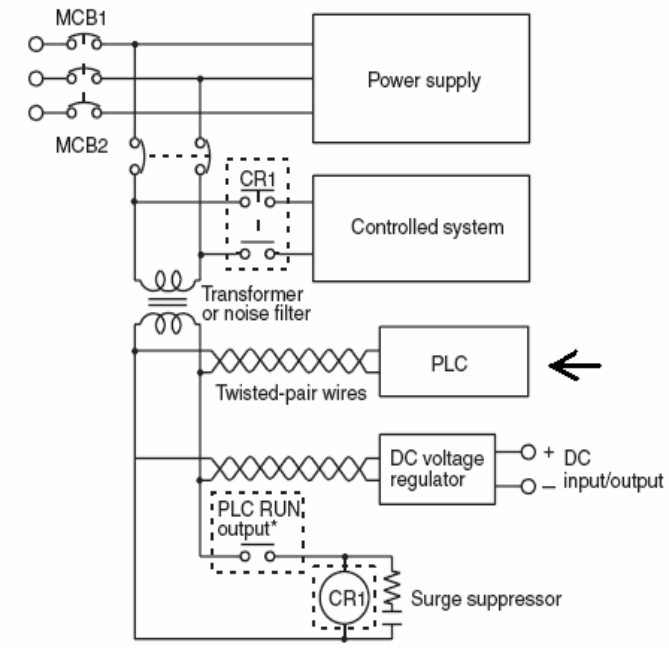
3. การแก้ปัญหาของสายสัญญาณ

การลด Noise ของสายสัญญาณ (Signal Circuit) อาจทำได้โดยใช้สาย Shielded และ Ferrite core ที่ปลายทั้งสองด้าน โดยปกติจะกราวด์ Shielded เข้ากับแหล่งจ่ายไฟสัญญาณเพียงด้านเดียว นอกจากนั้นเราควรแยกสายสัญญาณออกจากสายส่งกำลังและกราวด์ที่เป็นเส้นทางเดินของ Noise

2.1.4 การเดินสายสำหรับแหล่งจ่ายไฟ

- AC Power Supply

ในกรณีที่ระบบ PLC ที่ใช้งานต้องการแหล่งจ่ายไฟ AC ควรติดตั้ง Isolated Transformer หรือ Noise Filter เพิ่มเติมที่ด้านอินพุตของแหล่งจ่ายไฟ ดังแสดงในรูปที่ 2.2 เนื่องจาก Transformer มีหลักการทำงานด้วยการเหนี่ยวนำสนามแม่เหล็กไฟฟ้าที่เกิดจากขดลวดด้าน Primary และสร้างให้เกิดแรงดันไฟฟ้าขึ้นที่ด้าน Secondary โดยไม่มีการต่อถึงกันทางไฟฟ้า ด้วยหลักการนี้เมื่อเกิดการกระชากของแรงดันหรือกระแสไฟฟ้าที่ด้าน Primary จะส่งผลกระทบไปยังด้าน Secondary น้อยกว่าการต่อถึงกันโดยตรง เนื่องจากแกนเหล็กของ Transformer เกิดการอิ่มตัว



***รูปที่ 2.2** แสดงการติดตั้งแหล่งจ่ายไฟ AC*

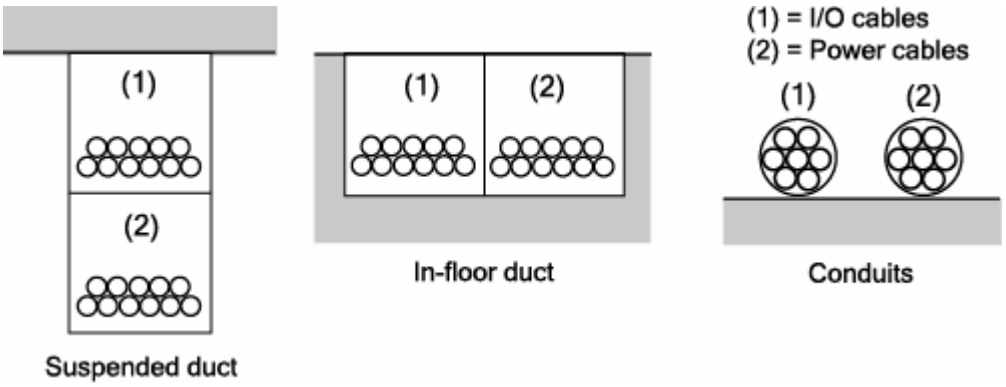
- DC Power Supply

ในกรณีที่ระบบ PLC ที่ใช้งานต้องการแหล่งจ่ายไฟ DC ซึ่งส่วนใหญ่แล้วจะใช้ไฟ 24VDC ดังนั้นจึงใช้ Switching Power Supply เพื่อจ่ายไฟให้กับ PLC เราไม่ขอแนะนำให้ใช้หม้อแปลงแล้วต่อกับไดโอดเพื่อทำเป็นวงจรจ่ายไฟ DC ให้กับ PLC เพราะไฟ DC ที่ได้จะไม่เรียบพอ ซึ่งอาจส่งผลต่อการทำงานของ PLC ได้ แต่มีผู้ใช้งานบางท่านที่ใช้อยู่เราขอแนะนำให้เปลี่ยนดีกว่า เพราะ PLC ราคาค่อนข้างแพงจะพังเพราะแหล่งจ่ายไฟที่ทำเองได้นะครับ

2.1.5 การเดินสายอย่างปลอดภัยและลดปัญหาสัญญาณรบกวน

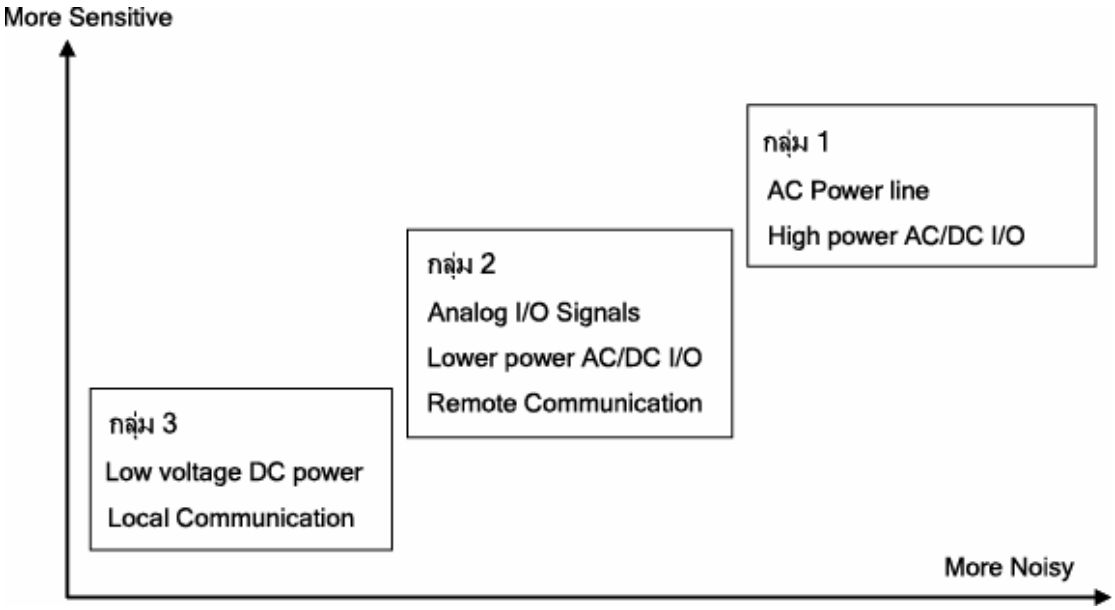
- การเดินสายสัญญาณอินพุต/เอาต์พุต

ควรเดินสายสัญญาณอินพุต/เอาต์พุตแยกออกจากสายไฟกำลังไม่ว่าจะเป็นภายในหรือภายนอกตู้ควบคุมโดยการแยกท่อหรือรางที่ใช้เดินสาย ทั้งนี้เพื่อป้องกัน Noise ที่เกิดจากสายไฟกำลังซึ่งอาจจะส่งผลให้อุปกรณ์อินพุต/เอาต์พุตและ PLC เกิดความเสียหายได้



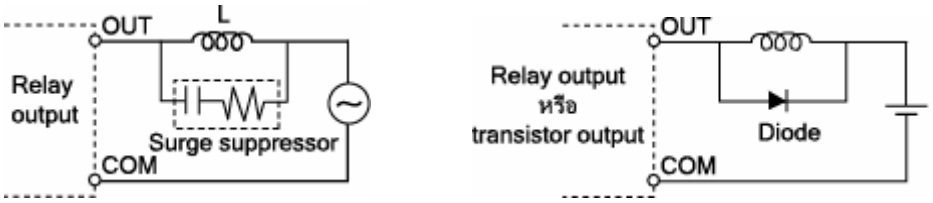
***รูปที่ 2.3** แสดงการสายไฟที่แยกท่อและราง*

การเดินสายแบบแบ่งกลุ่ม จะเป็นอีกวิธีหนึ่งที่ช่วยลดปัญหา Noise ที่เกิดขึ้นในระบบได้ ในรูปข้างล่างนี้แสดงการแบ่งกลุ่มหรือประเภทของสายตัวนำ โดยทั่วไปสายตัวนำกลุ่ม 1 ไม่ควรเดินร่วมกับสายตัวนำกลุ่มอื่น ควรเดินสายแยกท่อหรือแยกรางออกจากกัน



- การเดินสายเอาต์พุต Inductive

เอาต์พุต Inductive ที่พบบ่อยคือ Solenoid Valve โดยเฉพาะวาล์วไฮดรอริคที่ใช้กระแสไฟสูงเมื่อเริ่มทำงานเพื่อสั่งให้ลิ้นวาล์วเคลื่อนที่ ยิ่งใช้งานไปนานๆ ลิ้นวาล์วจะเริ่มติดขัด ทำให้เคลื่อนที่ไม่สะดวกเหมือนของใหม่จะยิ่งทำให้เอาต์พุตต้องรับกระแสกระชากที่สูงเป็นเวลานานกว่าปกติ เมื่อใช้งานไปนานๆ จะทำให้หน้าสัมผัสของรีเลย์เสียหายเร็วกว่าปกติ ควรติดตั้งวงจร Suppressor หรือ Diode ต่อคร่อมที่โหลด เพื่อช่วยลดกระแสกระชากที่เกิดขึ้นจากโหลด Inductive ดังแสดงในรูปข้างล่างนี้



รูปที่ 2.4 *วงจรลดปัญหา Surge*

คุณสมบัติของ Suppressor ที่ใช้ คือ R = 50 Ω และ C = 0.47μF, 200 V

คุณสมบัติของ Diode ที่ใช้ คือ Breakdown voltage: 3 เท่าของแรงดันโหลดต่ำสุด

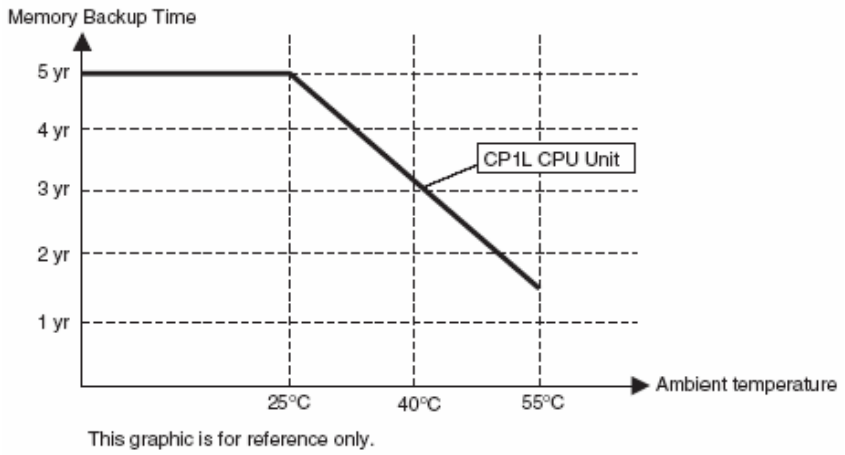
Mean rectification current: 1 A

2.1.6 การติดตั้งแบตเตอร์รี่

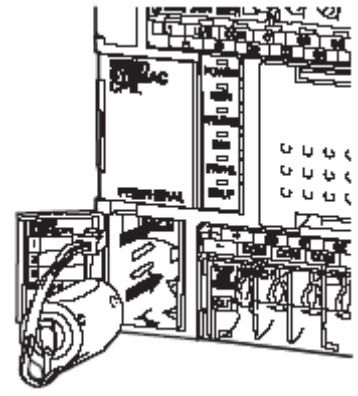
ส่วนใหญ่แล้ว PLC จะมีแบตเตอร์รี่ทำหน้าที่สำรองข้อมูล เช่น DM, Timer และ HR เป็นต้น แต่ไม่ได้ทำหน้าที่สำรองโปรแกรมแลดเดอร์ ดังนั้นเมื่อไฟดับ PLC จะเก็บค่าข้อมูลล่าสุดของ DM ไว้ ทำนองเดียวกันถ้าแบตเตอร์รี่หมดโปรแกรมจะไม่หายเพราะเก็บอยู่ใน Flash memory แต่ข้อมูลใน DM จะหาย

2.1.7 อายุการใช้งานของแบตเตอร์รี่

แบตเตอร์รี่จะมีอายุการใช้งานประมาณ 5 ปีเมื่อใช้งานที่อุณหภูมิ 25°C แต่ถ้าใช้ในสภาวะที่มีอุณหภูมิสูงกว่าปกติค่าอายุการใช้งานจะอ้างอิงตามกราฟข้างล่างนี้ ดังนั้นควรเปลี่ยนแบตเตอร์รี่ทุกๆ 5 ปี เพื่อป้องกันการเสื่อมคุณภาพของแบตเตอร์รี่ ซึ่งอาจทำให้เกิดกรดขึ้นและทำลายแผ่นวงจรอีเล็กทรอนิกส์ของ PLC ได้ ถ้าถึงขั้นนี้คงซ่อมไม่ไว้แน่ๆ ครับ



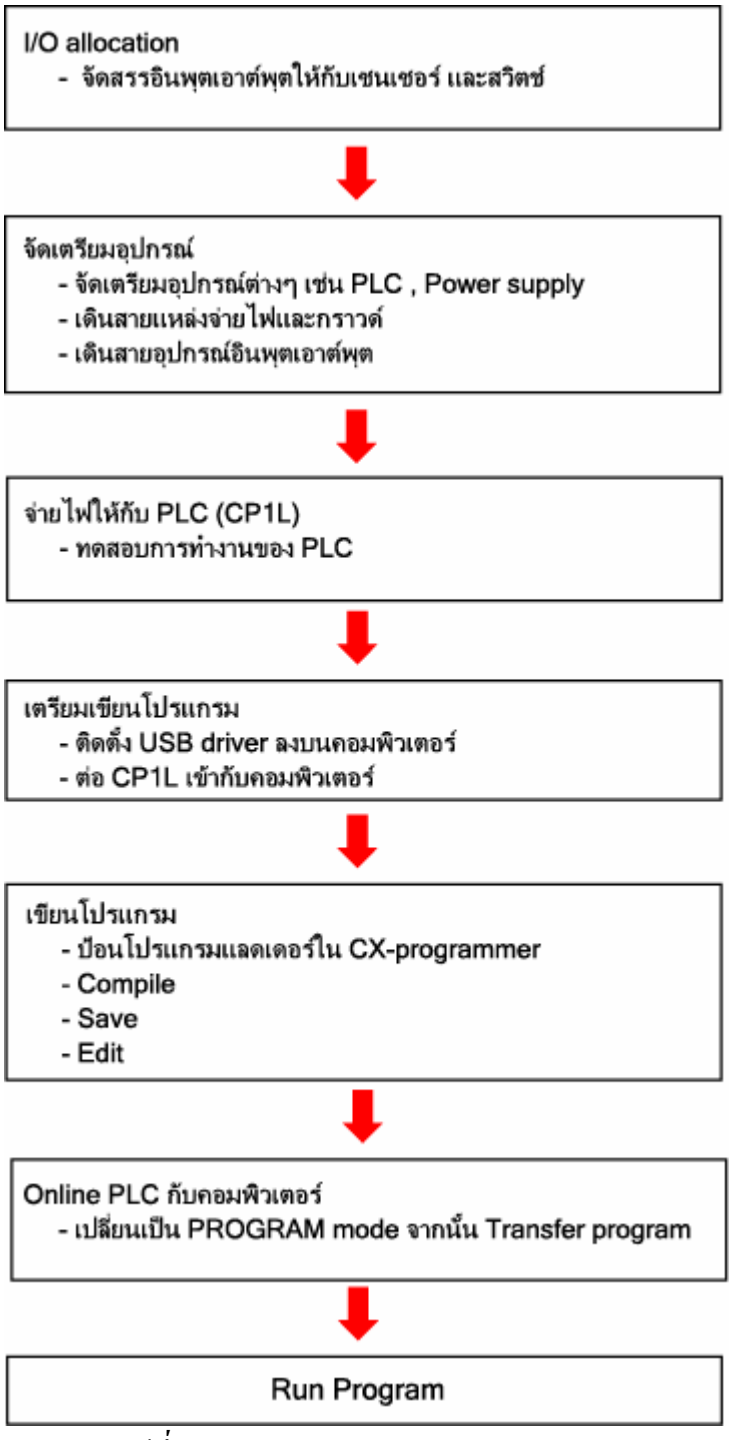
ขั้นตอนการเปลี่ยนแบตเตอร์รี่



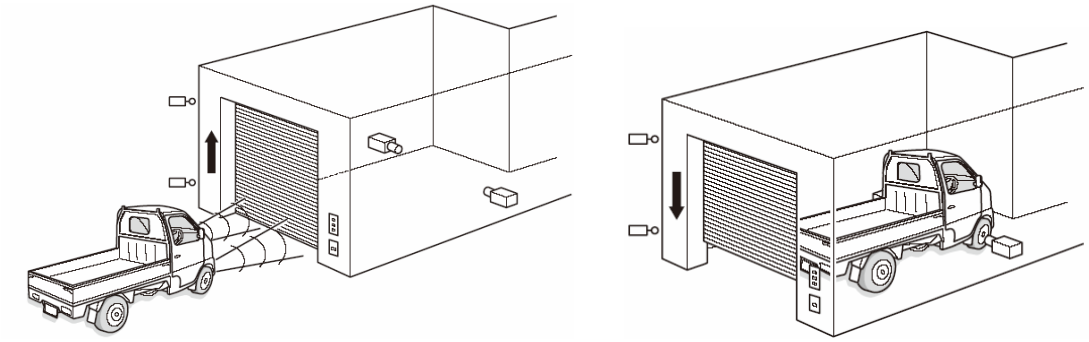
1. ปิดแหล่งจ่ายไฟของ PLC แต่ถ้า PLC ไม่มีไฟจ่ายให้เปิดไฟจ่ายให้ PLC อย่างน้อย 5 นาทีหลังจากนั้นให้ปิดไฟ
2. เปิดส่วนที่เก็บแบตเตอร์รี่ (ดูได้จากคู่มือ) จากนั้นให้ดึงก้อนแบตเตอร์รี่ออกด้วยความระมัดระวัง
3. ถอดคอนเน็กเตอร์ของแบตเตอร์รี่ออก
4. ต่อแบตเตอร์รี่ใหม่เข้าไปแล้วดันก้อนแบตเตอร์รี่เข้าที่แล้วปิดฝาครอบ

2.2 การออกแบบระบบ

เราขอยกตัวอย่างประกอบการออกแบบระบบควบคุมดังตัวอย่างต่อไปนี้ ซึ่งเป็นระบบควบคุมการเปิดปิดประตู(Shutter Control System) ของที่จอดรถยนต์ ส่วนรูปข้างล่างนี้แสดงให้เห็น Workflow ในการออกแบบระบบควบคุม



รูปที่ 2.5 *แสดง Workflow การออกแบบระบบ*



รูปที่ 2.6 *Shutter Control System*

2.3 ขั้นตอนการทำงาน

ขั้นตอนการทำงานสำหรับการใช้ CP1L เพื่อควบคุมการเปิดปิดประตูแสดงได้ตาม Workflow ในรูปที่ 2.5 ส่วนรูปที่ 2.6 แสดงการทำงานของ Shutter Control

การทำงานของระบบ Shutter Control

- เมื่อเซนเซอร์ตรวจจับได้ว่ามีแสงไฟหน้ารถเป็นเวลา 5 วินาที ประตูจะเปิดออก
- ประตูสามารถเปิด ปิด และ หยุดด้วยการกดสวิตช์
- เมื่อเซนเซอร์ตรวจจับว่ามีรถเข้าจอดในโรงรถแล้วประตูจะปิด
- เมื่อต้องการเอารถออกจากโรงจอดรถให้ใช้การกดสวิตช์

อุปกรณ์ต่างๆ ในระบบ

PLC

- CP1L (14 I/O)

อุปกรณ์การเขียนโปรแกรม

- CX-programmer
- Computer
- สาย USB

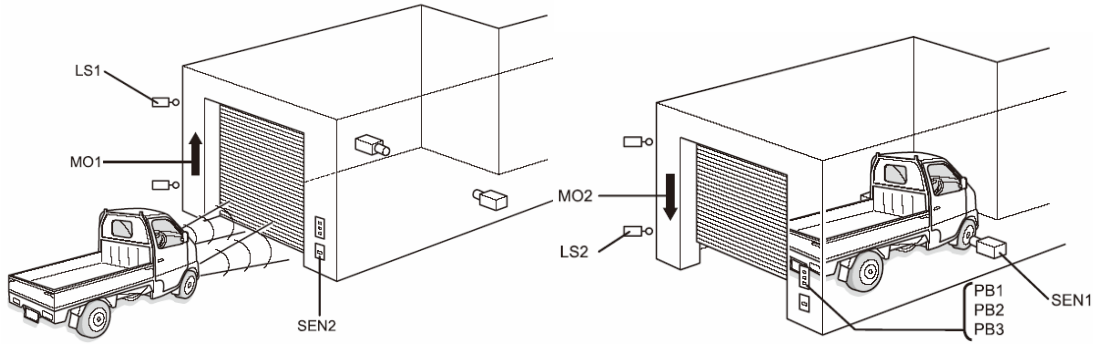
อินพุต

- Shutter OPEN button : PB1
- Shutter STOP button : PB2
- Shutter CLOSE button : PB3
- Car detection sensor : SEN1

- Headlight detection sensor : SEN2
- Limit switch, จะ ON เมื่อ shutter เปิดสุด : LS1
- Limit switch, จะ ON เมื่อ shutter ปิดสุด : LS2

เอาต์พุต

- คอนแทคสั่งให้ shutter motor เปิด : MO1
- คอนแทคสั่งให้ shutter motor ปิด : MO2



***รูปที่ 2.7** แสดงรายละเอียดอุปกรณ์*

<u>การจัดสรรอินพุต/เอาต์พุต สำหรับ Shutter control</u>

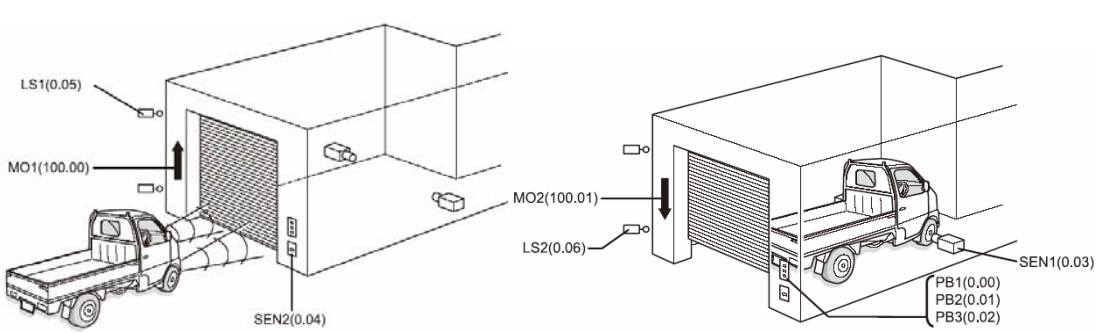
ในขั้นตอนนี้จะทำการจัดสรร I/O ให้กับอุปกรณ์ต่างๆ ดังนี้

อินพุต

Device	Contact	Address
OPEN button	PB1	0.00
STOP button	PB2	0.01
CLOSE button	PB3	0.02
Care detection sensor	SEN1	0.03
Light detection sensor	SEN2	0.04
Upper limit LS	LS1	0.05
Lower limit LS	LS2	0.06

เอาต์พุต

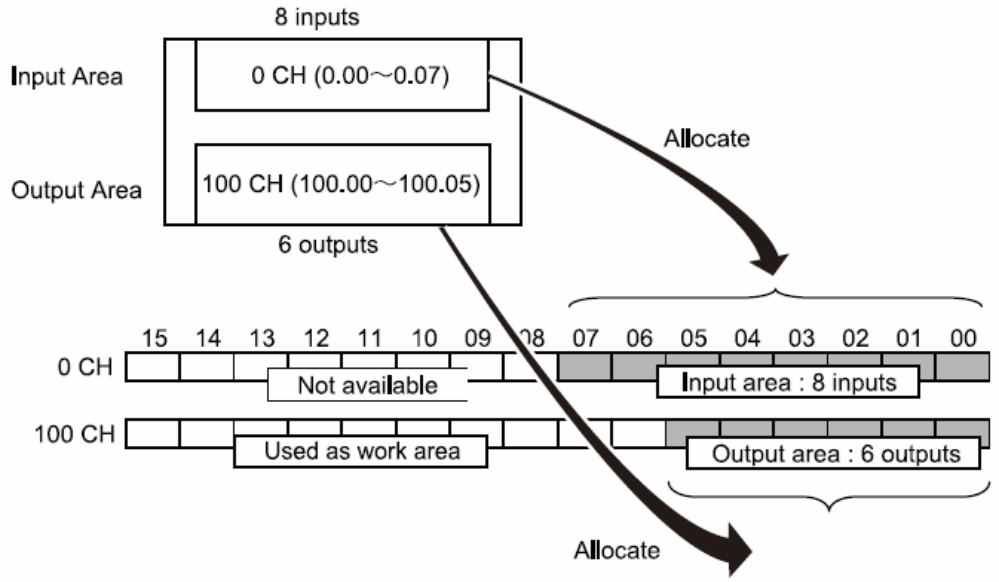
Device	Contact	Address
Escalation motor	MO1	100.00
De-escalation motor	MO2	100.01



รูปที่ 2.8 แสดงแอคเตรสที่จัดสรร

การจัดสรร I/O ของ CP1L (รุ่น 14 I/O)

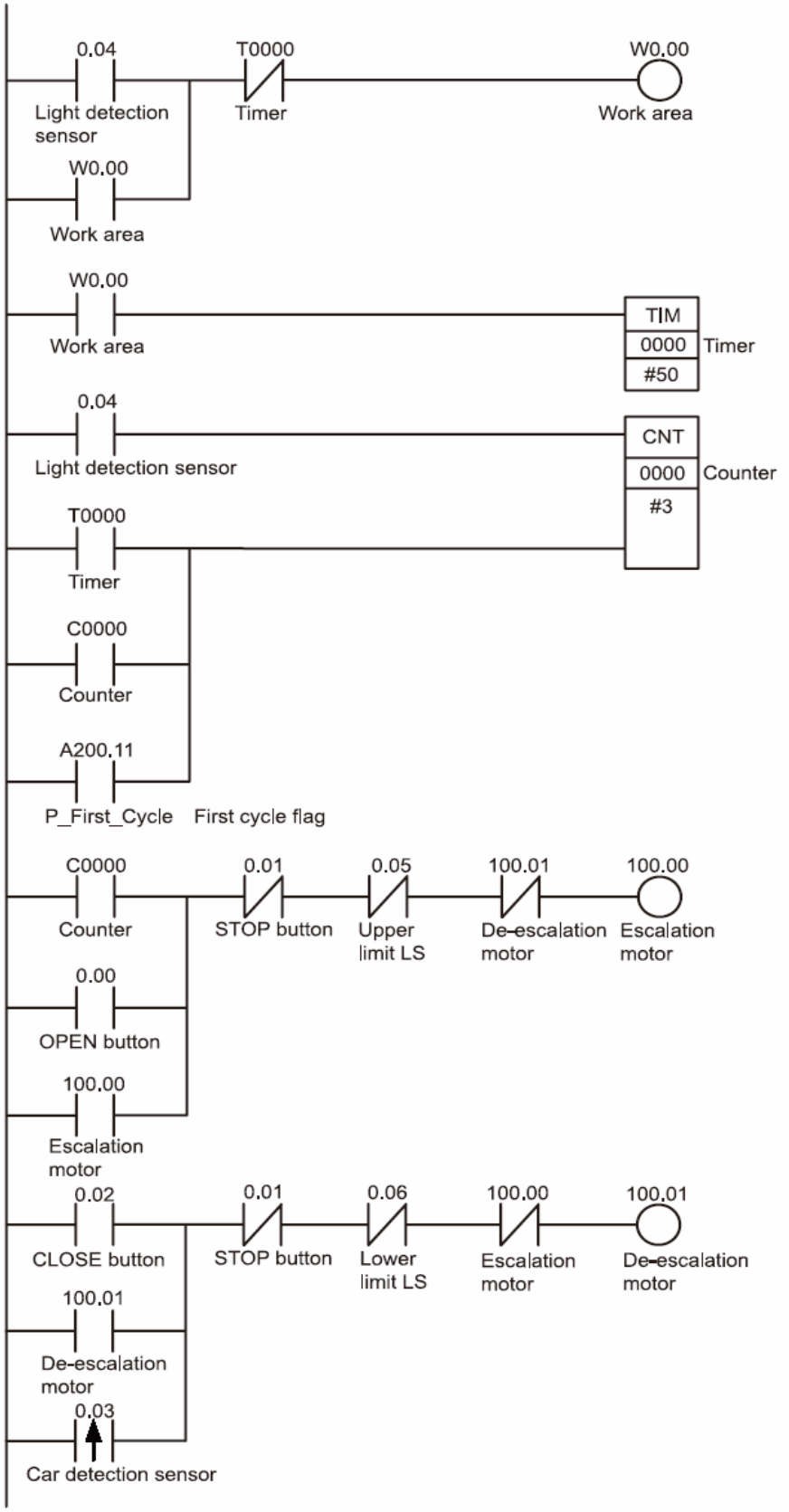
จากรูปข้างล่างนี้แสดงการจัดตำแหน่งอินพุตเอาต์พุตของ CP1L ขนาด 14 I/O ซึ่งอินพุตจะเริ่มต้น word ที่ '0' ส่วนเอาต์พุตจะเริ่มต้น word ที่ '100'



รูปที่ 2.9 การจัดตำแหน่ง I/O และเทอร์มินอล

การเขียนแลดเดอร์โปรแกรม

ต่อไปเป็นขั้นตอนที่สำคัญเพราะว่าต้องใช้ความคิดและความรู้พื้นฐานเกี่ยวกับคำสั่งของ PLC จึงจะสามารถเขียนโปรแกรมได้ ซึ่งจะกล่าวถึงวิธีการเขียนโปรแกรมและคำสั่งต่างๆ ของพีแอลซีในลำดับต่อไป



***รูปที่ 2.10** ตัวอย่างแลดเดอร์ไดอแกรม*

บทที่ 3
ความรู้พื้นฐานทางด้านดิจิตอล

เนื่องจากพื้นฐานของ PLC นั้นมาจากการทำงานของวงจรรีเลย์ ซึ่งมีสภาวะการทำงานแบบลอจิก (0 และ 1) ซึ่งก็คือค่าทางดิจิตอลนั่นเอง ดังนั้นก่อนการใช้งาน PLC จะต้องมีความรู้พื้นฐานในเรื่องของเลขฐานและวิธีการแปลงเลขฐานก่อน เพื่อจะเข้าใจการทำงานที่แท้จริงและสามารถใช้งาน PLC ได้ดี

3.1 ระบบเลขฐาน (Number System)

ระบบเลขฐาน จัดเป็นระบบตัวเลขที่ใช้งานอยู่ใน PLC ในบทนี้จะยกตัวอย่างเฉพาะการใช้งานระบบเลขฐานสอง, เลขฐานสิบ และเลขฐานสิบหกเท่านั้น

- **ระบบเลขฐานสอง (Binary: BIN)** มีตัวเลขที่ไม่ซ้ำกันอยู่ทั้งหมด 2 ตัว คือ **0** และ **1**
- **ระบบเลขฐาน BCD (Binary Code Decimal: BCD)** มีตัวเลขที่ไม่ซ้ำกันอยู่ทั้งหมด 10 ตัว คือ **0 1 2 3 4 5 6 7 8 9** หรือเรียกอีกอย่างหนึ่งว่า BCD code
- **ระบบเลขฐานสิบหก (Hexadecimal: HEX)** มีตัวเลขที่ไม่ซ้ำกันอยู่ทั้งหมด 16 ตัว คือ **0 1 2 3 4 5 6 7 8 9 A B C D E F** (ตัวอักษร 6 ตัว แทน ตัวเลข 10 –15)

ความสัมพันธ์ของเลข BIN, BCD และ HEX สามารถกำหนดให้เป็นตารางได้ดังตารางที่2.1

ตารางที่ 3.1 แสดงความสัมพันธ์ของเลขฐานต่างๆ

HEX	BCD	FOUR DIGIT BINARY			
		$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	8	1	0	0	0
9	9	1	0	0	1
A	-	1	0	1	0
B	-	1	0	1	1
C	-	1	1	0	0
D	-	1	1	0	1
E	-	1	1	1	0
F	-	1	1	1	1

หมายเหตุ **BIN** (Binary) = ระบบเลขฐานสอง

BCD (Binary Code Decimal) = ระบบเลขฐาน BCD

HEX (Hexadecimal) = ระบบเลขฐานสิบหก

ตัวอย่างที่ 3.1

การเปลี่ยนเลขฐานสิบหก(HEX)ให้เป็นเลขฐานสอง (BIN) โดยใช้ตารางที่ 2.1

วิธีทำ แปลงเลข 2F61 ฐานสิบหกให้เป็นเลขฐานสอง

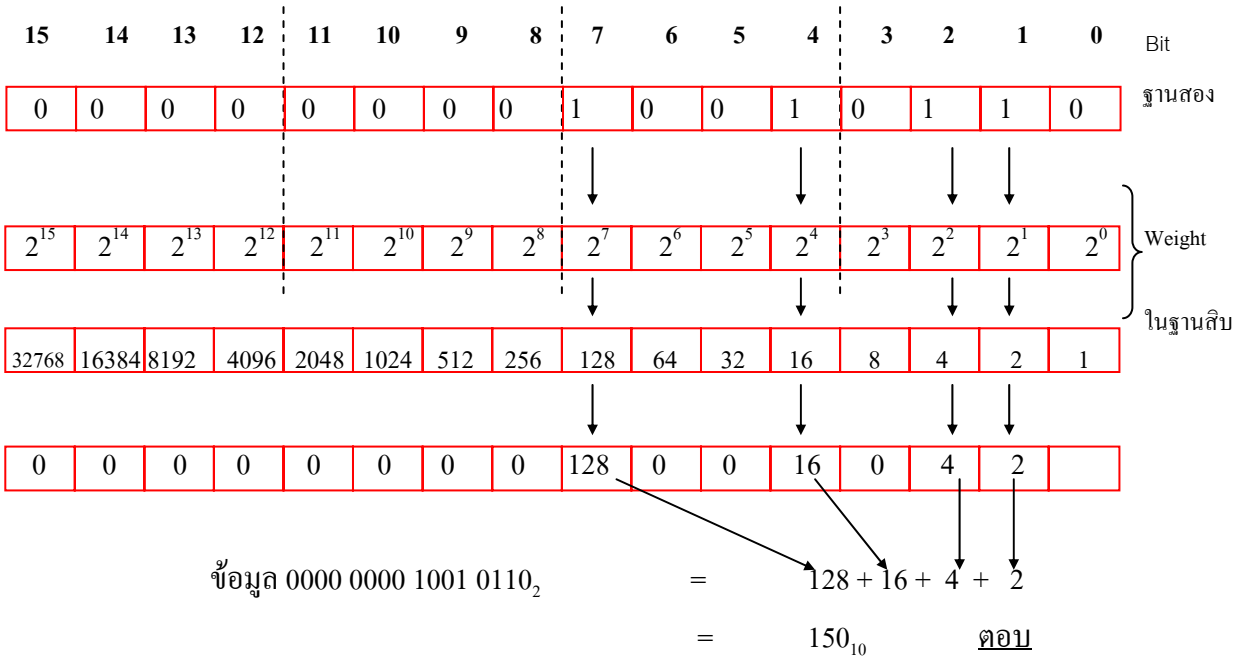
เลขฐานสิบหก	2	F	6	1
	↓	↓	↓	↓
เลขฐานสอง	0010	1111	0110	0001

3.2 การแปลงเลขฐาน

3.2.1 การแปลงเลขฐานสองให้เป็นเลขฐานสิบ

ตัวอย่างที่ 3.2 ข้อมูลซึ่งอยู่ในระบบเลขฐานสองขนาด 16 บิต มีค่า 0000 0000 1001 0110 ถ้าจะเปลี่ยนเป็นเลขฐานสิบ จะมีค่าเท่าใด

วิธีทำ



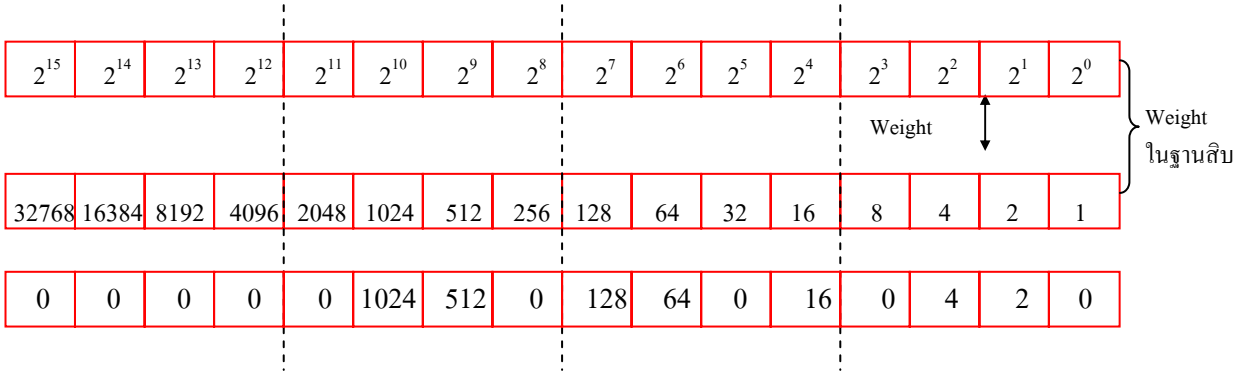
ข้อมูล $0000\ 0000\ 1001\ 0110_2$ = $128 + 16 + 4 + 2$

= 150_{10} ตอบ

3.2.2 การแปลงเลขฐานสิบให้เป็นเลขฐานสอง

ตัวอย่างที่ 3.3 ต้องการเปลี่ยนเลขฐานสิบ 1,750 ให้เป็นเลขฐานสอง ขนาด 16 บิต จะได้ค่าเท่าใด

วิธีทำ

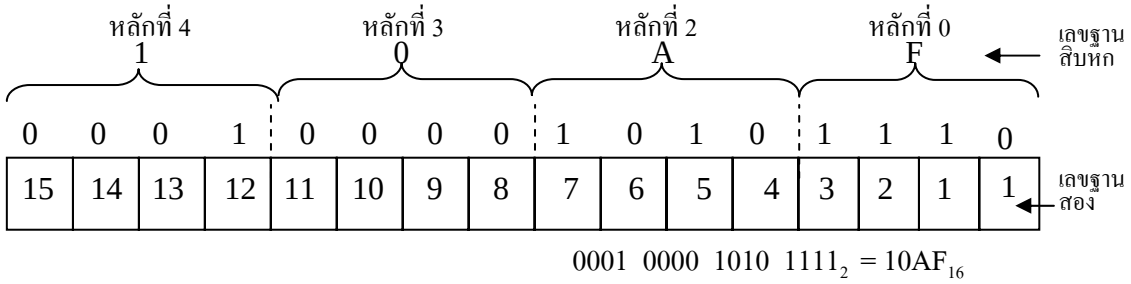


1750_{10} = $1024 + 512 + 128 + 64 + 16 + 4 + 2$

= $0000\ 0110\ 1101\ 0110_2$

3.2.3 การแปลงเลขฐานสองเป็นเลขฐานสิบหก และการแปลงเลขฐานสิบหกเป็นเลขฐานสอง

- การแปลงเลขฐานสองให้เป็นเลขฐานสิบหก จะกระทำได้โดยแปลงเลขฐานสองทีละ 4 บิตเป็นเลขฐานสิบหก 1 หลัก
- ถ้าต้องการแปลงเลขฐานสองขนาด 16 บิต ให้เป็นฐานสิบหก ต้องแบ่งเลขฐานสองเป็น 4 กลุ่ม กลุ่มละ4 บิต โดยแต่ละกลุ่มจะแทนได้ด้วยเลขฐานสิบหก 1 หลัก (1 ดิจิต)



- เช่นเดียวกับการแปลงเลขฐานสิบหกเป็นเลขฐานสอง เราจะแปลงเลขสิบหก 1 หลักเป็นเลขฐานสอง 4 บิต เช่น 0001 0000 1010 1111_2=$10AF_1$

3.3 การบวกและลบเลขฐาน

3.3.1 การบวกเลขฐานสอง เลขฐานสองมีความต่างกันของค่าน้ำหนัก (Weight) ของเลขฐานสองในแต่ละหลักที่อยู่ถัดกันเท่ากับ 2

	2^3	2^2	2^1	2^0
	0	1	1	0
+	0	1	0	1
	1	0	1	1

3.3.2 การบวกเลขฐานสิบหก เลขฐานสิบหกมีความต่างกันของค่าน้ำหนัก (Weight) ของเลขฐานสิบหกในแต่ละหลักที่อยู่ถัดกันเท่ากับ 16

	16^3	16^2	16^1	16^0
	0	4	B	6
+	0	C	6	4
	1	1	1	A

3.3.3 การลบเลขฐานสอง เลขฐานสองมีความต่างกันของค่าน้ำหนัก (Weight) ของเลข ฐานสองในแต่ละหลักที่อยู่ถัดกันเท่ากับ 2 ดังนั้นในการลบของเลขฐานสองแต่ละหลักนั้น หากตัวตั้งมีค่าน้อยกว่าตัวลบ จะต้องยืมค่าจากหลักถัดไปครั้งละ 2

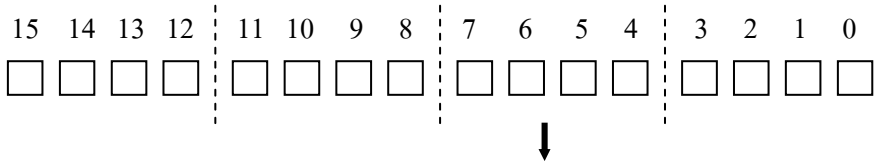
	2^3	2^2	2^1	2^0
	1	1	1	0
-	0	1	0	1
	1	0	0	1

3.3.4 การลบเลขฐานสิบหก เลขฐานสองมีความต่างกันของค่าน้ำหนัก (Weight) ของเลขฐานสองในแต่ละหลักที่อยู่ถัดกันเท่ากับ 16 ดังนั้นในการลบของเลขฐานสิบหกแต่ละหลักนั้น หากตัวตั้งมีค่าน้อยกว่าตัวลบ จะต้องยืมค่าจากหลักถัดไปครั้งละ16

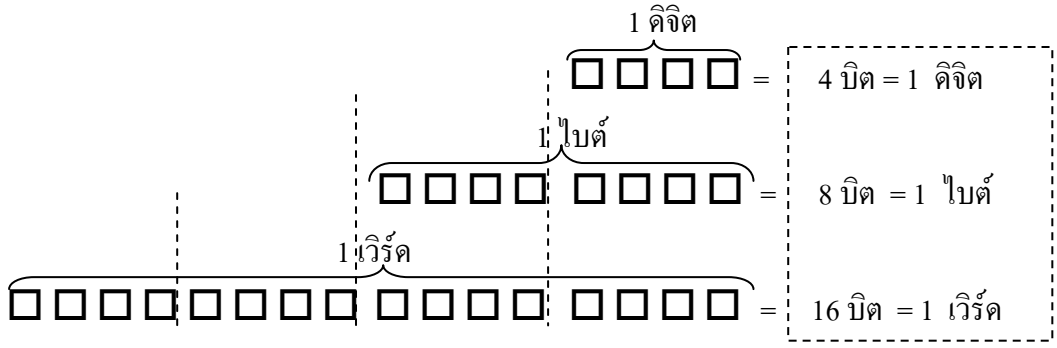
	16^3	16^2	16^1	16^0
	1	4	B	6
-	0	C	6	4
	0	8	5	2

3.4 ประเภทของข้อมูล

ข้อมูลภายใน PLC จะมีคำจำกัดความที่เรียกกันคือ บิต (Bit), ไบต์ (Byte), เวิร์ด (Word) หลักการเรียกและความหมายของแต่ละคำมีดังนี้



เรียกว่า ข้อมูลขนาด 16 บิต หรือ 1 เวิร์ดหรือ 1 Channel ก็ได้



ตัวอย่างที่ 3.4 ข้อมูลขนาด 256 กิโลบิต (kBits) จะสามารถเก็บข้อมูลได้กี่กิโลไบต์ (kBytes)

วิธีทำ

8 บิต = 1 ไบต์

256 กิโลบิต = $\frac{256 \times 1000}{8}$ = 32,000 ไบต์

= 32 กิโลไบต์

ตัวอย่างที่ 3.5 Data memory ขนาด 6 kWords ถ้าจะเปลี่ยนหน่วยเป็น kBytes จะได้เท่าไร

วิธีทำ

1 Word = 2 ไบต์

6 kWord = 2 x 6 x 1000 = 12,000 ไบต์

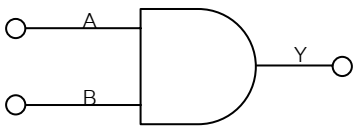
= 12 กิโลไบต์

3.5 หลักการพื้นฐานทางลอจิก

PLC ทำงานด้วยหลักการของ binary คือ เป็นอย่างใดอย่างหนึ่งใน 2 สถานะ เช่น สูงหรือต่ำ ปิดหรือเปิด, 0 หรือ 1

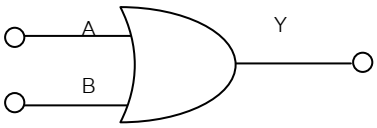
3.5.1 หลักการของ AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



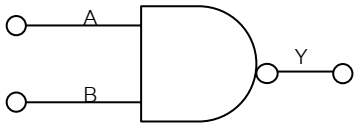
3.5.2 หลักการของ OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



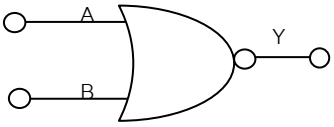
3.5.3 หลักการของ NAND

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



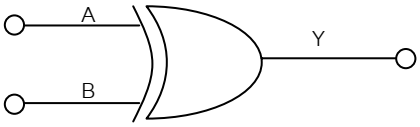
3.5.4 หลักการของ NOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



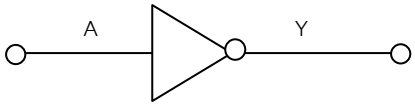
3.5.5 หลักการของ Exclusive OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



3.5.6 หลักการของ NOT

A	Y
0	1
1	0



บทที่ 4

การอ้างแอสเดรสของ PLC

การอ้างแอสเดรสของ PLC ถ้าเปรียบเทียบง่ายๆ คือ การเรียกชื่อตำแหน่งของอุปกรณ์อินพุต/เอาต์พุตที่นำมาต่อร่วมกับ PLC และเป็นการเรียกชื่อพื้นที่หน่วยความจำใน PLC นั่นเอง PLC แต่ละยี่ห้ออาจมีชื่อเรียกที่แตกต่างกันออกไป

4.1 โครงสร้างของข้อมูล

ในแต่ละบิตของ Word (จำนวน 16 บิต) จะบรรจุข้อมูลในเลขฐานสอง (0 หรือ 1) และเมื่อแยกบิตทั้ง 16 บิตออกเป็น 4 กลุ่มๆ ละ 4 บิต จะสามารถแสดงข้อมูลของแต่ละ Word หรือ Channel ในรูปของเลขฐานสิบหก 4 หลักหรือที่เรียกว่า 4 ดิจิต

3				2				1				0				ดิจิตที่
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	บิตที่
0	0	1	0	1	1	1	0	0	1	0	0	0	1	1	0	ข้อมูลในเลขฐานสอง

รูปที่ 4.1 *แสดงโครงสร้างของข้อมูลใน Word หรือ Channel ที่ 0*

4.2 การกำหนดเบอร์ของรีเลย์ (Relay) ใน PLC

โดยปกติแล้ว PLC ของออมรอน จะกำหนดพื้นที่รีเลย์ (Relay) เป็น word หรือ Channel ซึ่งแต่ละ Channel จะประกอบด้วยข้อมูลขนาด 16 บิต ในแต่ละบิตจะบรรจุข้อมูลในเลขฐานสองคือ เลข 1 ซึ่งแทนสถานะ ON และเลข 0 ซึ่งแทนสถานะ OFF ดังตัวอย่างข้างล่างนี้คือ เวิร์ด 000 ซึ่งประกอบด้วย 16 บิตจากบิตที่ 00 ถึง บิตที่15

การอ้างถึงรีเลย์แต่ละบิต เราจะแทนด้วยเลข 5 หลัก สามหลักแรกเป็นเวิร์ด (Word) หรือ Channel ส่วนสองหลักหลังเป็นบิต (Bit)

Internal Relay(IR) Word 000, หรือ Channel 000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	**บิตที่**
0	0	1	0	1	1	1	0	0	1	0	0	0	1	1	0	**ข้อมูลในเลขฐานสอง**

000.15 **000.06** **000.00**

สามหลักแรกหมายถึงเวิร์ด (เวิร์ดที่ 000)

สองหลักหลังหมายถึงบิต (บิตที่ 10 ของเวิร์ด 000)

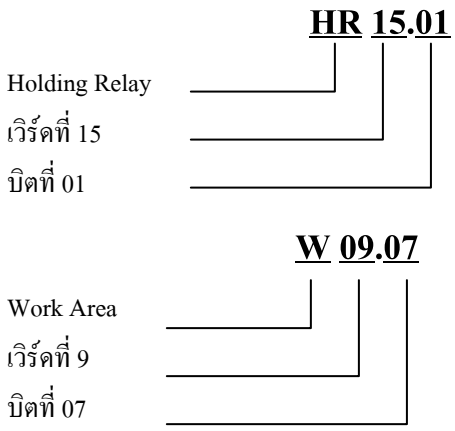
***รูปที่ 4.2** แสดงการกำหนดเบอร์รีเลย์ของ PLC*

หมายเหตุ

1. ใน PLC บางรุ่นใช้เป็นตัวเลข 6 หลักโดย 4 หลักแรกเป็นหมายเลข Channel ส่วน 2 หลักหลังเป็นหมายเลขบิตเช่น 0000.00 เป็นต้น
2. ในกรณีที่ท่านใช้งาน **Programming Console** จะไม่ปรากฏจุดทศนิยมระหว่างหมายเลขเวิร์ดและบิตเช่น เวิร์ดที่ 0 บิตที่ 10 จะแสดงเป็น **00010**
3. ในกรณีที่ท่านใช้งาน **CX-Programmer** หรือซอต์ฟแวร์อื่น จะแสดงจุดทศนิยมระหว่างหมายเลขเวิร์ดและบิตเช่นเวิร์ดที่ 0 บิตที่ 10 จะแสดงเป็น **000.10**

ที่กล่าวถึงข้างต้นเป็นการอ้างถึงแต่ละบิตของรีเลย์ในส่วนที่เรียกว่า CIO หรือ Internal Relay (IR) ซึ่งประกอบด้วย Input Area, Output Area และ Work Area สำหรับ PLC รุ่นใหม่ เช่น CP1 จะเพิ่มหน่วยความจำ Work Area ขึ้นมาและมีตัวอักษรนำหน้า W เช่น W0.00 ซึ่งสามารถนำไปใช้เป็น Internal Relay ในการเขียนโปรแกรมได้

ในกรณีของรีเลย์ชนิดอื่นๆ ก็มีการกำหนดเบอร์ในลักษณะเดียวกัน เช่น Holding Relay, Link Relay ดังตัวอย่างต่อไปนี้



นอกจากพื้นที่หน่วยความจำที่กล่าวถึงแล้ว PLC ยังแบ่งพื้นที่หน่วยความจำออกเป็นส่วนย่อยๆ อีกหลายส่วน สามารถแสดงรายละเอียดให้เห็นดังตารางที่ 4.1

4.3 ตารางแสดงข้อกำหนดของพื้นที่ใช้งานของ PLC

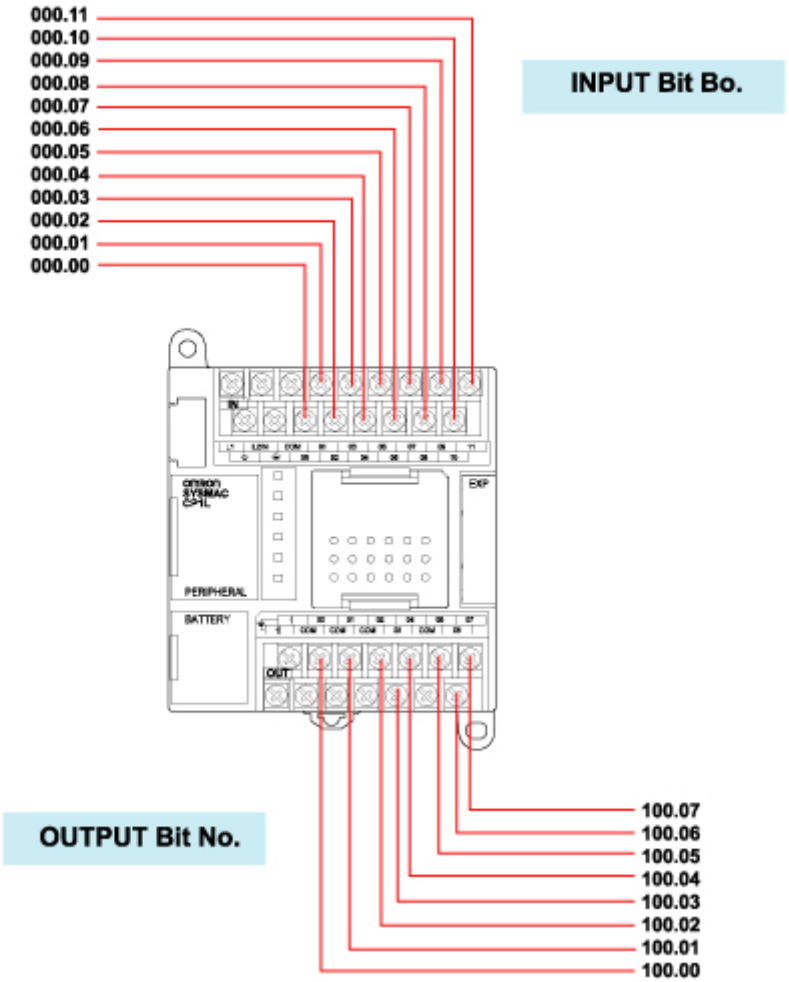
ตารางที่ 4.1 พื้นที่หน่วยความจำของ PLC (อ้างอิง CP1L)

Data area		Words	Bits	Function
CIO area	Input area	CIO 000 ถึง CIO 099 (100 words)	0.00 ถึง 099.15 (1,600 bits)	บิตเหล่านี้ถูกใช้ต่อไปยังขั้ว อินพุตเอาต์พุตภายนอก แต่ขึ้นอยู่กับรุ่นของ PLC ด้วยว่าใช้กี่บิต ดังนั้นบิตที่เหลือจะใช้งานเป็น Work Bits หรือ Internal relay ได้
	Output area	CIO 100 ถึง CIO 199 (100 words)	100.00 ถึง 199.15 (1,600 bits)	
	Work area	3800 ถึง 6143 (2,344 words)	3800.00 ถึง 6143.15 (37,504 bits)	Work Bits สามารถนำไปใช้ได้อย่างอิสระภายในโปรแกรม
Work area		W000 ถึง W511 (512 words)	W0.00 ถึง W511.15 (8192 bits)	บิตเหล่านี้ใช้เป็น Work bit
DM (Data Memory)		D00000 ถึง D32767		ใช้เก็บข้อมูล
HR area (Holding Relay)		H000 ถึง H511 (512 words)	H0.00 ถึง H511.15 (8192 bits)	บิตเหล่านี้ใช้เก็บข้อมูลและสถานะ ON/OFF ไว้ได้เมื่อเกิดไฟดับ
AR area (Auxiliary Relay)		A000 ถึง A959 (960 words)	A0.00 ถึง A959.15 (15,360 bits)	บิตเหล่านี้เป็นบิตพิเศษภายในซึ่งมีหน้าที่เฉพาะอย่างเช่นใช้เป็น Flags หรือ Control bits
Timer		T000 ถึง T4095		ใช้เป็น Timers
Counter		C000 ถึง C4095		ใช้เป็น Counters

4.4 การระบุตำแหน่งอินพุต/เอาต์พุตของ PLC

4.4.1 การระบุตำแหน่งอินพุต/เอาต์พุตของ PLC ชนิดบล็อก (ยกตัวอย่างรุ่น CP1L)

สำหรับ PLC แบบ Block นั้นตำแหน่งของอินพุต/เอาต์พุตนั้นจะแสดงหมายเลขไว้ที่ PLC อยู่แล้วยกตัวอย่างให้เห็นดังต่อไปนี้



***รูปที่ 4.3** แสดงตำแหน่งของอินพุต/เอาต์พุตแต่ละขั้ว*

จากรูปที่ 4.3 แสดงตำแหน่งของอินพุต/เอาต์พุตของ PLC รุ่น CP1L-L20DR ซึ่งมีจำนวน 20 จุด เป็นพื้นที่หน่วยความจำในส่วนของ CIO หรือ IR

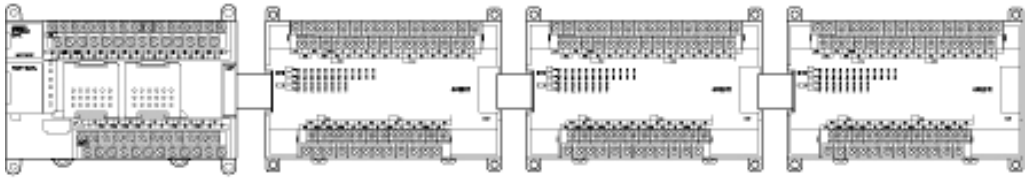
เช่นในการเขียนโปรแกรม ถ้าต้องการใช้งานอินพุตที่ต่ออยู่กับอินพุตของ PLC บิตที่ 0 เวลาอ้างตำแหน่งจะกำหนดเป็น 000.00 หมายถึงใช้ CIO เวิร์ดที่ 0 บิตที่ 0 (วิธีการกำหนดตำแหน่งดูได้จากหัวข้อ 4.2 การกำหนดเบอร์รีเลย์ของ CP1L)

สำหรับ CP1L รุ่นอื่น สามารถดูได้จากตารางที่ 4.2 ซึ่งจะแสดงตำแหน่งอินพุต/เอาต์พุตที่ต่อร่วมกับ Expansion I/O Unit

ตารางที่ 4.2 ตารางแสดงจำนวนและตำแหน่งอินพุต/เอาต์พุตบิตของ PLC รุ่น CP1L แต่ละรุ่น เมื่อใช้ร่วมกับ Expansion I/O Units (CP1W-20EDR1)

รุ่น	CP1L-L20		CP1L-M30		CP1L-M40		ชื่อรุ่น Expansion
จำนวน I/O บนตัว CPU	IN	OUT	IN	OUT	IN	OUT	-
	12 Points 000.00 - 000.11	8 Points 100.00 - 100.11	18 Points 000.00 - 000.11	12 Points 100.00 - 100.07	24 Points 000.00 - 000.11	16 Points 100.00 - 100.07	-
			001.00 - 001.05	101.00 - 101.03	001.00 - 001.11	101.00 - 101.07	-
Expansion I/O Unit ตัวที่ 1	IN	OUT	IN	OUT	IN	OUT	CP1W-20EDR1 CPM1A-20EDT1
	12 Points 002.00 - 002.11	8 Points 102.00 - 102.07	12 Points 002.00 - 002.11	8 Points 102.00 - 102.07	12 Points 002.00 - 002.11	8 Points 102.00 - 102.07	
Expansion I/O Unit ตัวที่ 2	ไม่สามารถต่อได้		IN	OUT	IN	OUT	CP1W-20EDR1 CPM1A-20EDT1
			12 Points 003.00 - 003.11	8 Points 103.00 - 103.07	12 Points 003.00 - 003.11	8 Points 103.00 - 103.07	
Expansion I/O Unit ตัวที่ 3	ไม่สามารถต่อได้		IN	OUT	IN	OUT	CP1W-20EDR1 CPM1A-20EDT1
			12 Points 004.00 - 004.11	8 Points 104.00 - 104.07	12 Points 004.00 - 004.11	8 Points 104.00 - 104.07	

CP1L **Expansion ตัวที่1** **Expansion ตัวที่2** **Expansion ตัวที่3**

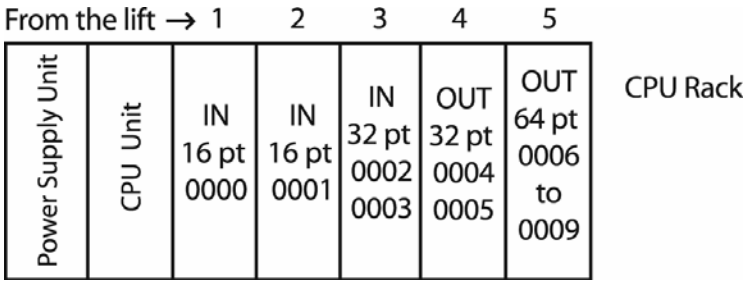


รูปที่ 4.4 *แสดงการเชื่อมต่อระหว่าง CP1L กับ Expansion I/O Unit*

4.4.2 การระบุตำแหน่งอินพุต/เอาต์พุตของ PLC ชนิดโมดูล

สำหรับ PLC ชนิดโมดูล ของ OMRON นั้นแบ่งเป็นหลายรุ่นได้แก่รุ่น CS1 และ CJ1 ซึ่งทั้งสองรุ่นมีการอ้างแอดเดรสเหมือนกัน จะขอยกตัวอย่างการอ้างแอสเดรสของแต่ละรุ่นดังนี้

- **การอ้างแอสเดรสของ PLC รุ่น CS1/CJ1**



***รูปที่ 4.5** ตัวอย่างการอ้างแอสเดรสของ PLC รุ่น CS1/CJ1*

การอ้างแอสเดรสของ PLC รุ่น CS1/CJ1จะอ้างตำแหน่งตามการติดตั้งยูนิตอินพุต/เอาต์พุตนั้น โดยไม่สนใจว่าจะติดตั้งที่ตำแหน่งใด การนับแอสเดรสจะนับเรียงต่อกันไปเรื่อยๆ ดังรูป*

หมายเหตุ *การอ้างแอสเดรสของ PLC รุ่น CS1/CJ1 นั้นกล่าวถึงเฉพาะ Standard I/O Unit เท่านั้น นอกจากนี้ยังมี Special I/O Unit อื่นๆ ซึ่งมีการกำหนดแอสเสรสแตกต่างกันออกไป สามารถศึกษาเพิ่มเติมได้จาก PLC รุ่นนั้นๆ

สำหรับเนื้อหาในบทต่อไปเราจะกล่าวถึงคำสั่งพื้นฐานต่างๆ ที่มีใช้งานในการเขียนโปรแกรม PLC

บทที่ 5
หลักการเขียนแลดเดอร์ไดอะแกรม และคำสั่งพื้นฐาน

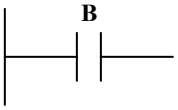
แลดเดอร์ไดอะแกรม (Ladder Diagram) จัดเป็นภาษาสัญลักษณ์ที่เทียบเคียงมาจากวงจรรีเลย์ สามารถดูตามโครงสร้างแล้วเข้าใจการทำงานได้ แต่เวลาที่ PLC ทำงานจะอาศัยชุดคำสั่ง (Instructions) ทำงานโดยวิธีการเขียนลงในส่วนหน่วยความจำข้อมูล ในหน่วยความจำนั้นจะจัดเก็บเป็นรหัส (Code) ไม่สามารถจัดเก็บในลักษณะของ Ladder Diagram ได้โดยตรง

ดังนั้นผู้ใช้จึงจำเป็นต้องเข้าใจชุดคำสั่งเพราะชุดคำสั่งนั้นแปลงภาษามาจาก Ladder Diagram นั่นเอง

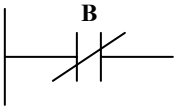
5.1 กลุ่มคำสั่งพื้นฐาน (Ladder Instruction & Output Control)

5.1.1 การใช้คำสั่ง LOAD (LD), LOAD NOT (LD NOT)

LOAD-**LD**



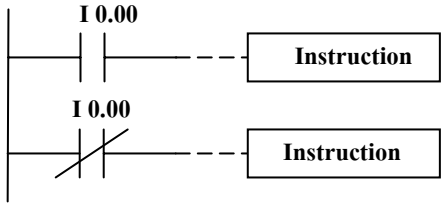
LOAD NOT-**LD NOT**



B : BIT
CIO,A,T,C,IR,DR,D,TK,H,W

พื้นที่หน่วยความจำที่ใช้กับคำสั่งได้

ตัวอย่างที่ 5.1 ชุดคำสั่งและการเขียน Ladder Diagram คำสั่ง LD และ LD NOT

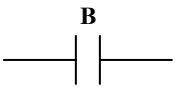


Address	Instruction	Operands
00000	LD	0.00
00001	Instruction	
00002	LD NOT	0.00
00003	Instruction	

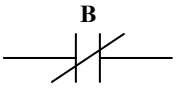
การเขียน Mnemonic อ้างอิงมาจาก PLC รุ่น CP1L เขียนใน CX Programmer Version 7 และไม่สามารถเขียนใน Programming Console ได้

5.1.2 การใช้คำสั่ง AND, AND NOT

AND-**AND**



AND NOT-**AND NOT**

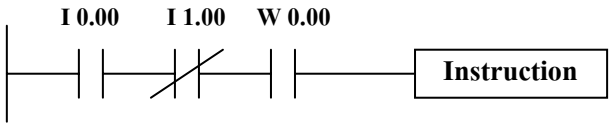


B : BIT
CIO,A,T,C,IR,DR,D,TK,H,W

พื้นที่หน่วยความจำที่ใช้กับคำสั่งได้

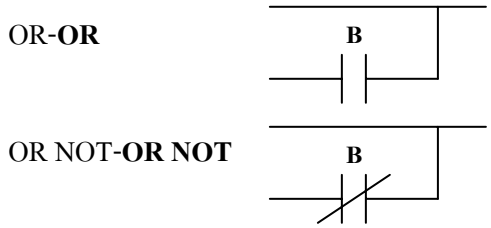
หลักการเขียน Ladder Diagram และคำสั่งพื้นฐาน อ้างอิงมาจาก PLC รุ่น CP1L

ตัวอย่างที่ 5.2 ชุดคำสั่งและการเขียน Ladder Diagram คำสั่ง AND, AND NOT



ddress	Instruction	Operands
00000	LD	0.00
00001	AND NOT	1.00
00002	AND	W 0.00
00003	Instruction	

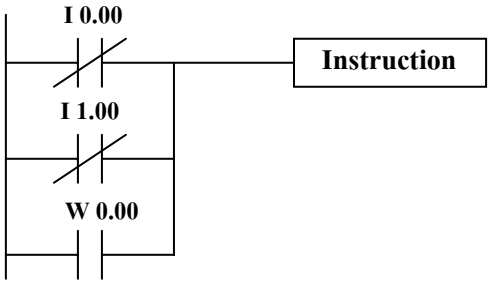
5.1.3 การใช้คำสั่ง OR, OR NOT



B : BIT
CIO,A,T,C,IR,DR,D,TK,H,W

พื้นที่หน่วยความจำที่ใช้คำสั่งได้

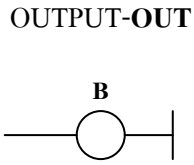
ตัวอย่างที่ 4.3 ชุดคำสั่งและการเขียน Ladder Diagram คำสั่ง OR, OR NOT



Address	Instruction	Operands
00000	LD NOT	0.00
00001	OR NOT	1.00
00002	OR	W 0.00
00003	Instruction	

5.1.4 การใช้คำสั่ง OUT, OUT NOT

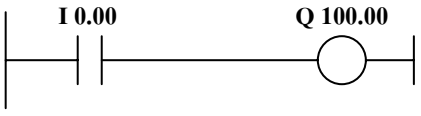
เป็นคำสั่งที่สั่งขับให้ OUTPUT ภายนอกทำงานหรือไม่ทำงานตามคำสั่ง



B : BIT
CIO,A,T,C,IR,DR,D,TK,H,W

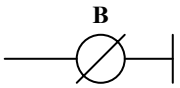
พื้นที่หน่วยความจำที่ใช้กับคำสั่งได้

ตัวอย่างที่ 5.4 รูปแบบชุดคำสั่งจาก Ladder Diagram



Address	Instruction	Operands
00000	LD	0.00
00001	OUT	100.00

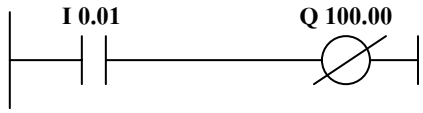
OUTPUT NOT-**OUT NOT:** การทำงานของคำสั่งนี้จะตรงกันข้ามกับคำสั่ง OUT



B : BIT
CIO,A,T,C,IR,DR,D,TK,H,W

พื้นที่หน่วยความจำที่ใช้กับคำสั่งได้

ตัวอย่างที่ 5.5 จงเขียนชุดคำสั่งจาก Ladder Diagram



Address	Instruction	Operands
00000	LD	0.01
00001	OUT NOT	100.00

5.1.5 การใช้คำสั่ง END (END 01)

เมื่อสิ้นสุดการเขียนโปรแกรมแล้วจะต้องจบด้วยคำสั่ง END(01) เสมอ ถ้าไม่มีคำสั่งนี้ เมื่อผู้ใช้สั่ง Run โปรแกรมที่เขียนขึ้น PLC จะเกิด Error โดยสังเกตที่ PLC ไฟ Error/Alarm สีแดงจะติดค้าง นั่นแสดงว่าไม่มีคำสั่ง END (01)

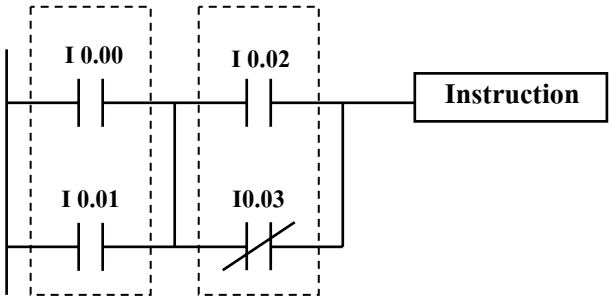
ในกรณีนี้โปรแกรมจะไม่สามารถ RUN ได้ เพราะฉะนั้นเมื่อเขียนโปรแกรมจบทุกครั้งควรใส่คำสั่ง END(01) ด้วย การเขียนโปรแกรมด้วย CX-Programmer ไม่จำเป็นต้องใส่คำสั่ง END(01) เพราะว่าซอฟต์แวร์จะใส่ให้เองอัตโนมัติ

5.1.6 การใช้คำสั่ง AND LOAD (AND LD), OR LOAD (OR LD)

คำสั่งทั้งสองจะทำหน้าที่เชื่อมต่อกลุ่มแลดเดอร์ไดอะแกรม (Ladder Diagram) ในกรณีที่ต่ออนุกรม หรือขนานกันมากกว่า 1หน้าสัมผัส ซึ่งการใช้คำสั่ง AND หรือ OR นั้น จะกระทำทีละ 1 หน้าสัมผัสเท่านั้นจึงต้องใช้ AND LD หรือ OR LD

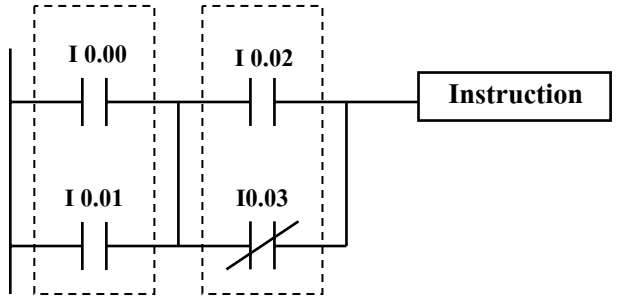
ในการเขียนแลดเดอร์ไดอะแกรม (Ladder Diagram) ด้วยซอฟต์แวร์นั้น ไม่มีสัญลักษณ์ของ AND LD และ OR LD ดังนั้นคำสั่งทั้ง 2 คำสั่งนี้ต้องเขียนเป็น Mnemonic หรือ Instruction list เท่านั้น

ตัวอย่างที่ 5.6 ชุดคำสั่งในรูปการเชื่อมแบบอนุกรมจะใช้คำสั่ง AND LD



Address	Instruction	Operands
00000	LD	0.00
00001	OR	0.01
00002	LD	0.02
00003	OR NOT	0.03
00004	AND LD	

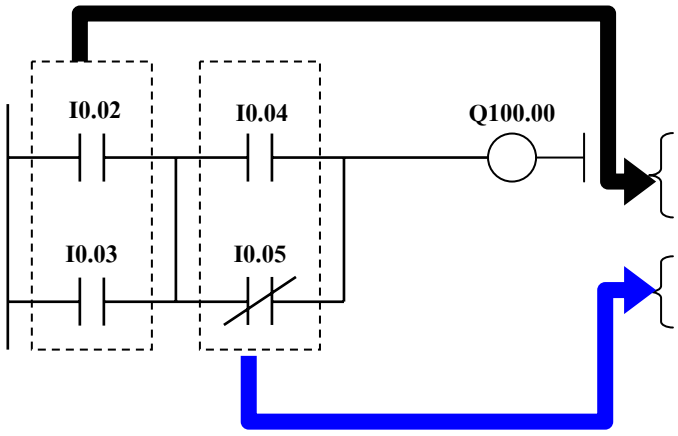
ตัวอย่างที่ 5.7 ชุดคำสั่งในรูปการเชื่อมแบบขนานจะใช้คำสั่ง OR LD



Address	Instruction	Operands
00000	LD	0.00
00001	AND	0.02
00002	LD	0.01
00003	AND NOT	0.03
00004	OR LD	- - -

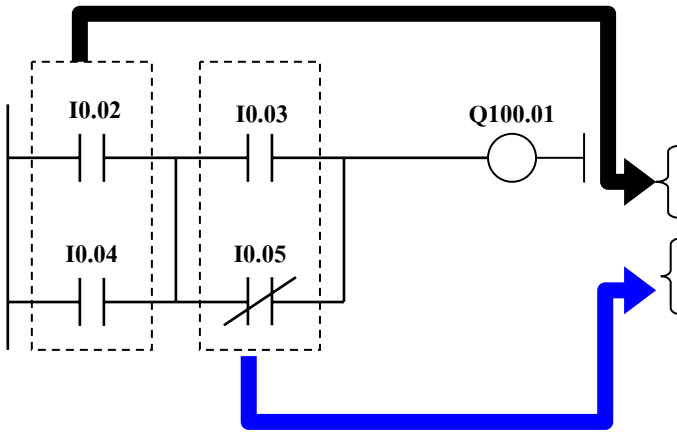
ตัวอย่างที่ 5.8 การเขียนโปรแกรมโดยใช้คำสั่ง AND LD และ OR LD

- **AND LD** คือ การเชื่อมโปรแกรม 2 block ในแบบอนุกรม



Address	Instruction	Data
00000	LD	0.02
00001	OR	0.03
00002	LD	0.04
00003	OR NOT	0.05
00004	AND LD	- - -
00005	OUT	100.00

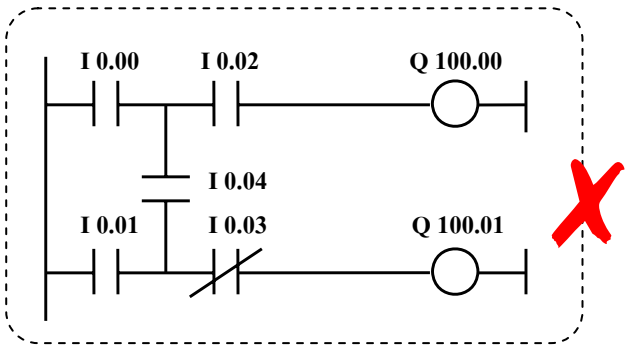
- **OR LD** คือ การเชื่อมโปรแกรม 2 block ในแบบขนาน



Address	Instruction	Data
00000	LD	0.02
00001	AND NOT	0.03
00002	LD	0.04
00003	AND	0.05
00004	OR LD	- - -
00005	OUT	100.01

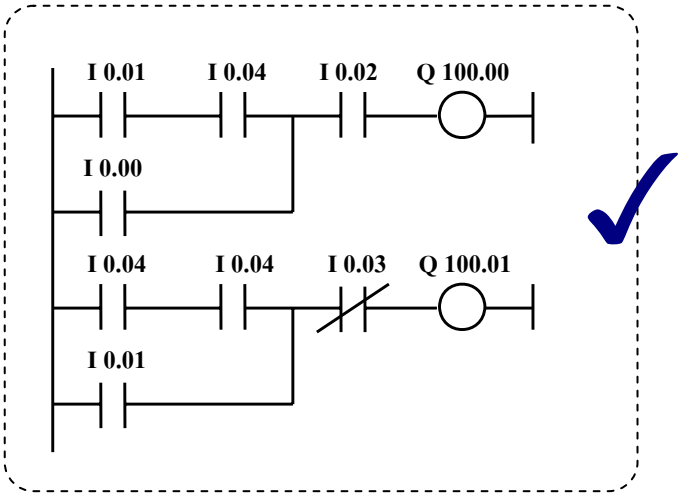
5.2 ข้อกำหนดในการเขียนแลดเดอร์ไดอะแกรม (Ladder Diagram)

5.2.1 จากแลดเดอร์ไดอะแกรม (Ladder Diagram) ข้างล่าง จะไม่สามารถเขียนโปรแกรมได้ จำเป็นต้องแปลงชุด Ladder Diagram ก่อน



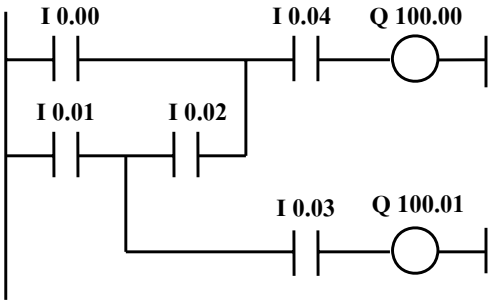
แลดเดอร์ไดอะแกรม (Ladder Diagram) ที่ผิด

สามารถเขียนใหม่ได้ และวงจรทำงานเหมือนเดิม คือ



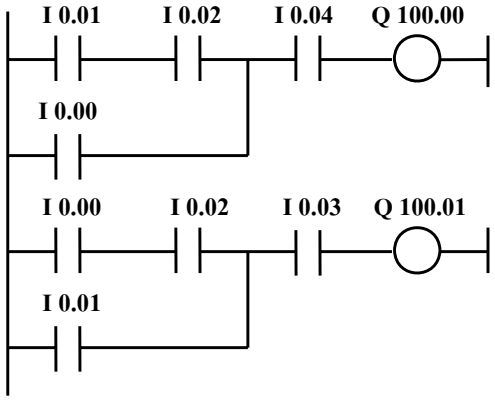
แลดเดอร์ไดอะแกรม (Ladder Diagram) ที่ถูก

5.2.2 สำหรับแลดเดอร์ไดอะแกรม (Ladder Diagram) จะพิจารณาการทำงานจากซ้ายไปขวาเท่านั้น ดังตัวอย่างเช่น



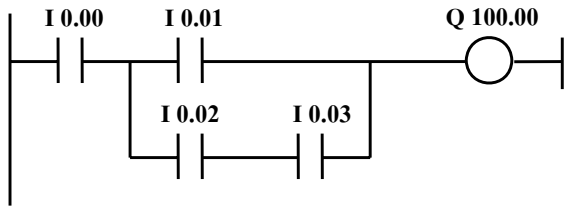
แลดเดอร์ไดอะแกรม (Ladder Diagram) A

จากแลดเดอร์ไดอะแกรม (Ladder Diagram) A ถ้าหน้าสัมผัส 0.00, 0.02 และ 0.03 มีสภาวะ “ON” ก็ไม่สามารถทำให้ เอาต์พุต 100.01 นั้น “ON” ได้เลย ดังนั้นผู้ใช้จะต้องทำการจัดโปรแกรมเสียใหม่เพื่อให้การทำงานกระทำจากซ้ายไปขวาดังรูปแลดเดอร์ไดอะแกรม B



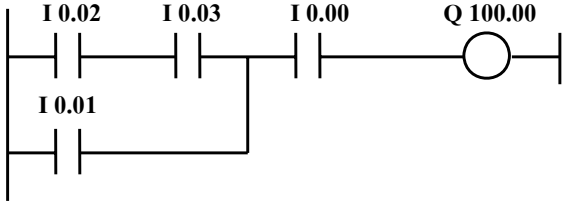
แลดเดอร์ไดอะแกรม (Ladder Diagram) B

5.2.3 จำนวนหน้าคอนแทคทั้ง NO และ NC ของอินพุต/เอาต์พุต, รีเลย์และไทม์เมอร์ (TIM)/เคาน์เตอร์ (CNT) จะนำมาเขียนโปรแกรมเป็นจำนวนเท่าใดก็ได้ตามความประสงค์ของผู้ใช้ แต่ถึงอย่างไรก็ตามการเขียนโปรแกรมที่ดีจะต้องพยายามประหยัดขนาดของโปรแกรมให้มากที่สุดเท่าที่จะสามารถทำได้ ซึ่งจะเปรียบเทียบให้เห็นในแลดเดอร์ไดอะแกรม (Ladder Diagram) A และแลดเดอร์ไดอะแกรม (Ladder Diagram) B จะสังเกตเห็นได้ว่าการเขียนในแลดเดอร์ไดอะแกรม (Ladder Diagram) B จะประหยัดคำสั่งได้ 2 คำสั่ง ในขณะที่โปรแกรมทำงานได้เหมือนกัน



แลดเดอร์ไดอะแกรม (Ladder Diagram) A

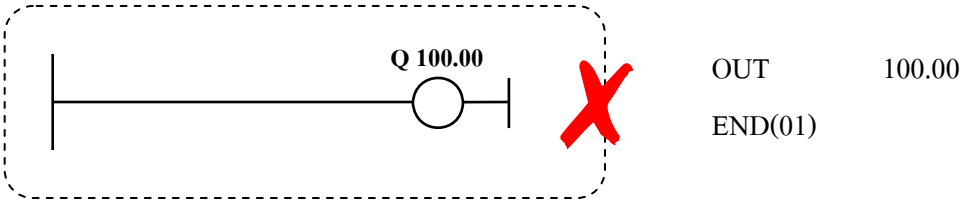
Address	Instruction	Operands
00000	LD	0.00
00001	LD	0.01
00002	LD	0.02
00003	AND	0.03
00004	OR LD	
00005	AND LD	
00006	OUT	100.00



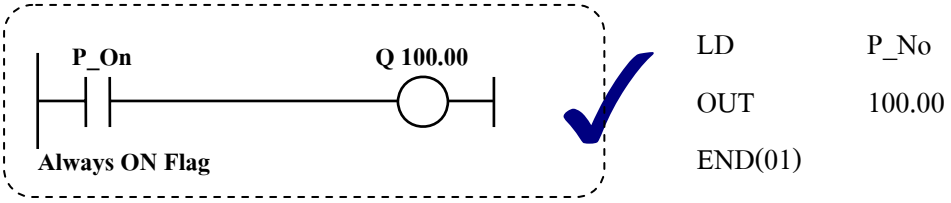
แลดเดอร์ไดอะแกรม (Ladder Diagram) B

Address	Instruction	Operands
00000	LD	0.02
00001	AND	0.03
00002	OR	0.01
00003	AND	0.00
00004	OUT	100.00

5.2.4 เมื่อต้องการให้เอาต์พุต ON ตลอดเวลาเราจะใช้แฟลค (Flag) ที่เป็นแบบ “Always ON Flag” (CF113) มาเป็นตัวสร้างเงื่อนไขเพราะไม่สามารถต่อคอยล์เอาต์พุตได้โดยตรงกับ Bus Bar แต่ก็มีข้อยกเว้นเป็นบางคำสั่ง เช่น INTERLOCK CLEAR, JUMP END และ STEP



แลดเดอร์ไดอะแกรม (Ladder Diagram) ที่ผิด

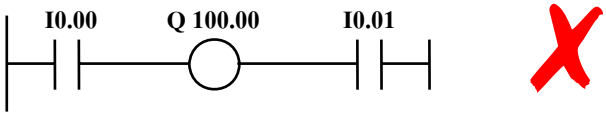


แลดเดอร์ไดอะแกรม (Ladder Diagram) ที่ถูก

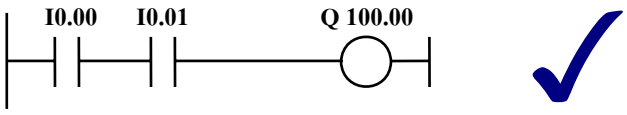
5.2.5 จำนวนหน้าสัมผัสที่ใช้ในการต่ออนุกรม หรือขนานไม่มีขีดจำกัดจะใช้เท่าใดก็ได้ขึ้นอยู่กับความต้องการของผู้ใช้

5.2.6 เอาต์พุตทุกๆ ตัวจะมี Auxiliary Contact เพื่อใช้งานในโปรแกรมได้ และสามารถใช้จำนวนไม่จำกัด

5.2.7 ไม่สามารถเขียนโปรแกรมให้หน้าสัมผัสอยู่ตำแหน่งหลังจากคอยล์ได้

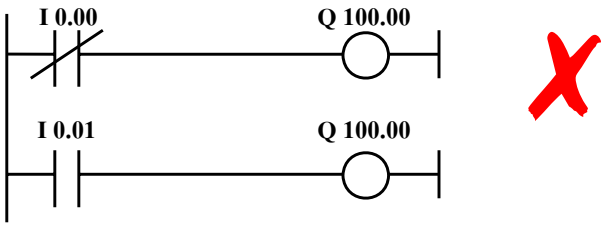


แลดเดอร์ไดอะแกรม (Ladder Diagram) ที่ผิด

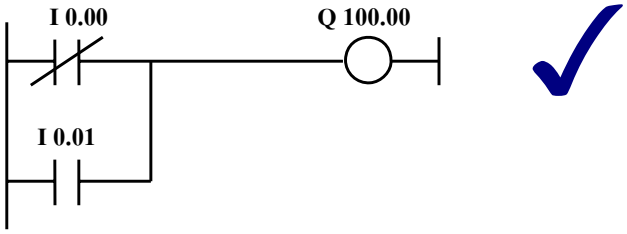


แลดเดอร์ไดอะแกรม (Ladder Diagram) ที่ถูก

5.2.8 ไม่สามารถเขียนโปรแกรมให้มีเอาต์พุตเบอร์เดียวกันซ้ำหลายๆ ครั้งได้ ต้องจัดรูปเสียใหม่

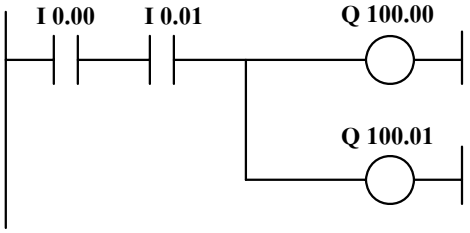


แลดเดอร์ไดอะแกรม (Ladder Diagram) ที่ผิด



แลดเดอร์ไดอะแกรม (Ladder Diagram) ที่ถูก

5.2.9 เอาต์พุตคอยล์ สามารถเขียนโปรแกรมให้ต่อขนานได้เลย กรณีรับเงื่อนไขของหน้าสัมผัสชุดเดียวกัน



5.2.10 PLC จะเริ่มประมวลผลโปรแกรมจาก Address แรกสุดจนกระทั่งถึงคำสั่ง END ตำแหน่งแรก โดยที่คำสั่ง END อาจจะมีหลายตำแหน่งในโปรแกรมที่เป็นเช่นนี้เพื่อจุดประสงค์สำหรับการทดสอบโปรแกรม กรณีที่ต้องการแยกโปรแกรมออกเป็นส่วนๆ เพื่อให้ง่ายต่อการตรวจสอบและแก้ไขโปรแกรม

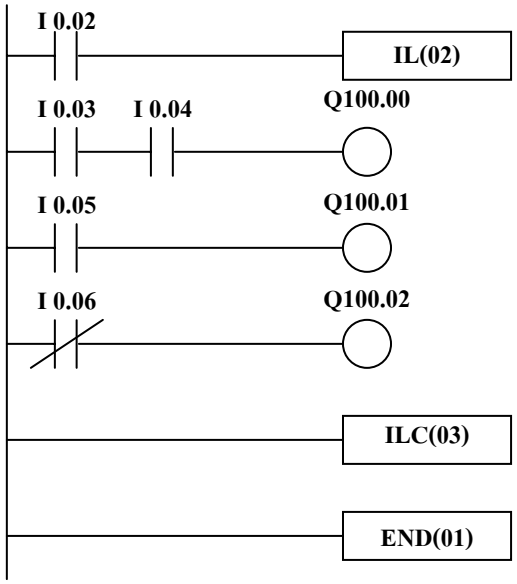
5.3 กลุ่มคำสั่ง Program Control Instruction

5.3.1 การใช้คำสั่ง IL(02), ILC(03)

คำสั่ง IL และ ILC จะต้องใช้ร่วมกันคือ ถ้าเริ่มต้นมีการใช้คำสั่งด้วย IL เมื่อใดแล้ว ถ้าต้องการสิ้นสุดการทำงานต้องจบด้วย ILC เงื่อนไขของคำสั่งคือ คอนแทคตรงหน้าส่วนของ IL มีสภาวะ “ON” จะทำให้โปรแกรมที่อยู่ระหว่าง IL และ ILC ทำงานเป็นปกติ แต่ถ้าคอนแทคตำแหน่งดังกล่าวมีสภาวะ “OFF” จะทำให้การทำงานของโปรแกรมระหว่าง IL และ ILC ไม่ทำงาน ในขณะเดียวกัน Output Coil ในช่วงนั้นจะมีสภาวะ “OFF” ด้วย

ตัวอย่างที่ 5.9 การใช้คำสั่ง

Ladder Diagram



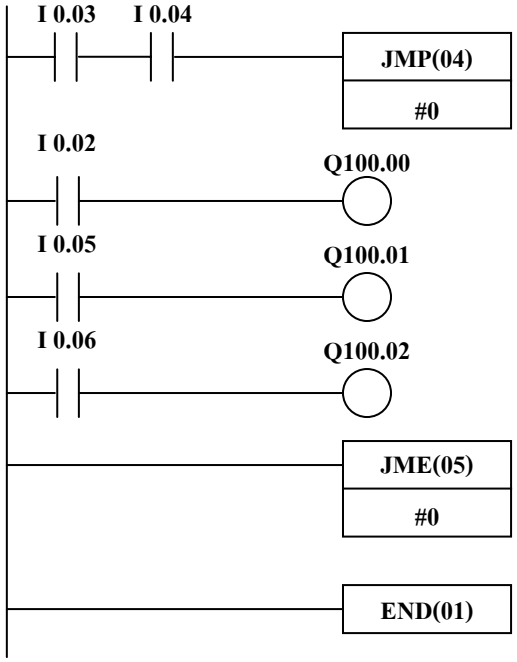
ชุดคำสั่ง

Address	Instruction	Data
00000	LD	0.02
00001	IL (02)	-
00002	LD	0.03
00003	AND	0.04
00004	OUT	100.00
00005	LD	0.05
00006	OUT	100.01
00007	LD NOT	0.06
00008	OUT	100.02
00009	ILC(03)	-
00010	END(01)	-

5.3.2 การใช้คำสั่ง JMP (04) และ JME (05)

การใช้งานของคำสั่งคู่นี้จะต้องใช้งานคู่กัน เงื่อนไขต่างๆ ที่อยู่ระหว่างคำสั่ง JMP และ JME จะมีเงื่อนไขการทำงานเป็นปกติ ในกรณีที่ชุดของคอนแทคตรงส่วนหน้าของ JMP มีสภาวะเป็น “ON” แต่ถ้าชุดคอนแทคดังกล่าวมีสภาวะเป็น “OFF” เมื่อใด Output, Timer, Counter, Keep ที่อยู่ระหว่างคำสั่งดังกล่าวจะยังคงค้างสภาวะเอาไว้เช่นเดิม และจะมีการเปลี่ยนแปลงอีกครั้งถ้าชุดของคอนแทคมีสภาวะ “ON” เราใช้ JUMP 00 ได้หลายครั้งตามต้องการ แต่ JUMP 01 ถึง 49 สามารถใช้ได้เพียงครั้งเดียว

Ladder Diagram

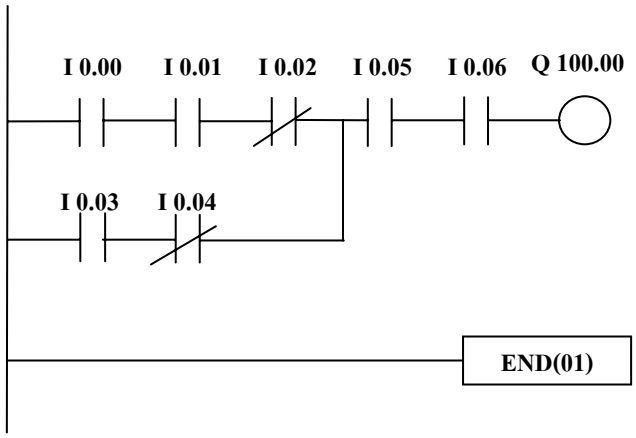


ชุดคำสั่ง

Address	Instruction	Operand
0000	LD	0.02
0001	AND	0.03
0002	JMP(04)	#0
0003	LD	0.04
0004	OUT	100.00
0005	LD	0.05
0006	OUT	100.01
0007	LD	0.06
0008	OUT	100.02
0009	JME(05)	#0
0010	END(01)	

แบบฝึกหัดทดสอบความเข้าใจเกี่ยวกับคำสั่งพื้นฐาน

1. จงเขียนโปรแกรมจาก Ladder Diagramให้เป็นรูป Mnemonic Code



Address	Instruction	Operands
00000		
00001		
0002		
00003		
00004		
00005		
00006		
00007		
00008		
00009		

2. จงเขียนโปรแกรมจาก Mnemonic ให้เป็นรูป Ladder Diagram

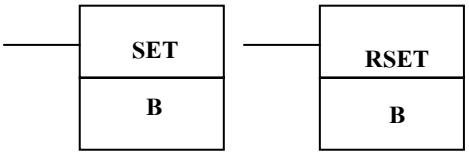
Address	Instruction	Operands
00000	LD	0.00
00001	AND	0.01
00002	LD NOT	0.02
00003	AND	0.03
00004	OR LD	
00005	LD	0.04
00006	AND NOT	0.05
00007	LD NOT	0.06
00008	AND	0.07
00009	OR LD	
00010	AND LD	
00011	OUT	100.00
00012	END (01)	

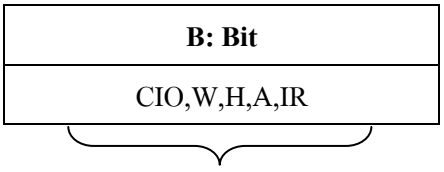
5.4 คำสั่งในกลุ่ม Bit Control Instruction

5.4.1 การใช้คำสั่งเซต (SET) และรีเซต (RESET)

คำสั่ง SET จะทำให้บิตที่ถูกสั่ง “ON” และคงค้างอยู่จนกว่าจะมีคำสั่ง RSET ที่บิตเดียวกัน บิตนั้นจึงจะ “OFF”

Ladder Symbol

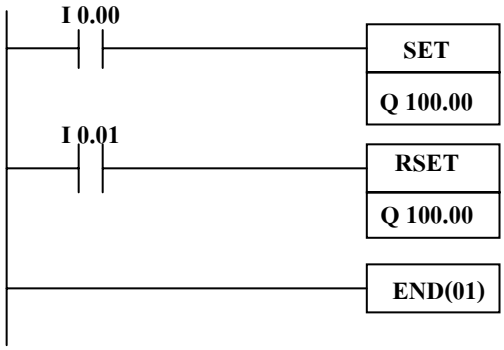




พื้นที่หน่วยความจำที่สามารถใช้กับคำสั่งได้

ตัวอย่างที่ 5.10 ต้องการให้หลอดไฟที่เอาต์พุต 100.00 ติดตลอดเวลา หลังจาก ON อินพุต 000.00 แล้วครั้งเดียวโดยไม่ต้อง Hold คำสั่งอินพุต จนกว่าจะมีการ Reset ที่อินพุต 000.01 หลอดไฟจะดับ

Ladder Diagram



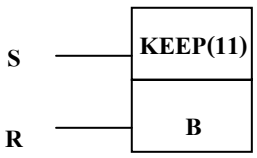
ชุดคำสั่ง

Address	Instruction	Operands
00000	LD	0.00
00001	SET	100.00
0002	LD	0.01
00003	RSET	100.00
00004	END	

5.4.2 การใช้คำสั่ง KEEP - KEEP(11)

การทำงานของคำสั่ง KEEP จะเหมือนกับคำสั่ง SET และ RESET เพียงแต่จับขา SET/RESET ให้รวมอยู่ในตัวเดียวกัน เพื่อให้ผู้ใช้งานสามารถเลือกใช้โปรแกรมได้สะดวกตามความเหมาะสม

Ladder Symbol



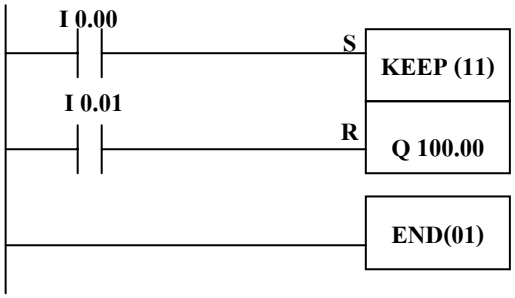
B: Bit
CIO,W,H,A,IR

พื้นที่หน่วยความจำที่สามารถใช้กับคำสั่งได้

เมื่อขา S มีสถานะ “ON” บิตที่ B จะทำงานจนกว่าขา R จะมีสถานะ “ON” บิต B ถึงจะเลิกทำงาน

ตัวอย่างที่ 5.11 ต้องการให้เอาต์พุต 100.00 เปลี่ยนเป็น “ON” ตลอดเวลาโดยการ ON อินพุต 0.00 ไม่ว่าจะ “OFF” แล้วก็ตามจนกว่าอินพุต 0.01 จะ ON (RESET)

Ladder Diagram



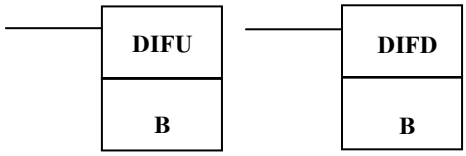
ชุดคำสั่ง

Address	Instruction	Operands
00000	LD	0.00
00001	LD	0.01
00002	KEEP	100.00
00003	END	

5.4.3 การใช้คำสั่ง DIFFERENTIATE UP และ DOWN–DIFU(13), DIFD(14)

คำสั่ง DIFU(13) และ DIFD(14) จะเป็นคำสั่งที่ทำงานเพียงขอบขาขึ้น หรือขอบขาลงของสัญญาณอินพุตเท่านั้น และจะทำงานเพียงช่วงเวลา One Cycle เท่านั้น

Ladder Symbols

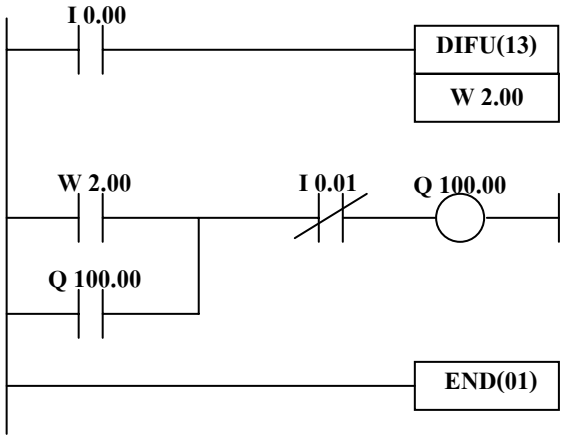


B: Bit
CIO,W,H,A,IR

พื้นที่หน่วยความจำที่สามารถใช้กับคำสั่งได้

ตัวอย่างที่ 5.12 ต้องการให้อินพุต 0.00 ทำงานเพื่อ ON บิต W2.00 ซึ่งจะทำให้เอาต์พุตหลอดไฟ 100.00 ให้ติดได้โดยอินพุต 0.01 เป็นตัวสั่ง OFF

Ladder Diagram



ชุดคำสั่ง

Address	Instruction	Operands
00000	LD	0.00
00001	DIFU	W2.00
0002	LD	W2.00
00003	OR	100.00
00004	AND NOT	
00005	OUT	100.00
00006	END	

5.5 กลุ่มคำสั่ง Timer/Counter

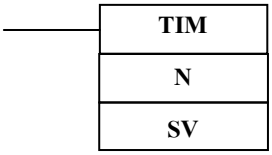
สำหรับ PLC บางรุ่น Timer และ Counter จะใช้พื้นที่เดียวกันซึ่งเรียกใช้ได้ทั้งหมด 256 ตัว ตั้งแต่ตัวที่ 000 ถึง 255 ภายใน 256 ตัวนี้สามารถกำหนดให้เป็น Timer หรือ Counter ก็ได้โดยที่หากตัวใดถูกกำหนดให้เป็น Timer แล้วจะนำไปใช้เป็น Counter อีกไม่ได้ ดังนั้นต้องดู Manual ของ PLC รุ่นนั้นประกอบด้วย ถ้า Timer/Counter อยู่ในพื้นที่เดียวกัน จะไม่สามารถใช้เบอร์เดียวกันได้

แต่ PLC บางรุ่น Timer/Counter จะอยู่คนละพื้นที่ ดังนั้นจึงสามารถใช้ Timer และ Counter เบอร์เดียวกันได้ เช่น T000 และ C000

สำหรับคำสั่งในกลุ่ม Timer/Counter มีหลายคำสั่ง ในที่นี้จะยกตัวอย่างการใช้งานคำสั่ง Timer/Counter แบบพื้นฐานคือ คำสั่ง TIM และ CNT ดังนี้

5.5.1 การใช้คำสั่ง TIMER: TIM

ใช้ในการจับเวลา, ตั้งเวลา โดยพื้นฐานแล้วต้องเข้าไปกำหนดค่า 2 ค่าคือ N และ SV ตามตัวอย่างข้างล่าง



N: Timer Number (หมายเลขของ Timer)
T000-T4095 (CP1L)

SV: Set Value (ค่าตั้งเวลา)
ค่าคงที่ (#) หรือการอ้างถึง CIO, W, H, A, T, C, D, E, E?_, @D, @E, @E?_, *D, *E, *E?_, DR, ,IR

N = Timer Number (เบอร์ 0000 - 4095) เลือกว่าจะใช้ Timer ตัวที่เท่าใด

SV = Set Value ค่าตั้งเวลา ใช้กำหนดว่าจะให้ Timer ตั้งเวลานานเท่าใด ซึ่ง SV ที่ตั้งนั้น จะถูกคูณด้วย 0.1 เพื่อแปลงเป็นระยะเวลาจริง ซึ่งสามารถ

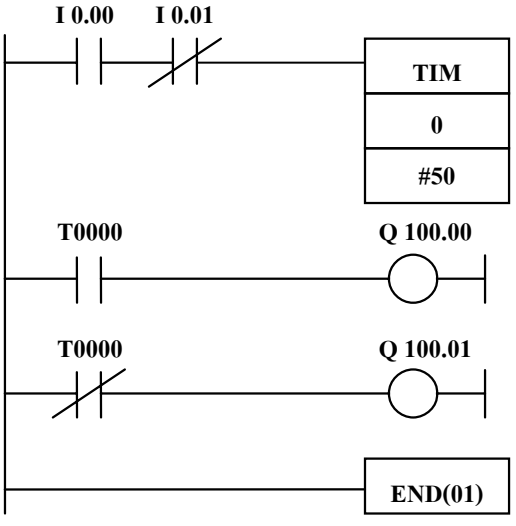
1. กำหนด SV เป็นค่าคงที่ #0000-9999 (000.0-999.9 วินาที คูณด้วย 0.1 วินาที)
2. กำหนด SV เป็น แอดเดรส CIO,A,T,C,IR,DR,D,TK,H,Wโดยใส่ค่าตั้งเวลาที่เป็นค่าคงที่ 0000-9999 ไว้ใน แอดเดรส ที่อ้างถึงอีกทีหนึ่ง (ค่าที่กำหนดจะคูณด้วย 0.1 วินาทีเช่นเดียวกับการกำหนดแบบค่าคงที่)

หมายเหตุ *ในที่นี้ยกตัวอย่างหมายเลข Timer ของ PLC รุ่น CP1L เท่านั้น สำหรับ PLC รุ่นอื่นๆ สามารถใช้ Timer ได้มากกว่าหรือน้อยกว่าที่กำหนด

เมื่อมีสัญญาณสั่งให้ Timer ทำงาน (Contact B มีสถานะ “ON”) คำสั่ง Timer จะเริ่มนับเวลาตามค่าที่ตั้งไว้ใน Timer เมื่อนับครบเวลา หน้า Contact ของ Timer ตัวนั้นๆ ก็จะ “ON” แต่ถ้าสัญญาณที่สั่งให้ Timer ทำงานหายไป (Contact B มีสถานะ OFF) Timer จะถูก Reset

ตัวอย่างที่ 5.13 การใช้งานของคำสั่ง Timer เมื่ออินพุต 0.00 ทำงาน (ON) ค้างเป็นเวลา 5 วินาที เอาต์พุต 100.00 จะ ON และ เอาต์พุต 100.01 จะ OFF

Ladder Diagram

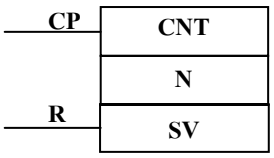


ชุดคำสั่ง

Address	Instruction	Operands
00000	LD	0.00
00001	AND NOT	0.01
00002	TIM 000	#050
00003	LD	TIM0000
00004	OUT	100.00
00005	LD NOT	TIM0000
00006	OUT	100.01
00007	END (01)	

5.5.2 การใช้คำสั่ง COUNTER - CNT

เป็นคำสั่งที่ใช้นับจำนวนครั้งของสัญญาณ อินพุต ที่ ON แต่ละครั้ง ซึ่งเป็นคำสั่งที่นับลงจากค่าที่ตั้งไว้ (Set Value)



N: Counter Number (หมายเลขของ Counter)
C000-C4095 (CP1L)

SV: Set Value (ค่าตั้งจำนวนนับ)
ค่าคงที่(#)หรือการอ้างถึง CIO, W, H, A, T, C, D, E, E?_, @D, @E, @E?_, *D, *E, *E?_, DR, ,IR

N = Counter Number (เบอร์ 000- 4095) เลือกว่าจะใช้ Counter ตัวที่เท่าใด

SV = Set Value ค่าตั้งจำนวนนับ ใช้กำหนดว่าจะให้ Counter นับสัญญาณอินพุตเป็นจำนวนกี่ครั้ง หน้า Contact เอาต์พุตของ Counter จึงจะเริ่มทำงานซึ่งสามารถ

1. กำหนด SV เป็นค่าคงที่ #0000-9999
2. กำหนด SV เป็น แอดเดรส CIO, W, H, A, T, C, D, E, E?_, @D, @E, @E?_, *D, *E, *E?_, #, DR, ,IR โดยใส่ค่าตั้งจำนวนนับที่เป็นค่าคงที่ 0000-9999 ไว้ในแอดเดรส ที่อ้างถึงอีกทีหนึ่ง

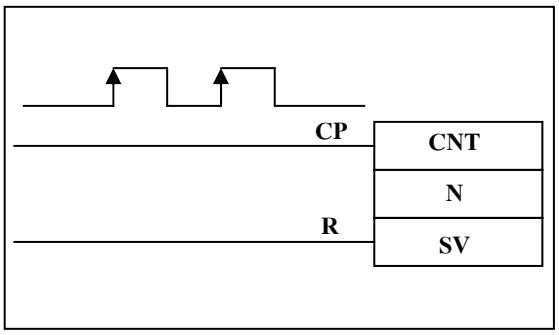
CP = ขานับ เมื่อมีสัญญาณอินพุตในช่วงที่เปลี่ยนสถานะจาก OFF เป็น ON  เข้ามาที่ขานี้ Counter จะนับถอยหลังลง 1

R = ขา Reset เมื่อมีสัญญาณอินพุตเข้ามาที่ขานี้ เอาต์พุตของ Counter จะหยุดทำงานและค่านับของ Counter จะถูก Reset กลับไปเท่ากับค่าตั้งจำนวนนับ (SV)

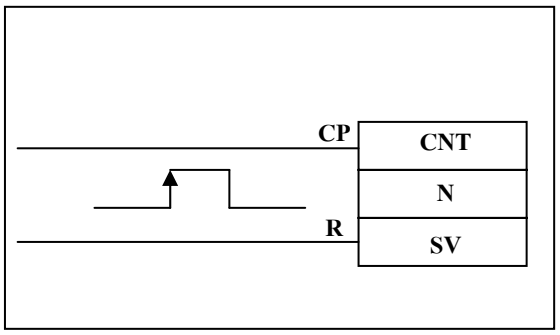
หมายเหตุ *ในที่นี้ยกตัวอย่างหมายเลข Counter ของ PLC รุ่น CP1L เท่านั้นสำหรับ PLC รุ่นอื่นๆ สามารถใช้ Counter ได้มากกว่าหรือน้อยกว่าที่กำหนด

*Memory Area ของ Timer และ Counter จะใช้พื้นที่แยกกัน แต่ใน PLC รุ่นเก่า เช่น CPM2A จะใช้ Timer และ Counter กับเบอร์เดียวกันไม่ได้

ตัวอย่างที่ 5.14 แสดงการใช้คำสั่ง Counter สัญญาณที่ส่งเข้าที่ขาสัญญาณนับจะทำให้ Counter นับถอยหลังเมื่อเทียบกับ SV ตัว Counter จะเริ่มนับที่ขอบขาขึ้นของสัญญาณอินพุตและจะนับใหม่เมื่อสัญญาณเปลี่ยนเป็น OFF หรือ '0' แล้วกลับมา ON อีกครั้ง และจะทำงานเช่นนี้ไปจนครบค่าที่ตั้งไว้ (SV) เอาต์พุตจึง ON



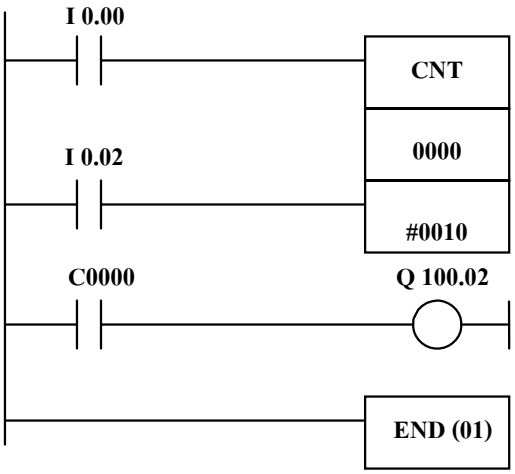
จังหวะการนับ



จังหวะการรีเซต

ตัวอย่างที่ 5.15 การใช้งานของคำสั่ง Counter เมื่อ อินพุต 0.00 ทำงาน (ON) 1 ครั้ง Counter จะนับ 1 ครั้ง ถ้าอินพุต 0.00 ทำงาน (ON) ครบ 10 ครั้ง จะทำให้คำสั่ง Counter ทำงานพร้อมกับ Contact ของ Counter (CNT0000) จะทำงานด้วย และจะถูก Reset ด้วยอินพุต 0.02

Ladder Diagram



ชุดคำสั่ง

Address	Instruction	Operands
00000	LD	0.00
00001	LD	0.02
00002	CNT 001	#0010
00003	LD	CNT0000
00004	OUT	100.02
00005	END (01)	

- **การประยุกต์ใช้งานของ TIMER และ COUNTER**

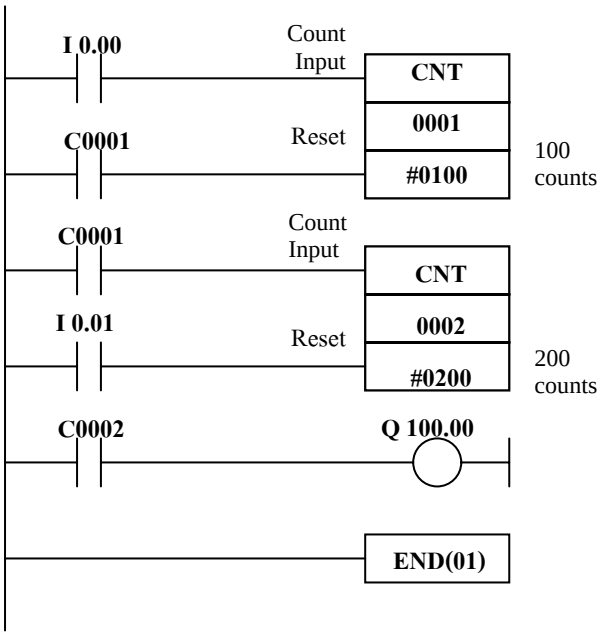
ตัวอย่างที่ 5.16

ต้องการนับจำนวนคนดูคอนเสิร์ต สถานที่จัดงานสามารถจุคนดูได้ 20,000 คน ใช้ Photo Switch นับจำนวนคน หลังจาก 20,000 คนแล้วให้ OUTPUT LAMP 100.00 ทำงาน เพื่อแสดงว่าคนเต็มแล้ว

I/O ที่กำหนด

PHOTO SWITCH	0.00
PB RESET	0.01
OUTPUT LAMP	100.00

Ladder Diagram



ชุดคำสั่ง

Address	Instruction	Data
00000	LD	0.00
00001	LD	CNT 0001
00002	CNT	0001
		#0100
00003	LD	CNT 0001
00004	LD	0.01
00005	CNT	0002
		#0200
00006	LD	CNT 0002
00007	OUT	100.00
00010	END (01)	

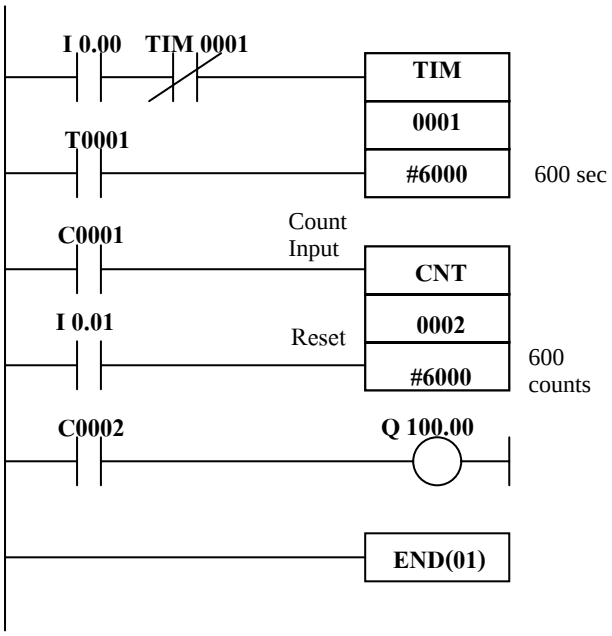
ตัวอย่างที่ 5.17

เครื่องจักรเครื่องหนึ่งต้องการอัดจารบีหลังจากใช้งานไปแล้วครบ 1,000 ชั่วโมง

I/O ที่กำหนด

PB START	0.00
PB RESET	0.01
VALUE LUBRICATE	100.00

Ladder Diagram

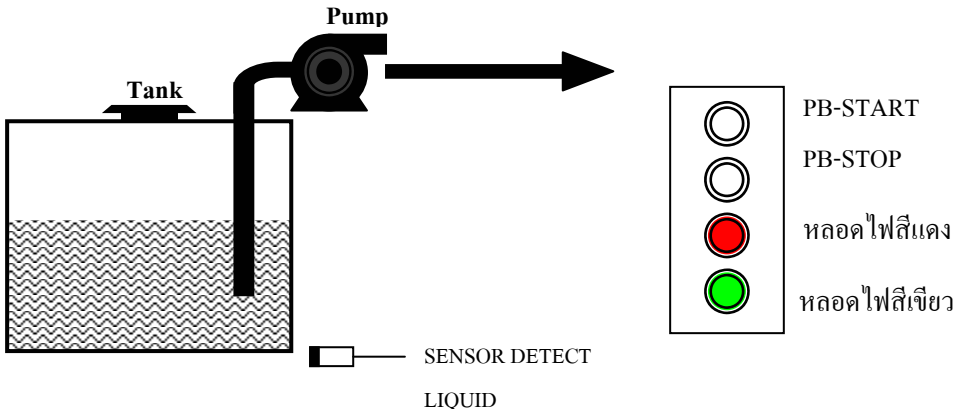


ชุดคำสั่ง

Address	Instruction	Data
00000	LD	0.00
00001	AND-NOT	TIM 0001
00002	TIM	0001
		# 6000
00003	LD	TIM 0001
00004	LD	0.01
00005	CNT	0002
		# 6000
00006	LD	CNT 0002
00007	OUT	100.00
00008	END (01)	

ตัวอย่างที่ 5.18

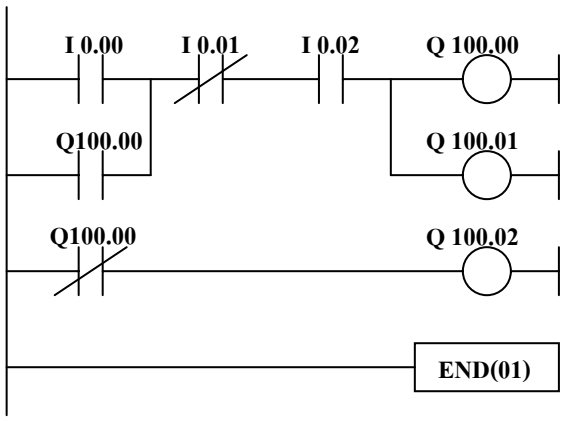
มอเตอร์ปั๊มน้ำ ถ้ากดปุ่ม PB-START หลอดไฟสีแดงจะติดเพื่อแสดงว่าปั๊มทำงาน และปั๊มน้ำจะหยุดการทำงานพร้อมกับหลอดไฟสีแดงดับ ก็ต่อเมื่อกด PB-STOP หลอดไฟสีเขียวจะติดแทน (ปั๊มจะทำงานได้ต้องมีน้ำในแทงค์เท่านั้น)



I/O ที่กำหนด

INPUT	0.00	PB-START
	0.01	PB-STOP
	0.02	SENSOR DETECT LIQUID
OUTPUT	100.00	MOTOR PUMP
	100.01	หลอดไฟสีแดง
	100.02	หลอดไฟสีเขียว

Ladder Diagram



ชุดคำสั่ง

Address	Instruction	Operand
00000	LD	0.00
00001	OR	100.00
00002	AND NOT	0.01
00003	AND	0.02
00004	OUT	100.00
00005	OUT	100.01
00006	LD NOT	100.00
00007	OUT	100.02
00008	END (01)	

แบบฝึกหัด

จงเขียนโปรแกรมตั้งเวลา 24 ชั่วโมง โดยเมื่อครบตามเวลาที่กำหนดต้องการให้มอเตอร์ทำงาน

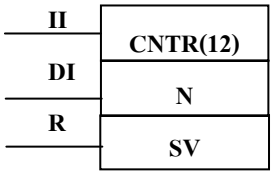
Ladder Diagram

Mnemonic Code

Address	Instruction	Data
00000		
00001		
00002		
00003		
00004		
00005		
00006		
00007		
00008		
00009		
00010		

5.5.3 การใช้คำสั่ง Reversible Counter CNTR (FUN 12) หรือ UP/DOWN Counter

คำสั่ง CNTR ใช้อินพุต 3 อินพุตควบคุมการทำงานคือ II (Increment Input), DI (Decrement Input) และ Reset Input (R) การใช้งานต้องระบุเบอร์ของตัวเคาท์เตอร์และกำหนดค่าการนับ (Set Value) เป็นจำนวนเท่าใดด้วย



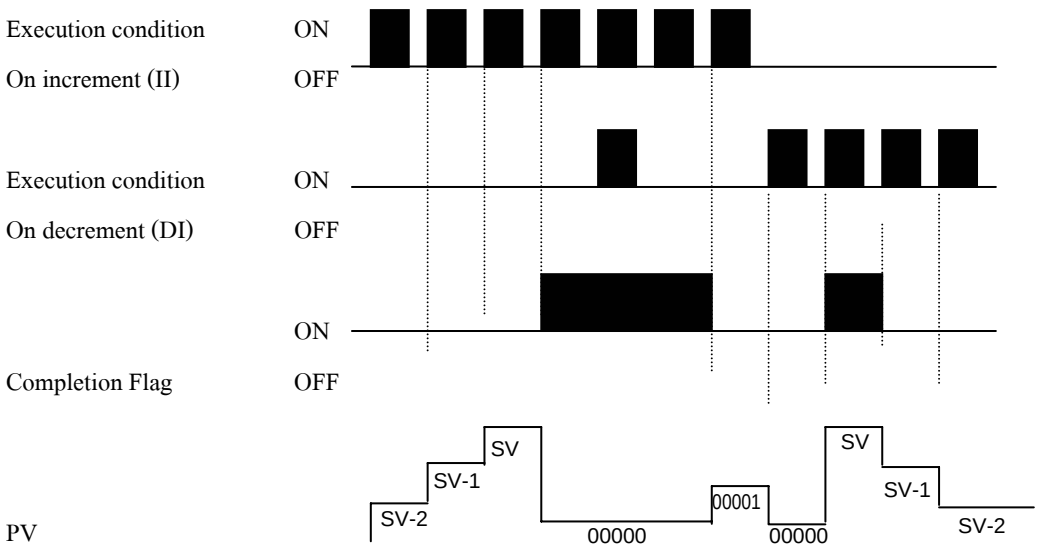
N: Counter Number (หมายเลขของ Counter)
C000-C4095

II- คือขานับสัญญาณแบบนับขึ้น
DI- คือขานับสัญญาณแบบนับลง
R- คือขา Reset

SV: Set Value (word, BCD)
CIO, W, H, A, T, C, D, E, E?_, @D, @E, @E?_, *D, *E, *E_, #, DR, ,IR

พื้นที่หน่วยความจำที่ใช้กับคำสั่งได้

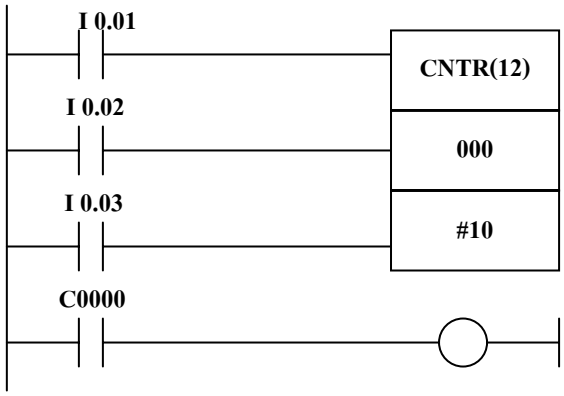
การกำหนดค่า SV (ค่าที่จะตั้งให้นับ) สามารถกำหนดเป็นค่าคงที่ หรือกำหนดผ่านหน่วยความจำต่างๆ ที่กำหนดไว้ให้เช่น DM ต้องตั้งอยู่ในย่าน 1-9,999 ลักษณะการทำงานของ CNTR แสดงได้ดังนี้



ตัวอย่างที่ 5.19

การใช้งาน Counter ชนิด UP/DOWN counter หรือ Reversible Counter

Ladder Diagram



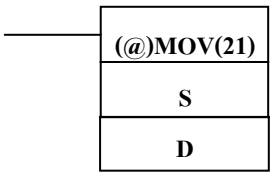
ชุดคำสั่ง

Address	Operand	Data	
00000	LD		0.01
00001	LD		0.02
00002	LD		0.03
00003	CNTR(12)		0000
00004		#	10
00005	LD	CNT	0010
00006	OUT		100.00

Reversible Counter หรือเรียกอีกชื่อว่า UP-DOWN Counter ทั้งนี้เพราะสามารถทำการนับขึ้นในกรณีที่มีสัญญาณเข้าที่ II และมีการนับลงเมื่อมีสัญญาณเข้าที่ DI แต่ถ้ามีสัญญาณ UP และ DOWN ที่อินพุตพร้อมกัน จะไม่ทำให้เกิดการนับในทิศทางใดๆ การนับจะนับเป็นลักษณะวนหรือต่อเนื่องกันไปเรื่อยๆ กล่าวคือ เมื่อนับครบตามจำนวนที่กำหนดไว้แล้ว จะวนกลับมาเป็น “0”หรือ “10” ใหม่ เป็นเช่นนี้ไปเรื่อย และการแสดงค่าเอาต์พุตของคำสั่ง CNTR จะมีสภาวะ “ON” เพียง 1 ครั้งเท่านั้น ในจังหวะที่เป็นการวนค่าเช่นจาก “10” เป็น “0” ในกรณีที่มีการเพิ่มหรือลดถึงค่าที่กำหนดเอาไว้ ส่วนสัญญาณ Reset ถือว่าเป็นอีกอินพุตหนึ่งของ CNTR ถ้ามีสภาวะ “ON” เมื่อใดจะทำให้ค่าที่นับได้มีค่าเริ่มต้นที่ “0000” ทันที

5.6 กลุ่มคำสั่ง Data Movement

5.6.1 การใช้คำสั่ง MOVE – MOV(21)



S: Source word
CIO, W, H, A, T, C, D, E, E?_, @D, @E, @E?_, *D, *E, *E?_, #, DR, ,IR

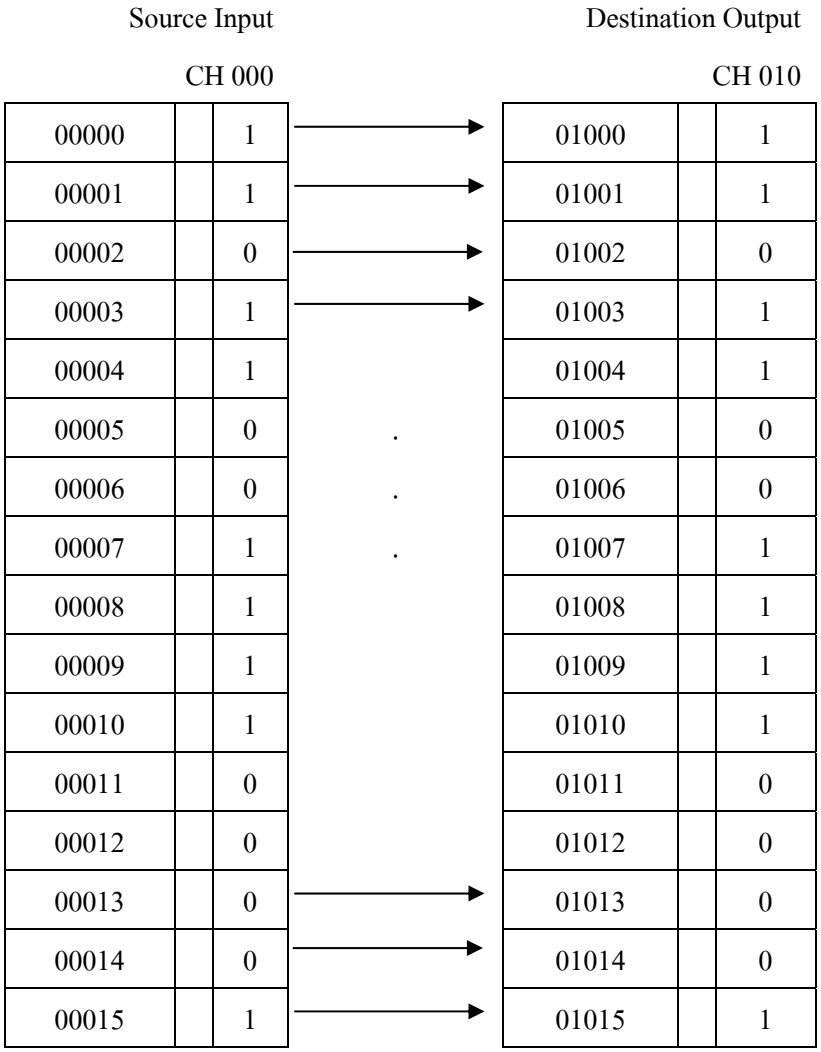
พื้นที่หน่วยความจำที่ใช้กับคำสั่งได้

D: Destination word
CIO, W, H, A, T, C, D, E, E?_, @D, @E, @E?_, *D, *E, *E?_, DR, ,IR

พื้นที่หน่วยความจำที่ใช้กับคำสั่งได้

S = เวิร์ดต้นฉบับที่ต้องการ Copy (Source word)
D = เวิร์ดปลายทางที่ Copy (Destination word)

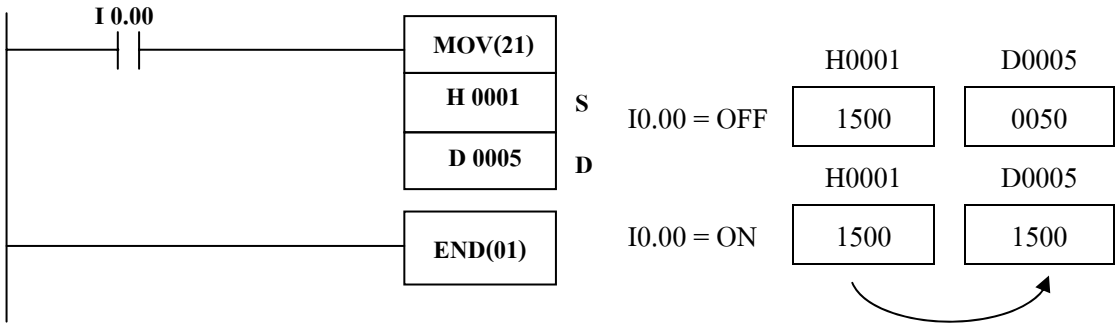
เมื่อคำสั่งนี้ทำงาน มันจะทำการสำเนา (Copy) ข้อมูลจาก S ไปยัง D โดยที่ S ยังมีข้อมูลเดิมอยู่ ถ้าใส่สัญลักษณ์ @ ข้างหน้า คำสั่งนี้จะทำงานเพียงแค่ 1 Scan time เท่านั้น จะไม่มีการสำเนาข้อมูลจาก S ไปยัง D อีก แม้ว่าจะยังมีสัญญาณอินพุต ON ที่คำสั่ง MOV



ตัวอย่างที่ 5.20

ค่าของ S = H 0001 (มีค่าข้อมูลคือ 1500) และ D = D 0005 (มีค่าข้อมูลคงคือ 0050) เมื่อคำสั่ง MOV(21) ทำงาน ค่าที่ D จะมีข้อมูลใหม่คือ 1500

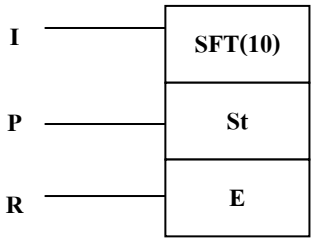
Ladder Diagram



5.7 กลุ่มคำสั่ง Data Shifting

5.7.1 การใช้คำสั่ง SHIFT REGISTER – SFT(10)

Ladder Symbol



Operand

St : Starting word
CIO, W, H, A

E : End word
CIO, W, H, A

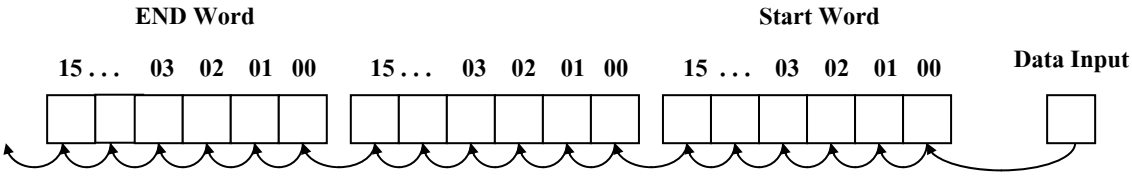
I คือ ขา Data Input กำหนดค่าที่ใช้ในการเลื่อนข้อมูลเป็น “0” หรือ “1”

P คือ ขา Pulse Input ใช้ป้อนสัญญาณพัลส์เพื่อเลื่อนข้อมูลจาก I เข้าไป

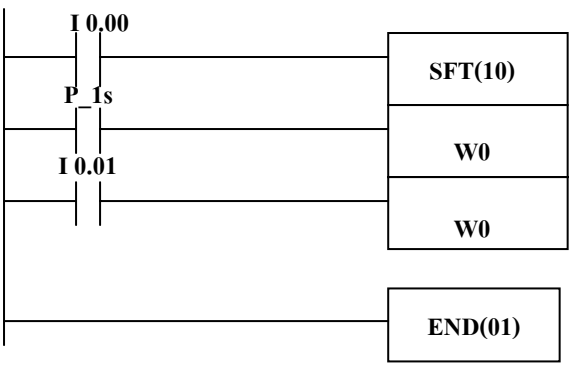
R คือ ขา Reset ใช้กำหนดค่าข้อมูลใน Starting Word ถึง End Word ให้มีค่าเป็น 0

การทำงานจะเลื่อนข้อมูลทีละบิต จากบิต 0 ของ Starting word ไปจนถึงบิต 15 ของ End word ถ้ามีการ ON ที่ขา P (Pulse) แต่ละครั้ง ถ้าขา P ON หนึ่งครั้ง ข้อมูลก็จะเลื่อนหนึ่งบิต

ตัวอย่างที่ 5.21



ทดลองเขียนคำสั่ง SFT(10) โดยมี 1 Second clock puls (P_1s) ที่ขา P และ Turn On สวิตช์ I0.00 ซึ่งเป็นค่าข้อมูลที่จะเลื่อนเข้าไป โดยเอาต์พุตเวิร์ด W0 จะ ON ตั้งแต่บิตที่ W0.00 แล้วเลื่อนไป ON ที่ W0.01



Address	Instruction	Operands
00000	LD	0.00
00001	LD	P_1s
00002	LD	0.01
00003	SFT(10)	
		W 0
		W 0
00004	END(01)	

บทที่ 6
การใช้ซอฟต์แวร์ป้อนโปรแกรม

ในบทที่ผ่านมาเราได้กล่าวถึงหลักการเขียนแลดเดอร์โปรแกรมและคำสั่งพื้นฐานต่างๆ เพื่อนำไปประยุกต์กับการใช้งานจริงได้อย่างเหมาะสม ต่อจากนี้ไปเราจะเริ่มต้นการเขียนโปรแกรมโดยจะใช้ซอฟ์ตแวร์ CX-programmer ในการสร้างแลดเดอร์ไดอะแกรม

6.1 การติดตั้งซอฟต์แวร์ CX-Programmer

6.1.1 PLC ที่สามารถใช้งานกับซอฟต์แวร์ CX-Programmer

CX-Programmer เป็นซอฟต์แวร์ที่พัฒนาขึ้นมาแทนซอฟต์แวร์ Syswin สามารถใช้งานได้กับ PLC ของ OMRON รุ่นต่างๆ ได้ดังตารางต่อไปนี้

PLC Series	รุ่น
CP-Series	CP1L, CP1H
CJ-Series	CJ1G, CJ1G-H, CJ1H-H, CJ1M
CS-Series	CS1G, CS1G-H, CS1H, CS1H-H, CS1D-H, CS1D-S
CV-Series	CV1000, CV2000, CV500, CVM1, CVM1-V2
C-Series	C1000H, C2000H, C200H, C200HE, C200HE-Z, C200HG, C200HGZ, C200HS, C200HX, C200HX- Z, CPM1A, CPM1, CPM2*, CPM2*-S*, CQM1, CQM1H
SRM1	SRM1,SRM1-V2

หมายเหตุ สำหรับรุ่นของ PLC ตามตารางนั้น จะเปลี่ยนไปตามการพัฒนาซอฟต์แวร์ ถ้าซอฟต์แวร์สูงขึ้น รุ่นของ PLC จะมีให้เลือกเพิ่มขึ้น

6.1.2 ข้อแนะนำสำหรับเครื่องคอมพิวเตอร์ที่ใช้งาน (System Requirements)

CX-Programmer (และ CX-Server) สามารถทำงานได้กับเครื่องคอมพิวเตอร์ PC (IBM-AT หรือ NEC PC-98) ตั้งแต่ Pentium II ขึ้นไป โดยทำงานภายใต้ระบบปฏิบัติการMS-Windows 95, 98, ME, XP หรือ Windows NT Service pack 5, 2000 (หรือ Version ที่ใหม่กว่า)

หมายเหตุ: CX-Programmer (และ CX-Server) ไม่รับประกันการทำงานบนเครื่องคอมพิวเตอร์ที่เป็นระบบปฏิบัติการอย่างอื่นนอกเหนือจากของ MS-Windows ปกติ (เช่น พวก Windows Emulation อย่าง Apple Macintosh, หรือเครื่อง PC ที่ใช้ระบบปฏิบัติการ Linux)

อย่างไรก็ตามข้อแนะนำสำหรับของระบบคอมพิวเตอร์ขั้นต่ำที่ CX-Programmer (และ CX-Server) จะสามารถทำงานได้อย่างมีประสิทธิผล มีดังนี้

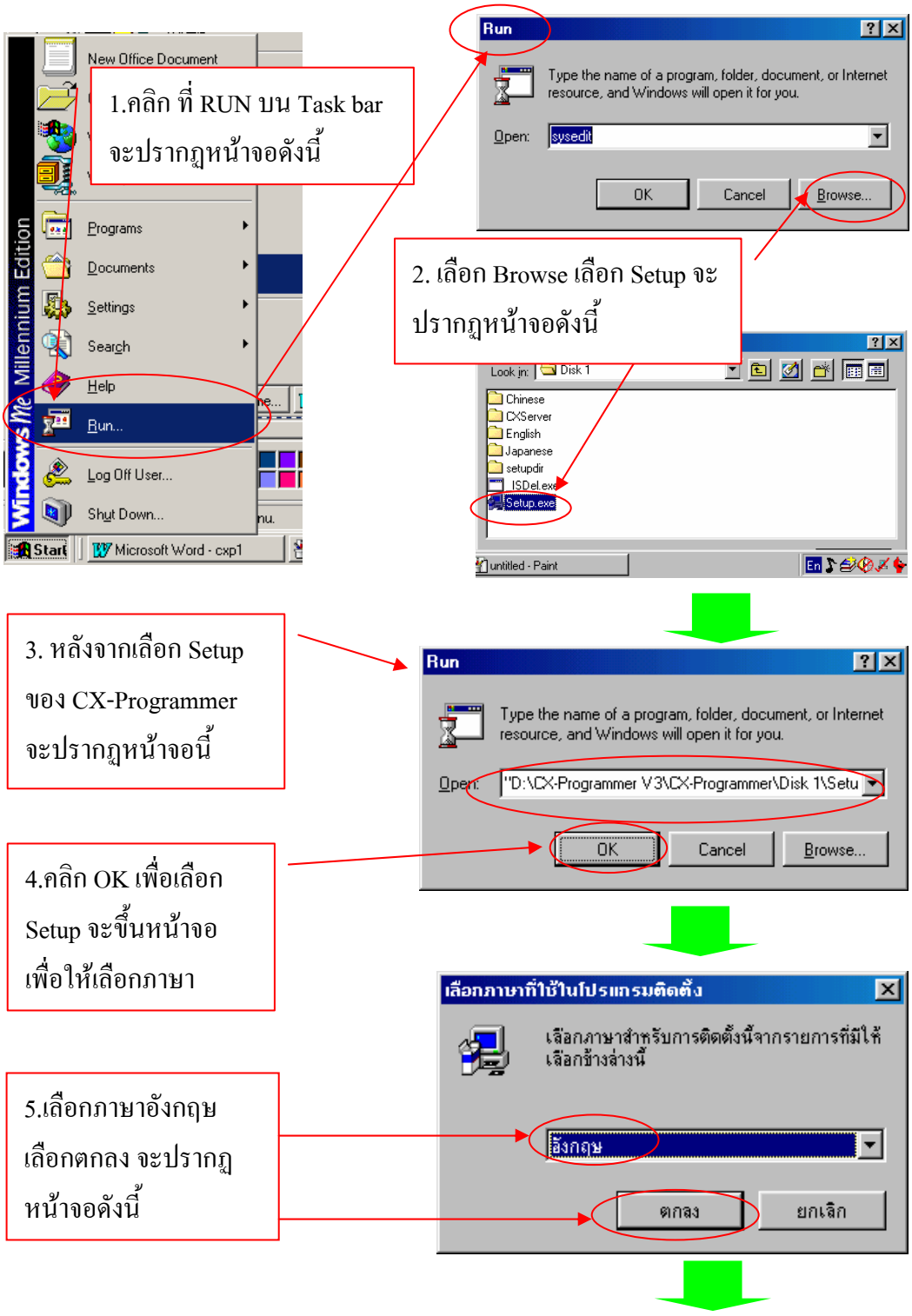
Minimum system Requirements

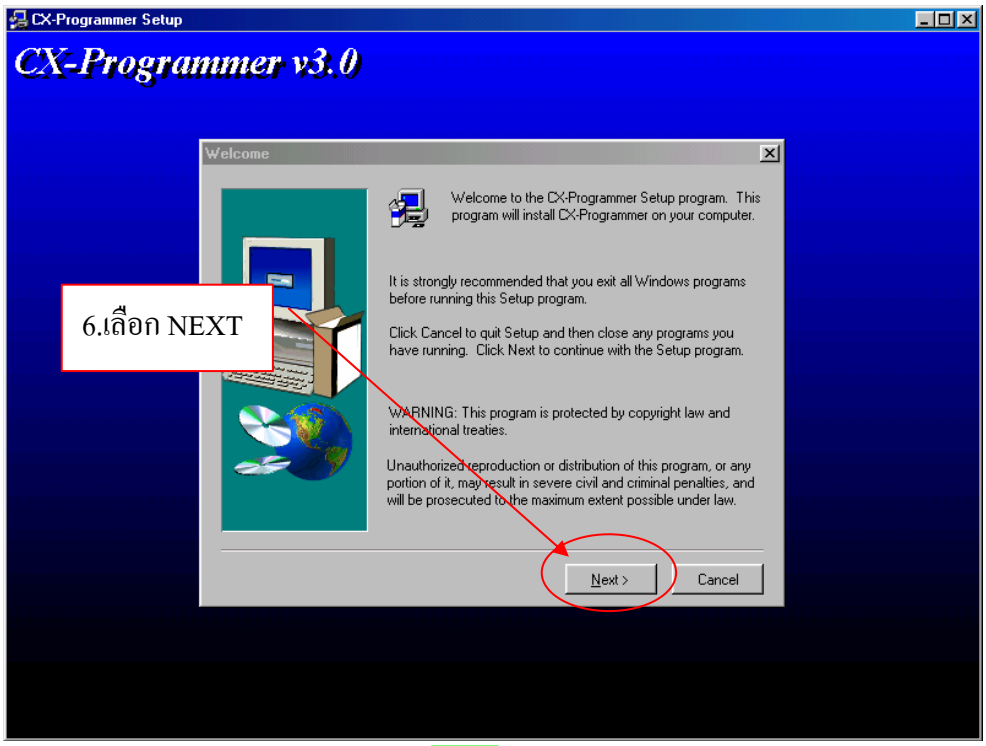
Operating system	Minimum Specification				Recommended Minimum Specification			
	CPU Type	Memory (RAM)	HDD Space	Display	CPU Type	Memory (RAM)	HDD Space	Display
Windows 95 Windows 98 Windows NT (with SP5)	Pentium Class 133MHz	32Mb	100Mb	800x600 SVGA	Pentium Class II 200MHz	64Mb	150Mb	1024x768 SVGA
Windows 2000 Windows ME	Pentium Class 150MHz	64Mb	100Mb	800x600 SVGA	Pentium Class II 200MHz	64Mb	150Mb	1024x768 SVGA
Windows XP Home Windows XP Professional	Pentium Class II 300MHz	128Mb	100Mb	800x600 SVGA	Pentium Class II 200MHz	64Mb	150Mb	1024x768 SVGA

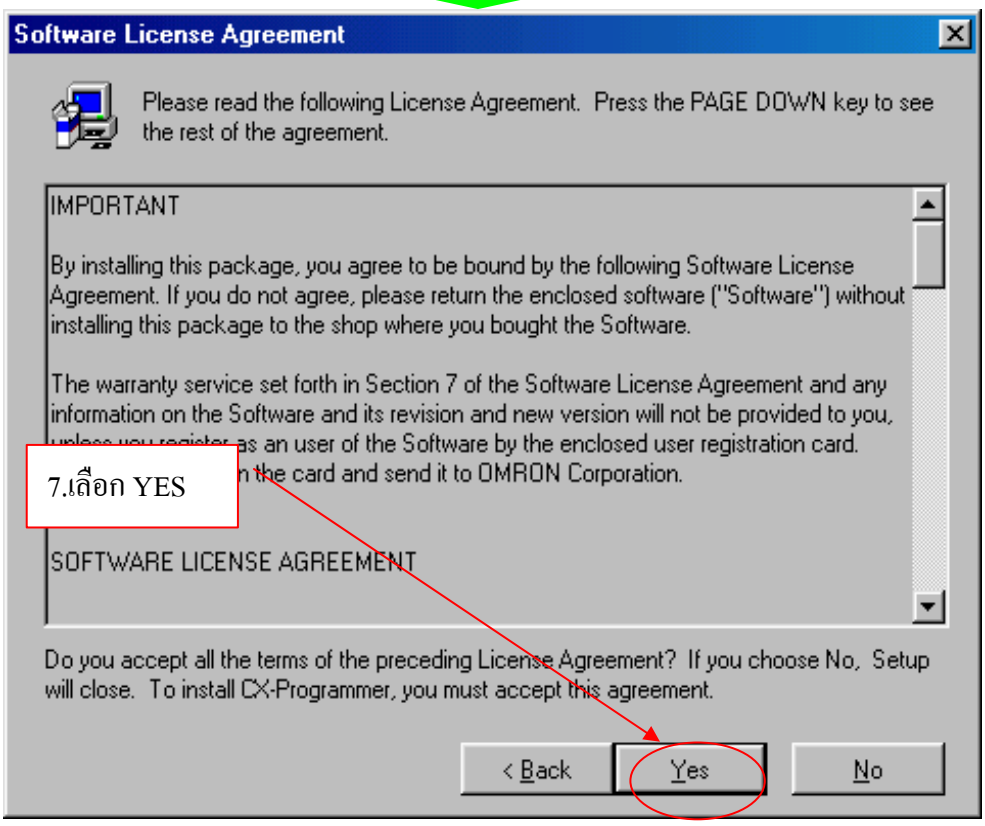
หมายเหตุ แม้ว่าจะสามารถทำงานโดยใช้ Keyboard ได้ทั้งหมด อย่างไรก็ตามเพื่อความสะดวกในการทำงานควรใช้ Mouse

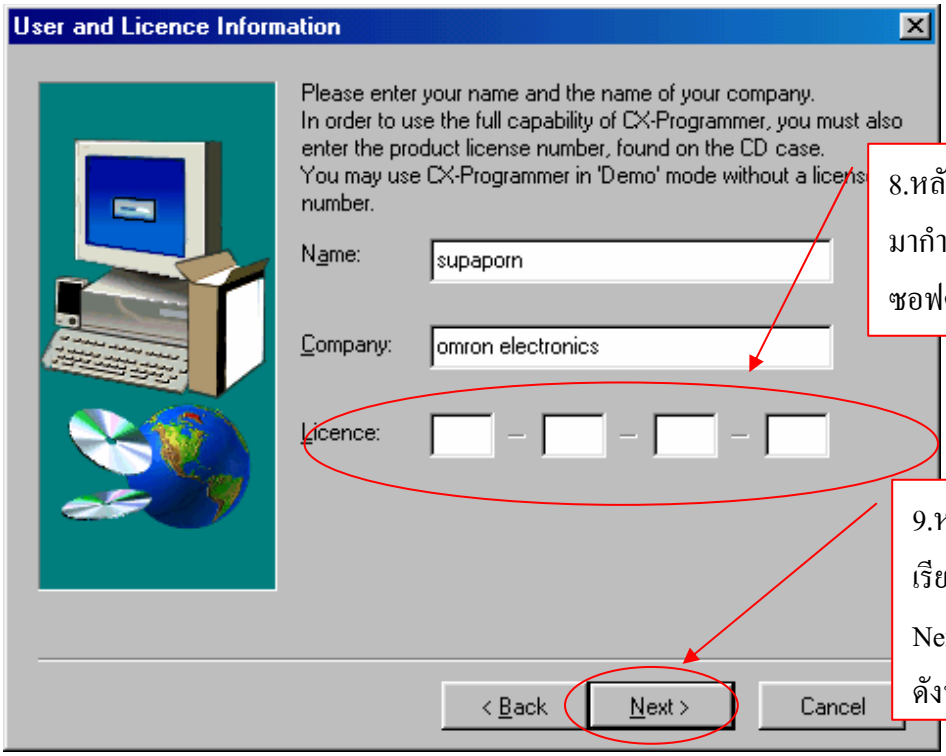
6.1.3 การติดตั้งซอฟต์แวร์ CX-Programmer

ซอฟต์แวร์ตัวนี้จะทำงานบนระบบปฏิบัติการ MS Windows95, 98, ME, NT หรือ 2000 ขึ้นไป



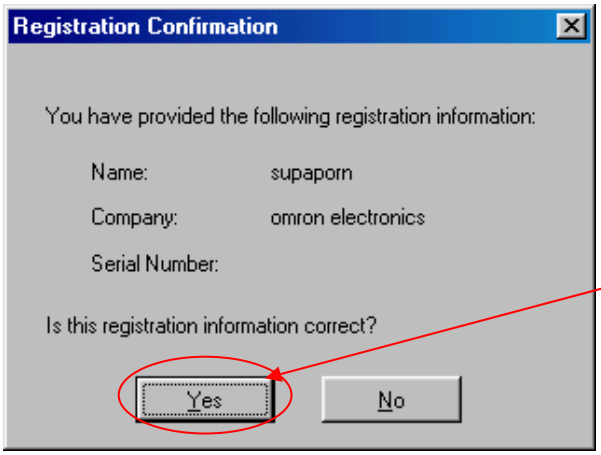

CX-Programmer Setup
CX-Programmer v3.0
Welcome
Welcome to the CX-Programmer Setup program. This program will install CX-Programmer on your computer.
It is strongly recommended that you exit all Windows programs before running this Setup program.
Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.
WARNING: This program is protected by copyright law and international treaties.
Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.
Next >
Cancel
6.เลือก NEXT


Software License Agreement
Please read the following License Agreement. Press the PAGE DOWN key to see the rest of the agreement.
IMPORTANT
By installing this package, you agree to be bound by the following Software License Agreement. If you do not agree, please return the enclosed software ("Software") without installing this package to the shop where you bought the Software.
The warranty service set forth in Section 7 of the Software License Agreement and any information on the Software and its revision and new version will not be provided to you,
as an user of the Software by the enclosed user registration card.
the card and send it to OMRON Corporation.
SOFTWARE LICENSE AGREEMENT
Do you accept all the terms of the preceding License Agreement? If you choose No, Setup will close. To install CX-Programmer, you must accept this agreement.
< Back
Yes
No
7.เลือก YES

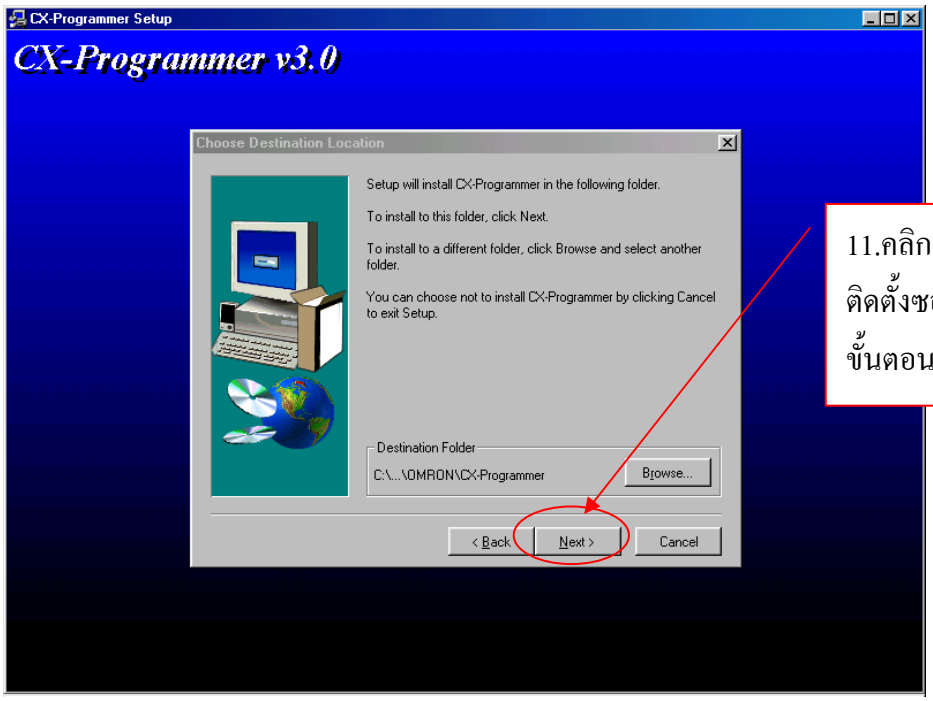


8.หลังจากนั้นจะให้เข้ามากำหนด License ของซอฟต์แวร์

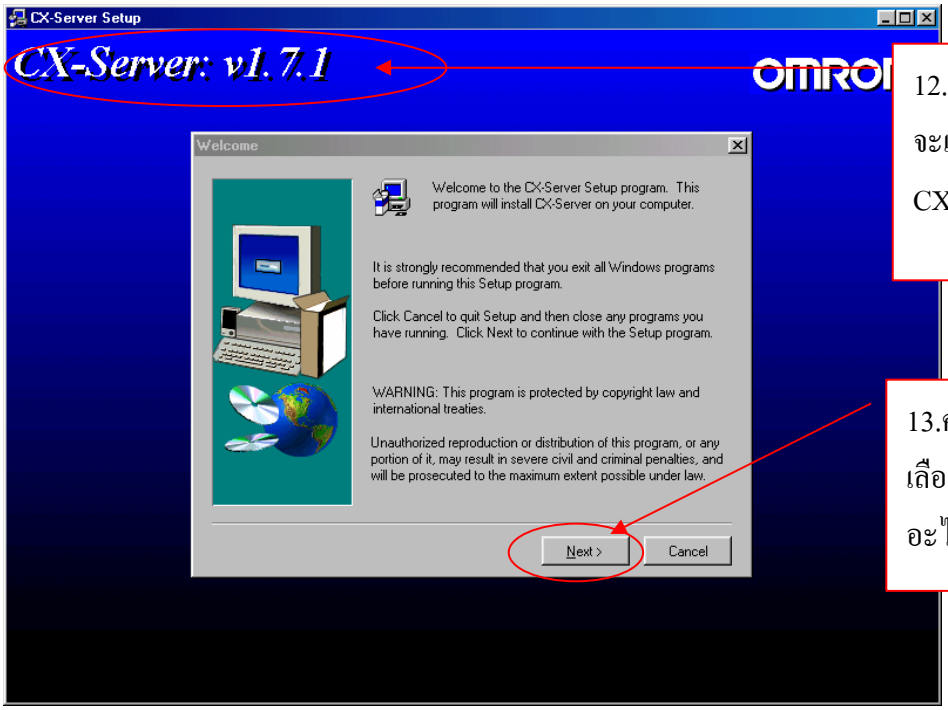
9.หลังจากใส่ License เรียบร้อยแล้วให้ คลิก Next จะปรากฏหน้าจอดังนี้



10.เลือก YES เพื่อ Install Program ตามขั้นตอน

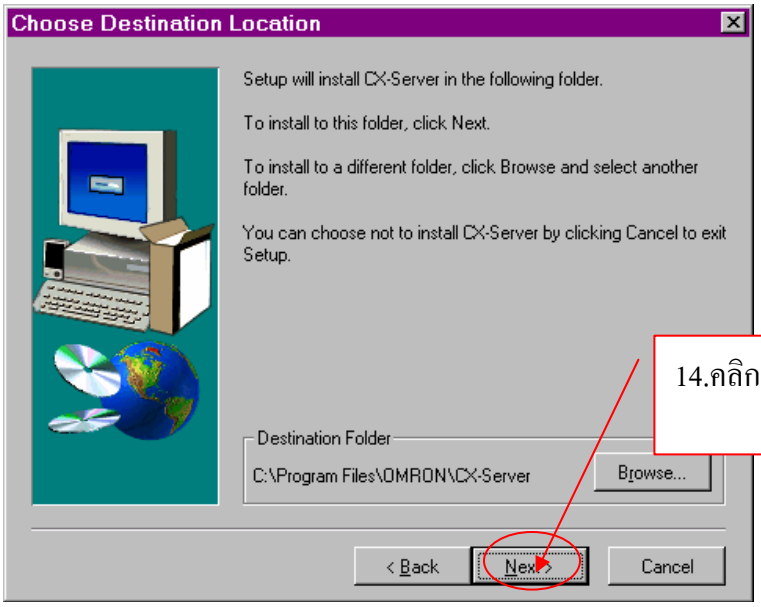


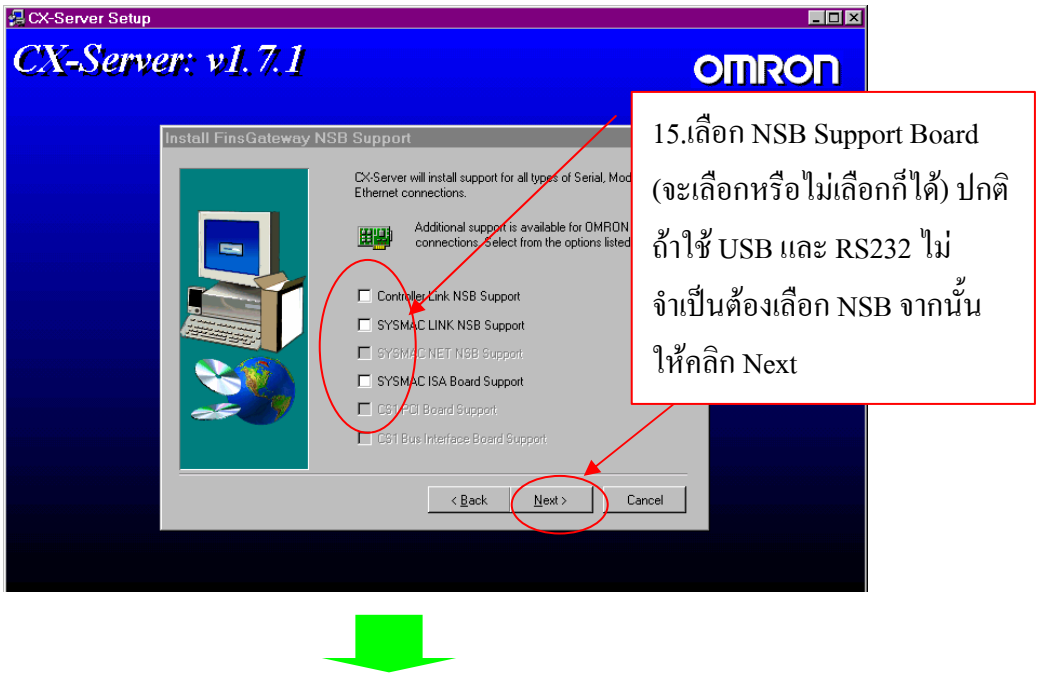
11.คลิก Next เพื่อติดตั้งซอฟต์แวร์ตามขั้นตอนต่อไป

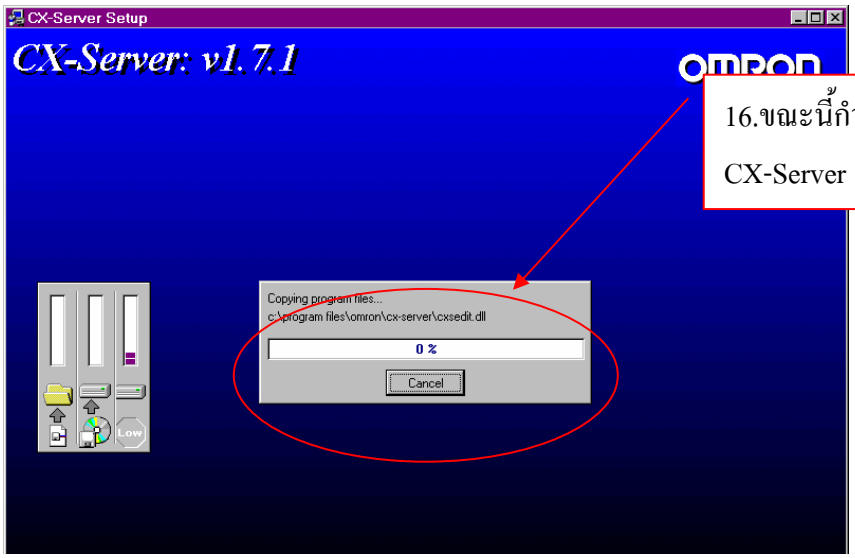


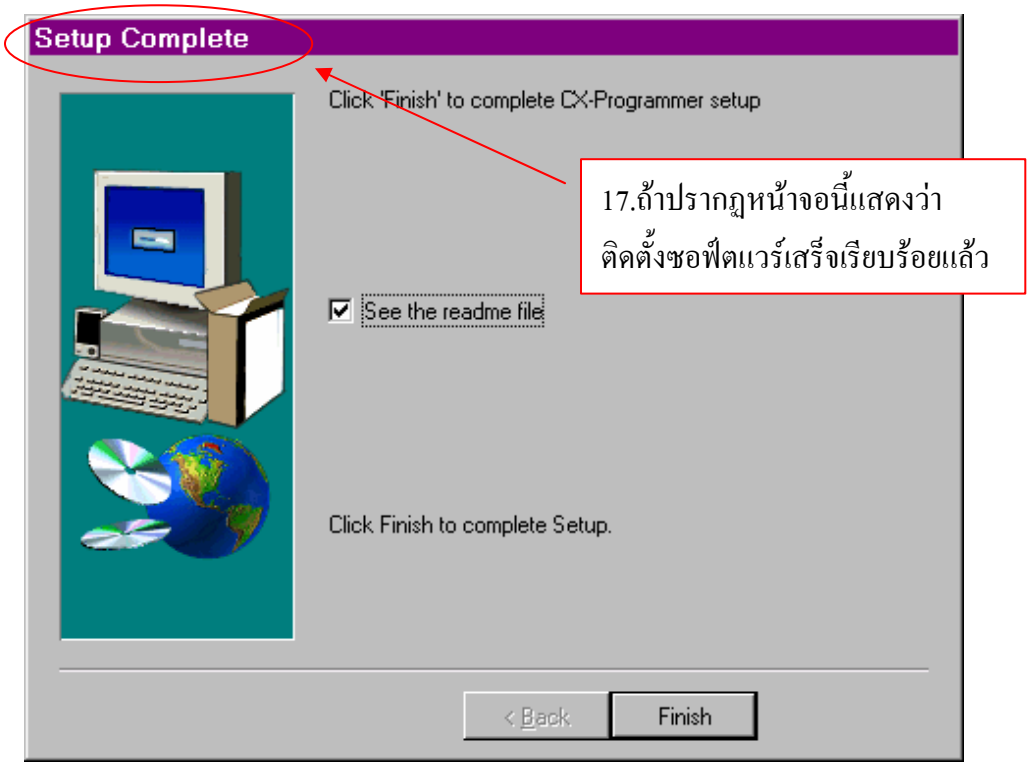
12.หลังจากนั้นจะเป็นการติดตั้ง CX-Server

13.คลิก Next เพื่อเลือกว่าจะติดตั้งอะไรบ้าง


Choose Destination Location
Setup will install CX-Server in the following folder.
To install to this folder, click Next.
To install to a different folder, click Browse and select another folder.
You can choose not to install CX-Server by clicking Cancel to exit Setup.
Destination Folder
C:\Program Files\OMRON\CX-Server
Browse...
< Back
Next >
Cancel
14.คลิก Next


CX-Server Setup
CX-Server: v1.7.1
OMRON
Install FinsGateway NSB Support
CX-Server will install support for all types of Serial, Mod
Ethernet connections.
Additional support is available for OMRON
connections. Select from the options listed
Controller Link NSB Support
SYSMAC LINK NSB Support
SYSMAC NET NSB Support
SYSMAC ISA Board Support
C91 PCI Board Support
C91 Bus Interface Board Support
< Back
Next >
Cancel
15.เลือก NSB Support Board (จะเลือกหรือไม่เลือกก็ได้) ปกติถ้าใช้ USB และ RS232 ไม่จำเป็นต้องเลือก NSB จากนั้นให้คลิก Next

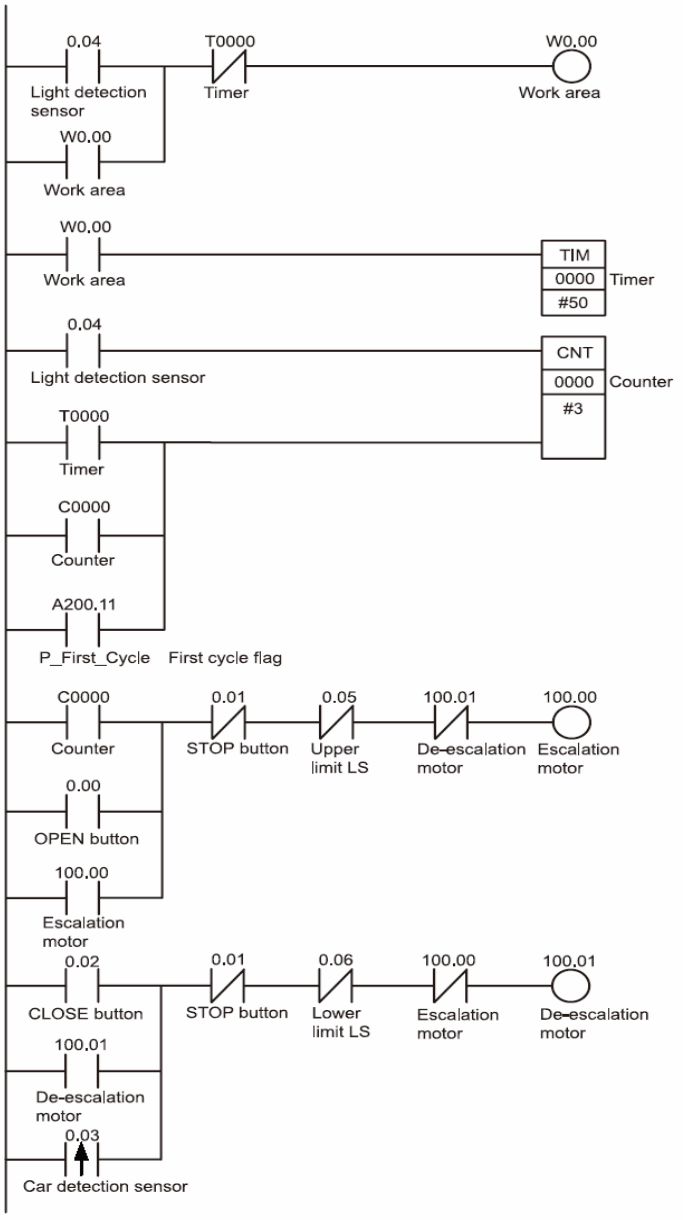




เมื่อหน้าต่างนี้ปรากฏขึ้นมาแสดงว่าการติดตั้งสมบูรณ์ จากนั้นเราสามารถเรียกใช้ซอฟ์ตแวร์ CX-programmer จาก Icon หรือ เมนูได้

6.2 การสร้างโปรแกรมแลดเดอร์

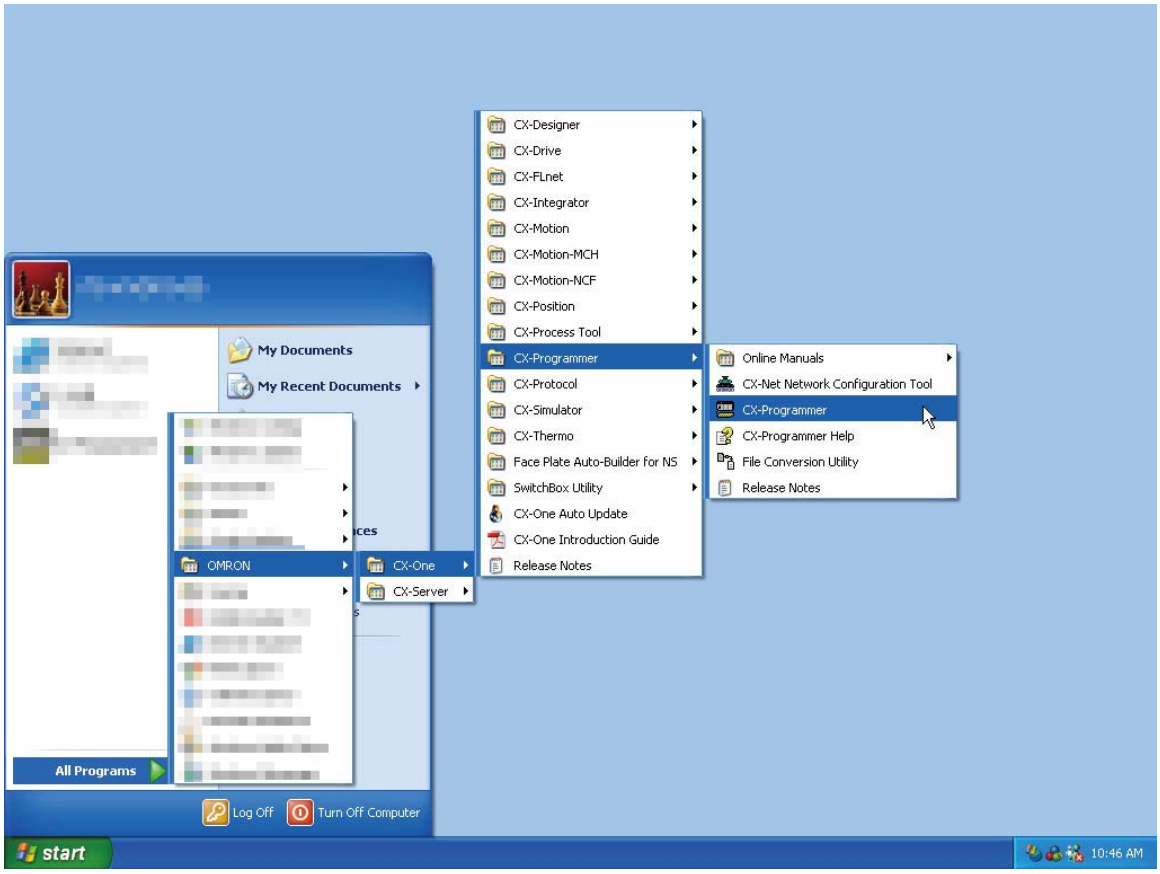
เพื่อให้เกิดความคุ้นเคยกับการใช้ซอฟต์แวร์ในการเขียนโปรแกรม PLC เราขอใช้ตัวอย่างแลดเดอร์ไดอะแกรมที่แสดงในบทที่ 2 มาประกอบการอธิบาย การเขียนโปรแกรมด้วย CX-programmer สามารถใช้ Mouse คลิกที่สัญลักษณ์ต่างๆ หรือใช้ Keyboard ก็ได้ แต่ในตัวอย่างนี้จะใช้ Keyboard เป็นตัวอย่าง



รูปที่ 6.1 ตัวอย่างโปรแกรมแลดเดอร์

6.2.1 การเปิดใช้ซอฟต์แวร์ CX-Programmer

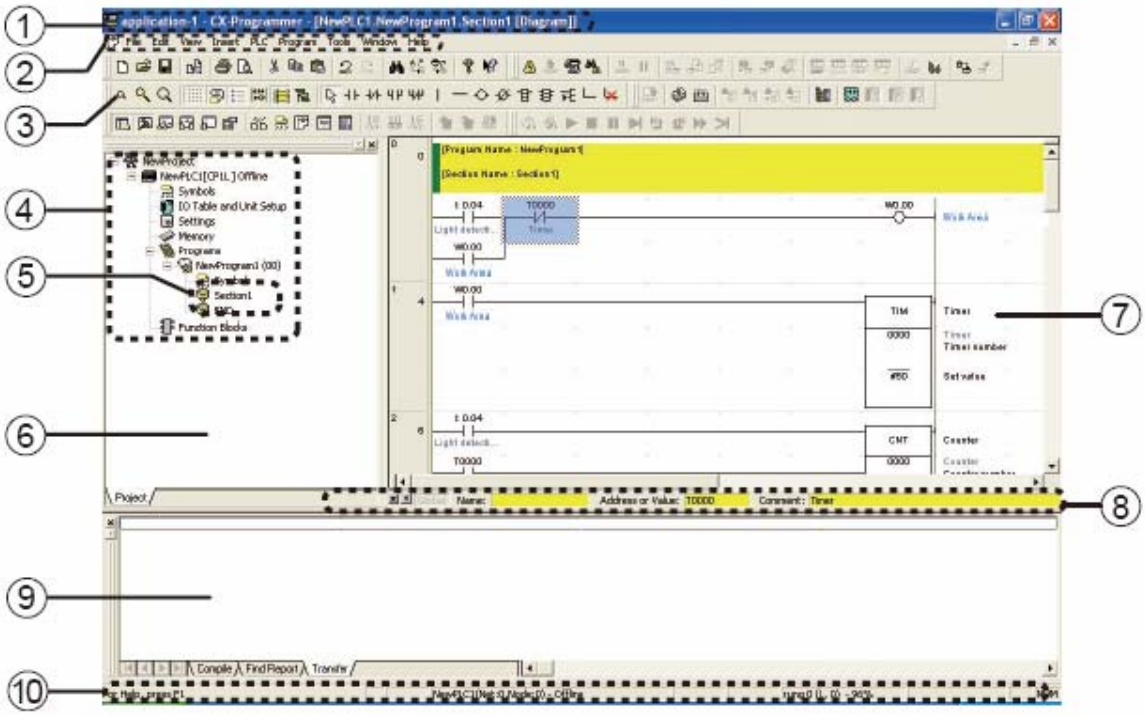
คลิกปุ่ม [start] ที่ task bar แล้วคลิก เลือก [program] → [OMRON]→[CX-One]→ [CX-Programmer] → [CX-Programmer] ดังรูป



6.2.2 คำอธิบายหน้าต่างและการใช้งาน

ในส่วนนี้จะอธิบายฟังก์ชันการใช้งานต่างๆ ของหน้าจอหลักของ CX-programmer

- **หน้าต่างหลัก (Main window)**



① Title bar

แสดงข้อมูลเกี่ยวกับชื่อไฟล์ที่ถูกสร้างโดย CX-programmer

② Main menu

ใช้เลือกฟังก์ชันต่างๆ ของ CX-programmer

③ Toolbars

แสดงไอคอนของฟังก์ชันที่ใช้งานบ่อย

④ Project tree/ (6) Project workspace

ใช้จัดการโปรแกรมและการตั้งค่าต่างๆ

⑤ Section

โปรแกรมที่เขียนขึ้นสามารถแยกเป็นส่วนๆ ได้

⑥ Diagram workspace

ใช้สร้างและแก้ไขแลดเดอร์โปรแกรม

⑦ I/O comment bar

ใช้แสดง name, address/value และ I/O comment ของตัวแปรที่ถูกเลือกโดย Mouse

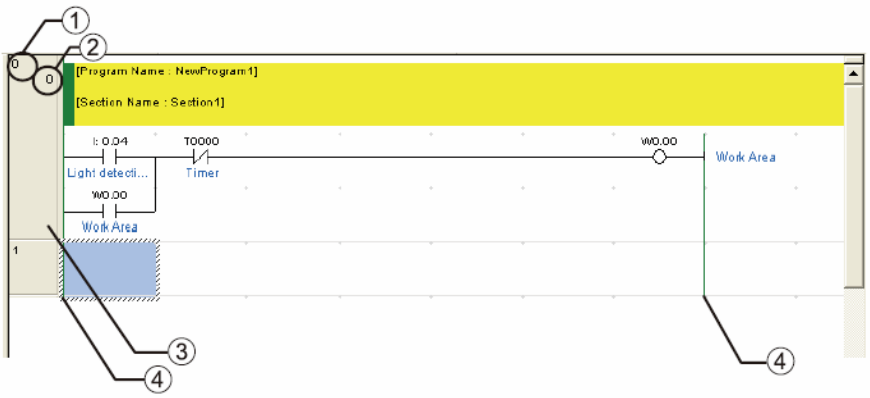
⑧ Output window

- ใช้แสดงผลของการตรวจสอบโปรแกรมที่เขียนขึ้น
- ใช้แสดงผลที่ได้จากการค้นหา contact instruction และ coil
- ใช้แสดงข้อผิดพลาดที่เกิดขึ้นขณะโหลดโปรแกรม

⑨ Status bar

แสดงข้อมูลต่างๆ เช่น PLC name และสถานะการ offline/online

- **พื้นที่สำหรับเขียนโปรแกรม (Diagram workspace)**



① Rung number

② Program address

③ Rung header

④ Bus bar

- **Shortcut Key**

แสดง Shortcut Key ที่ใช้ใน CX-programmer เวลาเขียนโปรแกรมเราสามารถกดปุ่มเหล่านี้หรือใช้ Mouse คลิกที่รูปบน Tool Bar ก็ได้ เช่น ถ้าต้องการใช้หน้าคอนแทค NO ให้กดปุ่ม **[C]**

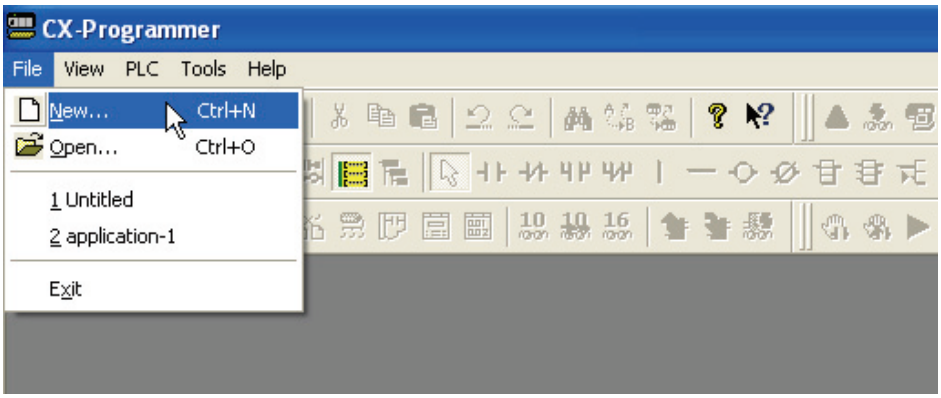
6.3 การป้อนโปรแกรม

ใช้แลดเดอร์ไดอะแกรมที่แสดงในรูปที่ 6.1 เพื่อเป็นตัวอย่างในการสร้างโปรแกรม

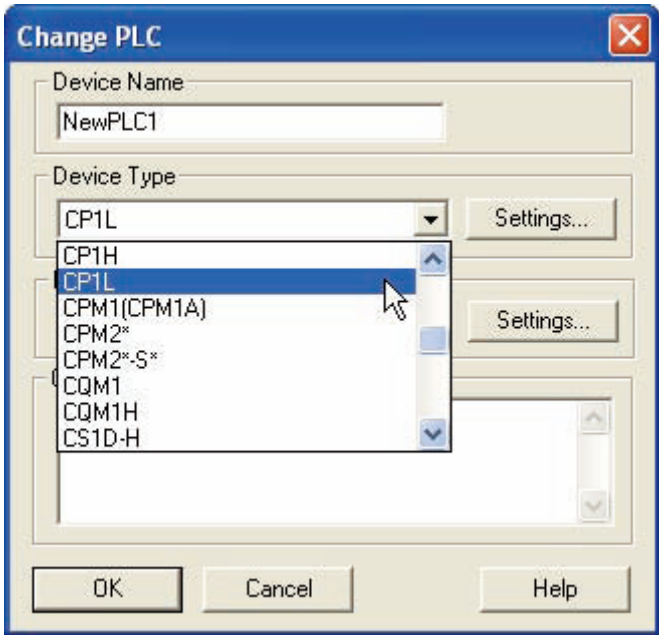
6.3.1 การสร้างโปรเจคใหม่

เมื่อเรียกใช้ CX-programmer เป็นครั้งแรกคุณจะต้องสร้างโปรเจคขึ้นมาใหม่ ตอนสร้างโปรเจคใหม่คุณจะต้องเลือก Device Type และ CPU ของโปรแกรมที่กำลังสร้างขึ้น โดยมีขั้นตอนดังต่อไปนี้

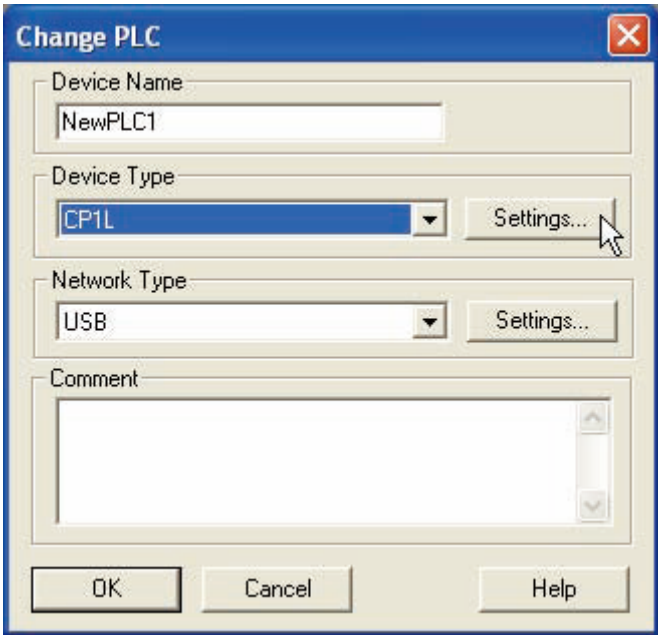
1. เลือก [File] - [New] จาก Main menu จะปรากฏ Dialog box ดังรูปข้างล่างนี้



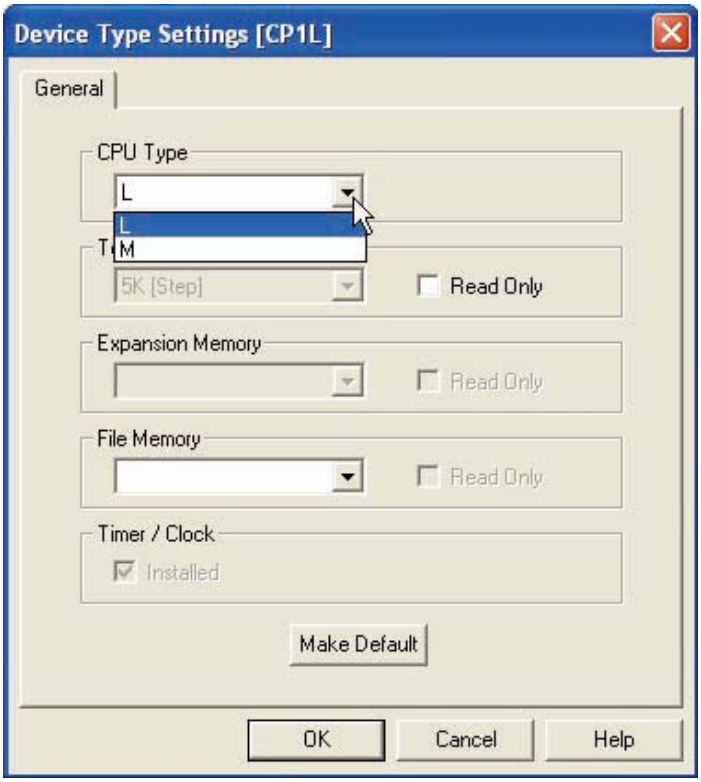
2. เลือก [CP1L] จาก Device Type



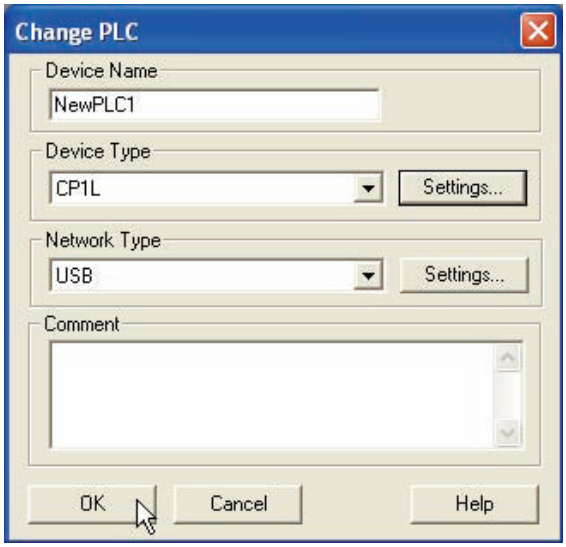
3. คลิก [Settings] ไดอะลอกบล๊อก “Device Type Settings” จะปรากฏดังต่อไปนี้



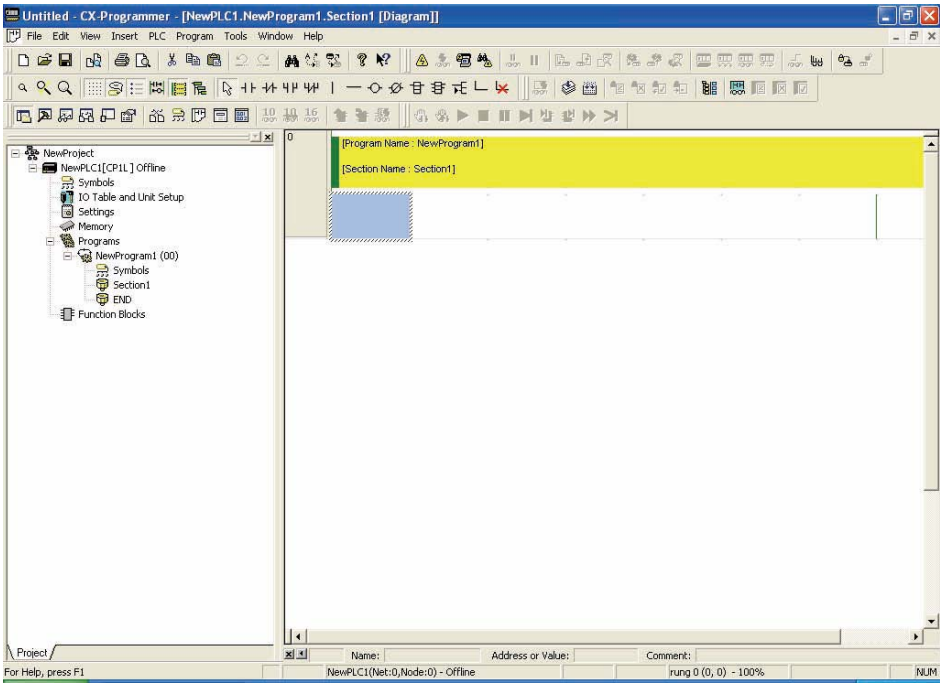
4. เลือกรุ่น CPU จาก CPU Type จากนั้นคลิก [OK] ไดอะลอกบล๊อก “Device Type Settings” จะปิดลง



5. ตรวจสอบด้วยว่า Network Type เป็น [USB] จากนั้นคลิก [OK] ถ้า PLC ไม่ใช่รุ่น CP1L/CP1H การเลือก Network Type ต้องขึ้นอยู่กับ PLC รุ่นนั้นๆ เช่น Toolbus หรือ Hostlink



เมื่อคลิก [OK] แล้วไดอะลอกบล๊อก “Change PLC” จะปิดลงและแสดงหน้าต่าง Main window ดังรูปข้างล่างนี้



<u>หมายเหตุ</u> ถ้าไม่สามารถเลือก [USB] ที่ Network Type ได้ ให้ติดตั้ง USB Driver ของ CP1L

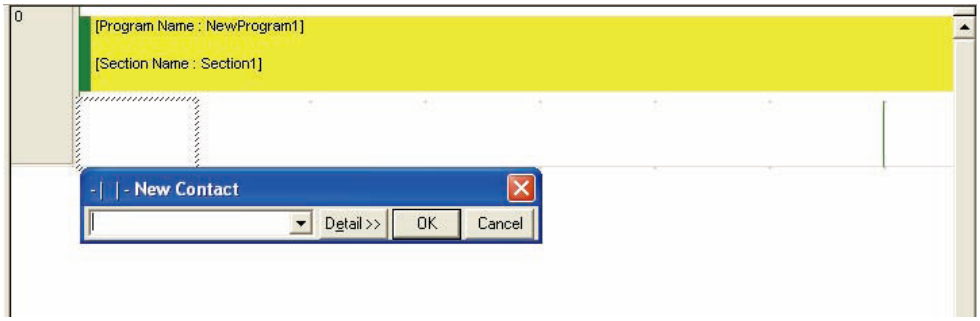
6.3.2 การป้อนคอนแทค

เราขออ้างอิงรูปวงจรแลดเดอร์โปรแกรมในรูปที่ 6.1 ส่วนการอธิบายการป้อนโปรแกรมจะเน้นการใช้ Shortcut Key เป็นหลัก

- **การป้อนคอนแทค NO**

1. กดคีย์ [C] หรือใช้ Mouse คลิกที่รูป 

แล้วเอามาวางที่ Workspace ไดอะลอกบล๊อก ‘New Contact’ จะปรากฏออกมา



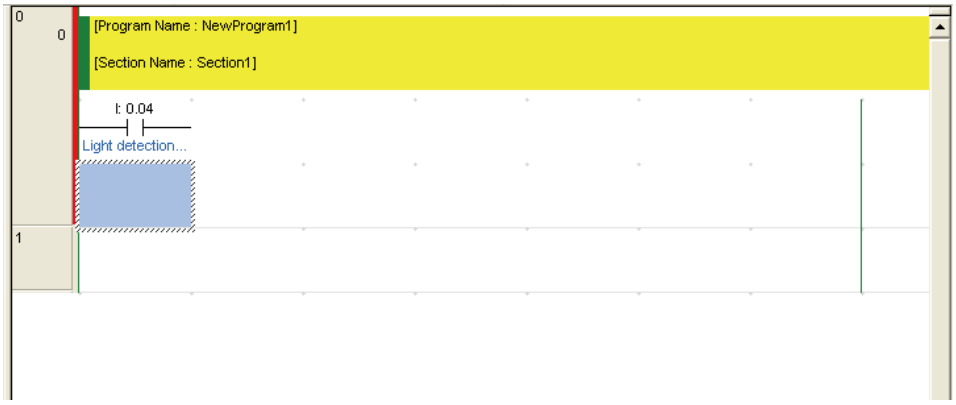
2. ป้อนแอดเดรส 0.04 ด้วยการกดปุ่มเลข "4" แล้วกด [Enter]
เมื่อ “4” ถูกป้อนเข้าไปแล้ว ไดอะลอกบล๊อก ‘Edit Comment’ จะแสดงออกมา

3. ป้อน “Light detection sensor” เป็น I/O comment แล้วกด [Enter]
คอนแทคที่ป้อนพร้อมคอมเมนต์ Light detection sensor จะปรากฏอยู่ในโปรแกรม

ขั้นตอนถัดไป ป้อนวงจร OR

- **การป้อนวงจร OR**

1. ที่ตำแหน่ง cursor ปัจจุบัน ให้กดปุ่ม [Enter]
   จะเกิดพื้นที่ว่างใหม่ขึ้นมาเพื่อสร้างวงจร OR ดังแสดงในรูปข้างล่างนี้



2. กดคีย์ [W]หรือใช้ Mouse คลิกที่รูป 

   ไดอะลอกบล๊อก ‘New Contact OR’ จะปรากฏดังรูปข้างล่างนี้

3. ป้อนแอดเดรส W0.00 ด้วยการคีย์ “W0” จากนั้นกด [Enter]
   เมื่อ “W0” ถูกป้อนเข้าไปแล้วไดอะลอกบล๊อก ‘Edit Comment’ จะแสดงออกมา

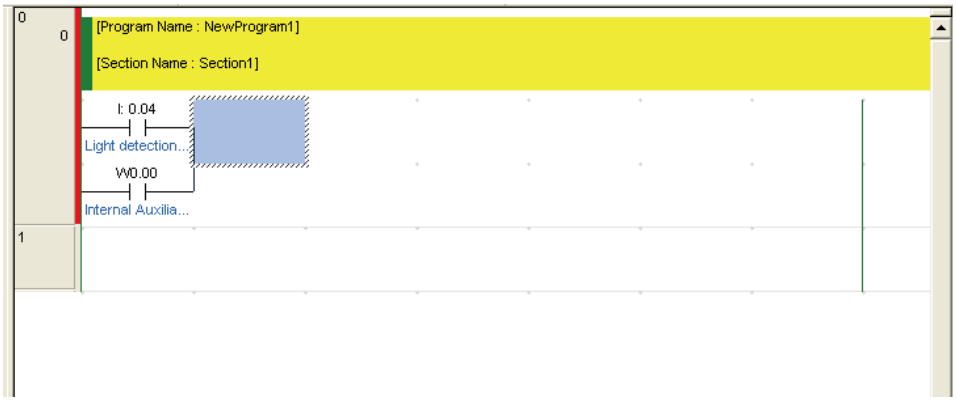
4. ป้อน "Work Area" เป็น I/O comment จากนั้นกด [Enter]
   วงจร OR จะปรากฏออกมา

ขั้นตอนถัดไป ป้อนคอนแทค NC (normally close)

- **การป้อนคอนแทค NC**

1. กดคีย์ลูกศรขึ้น (up arrow)

   Cursor จะถูกเลื่อนขึ้นข้างบนดังแสดงในรูป



2. ที่ตำแหน่ง Cursor ปัจจุบันให้กดคีย์ [/]

   ไดอะลอกบล๊อก ‘New Closed Contact’ จะปรากฏขึ้น

3. ป้อนแอดเดรส Timer หมายเลข 0 โดยการกด "T0" แล้วกด [Enter]

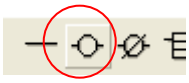
   “T0” ที่ถูกป้อนแล้วจะแสดงไดอะลอกบล๊อก ‘Edit Comment’ ตามรูปข้างล่างนี้

4. ป้อน "Timer" เป็น I/O comment จากนั้นกด [Enter]

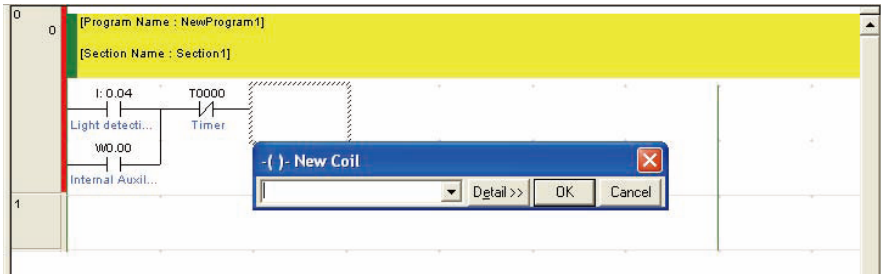
   วงจร AND ที่เป็นตัวแทนของ Timer หน้าคอนแทค NC จะปรากฏออกมา

ขั้นตอนถัดไป ป้อนเอาต์พุต Work area

6.3.3 การป้อนคอล์ยเอาต์พุต

1. กดคีย์ [O]หรือคลิก 

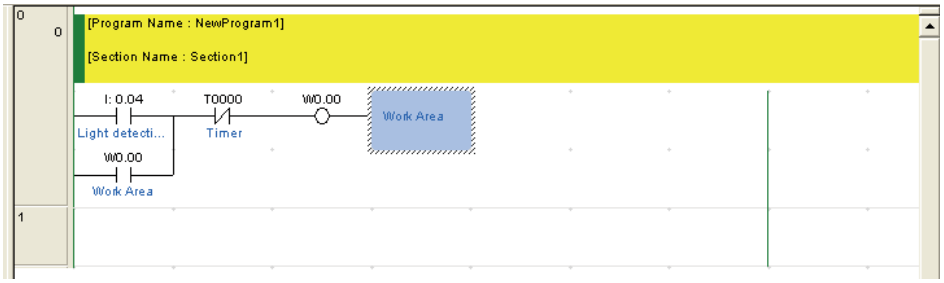
   ไดอะลอกบล๊อก ‘ New Coil’ จะปรากฏออกมา



2. ป้อนแอดเดรส W0.00 โดยการคีย์ "W0" จากนั้นกด [Enter]

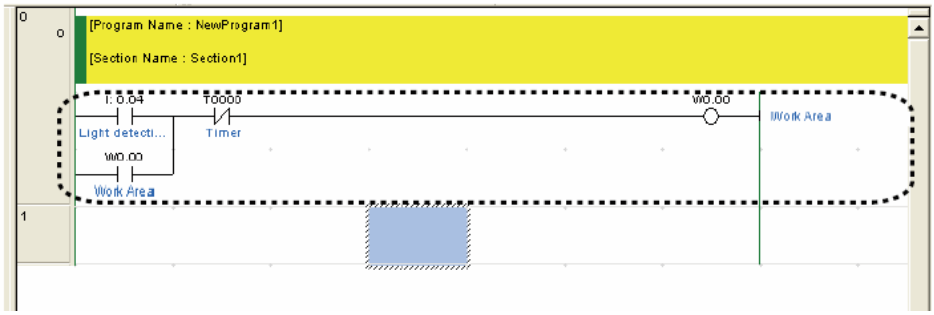
   เมื่อ "W0" ถูกป้อนไดอะลอกบล๊อก ‘Edit Comment’ จะปรากฏออกมา

3. ป้อน comment “Work Area” จากนั้นกด [Enter]



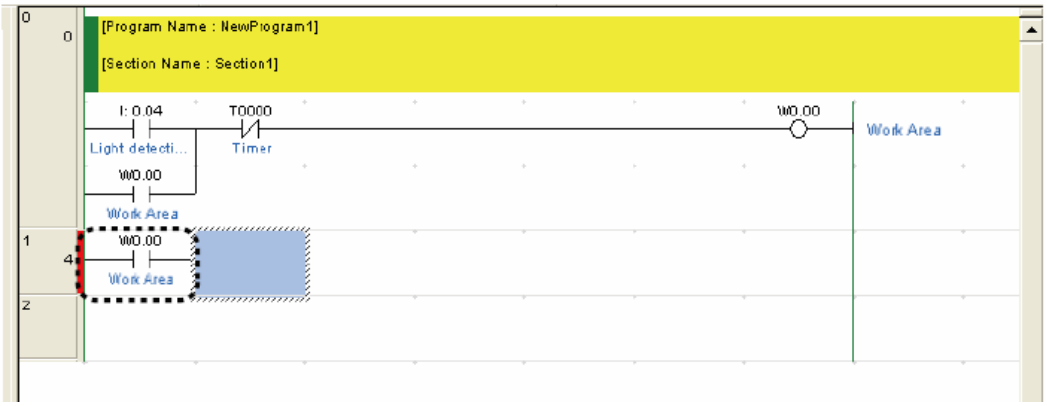
4. กดปุ่มลูกศรลง (down arrow) 2 ครั้ง

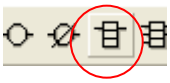
   Cursor จะถูกวางใน Rung ถัดไป การป้อนวงจรใน Rung แรกเสร็จสมบูรณ์



6.3.4 การป้อน Timer

1. เลื่อน Cursor ไปติดกับ Busbar จากนั้นกดคีย์ [C] แล้วป้อน "W000" จากนั้นกด [Enter] ขณะเดียวกันไดอะลอกบล๊อก 'Edit Comment' จะปรากฏออกมา จากนั้นทำตามขั้นตอนที่กล่าวมาแล้วข้างต้นเพื่อป้อนหน้าคอนแทค

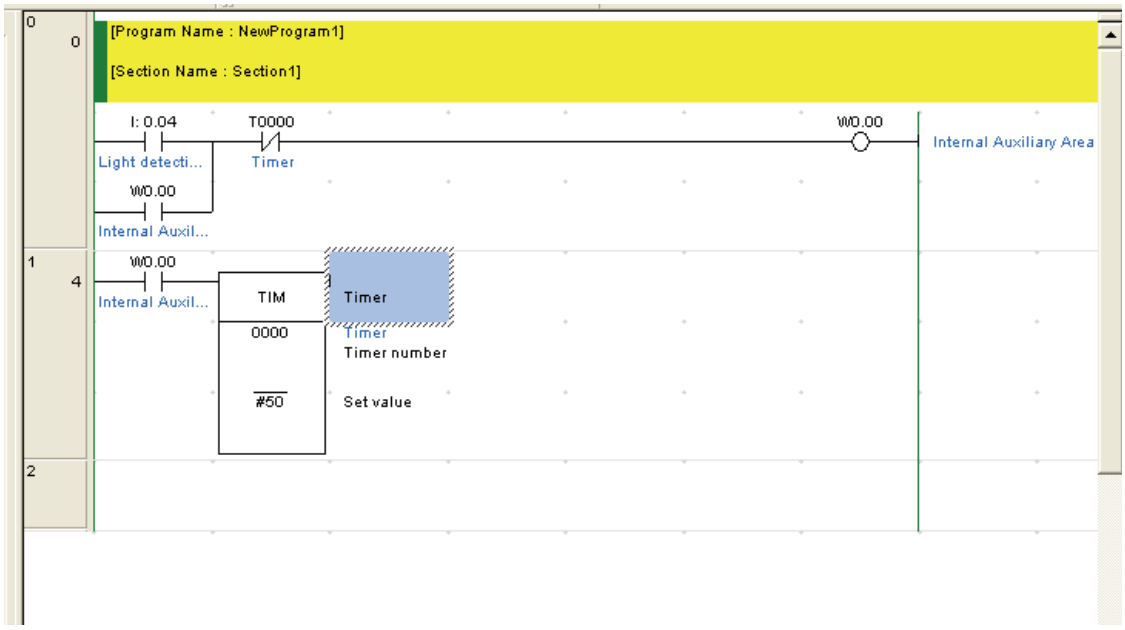


2. กดคีย์ [I] หรือคลิก  ไดอะลอกบล๊อก 'New Instruction' จะปรากฏขึ้น

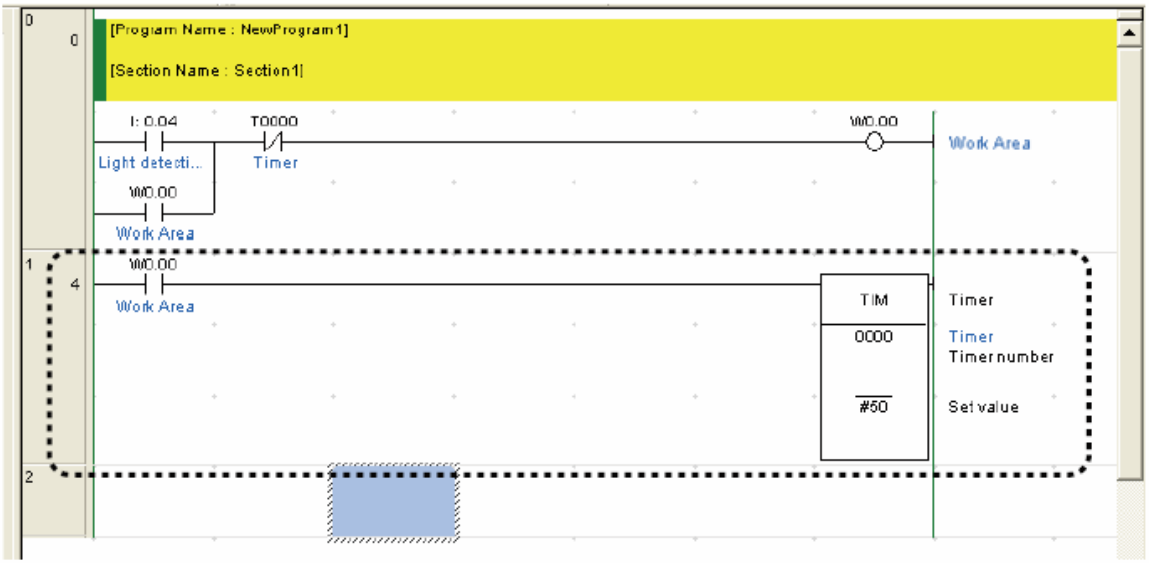
3. ป้อนคำสั่ง Timer โดยการพิมพ์ "TIM 0 #50" จากนั้นกด [Enter]
เมื่อ "TIM 0 #50" ถูกป้อน ไดอะลอกบล๊อก 'Edit Comment' จะปรากฏขึ้นเพื่อให้ป้อน I/O Comment
"TIM 0 #50" หมายถึงหน่วงเวลา 5.0 วินาที เมื่อครบเวลาที่ตั้งไว้ T0000 จะ On

4. ป้อน "Timer" ในไดอะลอกบล๊อก 'Edit Comment' จากนั้นกด [Enter]

คำสั่ง Timer จะปรากฏในโปรแกรมแลดเดอร์



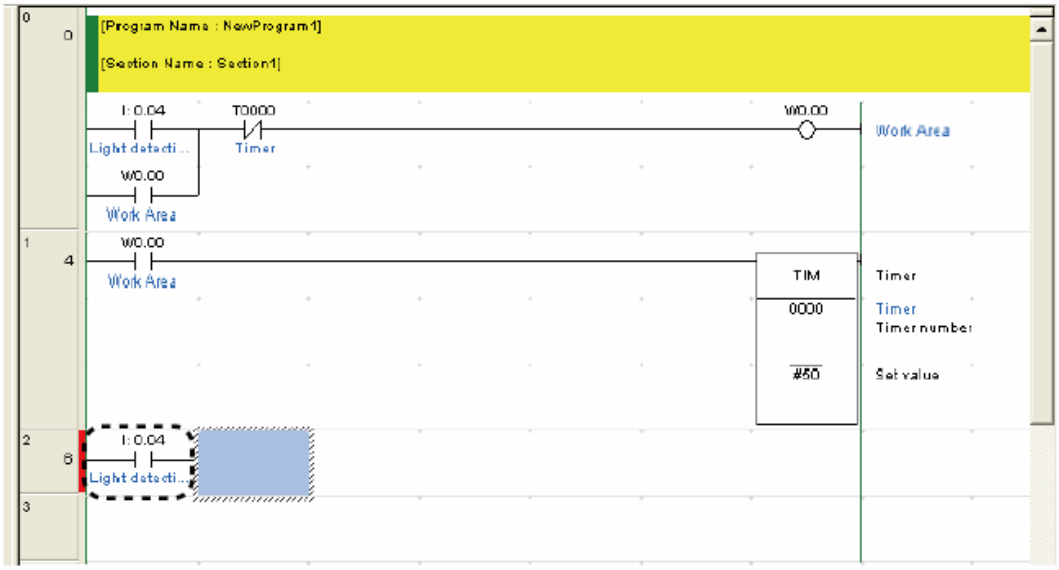
5. กดปุ่มลูกศรลง (down arrow) 3 ครั้ง
เมื่อ Cursor ถูกวางใน Rung ถัดไปการป้อนคำสั่งใน Rung 01 เสร็จสมบูรณ์

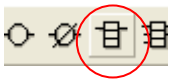


ขั้นตอนถัดไป ป้อนคำสั่ง Counter

6.3.5 การป้อน Counter

1. เลื่อน Cursor ไปติดกับ Busbar จากนั้นกดคีย์ [C] แล้วป้อน "004" จากนั้นกด [Enter] ขณะเดียวกันไดอะลอกบล๊อก 'Edit Comment' จะปรากฏออกมา จากนั้นทำตามขั้นตอนที่กล่าวมาแล้วข้างต้นเพื่อป้อนหน้าคอนแทค



2. กดคีย์ [I] หรือคลิก  ไดอะลอกบล๊อก 'New Instruction' จะปรากฏขึ้น

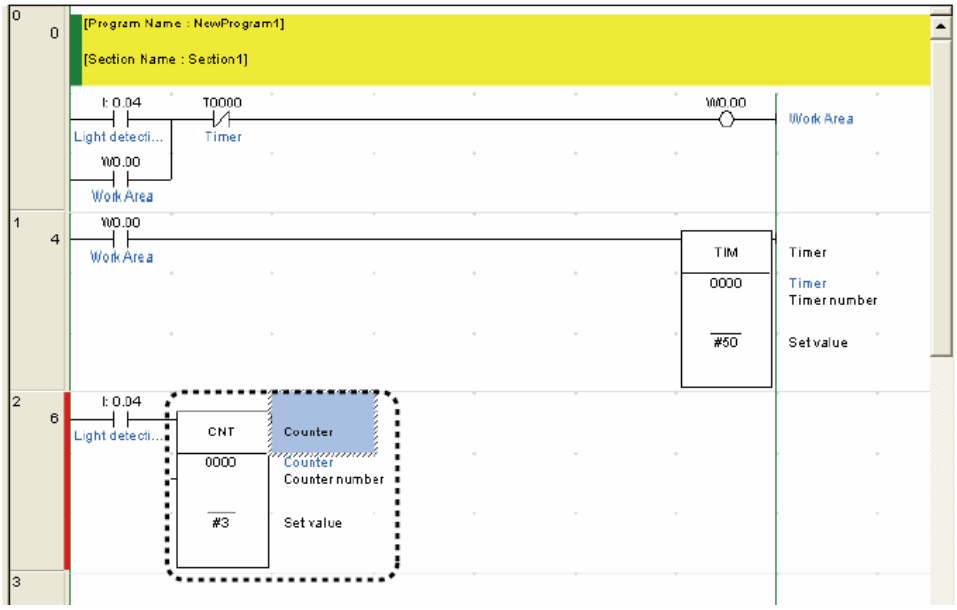
3. ป้อนคำสั่ง Counter โดยพิมพ์ "CNT 0 #3" จากนั้นกด [Enter]

เมื่อ "CNT 0 #3" ถูกป้อนไดอะลอกบล๊อก 'Edit Comment' จะปรากฏขึ้นเพื่อให้ป้อน I/O Comment

"CNT 0 #3" คือ counter ที่เริ่มต้นนับลงจาก 3 เมื่อครบตามที่ตั้งไว้ C0000 จะ On

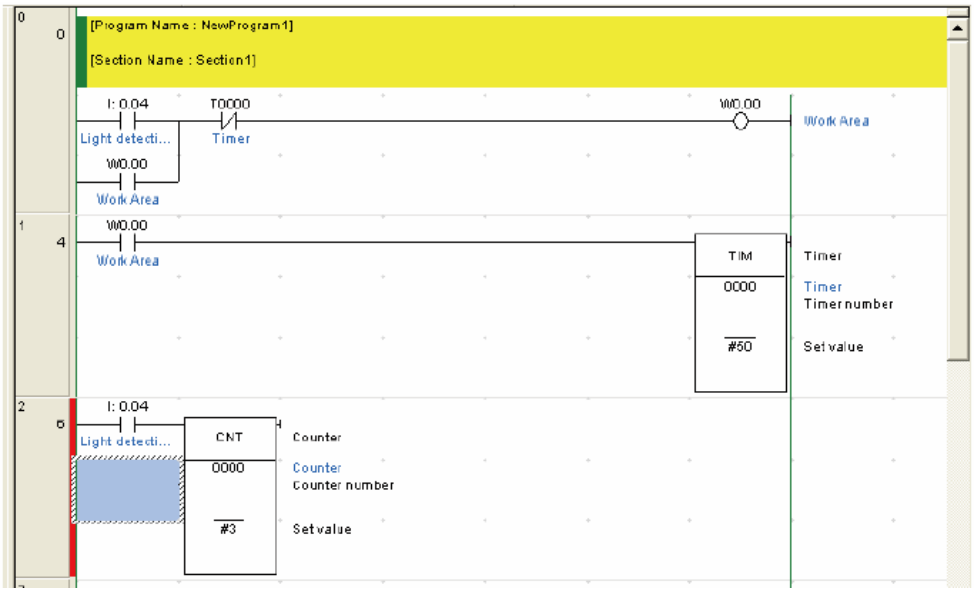
4. ป้อน Comment โดยการพิมพ์ "Counter" จากนั้นกด [Enter]

คำสั่ง Counter จะปรากฏในโปรแกรมแลดเดอร์

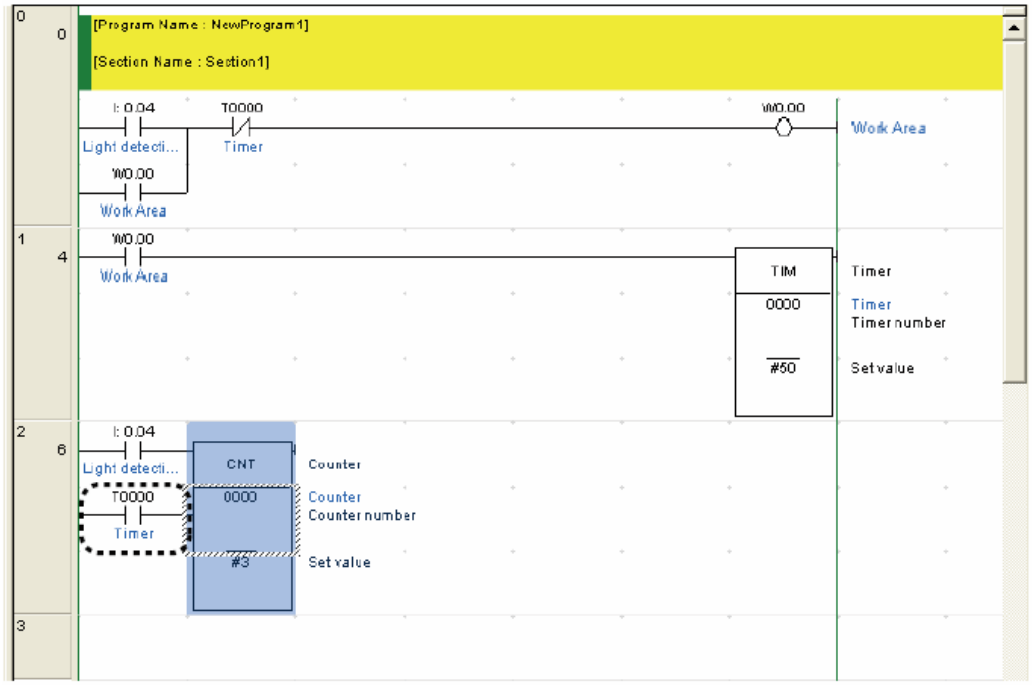


ต่อจากนั้นให้ป้อนขาสัญญาณ Reset ของคำสั่ง Counter โดยใช้หน้าคอนแทค Timer (T0000)

5. วาง Cursor ต่ำกว่าหน้าคอนแทคที่เพิ่งสร้างขึ้นใน step 1

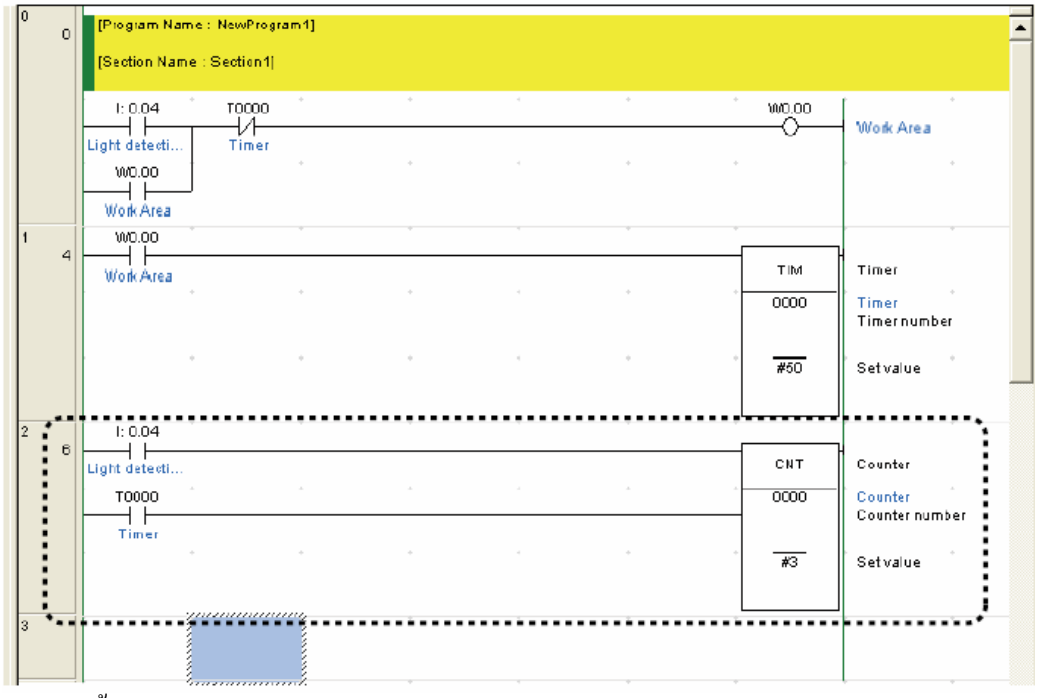


6. ป้อนหน้าคอนแทค "T0000"



7. กดลูกศรลง (down arrow) 2 ครั้ง

เมื่อ Cursor ถูกวางใน rung ถัดไป การป้อนคำสั่ง Counter เป็นอันเสร็จสมบูรณ์

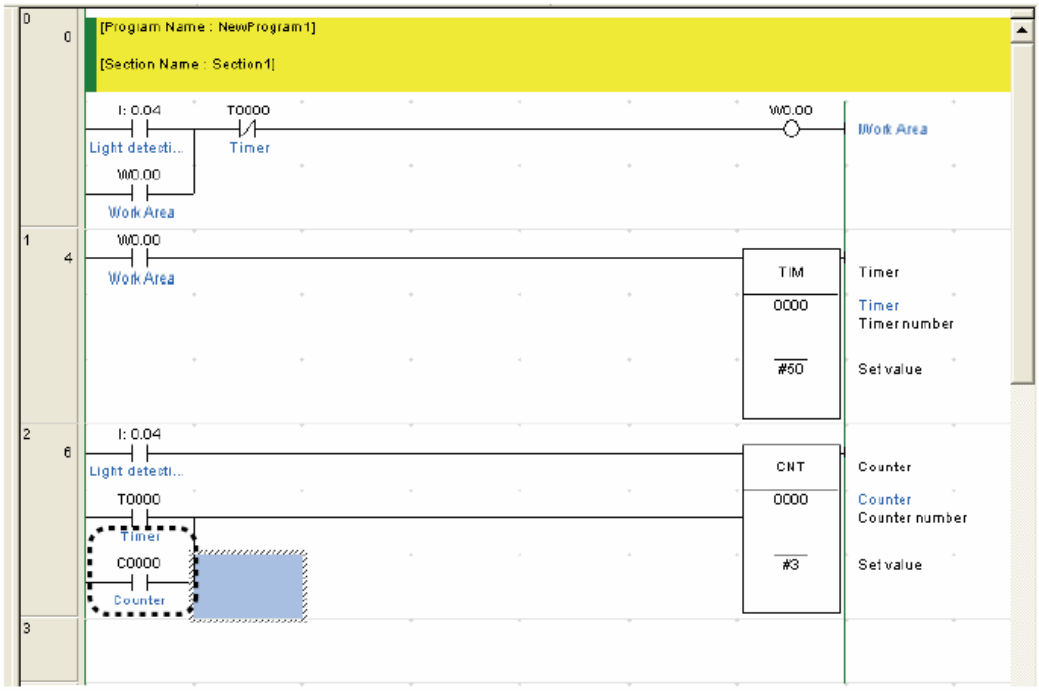


ขั้นตอนถัดไป ป้อน Auxiliary Area

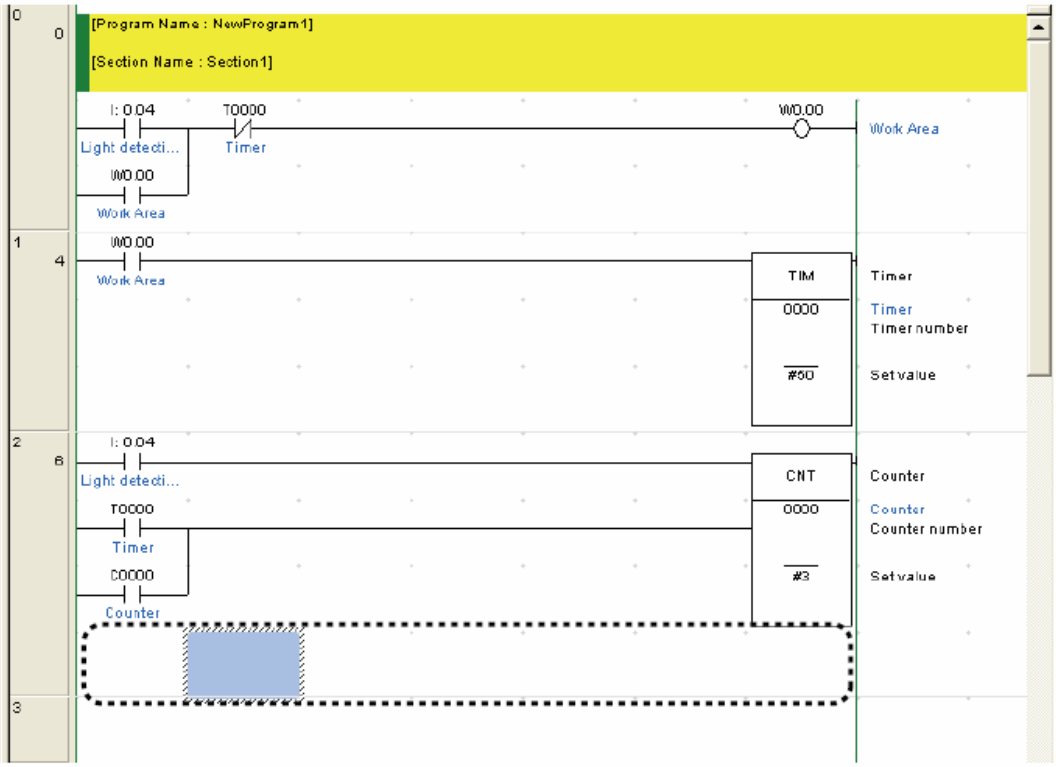
6.3.6 การป้อน Auxiliary Area

Auxiliary area คือ รีเลย์ที่ออกแบบมาให้ใช้งานเฉพาะ ตัวอย่างเช่น 'First cycle flag' ซึ่งจะ ON เพียง 1 cycle เท่านั้นหลังจากจ่ายไฟให้กับ PLC ในตัวอย่างที่เราเขียนโปรแกรมอยู่นี้ 'First cycle flag' จะถูกใช้รีเซ็ต Counter เมื่อตอนจ่ายไฟให้ PLC

1. กดคีย์ [W] จากนั้นป้อนวงจร OR หน้าคอนแทค "C0000" ต่อจากนั้นกด [Enter] ขณะเดียวกันไดอะลอกบล๊อก 'Edit Comment' จะปรากฏขึ้น (การป้อนหน้าคอนแทคได้กล่าวถึงไปแล้วในหัวข้อก่อนหน้านี้)



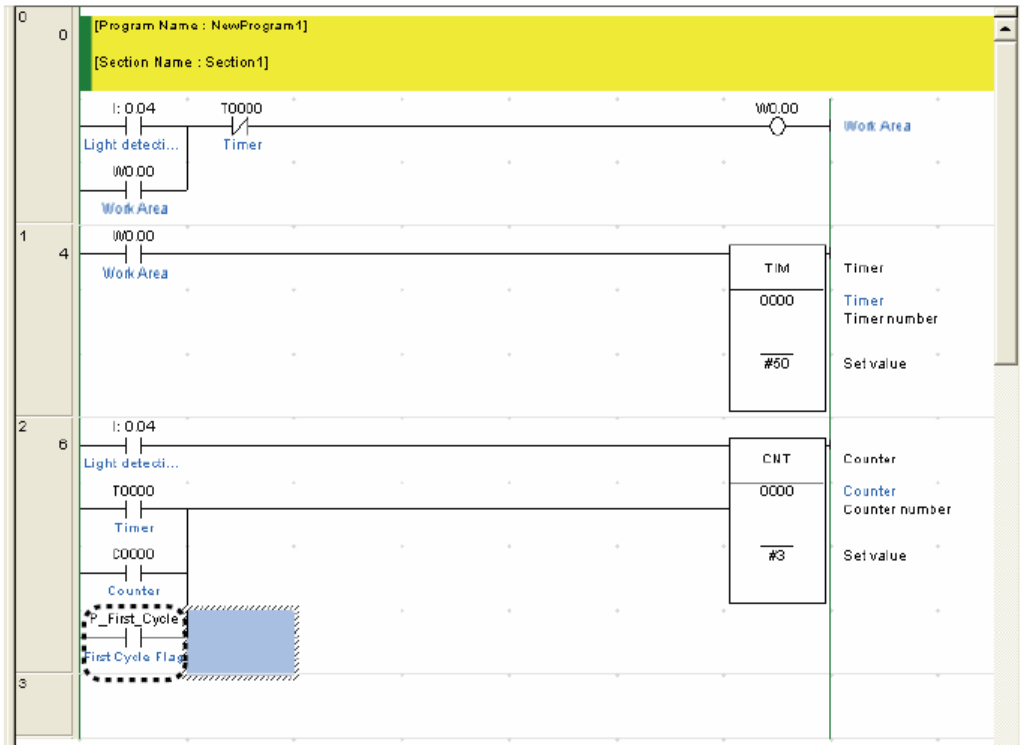
2. กด [Enter]
   จะเกิดพื้นที่ว่างข้างใต้วงจร OR ที่เพิ่งถูกสร้างขึ้น ดังรูปข้างล่างนี้



3. กดคีย์ลูกศรซ้ายมือ (left arrow)
4. กดคีย์ [W]

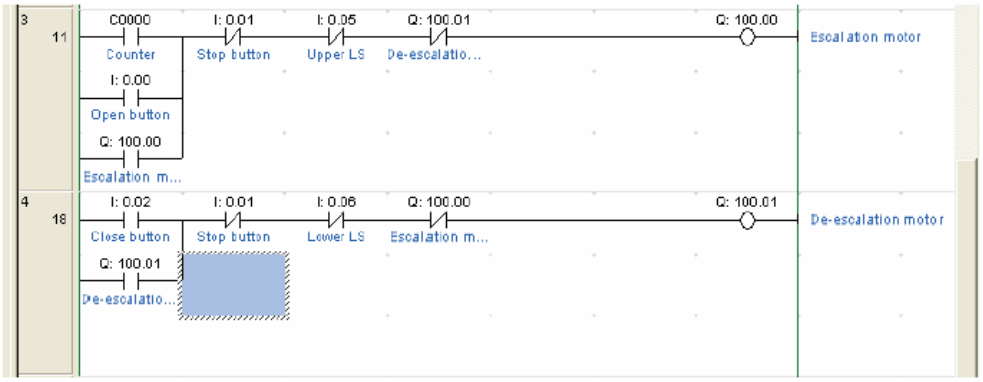
   ไดอะลอกบล๊อก ‘New Contact OR’ จะปรากฏขึ้น
5. ป้อนแอดเดรส "A20011" จากนั้นกด [Enter]

‘First cycle flag’ จะปรากฏบนโปรแกรมแลดเดอร์ดังรูปข้างล่างนี้



6.3.7 การป้อนคอนแทค Differentiated Up

1. ให้อ้างอิงโปรแกรมแลดเดอร์ในรูปที่ 6.1 จากนั้นให้ป้อนโปรแกรมตามขั้นตอนที่อธิบายข้างต้นจนถึงหน้าคอนแทค De-escalation motor contact แอดเดรส "10001" ดังแสดงในรูปข้างล่างนี้



2. กด [Enter]
จะเกิดพื้นที่ว่าง เพื่อใส่วงจร OR

3. กดคีย์ [W]

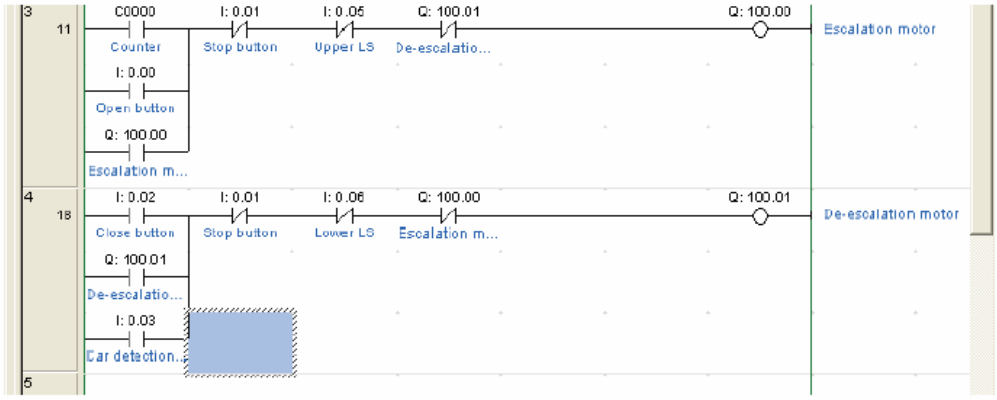
   ไดอะลอกบล๊อก ‘New Contact OR’ จะปรากฏขึ้น

4. ป้อนแอดเดรส 0.03 โดยการคีย์ "3" แล้วกด [Enter]

   ไดอะลอกบล๊อก ‘Edit Comment’ จะปรากฏขึ้น

5. ป้อน Comment โดยการคีย์ "Car detection sensor" จากนั้นกด [Enter]

   หน้าคอนแทค Car detection sensor จะปรากฏขึ้นเป็นวงจร OR



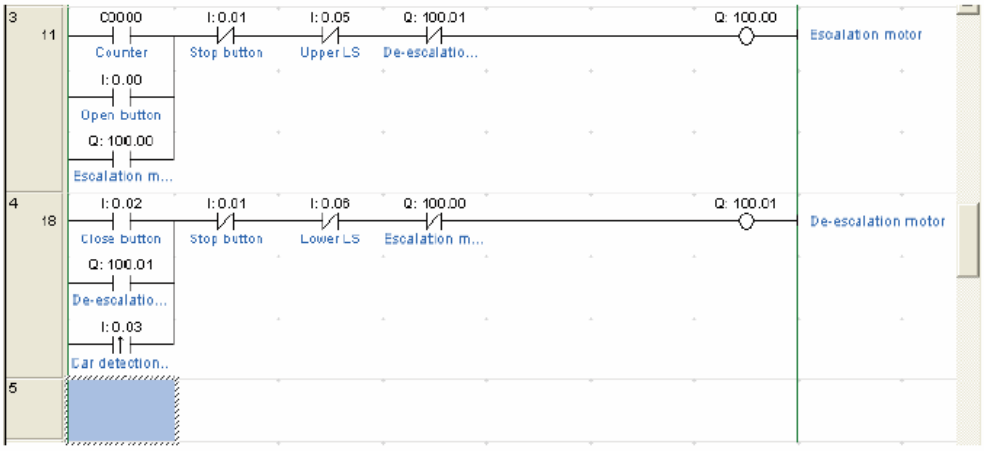
6. ให้ Double-click ที่แอดเดรส "0.03"

   ไดอะลอกบล๊อก ‘Edit Contact’ จะปรากฏขึ้น

7. คลิกที่ [Detail]

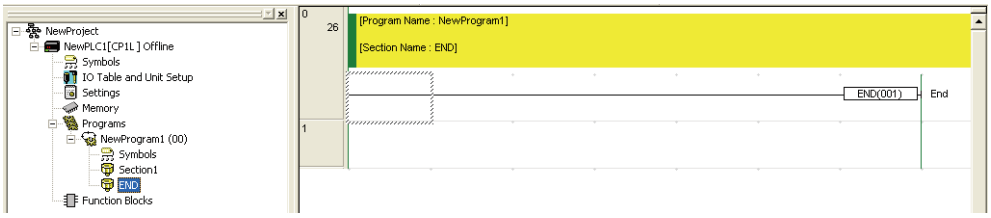
8. เลือก [Up] จากนั้นคลิก [OK]

หน้าคอนแทคที่มีลูกศรขึ้นเป็นสัญลักษณ์แสดง Differentiated Up จะปรากฏออกมา



6.3.8 การป้อนคำสั่ง END

โปรแกรมแลดเดอร์จำเป็นต้องจบด้วยคำสั่ง END แต่ CX-programmer จะสร้างส่วนที่เป็นคำสั่ง END ขึ้นเองอัตโนมัติ ถ้าต้องการดูคำสั่ง END ให้ Double-click ที่ส่วนของ END ดังรูป



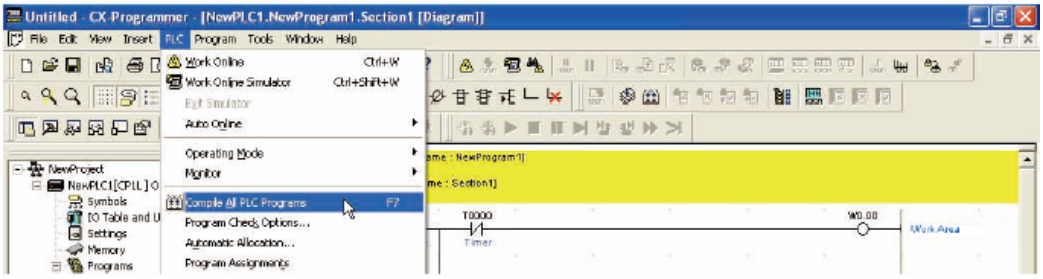
6.4 การโหลดและจัดเก็บโปรแกรม (Loading/Saving)

โปรแกรมที่สร้างขึ้นอาจต้องจัดเก็บ (Save) ซึ่งในหัวข้อนี้จะอธิบายวิธีการจัดเก็บ และเรียก (Load) โปรแกรมกลับขึ้นมาใช้อีก

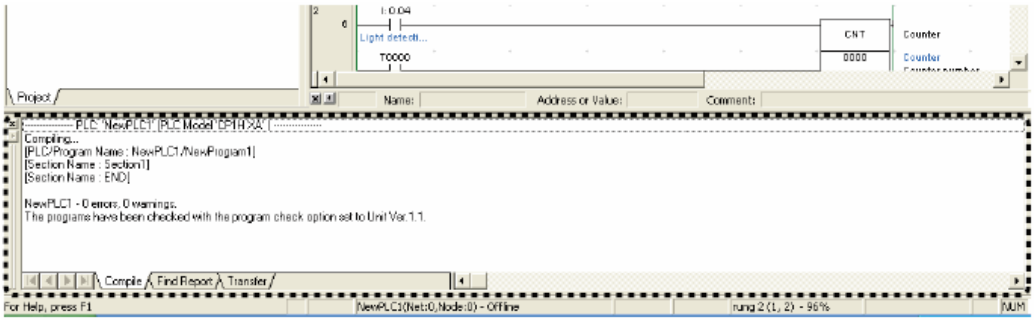
6.4.1 การ Compile โปรแกรม

การ Compile จะช่วยให้เราสามารถตรวจสอบข้อผิดพลาดในโปรแกรมได้

1. เลือก [PLC] - [Compile All PLC Programs] จาก Main menu

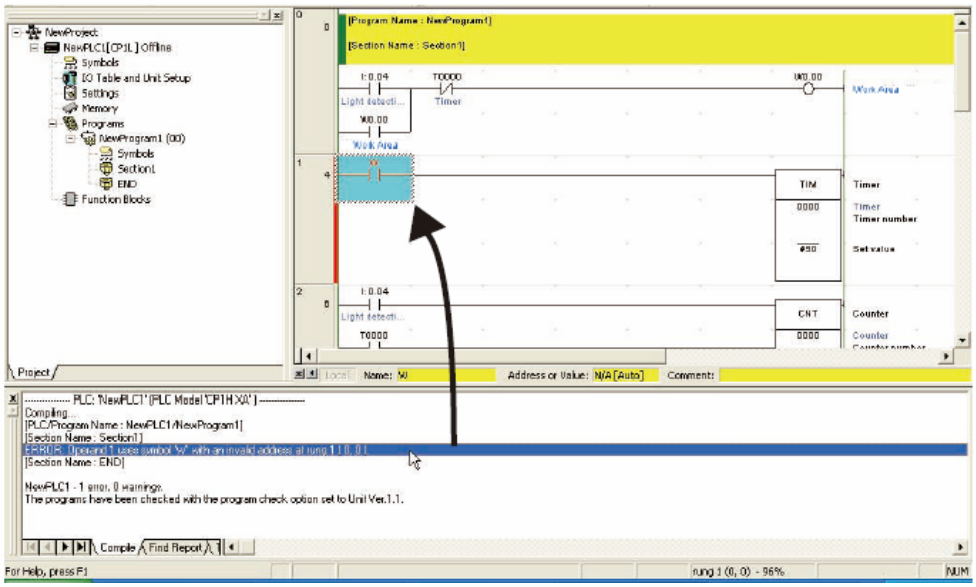


การ Compile จะเริ่มขึ้นเมื่อ Compile เสร็จผลที่ได้จากการตรวจสอบจะแสดงที่ Output window



2. ถ้ามีข้อผิดพลาด(error) เกิดขึ้น ให้ double-click ที่ข้อความ error นั้นใน Output window

Cursor จะกระโดดไปยังตำแหน่งที่ error นั้นถูกพบ จากนั้นจึงแก้ไข error

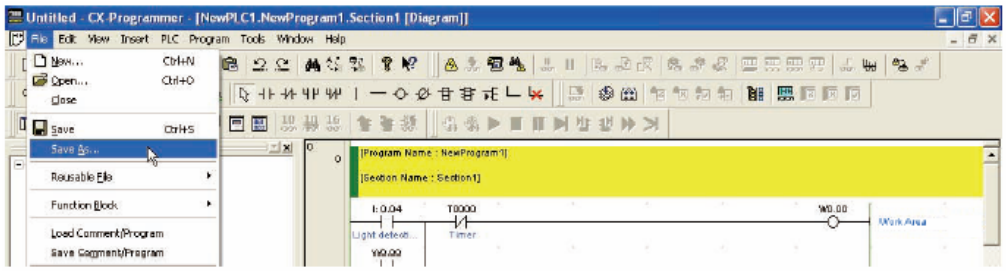


6.4.2 การ Save โปรแกรม

โปรแกรมจะถูกจัดเก็บเป็นกลุ่มตามแต่ละโปรเจคที่สร้างขึ้น

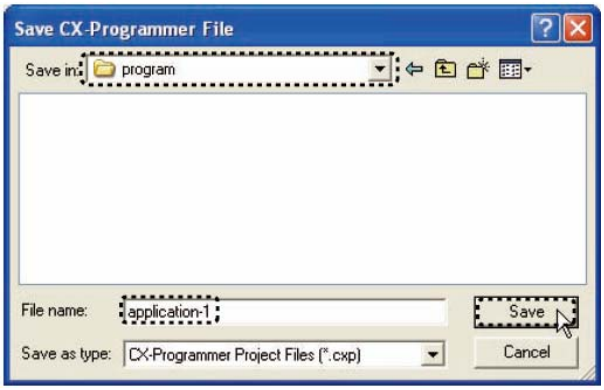
1. เลือก [File] - [Save As] จาก Main menu

   ไดอะลอกบล๊อก 'Save CX-Programmer File' จะปรากฏขึ้น



2. ให้ระบุสถานที่และชื่อไฟล์ที่จะจัดเก็บ แล้วคลิก [Save]

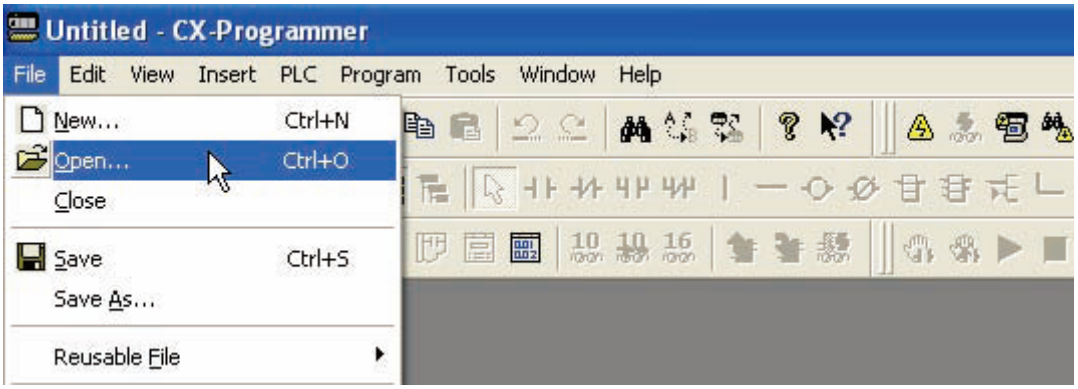
   ไฟล์จะถูกจัดเก็บ



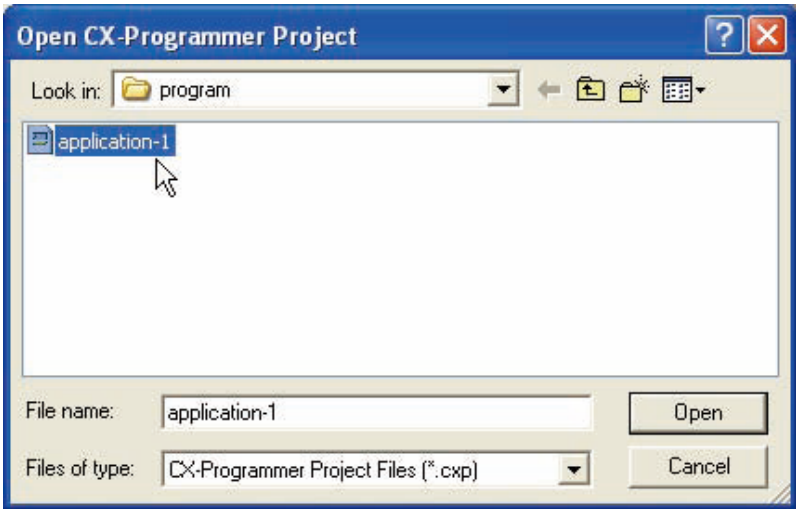
6.4.3 การ Load โปรแกรม

เราสามารถเรียกโปรแกรมที่จัดเก็บแล้วมาใช้งานได้ โดยโปรแกรมจะถูกโหลดเป็นกลุ่มของแต่ละโปรเจค

1. เลือก [File] - [Open] จาก Main menu
   ไดอะลอกบล๊อก 'Open CX-Programmer Project' จะปรากฏขึ้น



2. ให้ระบุสถานที่และชื่อไฟล์ที่จะเรียก แล้วคลิก [Open]
   โปรแกรมที่ถูกเรียกจะปรากฏออกมา



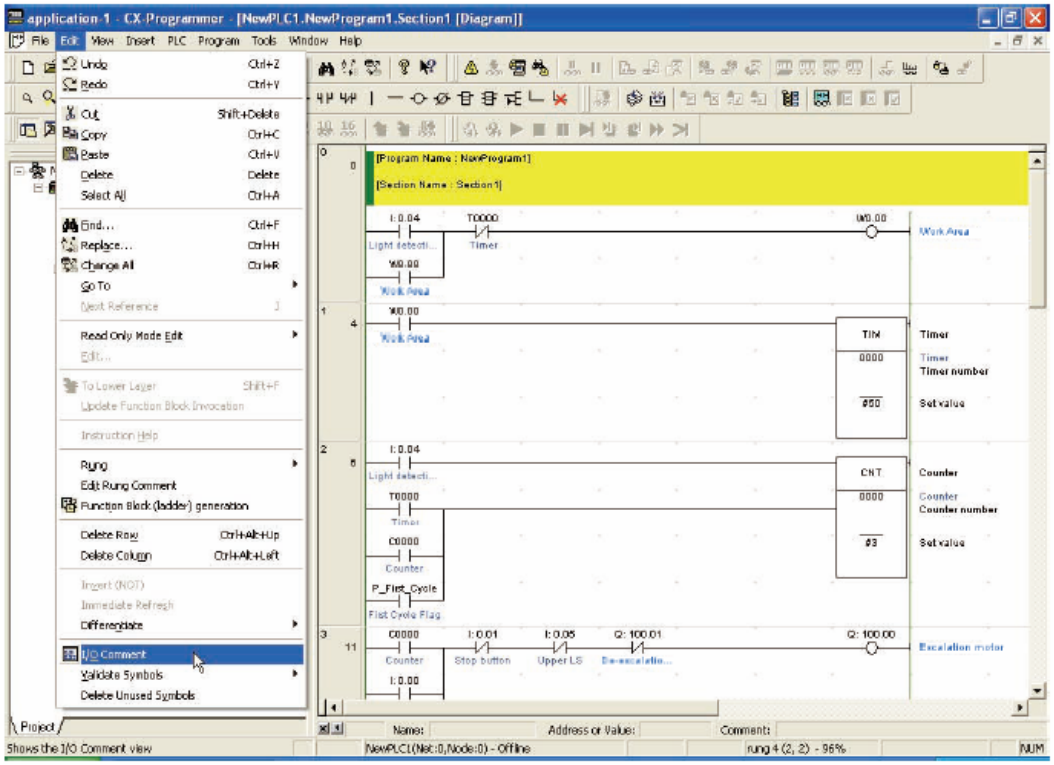
6.5 การแก้ไขโปรแกรม

โปรแกรมที่เขียนขึ้นสามารถแก้ไขใน CX-programmer ได้ รวมทั้งสามารถเพิ่มหรือแก้ไข I/O comment และ rung comment ได้เช่นเดียวกัน

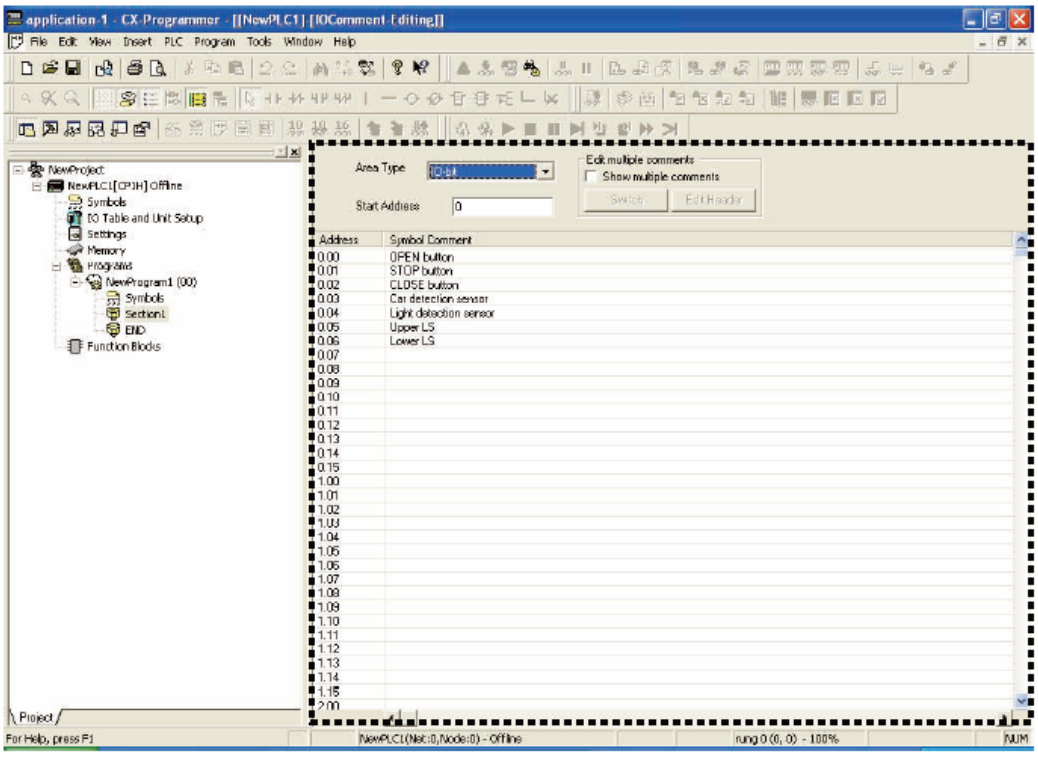
6.5.1 การแก้ไข I/O comment

เราสามรถแก้ไขและเพิ่ม I/O comment โดยใช้รายการแอดเดรส

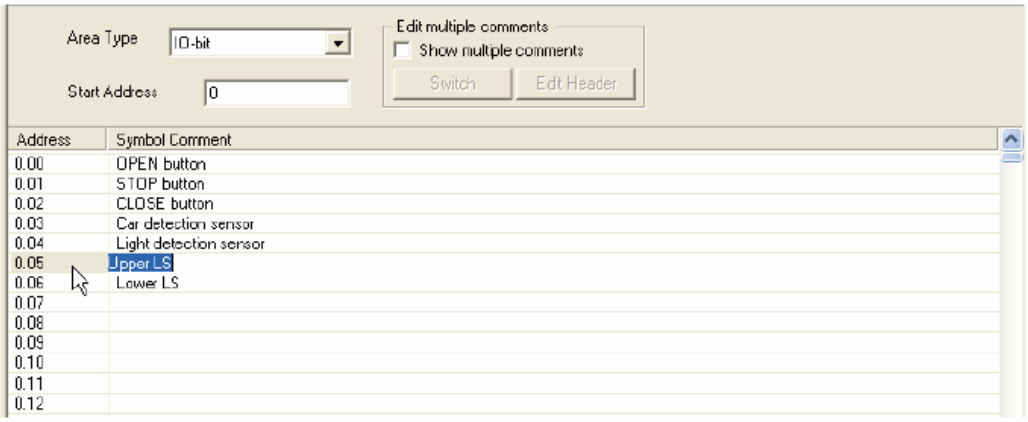
1. เลือก [Edit] - [I/O Comment] จาก Main menu



หน้าต่าง I/O comment จะปรากฏออกมาดังนี้

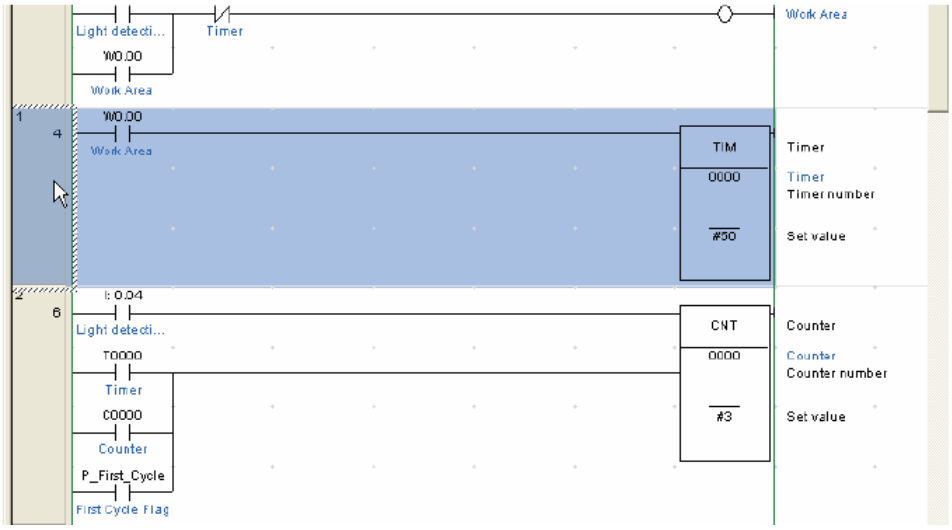


2. ให้ Double-click แอดเดรสที่คุณต้องการป้อนหรือแก้ไข I/O comment พื้นที่สำหรับป้อน I/O comment จะใช้งานได้ จากนั้นให้ป้อนหรือแก้ไข I/O comment

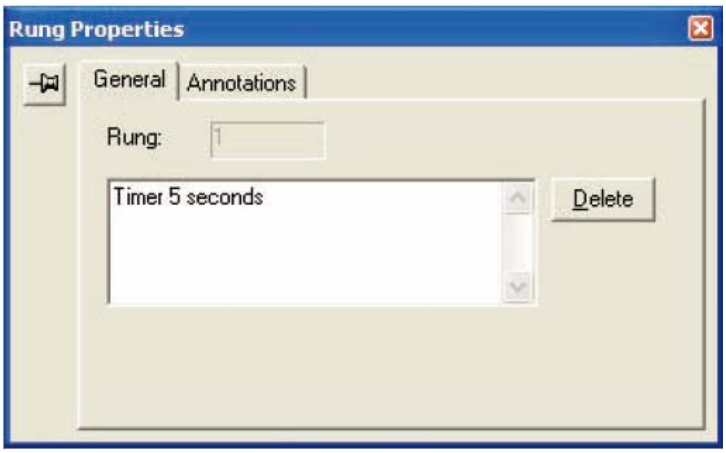


6.5.2 การป้อน Rung comment

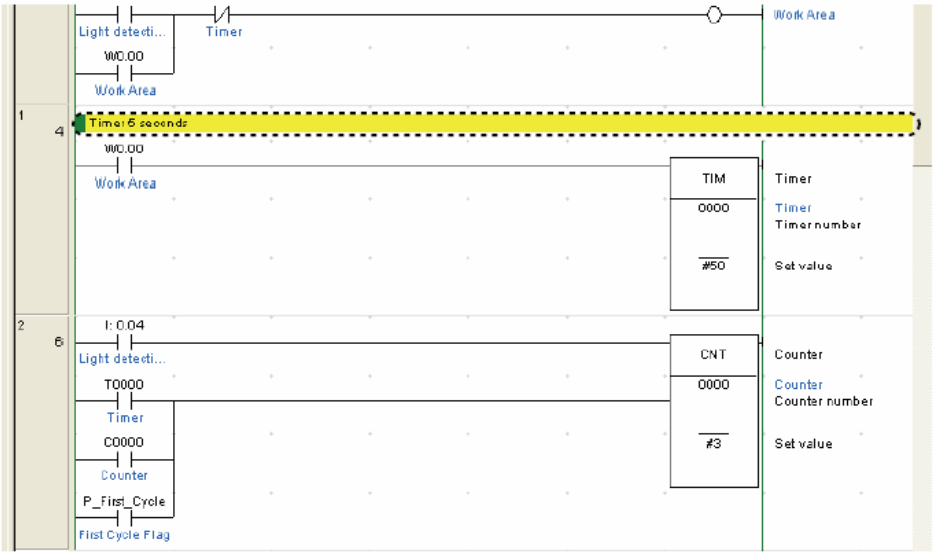
1. Double-click ที่ Rung header สำหรับ Rung ที่คุณต้องการใส่ Comment ไดอะลอกบล๊อก ‘Rung Properties’ จะปรากฏขึ้น



2. คลิกที่ General ป้อน Comment ตามที่ต้องการ



3. ปิดไดอะลอกบล๊อก ‘Rung Properties’
Comment ที่ป้อนจะปรากฏบนโปรแกรมแลดเดอร์



6.5.3 การแก้ไข Rung

➢ **การลบ**

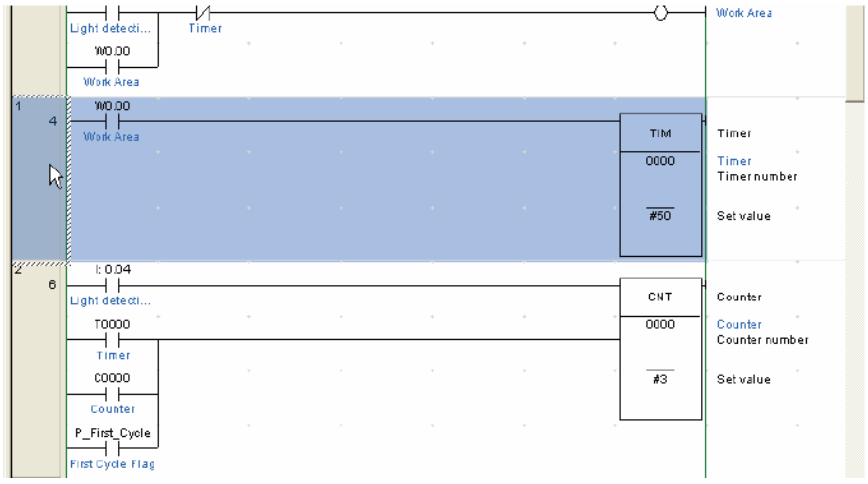
- ***หน้าคอนแทค/คำสั่ง***

1. วาง Cursor บนคอนแทคหรือคำสั่งที่ต้องการจะลบ จากนั้นกดคีย์ [Delete]

- ***Rung***

1. คลิกที่ Rung header

   โปรแกรมใน Rung จะถูกลบทั้งหมด



2. จากนั้นกดคีย์ [Delete]

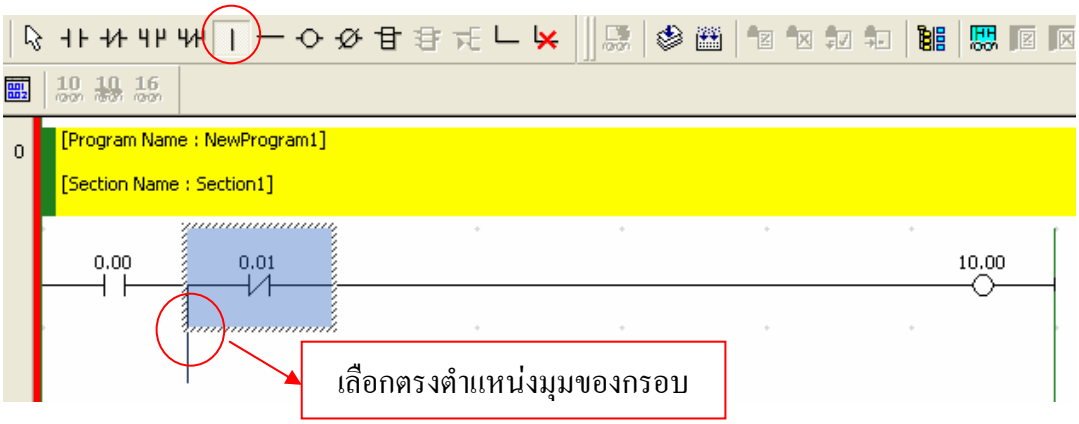
   Rung ที่เลือกไว้จะถูกลบทั้งหมด

6.5.4 การลากเส้นแนวนอนและแนวตั้งเพื่อเชื่อมสัญลักษณ์แต่ละตัว

ในกรณีที่ต้องการเชื่อมต่อสัญลักษณ์โดยการลากเส้นในแนวนอน ให้คลิกที่สัญลักษณ์รูปเส้นในแนวนอน (Horizontal Line) จากนั้นคลิกที่ตำแหน่งช่อง Cell ที่ต้องการวางรูปเส้น ก็จะเป็นการวางรูปเส้นในแนวนอนลงไปบนช่อง Cell นั้น

แต่ถ้าต้องการลากเส้นในแนวตั้งเพื่อต้องการจะต่อรูปสัญลักษณ์แบบขนาน จะมีจุดสังเกต โดยดูตามรูป

1. คลิกสัญลักษณ์ของเส้นให้ยุบลงไป



2. นำมาคลิกที่ตำแหน่งที่ต้องการโดยเลือก คลิกที่ตำแหน่งมุมของกรอบที่เหลี่ยมที่ต้องการลากเส้นแนวตั้ง
3. ในกรณีที่ต้องการลบเส้นให้ใช้เมาส์คลิกอีกครั้งหนึ่งที่ตำแหน่งเดิม
4. หลังจากนั้นจึงนำสัญลักษณ์ที่เป็น Contact มาวางขนานดังรูป

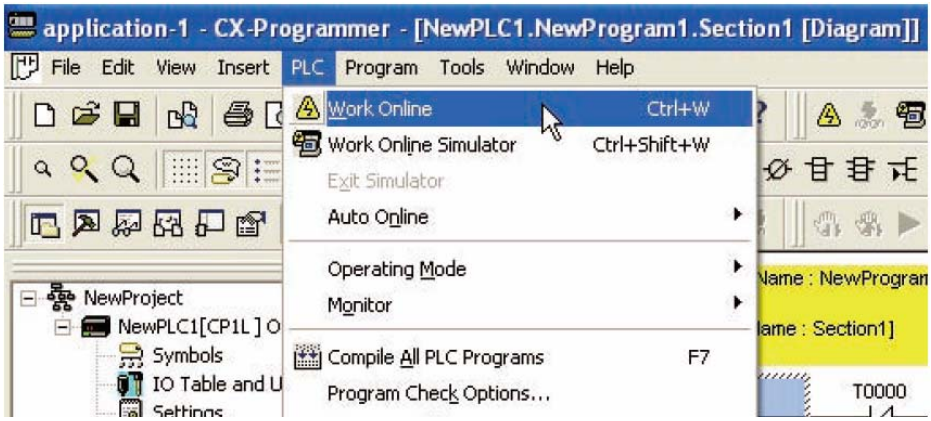
6.6 การ Online

6.6.1 การ Online เพื่อ Transfer โปรแกรม

เมื่อต้องการ Transfer โปรแกรมหรือทำการ Setting การทำงานต่างๆ ของ PLC เราต้องทำการ Online ก่อน

1. ใช้ CX-programmer เปิดโปรแกรมที่ต้องการ Transfer
2. เลือก [PLC] - [Work Online] จาก Main menu

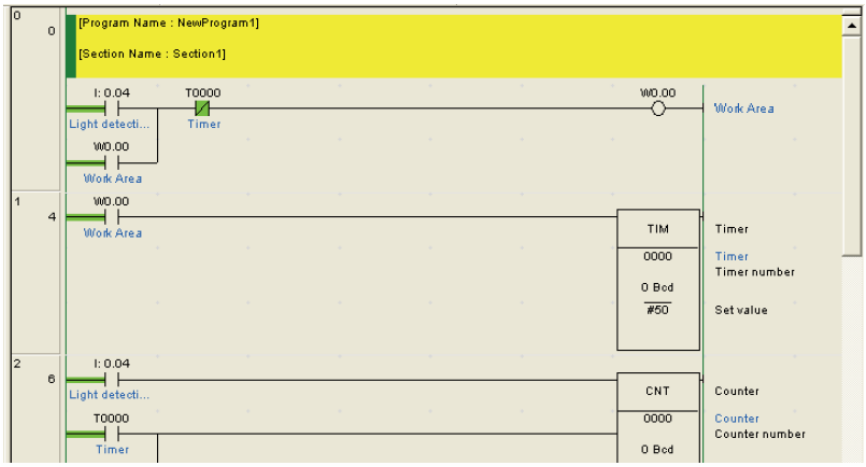
   ไดอะลอกบล๊อกจะปรากฏขึ้นเพื่อให้ยอมรับการ Online



3. คลิก [Yes]

   ไดอะลอกบล๊อกจะปิด

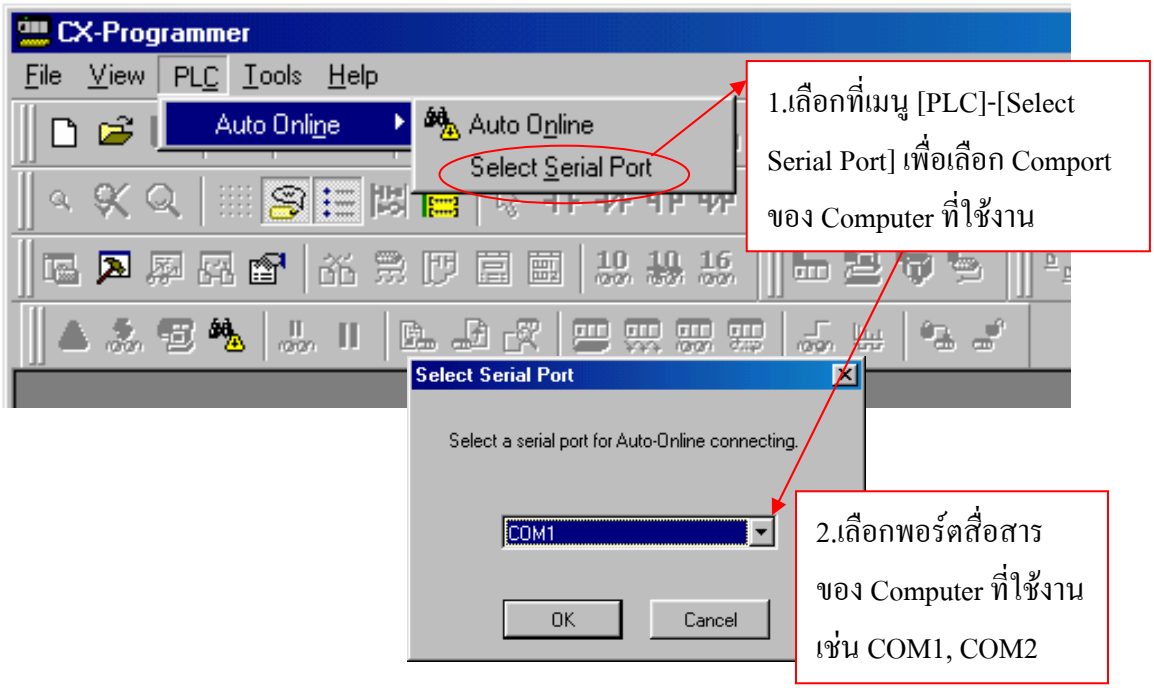
เมื่อระบบ Online ได้สำเร็จ ส่วนที่เป็นแลดเดอร์จะเปลี่ยนเป็นสีเทาอ่อน

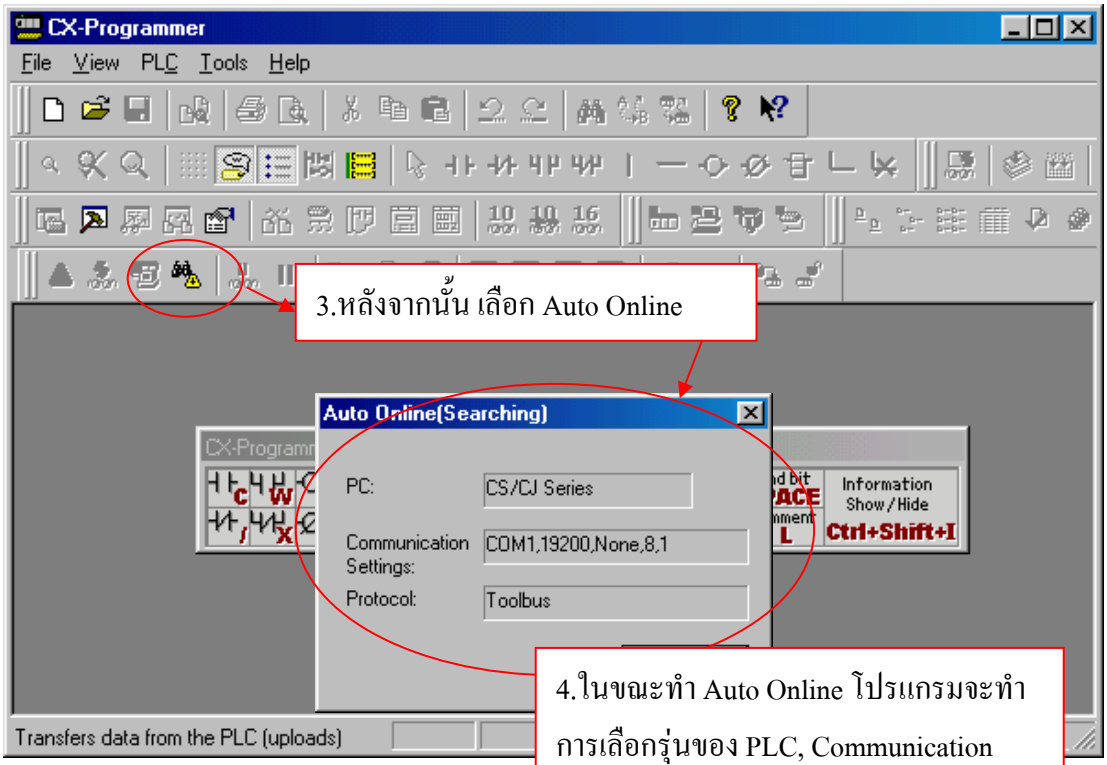


หมายเหตุ ถ้าไม่สามารถ Online ได้ กรุณาตรวจสอบการตั้งค่า PLC type และ Communication setting

6.6.2 การทำ Auto-online

เราสามารถทำ Auto-online ในกรณีที่ไม่ทราบรุ่นและการตั้งค่าพอร์ตสื่อสารของ PLC เมื่อเปิดซอฟ์ตแวร์ ตัว CX-Programmer ขึ้นมาในขั้นตอนแรก จะขึ้นหน้าจอดังรูป

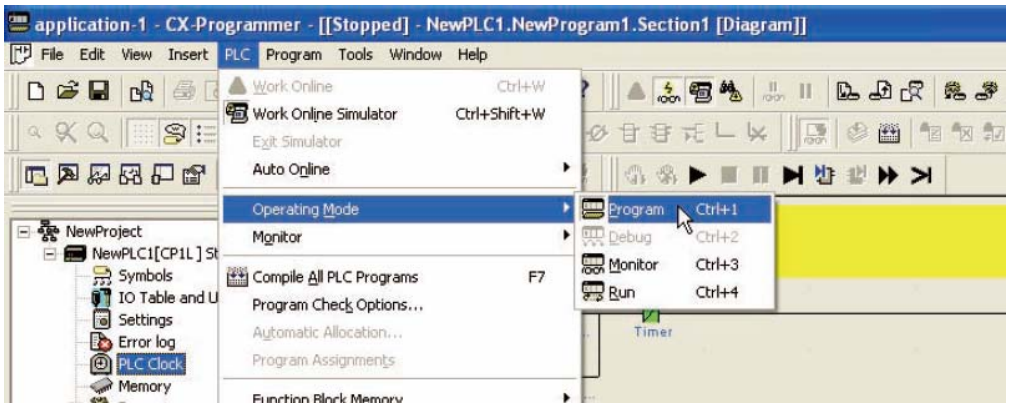



CX-Programmer
File View PLC Tools Help
3.หลังจากนั้น เลือก Auto Online
Auto Online(Searching)
PC:
CS/CJ Series
Communication Settings:
COM1,19200,None,8,1
Protocol:
Toolbus
Information Show/Hide
Ctrl+Shift+I
Transfers data from the PLC (uploads)
4.ในขณะทำ Auto Online โปรแกรมจะทำการเลือกรุ่นของ PLC, Communication Setting และ Protocol ที่ใช้งานให้อัตโนมัติ หลังจาก Online แล้วจะ Load โปรแกรมขึ้นมาอยู่บนคอมพิวเตอร์อัตโนมัติ

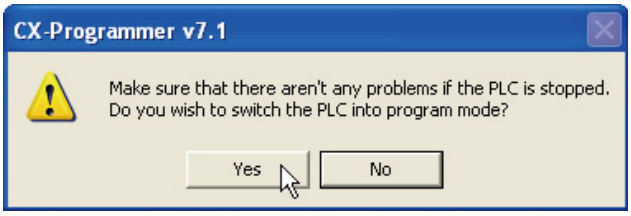
6.7 การเปลี่ยนโหมด PLC

เปลี่ยนโหมด PLC ไปที่ PROGRAM ขั้นตอนต่างๆ แสดงได้ดังต่อไปนี้

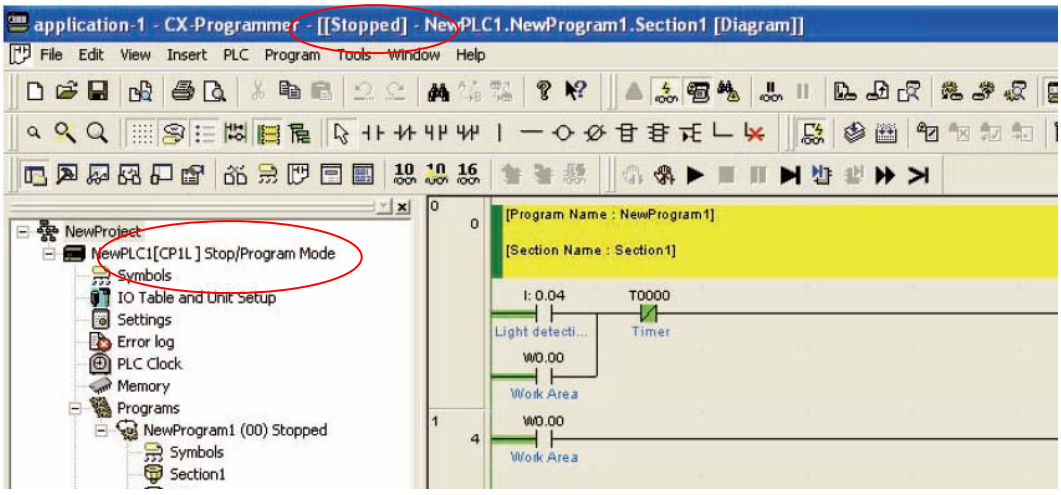
1. เลือก [PLC] - [Operating Mode] - [Program] จาก Main menu
   หรือคลิกที่ Toolbar  จากนั้นไดอะลอกบล๊อกสำหรับการเปลี่ยนโหมดจะปรากฏขึ้น ให้เลือก Program



2. คลิก [Yes] โหมดการทำงานจะเปลี่ยนไปตามที่เลือก



โหมดการทำงานจะถูกแสดงที่ Title bar และ Project tree



- **โหมดการทำงานของ PLC**

PLC จะมีโหมดการทำงานอยู่ 3 โหมด คือ PROGRAM, MONITOR และ RUN การเปลี่ยนโหมดจะมีผลต่อการทำงานของ PLC ซึ่งอธิบายได้ดังต่อไปนี้

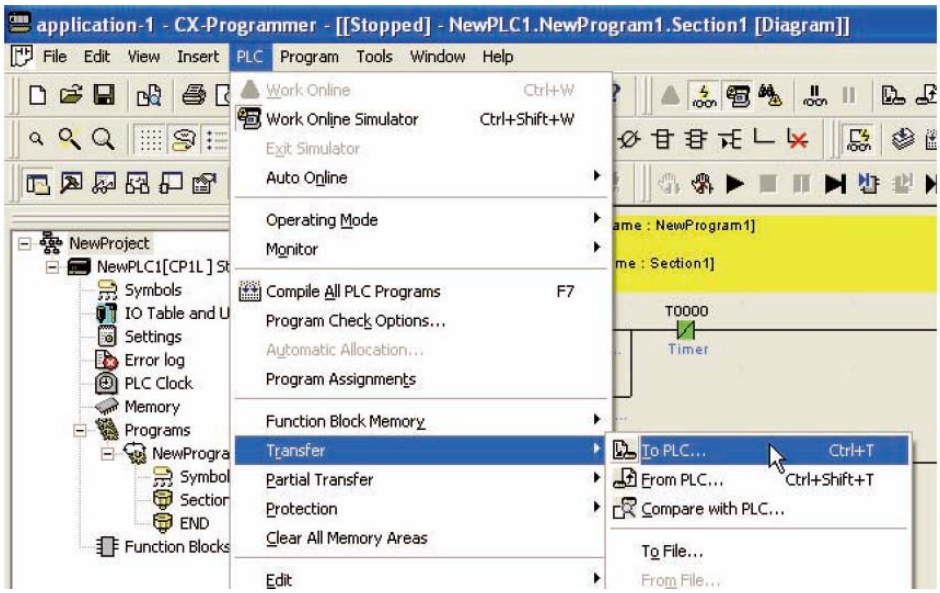
- **โหมด PROGRAM** ในโหมดนี้โปรแกรมจะหยุดการทำงาน ซึ่งเหมาะสำหรับการเตรียมการเพื่อใช้ในการตั้งค่าต่างๆ เช่น PLC setup, Transfer โปรแกรม และการทำ Force-set/Force-reset
- **โหมด MONITOR** ในโหมดนี้โปรแกรมจะทำงาน แต่สามารถทำ Online edit, Force-set/Force-reset และเปลี่ยนค่าในหน่วยความจำได้ โหมดนี้เหมาะสำหรับการปรับแต่งระหว่างการทดสอบโปรแกรม
- **โหมด RUN** ในโหมดนี้โปรแกรมจะทำงาน ใช้โหมดนี้เพื่อการควบคุมแบบปกติ

ตารางข้างล่างนี้แสดงความสัมพันธ์ระหว่างสถานะการทำงานในแต่ละโหมด

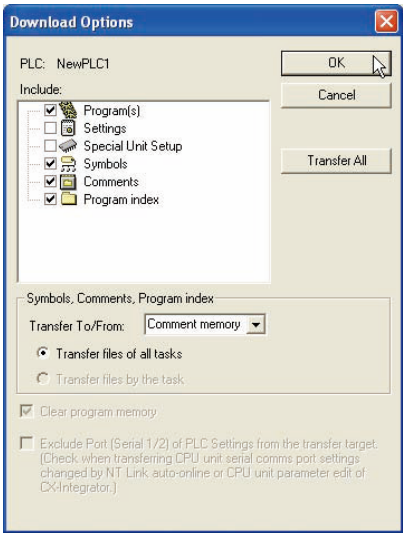
Operating Mode			**PROGRAM**	**RUN**	**MONITOR**
Program status			Stopped	Running	Running
I/O refreshing			Execute	Execute	Execute
External I/O status			OFF	ขึ้นอยู่กับโปรแกรม	ขึ้นอยู่กับโปรแกรม
I/O memory	Non-holding memory		Cleared	ขึ้นอยู่กับโปรแกรม	ขึ้นอยู่กับโปรแกรม
Operations from CX-Programmer	I/O memory monitoring		OK	OK	OK
	Program monitoring		OK	OK	OK
	Program transfer	From PLC	OK	OK	OK
		To PLC	OK	X	X
	Compiling		OK	X	X
	PLC setup		OK	X	X
	Changing program		OK	X	OK
	Force-setting/Force-resetting		OK	X	OK
	Changing timer/counter SV		OK	X	OK
	Changing timer/counter PV		OK	X	OK
	Changing I/O memory PV		OK	X	OK

6.8 การ Transfer โปรแกรม

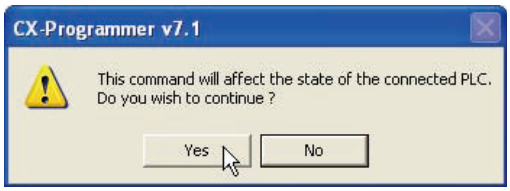
1. เลือก [PLC] - [Transfer] - [To PLC] จาก Main menu ไดอะลอกบล๊อก ‘Download Options’ จะปรากฏขึ้น



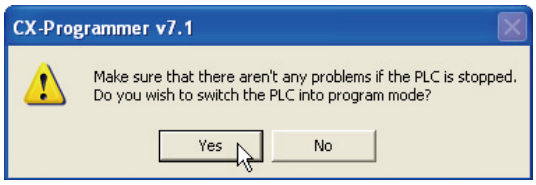
2. เลือกสิ่งที่ต้องการดาว์นโหลด โดยคลิกในเช็คบล็อกจากนั้นคลิก [OK]



3. คลิก [Yes] ถ้าไดอะลอกบล๊อกนี้ปรากฏขึ้น



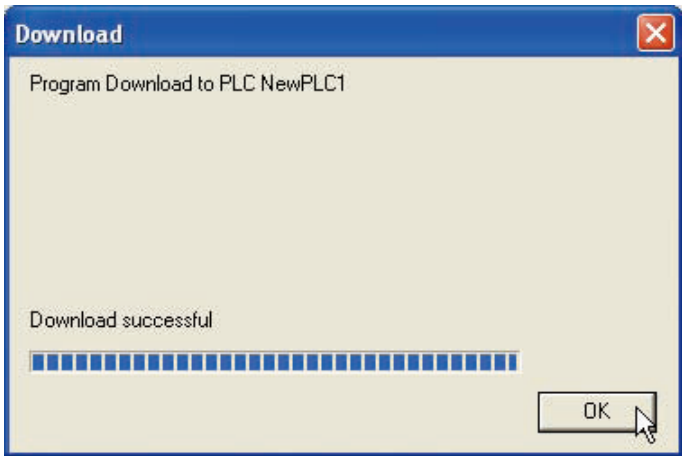
ถ้าไดอะลอกบล็อกนี้ปรากฏขึ้น ให้คลิก [Yes] อีกครั้งหนึ่ง



การ Transfer จะเริ่มขึ้น

4. คลิก [OK]

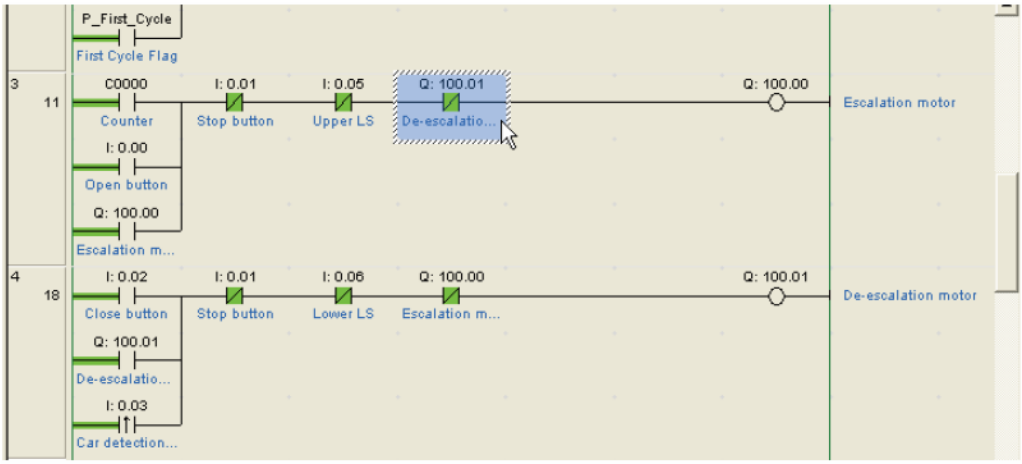
การ Transfer โปรแกรมเสร็จสิ้น



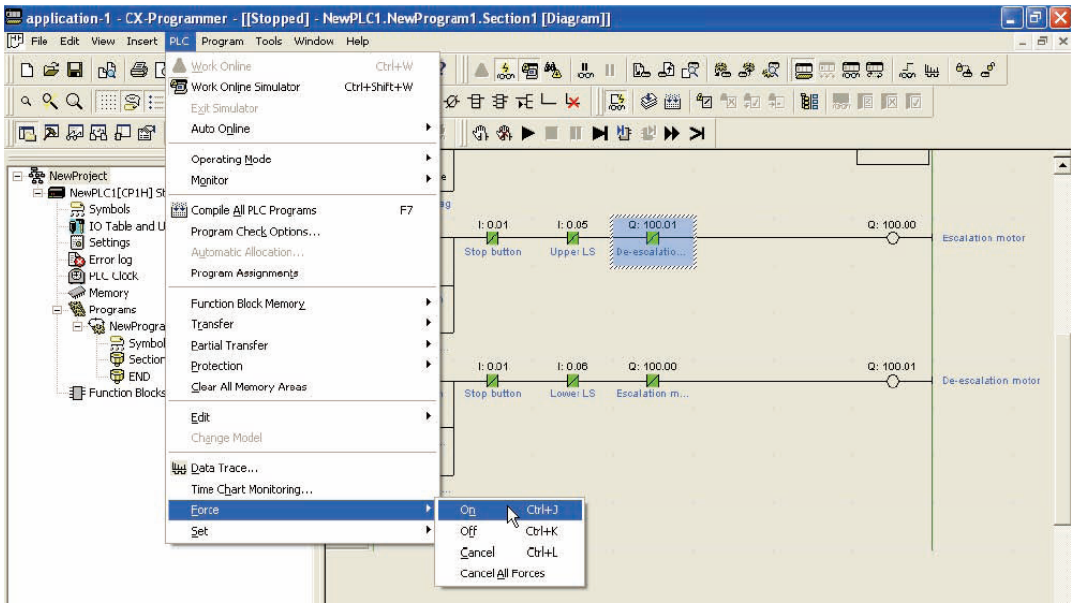
6.9 การทำ Force-set/Force Reset

CX-programmer สามารถควบคุมอุปกรณ์ I/O ต่างๆ ได้อย่างอิสระโดยใช้ฟังก์ชันนี้ ซึ่งสามารถสั่งให้บิตต่างๆ เหล่านั้น On หรือ Off ได้ตามต้องการ

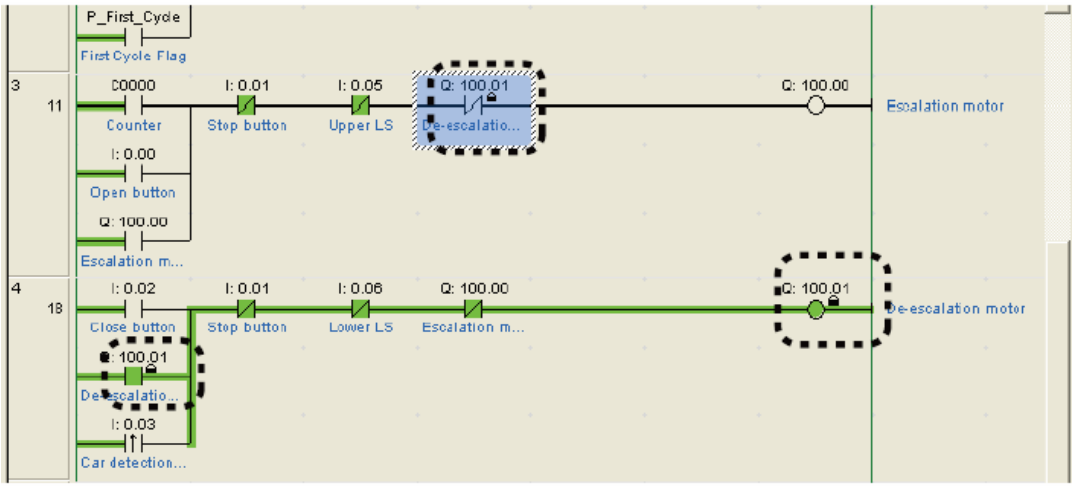
1. เปลี่ยนโหมด PLC ไปที่ MONITOR หรือ PROGRAM
2. วาง Cursor ที่ตำแหน่งที่ต้องการทำ Force-set



3. เลือก [PLC] - [Force] - [On] จาก Main menu



เมื่อทำการ Force-set หน้าคอนแทคนั้นจะมีสัญลักษ์ Force-set ปรากฏอยู่



หมายเหตุ

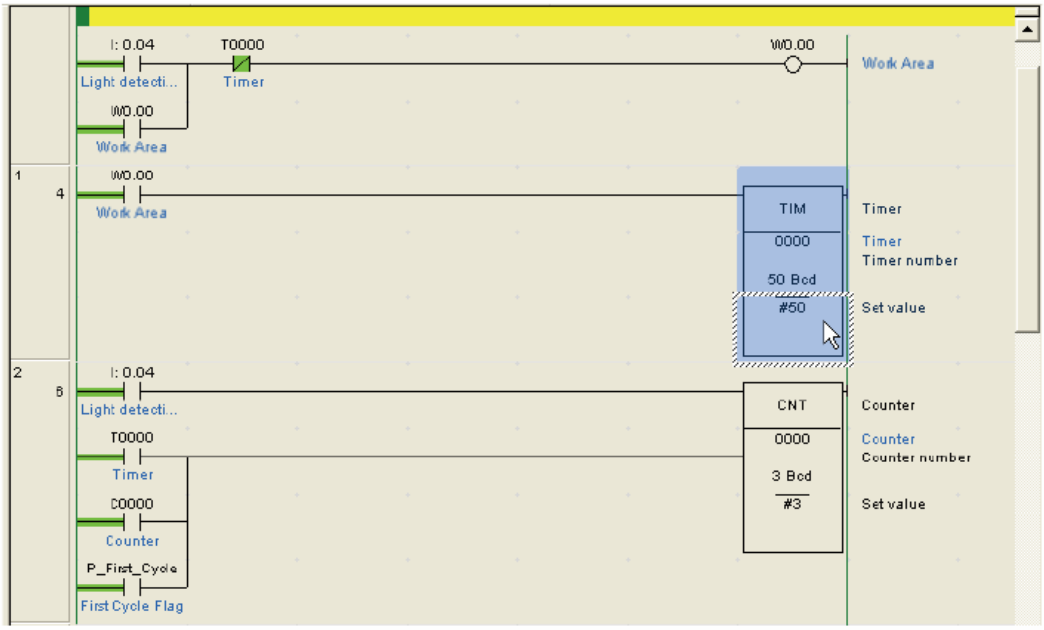
- เลือก [On] เพื่อบังคับให้หน้าคอนแทค ON และเลือก [Off] เพื่อบังคับให้หน้าคอนแทค OFF
- ถ้าต้องการยกเลิก force-set/force-reset ให้เลือก [Cancel]
- พื้นที่หน่วยความจำที่สามารถ force-set/force-reset ได้ CIO area, work area (WR), timer completion flag, holding area (HR), counter completion flag

6.10 การเปลี่ยนค่า Timer

ค่าตั้งเวลาของ Timer หรือหน่วยความจำต่างๆ สามารถเปลี่ยนแปลงได้ระหว่างทำงาน เพื่อให้ได้ค่าที่เหมาะสม

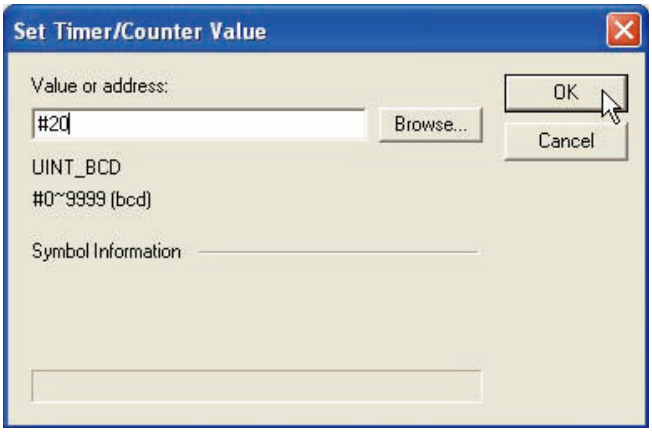
1. เปลี่ยนโหมด PLC ไปที่ MONITOR หรือ PROGRAM
2. Double-click ค่า Setting ของ timer ที่ต้องการจะเปลี่ยน

   ไดอะลอกบล๊อก 'Set Timer/Counter Value' จะปรากฏขึ้น



3. ป้อนค่าใหม่ จากนั้นคลิก [OK]

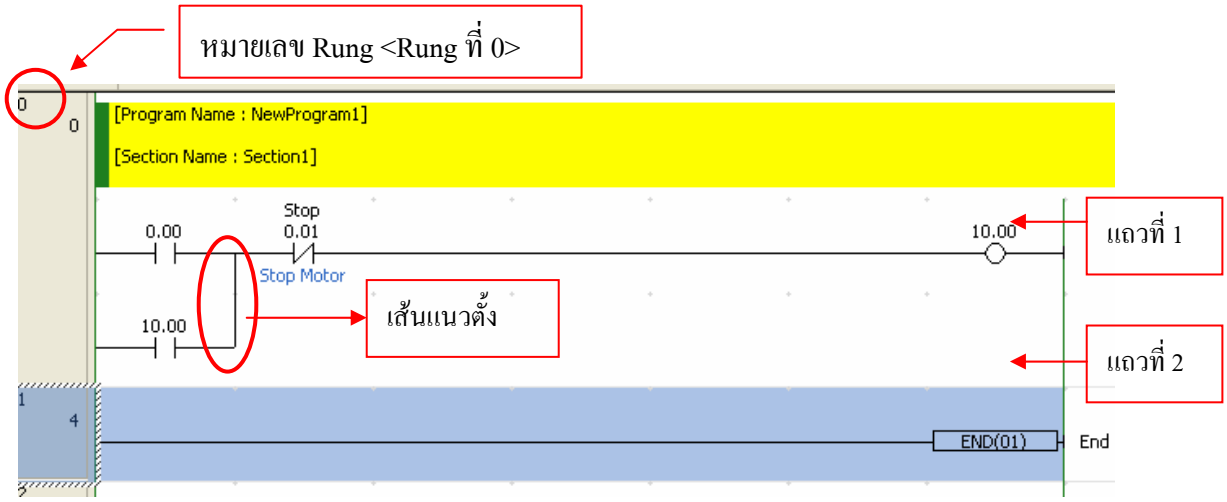
   ค่า Setting ของ Timer จะเปลี่ยนเป็นค่าใหม่



6.11 เทคนิคการใช้งานอื่นๆ

6.11.1 การแทรก/ลบ Rung

แต่ละช่องของ Ladder Diagram เรียกว่า 1 รัง (Rung) หรือ 1 เน็ตเวิร์ก (Network)

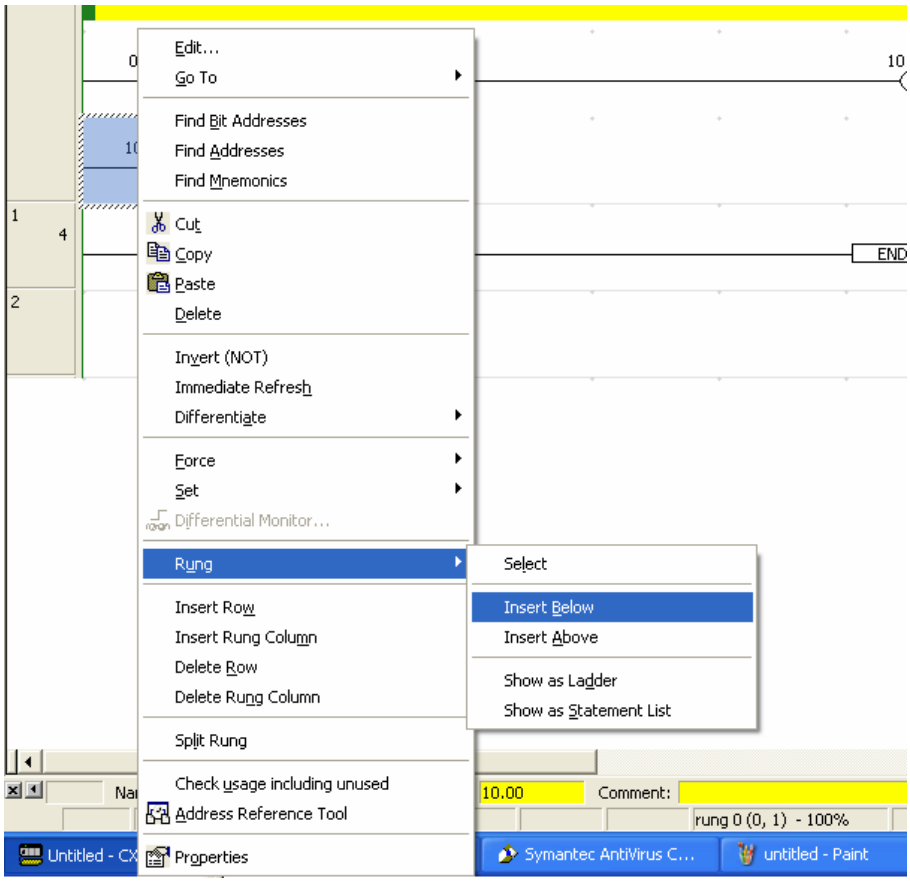


6.11.1.1) *เงื่อนไขในการแยก Rung* ให้สังเกตเส้นแนวตั้งระหว่าง Ladder Diagram แต่ละบรรทัดว่ามีเส้นแนวตั้ง ลากเชื่อมต่อระหว่างแถวหรือไม่

- ถ้ามีเส้นแนวตั้งต้องเขียน Ladder Diagram ใน Rung เดียวกัน เช่น Ladder Diagram ใน Rung ที่ 0 มีเส้นแนวตั้งลากเชื่อมต่อระหว่างแถวที่ 1 และ 2 ดังนั้นจึงเป็น Rung เดียวกัน
- ถ้าไม่มีเส้นแนวตั้ง ให้แยก Rung เช่น Rung ที่ 0 กับ Rung ที่ 1(ที่มีคำสั่ง END(01)) แยกกันคนละ Rung

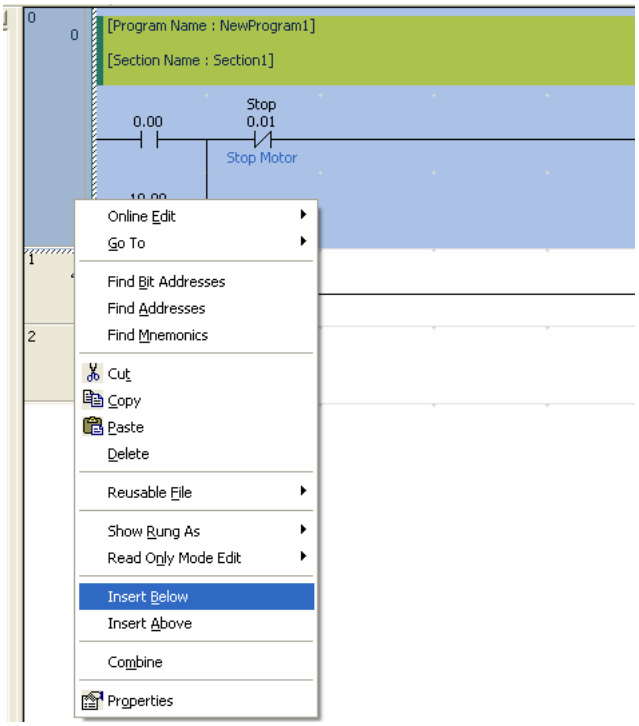
6.11.1.2) *การแทรก Rung* สามารถทำได้หลายวิธีดังนี้

วิธีที่ 1 เลือกที่ Rung ที่ต้องการ แล้ว คลิกขวา จะปรากฏหน้าจอดังรูป



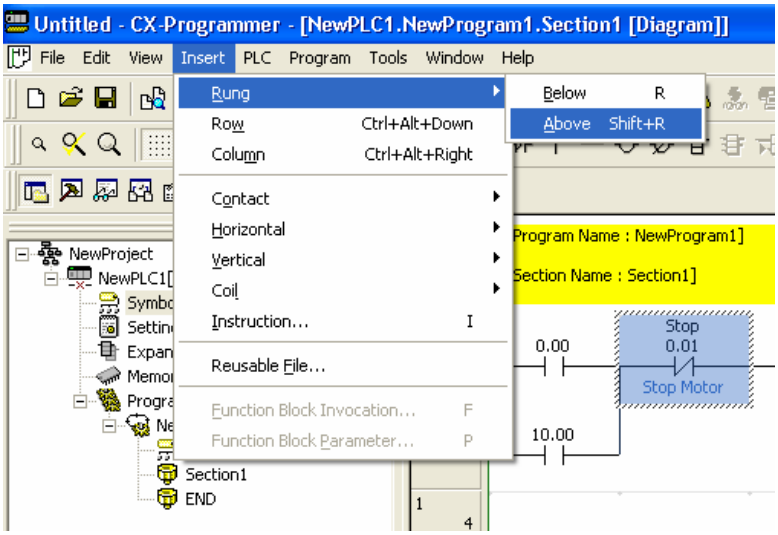
- เลือก<Rung>-<Insert Below> หมายถึง เพิ่ม Rung เข้าไปทางด้านล่างของ Rung ที่เลือก
- เลือก<Rung>-<Insert Above> หมายถึง เพิ่ม Rung เข้าไปทางด้านบนของ Rung ที่เลือก

วิธีที่ 2 เลือกทางด้านหน้าของ Rung (จะปรากฏแถบสีฟ้าครอบคลุมพื้นที่ของ Rung นั้นทั้งหมด) แล้วคลิกขวา จะปรากฏหน้าจอดังรูป

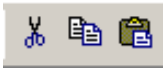


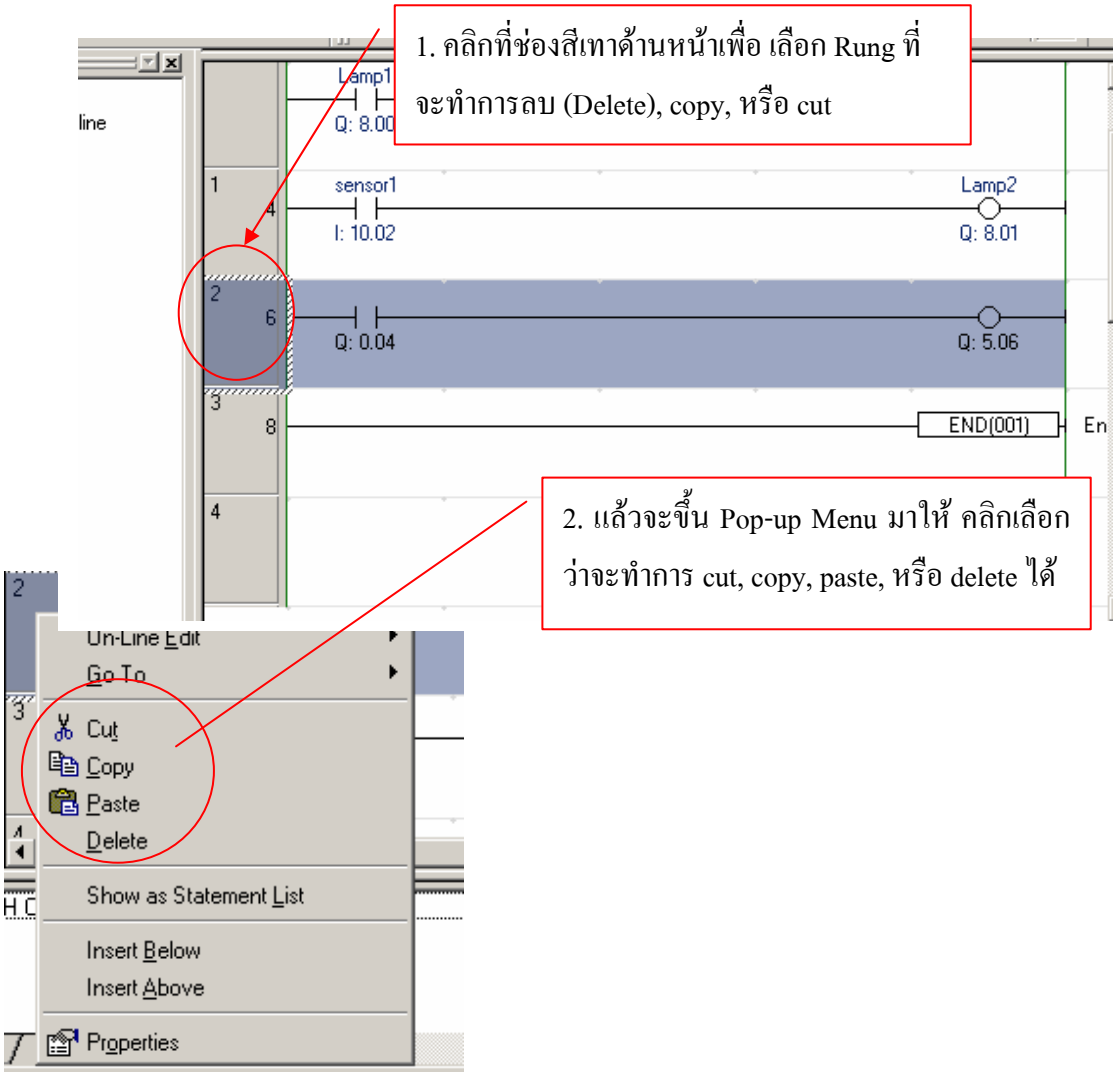
- เลือกที่ <Insert Below> เพื่อเพิ่ม Rung เข้าไปทางด้านล่างของ Rung ที่เลือก
- เลือกที่ <Insert Above> เพื่อเพิ่ม Rung เข้าไปทางด้านบนของ Rung ที่เลือก

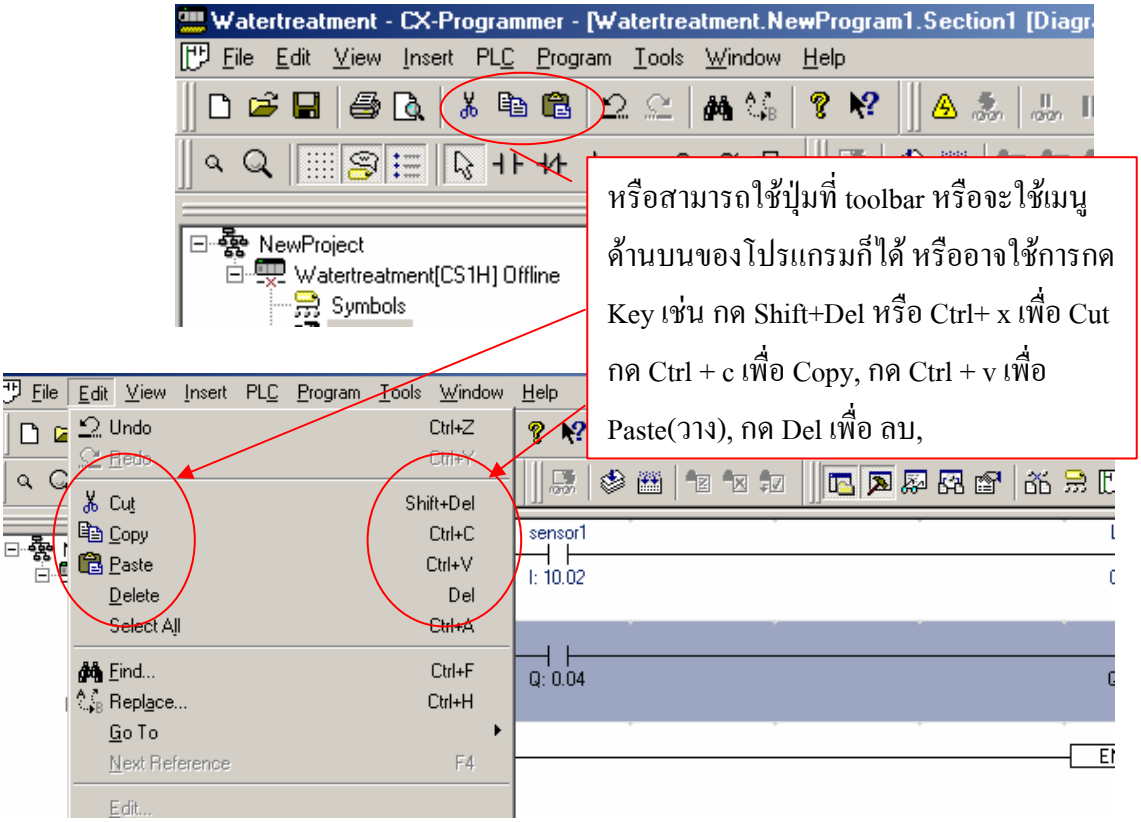
<u>วิธีที่ 3</u> เลือกที่ Rung แล้วเข้าที่เมนู<Insert>-<Rung>-<Below> (กดปุ่ม R บน Keyboard) หรือเลือก<Above> (กดปุ่ม Shift+R บน Keyboard) ดังรูป



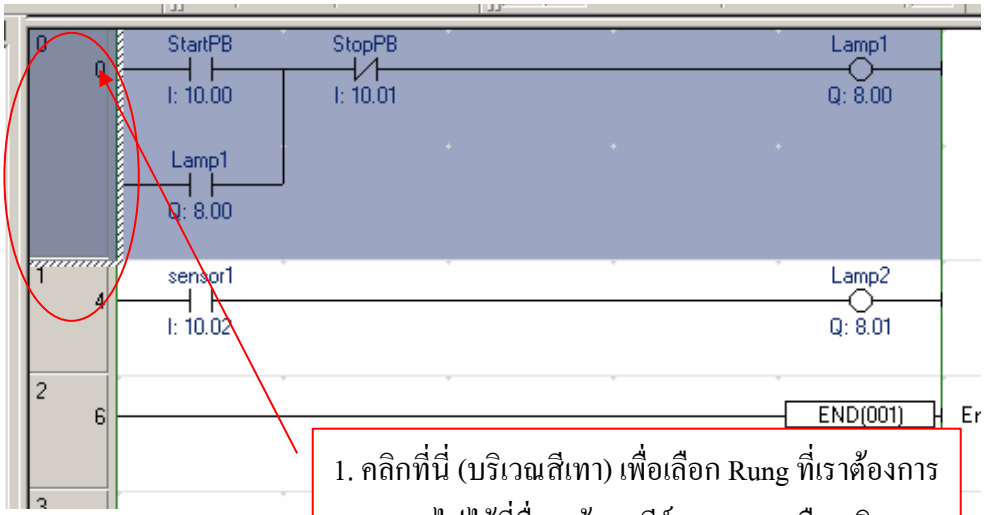
6.11.1.3) *การลบ (Delete) Rung, Copy และ Cut Rung*

สามารถทำการลบ (Delete) Rung นั้นทิ้งหรือการ Copy Rung นั้นไว้แล้วนำไปวาง (Paste) ไว้ที่อื่นเพื่อไปทำการแก้ไขดัดแปลงเล็กน้อย ถ้าโปรแกรมส่วนนั้นมันคล้ายๆ กันแทนที่จะต้องเขียนใหม่ทั้ง Rung หรือทำการ Cut เพื่อย้ายตำแหน่ง Rung นั้นไปไว้ที่ตำแหน่งอื่นของโปรแกรมได้โดยการใช้การ Cut, Copy และ Paste เช่นเดียวกับโปรแกรมอื่นๆ ทั่วไปที่ทำงานบน MS-Windows ดังรูป 

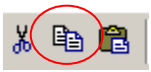


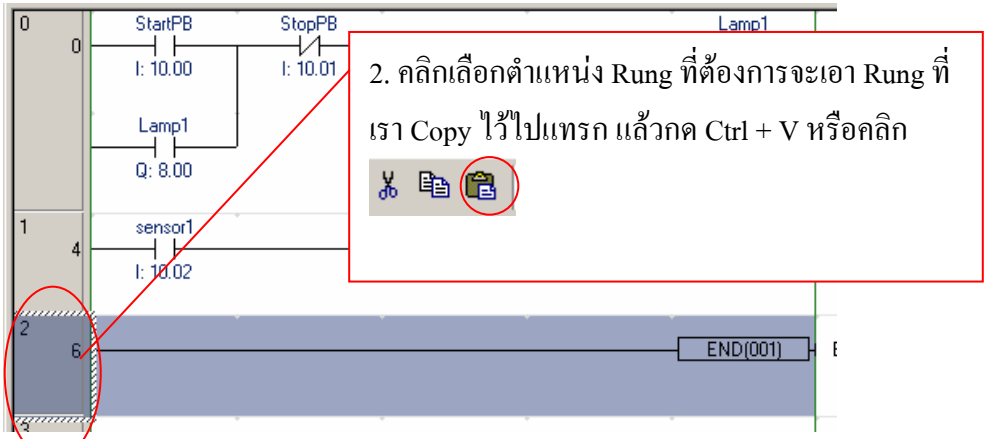


หรือสามารถใช้ปุ่มที่ toolbar หรือจะใช้เมนูด้านบนของโปรแกรมก็ได้ หรืออาจใช้การกด Key เช่น กด Shift+Del หรือ Ctrl+ x เพื่อ Cut กด Ctrl + c เพื่อ Copy, กด Ctrl + v เพื่อ Paste(วาง), กด Del เพื่อ ลบ,

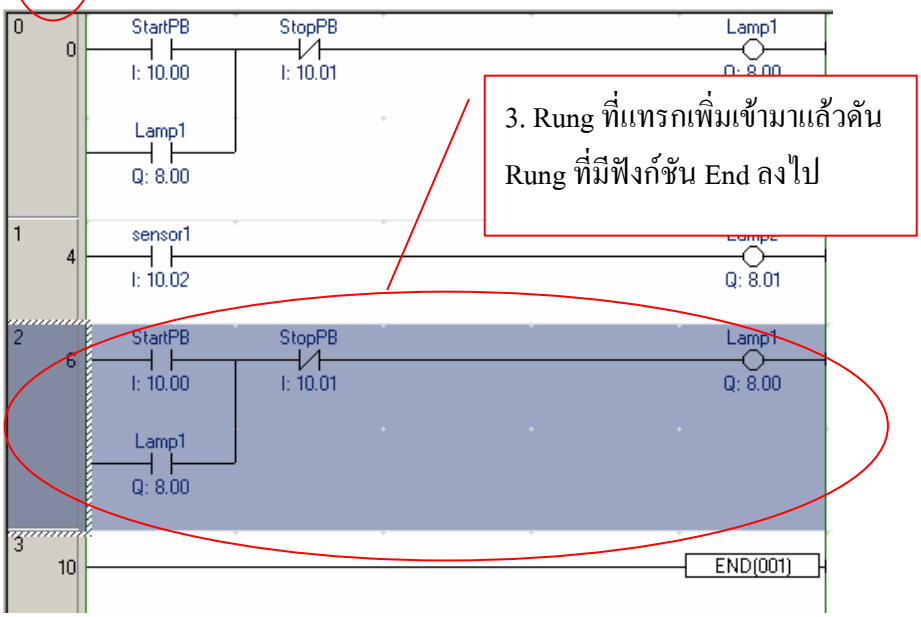


1. คลิกที่นี่ (บริเวณสีเทา) เพื่อเลือก Rung ที่เราต้องการจะ copy ไปไว้ที่อื่น แล้วกดคีย์ Ctrl + C หรือ คลิก

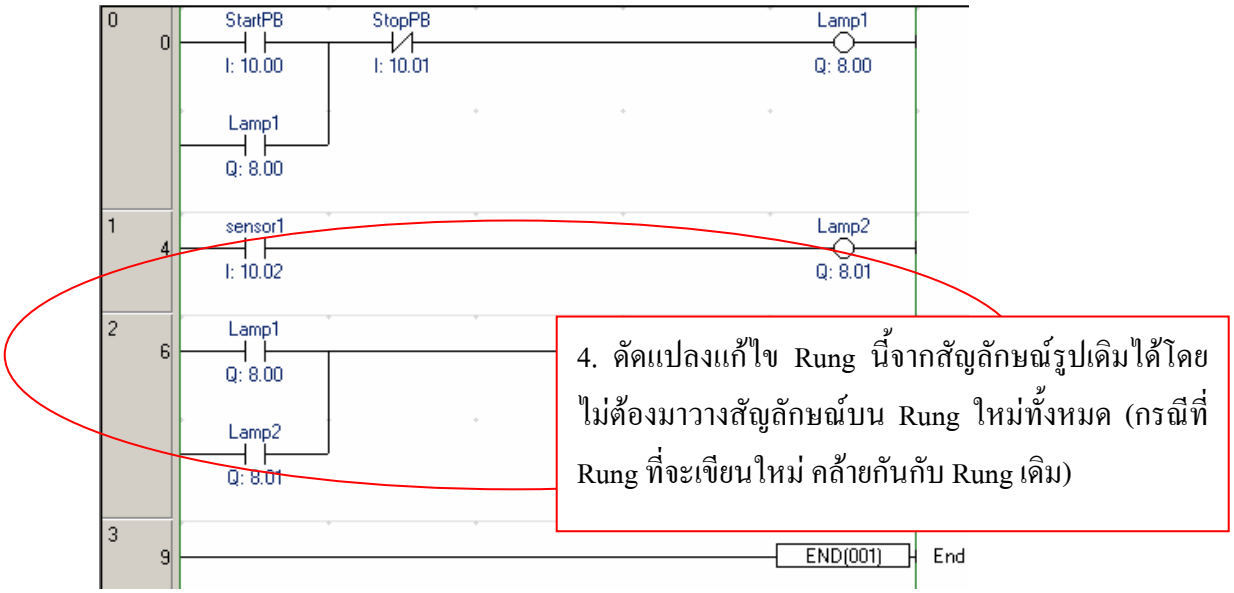




2. คลิกเลือกตำแหน่ง Rung ที่ต้องการจะเอา Rung ที่เรา Copy ไว้ไปแทรก แล้วกด Ctrl + V หรือคลิก



3. Rung ที่แทรกเพิ่มเข้ามาแล้วดัน Rung ที่มีฟังก์ชัน End ลงไป

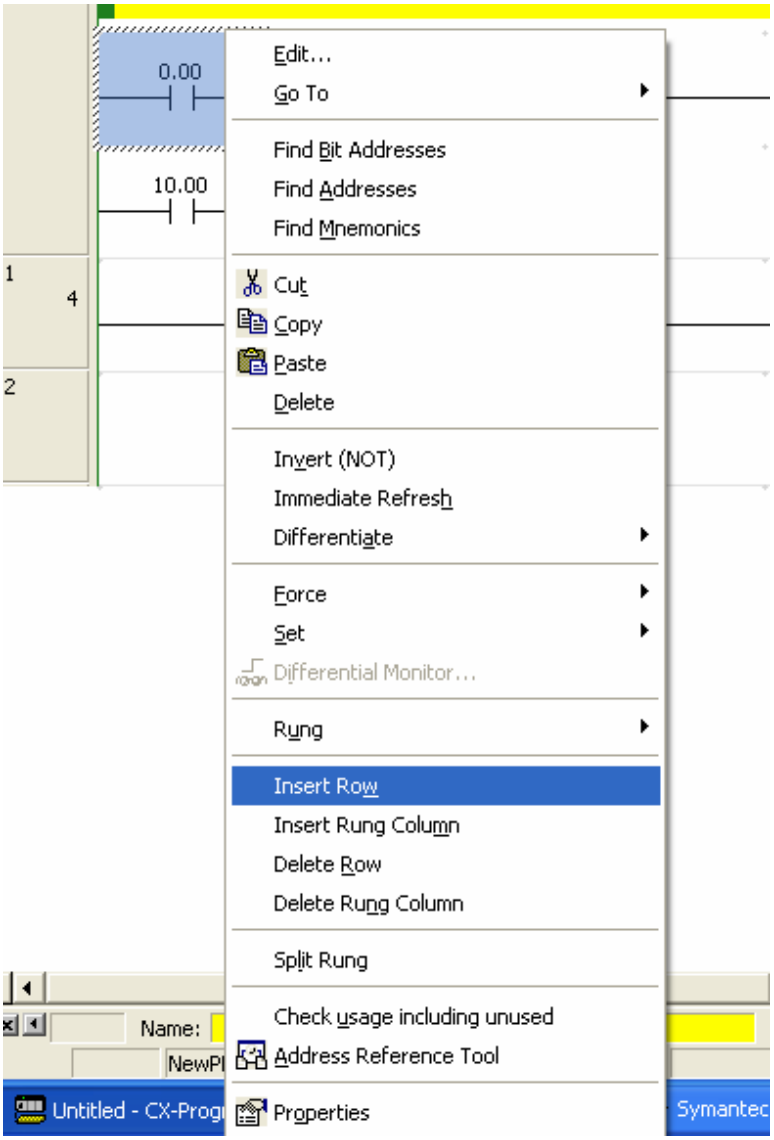


4. ดัดแปลงแก้ไข Rung นี้จากสัญลักษณ์รูปเดิมได้โดยไม่ต้องมาวางสัญลักษณ์บน Rung ใหม่ทั้งหมด (กรณีที่ Rung ที่จะเขียนใหม่ คล้ายกันกับ Rung เดิม)

6.11.2 แทรก/ลบแถวแนวนอนและแนวตั้ง (Row/Column)

สามารถทำได้หลายวิธีเช่นเดียวกับการแทรก Rung ดังนี้

วิธีที่ 1 คลิกขวาภายใน Rung ที่ต้องการแทรก Row/Column ดังรูป



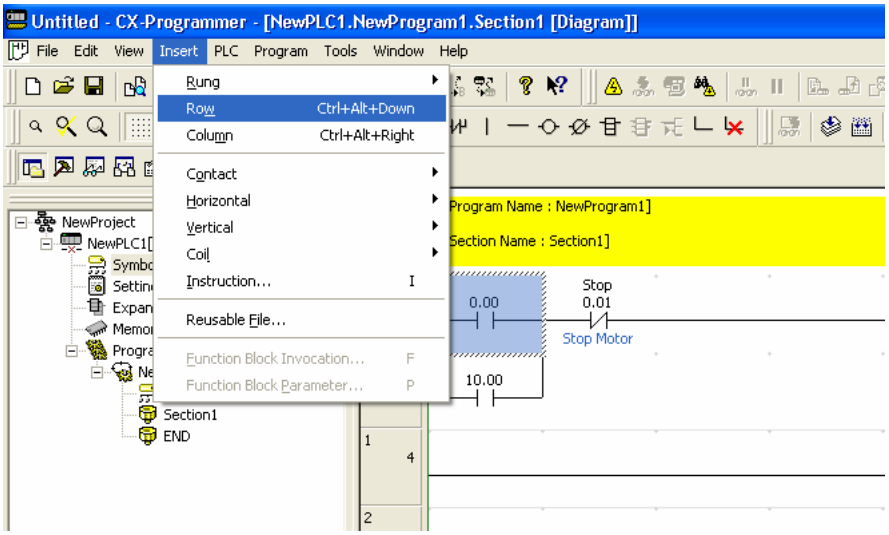
<Insert Row>-เพิ่มแถวทางแนวนอน

<Insert Rung Column>-เพิ่มแถวทางแนวตั้ง

<Delete Row>-ลบแถวทางแนวนอน

<Delete Rung Column>-ลบแถวทางแนวตั้ง

วิธีที่ 2 คลิกเมาส์ภายใน Rung ที่ต้องการเพิ่มแถว หลังจากนั้นเข้าไปที่เมนู<Insert>-<Row>(กดปุ่ม Ctrl+Alt+Down) หรือ <Column> (กดปุ่ม Ctrl+Alt+Right) ดังรูป

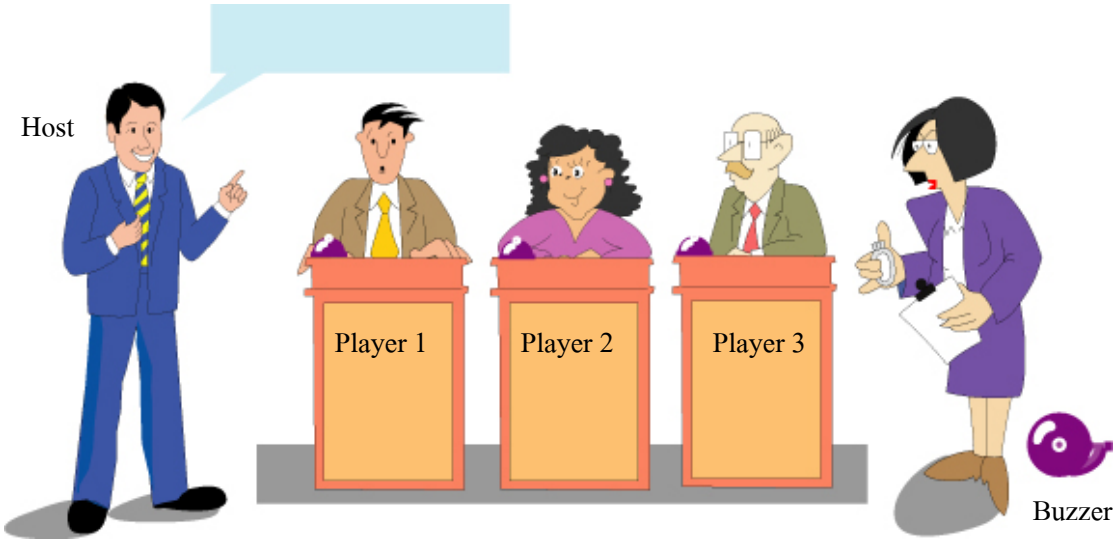


จากที่กล่าวมาข้างต้น เป็นการแนะนำวิธีใช้ซอฟ์ตแวร์ CX-Programmer ในการเขียนโปรแกรมให้กับพีแอลซี ซึ่งการเขียนนั้นสามารถทำได้ 2 แบบ คือ การใช้เมาส์กับการใช้คีย์บอร์ด แต่ท่านต้องมีพื้นฐานความรู้เกี่ยวกับคำสั่งต่างๆ เพื่อใช้ประกอบความเข้าใจ ข้อแนะนำที่อยู่ในบทนี้จะครอบคลุมการใช้งานพื้นฐานที่จำเป็นสำหรับการเขียนโปรแกรม ถ้าต้องการทราบวิธีการใช้งานที่สูงกว่านี้สามารถศึกษาเพิ่มเติมได้จาก Help และคู่มือการใช้งานของซอฟ์ตแวร์

บทที่ 7
ตัวอย่างการประยุกต์ใช้งาน

เนื้อหาในบทนี้จะกล่าวถึงตัวอย่างการประยุกต์ใช้งานโดยนำเอาคำสั่งพื้นฐานต่างๆ ที่กล่าวในบทที่ 5 มาสร้างโปรแกรมให้ได้การทำงานตามที่โจทย์กำหนดไว้ พร้อมเฉลยโปรแกรมในตัวอย่างเพื่อประกอบความเข้าใจ

7.1 ตัวอย่างการประยุกต์ใช้งานโดยใช้คำสั่ง Timer



<กติกาการเล่น>

หลังจากที่สิ้นสุดคำถามจากผู้ดำเนินรายการ (Host) ให้ผู้แข่งขัน 3 คน แย่งกันกดสวิตช์ที่อยู่ข้างหน้าเพื่อตอบคำถาม และผู้ที่กดสวิตช์ได้ก่อนจะมีเสียง Buzzer ดังขึ้นประมาณ 10 วินาที ขณะเดียวกันจะมีหลอดไฟติด ที่หน้าผู้แข่งขันที่กดก่อน โดยผู้ดำเนินรายการสามารถกดปุ่ม Reset ก่อนถึงเวลา10 วินาทีก็ได้

I/O Assignment

Input	Output
I0.00 - PB1	Q100.00 - Buzzer
I0.01 - PB2	Q100.01 - Player 1 Light
I0.02 - PB3	Q100.02 - Player 2 Light
I0.03 - RST (Reset)	Q100.03 - Player 3 Light

Ladder Diagram : Main 1 Rung 1

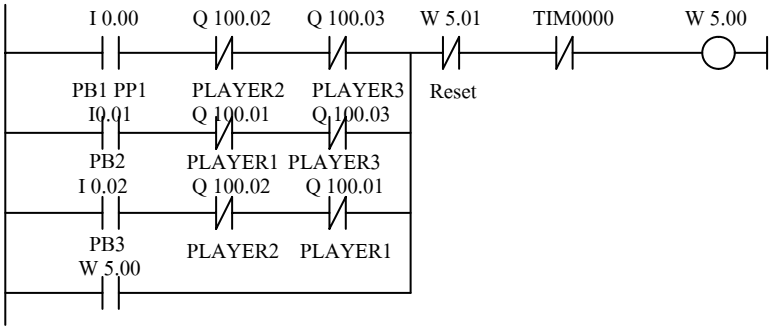
Main 1 – Who press first

(Priority Determination)

This program is to determine which players press the switch first, after the host have finished asking a question.

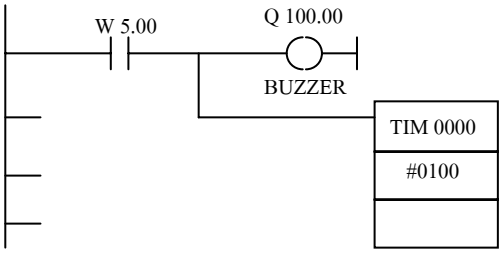
Rung 1 – Interlocked

Interlocked Rung for 3 player playing the game



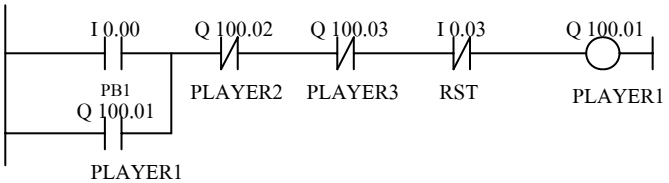
Rung 2 – Buzzer

ON Buzzer when any switch is pressed and timer will cut the buzzer after specified time



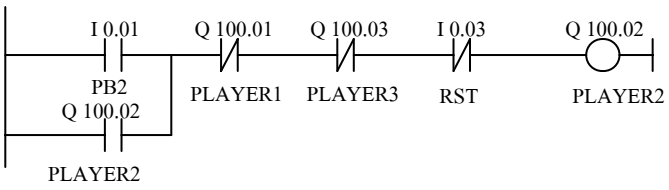
Rung 3 – Player 1

Player 1 Rung



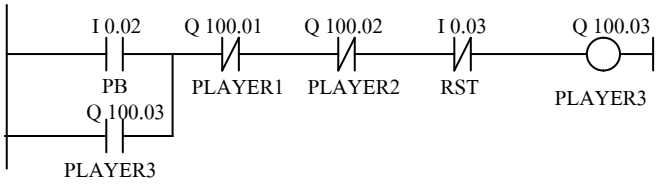
Rung 4 – Player 2

Player 2 Rung



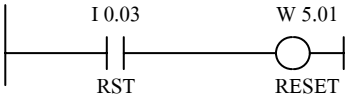
Rung 5 – Player 3

Player 3 Rung

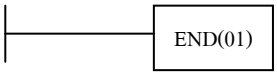


Rung 6 – Reset

Reset for the Game

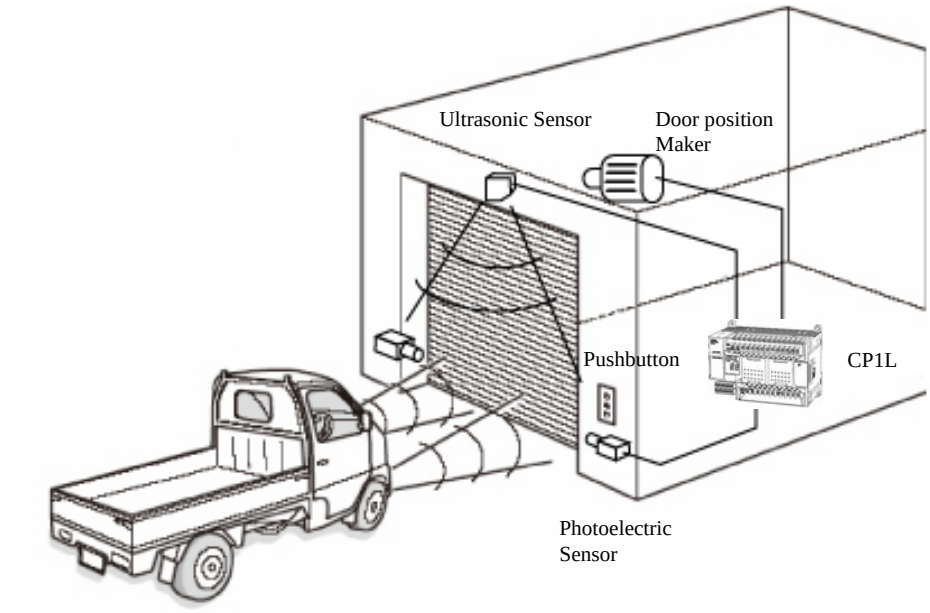


Rung 7



7.2 ตัวอย่างการควบคุมการปิด-เปิดประตู

จากรูป เมื่อรถวิ่งเข้าใกล้ประตู สัญญาณจากเซนเซอร์อุลตร้าโซนิคจะสั่งให้ประตูเปิด และเมื่อรถผ่านไป จะสั่งให้ประตูปิดตามเดิม



I/O Assignment

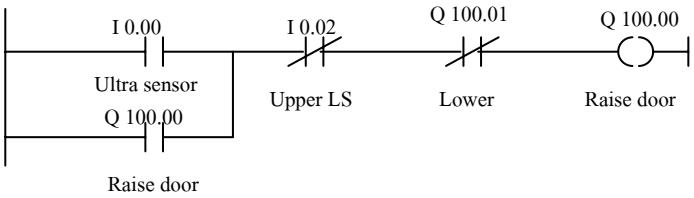
Input	***Output***
I0.00-Ultrasonic sensor	Q100.00-Motor to raise door
I0.01-Photoelectric sensor	Q100.01-Motor to lower door
I0.02-Door Upper limit switch	
I0.03-Door Lower limit switch	

Ladder Diagram : Main 1 Rung 1

Main 1 – Auto door

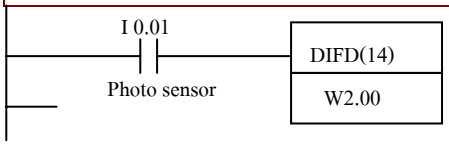
This program shows the automatic control of warehouse door.

Rung 1 – Raise door

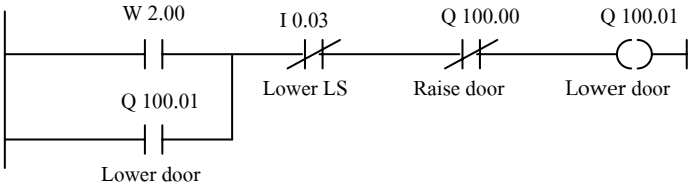


Rung 2 – Photo sensor

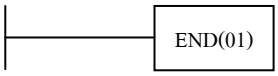
Sense unit differentiation down



Rung 3 – Lower door

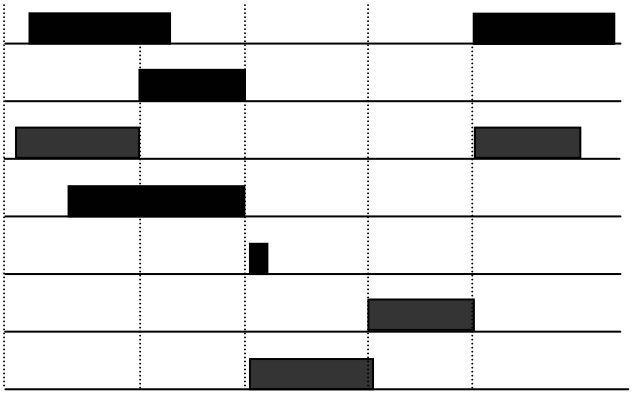


Rung 4 – End



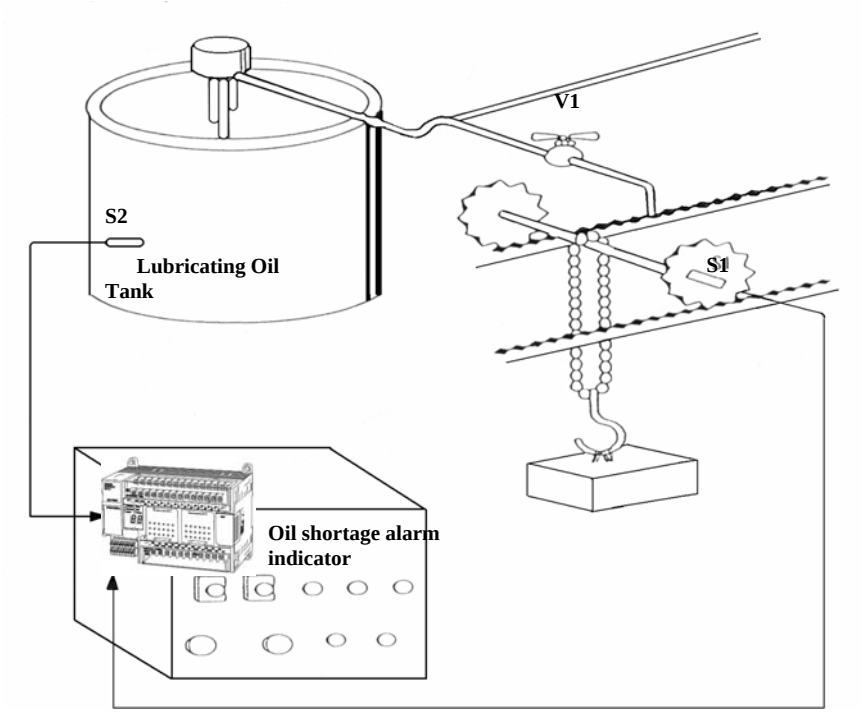
Timing diagram

I0.00	Ultrasonic switch
I0.02	Upper limit sensor
Q100.00	Motor to raise door
I0.01	Photoelectric sensor
W 2.00	DIFD
I0.03	Lower limit switch
Q100.01	Motor to low door



7.3 ตัวอย่างการควบคุมระบบ Lubrication ของเกียร์แบบอัตโนมัติ

เมื่อเกียร์เคลื่อนที่มายังตำแหน่ง S1 จะสั่งให้ Valve (V1) จ่ายน้ำมันหล่อลื่นให้กับชุดเกียร์ โดยใช้เวลาเป็นตัวสั่งหยุดจ่ายน้ำมันหล่อลื่น ถ้าน้ำมันหล่อลื่นในแท็งก์ (Tank) ลดลงต่ำกว่า Sensor (S2) ก็จะไปสั่งให้ Alarm ดังขึ้น



I/O Assignment

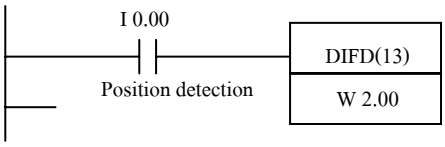
Input	*Output*
I0.00-Position detection (S1)	Q100.00-Electromagnetic valve for oil supply
I0.01-Lower limit of level (S2)	Q100.01-Oil shortage alarm indictor

Ladder Diagram : Main 1 Rung 1

Main 1 – Auto lubricate

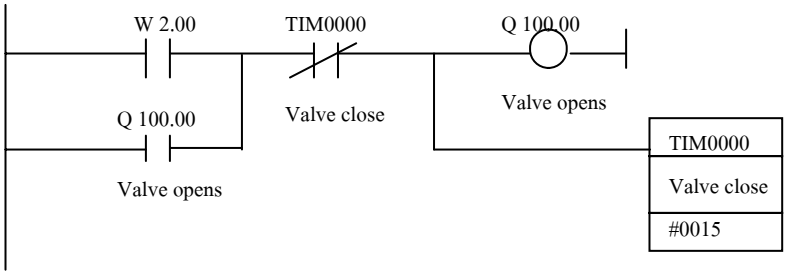
Auto lubrication of gear

Rung 1 – Start

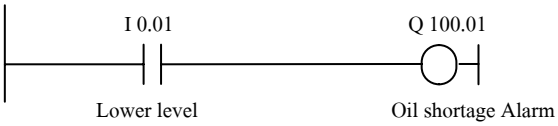


Rung 2 – Open valve

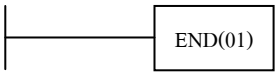
Open valve and delay 1.5 sec.



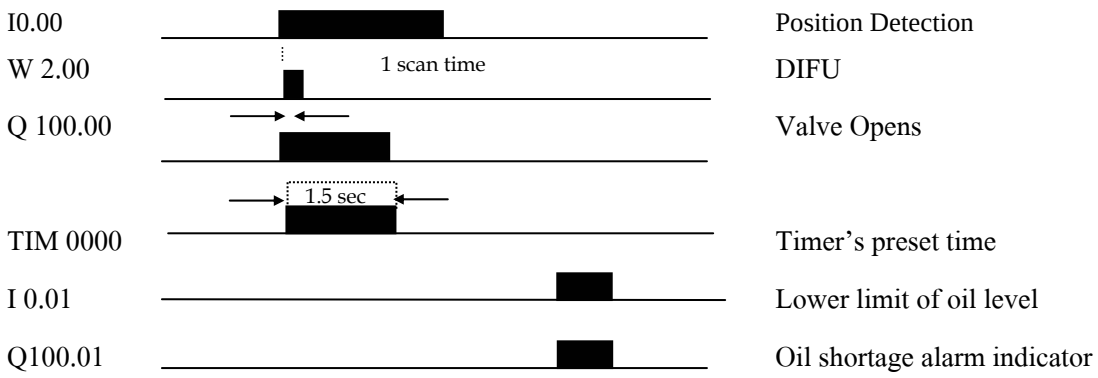
Rung 3 – Oil shortage



Rung 4 – End

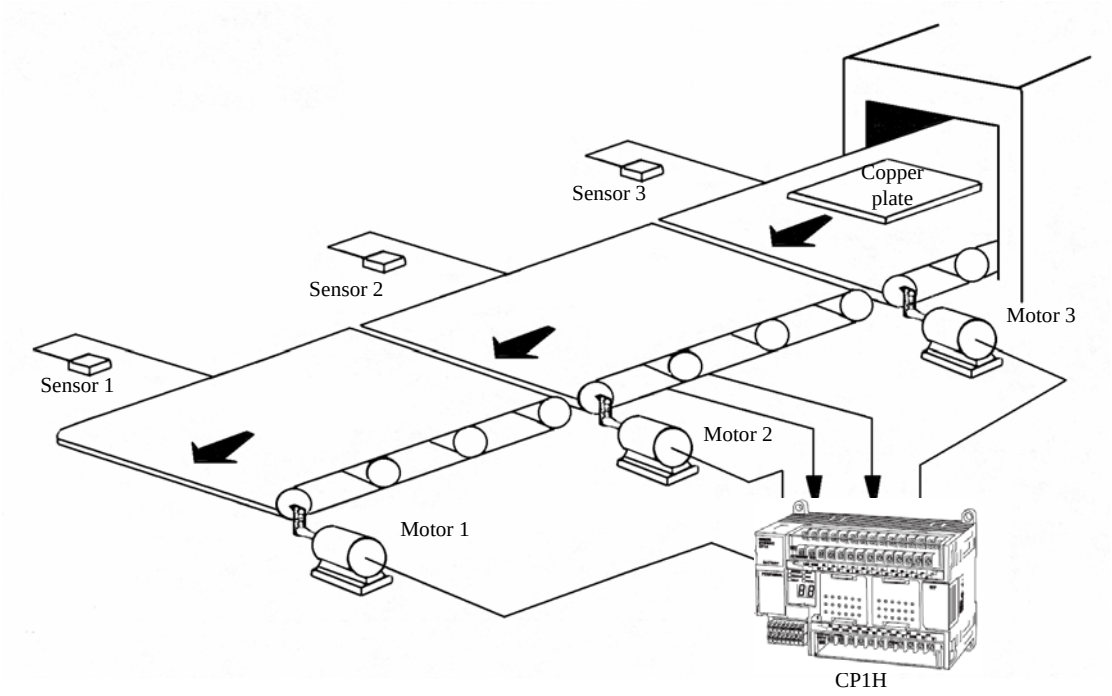


Timing diagram



7.4 ตัวอย่างการลำเลียงแผ่นทองแดงบนสายพานลำเลียง

มอเตอร์ 3 จะหมุนตลอดเวลา ขณะที่มอเตอร์ 2 จะหมุนเมื่อแผ่นทองแดงวิ่งผ่าน Sensor 3 และจะหยุดหมุนเมื่อแผ่นทองแดงวิ่งผ่าน Sensor 2 หรือหมุนครบเวลา 2 วินาที ส่วนมอเตอร์ 1 จะหมุนเมื่อแผ่นทองแดงวิ่งผ่าน Sensor 2 และจะหยุดหมุนเมื่อแผ่นทองแดงวิ่งผ่าน Sensor 1 ไปแล้ว 2 วินาที



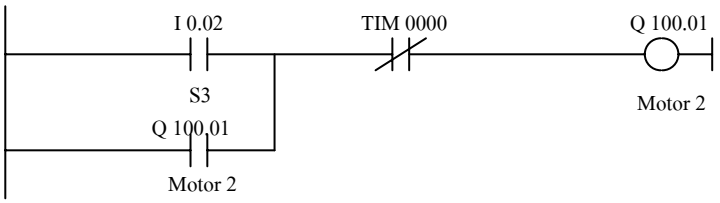
I/O Assignment

Input	***Output***
I0.00-Sensor 1	Q100.00-Motor 1
I0.01- Sensor 2	Q100.01-Motor 2
I0.02- Sensor 3	Q100.02-Motor 3

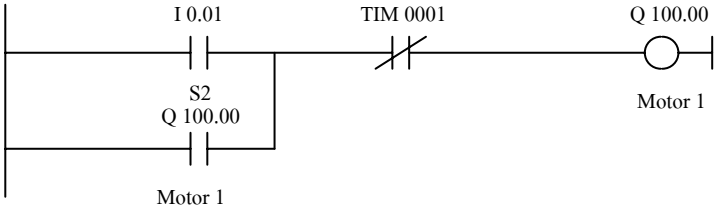
Ladder Diagram : Main 1 Rung 1

Main 1 – Conveyor control Conveyor belt control application

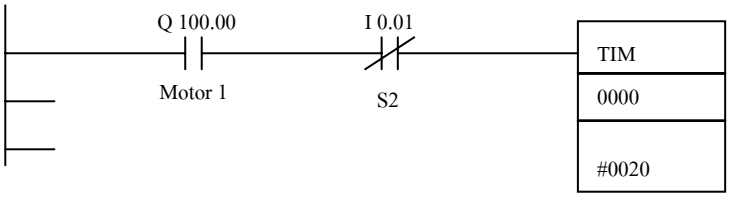
Rung 1 – Motor 2



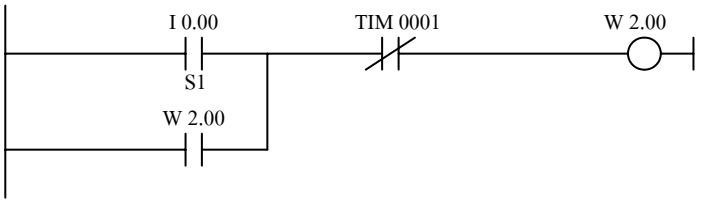
Rung 2 – Motor 1



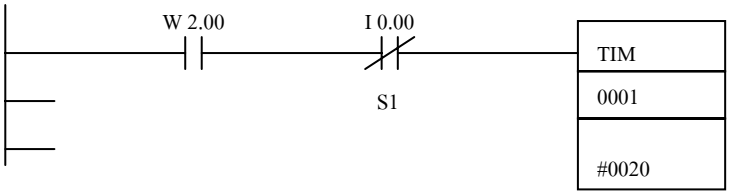
Rung 3 – Delay for 2 sec



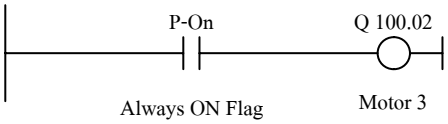
Rung 4 – Sensor 1



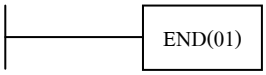
Rung 5 - Delay for 2 sec



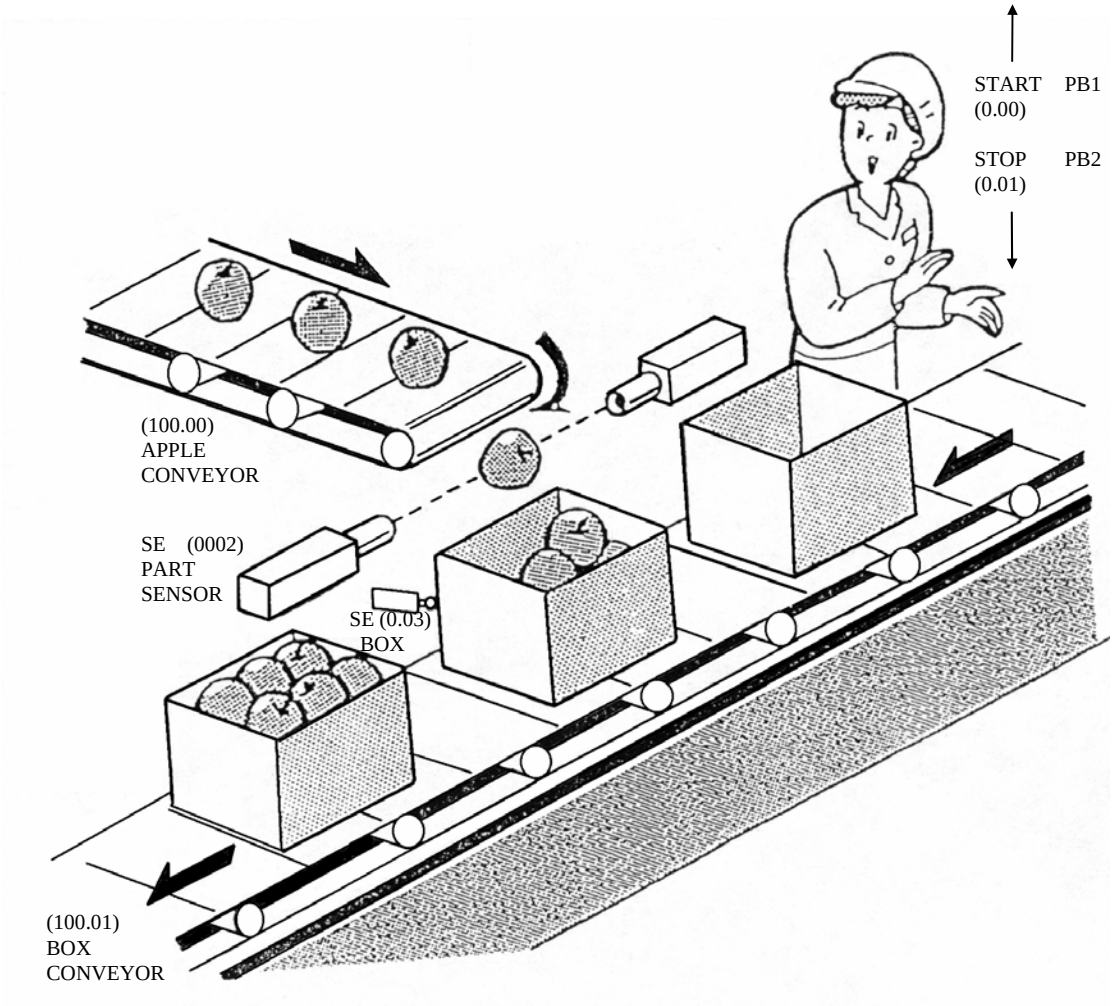
Rung 6 – Motor 3



Rung 7 – End



7.5 ตัวอย่างการใช้ Line Control ในการ Packing



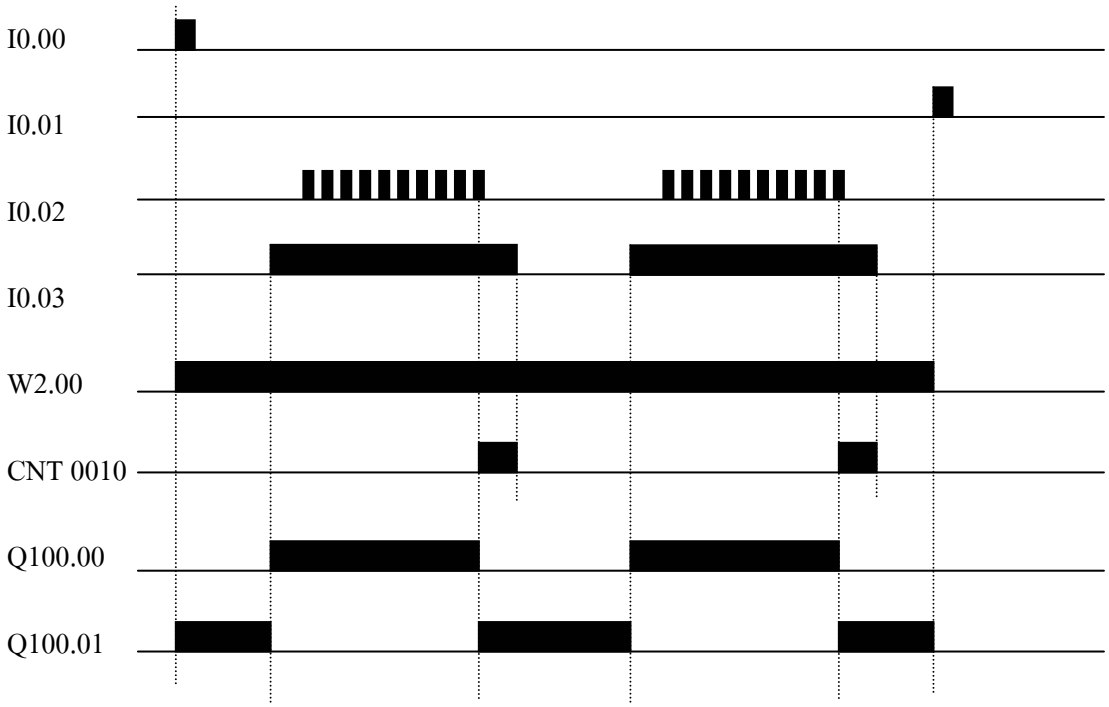
เงื่อนไขการทำงาน :

เมื่อกด PB 1 (Start) เพื่อเริ่มต้นการทำงาน กล่องที่จะใส่ลูกแอปเปิ้ลจะถูกลำเลียงมาโดยสายพานลำเลียง และจะหยุดเมื่อกล่องที่จะใส่ลูกแอปเปิ้ลมาบัง Sensor (SE2) หลังจากนั้นสายพานลำเลียงแอปเปิ้ลจะลำเลียงแอปเปิ้ลลงกล่องจำนวน 10 ลูก ซึ่งเช็คโดย Sensor (SE1) เมื่อครบ 10 ลูกแล้ว สายพานลำเลียงแอปเปิ้ลจะหยุดหมุน และสายพานลำเลียงที่ลำเลียงกล่องแอปเปิ้ลจะหมุนกล่องที่ใส่แอปเปิ้ลใบใหม่ เข้ามาแทนที่และระบบการทำงานจะเป็นอย่างนี้เรื่อยไปจนกว่าจะกด PB2 (Stop) เพื่อหยุดการทำงาน

I/O Assignment

Input	*Devices*
I0.00	START Push button (PB1)
I0.01	STOP Push button (PB2)
I0.02	Part Present (SE1)
I0.03	Box Present (SE2)

Output	*Devices*
Q100.00	Apple Conveyor
Q100.01	Box Conveyor



Mnemonic Codes

Address	*Instruction*	*Data*
0000	LD	0.00
0001	OR	W 2.00
0002	AND NOT	0.01
0003	OUT	W 2.00
0004	LD	W 2.00
0005	AND NOT	100.01
0006	OUT	100.00
0007	LD	0.02

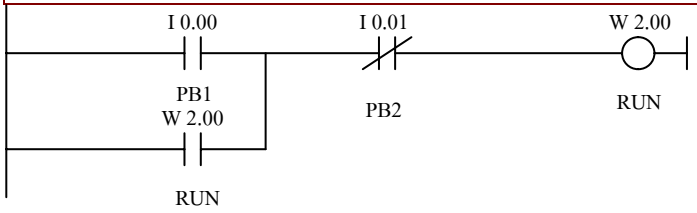
Address	*Instruction*	*Data*
0008	LD NOT	0.03
0009	CNT	0010
		# 0010
0010	LD CNT	0010
0011	OR NOT	0.03
0012	AND	W 2.00
0013	OUT	100.01
0014	END (01)	

Ladder Diagram : Main 1 Rung 1

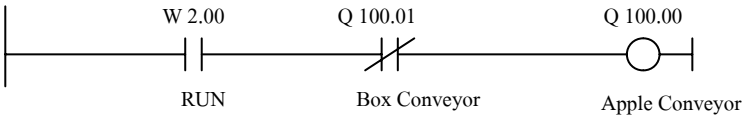
Main 1 – Packing

Packing line control for Apples

Rung 1 – Start condition

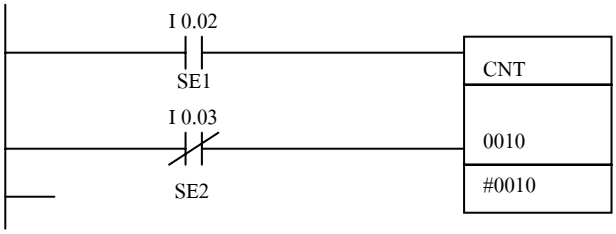


Rung 2 – Apples conveyor

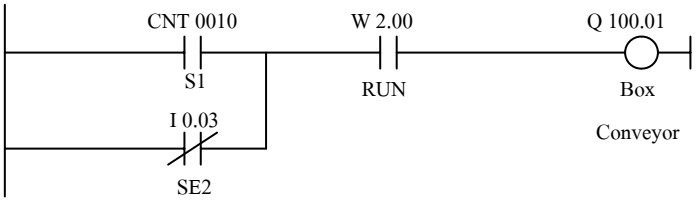


Rung 3 – Counter

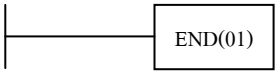
Counter preset at 10



Rung 4 – Box conveyor

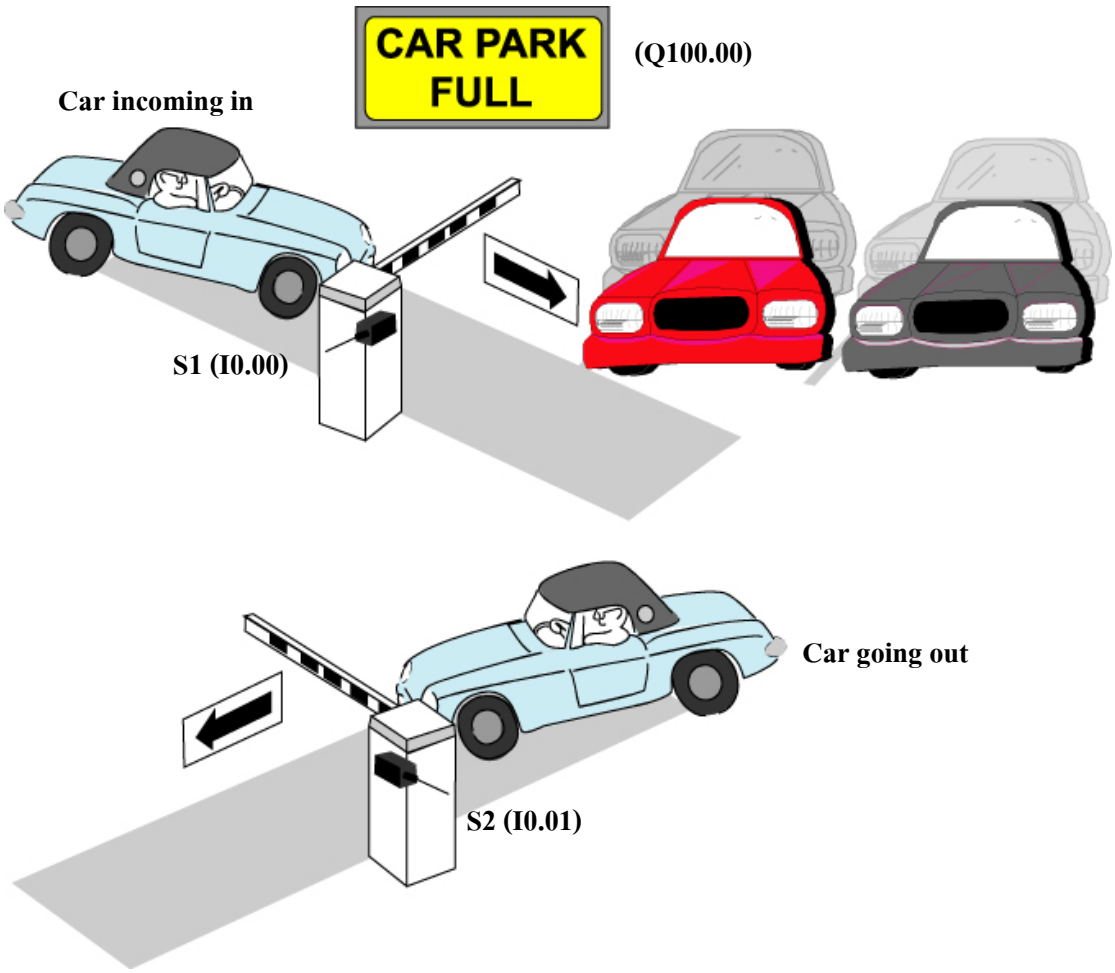


Rung 5 – END



7.6 ตัวอย่างการควบคุมจำนวนรถในลานจอดรถ

ในลานจอดรถแห่งนี้ สามารถจอดรถได้เพียงแค่ 100 คัน ตลอดเวลาจะมีรถเข้า-ออก ที่ทางเข้าจะมี Sensor (S1) และทางออกจะมี Sensor (S2) พร้อมกับมีป้ายแสดง (หลอดไฟนีออน) แสดงว่ารถเต็มลานจอดรถแล้วเมื่อครบ 100 คัน



I/O Assignment

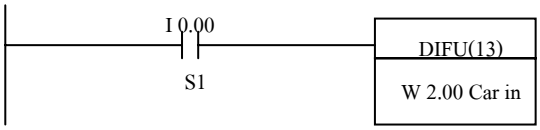
INPUT	*OUTPUT*
I0.00 - Sensor S1	Q100.00 - Car park full sign
I0.01- Sensor S2	

Ladder Diagram : Main 1 Rung 1

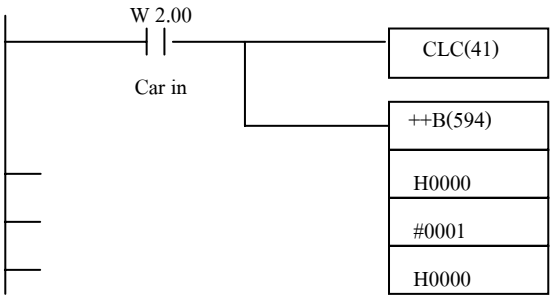
Main 1 – Car Park Control

Application: Car Park Control

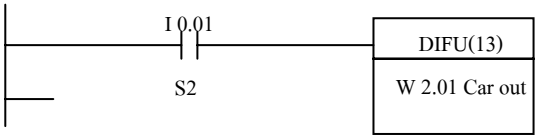
Rung 1 – Car in



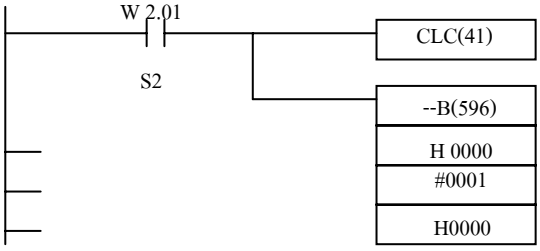
Rung 2 – Add 1



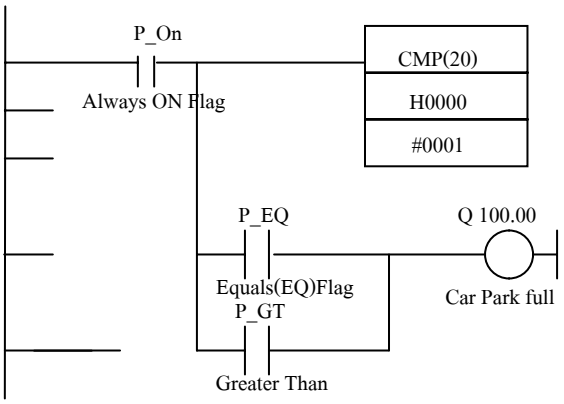
Rung 3 – Car out



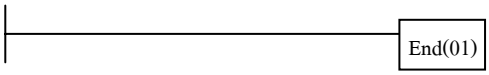
Rung 4 – Subtract 1



Rung 5 – Compare



Rung 6 - End



บทที่ 8

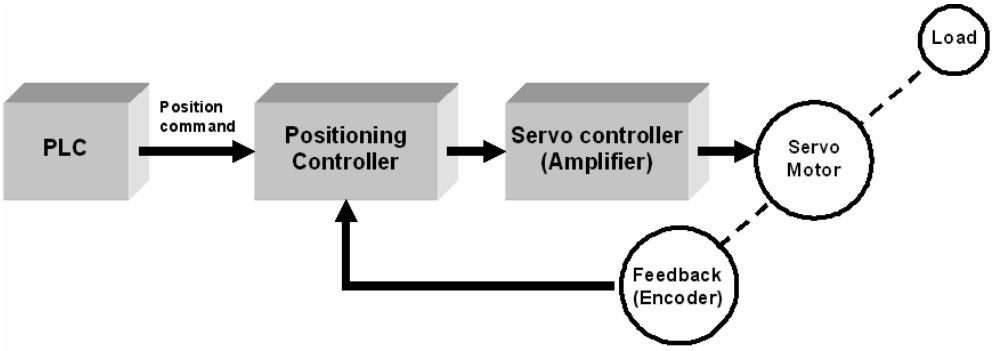
การประยุกต์ใช้งานกับเซอร์โวมอเตอร์

ในบทนี้เหมาะสำหรับท่านที่มีประสบการณ์การใช้งาน PLC มาบ้างแล้ว เนื้อหาจะกล่าวถึงการควบคุมเซอร์โวมอเตอร์ด้วย PLC อย่างง่ายๆ โดยการประยุกต์ใช้ฟังก์ชันบล๊อก ภายในเนื้อหาจะไม่ลงรายละเอียดเกี่ยวกับหลักการทำงานของเซอร์โวมอเตอร์ แต่จะเน้นที่การต่อใช้งานและเขียนโปรแกรมควบคุมเท่านั้น

8.1 หลักการควบคุมเบื้องต้น

ในรูปที่ 8.1 แสดงบล๊อกไดอะแกรมของระบบควบคุมเซอร์โวมอเตอร์เบื้องต้น โดยปกติระบบควบคุมเซอร์โวมอเตอร์จะมีองค์ประกอบหลักในการทำงานดังนี้

- **มอเตอร์** : ทำหน้าที่ขับโหลดมีทั้งชนิดที่มีเบรคและไม่มีเบรค
- **เอ็นโค้ดเดอร์** : ติดอยู่กับตัวมอเตอร์เพื่อทำหน้าที่ป้อนกลับการเคลื่อนที่ของมอเตอร์ ดังนั้นเราสามารถทราบตำแหน่งและความเร็วของการหมุนได้จากเอ็นโค้ดเดอร์นี้
- **เซอร์โวไดร์เวอร์** : ในปัจจุบันจะรวมส่วนที่เป็นเซอร์โวคอนโทรลเลอร์และ Positioning controller ไว้ด้วยกันซึ่งเซอร์โวคอนโทรลเลอร์จะทำหน้าที่เป็นตัวจ่ายไฟให้กับมอเตอร์ ส่วน Positioning controller จะทำหน้าที่ควบคุมตำแหน่งการเคลื่อนที่ของมอเตอร์โดยรับคำสั่งมาจากอุปกรณ์ภายนอก เช่น PLC เป็นต้น
- **PLC** : จะทำหน้าในการส่งคำสั่งไปยังเซอร์โวไดร์เวอร์ในรูปแบบต่างๆ เช่นอนาลอกและพลัส เป็นต้น จากนั้นเซอร์โวไดร์เวอร์จะควบคุมให้มอเตอร์หมุนให้ได้ตำแหน่งและความเร็วตามที่ได้รับคำสั่ง

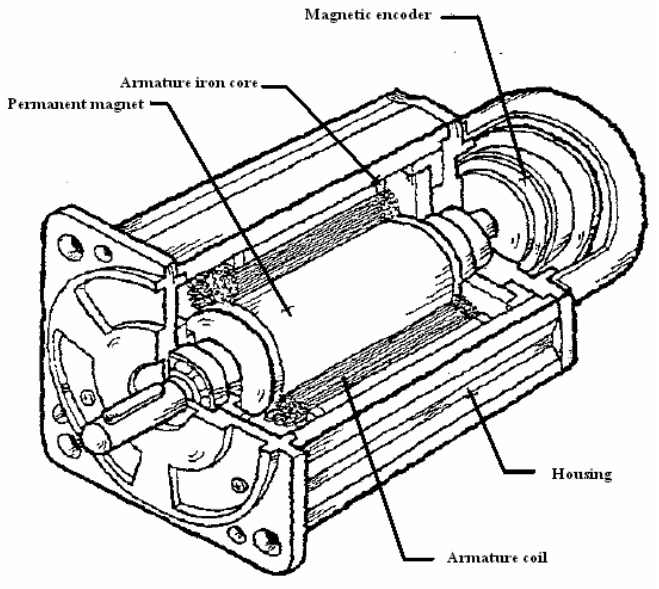


รูปที่ 8.1 หลักการควบคุมเซอร์โวมอเตอร์

จากรูป 8.1 การทำงานจะเริ่มโดย PLC ส่ง Position command ซึ่งเป็นสัญญาณพลัสหรืออนาลอกให้กับ Position controller ที่อยู่ในเซอร์โวไดว์เวอร์ จากนั้น Position controller จะสั่งให้ Amplifier จ่ายกระแสไฟให้กับมอเตอร์เพื่อทำให้มอเตอร์หมุนให้ได้ความเร็วและระยะตามคำสั่ง เอ็นโค้ดเดอร์ที่ติดอยู่กับมอเตอร์จะทำหน้าที่ป้อนกลับข้อมูลระยะและความเร็วในการหมุนกลับไปที่ Position controller ซึ่งมันจะมี Counter ทำหน้าที่เปรียบเทียบกับคำสั่งที่ได้รับจาก PLC ถ้ายังมีความแตกต่างกันมันส่งสัญญาณไปที่ Amplifier เพื่อสั่งให้มอเตอร์หมุนให้ได้ระยะและความเร็วตามต้องการ

8.2 โครงสร้างของเซอร์โวมอเตอร์

เซอร์โวมอเตอร์ของออมรอนเป็นประเภท AC Servo Motor แบบ Synchronous Servo Motor มีลักษณะของโครงสร้างดังนี้



รูปที่ 8.2แสดงโครงสร้างของเซอร์โวมอเตอร์

ส่วนของสเตเตอร์เป็นขดลวดพันในร่องสล็อต ส่วนโรเตอร์เป็นแม่เหล็กถาวร ดังนั้นความสัมพันธ์ของค่าพารามิเตอร์ ต่างๆ จะคล้าย DC motor และมอเตอร์ประเภทนี้ไม่มีแปรงถ่านบางทีจึงเรียกว่า เซอร์โวมอเตอร์แบบ DC Brushless ส่วนตัวเอ็นโค้ดเดอร์จะต่ออยู่กับเพลาเดียวกับโรเตอร์

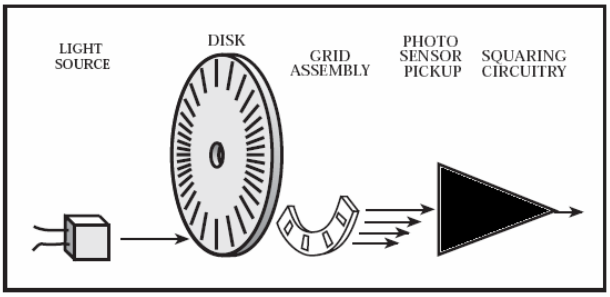
8.3 หลักการทำงานของเอ็นโค้ดเดอร์

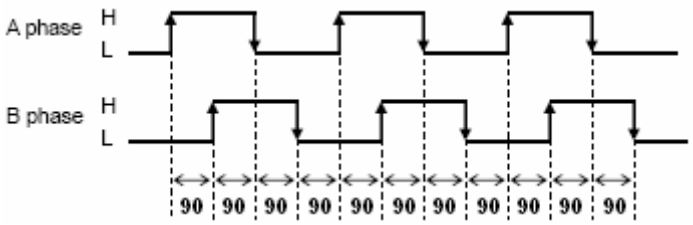
เอ็นโค้ดเดอร์ที่นิยมใช้กับเซอร์โวมอเตอร์มีอยู่ด้วยกัน 2 ชนิด ดังนี้

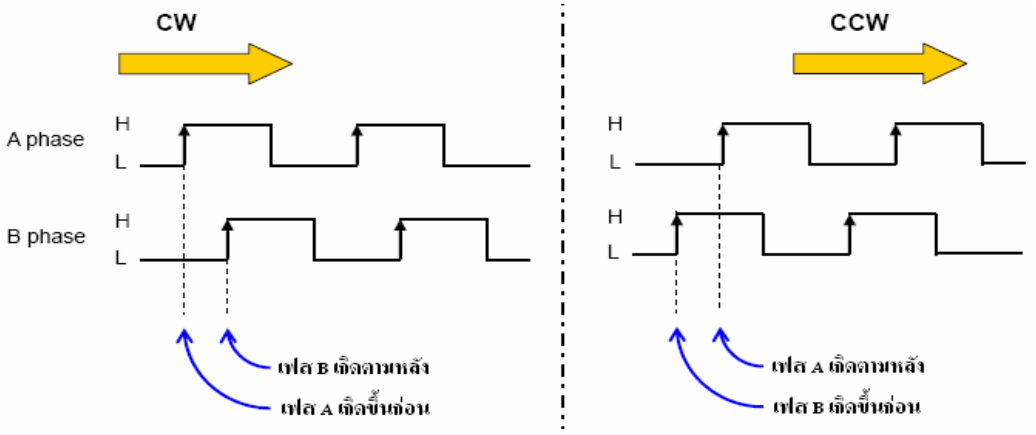
- ***เอ็นโค้ดเดอร์แบบ Incremental***

จากรูปด้านล่าง แสดงส่วนประกอบของเอ็นโค้ดเดอร์ชนิดนี้ โดยลำแสงจะถูกยิงจาก lighting diode ผ่าน fixed disc ไปยัง rotation disc ที่ติดตั้งอยู่บนแกนเพลา โดยมี photo diode เป็นตัวรับแสง ลำแสงจะผ่านรูบน fixed disc และ rotation disc ตามจังหวะของการหมุนทำให้ได้สัญญาณไฟฟ้าออกมาจาก photo sensor เนื่องจากรูของ A และ B บน fixed disc จะต่างเฟสกันอยู่ 90 องศา ดังนั้นสัญญาณเอาต์พุตทางไฟฟ้าจะได้รูปคลื่นที่ออกมาต่างเฟสกันอยู่ 90 องศา ตามรูป ส่วนรูของ Z บนfixed disc จะมีเพียงรูเดียวเท่านั้น

ถ้านับค่าเพาส์ที่ได้จากตัวเอ็นโค้ดเดอร์จะเป็นค่ามุมของการหมุนนั่นเอง ส่วนเอาต์พุตเฟส A และ B ที่ต่างเฟสกันอยู่ 90 องศาจะเป็นตัวชี้ถึงทิศทางการหมุนของมอเตอร์ ส่วนเฟส Z หรือ Zero signal เป็นตัวชี้ถึงจุด 0 องศาของการหมุน







รูปที่ 8.3แสดงโครงสร้างและกราฟการทำงานของเอ็นโค้ดเดอร์

- ***เอ็นโค้ดเดอร์แบบ Absolute***

โดยทั่วไป absolute encoder จะมีเอาต์พุตโค้ดให้เลือกเช่น gray code, binary หรือ BCD code แต่สำหรับการเลือกประเภทของ detector ของตัวเซอร์โว ไม่จำเป็นต้องเลือกโค้ดของเอาต์พุต ให้เลือกแต่เพียงความละเอียดที่ต้องการใช้งานเท่านั้น จะสังเกตเห็นว่าสิ่งที่แตกต่างจาก incremental encoder ก็คือจำนวนของสายสัญญาณเอาต์พุตที่มีจำนวนมากกว่าแบบ incremental ซึ่งจะขึ้นอยู่กับความละเอียดที่เลือกใช้ และอีกประการหนึ่งคือ ความหมายของสัญญาณของ absolute encoder ณ เวลาหนึ่งจะให้ค่าออกมาเป็นค่าสมบูรณ์ ไม่ใช่เป็นค่าที่เปรียบเทียบจากจุดเริ่มต้นเหมือนกับแบบ incremental encoder ดังนั้นถ้าเซอร์โวมอเตอร์ที่มี detector แบบ absolute encoder ก็ไม่จำเป็นจะต้อง search หาตำแหน่งเริ่มต้น (origin search) ใหม่ทุกครั้งที่ ปิดเครื่องแล้วเปิดเครื่องขึ้นมาใช้งานใหม่

8.4 ชนิดของอินพุตควบคุมสำหรับเซอร์โวไดว์เวอร์

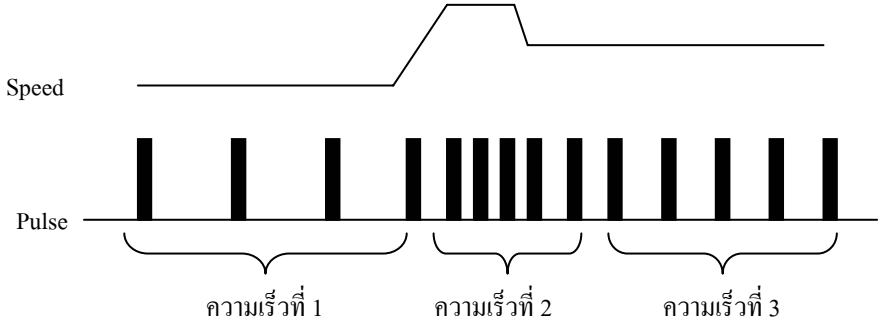
อินพุตของเซอร์โวไดว์เวอร์ ที่ต่อใช้งานกับ PLC มีอยู่ด้วยกันหลายแบบแต่ที่นิยมมากที่สุด คือ แบบ Pulse Train และ Linear (หรือ Analog) ซึ่งทั้ง 2 แบบจะเหมาะสมกับงานที่แตกต่างกัน แบบ Pulse trainจะเหมาะกับงานควบคุมตำแหน่ง เช่น Feed-to-Cut และ Pick & Place เป็นต้น ส่วนแบบ Linear จะใช้กับงานที่เคลื่อนที่เป็นเส้นโค้งหรือวงกลม ในที่นี้เราจะเน้นการใช้เซอร์โวไดว์เวอร์ที่รับสัญญาณอินพุตแบบ Pulse train เพื่อให้ง่ายในการทำความเข้าใจสำหรับผู้เริ่มต้น

8.4.1 PULSE TRAIN CONTROL

จากรูปที่ 8.1 PLC จะส่งสัญญาณเป็นพลัสให้กับเซอร์โวไดว์เวอร์จากนั้นไดว์เวอร์จะควบคุมการหมุนของมอเตอร์ให้ได้ตามการสั่งงานของ PLC ระยะในการเคลื่อนที่หรือการหมุนของเซอร์โวมอเตอร์จะขึ้นอยู่กับจำนวนพลัสที่ PLC ส่งให้ ขณะเดียวกันความเร็วของเซอร์โวมอเตอร์จะขึ้นอยู่กับความเร็วหรือความถี่ของพลัสที่ส่งมาจาก PLC ดังแสดงในรูปที่ 8.4

ระยะการหมุน α จำนวนพลัส

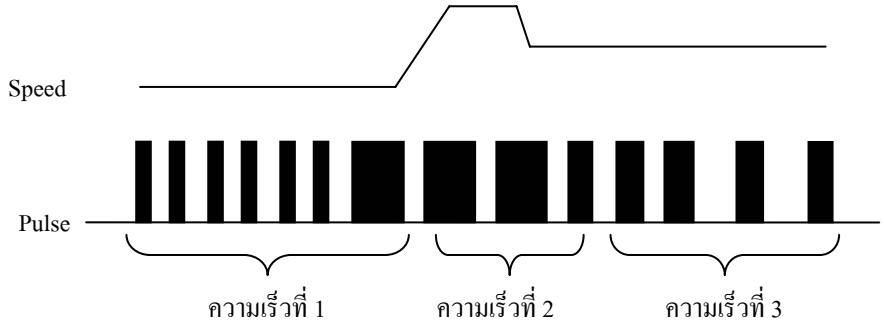
ความเร็ว α ความถี่ของพลัส



***รูปที่ 8.4** กราฟแสดงพลัสที่จ่ายให้กับเซอร์โวไดว์เวอร์*

8.4.2 PULSE WIDTH MODULATION (PWM)

การทำงานจะคล้ายๆ กับ Pulse Train คือ ระยะทางการหมุนจะขึ้นอยู่กับจำนวนพลัส แต่ความเร็วของมอเตอร์จะขึ้นอยู่กับความกว้างของพลัสดังแสดงในรูปที่ 8.5



***รูปที่ 8.5** กราฟแสดงพลัสที่จ่ายให้กับเซอร์โวไดว์เวอร์*

8.4.3 LINEAR

การควบคุมแบบนี้จะแตกต่างจากทั้งสองแบบที่กล่าวมาข้างต้น เพราะ PLC จะส่งสัญญาณอนาลอกให้กับเซอร์โวไดว์เวอร์ ซึ่งไดว์เวอร์จะต้องเป็นชนิดที่สามารถรับสัญญาณควบคุมเป็นอนาลอกได้ เช่น ± 10Vdc

8.5 การสร้างระบบควบคุมของเซอร์โวมอเตอร์

ในหัวข้อที่กล่าวมาข้างต้นเป็นการอธิบายหลักการกว้างๆ ของเซอร์โวมอเตอร์ ต่อไปนี้เราจะเข้าภาคปฏิบัติที่สามารถนำไปใช้งานได้จริง แต่จะเน้นเฉพาะระบบควบคุมเป็นหลัก เช่น PLC และเซอร์โวไดว์เวอร์ เป็นต้น จะไม่ขอกล่าวถึงระบบทางกล เช่น Ball screw, สายพาน รวมถึงการคำนวณทางกล เช่น Torque

เพื่อให้ง่ายต่อความเข้าใจเราจะใช้เซอร์โวมอเตอร์รุ่น Smart Step กับ CP1H เป็นตัวอย่าง ซึ่งท่านสามารถนำเอาตัวอย่างดังกล่าวนี้ไปประยุกต์ใช้งานได้จริง

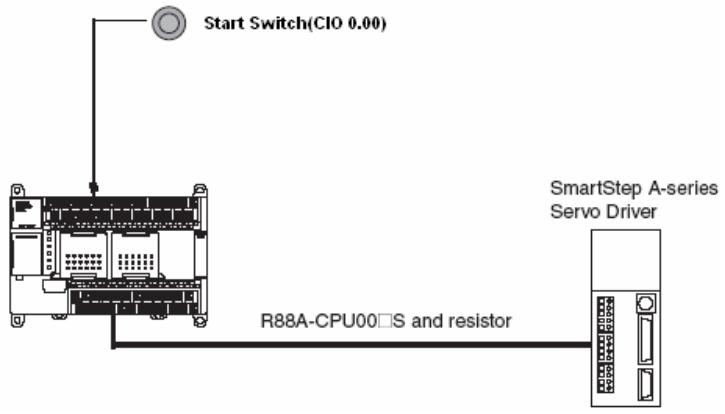
ตัวอย่างที่ 1 ระบบควบคุมการเคลื่อนที่แบบ Incremental

การเคลื่อนที่แบบ Incremental เป็นระบบที่เข้าใจได้ง่าย การหมุนของมอเตอร์จะเริ่มที่จุดใดก็ได้แต่ระยะการหมุนจะขึ้นอยู่กับจำนวนพลัสที่ PLC ส่งให้ Driver ส่วนทิศทางจะขึ้นอยู่กับเครื่องหมายบวกหรือลบของค่าจำนวนพลัส เช่น -1000 หมายถึง หมุนทวนเข็มนาฬิกา 1000 พลัส ในตัวอย่างนี้สมมติว่ามอเตอร์ต่อกับบอลสกรูที่หมุน 1 รอบ มีระยะการเคลื่อนที่ 10 มม. และตั้งค่า Resolution ของ Driver ไว้ที่ 1000 พลัสต่อ 1 รอบ ถ้า PLC ส่งพลัสให้ Driver เท่ากับ 10,000 พลัส มอเตอร์จะหมุนไป 10 รอบหรือ 10 ซม. ต่อนี้ไปคือขั้นตอนต่างๆ การทำระบบควบคุมเซอร์โวมอเตอร์แบบ Incremental (หรือ Relative) อย่างง่าย

- **ขั้นตอนที่ 1 การจัดองค์ประกอบ (Configuration)**

  ในรูปที่ 8.6 แสดง Configuration ของระบบที่ต้องจะควบคุมโดยใช้สวิตช์ 2 ตัว ทำหน้าที่ในการควบคุม

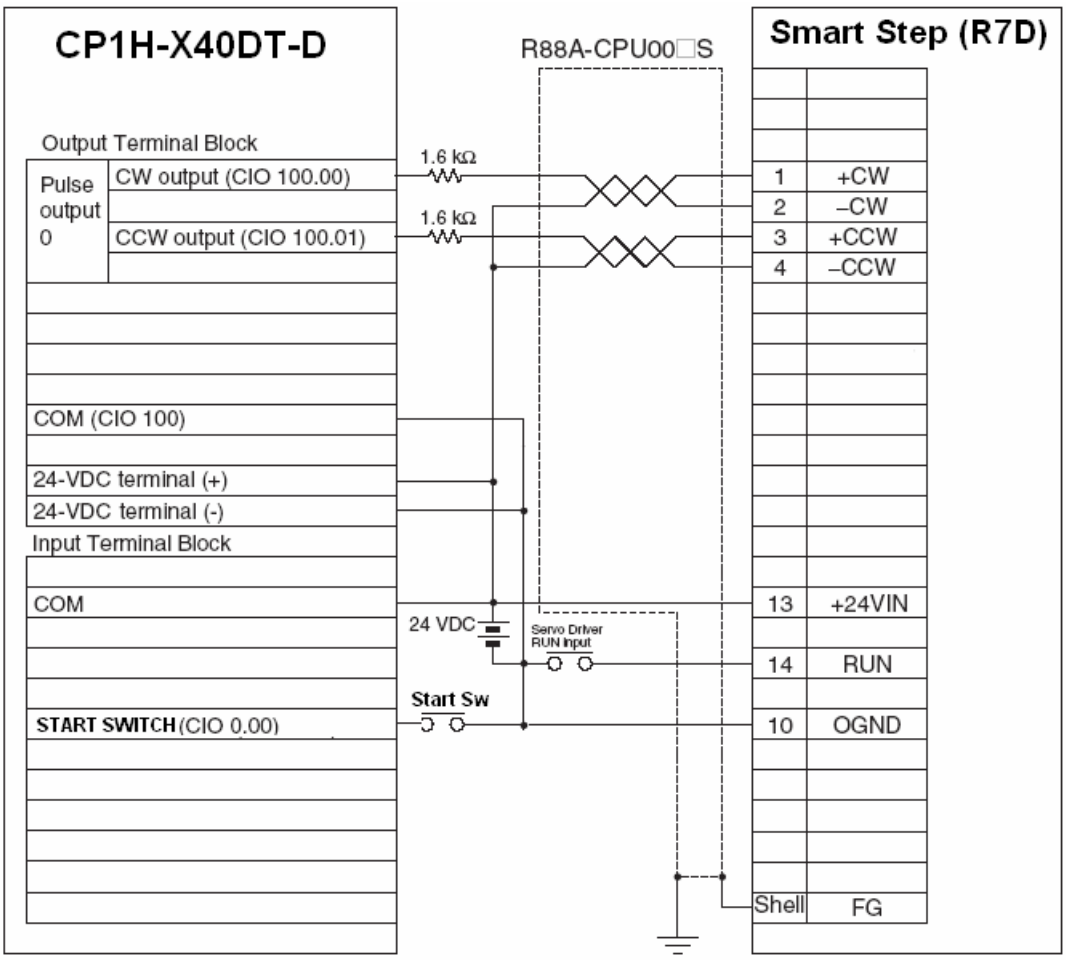
  START Switch คือ สวิตช์ที่ทำหน้าที่สั่งให้เซอร์โวมอเตอร์เริ่มทำงาน



รูปที่ 8.6 รูปแสดงระบบควบคุมเซอร์โวมอเตอร์

- **ขั้นตอนที่ 2 การเดินสายไฟ**

จากรูปที่ 8.6 เราสามารถเขียนเป็นผังการเดินสายไฟได้ดังรูปที่ 8.7 ในรูปนี้เป็นการแสดงการต่อสายไฟระหว่าง CP1H กับ Driver (รุ่น Smart Step) เท่านั้นเพราะว่าการเดินสายไฟระหว่าง Driver กับ Motor นั้นทำได้ง่ายเนื่องจากเป็นสายสำเร็จรูป



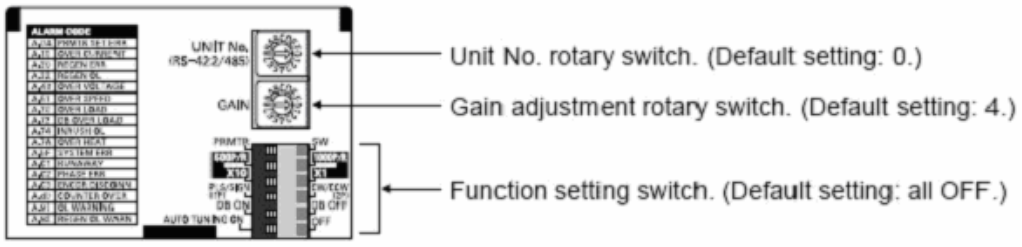
***รูปที่ 8.7** รูปแสดงการเดินสายไฟระหว่าง PLC กับ Driver*

- **ขั้นตอนที่ 3 การตั้งค่าการทำงานของระบบ**

หลังจากทำการติดตั้งอุปกรณ์และเดินสายไฟแล้ว สิ่งที่ท่านควรทำในลำดับถัดไป คือ การตั้งค่าการทำงานให้กับอุปกรณ์ต่างๆ ให้เหมาะสมกับการใช้งาน เช่น เซอร์โวไดว์เวอร์และการตั้งค่า Setup ของ PLC ในกรณีตัวอย่างนี้จะตั้งค่าเฉพาะที่ Driver เท่านั้น โดยมีขั้นตอนดังต่อไปนี้

การตั้งค่า Servo driver

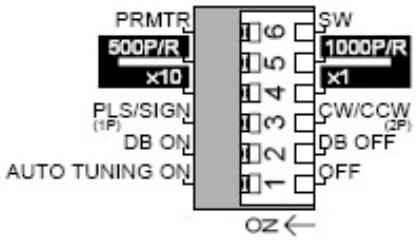
สำหรับ Servo driver รุ่น Smart Step สามารถทำการปรับตั้งค่าได้ง่าย เพียงแค่เลือกสวิตช์บริเวณหน้าตัวเครื่องก็สามารถเลือกโหมดการทำงานได้ แต่ถ้าต้องการใช้ซอฟ์ตแวร์ก็สามารถทำได้เช่นกัน ในที่นี้เราจะไม่กล่าวถึงการตั้งค่าด้วยซอฟ์ตแวร์



- Gain Adjustment Rotary Switch: GAIN

สวิตช์นี้ใช้เพื่อปรับความไวของการตอบสนองของ Servo Motor โดยยิ่งให้ค่ามาก Servo motor ก็จะตอบสนองได้เร็วขึ้น แต่ก็จะทำให้การเคลื่อนที่ราบลื่นลดลง ตรงข้ามกับการตั้งค่าที่น้อยกว่า Servo motor จะสามารถเคลื่อนที่ได้ราบลื่นกว่า แต่ตอบสนองช้าลง **ให้ตั้งเป็น "4"**

- Function Setting Switch



Switch 6: ใช้เลือกว่าจะใช้ค่าตั้งจากสวิตช์หรือซอฟต์แวร์

Switches 5 และ 4: สำหรับตั้งค่า Resolution

Switch 3: ตั้งค่าชนิดของอินพุตพลัส

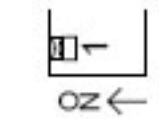
Switch 2: ตั้งค่าdynamic brake

Switch 1: ใช้ทำ online autotuning

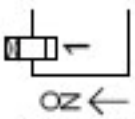
<u>ข้อควรระวัง</u> ควรปิดเครื่องก่อนทำการปรับตั้งค่าต่างๆ

การ เปิด – ปิด สวิตช์

ให้ใช้อุปกรณ์ที่สามารถดันสวิตช์ได้ เช่น ใช้ไขควงแบนขนาดเล็ก การดันสวิตช์ไปทางซ้ายคือการเปิด ส่วนสวิตช์ที่อยู่ในตำแหน่งคือการปิด หากสวิตช์ทั้ง 6 ปิดอยู่ เป็นการตั้งค่า default ให้กับ Servo drive



<u>Switch turned OFF</u>



<u>Switch turned ON</u>

มีสวิตช์หลายตัว แต่ในกรณีนี้ให้ตั้งเฉพาะที่จะใช้งานเท่านั้น ถ้าต้องการรายละเอียดเพิ่มเติมเกี่ยวกับการตั้งค่าให้ดูได้ที่ภาคผนวก F

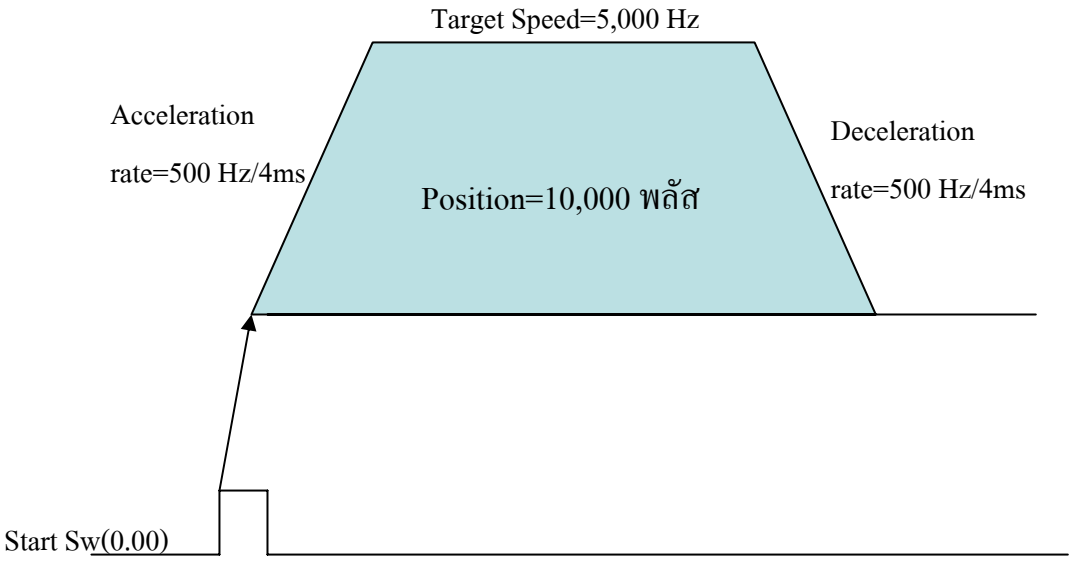
- Resolution Setting (Switches 4 and 5)

สวิตช์ 4 และ 5 นี้ใช้ในการปรับตังค่าความละเอียดของการหมุนของ Servo motor โดยกำหนดเป็นจำนวน pulse ต่อการหมุนของ Servo motor 1 รอบ

Switch		**Resolution Setting**
5	**4**	
OFF	**OFF**	1,000 pulse/revolution (0.36º/step) **ให้ตั้งที่ตำแหน่งนี้**
OFF	ON	10,000 pulse/revolution (0.036º/step)
ON	OFF	500 pulse/revolution (0.72º/step)
ON	ON	5,000 pulse/revolution (0.072º/step)

- **ขั้นตอนที่ 4 การสร้างรูปแบบการทำงาน (Operation pattern)**

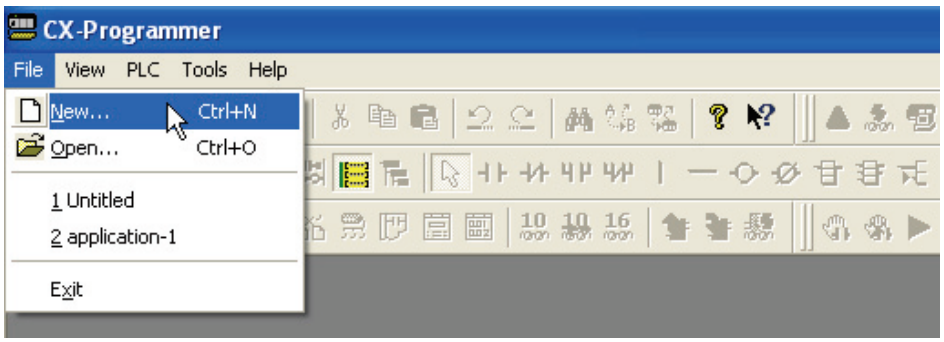
รูปแบบการเคลื่อนที่แสดงได้ดังรูปข้างล่างนี้ เมื่อมีสัญญาณ Start Sw(0.00) เข้ามา PLC จะส่งสัญญาณพลัสให้กับ Driver ที่อัตราเร่ง 500Hz/4ms จนได้ความเร็วที่ต้องการ (Target speed) จากนั้นจะเข้าสู่ช่วงอัตราหน่วง 500 Hz/4ms และหยุดหมุนเมื่อครบ 10,000 พลัส



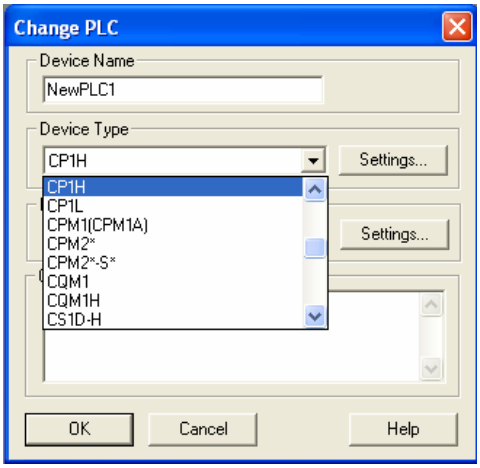
***รูปที่ 8.8** รูปแสดงรูปแบบการเคลื่อนที่ของมอเตอร์ที่ต้องการ*

- **ขั้นตอนที่ 5 การเขียนโปรแกรม PLC**

1. เรียกโปรแกรม CX-programmer จากนั้นเลือก [File] - [New] จากเมนู จะปรากฏไดอะลอกบล๊อกดังรูปข้างล่างนี้

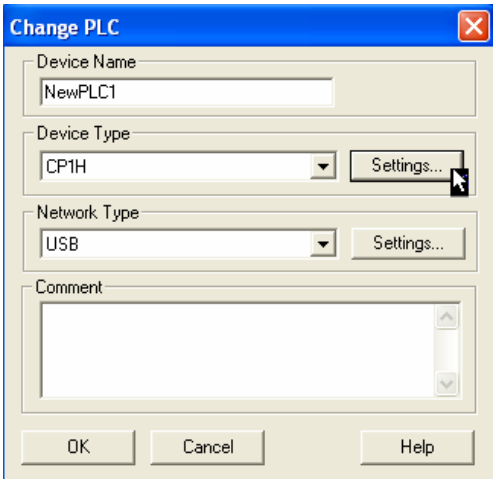


2. เลือก [CP1H] จาก Device Type



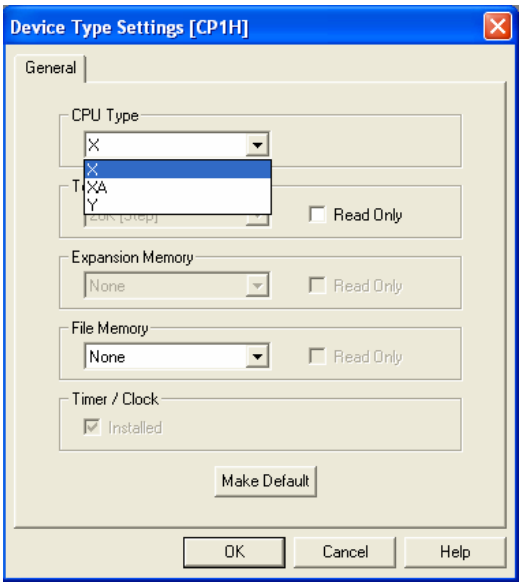
3. คลิก [Settings]

ไดอะลอกบล๊อก “Device Type Settings” จะแสดงดังต่อไปนี้



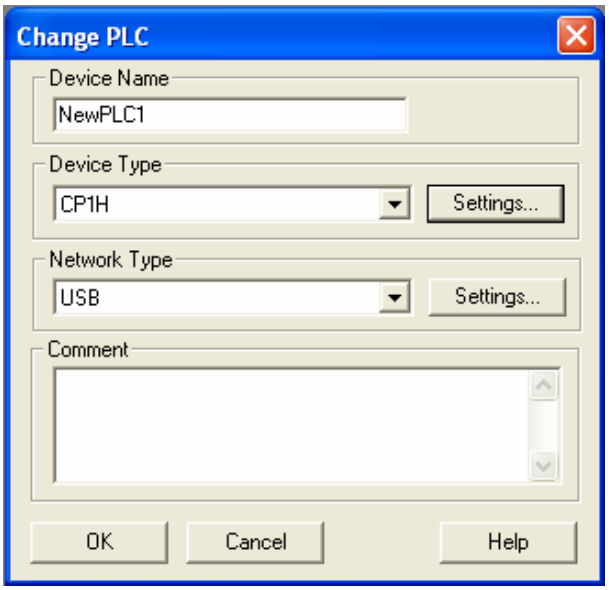
4. เลือกรุ่น CPU จาก CPU Type เป็น ‘X’ จากนั้นคลิก [OK]

ไดอะลอกบล๊อก “Device Type Settings” จะปิดลง

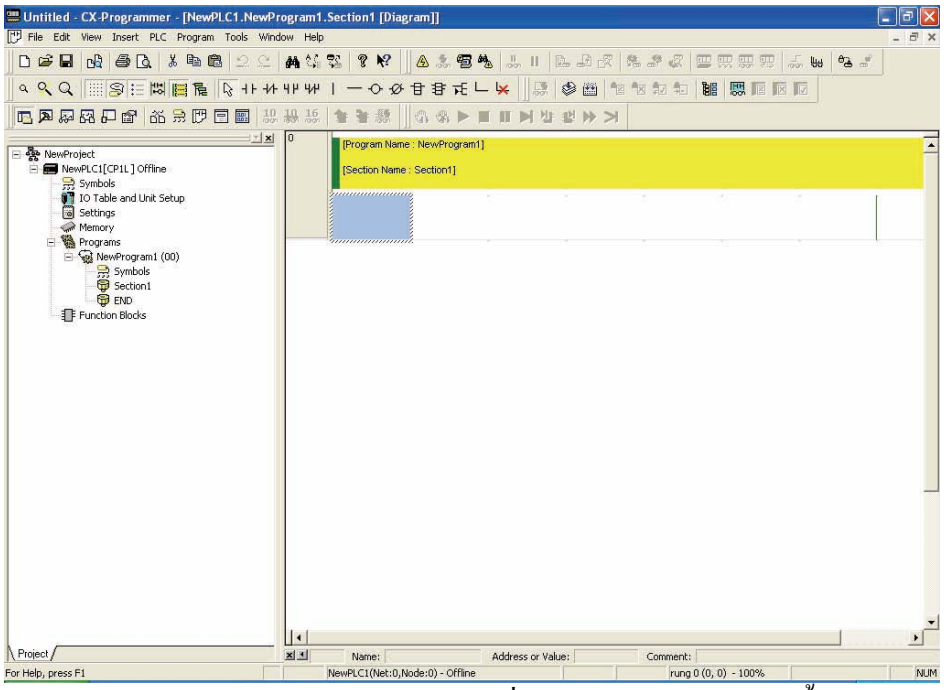


5. ตรวจสอบด้วยว่า Network Type เป็น [USB] จากนั้นคลิก [OK]

ถ้า PLC ไม่ใช่รุ่น CP1H การเลือก Network Type ต้องขึ้นอยู่กับ PLC รุ่นนั้นๆ เช่น Toolbus และ Hostlink

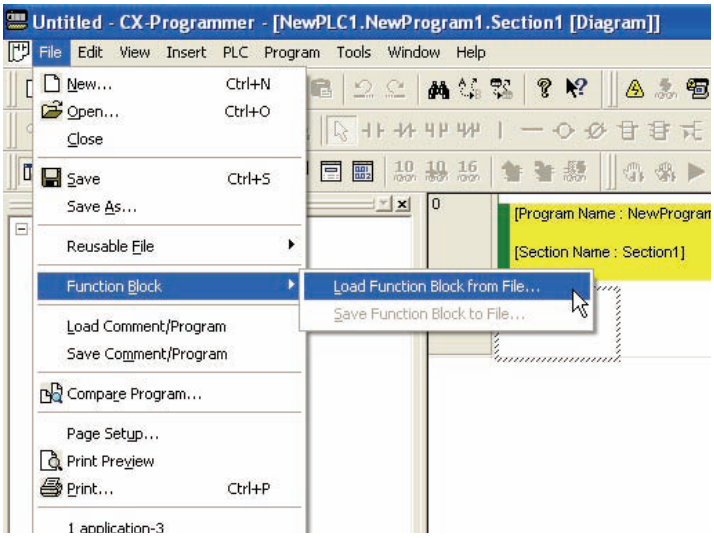


เมื่อคลิก [OK] แล้ว ไดอะลอกบล๊อก “Change PLC” จะปิดลงและแสดงหน้าต่าง Main window ดังรูปข้างล่างนี้

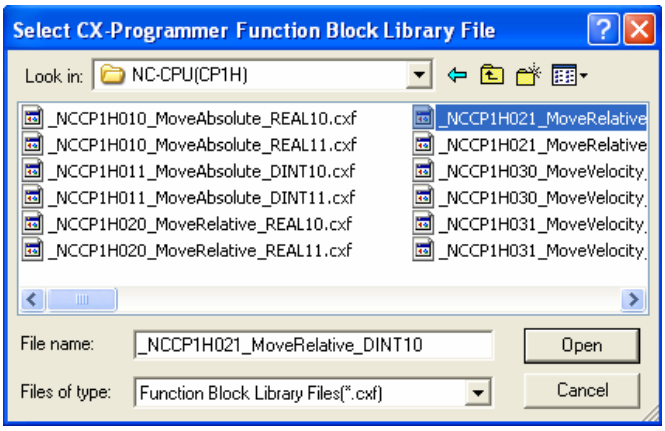


<u>หมายเหตุ</u> ถ้าไม่สามารถเลือก[USB] ที่ NetworkType ได้ ให้ติดตั้ง USB Driver ของ CP1H

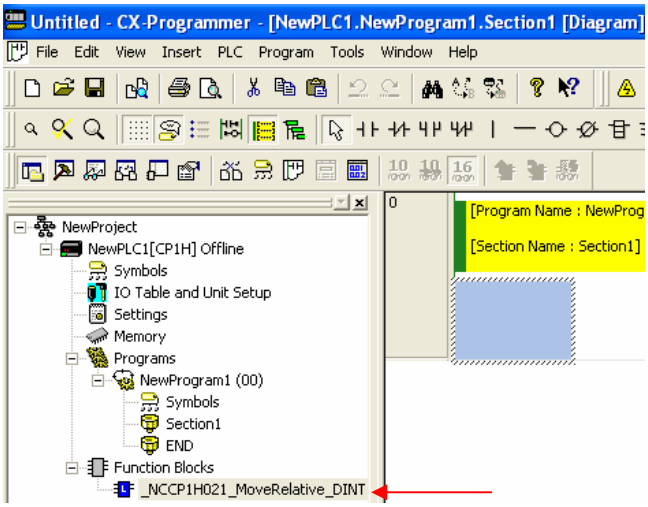
6. เลือก [File] - [Function Block] - [Load Function Block from File] จากเมนูไดอะลอกบล๊อก ‘Select CX-Programmer Function Block Library File’ จะปรากฏขึ้น



7. เลือก folder [FBL] - [omronlib] - [PositionController] - [NC-CPU(CP1H)] รายการของ FB library สำหรับควบคุมเซอร์โวมอเตอร์จะปรากฏขึ้น

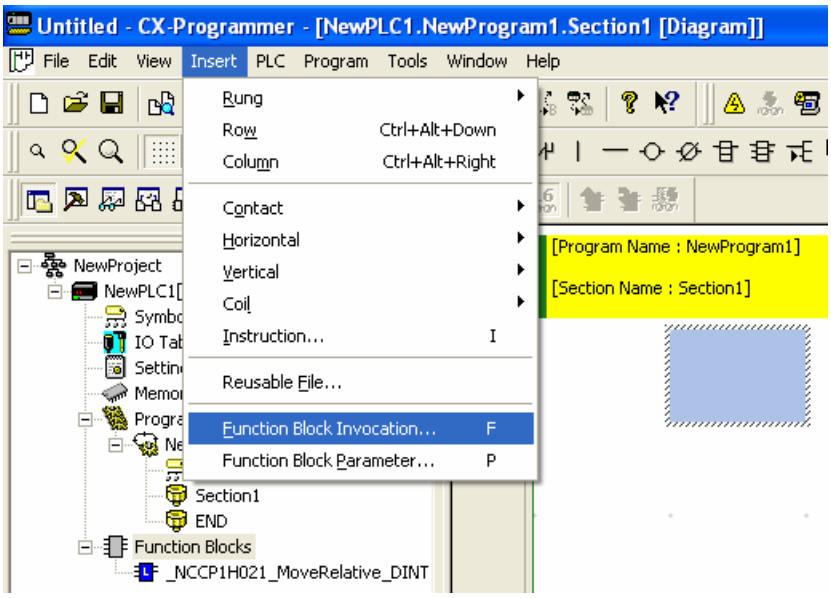


8. เลือก [_NCCP1H021_MoveRelative_DINT10.cxf] จากนั้นคลิก [Open] _NCCP1H021_MoveRelative_DINT10 จะถูกเพิ่มเข้ามาที่ project tree ในหัวข้อ [Function Blocks]

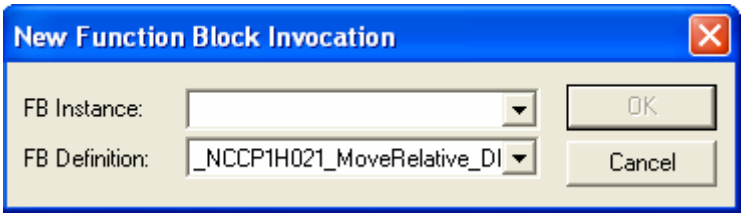


9. เลื่อน Cursor ในตำแหน่งที่ต้องการวาง FB (กรุณาอย่าวางติด Busbar) ‘_NCCP1H021_MoveRelative_DINT10’

10. เลือก [Insert] - [Function Block Invocation] จากเมนู

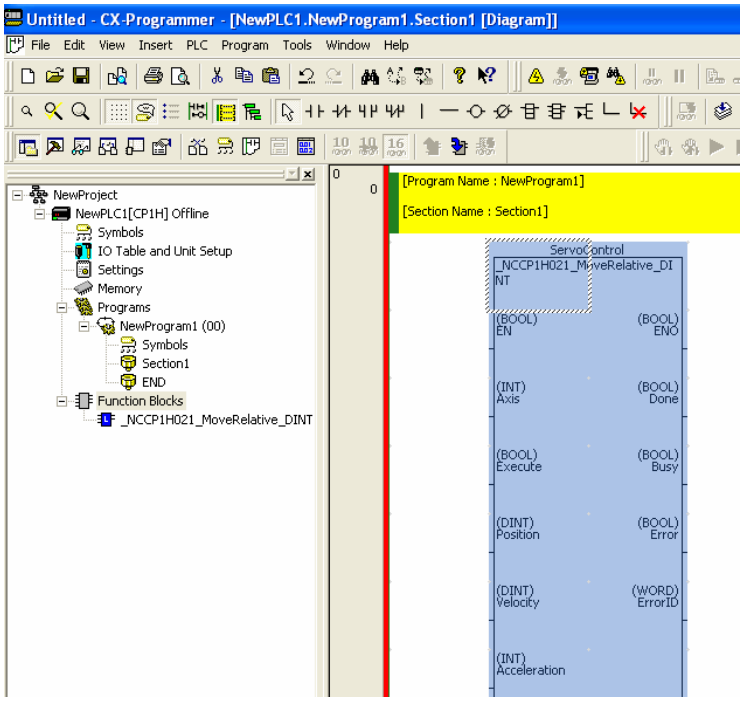


ไดอะลอกบล๊อก ‘New Function Block Invocation’ จะปรากฏขึ้น



11. ป้อนชื่อของ FB โดยพิมพ์ ‘ServoControl’ จากนั้นกด [Enter]

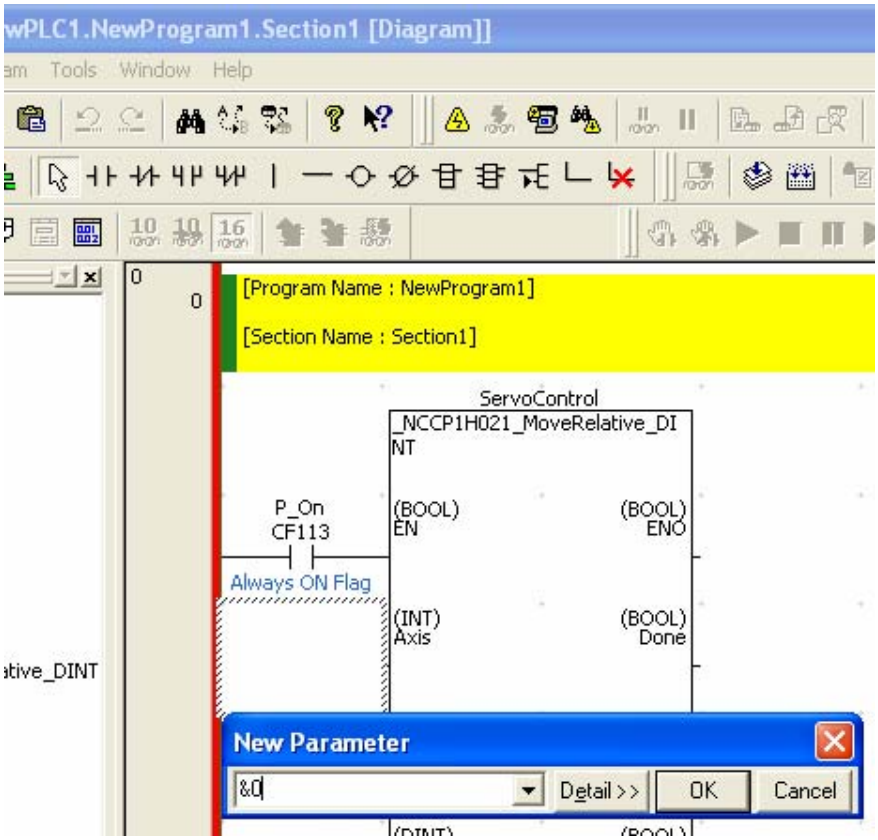
การพิมพ์ชื่อห้ามเว้นวรรค จากนั้นชื่อและ FB จะปรากฏขึ้น



12. ให้ลากเส้นอินพุตคอนแทค ‘P_On’ เข้ากับ FB

13. ทำการตั้งค่า I/O parameters ของ FB

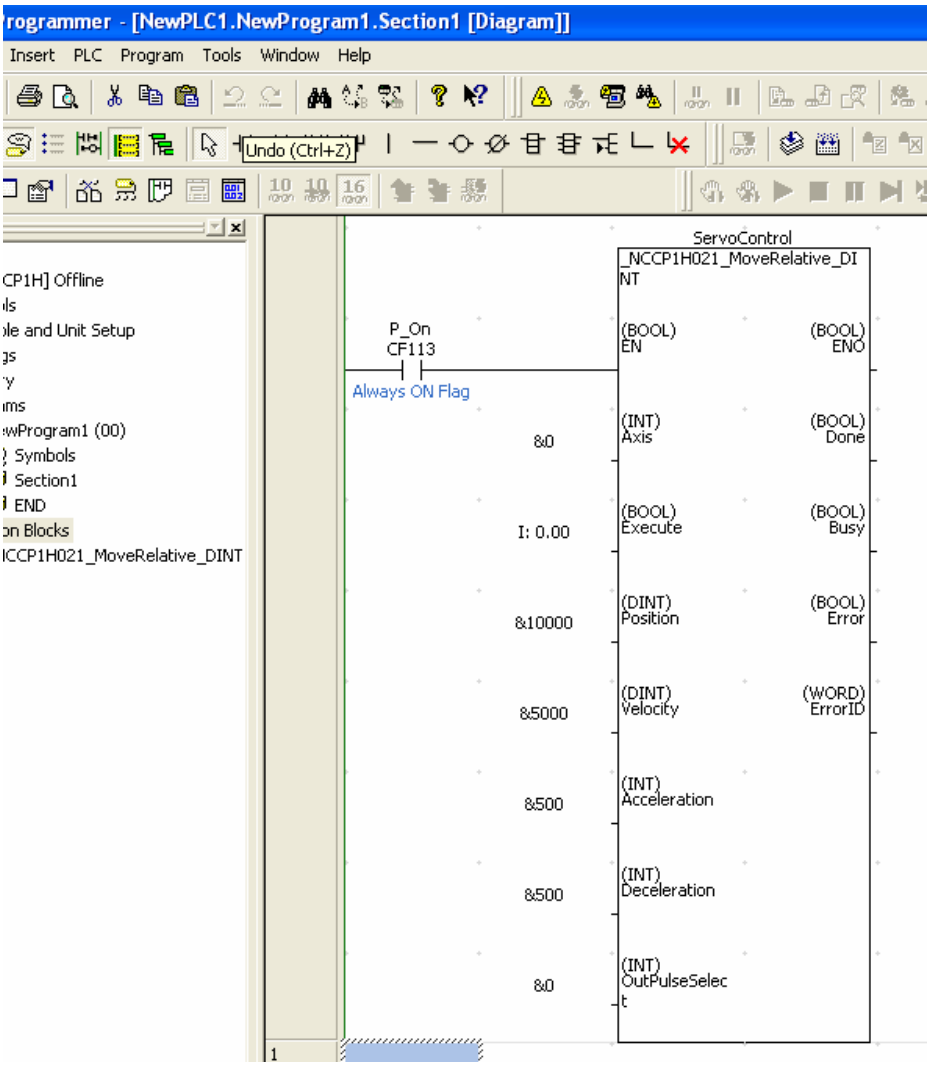
1) วาง cursor ที่ตำแหน่งพารามิเตอร์ของ FB ที่ต้องการจะตั้งค่าจากนั้นกด [Enter] ไดอะลอกบล๊อก ‘New Parameter’ จะปรากฏขึ้น
2) ป้อนค่าพารามิเตอร์จากนั้นกด [Enter]



ค่าพารามิเตอร์แต่ละตัวที่ต้องป้อนมีค่าดังนี้

Axis	= &0 (ให้พิมพ์ “&0” ที่ไดอะลอกบล๊อก ‘New Parameter’)
Execute	= 0.00 (Start switch)
Position	= &10000 (ระยะการหมุน 10,000 พลัส)
Velocity	= &5000 (ความเร็วในการหมุน 5,000 Hz)
Acceleration	= &500 (อัตราเร่ง 500 Hz/4ms)
Deceleration	= &500 (อัตราหน่วง 500 Hz/4ms)
OutPulseSelect	= &0 (เลือกโหมด CW และ CCW)
หมายเหตุ	1) ‘&’ หมายถึงเป็นเลขจำนวนเต็ม (INT)
	2) สามารถดูความหมายของพารามิเตอร์ได้ในภาคผนวก E

หลังจากป้อนพารามิเตอร์ โปรแกรมแลดเดอร์ที่เขียนเสร็จแล้ว จะแสดงดังรูปข้างล่างนี้



14. จากนั้นทำการ Online => Transfer => Run โปรแกรมตามขั้นตอนที่กล่าวในบทที่ 6 และเลือกโหมด PLC ไว้ที่ Monitor

15. ทำการทดสอบการทำงานโดยเปิดสวิตช์ ‘Servo Driver Run’ เพื่อให้ Driver จ่ายไฟเข้ามอเตอร์จากนั้นให้ Turn-On สวิทช์ ‘Start (0.00)’ เพื่อสั่งให้มอเตอร์เริ่มหมุน ในกรณีนี้เราจ่ายพลัสให้กับไดว์เวอร์ 10,000 พลัสจะทำให้มอเตอร์หมุน 10 รอบ เพราะเราตั้ง Resolution = 1,000 พลัสต่อรอบ

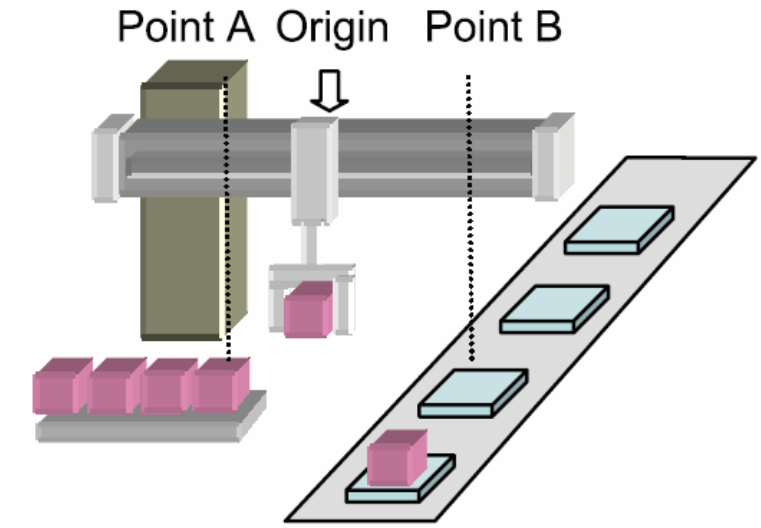
ตัวอย่างที่ 2 ระบบควบคุมการเคลื่อนที่แบบ Absolute

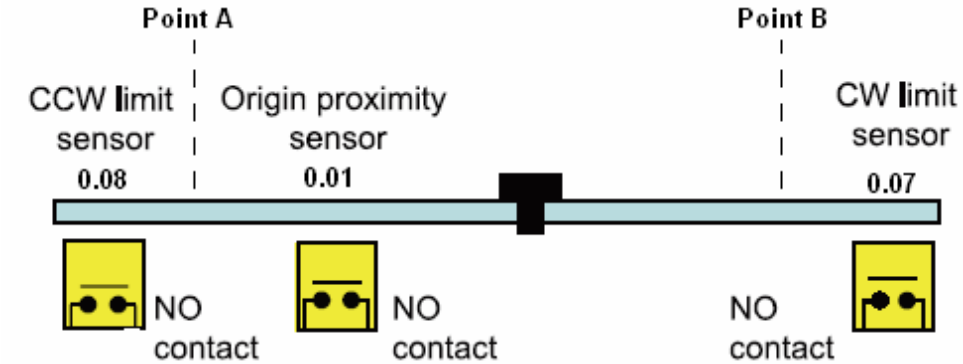
การเคลื่อนที่แบบ Absolute เป็นระบบที่ซับซ้อนและมีอุปกรณ์เกี่ยวข้องมากกว่า การหมุนของมอเตอร์จะอ้างอิงจากจุดศูนย์ (Origin) ดังนั้นจะต้องทำ Origin search เมื่อเริ่มเปิดเครื่องเพื่อหาจุดศูนย์ทุกครั้ง ระบบนี้เหมาะกับงานที่มีระยะการเคลื่อนที่แน่นอนกว่าเพราะมีจุดอ้างอิงในการเคลื่อนที่ ต่อไปนี้เป็นตัวอย่างแสดงการออกแบบและเขียนโปรแกรมควบคุมเซอร์โวมอเตอร์แบบ Absolute

ในตัวอย่างนี้เราจะใช้ CP1H ทำหน้าที่สั่งงานและเขียนโปรแกรมด้วย Function Block ซึ่งจะช่วยให้ผู้เริ่มต้นใช้ง่ายระบบควบคุมเซอร์โวมอเตอร์เข้าใจได้ง่ายกว่า

รูปแบบการทำงาน

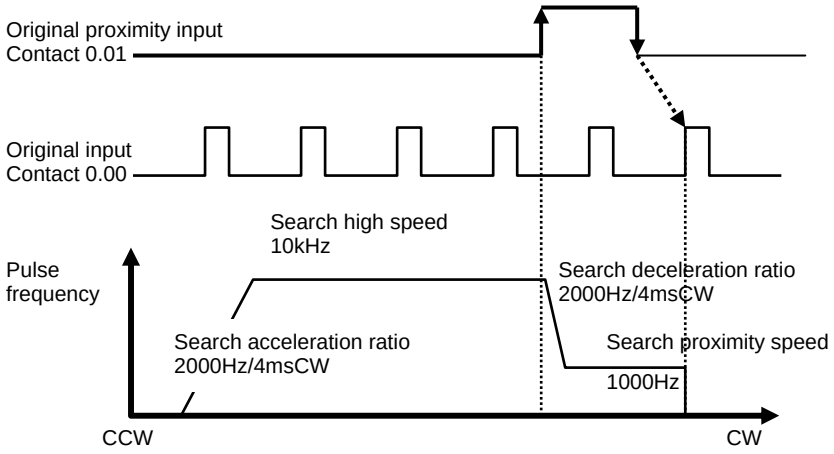
จากรูปข้างล่างนี้เครื่องจักรจะมีแกนเคลื่อนที่ถูกขับด้วย Ball screw ทำหน้าที่ในการเคลื่อนย้ายสินค้าระหว่างจุด A และ จุด B และมีจุด Origin อยู่ใกล้กับจุด A นอกจากนั้นยังมี CW Limit Sensor และ CCW Limit Sensor ทำหน้าที่ตรวจสอบการเคลื่อนที่เกินกว่าระยะที่กำหนด





- **การหาจุดศูนย์ (Origin search)**

การทำ Origin search จะต้องใช้สัญญาณอินพุตเอาต์พุตหลายจุด เช่น ‘Origin proximity input’, ‘Origin input’, ‘Positioning completed’ และ ‘Error counter reset’ การทำ Origin search สามารถทำได้ด้วยคำสั่งเพียงคำสั่งเดียว เมื่อสั่งให้ระบบทำ Origin Search เซอร์โวมอเตอร์จะหมุนเพื่อค้นหาสัญญาณ “Origin proximity input” เมื่อพบสัญญาณนี้แล้ว มอเตอร์ยังคงหมุนอยู่จนกระทั่งสัญญาณ Z-Phase ของเอ็นโค้ดเดอร์จะส่งจาก Driver มาให้ PLC จากนั้นมอเตอร์จะหยุดหมุน แสดงว่าการทำ Origin Search เสร็จสมบูรณ์ดังแสดงในกราฟข้างล่างนี้ นอกจากนั้นเรายังเลือกโหมดการหา Origin ได้ดังตารางข้างล่างนี้ สัญญาณ Z-Phase จาก Servo Driver นี้ เราเรียกว่าสัญญาณ Origin input

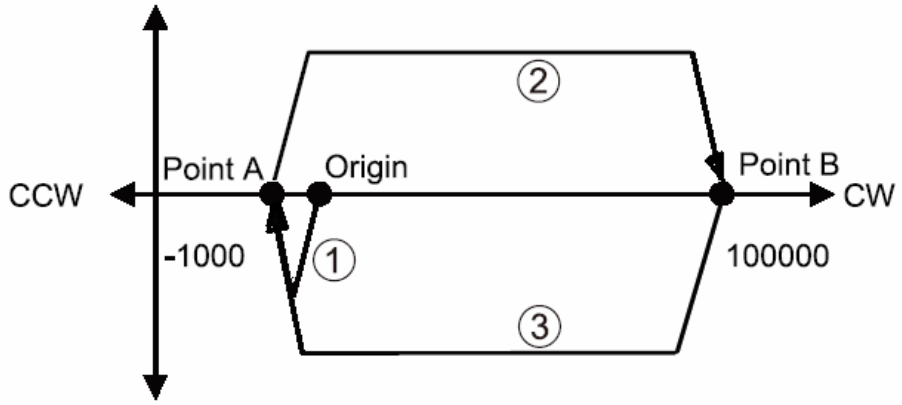


Origin Search Method	Setting	คำอธิบาย
Search direction	CW	Origin search จะทำงานในทิศทาง CW
Detection method	Methd 0	ตรวจจับสัญญาณอินพุต Origin ที่เกิดครั้งแรก หลังจากได้รับสัญญาณจาก Origin proximity switch เป็น OFF-ON-OFF ตามลำดับ
Search operation	Invers 1	เคลื่อนที่ในทิศทางตรงข้ามเพื่อหา Origin
Operating mode	Mode 1	ส่งสัญญาณออกไปที่เอาต์พุต Counter reset เมื่อการตรวจจับสัญญาณ Origin ได้แล้ว

- **การเคลื่อนที่ตามตำแหน่ง (Positioning)**

การตั้งค่าการเคลื่อนที่ไปตามตำแหน่งต่างๆ มีรายละเอียดดังนี้ และแสดงรูปแบบการเคลื่อนที่ได้ดังรูปข้างล่างนี้

- Target frequency = 50 KHz
- Acceleration/Deceleration = 2000Hz/4ms
- Initial frequency = 0Hz



(1) เมื่อกดสวิตช์ ‘Start origin search’ CP1H จะสั่งให้เซอร์โวมอเตอร์หมุนเพื่อหาตำแหน่ง Origin หลังจากทำ Origin search เสร็จ CP1H จะส่งสัญญาณออกที่เอาต์พุต ‘Origin search complete’ หลังจากนั้นมันจะเคลื่อนที่ไปที่จุด A (-1000)

(2) เมื่อมีชิ้นงานลำเรียงมาอยู่ที่ตำแหน่ง A เซนเซอร์ชื่อ ‘Positioning to point B’ ส่งสัญญาณให้ CP1H และสั่งให้เซอร์โวมอเตอร์หมุนเคลื่อนที่ไปยังตำแหน่ง B เพื่อวางชิ้นงานบนสายพานลำเลียง

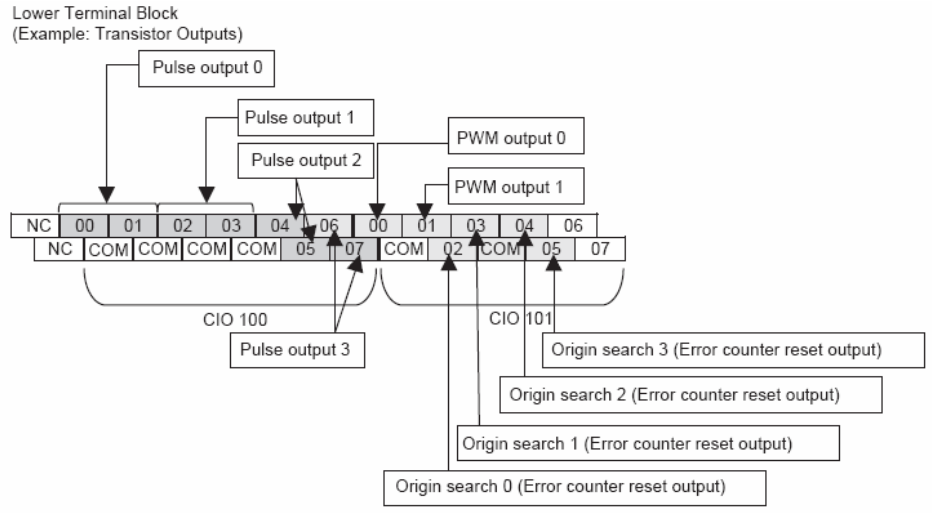
(3) เมื่อวางชิ้นงานบนสายพานที่จุด B เซนเซอร์ชื่อ ‘Positioning to point A’ ส่งสัญญาณให้ CP1H และสั่งให้เซอร์โวมอเตอร์หมุนเคลื่อนที่ไปยังตำแหน่ง A และจะทำงานซ้ำเช่นนี้ไปเรื่อยๆ จนกว่าจะไม่มีชิ้นงานที่จุด A

การจัดองค์ประกอบของระบบ (Configuration)

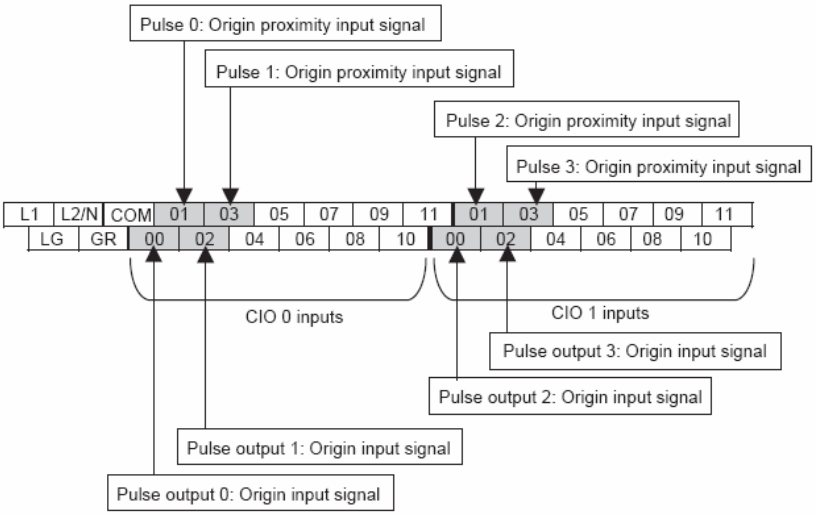
- **ตัวอย่างการเดินสายไฟ (ยกตัวอย่างรุ่น CP1H-X40DT-D)**

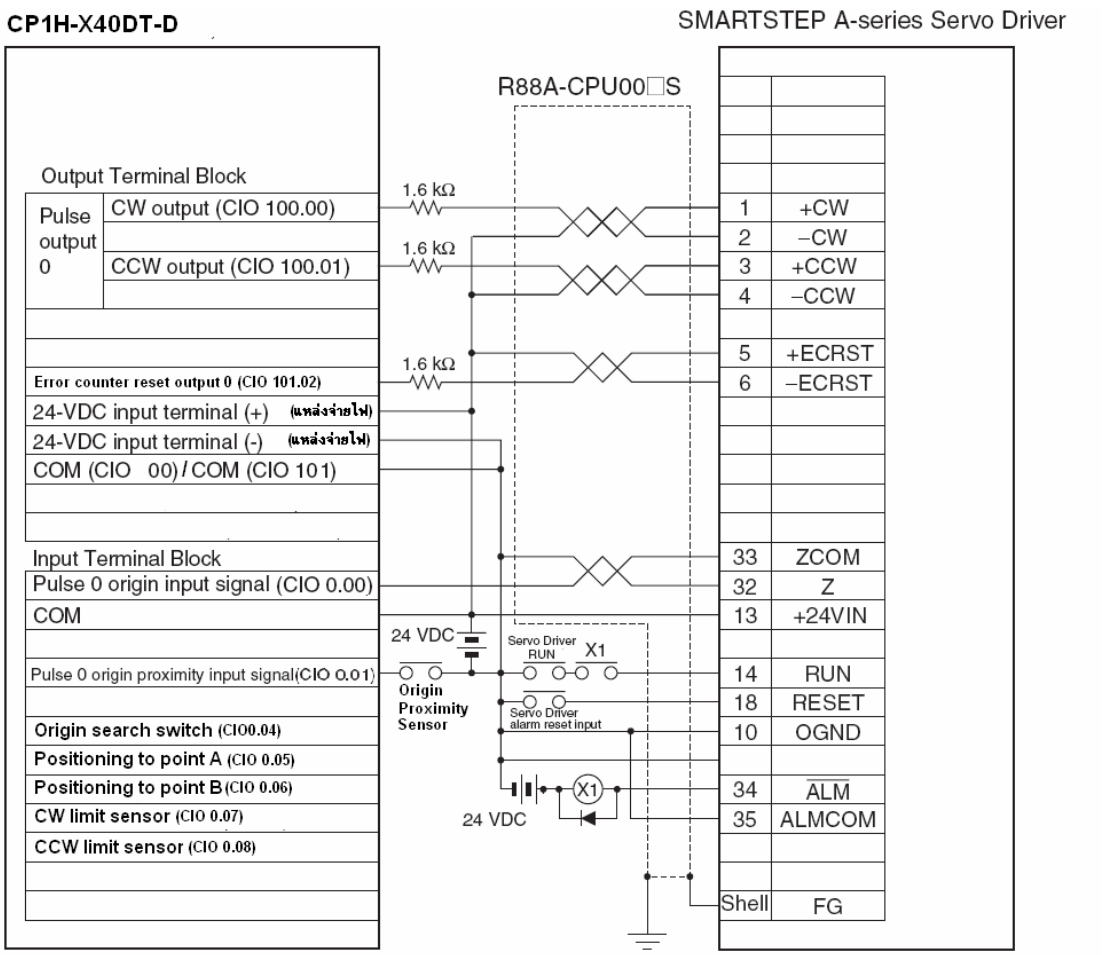
เนื่องจาก PLC รุ่นนี้ใช้ควบคุมเซอร์โวมอเตอร์ได้ 4 ตัว ถ้าใช้เพียงตัวเดียว I/O อื่นๆ สามารถใช้เป็น I/O ปกติได้

Output Terminal



Input Terminal





หมายเหตุ

1. ควรใส่ตัวต้านทาน 1.6-2.2 KΩ ที่เอาต์พุตพลัสของ PLC เพื่อป้องกันไม่ให้กระแสเกิน พิกัดกรณีที่ใช้ไฟ 24VDC
2. ความหมายของสัญญาณที่เทอร์มินอลของเซอร์โวไดร์เวอร์

   +CW/-CW (Input) : เป็นสัญญาณพลัสสั่งให้มอเตอร์หมุนตามเข็มนาฬิกา

   +CCW/-CCW (Input): เป็นสัญญาณพลัสสั่งให้มอเตอร์หมุนทวนเข็มนาฬิกา

   +ECRST/-ECRST(Input) : เป็นสัญญาณสั่งรีเซ็ทเคาท์เตอร์ในไดร์ดเวอร์

   Z/ZCOM(Output): เป็นสัญญาณแจ้งให้ PLC รู้ว่าเอ็นโค้ดเดอร์อยู่ที่จุดศูนย์

   RUN(Input) : เป็นสัญญาณสั่งให้ไดร์เวอร์จ่ายไฟเข้ามอเตอร์พร้อมทำงาน

   ALM/ALMCOM(Output): เป็นสัญญาณแจ้งให้รู้ว่าระบบเซอร์โวมีปัญหา
3. ความหมายของสัญญาณที่เทอร์มินอลของ CP1H

   Origin search switch(Input) : เป็นสวิตช์ใช้สตาร์ท Origin search

   Positioning to point A/B(input) : เซนเซอร์สั่งให้เคลื่อนที่ไปตำแหน่ง A และB

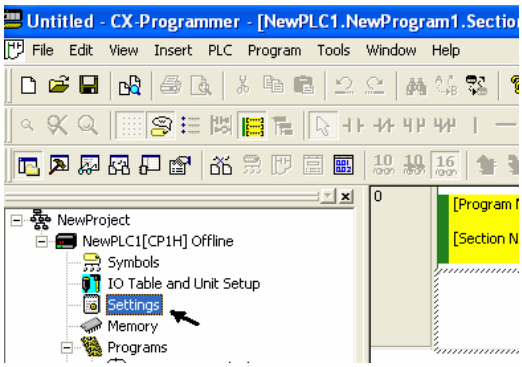
   CW/CCW limit sensor(Input) : เป็นเซนเซอร์ป้องกันไม่ให้วิ่งเกินระยะ

Origin proximity input : สัญญาณอินพุตที่เป็นตัวบอก PLC ว่าการเคลื่อนที่เข้าใกล้จุดศูนย์

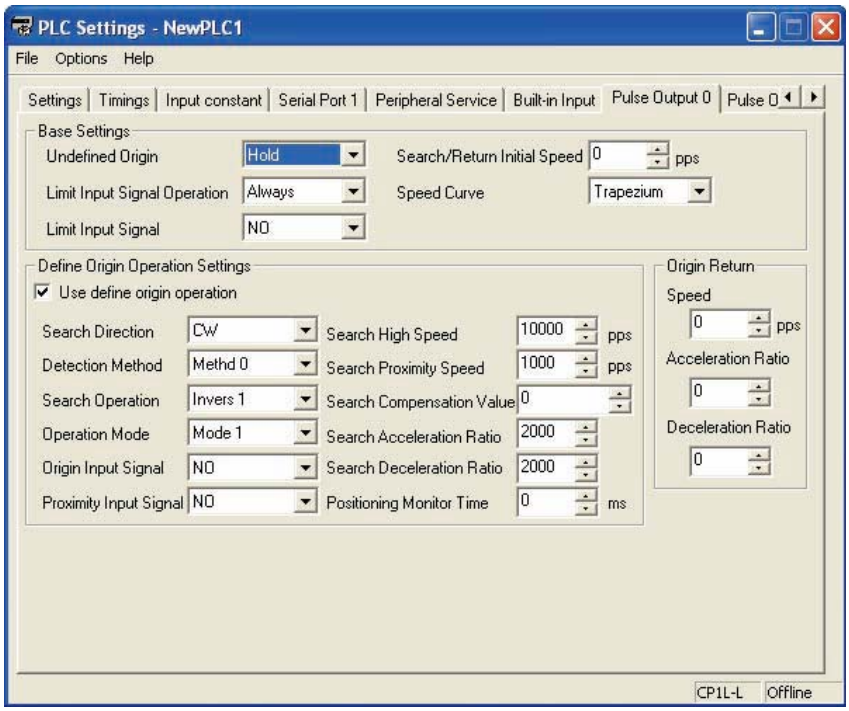
Origin input : เมื่อ PLC ได้รับสัญญาณ Origin proximity input แล้วมันจะตรวจสอบสัญญาณ Origin (Z-phase) ของเอ็นโค้ดเดอร์ที่อยู่บนตัวเซอร์โวมอเตอร์

- **การตั้งค่า Servo Driver setup ทำตามขั้นตอนในตัวอย่างที่ 8.1**
  - **PLC**

1. เปิดไดอะลอกบล๊อก 'PLC Setting' โดยการคลิกที่ Setting



2. คลิกที่ Tap 'Pulse Output 0' แล้วตั้งค่าตามนี้



Base Setting

Item	Setting
Undefined Origin	Hold
Limit input Signal Operation	Always
Limit input Signal	NO
Search/Return Initial Speed	Opps
Speed Curve	Trapezoidal

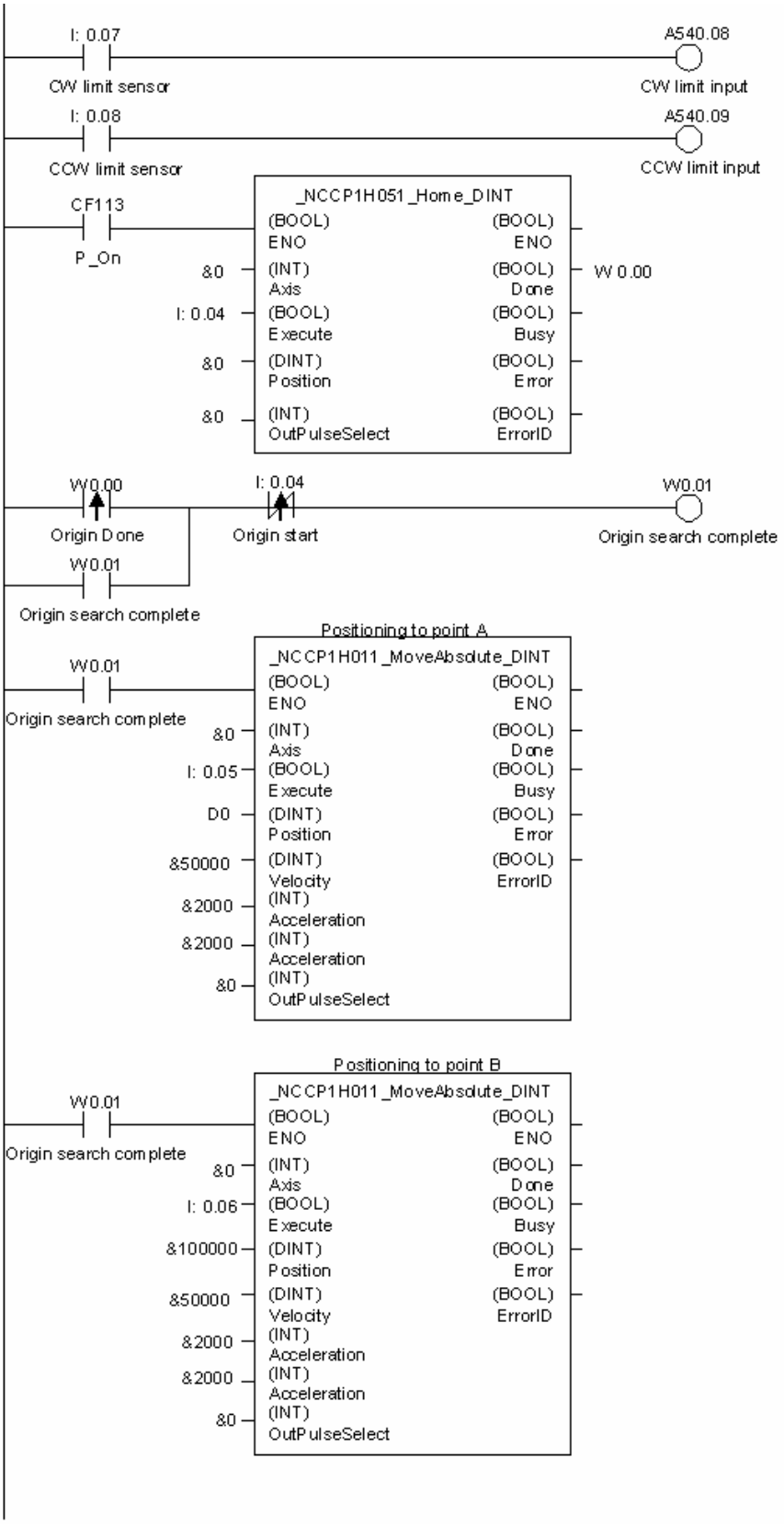
Origin Search

Item	Setting
Use define origin operation	Use
Search Direction	CW
Detection Method	Methd 0
Search Operation	Invers 1
Operating Mode	Mode 1
Origin input Signal	NO
Proximity Input Signal	NO
Search High Speed	10000pps
Search Proximity Speed	1000pps
Search Compensation Value	0
Search Acceleration Ration	2000
Search Deceleration Ration	2000
Positioning Monitor Time	0ms

3. ปิดไดอะลอกบล๊อก ‘PLC Setting’
4. เพื่อให้ค่าใน ‘PLC Setting’ มีผลให้ปิดไฟแล้วเปิดใหม่

การเขียนโปรแกรม

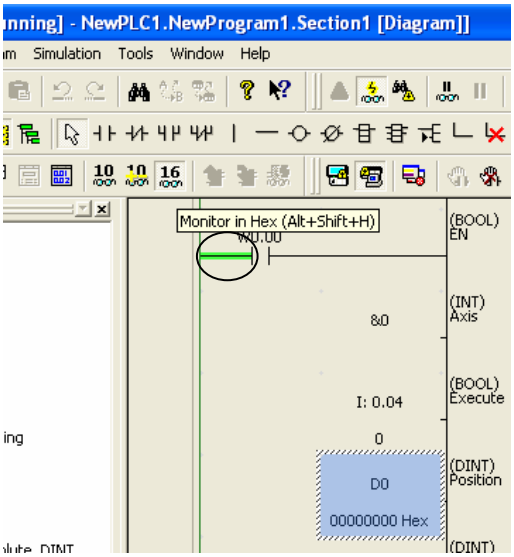
- **ตัวอย่างโปรแกรมแลดเดอร์**

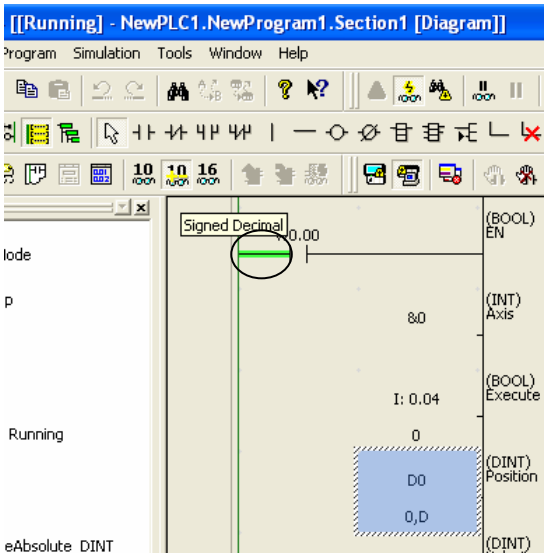


จากตัวอย่างโปรแกรมข้างต้นเป็นตัวอย่างในการทำงานอย่างง่ายๆ เท่านั้น ในการใช้งานจริงจำเป็นต้องเขียนให้ละเอียดกว่านี้ ส่วนวิธีการสร้างโปรแกรมสามารถทำตามขั้นตอนต่างๆ ที่ได้กล่าวไปแล้ว Function Block(FB) ที่ต้องเรียกใช้งานมีอยู่ด้วยกัน 2 ตัว คือ

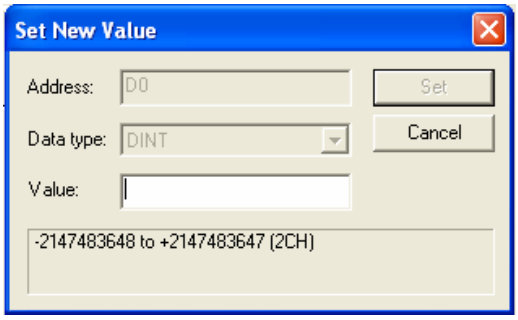
- **'_NCCP1H051_Home_DINT'** ทำหน้าที่เป็น Origin search โดยใช้ I:0.04 เป็นตัวสตาร์ท
- **'_NCCP1H011_MoveAbsolute_DINT'** ทำหน้าที่สั่งให้มอเตอร์เคลื่อนที่แบบ Absolute โดยใช้ทั้งหมด 2 ตัว ตัวที่หนึ่งสั่งให้เคลื่อนที่ไปตำแหน่ง A ค่า Position คือ DM0 และต้องป้อนค่า DM0 เท่ากับ -1000 เพราะว่า FB ไม่สามารถป้อนเลขลบได้โดยตรง ส่วนตัวที่สองสั่งให้เคลื่อนที่ไปตำแหน่ง B โดยมีค่า Position เท่ากับ 100000

  การใส่ค่าใน D0 จะต้องทำขณะ Online ถ้าการแสดงผลเป็นเลขฐาน 16 ให้เปลี่ยนเป็นเลขฐานสิบแบบมีเครื่องหมาย (Signed decimal) โดยการคลิกที่ icon ดังรูป





จากนั้นให้คลิกที่ D0 จะปรากฏไดอลอกบล๊อก 'Set New Value' ขึ้นมา จากนั้นให้ป้อน -1000 แล้วคลิก **Set** จากนั้นค่า D0 จะเป็น -1000



ทดสอบการทำงาน

- ทำการ Online->Tranfer->Run โปรแกรมตามขั้นตอนที่กล่าวในบทที่ 6 และเลือกโหมด PLC ไว้ที่ Monitor
- ทำการทดสอบการทำงานโดยเปิดสวิตช์ Origin Search (I0.04) เมื่อทำ Origin Search เสร็จให้เปิดสวิตช์ Position to Point A เพื่อให้ระบบเคลื่อนที่ไปยังตำแหน่ง A จากนั้นทดลองเปิดสวิตช์ Position to Point B เพื่อให้ระบบเคลื่อนที่ไปยังตำแหน่ง B

จากตัวอย่างทั้ง 2 แบบที่กล่าวมาข้างต้น จะช่วยให้ท่านเข้าใจถึงเทคนิคในการควบคุมเซอร์โวมอเตอร์โดยใช้ฟังก์ชันบล๊อก ซึ่งช่วยให้เรียนรู้ได้เร็วและง่าย นอกจากนั้นเราอาจสรุปได้ว่าการควบคุมตำแหน่งอาจแบ่งได้เป็น 2 ประเภทใหญ่ๆ คือ แบบIncremental กับแบบ Absolute ซึ่งแบบ Incremental การเคลื่อนที่จะอ้างอิงกับจุดปัจจุบัน ส่วนแบบ Absolute อ้างอิงกับจุดศูนย์ ดังนั้นจะต้องทำ Origin Search เสมอเมื่อเริ่มใช้งาน