I AM BEGINNER

เรียนรู้การใช้งานพีแอลซีเบื้องต้น

Allen-Bradley Micro800

SONIC AUTOMATION
A Sonepar Company

คำนำ

หนังสือเล่มนี้จัดทำขึ้นเพื่อให้ความรู้เกี่ยวกับการใช้งานพีแอลซีเบื้องต้นเหมาะสำหรับผู้เริ่มต้นเรียนรู้การใช้งาน PLC สำหรับระบบควบคุมอัตโนมัติ โดยมีเนื้อหาครอบคลุมจากพื้นฐานตั้งแต่อุปกรณ์อินพุตและเอาท์พุตจนถึงวิธีการเขียนโปแกรม PLC โดยอ้างอิงตามมาตรฐาน IEC 61131-3 ซึ่งเป็นรูปแบบสากลและเป็นที่นิยมใช้กันอย่างกว้างขวาง การเขียนโปรแกรมจะเน้นที่แลดเดอร์ไดอะแกรมเป็นหลักพร้อมตัวอย่างประกอบการเขียนโปรแกรมเบื้องต้น

ทางบริษัทฯ หวังเป็นอย่างยิ่งว่าหนังสือเล่มนี้คงเป็นประโยชน์ต่อการสร้างพื้นฐานในการเรียนรู้และเรายินดีรับข้อเสนอแนะ เพื่อจะได้นำมาปรับปรุงให้ดียิ่งขึ้น

บริษัท โซนิค ออโตเมชั่น จำกัด

พ.ศ. 2562

บริษัท โซนิค ออโตเมชั่น จำกัด โทรศัพท์ 0-2835-3933 โทรสาร 0-2835-3935

เว็บไซต์ www.sonicautomation.co.th

สารบัญ

Chapter

1

ความรู้เบื้องต้นเกี่ยวกับพีแอลซี

ก่อนที่จะเรียนรู้การใช้งาน PLC สิ่งหนึ่งที่ไม่ควรข้ามไปก็คือประวัติความเป็นมาของ PLC ซึ่งในบทนี้จะกล่าวถึงต้นกำเนิดของมันและทำไมอุปกรณ์นี้ถึงได้ถูกพัฒนาขึ้นมา นอกจากนั้นยังเป็นที่นิยมใช้งานกันอยู่ในทุกวันนี้

1.1 กำเนิด PLC

PLC (Programmable Logic Controller) ได้พลิกหน้าประวัติศาสตร์อุตสาหกรรมอัตโนมัติ ทุกวันนี้ PLC ถูกใช้งานอย่างกว้างขวางในภาคอุตสาหกรรมการผลิต รวมถึงระบบโครงสร้างพื้นฐานต่างๆ เช่น การผลิตน้ำประปาและผลิตไฟฟ้า เป็นต้น

ก่อนหน้าปี 1968 อุปกรณ์ควบคุมเครื่องจักรส่วนใหญ่ยังใช้ relay, cam timers และ drum sequencers เป็นอุปกรณ์หลักในการทำวงจรควบคุมเครื่องจักร PLC เริ่มพัฒนาขึ้นในปี 1968 โดยที่ GM Hydramatic (แผนกขนถ่ายอัตโนมัติของ General Motors) ต้องการเปลี่ยนระบบควบคุมด้วยรีเลย์ให้เป็นระบบอิเล็กทรอนิกส์ ในขณะนั้น Bedford Associates ได้รับการคัดเลือกให้เป็นผู้สร้าง PLC และแล้ว PLC ตัวแรกก็ได้กำเนิดขึ้นโดยใช้ชื่อว่า 084 เพราะมันเป็นโครงการที่ 84 ของ Bedford Associates และ PLC เริ่มใช้งานจริงจังในปี 1969 เป็นต้นมา จากนั้น Bedford Associates ได้เริ่มก่อตั้ง บริษัทใหม่ขึ้นเพื่อการพัฒนา, ผลิต, ขายและให้บริการ ผลิตภัณฑ์ใหม่นี้ใช้ชื่อว่า Modicon ซึ่งย่อมาจาก MOdular DIdital CONtroller

1.2 ประวัติของ Allen Bradley

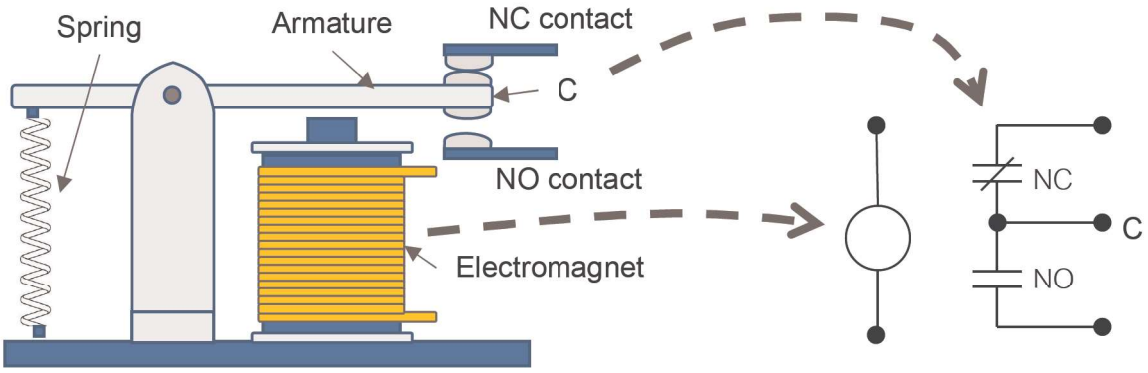
Allen Bradley ก่อตั้งขึ้นโดย Dr. Stanton Allen และ Lynde Bradley ในปี 1903 โดยใช้ชื่อว่า Compression Rheostat Company ต่อมาในปี 1910 ได้เปลี่ยนชื่อใหม่เป็น Allen-Bradley Company เริ่มต้นธุรกิจจากผลิตตัวต้านทาน(Resistor) เพื่อใช้ในอุปกรณ์อีเลคทรอนิคส์และอุปกรณ์อื่นๆ ต่อมาปี 1985 บริษัท Rockwell International ได้เข้าซื้อกิจการของ Allen Bradley

รูป 1.1 แสดง PLC รุ่น Bulletin 1774

ในปี 1974 ได้จดสิทธิบัตรและเปิดตัว PLC รุ่นแรกชื่อ Bulletin 1774 และได้มีการพัฒนา PLC รุ่นต่างๆ ออกมาเป็นลำดับ เช่น PLC2 PLC3 และ PLC5 เป็นต้น จนในปัจจุบันมีรุ่นที่นิยมใช้งาน คือ CompactLogix และ ControlLogix ที่พัฒนาขึ้นสอดคล้องตามมาตรฐานการเขียนโปรแกรม IEC 61131 ซึ่งจะกล่าวถึงวิธีการเขียนโปรแกรมตามมาตรฐานนี้ในภายหลัง

1.3 วงจรแลดเดอร์

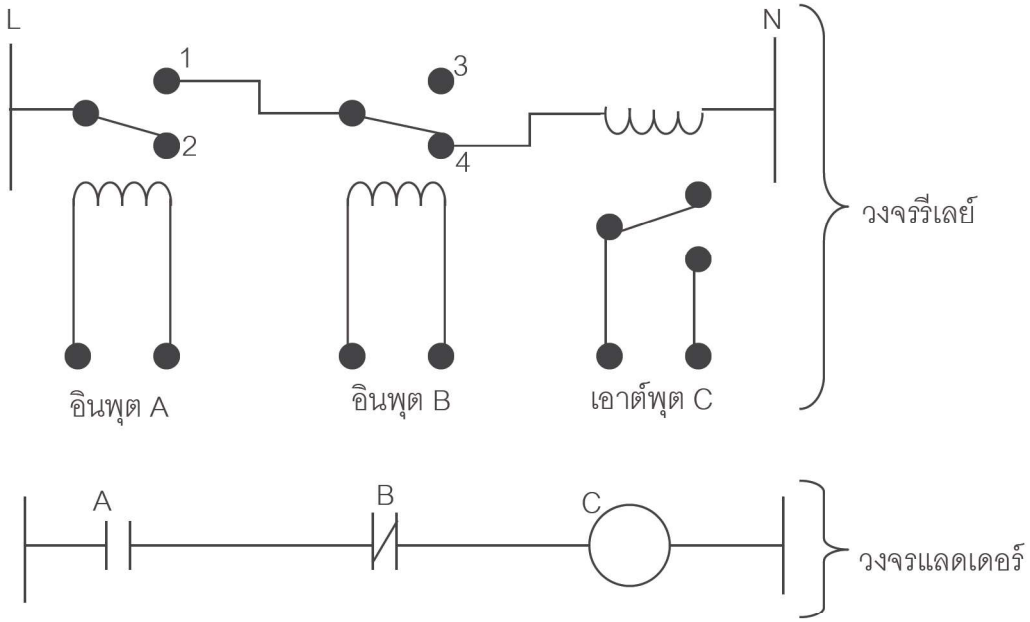
วงจรแลดเดอร์เป็นวิธีการสร้างโปรแกรมของ PLC ซึ่งพัฒนามาจากวงจรรีเลย์แบบดั้งเดิม รีเลย์เป็นอุปกรณ์ที่ใช้สนามแม่เหล็กเพื่อควบคุมการสับเปลี่ยนหรือตัดต่อวงจร ดังแสดงในรูปที่ 1.2 เมื่อจ่ายแรงดันไฟเข้าที่คอยล์กระแสไฟฟ้าที่ไหลผ่านคอยล์จะสร้างสนามแม่เหล็กขึ้นซึ่งมันจะดูดคานโลหะให้เคลื่อนที่และหน้าคอนแทคจะต่อกัน ถ้าหน้าคอนแทคต่อกันขณะจ่ายไฟเราเรียกว่า ปกติเปิด (Normally Open) ถ้าหน้าคอนแทคต่อกันขณะยังไม่จ่ายไฟเราเรียกว่า ปกติปิด (Normally Closed) รีเลย์มักจะวาดด้วยสัญลักษณ์วงกลมเพื่อแทนคอยล์ ส่วนหน้าคอนแทคปกติเปิด (NO) จะใช้เส้นขนาน 2 เส้น และหน้าคอนแทคปกติปิด (NC) จะใช้เส้นขนาน 2 เส้น พร้อมเส้นทะแยง 1 เส้น



รูป 1.2 แสดงโครงสร้างของรีเลย์และสัญลักษณ์ตามมาตรฐาน ANSI

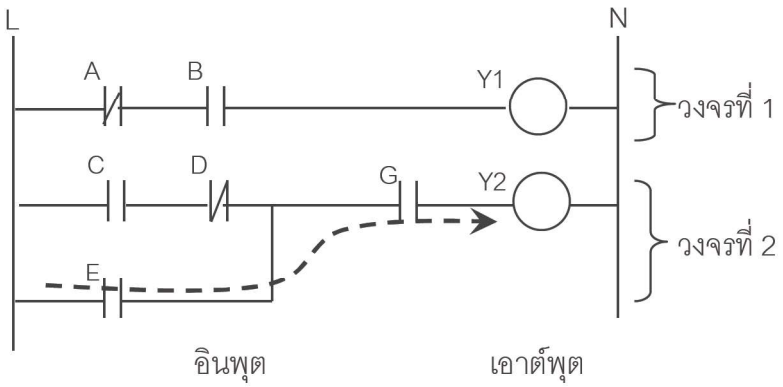
ในรูปที่ 1.3 เป็นตัวอย่างการใช้รีเลย์ในงานควบคุมอย่างง่าย ในวงจรนี้รีเลย์ตัวแรกด้านซ้ายมือใช้หน้าสัมผัส NO ซึ่งจะต่อโดยตรงเข้ากับแหล่งจ่ายไฟ กระแสไฟจาก L จะไหลมาค้างอยู่ที่หน้าคอนแทคนี้จนกว่าจะมีแรงดันไฟจ่ายที่คอยล์ A (Input A) ซึ่งจะทำให้หน้าคอนแทคนี้

เปลี่ยนไปยังตำแหน่ง 1 กระแสไฟจะไหลต่อไปยังหน้าคอนแทคของรีเลย์ตัวที่ 2 ได้(เป็นแบบ NC) ทำให้กระแสไฟไหลผ่านไปยังคอยล์ของรีเลย์ตัวที่ 3 ซึ่งทำให้หน้าคอนแทคเอาต์พุต C ทำงาน แต่ถ้ามีแรงดันไฟจ่ายที่คอยล์ B (Input B) จะทำให้ตัดกระแสไฟไปยังคอยล์ของรีเลย์ตัวที่ 3 วงจรนี้สามารถสร้างใหม่ให้เป็นรูปแบบของวงจรแลดเดอร์ได้ดังรูปที่ 1.3



รูป 1.3 วงจรควบคุมรีเลย์อย่างง่าย

คราวนี้เรามาลองแปลความหมายของวงจรแลดเดอร์ในรูปที่ 1.4 กันดู ให้จินตนาการว่าแหล่งจ่ายไฟคือเส้นแนวตั้งด้านซ้ายมือ เรียกว่า ไลน์(Line) ส่วนด้านขวา เรียกว่า นิวตรอน (Neutral) ในรูปจะมี 2 วงจร และแต่ละวงจรจะประกอบด้วยอินพุต(รูปเส้นตั้ง 2 เส้น) และเอาต์พุต (วงกลม) ถ้าอินพุตเปิดหรือปิดวงจร กระแสไฟจะสามารถไหลจาก L ผ่านอินพุตและจ่ายไฟให้กับเอาต์พุตและครบวงจรที่ N สัญญาณอินพุตอาจมาจาก สวิตช์ และเซ็นเซอร์ เป็นต้น เอาต์พุตจะเป็นอุปกรณ์ภายนอก PLC ที่ต้องการสั่งให้เปิดหรือปิด เช่น หลอดไฟ หรือ มอเตอร์



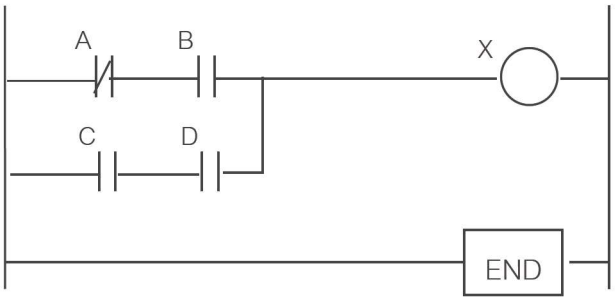
รูป 1.4 วงจรแลดเดอร์อย่างง่าย

วงจรที่ 1 มีหน้าคอนแทค A (NC) และ B (NO) ตามลำดับ ถ้าอินพุต A ไม่ทำงาน (OFF) และอินพุต B ทำงาน (ON) จะทำให้ไฟสามารถไหลผ่านหน้าคอนแทค A และหน้าคอนแทค B ไปที่คอยล์เอาต์พุต Y1 และสั่งให้อุปกรณ์ที่ต่อกับเอาต์พุตนั้นทำงาน แต่ถ้าเงื่อนไขเปลี่ยนเป็นอย่างอื่น เอาต์พุต Y1 จะไม่ทำงาน (OFF)

วงจรที่ 2 จะดูซับซ้อนกว่า ซึ่งจะมีอินพุตหลายตัวที่ต้องทำงานร่วมกันแล้วจึงทำให้กระแสไฟไหลไปยังคอยล์เอาต์พุต Y2 ได้ ที่ด้านซ้ายมือสุดไฟจะไหลผ่านหน้าคอนแทค C และ D ได้ถ้าอินพุต C ทำงาน (ON) และ D ไม่ทำงาน (OFF) นอกจากนั้น ไฟยังสามารถไหลผ่านหน้าคอนแทค E ได้ถ้าอินพุต E ทำงาน ซึ่งจะทำให้ไฟไหลผ่านได้แค่ครึ่งวงจรเท่านั้น ถ้าหน้าคอนแทค G ทำงาน (ON) ไฟจะไหลไปที่เอาต์พุต Y2 ได้

1.4 การเขียนโปรแกรม

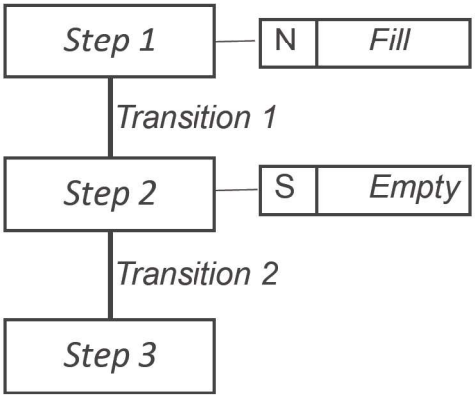
ในช่วงที่ PLC เริ่มนำมาใช้งานจริงจังในอุตสาหกรรมจะเขียนโปรแกรมโดยอาศัยพื้นฐานของวงจรรีเลย์ ซึ่งวิธีการนี้ทำให้ช่างเทคนิค และวิศวกร สามารถทำความเข้าใจได้ง่าย และในปัจจุบันวิธีการนี้ก็ยังคงใช้กันอยู่ ผู้เขียนโปรแกรมสามารถใช้ซอฟต์แวร์สร้างกราฟฟิกตามวงจรแลดเดอร์และดาวน์โหลดใส่ PLC จากนั้น PLC จะทำงานตามเงื่อนไขต่างๆ ของวงจรแลดเดอร์



รูป 1.5 ตัวอย่างวงจรแลดเดอร์

ปัจจุบันมีมาตรฐานสากล IEC 61131-3 สำหรับใช้ในการเขียนโปรแกรม โดยมาตรฐานนี้จะครอบคลุม 5 ภาษา ได้แก่ LADDER, SFC, Structure Text, Instruction List และ Function Block Diagram การเขียนโปรแกรมในรูปแบบอื่นๆ อีก เช่น Sequential Function Charts (SFC) ได้ถูกพัฒนาขึ้นเพื่อการสร้างโปรแกรมสำหรับระบบที่มีความซับซ้อนขึ้น วิธีการจะคล้ายกับโฟลว์ชาร์ต แต่มีขีดความสามารถสูงกว่า ตัวอย่าง SFC แสดงดังรูปที่ 1.6

จากรูปที่ 1.6 ฟังก์ชั่นการทำงาน เช่น Fill และ Empty ที่อยู่ในชาร์ตดูแล้วเหมือนไม่มีความหมายอะไร แต่ความจริงแล้ว ในฟังก์ชั่นจะประกอบด้วยโปรแกรมแลดเดอร์กลุ่มย่อยๆ วิธีการนี้แตกต่างจากโฟลว์ชาร์ตตรงที่มีเส้นทางการทำงานมากกว่าหนึ่งเส้นทางได้



รูป 1.6 ตัวอย่างโปรแกรม SFC

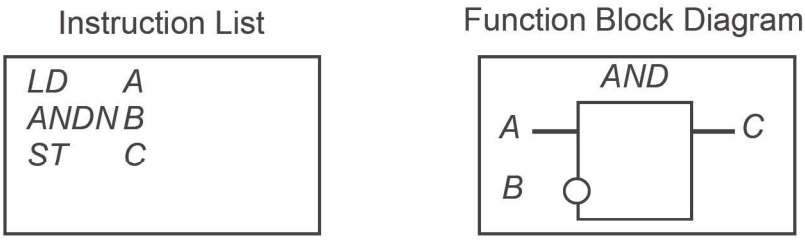
Structured Text เป็นวิธีการเขียนโปรแกรมอีกวิธีการหนึ่งที่ถูกพัฒนาขึ้นเพื่อให้ PLC รองรับการเขียนโปรแกรมภาษาสูงได้ ซึ่งจะคล้ายกับภาษา BASIC ตัวอย่างโปรแกรมอย่างง่ายแสดงในรูปที่ 1.7

ถ้าผู้ใช้งานมีความคุ้นเคยกับการเขียนโปรแกรมด้วยภาษา BASIC และ C มาบ้างแล้ว จะช่วยให้สามารถประยุกต์ใช้วิธีการเขียนโปรแกรมแบบนี้ได้อย่างมีประสิทธิผลและสามารถนำไปสร้างโปรแกรมควบคุมที่มีความซับซ้อนได้ง่ายกว่าวงจรแลดเดอร์ทั่วไป

```
IF x > 0 THEN
  y := TRUE;
  a := b + 1;
ELSE
  y := FALSE;
```

รูป 1.7 ตัวอย่างโปรแกรม Structured Text

ส่วนในรูปที่ 1.8 แสดงตัวอย่างการเขียนโปรแกรมด้วยภาษา Instruction List และ Function Block Diagram



รูป 1.8 ตัวอย่างโปรแกรม Instruction List และ Function Block Diagram

สรุปท้ายบท

PLC ถูกพัฒนาขึ้นเพื่องานควบคุมอัตโนมัติโดยนำระบบคอมพิวเตอร์มาประยุกต์ใช้งานแทนวงจรรีเลย์ จึงทำให้วงจรควบคุมที่ต้องใช้อุปกรณ์ต่างๆ มากมายถูกยุบอยู่ในรูปแบบโปรแกรมแลดเดอร์หรืออาจเรียกว่าแลดเดอร์ไดอะแกรม ต่อมาได้มีการพัฒนาการสร้างโปรแกรมในรูปแบบภาษาอื่นๆ ที่สามารถใช้กับโปรเซสหรือกระบวนการที่มีความซับซ้อนมากขึ้น ซึ่งเราจะกล่าวถึงในภายหลัง

Chapter

2

ระบบเลขฐานและรหัสข้อมูล

ก่อนอื่นเรามาเริ่มต้นด้วยการรู้จักคำว่า ดิจิต กันก่อนดีกว่า ดิจิต (digit) มาจากภาษาลาตินแปลว่า นิ้ว ตั้งแต่ไหนแต่ไรมามนุษย์เราเรียนรู้ที่จะใช้นิ้วมาช่วยในการนับ แต่เนื่องจากคนเรามีเพียง 10 นิ้ว เราจึงเคยชินกับการนับเลขฐานสิบ แต่ยังมีเลขฐานประเภทอื่นๆ อีกที่เราควรทำความใจเพื่อสร้างพื้นฐานที่ดีในการเรียนรู้ต่อไป

2.1 ระบบเลขฐาน

เนื่องจากวงจรแลดเดอร์ของ PLC นั้นเทียบเคียงมาจากวงจรรีเลย์ซึ่งมีสภาวะการทำงานแบบลอจิก (0 และ 1) ดังนั้นก่อนการใช้งาน PLC ควรมีความรู้พื้นฐานในเรื่องของเลขฐานและวิธีการแปลงเลขฐานเสียก่อน เพื่อทำให้เข้าใจการทำงานเบื้องต้นและสามารถประยุกต์ใช้งานได้ดีในลำดับถัดไป

เลขฐาน 10 (Decimal) เป็นเลขพื้นฐานที่เราคุ้นเคยมาตั้งแต่เด็กๆ เราสามารถบวก ลบ คูณ หาร ได้เป็นอย่างดี เมื่อมาถึงยุคดิจิตอลหรือยุคที่มีคอมพิวเตอร์ทำหน้าที่ในการคำนวณและบันทึกข้อมูล คอมพิวเตอร์เหล่านี้จะรู้จักตัวเลขเพียง 2 ตัวเท่านั้น คือ 0 กับ 1 ดังนั้นตัวเลขที่มีค่าเป็น 0 และ 1 เราจึงเรียกว่าเลขฐาน 2 ในระบบคอมพิวเตอร์จะทำงานโดยใช้เลขฐาน 2 เป็นหลักเพียงแต่มันจะใช้เลขฐาน 2 แทนตัวเลขฐานต่างๆ นอกจากเลขฐาน 2 และฐาน 10 แล้วยังมีเลขฐานอื่น ๆ ที่ใช้งานอีก เช่น ฐาน 4, ฐาน 8 และฐาน 16 เป็นต้น ตารางในรูป 2.1 และ 2.2 แสดงให้เห็นเลขฐานชนิดต่าง ๆ

ตารางที่ 2.1 ระบบเลขฐาน

ฐาน	ชื่อเรียก	หน่วยข้อมูล
2	Binary	Bit
8	Octal	Nibble
10	Decimal	Digit
16	Hexadecimal	Byte

ตารางที่ 2.2 ตัวเลขของ ฐาน 10, 2, 8 และ 16 ตามลำดับ

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

ในตารางที่ 2.2 แสดงเลขฐานต่างๆ การเปลี่ยนเลขฐานไม่มีผลต่อมูลค่าจริงของตัวเลข เป็นเพียงแค่การเขียนอยู่ในรูปแบบที่แตกต่างกันเท่านั้น กฎทางคณิตศาสตร์ยังคงใช้ได้เหมือนเดิม

2.1.1 **เลขไบนารี่** (Binary)

เลขไบนารี่ (หรือเลขฐาน 2) คือ ระบบเลขฐานที่ใช้ในระบบคอมพิวเตอร์ เลขฐาน 2 หนึ่งตัวเปรียบเหมือนหลอดไฟหนึ่งดวง ถ้ามีแรงดันไฟจ่ายให้หลอดไฟมันจะสว่างค่าบิตจะมีค่าเป็น “1” และเมื่อไม่มีแรงดันไฟหลอดไฟจะดับค่าบิตจะเป็น “0” ถ้ามีหลอดไฟมากกว่าหนึ่งดวง แต่ละดวงจะมีค่าแตกต่างกันตามลำดับตัวเลขไบนารี่

ตัวเลขฐาน 2 จำนวน 1 หลักจะเรียกว่า “บิต” (Bit มาจากคำว่า Binary Digit) จะเห็นว่าบิตทางด้านซ้ายมือจะมีค่ามากกว่าบิตทางด้านขวามือ โดยบิตทางด้านซ้ายมือที่มีค่ามากสุดเรียกว่า“เอ็มเอสบี ” (MSB ย่อมาจาก Most Significant Bit) ส่วนบิตขวามือที่มีค่าน้อยสุดเรียกว่า “แอลเอสบี ” (LSB ย่อมาจาก Least Significant Bit)

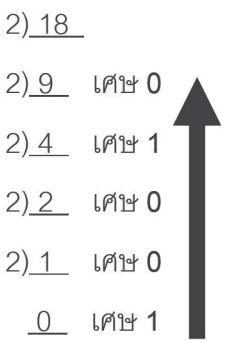
การแปลงเลขฐานสองเป็นเลขฐานสิบต้องอาศัยค่าประจำหลักของแต่ละบิตในเลขฐานสองที่ต้องการแปลง โดยเราจะแยกตัวเลขในแต่ละบิตมาคูณด้วยค่าประจำหลักแล้วนำผลลัพธ์จากการคูณดังกล่าวมารวมกัน จะได้เลขฐานสิบที่มีค่าตรงกับเลขฐานสองดังตัวอย่างต่อไปนี้ ค่าประจำหลักเกิดจากการใช้ 2 ยกกำลังด้วยตำแหน่งของหลักนั้นๆ ลองพิจารณาหลักที่ 4 เมื่อคำนวณค่า 2 ยกกำลัง 4 จะได้เท่ากับ 16

ตัวอย่างที่ 2.1 แสดงการแปลงเลข 10010 ให้อยู่ในรูปเลขฐานสิบ

$$10010 = (1 \times 2^4) + (0 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$$
$$= 16 + 0 + 0 + 2 + 0$$
$$= 18$$

ในทางกลับกัน เลขฐานสิบสามารถแปลงเป็นเลขฐาน 2 โดยการหาร ในตัวอย่างที่ 2.2 แสดงวิธีการแปลงนี้ เริ่มต้นด้วยการหารเลขฐาน 10 ด้วยเลข 2 เศษที่ได้ของการหารครั้งแรก จะเป็นค่าหลักต่ำสุดของเลขฐาน 2 ดังตัวอย่าง เราต้องการแปลง 19 ไปเป็นเลขฐาน 2 ให้นำ 19 ตั้งหารด้วย 2 จะได้ผลลัพธ์เป็น 9 เหลือเศษ 1 ซึ่งเศษนี้จะเป็นเลขหลักต่ำสุดของเลขฐาน 2 จากนั้นให้นำผลลัพธ์ที่ได้ไปหารด้วย 2 อีก ทำเช่นนี้ไปจนกว่าจะได้ผลลัพธ์เท่ากับ 0 จากนั้นจึงเอาเศษมาเรียงเป็นเลขฐานสองที่แปลงได้

ตัวอย่างที่ 2.2 แสดงการแปลงเลข 19 ให้อยู่ในรูปเลขฐานสอง



2) 18 *1,เริ่มต้นเอา 19 ตั้งหารด้วย 2*

2) 9 เศษ **0** *2,จากข้อ 1 ได้ผลลัพธ์เท่ากับ 9 เหลือเศษ 0*

2) 4 เศษ **1** *3,ผลลัพธ์จากข้อ 2 หารด้วย 2 ผลลัพธ์เท่ากับ 4 เหลือเศษ 1*

2) 2 เศษ **0** *4,ผลลัพธ์จากข้อ 3 หารด้วย 2 ผลลัพธ์เท่ากับ 2 เหลือเศษ 0*

2) 1 เศษ **0** *5,ผลลัพธ์จากข้อ 4 หารด้วย 2 ผลลัพธ์เท่ากับ 1 เหลือเศษ 0*

0 เศษ **1** *6,ผลลัพธ์จากข้อ 5 หารด้วย 2 ผลลัพธ์เท่ากับ 0 เหลือเศษ 1*

7,เมื่อหารจนได้ผลลัพธ์เป็น 0 แล้ว ให้นำเอาเศษมาเขียนเรียงกันจะได้ **10010** *ฐานสอง ที่มีค่าเท่า 18 ฐานสิบ*

ปัจจุบันเครื่องคิดเลขวิทยาศาสตร์สามารถแปลงเลขฐานต่าง ๆ เหล่านี้ได้ แต่สิ่งสำคัญคือความเข้าใจวิธีการแปลงค่าระหว่างเลขฐานและเมื่อทำบ่อย ๆ มันจะเข้าไปอยู่ในสมองของเราเอง

2.1.2 การคำนวณเลขฐานสอง

เนื่องจากเลขฐานสองเป็นพื้นฐานสำคัญของระบบดิจิตอลจึงขอกล่าวถึงวิธีการคำนวณหาค่าการบวก การลบ การคูณ และการหาร เลขฐานสอง ซึ่งมี วิธีการดังต่อไปนี้

2.1.2.1 การบวกและการลบเลขฐานสอง

การบวกและการลบเลขฐานสองมีวิธีการเหมือนกับการบวกลบเลขฐานสิบที่เราคุ้นเคย เพียงต่างกันที่การยืม เลขฐานสิบค่าของการยืมจะได้ครั้งละสิบ แต่ถ้าเป็นเลขฐานสองค่าของการยืมก็จะได้ครั้งละ10 ซึ่งมีหลักเกณฑ์สรุปดังตารางข้างล่างนี้

ตารางที่ 2.3 การบวกเลขฐานสอง

การบวกเลขฐานสอง			
ตัวตั้ง	ตัวบวก	ผลลัพธ์	ตัวทด
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

การบวกเลขฐานสอง จะมีวิธีการเช่นเดียวกับเลขฐาน 10 เพียงแต่เลขไบนารีมีแค่ “0” กับ “1” เมื่อมีค่าเกินกว่า “1” ก็จะทดไปหลักถัดไปที่สูงกว่า ดังแสดงข้างล่างนี้

0+0 = 0

0+1 = 1

1+0 = 1

1+1 = 10

ถ้าเลข 1+1 จะเท่ากับ 2 ของเลขฐานสิบ แต่เท่ากับ 10 ของเลขฐานสองเราจึงได้ผลลัพธ์เป็น 0 แล้วทด 1 ลองดูตัวอย่างที่ 2.4

ตัวอย่างที่ 2.3 จงบวกเลขฐานสอง $11011_2 + 11101_2$

```
  1 1 0 1 1
  1 1 1 0 1   +
-------------
ตอบ 1 1 1 0 0 0₂
```

การลบเลขฐานสอง ถ้าเลข 0 – 1 ตัวตั้งมีค่าน้อยกว่าตัวลบ จึงต้องไปยืมหลักหน้ามา 1 ในการยืมแต่ละครั้งของเลขฐานสองมีค่าเท่ากับ 2 (ในอีกความหมายหนึ่งคือ 10) เมื่อนำมาลบกับตัวลบคือ 1 จึงได้ผลลัพธ์เป็น 1 และอย่าลืมหักหลักที่ถูกยืมออกอีก 1 ด้วย แสดงดังตัวอย่าง 2.4

ตารางที่ 2.4 ตารางการลบเลขฐานสอง

การลบเลขฐานสอง			
ตัวตั้ง	ตัวลบ	ผลลัพธ์	ตัวยืม
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

ตัวอย่างที่ 2.4 จงลบเลขฐานสอง $11101_2 - 10110_2$

$$\begin{array}{r} 1\,1\,1\,0\,1 \\ 1\,0\,1\,1\,0 \quad - \\ \hline \end{array}$$

ตอบ $0\,0\,1\,1\,1_2$

นอกจากการคำนวณคณิตศาสตร์กับเลขไบนารีแล้ว จำนวนที่เป็นลบของเลขไบนารีจะสร้างปัญหาในการใช้เลขไบนารี ในตารางที่ 2.5 แสดงเลขไบนารีไม่มีเครื่องหมาย (Unsigned Binary) กับไบนารีมีเครื่องหมาย (Signed Binary)

ตารางที่ 2.5 ชนิดของเลขไบนารี่

ชนิด	คุณลักษณะ	ช่วงข้อมูล 1 ไบท์
Unsigned	เป็นเลขไบนารี่ที่มีแต่ค่าบวก	0 ถึง 255
Signed	บิตลำดับสูงสุด(MSB) ของเลขไบนารี่ใช้แสดงค่าบวกหรือลบ	-127 ถึง 127
2s compliment	เลขลบแทนด้วยการทำคอมพลีเมนต์เลขไบนารี่แล้วบวนด้วย 1	-128 ถึง 127

เลข Unsigned Binary บางครั้งถูกเรียกว่า เลขจำนวนเต็ม (Integer) โดยมีค่าเป็นบวกเท่านั้น ส่วน Signed Binary จะเป็นเลขที่มีจำนวนทั้งบวกและลบ ดังแสดงในตารางที่ 2.6 บิตอันดับสูงสุด (MSB) จะเป็นตัวแสดงว่ามีค่าเป็นบวกหรือลบ

ตารางที่ 2.6 เลข Signed binary

เลขฐานสิบ	เลขไบนารี่ 1 ไบท์
2	0000 0010
1	0000 0001
0	0000 0000
-0	1000 0000
-1	1000 0001
-2	1000 0010

จากตารางที่ 2.6 จะมีเลข 0 ถึง 2 ตัว ที่เป็นทั้งค่าบวกและลบ เพื่อแก้ปัญหาดังกล่าวเราสามารถทำ 2'S Complement เพื่อสร้างเลขที่มีค่าเป็นลบได้ดังแสดงในตัวอย่างที่ 2.5 ส่วนตารางที่ 2.7 แสดงเลขไบนารี่ที่ใช้แทนเลขบวกและลบหลังทำ 2'S Complement แล้ว

ตัวอย่างที่ 2.5 จงสร้างเลข -30_d ด้วยการทำ 2'S Complement

1, เขียนเลขไบนารี่ของค่าที่เป็นบวก เลข -30 ให้เขียนเป็น $30 = 00011110_2$

2, กลับค่าแต่ละบิตให้ตรงข้ามกัน ดังนั้น 00011110 => 11100001

3, นำเลขที่กลับค่าแล้ว บวกด้วย 1 11100001 + 1 = 11100010

ดังนั้น **11100010_2** จะใช้แทนเลข **-30**

ตารางที่ 2.7 เลขไบนารี่ที่ทำ 2'S Complement

เลขฐานสิบ	เลขไบนารี่ 1 ไบท์
2	0000 0010
1	0000 0001
0	0000 0000
-1	1000 0001
-2	1000 0010

2.2 รหัสในระบบดิจิตอล

การจัดชุดข้อมูลเลขฐานสอง 0 หรือ 1 เข้าด้วยกันให้เป็นกลุ่มแล้วแทนเลขใดเลขหนึ่ง เรียกว่า รหัส (Code) ในการรวมกลุ่มของเลขฐานสองจะมีคำจำกัดความของกลุ่มนั้นๆ แตกต่างกัน

เลขไบนารี่มักเขียนกันอยู่ใน 4 รูปแบบ คือ บิต, นิบเบิล, ไบต์, และเวิร์ด (Bit, Byte, Word) ดังตัวอย่างข้างล่างนี้

บิต (Bit) = เลขไบนารี 1 บิต

นิบเบิล (Nibble) = เลขไบนารี 4 บิต

ไบต์ (Byte) = เลขไบนารี 8 บิต

เวิร์ด (Word) = เลขไบนารี 16 บิต

รหัสในระบบดิจิตอลสามารถแบ่งออกได้ เป็น 2 ประเภทคือ

1.รหัสมีน้ำหนัก (Weighted Code)

เป็นรหัสเลขฐานสองที่ กำหนดให้มีค่าประจำตำแหน่งของแต่ละบิต เช่น รหัสบีซีดี (Binary Coded Decimal)

2.รหัสไม่มีน้ำหนัก (Non-Weighted Code)

เป็นรหัสเลขฐานสองที่ไม่ได้กำหนดให้มีค่าประจำตำแหน่งของแต่ละบิต เช่น รหัสเกรย์ (Gray Code) และรหัสแอสกี้ เป็นต้น

2.2.1 รหัส BCD (Binary Coded Decimal)

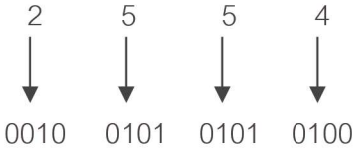
รหัส BCD หรือบางครั้งก็มีคนเรียกว่าเลข BCD เพราะรหัสจะใช้แทนตัวเลขเท่านั้น ถ้าจะแปลกันตามภาษาอังกฤษก็คือ รหัสเลขไบนารี่ที่ใช้แทนเลขฐานสิบ รหัส BCD จะใช้เลขไบนารี 4 บิต แทนเลข 1 ตัว ซึ่งหมายความว่าเลขไบนารี่ 1 ไบต์ จะใช้เป็นรหัส BCD ได้ 2 ตัว (Digit) จาก 00 ถึง 99 ซึ่งค่าไบนารีที่แท้จริง คือ 0-255 ขณะเดียวกันยังมีบิตแยกต่างหากเพื่อกำหนดเลขที่มีค่าเป็นลบ วิธีการนี้เป็นที่นิยมใช้กันมากเมื่อต้องการป้อนตัวเลขหรือแสดงผลตัวเลขบนคอมพิวเตอร์ ตัวอย่างของรหัส BCD แสดงในตารางที่ 2.8

ในตารางที่ 2.8 ใช้เลขไบนารี 4 บิต แทนตัวเลข 1 ตัว แต่ถ้ามีเลขมากกว่า 1 ตัว ก็จะใช้หลักการเดียวกัน คือ ใช้เลขไบนารี 4 บิต แทนเลขแต่ละตัว เช่น เลขฐานสิบ 4 ตัว จะใช้ใช้เลขไบนารี 16 บิต ดิจิตหรือหลักสูงสุด (Most Significant Digit, MSD) จะอยู่ด้านซ้ายมือสุด รหัส BCD จะมีค่าเทียบเท่าค่าเลขไบนารีในชุดนั้น ๆ แต่ไม่เกิน 9 ตัวอย่างที่ 2.6 แสดงการแปลงเลขฐานสิบเป็นรหัส BCD

ตารางที่ 2.8 รหัส BCD ที่ใช้แทนตัวเลขฐานต่างๆ

รหัสบีซีดี	เลขฐานสิบ	เลขฐานแปด	เลขฐานสิบหก
0 0 0 0	0	0	0
0 0 0 1	1	1	1
0 0 1 0	2	2	2
0 0 1 1	3	3	3
0 1 0 0	4	4	4
0 1 0 1	5	5	5
0 1 1 0	6	6	6
0 1 1 1	7	7	7
1 0 0 0	8		8
1 0 0 1	9		9

ตัวอย่างที่ 2.6 จงแปลงเลข 2554_{10} เป็นรหัส BCD



ตอบ 0010010101010100_{BCD}

ต้องพึงระวังอยู่เสมอว่าคำตอบที่ได้จากตัวอย่างที่ 2.6 เป็นรหัส BCD ที่ใช้แทนตัวเลขเท่านั้น แต่ไม่ใช่ค่าของตัวเลขที่เกิดจากการแปลงเลข 2554 เป็นเลขฐานสอง ซึ่งเลขฐานสองที่ได้จากการแปลงจะมีค่าเท่ากับ 100111111010_2

PLC ส่วนใหญ่จะเก็บรหัส BCD ในรูปของเวิร์ดซึ่งมีค่าอยู่ระหว่าง 0000 ถึง 9999 และมีฟังก์ชั่นหรือคำสั่งที่ใช้แปลงตัวเลขไปเป็น BCD หรือ จาก BCD กลับมาเป็นตัวเลขไบนารี่ได้ นอกจากนั้นยังสามารถทำการคำนวณคณิตศาสตร์ของ BCD ได้อีกด้วย เช่น บวก ลบ คูณ และหาร เป็นต้น แต่ควรหลีกเลี่ยงการคำนวณโดยใช้เลข BCD และควรใช้เลขจำนวนเต็ม (Integer) จะเหมาะสมกว่า ต้องระมัดระวังเสมอเมื่อตัวเลขเป็นรหัส BCD ควรแปลงเป็นตัวเลขจำนวนเต็มหรือไบนารีก่อนทำการคำนวณใด ๆ

2.2.2 รหัสเกรย์ (Gray Coded)

รหัสเกรย์นิยมนำมาใช้ในระบบควบคุมกลไกลเชิงแกนหมุน(motion)เพื่อบอกตำแหน่งของเพลาหมุน เช่น โรตารี่เอ็นโค้ดเดอร์ (Encoder) รหัสเกรย์เป็นรหัสที่ไม่มีน้ำหนักในตัว ซึ่งมีหลักในการเปลี่ยนเลขฐานสองเป็นรหัสเกรย์และเปลี่ยนจากรหัสเกรย์เป็นเลขฐานสองดังนี้

2.2.2.1 การเปลี่ยนเลขฐานสองเป็นรหัสเกรย์

- นำเลขฐานสองมาเขียนเรียงกันโดยเว้นช่องว่าง
- ดึงบิตสูงสุดหรือซ้ายมือสุดลงมา (MSB)
- บวกบิต MSB กับบิตถัดไปทางขวามือใส่ค่าที่ ได้โดยตัดตัวทดทิ้ง
- บวกบิตรองจาก MSB กับบิตถัดไปทางขวามือใส่ค่าที่ ได้โดยตัดตัวทดทิ้ง เช่นกันทำเช่นนี้ ไปจนถึงบิต LSB
- นำค่าที่ได้เขียนเรียงต่อกันนั่นคือคำตอบ

ตัวอย่างที่ 2.7 จงแปลงเลขฐานสอง 1011_2 ให้เป็นรหัสเกรย์

ฐานสอง 1 + 0 + 1 + 1

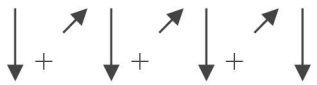
ตอบ 1 1 1 0 => 1110_2

2.2.2.1 การเปลี่ยนรหัสเกรย์เป็นเลขฐานสอง

- นำเลขฐานสองมาเขียนเรียงกันโดยเว้นช่องว่าง
- ดึงบิตสูงสุดลงมา (MSB)
- บวกบิตMSBที่ดึงลงมากับบิ ตถัดไปทางขวามือใส่ ค่าที่ได้โดยตัดตัวทดทิ้ง
- บวกผลลัพธ์ที่ ได้กับบิตถัดไปทางขวามื อใส่ค่าที่ได้โดยตัดตัวทดทิ้งเช่นกัน ทำเช่นนี้ ไปจนถึงบิต LSB
- นำค่าที่ได้เขียนเรียงต่อกันนั่นคือคำตอบ

ตัวอย่างที่ 2.8 จงแปลงรหัสเกรย์ 1110_{gray} ให้เป็นเลขฐานสอง

รหัสเกรย์ 1 1 1 0



ตอบ 1 0 1 1 => 1011_2

ตารางที่ 2.9 แสดงรหัสเกรย์ที่ใช้แทนเลขฐานสิบ ถ้าสังเกตให้ดีจะเห็นว่าค่าที่เพิ่มขึ้นของเลขฐานสิบแต่ละค่าจะทำให้บิตของรหัสเกรย์ เปลี่ยนเพียงบิตเดียวเท่านั้น ดังนั้นรหัสเกรย์จึงนำไปใช้งานกับ Absolute encoder อย่างกว้างขวางเพื่อแก้ปัญหาข้อมูลไม่ถูกต้องจากสัญญาณรบกวน

ตารางที่ 2.9 รหัสเกรย์ที่ใช้แทนตัวเลขฐานสิบ

เลขฐานสิบ	รหัสเกรย์
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111

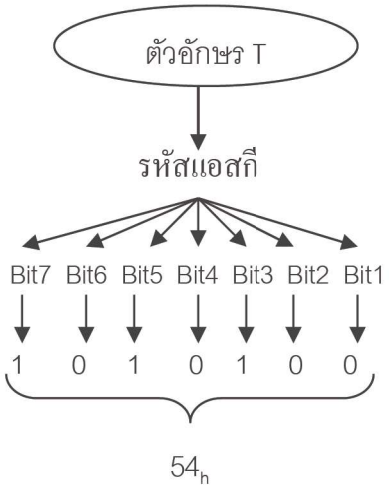
2.2.3 รหัสแอสกี(ASCII)

ASCII (American Standard Code for Information Interchange) เป็นการเรียกคำย่อจากคำเต็มจะอ่านว่า แอสกี รหัสแอสกีเป็นรหัสมาตรฐานของอเมริกัน ที่ใช้แทนตัวอักษร ตัวเลข ตัวอักขระต่างๆ ที่ใช้ติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ กับอุปกรณ์อินพุทเอาท์พุทต่างๆ เช่น คีย์บอร์ด จอแสดงผล เครื่องพิมพ์ เป็นต้น รหัสแอสกี มีขนาด 7 บิต เราสามารถถอดรหัสแอสกี ได้จากตารางที่ 2.10 โดยนำค่าบิตจากตารางมาเขียนเรียงต่อกัน ดังตัวอย่างต่อไปนี้

ตารางที่ 2.10 รหัสแอสกี

				B7→	0	0	0	0	1	1	1	1
				B6→	0	0	1	1	0	0	1	1
				B5→	0	1	0	1	0	1	0	1
B4	B3	B2	B1	Col (Hex) / Row (Hex)	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	′	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	″	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	′	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	|
1	1	0	1	D	CR	GS	-	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

ตัวอย่างที่ 2.9 จงแปลงตัวอักษร T ให้เป็นรหัสแอสกี



ตอบ 101 0100 หรือ 54_h

2.2.4 บิตพาริตี้ (Parity Bit)

ในการสื่อสารข้อมูลดิจิตอลมักมีการผิดพลาดเกิดขึ้นได้ ซึ่งอาจเกิดจากสัญญาณรบกวนภายในโรงงาน ดังนั้น เราสามารถตรวจสอบความถูกต้องในการสื่อสารหรือการรับส่งข้อมูลโดยวิธีการเพิ่ม Parity bit เข้ากับข้อมูลเพื่อเป็นการตรวจสอบอย่างง่าย ถ้าข้อมูลมีข้อผิดพลาดเกิดขึ้นมันอาจทำการส่งใหม่หรือยกเลิกการส่ง

การเช็ค Parity bit คือ การเพิ่มบิตเข้าไปอีก 1 บิตให้กับทุกๆ 8 บิตของข้อมูลจนกลายเป็น 9 บิต บิตที่เพิ่มขึ้นไม่ใช่ข้อมูล แต่ใส่เพื่อตรวจสอบว่า ข้อมูลมีความผิดพลาดหรือไม่ โดยใช้หลักการนับจำนวนบิตข้อมูลที่มีค่าเป็น 1 ในทุกๆ 8 บิต การเช็ค Parity นี้แบ่งได้ 2 วิธี คือ Odd Parity (Parity คี่) และ Even Parity (Parity คู่)

1.Odd Parity หมายถึง จำนวนบิตที่เป็น 1 ของข้อมูลที่จะรับ/ส่งเป็นเลขคี่ การคำนวณก็นับจำนวนบิตที่เป็น 1 ในข้อมูลจริง 8 บิตแรกว่ามีจำนวนเป็นเลขคี่และเลขคู่ ถ้าได้เลขคี่ให้เติม parity bit เป็น 0 ถ้านับได้เลขคู่ ก็เติม parity bit เป็น 1 เพื่อให้จำนวนเป็นเลขคี่ เช่น 10010011 มีจำนวนบิตที่เป็น 1 เป็น 4 ตัว ซึ่งเป็นจำนวนคู่ดังนั้น parity bit จะต้องเป็น 1 เพื่อให้จำนวนเป็นคี่ ดังนั้นข้อมูลที่เพิ่ม Parity bit แล้ว จะได้ **1** 1001 0011

2.Even Parity หมายถึง จำนวนบิตที่เป็น 1 ของข้อมูลที่จะรับ/ส่งเป็นเลขคู่ การคำนวณก็เพียงแต่นับจำนวนบิตที่เป็น 1ในข้อมูลจริง 8 บิตแรกว่ามีจำนวนเป็นเลขคี่และเลขคู่ ถ้าได้เลขคู่ให้เติม parity bit เป็น 0 ถ้านับได้เลขคี่ ก็เติม parity bit เป็น 1 เพื่อให้จำนวนเป็นเลขคู่ เช่น 1010 1100 มีจำนวนบิท 1เป็น 4 ตัว ซึ่งเป็นจำนวนคู่อยู่แล้ว จึงเติม 0 ดังนั้นข้อมูลที่เพิ่ม Parity bit แล้ว จะได้ **0** 1010 1100

การส่งข้อมูลระหว่าง PLC กับอุปกรณ์สื่อสารชนิดอื่น ทั้ง PLC และอุปกรณ์นั้นต้องตั้งค่า Parity ที่ตรงกัน ถ้าไม่ตรงกันก็จะทำให้แปลความหมายของข้อมูลที่รับเข้ามาไม่ถูกต้อง

ตารางที่ 2.11 แสดง Odd Parity และ Even Parity ในข้อมูล 1 ไบท์

	บิตข้อมูล	บิตพาริตี้
Odd Parity	10101110	1
	10111000	0
Even Parity	00101010	0
	10111101	1

2.2.5 Checksums

นอกจากวิธีการตรวจความผิดพลาดด้วยบิตพาริตี้แล้ว ยังมีวิธีการอื่น เช่น Checksum ที่สามารถใช้ตรวจสอบข้อมูลจำนวนมากๆ ได้ดีกว่า Checksum คือ การตรวจสอบความถูกต้องของข้อมูลที่ฝ่ายส่งข้อมูลจะทำการคำนวณค่า checksum แล้วใส่ค่า checksum ลงไปในข้อมูลที่จะส่ง ส่วนฝ่ายรับข้อมูลเมื่อรับข้อมูลมาแล้วจะนำเอาค่า checksum นี้มาตรวจสอบว่าถูกต้องหรือไม่

Checksum มีความสำคัญมากกับการรับ/ส่งข้อมูล แต่ผู้ใช้งานมักไม่ให้ความสำคัญกับมัน ถ้า PLC ที่ใช้งานมีความจำเป็นต้องส่งข้อมูล การใช้ Parity และ Checksum จะเป็นหัวใจสำคัญในการตรวจสอบข้อมูล เพราะว่าการส่งข้อมูลผิดเพียงบิตมีผลกระทบอย่างมากต่อระบบ ลองดูตัวอย่างข้างล่างนี้ สมมุติว่าตัวควบคุมอุณหภูมิส่งค่าอุณหภูมิให้กับ PLC ซึ่งมีค่าเท่ากับ 150 องศา แต่ข้อมูลที่ส่งให้เกิดผิดพลาดเพียงบิตเดียว จะทำให้ PLC รับค่า 70 องศา แทนที่จะเป็น 150 องศา จะเห็นว่าการส่งข้อมูลที่ผิดพลาดอาจมีผลกระทบกับระบบควบคุมได้

ข้อมูลที่ส่ง $1\,0\,0\,\mathbf{1}\,0\,1\,1\,0_2 = 150_d$

ข้อมูลที่รับ $1\,0\,0\,\mathbf{0}\,0\,1\,1\,0_2 = 70_d$

สรุปท้ายบท

ระบบควบคุม PLC และคอมพิวเตอร์จะคล้ายกัน คือ ใช้รหัสข้อมูลเป็นเลขฐาน 2 สำหรับพื้นฐานในการทำงาน ซึ่งเลขฐาน 2 นี้สามารถใช้เป็นตัวแทนรหัสข้อมูลประเภทอื่นๆได้ การเรียนรู้เรื่องเลขฐานและรหัสข้อมูลจะช่วยให้เราสามารถพัฒนาโปรแกรมแลดเดอร์ในระดับที่สูงขึ้นต่อไปได้ง่าย

Chapter

3

หลักการทำงานของพีแอลซี

ผู้ผลิต PLC อาจแบ่ง PLC ออกเป็นกลุ่มตามลักษณะโครงสร้างภายนอก บางผู้ผลิตแบ่งตามขนาดอินพุตเอาต์พุต แต่ไม่ว่าจะแบ่งเป็นกลุ่มอย่างไรก็ตามองค์ประกอบหลักของ PLC ยังคงเหมือนเดิมและมีกระบวนการทำงานที่ไม่ต่างกันมากนัก ต่อไปเราจะกล่าวถึงรายละเอียดหลักการทำงานเบื้องต้นของ PLC

3.1 ประเภทของ PLC

เราสามารถแบ่ง PLC ตามลักษณะโครงสร้างหรือรูปลักษณ์ภายนอกได้ 2 แบบ คือ แบบบล็อก (Block) กับ แบบโมดูล่าร์(Modular)

PLC แบบบล็อก จะมีขนาดเล็กและหน่วยความจำน้อย ส่วนอินพุตเอาต์พุตและแหล่งจ่ายไฟจะรวมอยู่ภายในตัว CPU แต่ผู้ผลิตมักออกแบบให้สามารถเพิ่มอุปกรณ์เสริมได้เพื่อความยืดหยุ่นในการใช้งาน เช่น ยูนิตขยายอินพุต/เอาต์พุต เป็นต้น PLC ชนิดนี้มีราคาถูก ความเร็วต่ำและฟังก์ชั่นการทำงานไม่สูงมากนัก เหมาะสำหรับงานควบคุมเครื่องจักรขนาดเล็ก

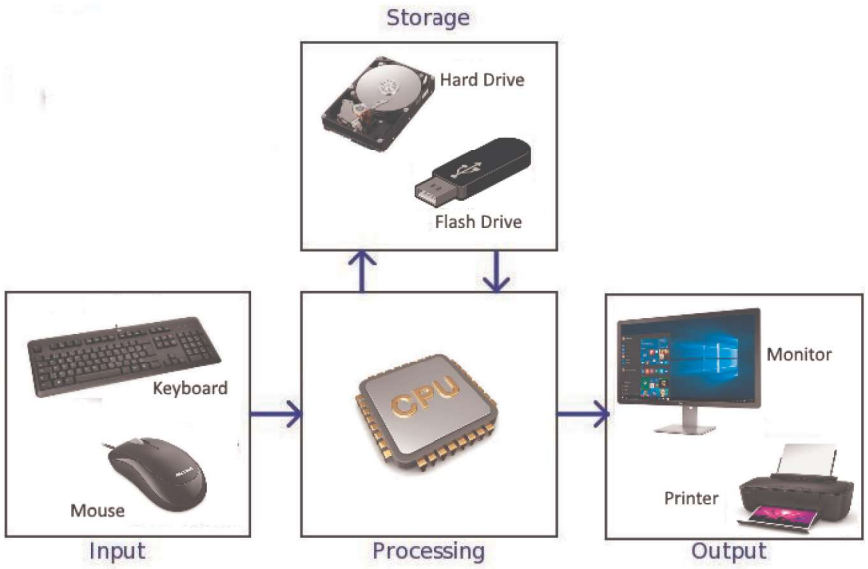
รูป 3.1 พีแอลซีแบบบล็อกของ Allen-Bradley รุ่น Micro 830

PLC แบบโมดูลล่าร์ จะมีอุปกรณ์ต่างๆแยกออกเป็นโมดูลหรือยูนิต เช่น ยูนิตอินพุต ยูนิตเอาต์พุต ยูนิต CPU และยูนิตแหล่งจ่ายไฟ เป็นต้น ทำให้ยืดหยุ่นในการออกแบบเพราะสามารถเลือกโมดูลที่ต้องการให้เหมาะสมกับการใช้งานได้ นอกจากนั้นยังมีชนิดของโมดูลให้เลือกใช้งานหลากหลาย เช่น โมดูลสื่อสาร โมดูลอนาลอก เป็นต้น แต่ PLC แบบนี้ราคาจะสูงกว่าแบบบล็อก

รูป 3.2 ตัวอย่าง PLC แบบโมดูลล่าร์

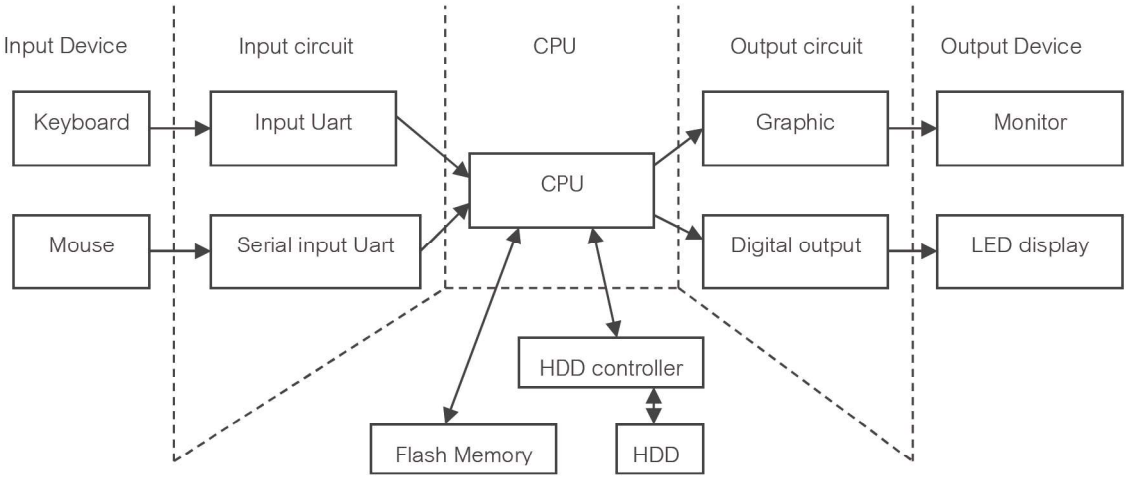
3.2 องค์ประกอบของ PLC

ก่อนที่จะทำความเข้าใจหลักการทำงานของ PLC เราลองมาพิจารณาดูหลักการทำงานของคอมพิวเตอร์กันก่อน ดังตัวอย่างในรูป 3.1 คือองค์ประกอบของคอมพิวเตอร์ โดยมีคีย์บอร์ดและเมาส์เป็นอุปกรณ์อินพุต ส่วนอุปกรณ์เอาต์พุตจะเป็นจอภาพแสดงผลและปริ้นเตอร์ มีฮาร์ดดิสก์และหน่วยความจำใช้สำหรับเก็บข้อมูล

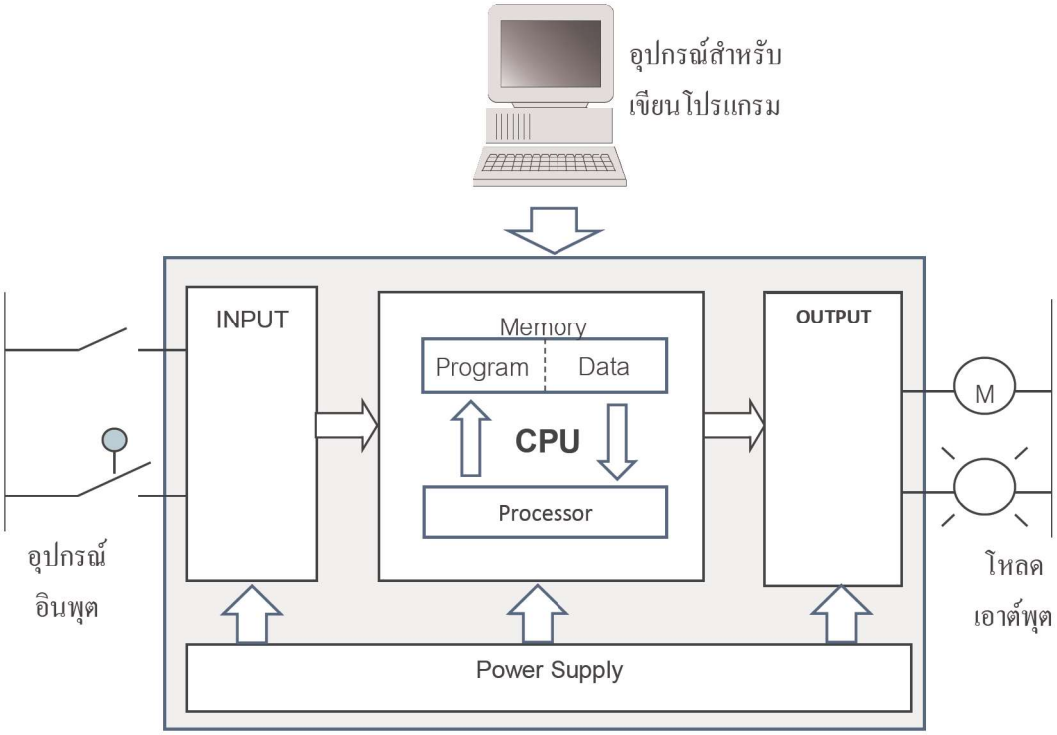


รูป 3.1 สถาปัตยกรรมของระบบคอมพิวเตอร์

จากรูป 3.1 อาจวาดใหม่เพื่อจัดองค์ประกอบให้ดูง่ายขึ้นดังแสดงในรูป 3.2 จากรูปจะเห็นว่าอินพุตและเอาต์พุตถูกแยกออกจากกัน อุปกรณ์อินพุตที่ใช้ป้อนข้อมูลจะอยู่ด้านซ้ายมือซึ่งต่อผ่านวงจรอินพุตและวงจรบัฟเฟอร์ก่อนเข้า CPU โดย CPU จะส่งข้อมูลออกเอาต์พุตผ่านวงจรอีกด้านหนึ่ง โดยมีหน่วยความจำและฮาร์ดดิสก์ใช้สำหรับเก็บข้อมูล



รูป 3.2 แสดงสถาปัตยกรรมคอมพิวเตอร์ในรูปแบบอินพุต/เอาต์พุต



รูป 3.3 แสดงองค์ประกอบต่างๆของ PLC

PLC มีหลักการทำงานเช่นเดียวกับระบบคอมพิวเตอร์แต่ออกแบบมาเพื่อใช้กับงานควบคุมอัตโนมัติโดยเฉพาะ ซึ่งสามารถเปรียบเทียบองค์ประกอบต่างๆ ได้ดังนี้

อุปกรณ์อินพุต คีย์บอร์ดและเมาส์ของคอมพิวเตอร์จะทำงานคล้ายกับ สวิตช์ เซ็นเซอร์ เป็นต้น อุปกรณ์เหล่านี้ทำหน้าที่รับข้อมูลเข้าสู่ระบบ

วงจรอินพุต(Input circuit) ของคอมพิวเตอร์เปรียบเสมือนยูนิตหรือการ์ดอินพุตของ PLC

ซีพียู เป็นหน่วยประมวลผลของ PLC ทำหน้านี้เช่นเดียวกับซีพียูของคอมพิวเตอร์ ในปัจจุบัน PLC หลายยี่ห้อก็ใช้ CPU ของ Intel เช่นเดียวกับคอมพิวเตอร์

วงจรเอาต์พุต*(Output circuit)* การ์ดกราฟฟิกของคอมพิวเตอร์เสมือนยูนิตหรือการ์ดเอาต์พุตของ PLC

อุปกรณ์เอาต์พุต เช่น หลอดไฟ มอเตอร์ ที่ต่อกับยูนิตเอาต์พุตของ PLC ก็จะคล้ายกับจอมอนิเตอร์ของคอมพิวเตอร์ที่ใช้แสดงผล

หน่วยความจำ*(Memory)* PLC จะแบ่งหน่วยความจำเป็นสองส่วนสำหรับเก็บโปรแกรมและข้อมูล

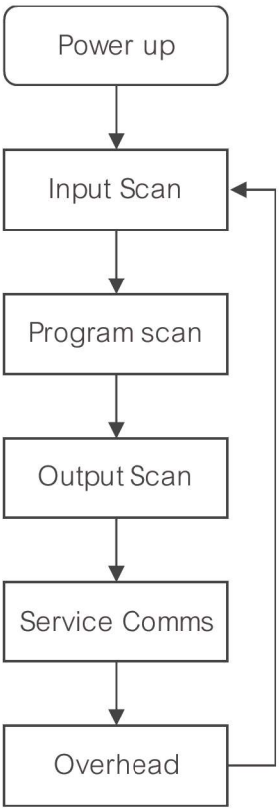
แหล่งจ่ายไฟ (Power Supply) เป็นแหล่งจ่ายพลังงานให้วงจรอินพุตเอาต์พุตและซีพียู

3.3 ลำดับการทำงาน (Operation Sequence)

PLC ส่วนใหญ่จะมีลำดับการทำงานพื้นฐาน 4-5 ขั้นตอนและจะทำงานวนซ้ำกันเช่นนี้ไปเรื่อยๆ เมื่อจ่ายไฟให้กับ PLC มันจะเริ่มตรวจสอบการทำงานของฮาร์ดแวร์และซอฟต์แวร์เพื่อหาข้อบกพร่อง ถ้าไม่มีปัญหาใด ๆ มันจะอ่านข้อมูลอินพุต (สัญญาณอินพุตต่าง ๆ) เข้ามาเก็บไว้ที่หน่วยความจำซึ่งเรียกว่า สแกนอินพุต (Input Scan) จากนั้นจะประมวลผลตามโปรแกรมแลดเดอร์(Ladder Program) โดยใช้ข้อมูลจากหน่วยความจำ การประมวลผลนี้ เรียกว่า โปรแกรมสแกน (Program Scan) ขณะที่ PLC ประมวลผลตามโปรแกรมแลดเดอร์นั้นค่าเอาต์พุตของโปรแกรมแลดเดอร์จะเปลี่ยนแปลงตามเงื่อนไขต่างๆ ของโปรแกรม แต่การเปลี่ยนแปลงนี้จะยังคงอยู่ในหน่วยความจำชั่วคราว(Temporary Memory) เท่านั้น เมื่อการสแกนแลดเดอร์ทำงานเสร็จสมบูรณ์แล้วข้อมูลเอาต์พุตในหน่วยความจำชั่วคราวนี้จะถูกส่งไปที่ยูนิตเอาต์พุตทำให้อุปกรณ์ที่ต่ออยู่ภายนอกทำงานตามผลลัพธ์ที่ได้จากการประมวลผลซึ่งเรียกว่า สแกนเอาต์พุต (Output Scan)

เมื่อสิ้นสุดการสแกนเอาต์พุต PLC จะสู่การให้บริการการสื่อสารกับอุปกรณ์ภายนอก (ถ้ามี) เช่น อุปกรณ์ป้อนโปรแกรม คอมพิวเตอร์ เป็นต้น จากนั้นก็จะเข้าสู่การทำ Overhead ซึ่งคือการใช้เวลาในการจัดการหน่วยความจำและปรับค่าไทม์เมอร์ เป็นต้น จากนั้นจะวนกลับไปเริ่มต้น

การทำงานใหม่ ซึ่งกระบวนการดังกล่าวจะใช้เวลามากหรือน้อยจะขึ้นอยู่กับความเร็วในการทำงานของ CPU รูปที่ 3.4 แสดงกระบวนการทำงานดังกล่าวนี้



รูป 3.4 วงรอบการทำงานของ PLC

3.3.1 การสแกนอินพุตและเอาต์พุต

การสแกนอินพุตและเอาต์พุตอาจทำให้ผู้ใช้งานสับสนบ้าง แต่มันเป็นเรื่องสำคัญมากที่ต้องทำความเข้าใจถ้าคุณต้องใช้งาน PLC กับงานที่ซับซ้อนมากขึ้นในอนาคต การสแกนอินพุตจะเป็นการรับรู้สถานะของสัญญาณอินพุตหรืออุปกรณ์อินพุตในเวลานั้นๆ แล้วนำข้อมูลเหล่านั้นมาประมวลผลตามโปรแกรมแลดเดอร์ ถ้าสถานะของอินพุตมีการเปลี่ยนแปลงในระหว่างการประมวลผลแลดเดอร์ PLC จะไม่สนใจสถานะที่เปลี่ยนแปลงนี้ยกเว้นกรณีที่เป็นอินพุตแบบขัดจังหวะ(Interrupt input) อีกปัญหาหนึ่งที่อาจเกิดขึ้นได้ถ้าการเปลี่ยนแปลงสถานะของอินพุตเร็วเกินไปและเกิดขึ้นระหว่างการประมวลผลแลดเดอร์จะทำให้ PLC ไม่สามารถรับรู้สถานะอินพุตดังกล่าวได้เลย

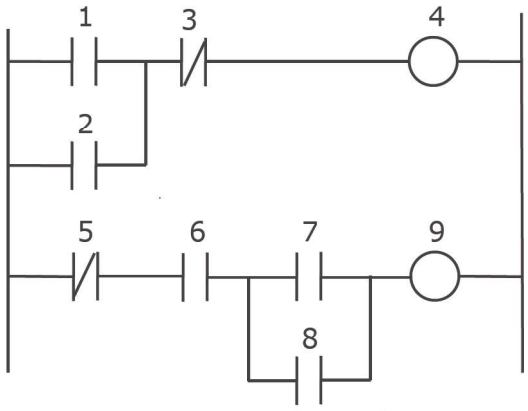
เมื่ออินพุตต่าง ๆ ที่ต่อกับ PLC ถูกสแกนมันจะเก็บค่าหรือสถานะต่างๆ ไว้ในหน่วยความ

จำอินพุต ส่วนเอาต์พุตที่ต่อกับ PLC ถูกสแกน มันจะนำข้อมูลจากหน่วยความจำเอาต์พุตส่งออกไปให้ภาคเอาต์พุต ขณะที่ PLC ทำการสแกนโปรแกรมแลดเดอร์จะใช้ค่าลอจิกหรือข้อมูลในหน่วยความจำเท่านั้นโดยไม่สนใจค่าหรือสถานะจริง ๆ ของอินพุตและเอาต์พุตในขณะนั้น

3.3.2 การสแกนโปรแกรมแลดเดอร์

โปรแกรมแลดเดอร์ใน PLC จะไม่มีกระแสไฟฟ้าที่ไหลจริงๆอย่างวงจรรีเลย์ แต่มันเป็นการประมวลผลทางลอจิกภายในโปรแกรมเท่านั้น การประมวลผลในหนึ่งรอบการสแกนจะเริ่มต้นที่ลอจิกแรกสุดเป็นลำดับขั้นไปเรื่อยๆ จนจบโปรแกรม

เราลองพิจารณาตัวอย่างในรูป 3.5 โปรแกรมแลดเดอร์จะถูกประมวลผลจากทางซ้ายไปขวาและจากบนลงล่างให้สังเกตุตัวเลขที่กำกับไว้ซึ่งใช้แสดงลำดับในการประมวลผล จากรูปการประมวลผลจะเริ่มจากวงจรแลดเดอร์ชุดบนสุดก่อนและประมวลผลไปจนถึงจุดสิ้นสุดของวงจรแลดเดอร์



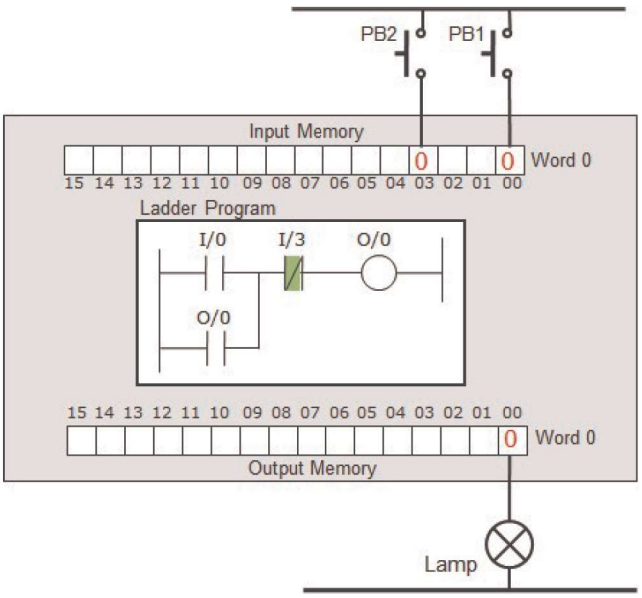
รูปที่ 3.5 แสดงลำดับในการประมวลผลแลดเดอร์

ลำดับขั้นในการสแกนแลดเดอร์มีความสำคัญเป็นอย่างมากเมื่อต้องแก้ไขปัญหาโปรแกรมแลดเดอร์ที่ใช้บิตเอาต์พุตทำหน้าที่เป็นอินพุต(หน้าคอนแทค)ในจุดอื่น ๆ ของโปรแกรม หรืออาจกล่าวอีกอย่างหนึ่งคือ เราสามารถเขียนโปรแกรมที่เป็นหน้าคอนแทค NO และ NC ที่มีชื่อเหมือนเอาต์พุตตัวหนึ่งแล้วใช้เป็นเงื่อนไขอินพุตในโปรแกรมจุดอื่นๆ ได้

เพื่อให้เห็นภาพที่ชัดเจนมากขึ้นของการประมวลผลโปรแกรมแลดเดอร์และสถานะของหน่วยความจำอินพุตและเอาต์พุตเราขอยกตัวอย่างดังแสดงในรูปที่ 3.6A ถึง 3.6G

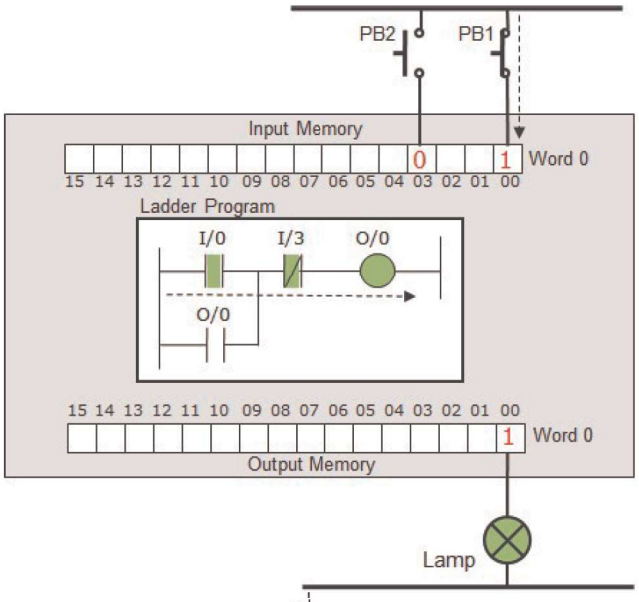
จากรูปที่ 3.6A จะมีสวิตช์ Push Button (PB1, PB2) ต่อเข้ากับอินพุต 00 และ 03 ซึ่งการ ON และ OFF ของสวิตช์จะส่งผลต่อลอจิกในหน่วยความจำอินพุต Word 0 บิตที่ 00 และ 03 ส่วนลอจิกของหน่วยความจำเอาต์พุต Word 0 บิตที่ 00 จะมีผลต่อการทำงานของหลอดไฟเช่นกัน

รูปที่ 3.6A แสดงสภาวะเริ่มต้นคือสวิตช์ PB1 และ PB2 อยู่ในสภาวะ OFF ส่วนลอจิกของหน่วยความจำอินพุตและเอาต์พุตเป็น '0'



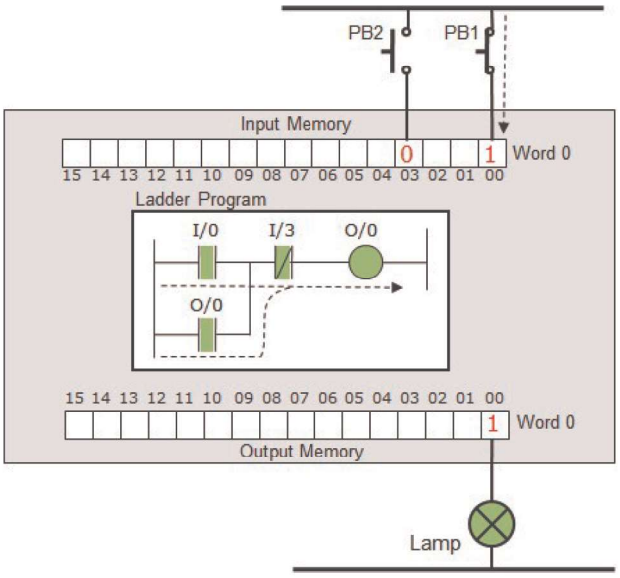
รูปที่ 3.6A

รูปที่ 3.6B แสดงสภาวะที่สวิตช์ PB1 ถูกกดซึ่งจะทำให้หน่วยความจำอินพุต word 0 บิตที่ 00 (I/0) มีลอจิก '1' จึงทำให้หน้าคอนแทค I/0 ทำงานและส่งผลให้หน่วยความจำเอาต์พุต O/0 มีลอจิกเป็น '1' และส่งค่าลอจิกนี้ออกไปที่เอาต์พุต O/0 ให้ทำงาน(ON)



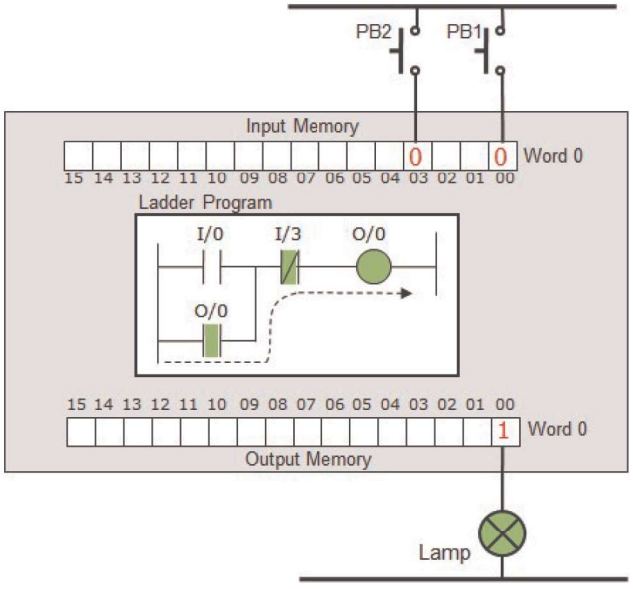
รูปที่ 3.6B

รูปที่ 3.6C แสดงสภาวะของวงรอบการทำงานของ PLC ในรอบถัดไปหลังจากที่เอาต์พุต O/0 มีลอจิกเป็น '1' PLC จะนำค่าลอจิกนี้มาประมวลผลโปรแกรมแลดเดอร์ในรอบถัดไป ดังนั้นในรอบนี้เราจะเห็นว่าหน้าคอนแทค O/0 จะ ON ทำให้เกิดเส้นทางที่ทำให้คอยล์เอาต์พุต O/0 ทำงานเป็น 2 ทางคือ ผ่านทางหน้าคอนแทค I/0 และหน้าคอนแทค O/0



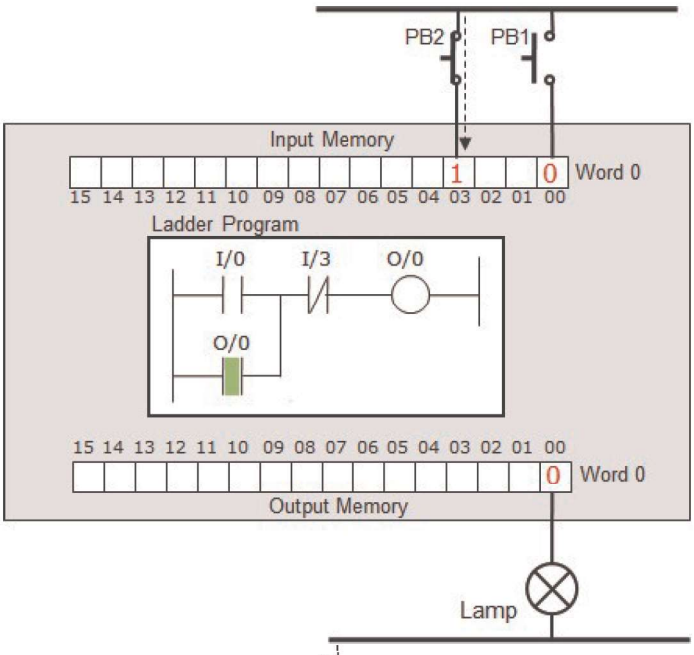
รูปที่ 3.6C

รูปที่ 3.6D แสดงสภาวะเมื่อปล่อยสวิตช์ PB1 แล้วลอจิกหน่วยความจำอินพุต I/0 จะเป็น '0' แต่คอยล์เอาต์พุต O/0 ยังคงทำงานค้างอยู่



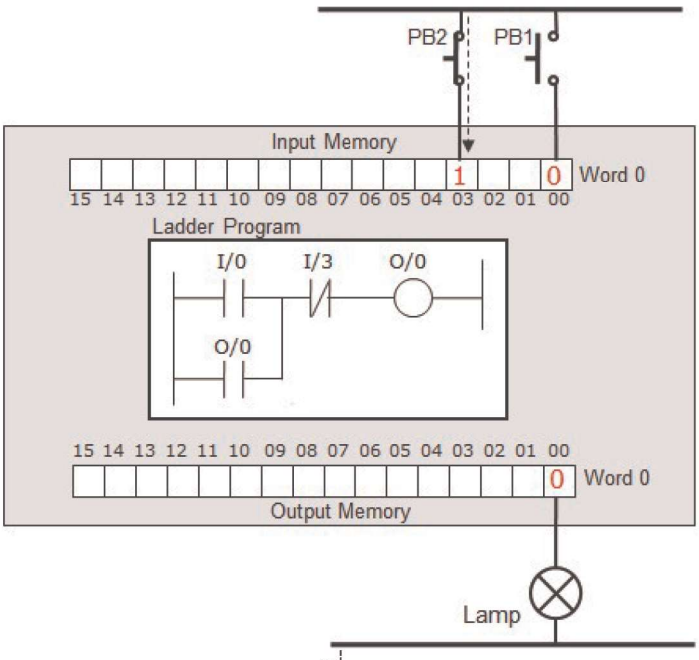
รูปที่ 3.6D

รูปที่ 3.6E แสดงสภาวะที่มีการกดสวิตช์ PB2 ซึ่งจะทำให้ลอจิกของหน่วยความจำอินพุต I/3 มีค่าเป็น '1' ทำหน้าคอนแทค NC ของ I/3 จะเปิดวงจรทำให้ O/0 มีลอจิกเป็น '0' ส่งผลให้เอาต์พุตไม่ทำงาน (OFF)



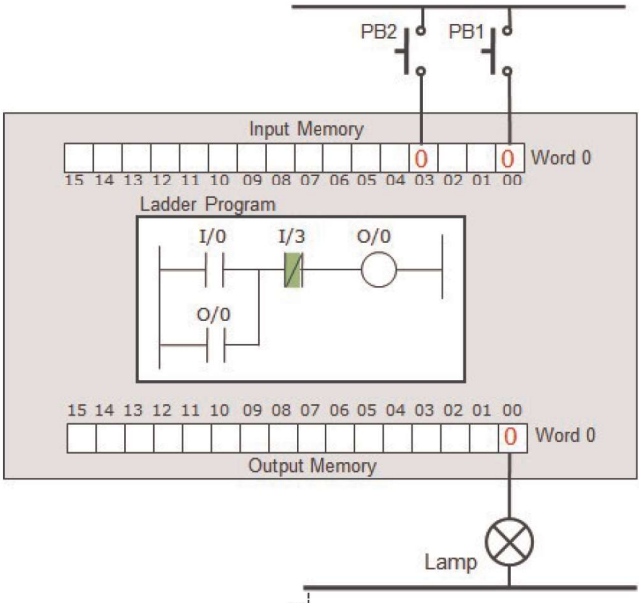
รูปที่ 3.6E

รูปที่ 3.6F แสดงสภาวะในรอบการทำงาน (scan) ถัดมาหน้าคอนแทค O/0 จะไม่ทำงาน



รูปที่ 3.6F

รูปที่ 3.6F แสดงสภาวะเมื่อปล่อยสวิตช์ PB2 แล้ว ทุกอย่างจะกลับสู่สภาวะเหมือนตอนเริ่มต้นการทำงาน



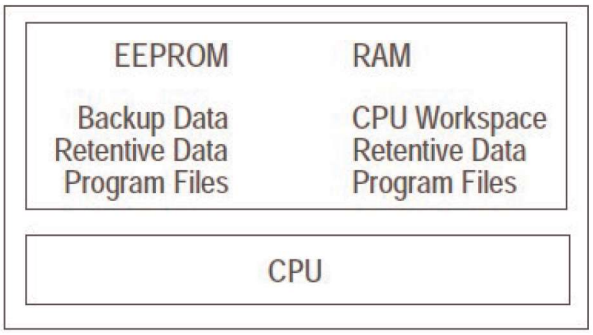
รูปที่ 3.6G

3.4 หน่วยความจำ (Memory)

หน่วยความจำเป็นอุปกรณ์ที่ใช้เก็บโปรแกรมและข้อมูลต่างๆ เมื่อ PLC ทำงานจะนำเอาโปรแกรมและข้อมูลในหน่วยความจำมาประมวลผลโดยซีพียู หน่วยความจำสามารถจำแนกตามการใช้งานได้ 2 ประเภท

> หน่วยความจำ RAM (Random Access Memory)

> หน่วยความจำ ROM (Read Only Memory)



รูปที่ 3.7 แสดงโครงสร้างหน่วยความจำ

3.4.1 หน่วยความจำ RAM

เป็นหน่วยความจำที่เก็บข้อมูลสำหรับใช้งานทั่วไป การอ้างอิงตำแหน่งที่อยู่ของข้อมูลใดๆ เพื่อการเขียนและการอ่านจะกระทำแบบการเข้าถึงโดยสุ่มคือ เรียกไปที่ตำแหน่งที่อยู่ข้อมูลใดก็ได้ หน่วยความจำนี้เรียกว่า แรม หน่วยความจำประเภทนี้จะเก็บข้อมูลไว้ตราบเท่าที่มีกระแสไฟฟ้ายังจ่ายให้วงจร หากไฟฟ้าดับเมื่อใด ข้อมูลก็จะสูญหายทันที

3.4.2 หน่วยความจำ ROM

เป็นหน่วยความจำอีกชนิดหนึ่ง โดยที่ข้อมูลใน ROM จะยังอยู่แม้ว่าจะไม่จ่ายไฟให้กับมันแล้วก็ตาม แต่ก็มีข้อเสียเรื่องการเข้าถึงข้อมูลจะช้ากว่า RAM นอกจากนั้นเวลาจะเขียนข้อมูลลงไปต้องใช้เครื่องเขียนเฉพาะจึงไม่เป็นที่นิยมใช้ ปัจจุบันได้มีการพัฒนา ROM ให้ใช้งานได้ง่ายขึ้นโดย ROM ที่นิยมใช้คือ EEPROM (Electrical Erasable Programmable ROM) สามารถเขียนโปรแกรมได้เช่นเดียวกับ ROM และลบได้ด้วยแรงดันไฟฟ้า สามารถเขียนโปรแกรมซ้ำใหม่ได้

สรุปท้ายบท

PLC และคอมพิวเตอร์เป็นอุปกรณ์ที่มีองค์ประกอบที่คล้ายคลึงกัน เช่น อินพุต และเอาต์พุต ส่วนการทำงานของ PLC จะเริ่มต้นจากการตรวจสอบตัวเอง จากนั้นจะเริ่มสแกนอินพุต ประมวลผลคำสั่งลอจิก และสแกนเอาต์พุตตามลำดับ ในขณะที่ PLC กำลังประมวลผลคำสั่งลอจิกอยู่นั้น PLC จะไม่สนใจสถานะอินพุตที่เปลี่ยนไป เช่นเดียวกับเอาต์พุตภายนอกก็จะไม่มีการเปลี่ยนแปลงเช่นกันจนกว่าการประมวลผลโปรแกรมแลดเดอร์จะสิ้นสุดลง ถ้าเราเข้าใจการกระบวนการทำงานของ PLC จะทำให้เราสามารถพัฒนาการเขียนโปรแกรมระดับสูงได้เป็นอย่างดี

Chapter

4

อุปกรณ์อินพุต

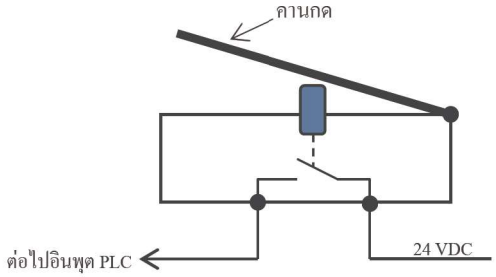
อุปกรณ์อินพุตที่ช่วยให้ PLC สามารถรับรู้สถานะของการเปลี่ยนแปลงต่างๆ ในระบบการผลิตหรือเครื่องจักรเปรียบเสมือนกับระบบประสาทสัมผัสต่างๆ ของมนุษย์ เช่น ตา หู จมูก และผิวหนัง เป็นต้น อุปกรณ์อินพุตจะใช้เพื่อตรวจสอบสถานะเป็นแบบดิจิตอล(ON/OFF) และแบบอนาล็อก(Analog) ในบทนี้เราจะกล่าวเฉพาะอุปกรณ์อินพุตที่ให้สัญญาณแบบลอจิกเท่านั้น

4.1 ประเภทของอุปกรณ์อินพุต

วิธีการตรวจจับของอุปกรณ์อินพุตว่า **มีวัตถุ** หรือ **ไม่มีวัตถุ** ทำได้ 2 วิธี คือ แบบสัมผัส (Contact) และแบบไม่สัมผัส แบบสัมผัสจะมีการสัมผัสทางกลโดยตรงกับวัตถุและผลการตรวจจับมาจากแรงที่เกิดขึ้นระหว่างอุปกรณ์ตรวจจับกับวัตถุ ในขณะที่แบบไม่สัมผัสจะทำงานเมื่อมีวัตถุอยู่ใกล้หรือในรัศมีการตรวจจับแต่วัตถุไม่ต้องสัมผัสกับอุปกรณ์ตรวจจับ ต่อไปเราจะอธิบายรายละเอียดของอุปกรณ์อินพุตดังต่อไปนี้

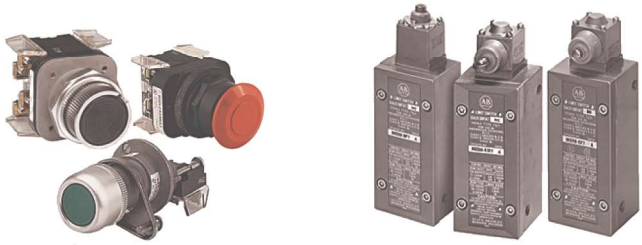
4.1.1 **สวิตช์ทางกล** (Mechanical Switches)

สวิตช์ทางกลจะทำงานเมื่อมีแรงมากระทำที่ตัวมัน แรงทางกลจะส่งผ่านกลไกลซึ่งอาจเป็นคานหรืออาจเป็นสปริงเพื่อลดแรงที่กระทำกับสวิตช์ดังแสดงในรูปที่ 4.1 เป็นสวิตช์ที่ใช้คานในการรับแรง สวิตช์ทางกลจะมีหน้าคอนแทคให้เลือกใช้งานทั้งแบบ NO(Normally Open) กับ NC(Normally Closed) ทั้งนี้ขึ้นอยู่กับการออกแบบของผู้ผลิต



รูปที่ 4.1 แสดงโครงสร้างของสวิตช์ทางกล

รูปที่ 4.2 แสดงตัวอย่างสวิตช์ทางกลที่นิยมใช้ในภาคอุตสาหกรรม คือ สวิตช์ Push Button และลิมิตสวิตช์ จากรูปจะเห็นว่าลิมิตสวิตช์จะมีตัวรับแรงหลากหลายรูปแบบ เช่น ลูกกลิ้ง และก้านสัมผัส เป็นต้น นอกจากนั้นลิมิตสวิตช์ยังมีให้เลือกใช้ทั้งกับโหลดเบาและโหลดหนักได้ตามความต้องการ



รูปที่ 4.2 ตัวอย่างสวิตช์ Push Button และลิมิตสวิตช์

4.1.2 พร็อกซิมิตี้เซ็นเซอร์ (Proximity Sensor)

พร็อกซิมิตี้เซ็นเซอร์หรือเรียกอีกอย่างหนึ่งว่าพร็อกซิมิตี้สวิตช์ที่ใช้ตรวจจับวัตถุที่เข้ามาใกล้แล้วเปลี่ยนสถานะของเอาต์พุตให้ทำงาน (ON) หรือหยุดทำงาน (OFF) ในโรงงานอุตสาหกรรมจะมีพร้อกซิมิตี้เซ็นเซอร์ที่นิยมใช้อยู่ 2 ประเภท ดังนี้

4.1.2.1 พร็อกซิมิตี้เซ็นเซอร์แบบคาปาซิตีฟ (Capacitive)

รูปที่ 4.3 ตัวอย่างพร็อกซิมิตี้เซ็นเซอร์แบบคาปาซิตีฟ

พร็อกซิมิตี้เซ็นเซอร์แบบคาปาซิตีฟ สามารถตรวจจับวัตถุๆ ได้ทุกชนิดไม่ว่าจะเป็นโลหะหรืออโลหะ โดยทั่วไปมีระยะตรวจจับประมาณ 2-3 ซม. หรือมากกว่าทั้งนี้ขึ้นอยู่กับวัตถุที่มันตรวจจับ การตรวจจับ(Sensing)อาศัยหลักการของตัวเก็บประจุไฟฟ้าที่ประกอบด้วยขั้วไฟฟ้า (หรือแผ่นเพลต) 2 ขั้ว แต่ละขั้วจะเก็บประจุไฟฟ้าชนิดตรงกันข้ามกัน ประจุไฟฟ้านั้นจะถูกเก็บไว้ที่ผิวหน้าของแผ่นเพลตโดยมีฉนวนกั้นเอาไว้ เนื่องจากแผ่นเพลตแต่ละแผ่นจะเก็บประจุไฟฟ้าชนิดตรงข้ามกันแต่จะมีปริมาณเท่ากัน ดังนั้นพร็อกซิมิตี้สวิตช์แบบคาปาซิตีฟจึงได้นำหลักการดังกล่าวมาใช้งานเพียงแต่ค่าฉนวนจะแตกต่างกันตามวัตถุที่ต้องการตรวจจับซึ่งส่งผลกับการเปลี่ยนแปลงค่าการเก็บประจุไฟฟ้า (Capacitance) ดังแสดงในสมการข้างล่างนี้

$$C = \frac{Ak}{d}$$

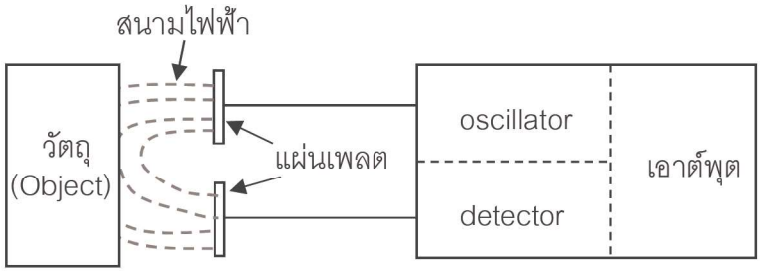
C = ค่าความจุ

A = พื้นที่ผิวของแผ่นเพลต(Plate)

k = ค่าคงที่ฉนวน

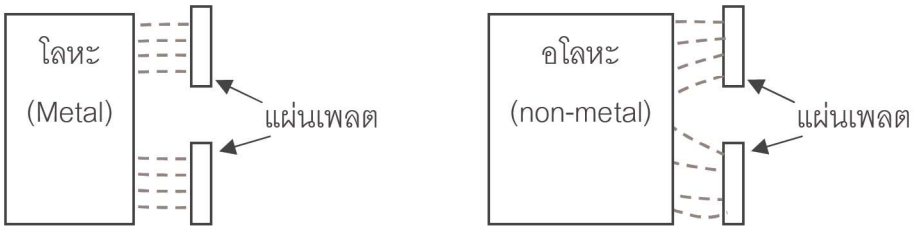
d = ระยะห่างของแผ่นเพลตหรืออีเลคโทรด

พื้นที่ของแผ่นเพลตและระยะห่างของมันจะคงที่ แต่ค่าคงที่ฉนวนของช่องว่างรอบตัวมันจะแปรเปลี่ยนตามวัตถุที่เข้ามาใกล้ดังแสดงในรูป 4.4 ตัว Oscillator ถูกใช้เพื่อสร้างสนามไฟฟ้าซึ่งส่งผลต่อค่าความจุ (Capacitance) ของแผ่นเพลต เมื่อค่านี้เปลี่ยนแปลงมากกว่าค่าความไว (Sensitivity) ที่กำหนดไว้มันจะทำให้เอาต์พุตทำงาน



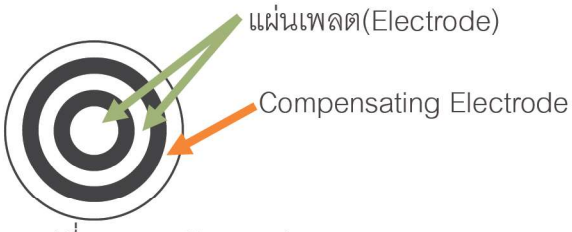
รูปที่ 4.4 แสดงหลักการทำงานพร็อกซิมิตี้เซ็นเซอร์แบบคาปาซิตีฟ

พร็อกซิมิตี้เซ็นเซอร์แบบคาปาซิตีฟทำงานได้ดีกับวัตถุพวกอโลหะและฉนวนด้วย เช่น พลาสติกที่มีค่าความเป็นฉนวนสูงซึ่งจะช่วยเพิ่มค่าความจุ (Capacitance) ได้ดี ในขณะเดียวกันก็ทำงานกับโลหะได้เช่นกัน เพราะวัตถุที่นำไฟฟ้านั้นเสมือนทำให้แผ่นเพลต (electrode) ใหญ่ขึ้น ทำให้ค่าความจุเพิ่มขึ้น



รูปที่ 4.5 แสดงการการใช้งานกับโลหะและอโลหะ

ในทางปฏิบัติเราไม่สามารถใช้แผ่นเพลตได้ซึ่งอาจทำให้เซ็นเซอร์มีขนาดใหญ่ ดังนั้น มันจะถูกม้วนเป็นวงแหวน ในรูปที่ 4.6 แสดงให้เห็นว่ามีโลหะวงแหวนอยู่ข้างใน 2 ชุด ซึ่งเรียกว่า Capacitor Electrode แต่โลหะวงแหวนด้านนอกสุด (Compensating electrode) จะถูกเพิ่มเข้าไปเพื่อชดเชยการผันแปรที่อาจเกิดขึ้น ถ้าไม่มีวงแหวนชดเชยนี้จะทำให้เซ็นเซอร์มีความไวต่อสิ่งสกปรก เช่น น้ำมัน ฝุ่นผงต่างๆ ที่อาจมาเกาะติดกับตัวเซ็นเซอร์ได้



รูปที่ 4.6 การจัดวางแผ่นเพลต(Electrode)

ตารางที่ 4.1 แสดงคุณสมบัติค่าคงที่ฉนวนถูกใช้เพื่อประเมินขนาดและความไว (Sensitivity) ของเซ็นเซอร์ ค่าความเป็นฉนวนที่ต่างกันนี้อาจนำเซ็นเซอร์ถูกนำไปใช้งานเพื่อจำแนกประเภทของวัสดุได้ นอกจากนั้นระยะและความแม่นยำในการตรวจจับจะถูกกำหนดด้วยขนาดของตัวเซ็นเซอร์ ถ้าขนาดใหญ่จะตรวจจับได้ระยะไกลกว่าแต่ความแม่นยำอาจน้อยกว่า

ตารางที่ 4.1 แสดงค่าคงที่ฉนวน (Dielectric constant) ของวัสดุต่างๆ

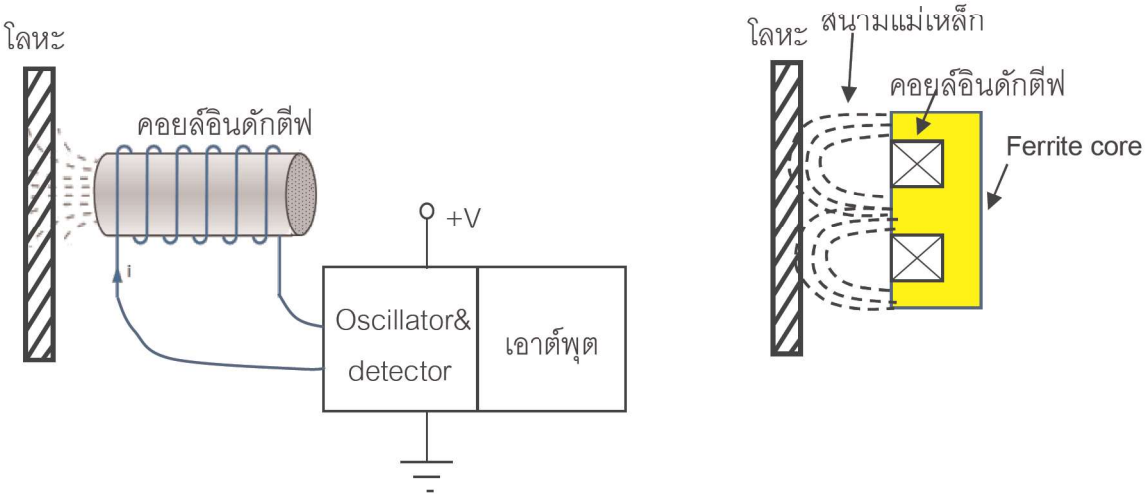
วัสดุ(Material)	ค่าคงที่ฉนวน	วัสดุ(Material)	ค่าคงที่ฉนวน
ABS resin pellet	1.5-2.5	paper	1.6-2.6
acrylic resin	2.7-4.5	petroleum	2.0-2.2
air	1.0	rubber	2.5-35
glass	3.1-10	salt	6.0
milk, powdered	3.5-4	Teflon	2.3-2.8
paint	5-8	water	80
PVC	3.7	wood	2-7

4.1.2.2 **พร็อกซิมิตี้เซ็นเซอร์แบบอินดักตีฟ** (Inductive)

พร็อกซิมิตี้เซ็นเซอร์ชนิดนี้เป็นเซ็นเซอร์ชนิด ON/OFF ที่นิยมใช้กันมากในงานอุตสาหกรรม ข้อดีของเซ็นเซอร์ชนิดนี้คือไม่มีส่วนที่เคลื่อนที่และอายุการใช้งานยาวนาน แต่ข้อเสียที่พบได้บ่อย คือชิ้นงานอาจมากระแทกที่ตัวเซ็นเซอร์ทำให้เกิดความเสียหายได้

รูปที่ 4.7 ตัวอย่างพร็อกซิมิตี้เซ็นเซอร์แบบอินดักตีฟ

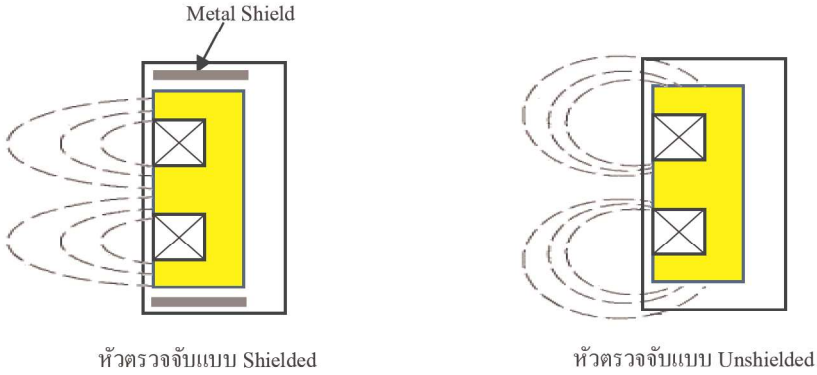
พร็อกซิมิตี้เซ็นเซอร์ชนิดนี้ใช้กระแสเหนี่ยวนำที่เกิดจากสนามแม่เหล็ก เมื่อมีวัตถุที่เป็นโลหะมาอยู่ใกล้จะทำให้ค่าเหนี่ยวนำ (Inductance) เกิดการเปลี่ยนแปลง สนามแม่เหล็กความถี่สูงจะถูกสร้างจากคอยล์ (Inductor) ดังแสดงในรูปที่ 4.8 ถ้ามีโลหะเข้ามาใกล้สนามแม่เหล็กจะเปลี่ยนแปลงและทำให้เกิดกระแสไฟไหลขึ้นในวัตถุนั้น ซึ่งกระแสนี้ทำให้เกิดสนามแม่เหล็กใหม่ที่ตรงกันข้ามกับสนามแม่เหล็กเดิม ผลลัพธ์ดังกล่าวทำให้เกิดการเปลี่ยนแปลงค่าความเหนี่ยวนำ (Inductancc) ของคอยล์ในคอยล์อินดักตีฟ การวัดค่าเหนี่ยวนำของคอยล์อินดักตีฟสามารถทำให้รู้ว่ามีโลหะเข้ามาอยู่ใกล้เซ็นเซอร์นี้หรือไม่ ด้วยวิธีการนี้เราสามารถตรวจจับโลหะได้หลายประเภทแต่ต้องใช้พร็อกซิมิตี้สวิตช์หรือเซ็นเซอร์ที่ออกแบบมาตามประเภทของโลหะนั้นจึงจะใช้ได้ผลดี



รูป 4.8 แสดงหลักการทำงานของพร็อกซิมิตี้เซ็นเซอร์แบบอินดักตีฟ

พร็อกซิมิตี้เซ็นเซอร์แบบอินดักตีฟสามารถตรวจจับวัตถุที่อยู่ห่างออกไปได้ไกลถึง 2-3 ซม. หรือมากกว่า จากรูปที่ 4.9 สนามแม่เหล็กของพร็อกซิมิตี้เซ็นเซอร์ที่หัวตรวจจับเป็นแบบ

Unshielded จะครอบคลุมพื้นที่การตรวจจับกว้างกว่าในบริเวณรอบหัวคอยล์ แต่การใส่ Shield ที่หัวตรวจจับซึ่งเป็นตัวห่อหุ้มรอบคอยล์จะทำให้สนามแม่เหล็กถูกบีบให้เล็กลง แต่จะมีทิศทางที่แน่นอนกว่า ดังนั้นพร็อกซิมิตี้เซ็นเซอร์ที่หัวตรวจจับแบบ Shielded จึงนิยมใช้กับงานที่ต้องการทิศทางและความแม่นยำในการตรวจจับที่ดีขึ้น

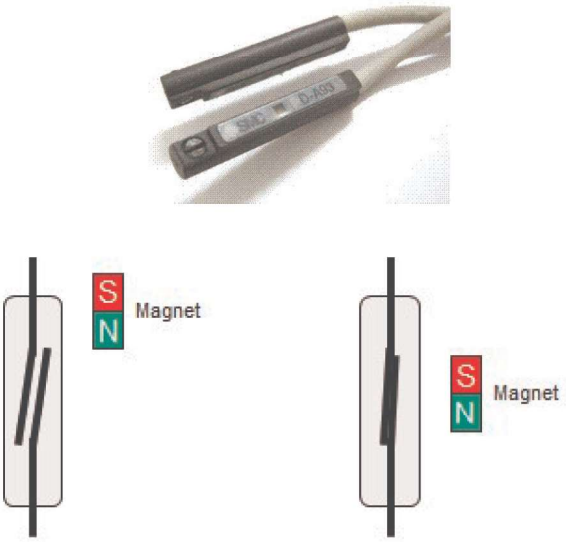


รูปที่ 4.9 หัวตรวจจับแบบ Shielded กับ Unshielded

4.1.2.3 **รีดสวิตช์** (Reed Switch)

รีดสวิตช์เป็นสวิตช์ไฟฟ้าชนิดหนึ่ง ซึ่งภายในจะประกอบด้วยโลหะที่มีคุณสมบัติเป็นแม่เหล็ก 2 ชิ้นที่วางไว้ใกล้กัน ถ้ามีสนามแม่เหล็กเข้ามาใกล้สวิตช์จะทำให้ปิดวรจรและจะเปิดวงจรเมื่อไม่มีสนามแม่เหล็ก ดังแสดงในรูปที่ 4.10

ในงานอุตสาหกรรมเรามักพบรีดสวิตช์ติดตั้งกับกระบอกสูบลมเพื่อตรวจสอบการเคลื่อนที่เข้าหรือเคลื่อนที่ออก นอกจากนั้นยังสามารถนำไปใช้เป็นสวิตช์แม่เหล็กเพื่อตรวจสอบการเปิดปิดประตูเครื่องจักรได้อีกด้วย



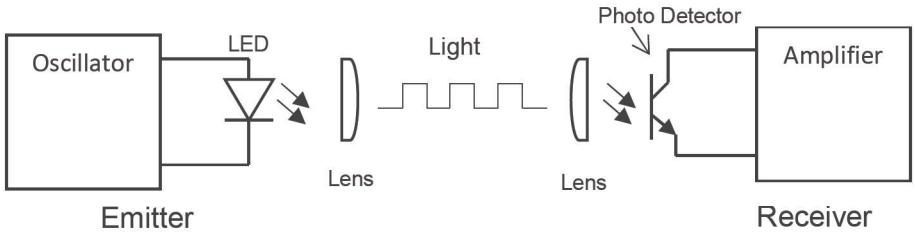
รูป 4.10 ตัวอย่างรีดสวิตช์

4.1.3 โฟโตอิเลคทริกเซ็นเซอร์ (Photo-electric sensor)

โฟโตอิเลคทริกเซ็นเซอร์หรือโฟโตอิเลคทริกสวิตช์ คือ อุปกรณ์ที่ใช้ตรวจจับความหนาแน่นหรือความเข้มของแสงที่นำมาประยุกต์ใช้งานเพื่อตรวจจับวัตถุในงานอุตสาหกรรม

รูปที่ 4.11 โฟโตอิเล็กทริกเซ็นเซอร์

โฟโตอิเล็กทริกเซ็นเซอร์ประกอบด้วยตัวส่งแสง(Emitter) และตัวรับแสง (Detector หรือ Receiver) ตัวส่งแสงจะส่งลำแสงซึ่งมีทั้งแบบมองเห็นและมองไม่เห็นส่วนใหญ่จะใช้ LED หรือ Laser Diode ส่วนตัวรับแสงจะทำจาก Photo diode หรือ Photo-transistor เพื่อทำหน้าที่รับแสง รูปที่ 4.12 แสดงหลักการทำงานเบื้องต้นของโฟโตอิเล็กทริกเซ็นเซอร์



รูปที่ 4.12 แสดงหลักการทำงานเบื้องต้นของโฟโตอิเล็กทริกเซ็นเซอร์

จากรูปข้างบนลำแสงจะถูกกำเนิดจากตัวส่งด้วย LED ผ่านเลนส์เพื่อโฟกัสแสง ส่วนที่ตัวรับแสงจะมีเลนส์อีกตัวเพื่อโฟกัสแสง ถ้าลำแสงนี้ถูกบังแสงที่ตัวรับจะหายไปหรือน้อยกว่าปกติ ทำให้ตัวรับแสงรู้ว่ามีวัตถุบังอยู่ ลำแสงที่กำเนิดขึ้นจะมีความถี่คลื่นแสงที่เซ็นเซอร์สามารถแยกแยะจากแสงปกติทั่วไปได้ โดยแสงจากตัวส่งจะเป็นพลัส (Pulse) ตามความถี่ที่ตั้งไว้ ความถี่ที่กำเนิดขึ้นจะอยู่ในย่านกิโลเฮิรตซ์ (Khz) เมื่อตัวรับแสงได้รับแสงมันจะตรวจสอบให้แน่ใจว่าเป็นแสงที่มีความถี่เดียวกัน ถ้าแสงที่ได้รับมีความถี่เดียวกันมันจะรู้ทันทีว่าแสงที่ส่งมาไม่มีวัตถุบัง โฟโตอิเล็กทริกเซ็นเซอร์ แบ่งตามลักษณะของตัวส่งและตัวรับแสงได้ 3 ประเภท ดังนี้

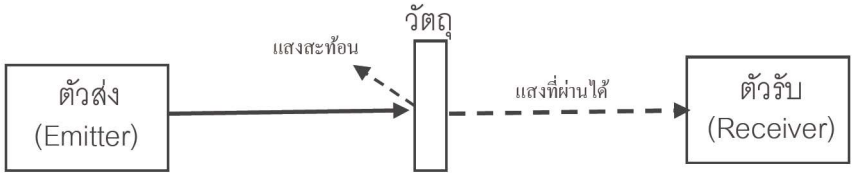
1.แบบตัวรับ/ตัวส่งแยกกัน (Through Beam)

2.แบบใช้แผ่นสะท้องแสง (Reto-Reflective)

3.แบบตรวจจับวัตถุโดยตรง (Diffuse-Reflective)

4.1.3.1 แบบตัวรับ/ส่งแยกกัน (Through Beam)

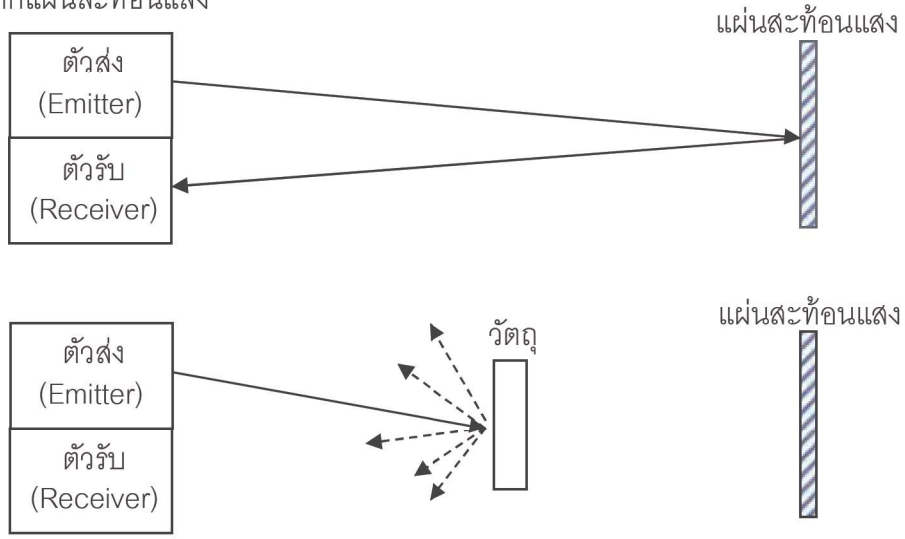
ตัวส่งแสงจะติดตั้งให้แสงส่งตรงไปยังที่ตัวรับเมื่อแสงถูกบังแสดงว่ามีวัตถุ ดังแสดงในรูปที่ 4.13 วิธีการนี้เหมาะสำหรับวัตถุทึบแสงและวัตถุที่เป็นมันเงา นอกจากนั้นระยะการตรวจจับจะไกลที่สุดเมื่อเทียบกับวิธีการอื่น แต่เซ็นเซอร์แบบตัวรับ/ตัวส่งแยกกันมีข้อเสียเรื่องการซ่อมบำรุงและการวางแนวของตัวรับและตัวส่งให้ตรงกัน



รูปที่ 4.13 โฟโตอิเล็กทริกเซ็นเซอร์แบบตัวรับ/ตัวส่งแยกกัน

4.1.3.2 แบบใช้แผ่นสะท้อน (Reto-Reflective)

ตัวส่งแสงกับตัวรับแสงจะอยู่ในตัวเดียวกัน แต่จะใช้แผ่นสะท้องแสงดังแสดงในรูปที่ 4.14 ตัวส่งจะส่งแสงออกไปถ้าแสงถูกส่งกลับมาจากแผ่นสะท้อนแสงซึ่งแสงส่วนใหญ่จะกลับมาที่ตัวรับ เมื่อมีวัตถุมาบังแสงระหว่างตัวส่งกับแผ่นสะท้อน ลำแสงจะไม่สามารถสะท้อนกลับมาที่ตัวรับได้ จะทำให้เซ็นเซอร์เปลี่ยนสถานะของเอาต์พุต ปัญหาที่อาจเกิดขึ้นกับเซ็นเซอร์นี้ก็คือ วัตถุที่ตรวจจับสามารถสะท้อนแสงกลับได้ดีเหมือนกับแผ่นสะท้อนแสง ปัญหานี้ถูกแก้ไขด้วยการจัดขั้วแสงที่ตัวส่งด้วยฟิลเตอร์และใช้ฟิลเตอร์ Polarized ที่ตัวรับ ส่วนตัวแผ่นสะท้อนแสงจะใช้แบบลูกบาตรขนาดเล็กอยู่ภายในเมื่อแสงมากระทบมันจะสะท้อนกลับขั้ว 90 องศา ซึ่งต่างกับแสงที่สะท้อนมาจากวัตถุแสงจะไม่กลับขั้ว จึงทำให้ตัวรับสามารถแยกแยะได้ระหว่างการสะท้อนจากวัตถุหรือจากแผ่นสะท้อนแสง

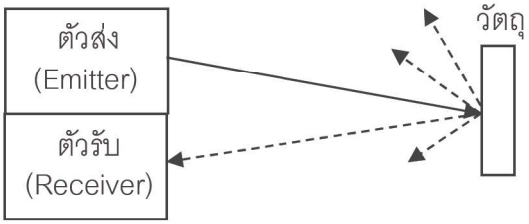


รูปที่ 4.14 โฟโตอิเล็กทริกเซ็นเซอร์แบบใช้แผ่นสะท้อน

4.1.3.3 แบบร่วมแสงหรือตรวจจับวัตถุโดยตรง (Diffuse)

ถึงแม้ว่าโฟโตอิเล็กทริกเซ็นเซอร์แบบแผ่นสะท้อนแสงจะติดตั้งง่าย แต่ยังจำเป็นต้องติดตั้งอุปกรณ์ 2 ตัว คือ เซ็นเซอร์และแผ่นสะท้อนแสง ในขณะที่แบบตรวจจับวัตถุโดยตรง (Diffuse) ไม่จำเป็นต้องใช้แผ่นสะท้อน แต่ใช้แสงโฟกัสแทนดังแสดงในรูป 4.15

เซ็นเซอร์แบบตรวจจับวัตถุโดยตรง (Diffuse) จะใช้แสงโฟกัสและสามารถตั้งค่าปรับความไวเพื่อใช้ตั้งระยะทางตรวจจับได้ ซึ่งทำให้ติดตั้งง่ายแต่ต้องมีเงื่อนไขในการใช้งานที่ควบคุมได้ เช่น วัตถุต้องไม่เป็นสีดำเพราะอาจส่งผลต่อการสะท้อนแสง

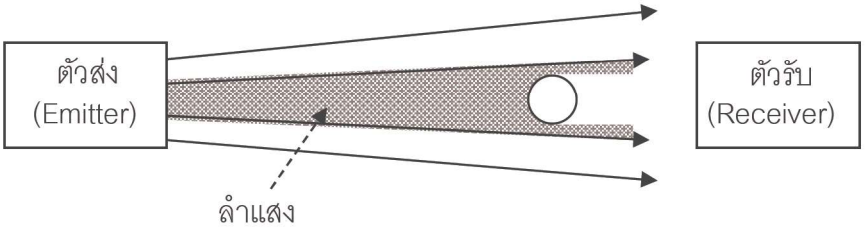


รูปที่ 4.15 โฟโตอิเล็กทริกเซ็นเซอร์แบบตรวจจับวัตถุโดยตรง(Diffuse)

ข้อควรระวังในการเลือกใช้โฟโตอิเล็กทริกเซ็นเซอร์

1) วัตถุเล็กกว่าลำแสง

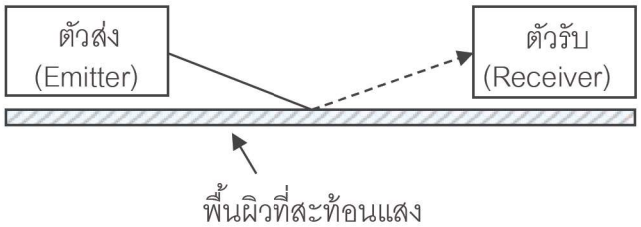
ถ้าวัตถุตรวจจับมีขนาดเล็กกว่าความกว้างของลำแสงจะทำให้มันไม่สามารถบังลำแสงได้ ดังแสดงในรูปที่ 4.16 ซึ่งจะทำให้การตรวจจับทำได้ยาก วิธีการแก้ไขปัญหานี้ คือ ต้องเลือกใช้ลำแสงที่แคบกว่าวัตถุ หรือวัตถุต้องมีขนาดใหญ่กว่าลำแสง



รูปที่ 4.16 ความสัมพันธ์ระหว่างความกว้างของลำแสงกับขนาดของวัตถุ

2) การสะท้อนแสงจากวัตถุข้างเคียง

ในการติดตั้งเซ็นเซอร์ใกล้กับวัตถุอื่นๆที่สะท้อนแสงอาจส่งผลให้เซ็นเซอร์ทำงานผิดพลาดได้



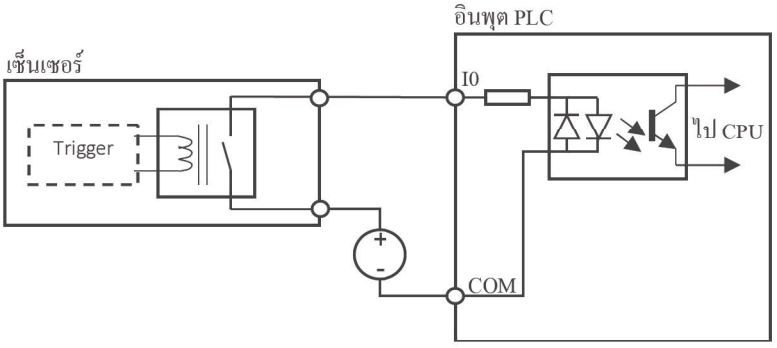
รูปที่ 4.17 แสดงการสะท้อนของลำแสงจากวัตถุข้างเคียง

4.2 ลักษณะเอาต์พุตของเซ็นเซอร์

เมื่อเซ็นเซอร์ตรวจจับวัตถุได้ มันจะส่งสัญญาณดิจิตอลให้กับ PLC โดยการสับเปลี่ยนระดับแรงดันหรือกระแสให้ ON หรือ OFF ในบางครั้งเอาต์พุตของเซ็นเซอร์จะต่อเข้ากับโหลดโดยตรง ลักษณะเอาต์พุตของเซ็นเซอร์ที่นิยมใช้งานมีดังต่อไปนี้

4.2.1 เอาต์พุตแบบสวิตช์ (Switch)

ตัวอย่างเอาต์พุตที่ง่ายที่สุดของเซ็นเซอร์ชนิดนี้ คือ สวิตช์ และ รีเลย์ ดังแสดงในรูปที่ 4.18 ในรูปแสดงหน้าคอนแทค NO ของรีเลย์ถูกต่อเข้ากับอินพุต I0 เมื่อเซ็นเซอร์ตรวจจับวัตถุได้มันจะสั่งให้รีเลย์ทำงาน ส่งผลให้กระแสไฟไหลจากขั้วบวก (V+) ผ่านเอาต์พุตของเซ็นเซอร์เข้าไปที่อินพุต I0



รูปที่ 4.18 ตัวอย่างเซ็นเซอร์แบบรีเลย์

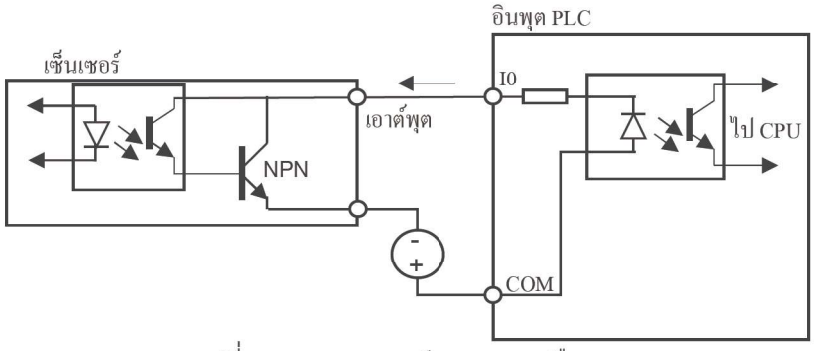
4.2.2 เอาต์พุตแบบทรานซิสเตอร์ (Transistor)

เอาต์พุตเซ็นเซอร์แบบทรานซิสเตอร์สามารถแบ่งตามการไหลของกระแสไฟเอาต์พุตได้ 2 ชนิด คือ Sinking และ Sourcing เอาต์พุตชนิด Sinking กระแสไฟจะไหลจากภายนอกเข้าสู่ขั้วเอาต์พุตของเซ็นเซอร์และผ่านไปยังขั้วคอมมอน (Common) ในขณะที่เซ็นเซอร์แบบ Sourcing กระแสไฟจากแหล่งจ่ายไฟบวกไหลผ่านออกไปที่ขั้วเอาต์พุตสู่โหลดภายนอก เอาต์พุตทั้งสองชนิด

นี้จะเน้นที่ทิศทางการไหลของกระแสไฟฟ้าไม่ใช่แรงดัน ซึ่งวิธีการกำหนดการไหลของกระแสไฟแทนแรงดันไฟจะช่วยลดปัญหาสัญญาณรบกวนได้ดีกว่า

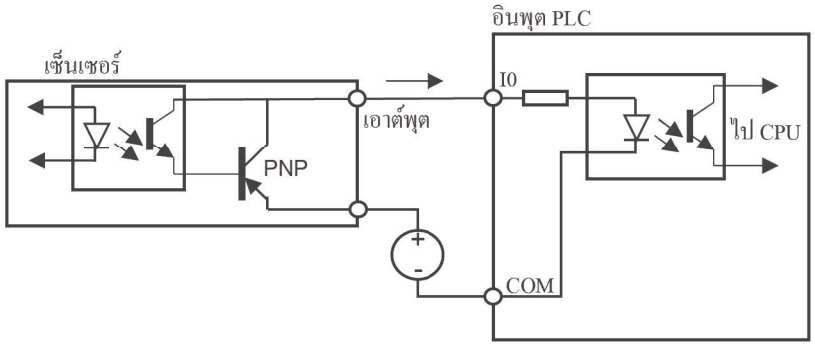
เมื่อกล่าวถึง Sinking และ Sourcing เรากำลังอ้างถึงเอาต์พุตของเซ็นเซอร์ที่ทำงานคล้ายกับสวิตช์ตัวหนึ่ง แต่ในความเป็นจริงแล้วเอาต์พุตของมัน คือ ทรานซิสเตอร์ตัวหนึ่งที่ทำหน้าที่เหมือนสวิตช์ โดยทรานซิสเตอร์ชนิด NPN ถูกใช้กับเอาต์พุต Sinking และ PNP ถูกใช้กับเอาต์พุต Sourcing บางครั้งเราอาจเรียกว่าเอาต์พุตแบบ NPN และ PNP ตามลำดับ

ตัวอย่างของเซ็นเซอร์ชนิดเอาต์พุต Sinking(NPN) แสดงในรูปที่ 4.19 เซ็นเซอร์จะใช้แหล่งจ่ายไฟภายนอกเพื่อให้มันสามารถตรวจจับวัตถุและทำงานได้ เมื่อตรวจจับวัตถุได้มันจะส่งสัญญาณกระตุ้นให้ทรานซิสเตอร์ NPN ทำงานและกระแสไฟจะไหลผ่านโหลดเข้ามาที่ขั้วเอาต์พุตของเซ็นเซอร์และครบวงจรที่ 0V



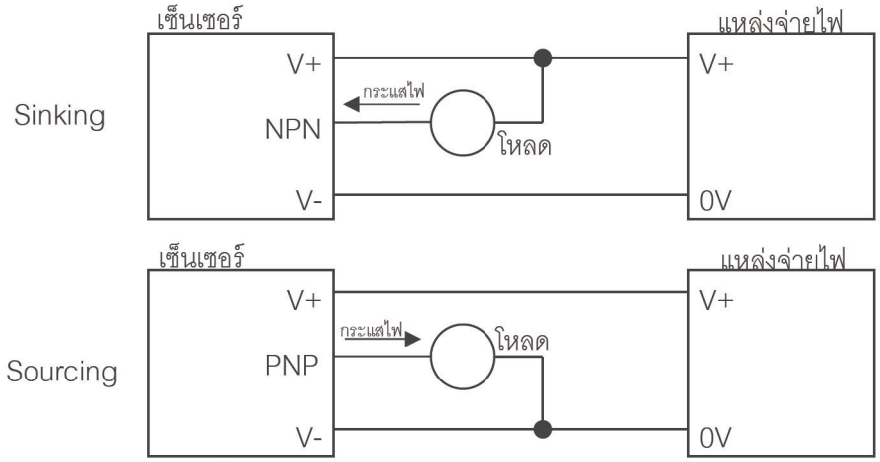
รูปที่ 4.19 วงจรเอาต์พุต NPN หรือ Sinking

ส่วนเอาต์พุตเซ็นเซอร์แบบ Sourcing แสดงในรูป 4.20 การทำงานจะตรงกันข้ามกับ Sinking เมื่อเซ็นเซอร์ตรวจจับวัตถุได้ทำให้ทรานซิสเตอร์ PNP ทำงานและกระแสไฟจะไหลผ่านขั้วเอาต์พุตออกไปด้านนอกเซ็นเซอร์ไหลผ่านโหลดและครบวงจรที่ 0V



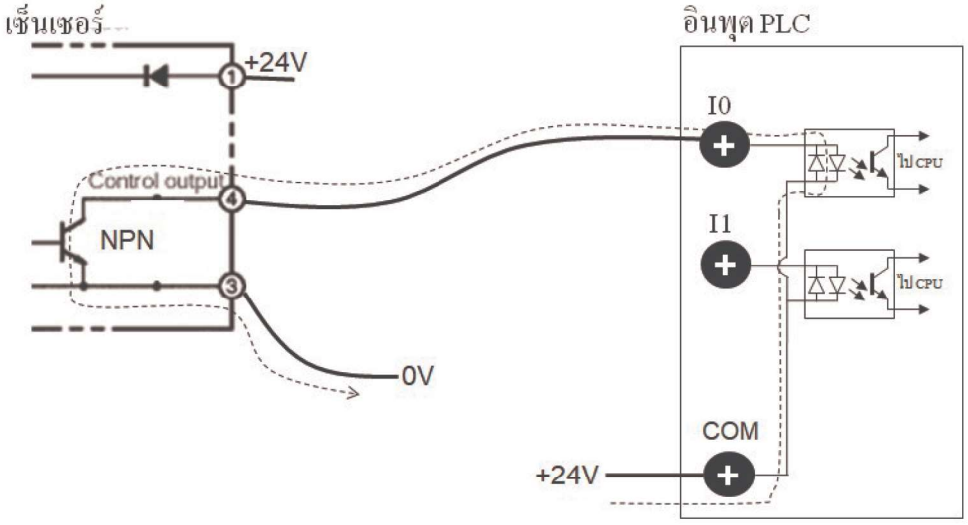
รูปที่ 4.20 วงจรเอาต์พุต PNP หรือ Sourcing

เซ็นเซอร์เอาต์พุตชนิด NPN และ PNP จะขับกระแสได้ไม่สูงมากนัก แต่ก็สามารถขับโหลดได้โดยตรง ตัวอย่างการนำไปใช้ขับโหลดหลอดไฟแสดงในรูปที่ 4.21



รูปที่ 4.21 เซ็นเซอร์แบบ NPN/PNP ที่ขับโหลดโดยตรง

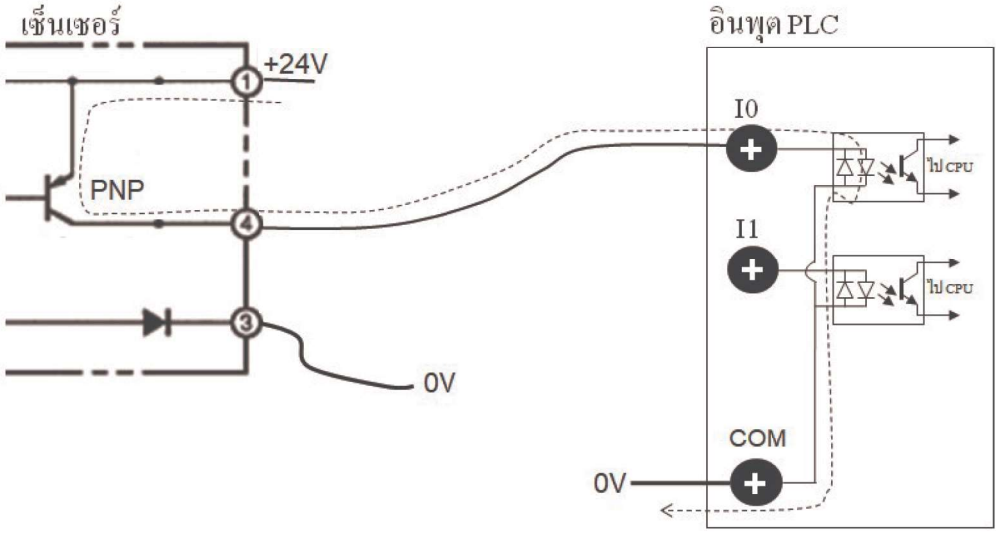
การต่อเอาต์พุตแบบ Sinking ทำได้โดยต่อโหลดเข้ากับขั้วบวกของแหล่งจ่ายไฟ และอีกด้านหนึ่งจะต่อเข้ากับเอาต์พุต NPN เมื่อเซ็นเซอร์ทำงานกระแสไฟจะไหลจากไฟบวกผ่านโหลดเข้าที่ขั้วเอาต์พุตและครบวงจรที่ 0V ในทางกลับกันเอาต์พุตแบบ Sourcing เมื่อเซ็นเซอร์ทำงานกระแสไฟจะไหลจากไฟบวกผ่านเอาต์พุตของเซ็นเซอร์ออกไปยังโหลดและครบวงจรที่ 0V



รูปที่ 4.22 โมดูลหรือการ์ดอินพุตที่ต่อเซ็นเซอร์แบบ Sinking (Common +)

จากรูป 4.22 แสดงการต่อเซ็นเซอร์ชนิด Sinking เข้ากับอินพุตของ PLC เส้นประแสดงเส้นทางการไหลของกระแสไฟขณะที่เซ็นเซอร์ทำงาน(ON) โดยกระแสจะไหลจากไฟขั้ว +24V

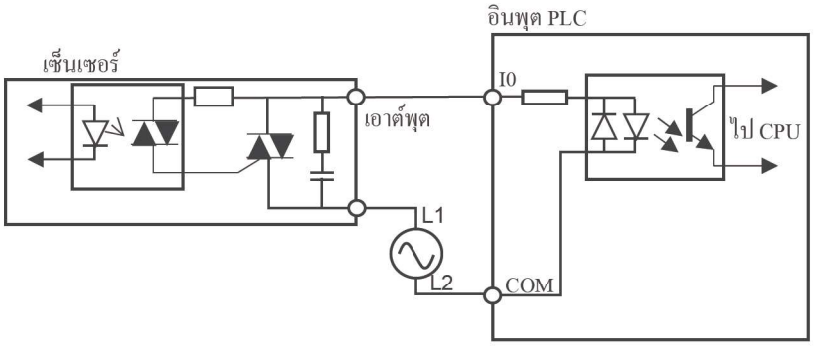
ผ่านเข้าที่ COM ของอินพุต PLC ผ่านไปยัง Optocoupler ของอินพุต PLC กระแสไฟนี้จะทำให้แสงของ Optocoupler ติด เพื่อให้หน่วยประมวลทราบว่ามีกระแสไฟไหลที่อินพุต จากนั้นกระแสจะไหลออกจากขั้วอินพุต I0 ไฟยังขั้วเอาต์พุตของเซ็นเซอร์และครบวงจรที่ 0V เมื่อเซ็นเซอร์ไม่ทำงานกระแสไฟจะไม่ไหลและแสงที่ Optocoupler จะดับ เหตุที่ต้องใช้ Optocoupler เพราะว่ามันจะช่วยป้องกันอุบัติเหตุทางไฟฟ้าจากภายนอกตัว PLC เช่น ไฟฟ้าลัดวงจร ซึ่งมันจะทำให้อินพุตพังเท่านั้นแต่ตัวประมวลผลยังคงทำงานได้ตามปกติ ส่วนการต่อเซ็นเซอร์ชนิด Souring กับอินพุต PLC แสดงในรูปที่ 4.23



รูปที่ 4.23 โมดูลหรือการ์ดอินพุตที่ต่อเซ็นเซอร์แบบ Sourcing (Common -)

4.2.3 เอาต์พุตแบบ Triac

เอาต์พุตแบบ Triac มักใช้ในการเปิดปิดไฟ AC เหมาะสำหรับโหลดที่ต้องใช้กระแสไฟมากกว่าเอาต์พุตแบบทั่วไป รูปที่ 4.24 แสดงวงจรภายในของเซ็นเซอร์ประเภทนี้



รูปที่ 4.24 เซ็นเซอร์เอาต์พุตแบบ Triac

สรุปท้ายบท

อุปกรณ์อินพุตแบบดิจิดอลที่ใช้งานในอุตสาหกรรมมีหลากหลายรูปแบบให้เลือกใช้งาน ความรู้พื้นฐานในบทนี้ทำให้คุณเข้าใจหลักการทำงานเบื้องต้นของเซ็นเซอร์เหล่านี้และสามารถเลือกใช้งานได้อย่างเหมาะสม เนื่องจากเซ็นเซอร์เป็นอุปกรณ์ภายนอกตัว PLC ที่สำคัญเป็นอันดับแรกๆ เพราะถ้าไม่มีเซ็นเซอร์เราจะไม่สามารถรับการแปรเปลี่ยนไปของกระบวนการผลิตได้ จึงทำให้ PLC ไม่สามารถควบคุมอะไรได้เลย

Chapter

5

อุปกรณ์เอาต์พุต

ในบทนี้เราจะกล่าวถึงอุปกรณ์เอาต์พุตหรืออุปกรณ์ทำงานแบบ ON/OFF ที่นิยมใช้งานในโรงงานอุตสาหกรรม ซึ่งหมายถึงอุปกรณ์ที่ทำงาน 2 ตำแหน่ง เช่น เปิดกับปิด อุปกรณ์เหล่านี้มักจะใช้พลังงานไฟฟ้าแล้วเปลี่ยนให้เป็นพลังงานกล โดยมีอยู่ด้วยกันหลายประเภทดังรายละเอียดต่อไปนี้

5.1 แมกเนติกคอนแทคเตอร์ (Magnetic Contactor)

แมกเนติกคอนแทคเตอร์ หรือแมกเนติกสวิตซ์ (Magnetic Switch) เป็นอุปกรณ์ที่ใช้ในการตัดต่อวงจรไฟฟ้าโดยการปิดเปิดของหน้าสัมผัสนั้นอาศัยอำนาจแรงแม่เหล็กไฟฟ้า สามารถนำไปประยุกต์ใช้กับวงจรควบคุมต่างๆ เช่น ควบคุมมอเตอร์ ฮีตเตอร์ เป็นต้น

แมกเนติกคอนแทคเตอร์สามารถต่อเข้าโดยตรงกับเอาต์พุตของ PLC ได้ แต่โดยทั่วไปมักใช้รีเลย์มาเป็นตัวกลางในการสั่งงาน เนื่องแมกเนติกคอนแทคเตอร์บางตัวมีขนาดใหญ่และต้องใช้กระแสไฟจ่ายคอยล์ค่อยข้างสูงการใช้รีเลย์เป็นตัวกลางจะแก้ปัญหาเอาต์พุต PLC เสียหายได้

รูปที่ 5.1 แสดงตัวอย่างแมกเนติกคอนแทคเตอร์

ส่วนประกอบของแมกเนติกคอนแทคเตอร์

1.แกนเหล็ก (Core) แบ่งเป็น 2 ส่วนดังนี้

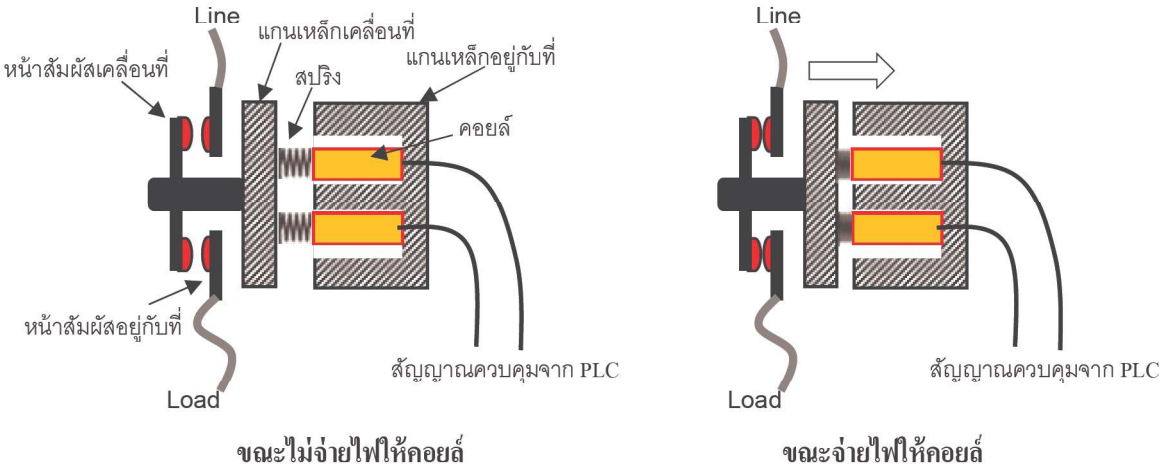
แกนเหล็กอยู่กับที่ (Fixed Core) จะมีลักษณะคล้ายรูปตัว E มีขดลวดทองแดงพันอยู่ จะมีวงแหวนฝังอยู่ที่ผิวหน้าของแกนเพื่อลดการสั่นสะเทือนของแกนเหล็กอันเนื่องมาจากไฟฟ้ากระแสสลับ วงแหวนนี้เรียกว่า เช็ดดิงริ่ง (Shading ring)

แกนเหล็กเคลื่อนที่ (Moving Core) เป็นแกนเหล็กรูปตัว E เช่นกัน แต่จะมีชุดหน้าสัมผัสเคลื่อนที่ (Moving Contact) ยึดติดอยู่ด้วย

2.คอยล์ (Coil) หรือ ขดลวดสำหรับสร้างสนามแม่เหล็กบนแกนเหล็กอยู่กับที่และจะดึงดูดแกนเหล็กเคลื่อนที่ให้เข้ามาหาแกนเหล็กอยู่กับที่

3.สปริง(Spring) ใช้สำหรับผลัก Moving Contact ออกเมื่อไม่มีกระแสไฟเลี้ยงคอยล์

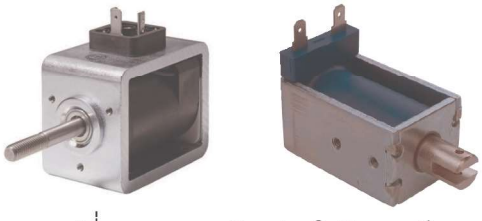
4.หน้าสัมผัส(Contact) เป็นส่วนประกอบที่ใช้ตัดต่อวงจรไฟฟ้า



รูปที่ 5.2 แสดงองค์ประกอบของแมกเนติกคอนแทคเตอร์

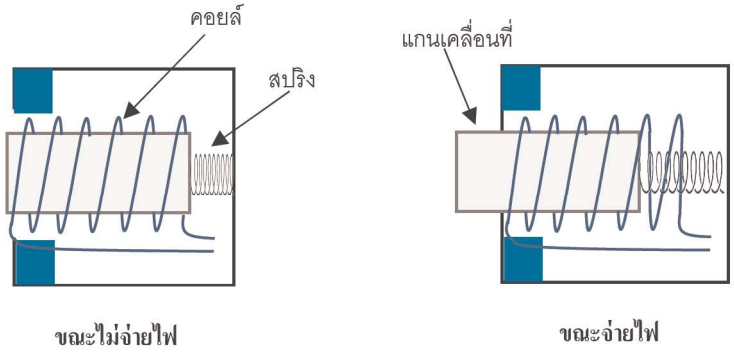
5.2 โซลินอยด์ (Solenoid)

โซลินอยด์ คือ อุปกรณ์ทำงานที่นิยมใช้งานกันทั่วไปในโรงงานอุตสาหกรรม เช่น ใช้ดัน (Reject) ชิ้นงานที่ผิดปกติออกจากกระบวนการผลิต ใช้ควบคุมวาล์ว (Valve) สำหรับควบคุมการทำงานของกระบอกสูบลมและไฮดรอลิก เป็นต้น รูปข้างล่างนี้แสดงตัวอย่างโซลินอยด์



รูปที่ 5.3 แสดงตัวอย่างโซลินอยด์

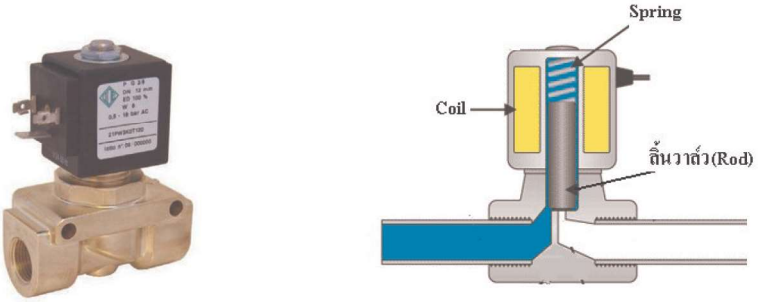
หลักการทำงานของโซลินอยด์จะมีแกนเคลื่อนที่ (Moving core) ที่สามารถเคลื่อนที่อยู่ภายในขดลวดไฟฟ้า (coil) ดังแสดงในรูปที่ 5.4 โดยปกติแกนเคลื่อนที่จะถูกยึดด้วยสปริง เมื่อจ่ายกระแสไฟฟ้าให้กับขดลวดจะสร้างสนามแม่เหล็กที่จะดูดแกนเคลื่อนที่ให้วิ่งเข้าไปภายในแกนกลางของขดลวด ซึ่งแกนเคลื่อนที่นี้จะถูกใช้เพื่อส่งแรงไปให้อุปกรณ์อื่น เช่น วาล์ว เป็นต้น ในทางกลับกันแกนเคลื่อนที่จะเคลื่อนกลับตำแหน่งเดิมเมื่อหยุดจ่ายกระแสไฟฟ้าให้กับขดลวดด้วยสปริง



รูปที่ 5.4 แสดงหลักการทำงานของโซลินอยด์

5.3 โซลินอยด์วาล์ว (Solenoid Valve)

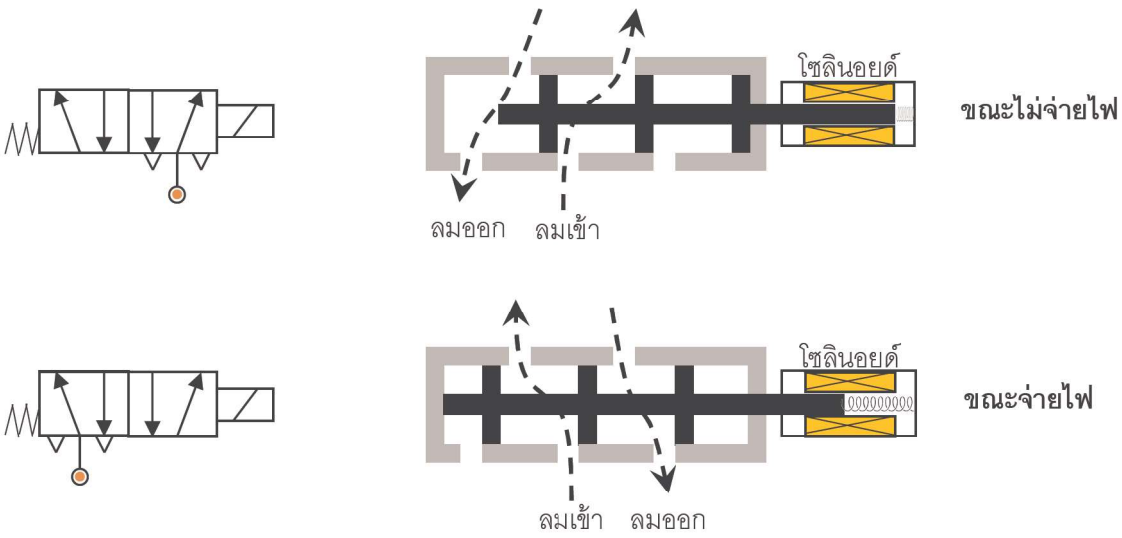
การควบคุมการไหลของอากาศและของเหลวสามารถทำได้ด้วยโซลินอยด์ที่ควบคุมการเปิดปิดของวาล์ว โดยทั่วไปเราเรียกว่า “โซลินอยด์วาล์ว” โซลินอยด์จะถูกยึดเข้ากับวาล์วเมื่อจ่ายไฟให้กับโซลินอยด์จะทำให้แกนเคลื่อนที่ของโซลินอยด์เคลื่อนไปดันลิ้นวาล์วให้เคลื่อนที่ซึ่งส่งผลให้ลิ้นวาล์วสามารถเปิดและปิดได้ตามการจ่ายกระแสไฟฟ้า



รูปที่ 5.5 แสดงตัวอย่างวาล์วพร้อมโซลินอยด์

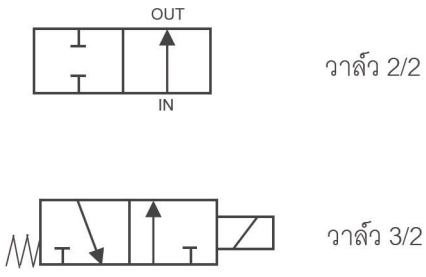
รูปที่ 5.6 แสดงภาพของวาล์วลมที่มีรู 2 รู อยู่ที่ด้านบนตัววาล์วเพื่อใช้ต่อกับอุปกรณ์อื่น เช่น กระบอกสูบ ส่วนด้านล่างของวาล์วจะมี 3 รู ใช้สำหรับต่อกับชุดจ่ายลม 1 รู และเป็นรูระบายลม 2 รู ในรูปด้านบนของรูปที่ 5.6 นั้นแสดงขณะที่ไม่ได้จ่ายไฟให้กับโซลินอยด์ซึ่งลิ้นวาล์วจะอยู่ทางด้านขวาสุด ในตำแหน่งนี้ลมจะไหลจากรูจ่ายลมส่งผ่านตัววาล์วออกทางรูด้านขวา ส่วนรูป

ด้านล่างแสดงขณะที่จ่ายไฟให้กับโซลินอยด์ลิ้นวาล์วจะเคลื่อนที่ไปทางด้านซ้ายลมในตำแหน่งนี้ลมจากรูจ่ายลมจะไหลเข้าตัววาล์วและออกทางรูด้านซ้าย (รูปทางด้านซ้ายมือเป็นไดอะแกรมที่แสดงตำแหน่งการทำงานของวาล์ว)



รูปที่ 5.6 แสดงทำงานของวาล์ว 4/2

วาล์วมีหลายชนิดซึ่งจะแบ่งตามจำนวนรูเข้าและรูออก โดยจะมีคำเรียกมาตรฐานว่า เป็นวาล์ว 2 ทาง และวาล์ว 3 ทาง เป็นต้น นอกจากนั้นยังมีการกำหนดสภาวะปกติปิดNC/ปกติเปิด (NO) ซึ่งใช้แสดงว่าวาล์วนั้นอยู่ในสภาวะใดเมื่อยังไม่ทำงานหรือยังไม่ได้จ่ายไฟ ซึ่งคล้ายกับหน้าคอนแทคไฟฟ้าที่มี NO และ NC แทนตำแหน่งของหน้าคอนแทคในขณะที่ยังไม่ได้จ่ายไฟ ตามรูปข้างล่างนี้ เป็นตัวอย่างวาล์วชนิด 2/2 และ 3/2



รูปที่ 5.7 แสดงตัวอย่างวาล์ว 2/2 และ 3/2

5.4 มอเตอร์ (Motor)

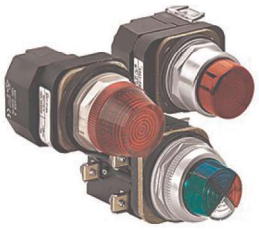
มอเตอร์เป็นอุปกรณ์ทำงานที่นิยมใช้มากอีกประเภทหนึ่งในโรงงานอุตสาหกรรม เช่น อินดักชั่นมอเตอร์ สเต็ปปิ้งมอเตอร์ และเซอร์โวมอเตอร์ เป็นต้น ในที่นี้จะกล่าวถึงเฉพาะอินดักชั่นมอเตอร์เท่านั้น โดยทั่วไปการควบคุมการทำงานของอินดักชั่นมอเตอร์มักจะใช้แมกเนติคอนแทคเตอร์หรือมอเตอร์สตาร์ทเตอร์เพื่อจ่ายไฟให้กับมอเตอร์ การควบคุมมักจะสั่งให้มอเตอร์หมุนและหยุดหมุนเป็นหลัก บางครั้งอาจมีการปรับความเร็วรอบโดยการใช้อินเวอร์เตอร์ รูปข้างล่างนี้แสดงตัวอย่างอินดักชั่นมอเตอร์ที่ใช้ในอุตสาหกรรม

รูปที่ 5.8 อินดักชั่นมอเตอร์

5.5 อุปกรณ์ทำงานประเภทอื่น ๆ

นอกจากอุปกรณ์ที่กล่าวมาข้างต้น ยังมีอุปกรณ์ทำงานประเภทอื่น ๆ ที่ใช้งานกันทั่วไป ดังต่อไปนี้

หลอดไฟ มีใช้กับเครื่องจักรเกือบทุกเครื่อง ใช้สำหรับแสดงสถานะของเครื่องจักรและส่งสัญญาณต่างๆ ให้ผู้ปฏิบัติงานรับรู้ หลอดไฟส่วนใหญ่มักกินกระแสไฟต่ำจึงต่อเข้ากับเอาต์พุตของ PLC ได้โดยตรง



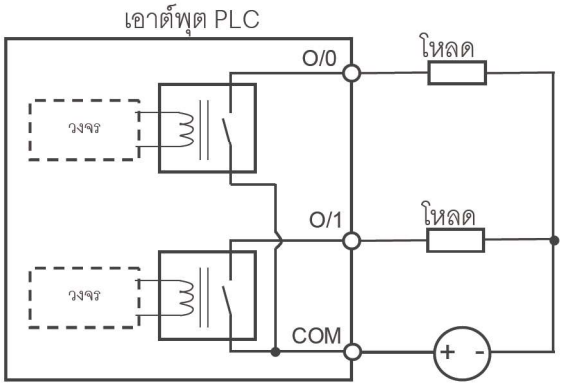
ฮีตเตอร์ เป็นอุปกรณ์ให้ความร้อนที่มักจะถูกควบคุมผ่านรีเลย์หรือแมกเนติกคอนแทคเตอร์

5.6 การใช้งานอุปกรณ์เอาต์พุตร่วมกับ PLC

เอาต์พุตของ PLC จะทำหน้าที่ตัดต่อไฟที่จ่ายให้อุปกรณ์ภายนอก โดยปกติโมดูลเอาต์พุตจะมีหลายชนิดให้เลือกใช้งาน เช่น รีเลย์ ทรานซิสเตอร์ และไตรแอก (Triac) เป็นต้น เวลาสั่งซื้อจะต้องเลือกว่าเป็นเอาต์พุตชนิดใด เช่น รีเลย์ ทรานซิสเตอร์ หรือไตรแอก ถ้าเป็น PLC ชนิด RACK หรือโมดูลล่าร์ จะมีให้เลือกใช้งานหลายขนาด เช่น 8 จุด, 16 จุด หรือ 32 จุด เป็นต้น

5.6.1 เอาต์พุตรีเลย์

เอาต์พุตแบบรีเลย์บางครั้งเรียกว่า “Dry Contact” จะมีความยืดหยุ่นสูงกว่าเพราะใช้ได้ทั้งไฟ AC และ DC แต่มันทำงานช้ากว่าทรานซิสเตอร์นอกจากนั้นหน้าสัมผัสของรีเลย์จะสึกหรอและเสื่อมสภาพตามการใช้งาน



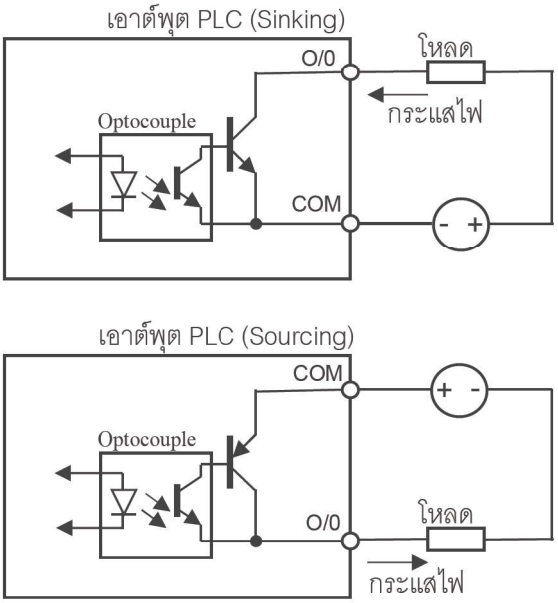
รูปที่ 5.9 แสดงวงจรเอาต์พุตรีเลย์ของ PLC

ในกรณี PLC ขนาดเล็กที่มีเอาต์พุตในตัวอาจมีเอาต์พุตมากกว่าหนึ่งกลุ่มและมีขั้วต่อคอมมอน (COM) แยกกันสำหรับเอาต์พุตกลุ่มนั้นๆ สิ่งที่ควรระวังคือการใช้ไฟ AC และไฟ DC ร่วมกันในวงจรเอาต์พุตซึ่งต้องแยกคอมมอน (COM) ให้ชัดเจน ไม่เช่นนั้นอาจเกิดการลัดวงจรเกิดขึ้นสร้างความเสียหายให้กับ PLC และแหล่งจ่ายไฟได้

5.6.2 เอาต์พุตทรานซิสเตอร์

เอาต์พุตทรานซิสเตอร์มีอยู่ด้วยกัน 2 ชนิด คือ NPN (Sinking) และ PNP (Sourcing) ซึ่งจะสังเกตุว่าเป็น Sinking หรือ Sourcing จากการไหลของกระไฟที่ขั้วเอาต์พุตถ้าไหลเข้าขั้วเอาต์พุตจะเป็น Sinking ส่วนกระแสไฟไหลออกจากขั้วเอาต์พุตจะเป็น Sourcing

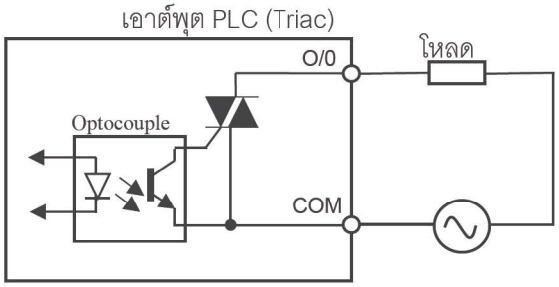
บางครั้งเราก็เรียกตามการต่อใช้งานโดยชนิด Sinking จะเรียกว่า Common (+) เพราะว่าขั้วบวกของแหล่งจ่ายไฟจะต่อเข้ากับขั้วคอมมอน (COM) ของเอาต์พุต PLC ส่วนชนิด Sourcing จะเรียกว่าCommon (-) เพราะว่าขั้วลบของแหล่งจ่ายไฟจะต่อเข้ากับขั้วคอมมอน (COM) ของเอาต์พุต PLC



รูปที่ 5.10 แสดงการต่อวงจรเอาต์พุตของ PLC ชนิด Sinking และ Sourcing

5.6.3 เอาต์พุตไตรแอก (Triac)

เอาต์พุตไตรแอกเป็นเอาต์พุตประเภทโซลิดสเตท(Solid state) ที่ทำจากสารกึ่งตัวนำ เช่นเดียวกับทรานซิสเตอร์ ปกติจะใช้กับโหลดไฟ AC การทำงานจะเร็วและมีอายุการใช้งานที่ยาวนานกว่าเอาต์พุตรีเลย์เนื่องจากไม่มีชิ้นส่วนที่เคลื่อนที่ การต่อเอาต์พุตไตรแอกแสดงดังรูปข้างล่างนี้



รูปที่ 5.11 แสดงการต่อเอาต์พุตของ PLC ชนิด Triac

สรุปท้ายบท

โซลินอยด์เป็นอุปกรณ์เอาต์พุตที่เปลี่ยนกระแสไฟฟ้าให้เป็นการเคลื่อนที่แนวเส้นตรงมักใช้ร่วมกับวาล์ว นอกจากนี้ยังมีอุปกรณ์เอาต์พุตประเภทอื่นๆดังที่กล่าวมาแล้วข้างต้น การเลือกอุปกรณ์เหล่านี้ต้องเลือกให้เหมาะสมกับประเภทของเอาต์พุต PLC ด้วยจึงจะให้เกิดประสิทธิภาพและอายุการใช้งานสูงสุด

Chapter

6

การออกแบบและติดตั้ง

การออกแบบและติดตั้งนั้นมีผลต่อการทำงานของระบบควบคุมค่อนข้างมากซึ่งบางท่านอาจมองข้าม เป็นเรื่องปกติที่ผู้ใช้งาน PLC มักไม่ได้ให้ความสำคัญกับการออกแบบวงจรไฟฟ้าและการติดตั้งที่ถูกต้องเหมาะสมกับระบบควบคุม ถ้าการติดตั้งไม่ดีพอมันอาจก่อปัญหาให้กับระบบและอุปกรณ์ต่างๆ ได้ ในบทนี้เราจะกล่าวถึงรายละเอียดดังกล่าวให้กระจ่างมากขึ้น

6.1 อุปกรณ์ประกอบระบบควบคุม PLC

การติดตั้ง PLC สำหรับระบบควบคุมอัตโนมัติต้องมีองค์ประกอบหลายอย่างด้วยกัน รายการข้างล่างนี้คือตัวอย่างอุปกรณ์ที่ติดตั้งในตู้ควบคุม PLC

1.Programable Logic Controller (PLC) มันคืออุปกรณ์ที่สำคัญที่สุดเปรียบเป็นสมองในตู้ควบคุม

2.เซอร์กิตเบรกเกอร์ ทำหน้าที่เปิดปิดไฟให้กับระบบและตัดไฟเมื่อเกิดกระแสไฟลัดวงจรหรือเกินพิกัด

3.หม้อแปลงไฟฟ้า ใช้สำหรับลดระดับแรงดันไฟ AC ให้ต่ำลงและยังช่วยลดไฟกระชาก (Surge) จากแหล่งจ่ายไฟภายนอกได้อีกด้วย ปกติจะนิยมใช้หม้อแปลงแบบ Isolated

4.เทอร์มินอล ใช้สำหรับต่อกับอุปกรณ์ต่างๆ ภายในและภายนอกตู้ควบคุม เช่น เซ็นเซอร์ โซลินอย์วาล์ว เป็นต้น

5.ระบบกราวด์ เป็นระบบความปลอดภัยที่สำคัญต่อผู้ใช้งานโดยใช้เป็นทางเดินของกระแสไฟฟ้าเมื่อเกิดการขัดข้องหรือไฟฟ้ารั่ว

6.แมกเนติกคอนแทคเตอร์ ใช้สำหรับตัดแหล่งจ่ายไฟเข้ามอเตอร์

7.แหล่งจ่ายไฟ DC เป็นตัวแปลงไฟ AC ไปเป็น DC โดยทั่วไปจะแปลงเป็นไฟ 24 Vdc เนื่องจากมีความปลอดภัยมากกว่าการใช้ไฟ AC นอกจากนั้นอุปกรณ์อินพุตเอาต์พุตจะใช้พลังงานจากแหล่งจ่ายไฟ 24Vdc

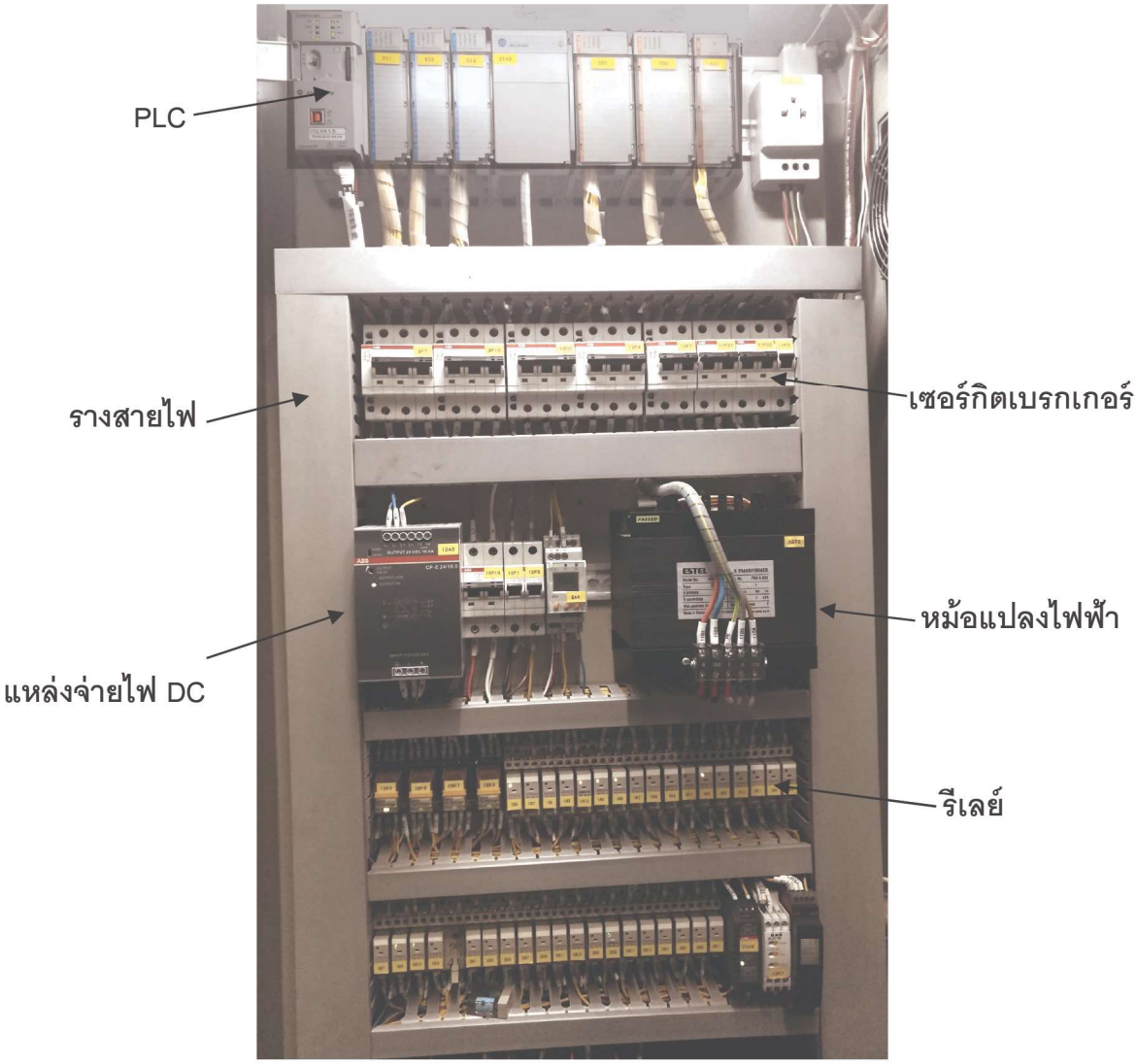
8.ตู้ควบคุม สำหรับป้องกันอุปกรณ์และผู้ใช้งานจากอุบัติเหตุที่อาจเกิดขึ้นได้

9.รางสายไฟ(Wire Duct) ปกติจะวางเป็นแถวตามแนวอุปกรณ์และรอบๆตู้ ใช้วางสายไฟที่ต่อถึงกันภายในตู้

10.ฟิวส์(Fuses) ทำหน้าคล้ายกับเซอร์กิตเบรกเกอร์ในการป้องกันโหลดที่ใช้กระแสไฟมากเกินไปแต่ใช้ได้เพียงครั้งเดียวเท่านั้น ไม่ได้หมายความว่าเซอร์กิตเบรกเกอร์ดีกว่าที่มันสามารถรีเซ็ทและใช้งานใหม่ได้แต่ละอย่างมีความเหมาะสมในการใช้งานไม่เหมือนกัน

11.รีเลย์ การต่อเอาต์พุต PLC เข้ากับโหลดโดยตรงอาจส่งผลให้เอาต์พุตเกิดความเสียหายได้ ถ้าโหลดที่ใช้กระแสไฟมากควรใช้รีเลย์เป็นตัวไปขับโหลดแทน

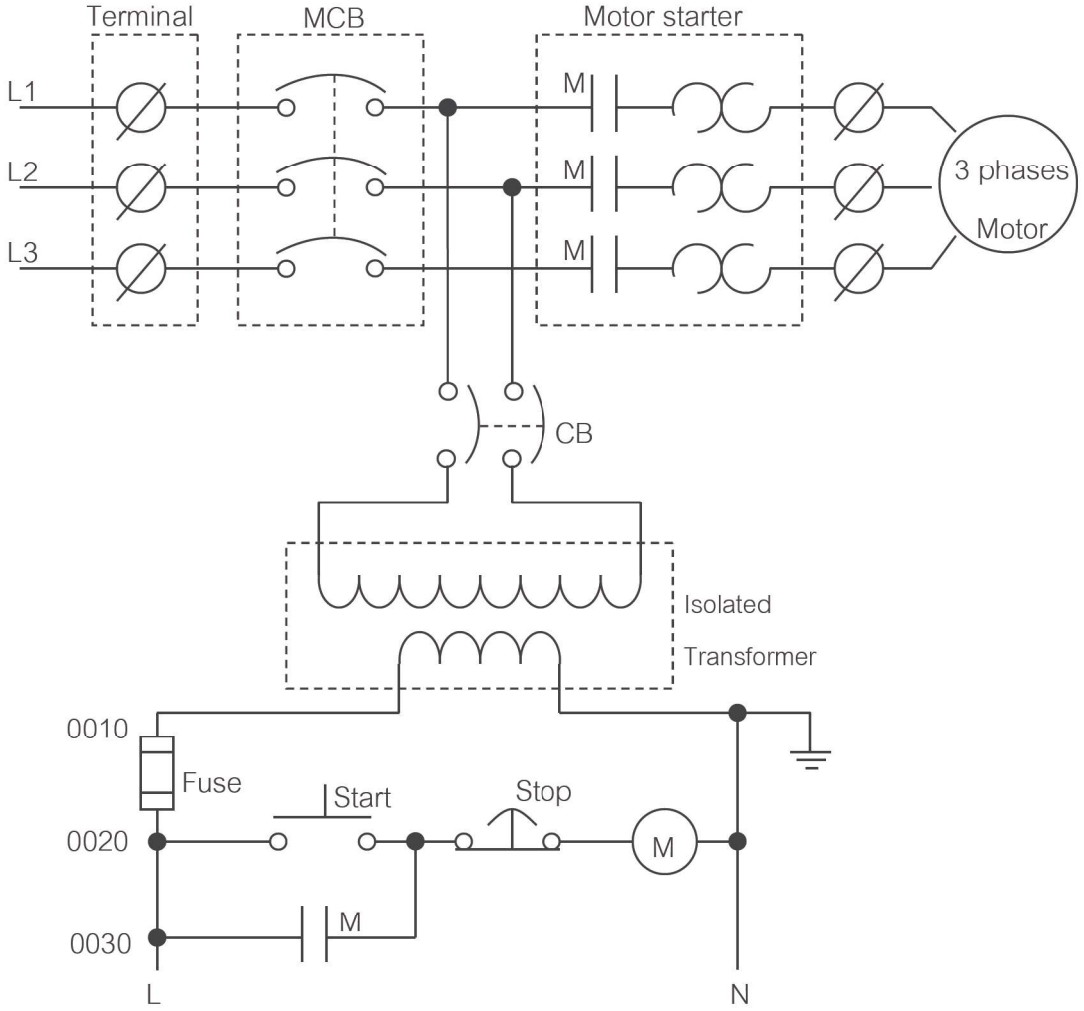
12.ราง DIN (DIN Rail) เป็นรางที่ทำจากโลหะใช้สำหรับยึดอุปกรณ์ควบคุมต่างๆ คำว่า DIN ย่อมาจาก Deutsches Institut für Normung ซึ่งเป็นผู้กำหนดคุณสมบัติในครั้งแรกที่เริ่มใช้



รูป 6.1 แสดงตัวอย่างอุปกรณ์ที่อยู่ในตู้ควบคุม

โดยปกติระบบควบคุมจะใช้ไฟ AC และ DC ที่ระดับแรงดันต่างกัน บ่อยครั้งที่ตู้ควบคุมจะถูกจ่ายด้วยไฟหนึ่งเฟส 220V หรือสามเฟส 380V แหล่งจ่ายไฟเหล่านี้มักถูกปรับให้แรงดันต่ำลงสำหรับระบบควบคุมและสำหรับแหล่งจ่ายไฟ DC นอกจากนั้นระบบไฟในตู้ควบคุมจะถูกจ่ายให้กับอุปกรณ์ทำงานต่างๆ ด้วย เช่น มอเตอร์ไฟฟ้า ดังนั้น ภายในตู้ควบคุมจึงมีทั้งระบบไฟควบคุมและระบบไฟกำลังอยู่ด้วยกัน

ตัวอย่างของแผนผังการเดินสายไฟสำหรับควบคุมมอเตอร์แสดงในรูป 6.2 เส้นประแสดงว่าเป็นอุปกรณ์ชุดเดียวกัน ระบบนี้ใช้ไฟ 3 เฟส (L1, L2, L3) ต่อเข้ากับเทอร์มินอลเพื่อต่อเข้ากับตัวตัดไฟ (Power circuit breaker) จากนั้นไฟ 3 เฟสถูกต่อเข้ากับมอเตอร์สตาร์ทเตอร์ (Motor starter) จากนั้นต่อเข้ากับมอเตอร์ 3 เฟส นอกจากนั้นแหล่งจ่ายไฟจะถูกต่อเข้ากับหม้อแปลงไฟเพื่อลดระดับแรงดันไฟเป็น 220V สำหรับจ่ายให้ระบบควบคุมโดยการแยกเป็นไลน์ (line) และนิวตรอน (Neutral) ซึ่งนิวตรอนจะต่อลงกราวด์ ในไดอะแกรมจะมีสวิทช์ 2 ตัว สำหรับสั่งให้มอเตอร์ทำงานและหยุดทำงาน



รูปที่ 6.2 สคีมเมติกไดอะแกรม (Schematic)ควบคุมมอเตอร์ มาตรฐาน ANSI

จากไดอะแกรมข้างบนจะมีหมายเลขกำกับตำแหน่งของสายไฟของอุปกรณ์ ซึ่งเป็นสิ่งสำคัญของระบบควบคุมอุตสาหกรรมเนื่องจากมีสายจำนวนมาก การระบุหมายเลขจะให้การติดตั้งและซ่อมบำรุงทำได้ง่ายขึ้น

6.2 สิ่งที่ต้องพิจารณาในการติดตั้ง PLC

การใช้งานส่วนใหญ่ต้องติดตั้งในตู้อุตสาหกรรม เพื่อลดผลกระทบของสัญญาณรบกวนทางไฟฟ้าและการสัมผัสสิ่งแวดล้อม ควรติดตั้ง PLC ให้อยู่ห่างจากสายไฟกำลัง และ แหล่งที่มาของสัญญาณรบกวนทางไฟฟ้าอื่นๆ เช่น รีเลย์และมอเตอร์ไฟฟ้า AC เป็นต้น นอกจากนั้นเรายังต้องพิจารณาและพึงระวังในการติดตั้ง PLC ดังรายละเอียดต่อไปนี้

6.2.1 ระบบกราวด์ (Grounding)

กราวด์ (Ground) คือ ระบบไฟฟ้าที่ต้องการหาจุดแรงดันไฟฟ้าอ้างอิงโดยการฝังโลหะลงในดิน ตัวอย่างระบบกราวด์ที่พบเห็นได้บ่อย เช่น ปลั๊กไฟต่าง ๆ ในบ้านหรืออาคารจะถูกต่อลงกราวด์หรือบางทีเรียกว่าสายดิน เมื่อมีไฟฟ้ารั่วเกิดขึ้นกระแสไฟที่รั่วนั้นจะไหลลงดิน

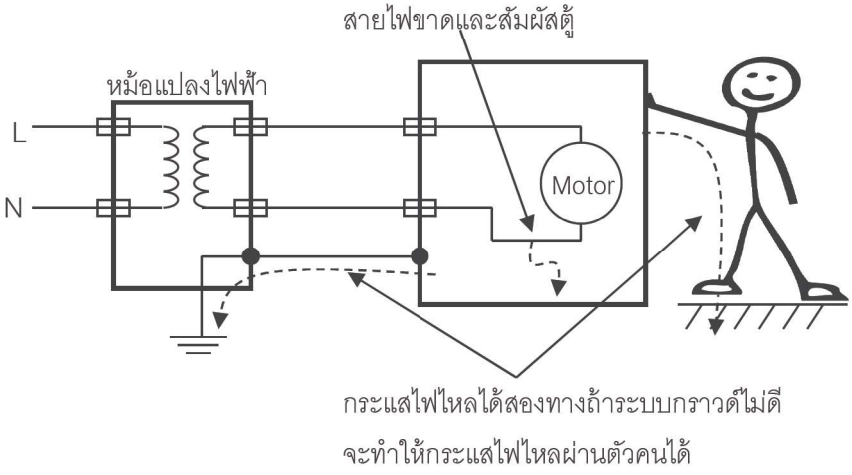
เหตุผลสำคัญสำหรับการต่อกราวด์ก็เพื่อความปลอดภัยของคน การไหลของกระแสไฟฟ้าผ่านร่างกายคนอาจมีผลต่อการทำงานของอวัยวะต่างๆ โดยเฉพาะอย่างยิ่งหัวใจ ตารางที่ 6.1 แสดงระดับของกระแสไฟฟ้าที่แตกต่างกันและผลกระทบต่อร่างกาย กระแสไฟฟ้าที่ไหลจะขึ้นอยู่กับความต้านทานของร่างกาย และจุดสัมผัส

ตารางที่ 6.1 แสดงระดับของกระแสไฟฟ้าและผลกระทบต่อร่างกาย

ปริมาณกระแสไฟฟ้า	ผลกระทบที่มีต่อร่างกาย
1 mA หรือ น้อยกว่า	ไม่มีผลกระทบต่อร่างกาย
มากกว่า 5 mA	ทำให้เกิดการช็อก และเกิดความเจ็บปวด
มากกว่า 15 mA	กล้ามเนื้อบริเวณที่ถูกกระแสไฟฟ้าดูดเกิดการหดตัว และร่างกายจะเกิดอาการเกร็ง
มากกว่า 30 mA	การหายใจติดขัด และสามารถทำให้หมดสติได้
50 ถึง 200 mA	ขาดเลือดไปเลี้ยงหัวใจ และอาจจะเสียชีวิตได้ภายในเวลาไม่กี่วินาที
มากกว่า 200 mA	เกิดการไหม้บริเวณผิวหนังที่ถูกกระแสไฟฟ้าดูด และหัวใจจะหยุดเต้นภายในเวลารวดเร็ว
ตั้งแต่ 1A ขึ้นไป	ผิวหนังบริเวณที่ถูกกระแสไฟฟ้าดูดถูกทำลายอย่างถาวร และหัวใจจะหยุดเต้นภายในเวลาไม่กี่วินาที

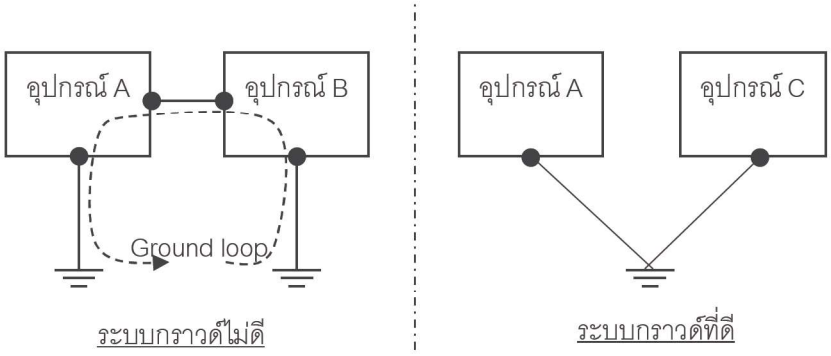
ในรูปที่ 6.3 แสดงระบบกราวด์ที่ตู้ควบคุมทำด้วยโลหะ ด้านซ้ายมือจะมีตู้ที่บรรจุหม้อแปลงไฟฟ้าอยู่ซึ่งตู้ต่อลงดินโดยตรง ส่วนตู้ที่ 2 จะมีโหลดมอเตอร์อยู่และต่อสายกราวด์เข้ากับตู้

แรก ถ้ามี Fault เกิดขึ้นในระบบอาจทำให้สายไฟรั่วและสัมผัสกับตู้ ถ้าตู้ไม่ได้ต่อลงกราวด์และมีคนมาสัมผัสตู้อาจถูกไฟดูดได้ ถ้าตู้ถูกต่อลงกราวด์ความต้านทานระหว่างตู้กับกราวด์จะต่ำมากๆ (ประมาณ 1 โอห์ม) แต่ความต้านทานที่ตัวคนจะสูงกว่ากราวด์ ดังนั้นกระแสจะไหลลงกราวด์มากกว่าแล้วจะทำให้ระบบตัดไฟออก เช่น ฟิวส์ขาด หรือเซอร์กิตเบกเกอร์ตัดวงจร



รูปที่ 6.3 ระบบกราวด์เพื่อความปลอดภัย

ถ้าการต่อกราวด์ทำได้ไม่ดีเพียงพอจะทำให้ระบบสูญเสียความปลอดภัยได้ เช่น อุปกรณ์แต่ละตัวต่อลงกราวด์โดยตรงในขณะเดียวกันอุปกรณ์แต่ละตัวอาจเชื่อมถึงกันโดยรางสายไฟหรือท่อร้อยสายไฟที่ทำจากโลหะซึ่งจะทำให้เกิดกราวด์ลูป (Ground loop) ขึ้น รูปที่ 6.4 ด้านซ้ายมือแสดงให้เห็นกราวด์ลูปที่เกิดขึ้นเพราะมีการต่อกราวด์ระหว่างอุปกรณ์ A และอุปกรณ์ B ในขณะที่อุปกรณ์ทั้งสองก็ต่อลงกราวด์ด้วยซึ่งการต่อแบบนี้จะสร้างลูปขึ้น ถ้ามีกระแสไฟรั่วลูปนี้จะทำให้มีระดับแรงดันแตกต่างกันที่จุดต่างๆ การต่อกราวด์ที่ถูกต้องควรต่อเป็นแบบ T (Tree) ซึ่งกราวด์ของอุปกรณ์ A และB ควรต่อเข้ากับกราวด์ของระบบจ่ายไฟดังแสดงในรูปที่ 6.4 ด้านขวามือ



รูป 6.4 แสดงระบบกราวด์ที่ทำให้เกิดกราวด์ลูป

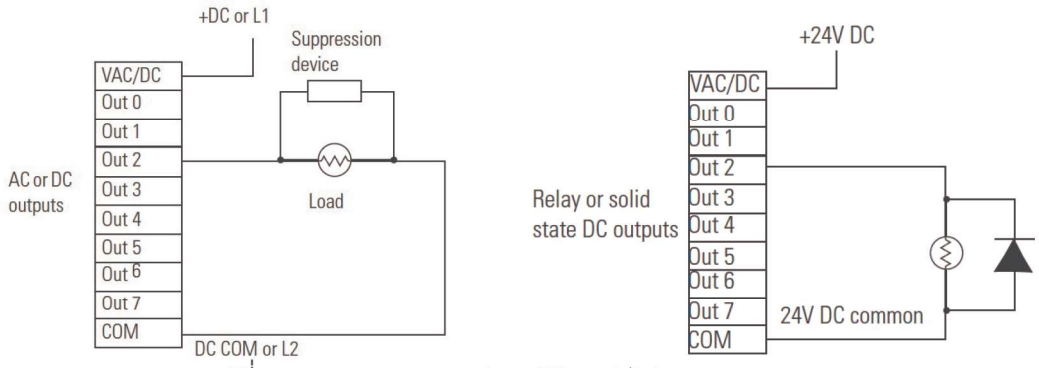
บ่อยครั้งปัญหานี้มักเกิดจากระบบจ่ายไฟหลักของโรงงาน เพราะมันมีจุดต่อกราวด์หลายจุดภายในอาคารหรือแต่ละอาคารอาจมีกราวด์แยกกัน จึงเป็นเหตุให้มีกระแสไหลวนอยู่บนสายกราวด์ กระแสไฟนี้จะทำให้เกิดแรงดันไฟฟ้าที่แตกต่างกันไปตามจุดต่างๆบนสายกราวด์ ปัญหานี้สามารถแก้ไขได้ด้วยการแยกระบบไฟฟ้าออกจากกัน ดังนั้นเมื่อท่านออกแบบและสร้างระบบควบคุมไฟฟ้าจุดที่ต้องพึงระมัดระวังมีดังนี้

- หลีกเลี่ยงกราวด์ลูป
  - ควรต่อโครงโลหะต่างๆ เข้ากับกราวด์บัส (Ground Bus)
  - อุปกรณ์ต่างๆ ของ PLC ควรต่อกราวด์เข้ากับโครงของ PLC หลัก
  - สายกราวด์ภายในตู้ควรเดินแยกจากสายไฟฟ้ากำลัง เช่น สายไฟมอเตอร์
  - ต่อกราวด์ของเครื่องจักรเข้ากับกราวด์ของตู้เพียงจุดเดียว
- ตรวจสอบคุณภาพของการเชื่อมต่อทางไฟฟ้าให้อยู่ในสภาพที่ดี
  - ควรใช้แหวนรองน็อตที่จุดต่อต่างๆ เพื่อไม่ให้เกิดการคลายตัว
  - ยึดสายกราวด์กับโลหะที่แบนราบและควรขัดสีที่เคลือบออกด้วย(ถ้ามี)
  - ตรวจสอบความต้านทานของกราวด์ ควรต่ำกว่า 0.1 โอห์มจึงจะดี

6.2.2 อุปกรณ์ป้องกันไฟกระชาก (Suppressors)

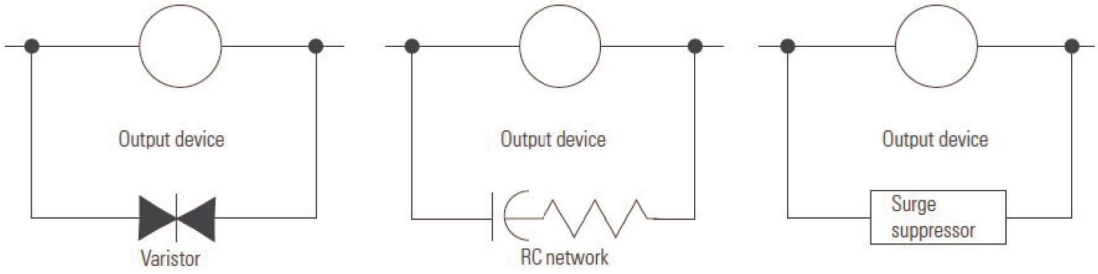
โหลดประเภทอินดักทีฟ (Inductive) หลายอย่างมักทำให้เกิดการสปากค์ (Spark) เช่น โซลินอยด์ และมอเตอร์ขนาดใหญ่ ซึ่งอุปกรณ์เหล่านี้อาจสร้างปัญหาให้กับระบบควบคุม ต้นเหตุของปัญหา คือ ขดลวดไฟฟ้าที่ทำหน้าที่เป็นอินดักเตอร์ (Inductor) เมื่อเราจ่ายกระแสไฟให้กับขดลวดมันจะสร้างสนามแม่เหล็กและสะสมพลังงานเอาไว้ เมื่อเราหยุดจ่ายไฟให้ขดลวดสนามแม่เหล็กจะลดลงในขณะเดียวกันพลังงานที่สะสมไว้จะจ่ายย้อนกลับออกมา ดังนั้น โหลดประเภทอินดักทีฟ(Inductive) จะกินกระแสไฟสูงขณะเริ่มจ่ายไฟและทำให้เกิดไฟกระชากขณะหยุดจ่ายไฟ ยิ่งโหลดอินดักทีฟขนาดใหญ่จะทำให้เกิดแรงดันกระชากที่รุนแรงและอาจทำให้อุปกรณ์อื่นๆในระบบเกิดความเสียหายได้

อุปกรณ์ป้องกันไฟกระชาก (Surge suppressor) อาจถูกใช้เพื่อป้องกันอุปกรณ์อื่นในระบบควบคุมจากไฟกระชากที่เกิดจากโหลดอินดักทีฟได้ รูปที่ 6.5 แสดงวงจรที่ใช้เพื่อช่วยลดไฟกระชากในระบบ ส่วนในรูปซ้ายมือเป็นวงจรเอาท์พุต PLC ที่ต่อเข้ากับโหลดอินดักทีฟโดยมีวงจรป้องกันไฟกระชาก



รูปที่ 6.5 แสดงการต่ออุปกรณ์ป้องกันไฟกระชาก

เนื่องจากในรูปซ้ายมือแหล่งจ่ายไฟอาจเป็นไฟ AC หรือ DC ถ้าใช้วงจร RC เป็น Surge suppressor มันจะทำงานเหมือนตัวกรองความถี่สูงที่จะยอมให้แรงดันสไปท์ (Spike) ความถี่สูงไหลผ่านวงจรไปได้จึงทำให้แรงดันสไปท์ (Spike) เหล่านั้นไหลผ่านวงจร RC แทนที่จะไหลกลับไปที่เอาท์พุตของ PLC วงจร Surge suppressor บางประเภทอาจใช้ Varistor ก็ได้



ส่วนรูปที่ 6.5 ขวามือเป็นวงจรที่มีวงจรป้องกันไฟกระชากโดยใช้ไดโอด (Diode) เหมาะสำหรับโหลดไฟ DC ไดโอดจะยอมให้กระแสไฟไหลผ่านตัวมันเมื่อโหลดอินดักทีฟคลายพลังงานไฟฟ้าออกมาแทนที่จะไหลกลับไปที่เอาต์พุต

6.2.3 ข้อพิจารณาในการติดตั้งและเลือกใช้ตู้ควบคุม

ถึงแม้ว่า PLC จะถูกออกแบบมาเพื่อใช้ในงานอุตสาหกรรม แต่มันก็มีโอกาสเสียหายได้ง่ายเมื่อติดตั้งอยู่ในโรงงาน ดังนั้นตู้ควบคุมจะถูกใช้เพื่อป้องกัน PLC จากสิ่งแวดล้อมต่างๆ ตัวอย่างของสิ่งแวดล้อมที่อาจส่งผลกระทบต่อ PLC มีดังนี้

1) ความสกปรก เช่น ฝุ่นผงและสารเคมีต่างๆ สามารถเข้าไปในตัว PLC ได้โดยผ่านระบบระบายความร้อน ฝุ่นจะจับตัวที่แผงวงจรภายใน PLC ซึ่งอาจส่งผลกระทบต่อการทำงานดังนั้นตู้ควบคุมควรป้องกันได้ เช่น ตู้ควบคุมที่มีมาตรฐาน NEMA 4 หรือ NEMA12 จะสามารถป้องกันฝุ่นได้

2) ความชื้น มักไม่ค่อยเป็นปัญหากับวัสดุต่างๆ แต่มันอาจสะสมจนกลายเป็นน้ำ ซึ่งอาจทำให้เกิดสนิมที่สามารถนำไฟฟ้าได้และเกิดลัดวงจรในที่สุด
3) อุณหภูมิแวดล้อม อุปกรณ์สารกึ่งตัวนำต่างๆ ใน PLC จะมีย่านอุณหภูมิที่เหมาะสมในการทำงาน ถ้าอุณหภูมิไม่อยู่ในย่านที่ผู้ผลิตกำหนด PLC อาจหยุดการทำงานได้ ในสถานที่ติดตั้งที่มีอุณหภูมิต่ำกว่าย่านต่ำสุด เช่น 0 องศาเซลเซียส อาจต้องติดตั้งตัวฮีตเตอร์ไว้ในตู้ควบคุมด้วย เพื่อรักษาอุณหภูมิของ PLC ให้สามารถทำงานได้ แต่ถ้าอุณหภูมิสูงเกินกว่าย่านที่กำหนด เช่น 60 องศาเซลเซียส อาจต้องติดตั้งระบบปรับอากาศเพื่อรักษาช่วงอุณหภูมิทำงานไว้
4) การสั่นสะเทือน ธรรมชาติของเครื่องจักรหรืออุปกรณ์ในงานอุตสาหกรรมต้องใช้กำลังทางกลในการทำงานต่างๆ เมื่อกำลังทางกลนี้เกิดขึ้นจะส่งผลให้มีแรงกระแทกหรือการสั่นสะเทือนเกิดขึ้นได้ ซึ่งแรงนี้จะส่งต่อไปยังอุปกรณ์อื่นๆ เช่น PLC ถึงแม้ว่า PLC จะออกมาให้ต้านทานการสั่นสะเทือนได้ แต่บางครั้งการสั่นสะเทือนอาจส่งผลต่อการเดินสาย เช่น ขั้วต่อสายต่างๆ อาจหลวมได้
5) สัญญาณรบกวน จะส่งผลกระทบกับ PLC และอุปกรณ์อินพุตซึ่งอาจทำให้เกิดความคาดเคลื่อนในการทำงานหรือทำให้เกิดความเสียหายได้
6) ระบบจ่ายไฟ แรงดันไฟที่จ่ายให้กับระบบควบคุมอาจมีการแกว่งตัวบ้างในบางครั้ง เมื่อมีอุปกรณ์ไฟฟ้าขนาดใหญ่เปิด/ปิด เพื่อหลีกเลี่ยงปัญหาเหล่านี้ ควรใช้ Isolation Transformer, UPS หรืออุปกรณ์ป้องกันแรงดันไฟตก เพื่อป้องกันปัญหา
7) ไม่ควรติดตั้ง PLC ในสภาพแวดล้อมที่มี Corrosive gas เช่น คอลลีน และแอมโมเนีย เป็นต้น รวมถึงที่อยู่ในรูปแบบของแข็ง เช่น ฟอสฟอรัสและฟีนอล

ตารางที่ 6.2 เป็นมาตรฐานของตู้ที่จัดทำโดย NEMA (National Electric Manufacturers Association) เป็นตู้ที่ใช้กับแรงดันไฟต่ำกว่า 1000VAC ตัวอย่าง NEMA12 เป็นที่เลือกใช้กันทั่วไป

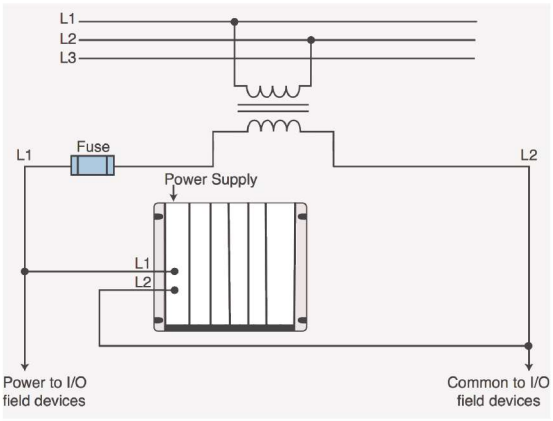
ตารางที่ 6.2 ตัวอย่างพิกัดป้องกันของตู้ตามมาตรฐาน NEMA

พิกัดป้องกัน	การใช้งาน	ความสามารถในการป้องกัน
NEMA1	ใช้ภายในอาคาร	สำหรับงานทั่วไป
NEMA12	ใช้ภายในอาคาร	ป้องกันฝุ่น สิ่งสกปรกที่ตกลงบนตู้ และของเหลวที่ไม่ก่อสนิม
NEMA3R	ใช้นอกอาคาร	ป้องกันฝุ่น น้ำ ฝนสาด และการเกิดน้ำแข็ง
NEMA4	ใช้นอกอาคาร	ป้องกันฝุ่น น้ำ ฝนสาด น้ำแข็ง การฉีดน้ำโดยตรง
NEMA4X	ใช้นอกอาคาร	ป้องกันฝุ่น น้ำ ฝนสาด น้ำแข็ง การฉีดน้ำโดยตรง และป้องกันสนิม

6.2.4 แหล่งจ่ายไฟและการเดินสาย

แหล่งจ่ายไฟกระแสสลับ (AC Power Supply)

กรณี PLC ที่ใช้งานต้องการแหล่งจ่ายไฟ AC ควรติดตั้ง Isolated Transformer หรือ Noise Filter เพิ่มเติมที่ด้านอินพุตของแหล่งจ่ายไฟก่อนจ่ายไฟให้กับ PLC ดังแสดงในรูปที่ 6.6 เนื่องจาก Isolated Transformer มีหลักการทำงานด้วยการเหนี่ยวนำสนามแม่เหล็กไฟฟ้าที่เกิดจากขดลวดด้าน Primary และสร้างให้เกิดแรงดันไฟฟ้าขึ้นที่ด้าน Secondary โดยไม่มีการต่อถึงกันทางไฟฟ้า ด้วยหลักการนี้เมื่อเกิดการกระชากของแรงดันหรือกระแสไฟฟ้าที่ด้าน Primary จะส่งผลกระทบไปยังด้าน Secondary น้อยกว่าการต่อถึงกันโดยตรง เนื่องจากแกนเหล็กของ Transformer เกิดการอิ่มตัวจึงไม่สนองตอบกระแสไฟที่กระชากขึ้น



รูป 6.6 แสดงการติดตั้งแหล่งจ่ายไฟ AC

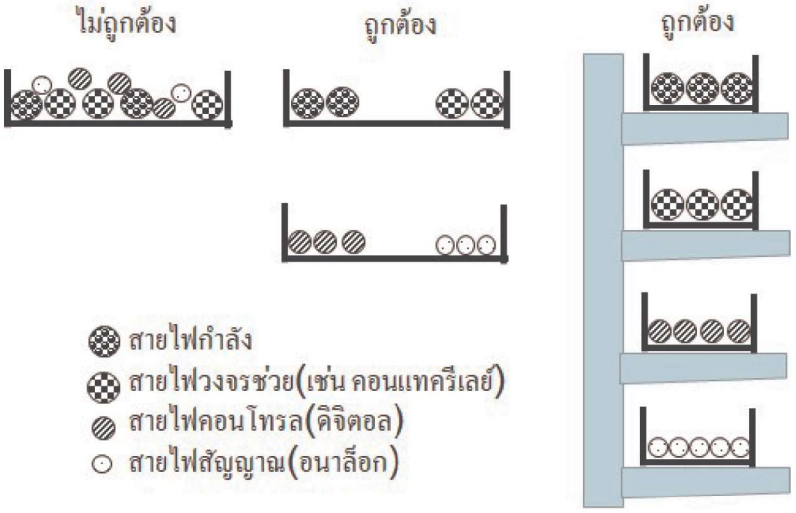
แหล่งจ่ายไฟกระแสตรง (DC Power Supply)

กรณี PLC ที่ใช้งานต้องการแหล่งจ่ายไฟ DC ซึ่งส่วนใหญ่แล้วจะใช้ไฟ 24VDC ดังนั้นจึงใช้ Switching Power Supply เพื่อจ่ายไฟให้กับ PLC เราไม่ขอแนะนำให้ใช้วงจรเรกติไฟเออร์แบบทั่วไปที่ใช้หม้อแปลงไฟฟ้าต่อกับไดโอดเพื่อทำให้เป็นไฟ DC เพราะไฟ DC ที่ได้จะไม่เรียบเพียงพอและอาจส่งผลต่อการทำงานของ PLC ได้ แต่มีผู้ใช้งานบางท่านที่ใช้อยู่เราขอแนะนำให้เปลี่ยนดีกว่า เพราะ PLC ราคาค่อนข้างแพงจะเสียหายเพราะแหล่งจ่ายไฟที่คุณภาพไม่ดี

6.2.5 การเดินสายไฟและการจัดกลุ่ม

การเดินสายสัญญาณอินพุต/เอาท์พุต

ควรเดินสายสัญญาณอินพุต/เอาท์พุตแยกออกจากสายไฟกำลังไม่ว่าจะเป็นภายในหรือภายนอกตู้ควบคุมโดยการแยกท่อหรือรางที่ใช้เดินสาย ทั้งนี้เพื่อป้องกันสัญญาณรบกวน (Noise) ที่เกิดจากสายไฟกำลังซึ่งอาจส่งผลกระทบต่ออุปกรณ์อินพุต/เอาท์พุตและ PLC ทำให้เกิดความเสียหายได้ รูปที่ 6.7 แสดงการเดินสายไฟที่แยกรางของสายไฟกำลังและสายไฟควบคุม



รูป 6.7 แสดงการเดินสายไฟที่แยกราง

การเดินสายแบบแบ่งกลุ่มหรือประเภทของสายไฟ จะเป็นอีกวิธีหนึ่งที่ช่วยลดปัญหาสัญญาณรบกวน (Noise) ที่เกิดขึ้นในระบบได้ โดยทั่วไปควรรวมสายไฟที่ใช้กับอุปกรณ์ทำงานต่างๆ เช่น มอเตอร์ไฟฟ้า ฮีตเตอร์ เป็นต้น ไว้ด้วยกัน และไม่ควรเดินสายไฟประเภทนี้ร่วมกับสายไฟประเภทอื่นๆ

เนื่องจากสัญญาณรบกวนที่เกิดขึ้นอาจแพร่กระจายไปตามสายไฟและโลหะต่างๆ ในบางครั้งอาจแพร่กระจายผ่านทางอากาศได้อีกด้วย การต่อโครงโลหะต่างๆ รวมทั้งท่อและรางสายไฟลงกราวด์เป็นอีกวิธีการหนึ่งที่ช่วยลดสัญญาณรบกวนได้ดี

ในบริเวณที่มีสัญญาณรบกวนมากสายไฟของอุปกรณ์อินพุต เช่น เซ็นเซอร์ อาจต้องใช้สายชิลด์ (Shield) และต่อลงกราวด์ที่ปลายสายเพียงด้านเดียว เพื่อลดสัญญาณรบกวน

6.3 การใช้ Master Control Relay (MCR)

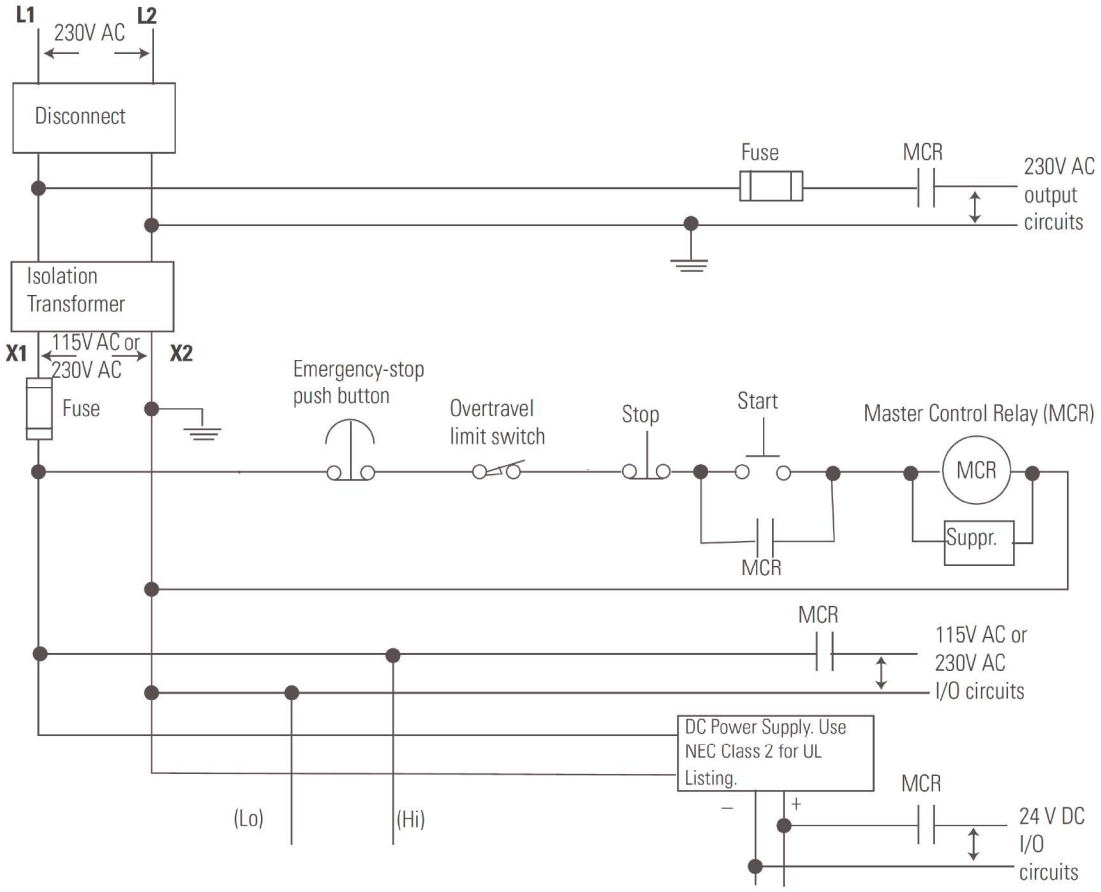
การใช้งวงจร MCR จะช่วยให้การสั่งให้เครื่องหยุดฉุกเฉินมีความน่าเชื่อถือสูง เพราะว่า MCR จะช่วยจัดวางสวิตช์หยุดฉุกเฉินได้หลายตำแหน่งในระบบ การติดตั้ง MCR จึงเป็นเรื่องเกี่ยวกับความปลอดภัยเป็นหลัก อุปกรณ์จำพวกลิมิตสวิตช์และสวิตช์หยุดฉุกเฉินต่างๆ ควรติดตั้งในตำแหน่งที่พนักงานสามารถเข้าถึงได้ง่ายและถูกต่ออนุกรมกัน เมื่ออุปกรณ์ตัวหนึ่งเปิดวงจรจะทำให้ MCR หยุดทำงานและตัดไฟที่จ่ายให้กับวงจรอินพุตและเอาต์พุต

ถึงแม้ว่าแหล่งจ่ายไฟของอินพุตเอาต์พุตจะถูกตัด แต่ยังคงมีไฟจ่ายให้กับ PLC เพื่อใช้สังเกตุไฟแสดงผลต่างๆบน CPU

การใช้สวิตช์หยุดฉุกเฉิน (Emergency-Stop Switches) มีสิ่งที่ควรระวังดังนี้

• อย่าเขียนโปรแกรมเพื่อทำงานแทนสวิตช์หยุดฉุกเฉิน ควรใช้สวิตช์หยุดฉุกเฉินเพื่อสั่งหยุดเครื่องจักรด้วยการสั่งให้ MCR หยุดทำงาน

• ตำแหน่งติดตั้งต้องเข้าถึงได้ง่ายและมีป้ายแสดงชัดเจน



รูป 6.8 แสดงวงจร MCR

6.4 ระบบป้องกันการล้มเหลว (Fail-Safe)

ระบบทุกระบบอาจเกิดความล้มเหลวในการทำงานได้ การออกแบบระบบป้องกันภัย (Fail-Safe) จะช่วยลดความเสียหายที่อาจเกิดขึ้นกับคนและอุปกรณ์ได้ ตัวอย่างเช่น สวิตช์ Stop ที่ใช้หน้าคอนแทค NC ต่อเข้ากับ PLC ออกแบบเพื่อสั่งให้เครื่องจักรหยุดทำงานเมื่อกดสวิตช์ ถ้าสายไฟเกิดขาดขึ้นมันจะทำให้เครื่องจักรหยุดทำงานทันที เหมือนกับการกดสวิตช์นั่นเอง จะเกิดอะไรขึ้นถ้าใช้หน้าคอนแทคแบบ NO ข้างล่างนี้อธิบายการทำงานของระบบ Fail-Safe ที่ใช้หน้าคอนแทคแบบ NO และ NC

NO (Normally open) – เมื่อต่อสายไฟกับสวิตซ์หรือเซ็นเซอร์สำหรับทำหน้าที่ Start ควรใช้หน้าคอนแทค NO เพราะว่าเมื่อมีปัญหาเกิดขึ้นเครื่องจักรจะไม่สามารถทำงานได้

NC (Normally Closed) – เมื่อต่อสายไฟกับสวิตซ์ที่ใช้หยุดเครื่องจักรควรใช้หน้าคอนแทค NC เพราะว่าถ้าระบบล้มเหลวจะสั่งให้เครื่องจักรหยุดทำงาน เช่น E-Stop ต้องใช้ NC เสมอ และมันจะใช้ตัดแหล่งจ่ายโดยตรง ไม่ควรต่อเข้าอินพุตของ PLC และให้ PLC สั่งตัดการทำงานเมื่อมีสิ่งผิดปกติเกิด เนื่องจาก PLC อาจทำงานล้มเหลวหรือทำงานผิดพลาดได้

ตัวอย่างฮาร์ดแวร์ของระบบ Fail-Safe

- ใช้ระบบทำงานซ้อน (Redundancy) ขึ้นอยู่กับความจำเป็นของระบบ
- การต่อสวิตซ์ E-Stop ต้องสั่งตัดระบบไฟฟ้าโดยตรงไม่ควรต่อผ่าน PLC
- สวิตซ์สั่ง Shutdown ควรเข้าถึงได้ง่ายจากผู้ปฏิบัติงาน
- ใช้ขั้นตอนการ Startup เครื่องจักรที่มีระบบตรวจสอบปัญหาต่างๆที่อาจเกิดขึ้น

เมื่อออกแบบวงจรไฟฟ้าต่าง ๆ แล้ว ให้ดำเนินการออกแบบผังการวางอุปกรณ์ภายในตู้ควบคุม การออกแบบต้องคำนึงถึงระยะห่างและความเหมาะสมตามที่ผู้ผลิตแนะนำโดยดูได้จากคู่มือการใช้งาน

สรุปท้ายบท

เราจะเห็นว่าการออกแบบและติดตั้งนั้นมีผลต่อการทำงานของระบบควบคุมค่อนข้างมาก ซึ่งมีสิ่งจำเป็นที่ต้องคำหนึ่งถึงหลายอย่าง เช่น ระบบกราวด์ การป้องกันสัญญาณรบกวน เป็นต้น ถ้าการติดตั้งไม่เหมาะสมอาจส่งผลกระทบต่ออายุการใช้และเสถียรภาพการทำงานของระบบควบคุม

Chapter

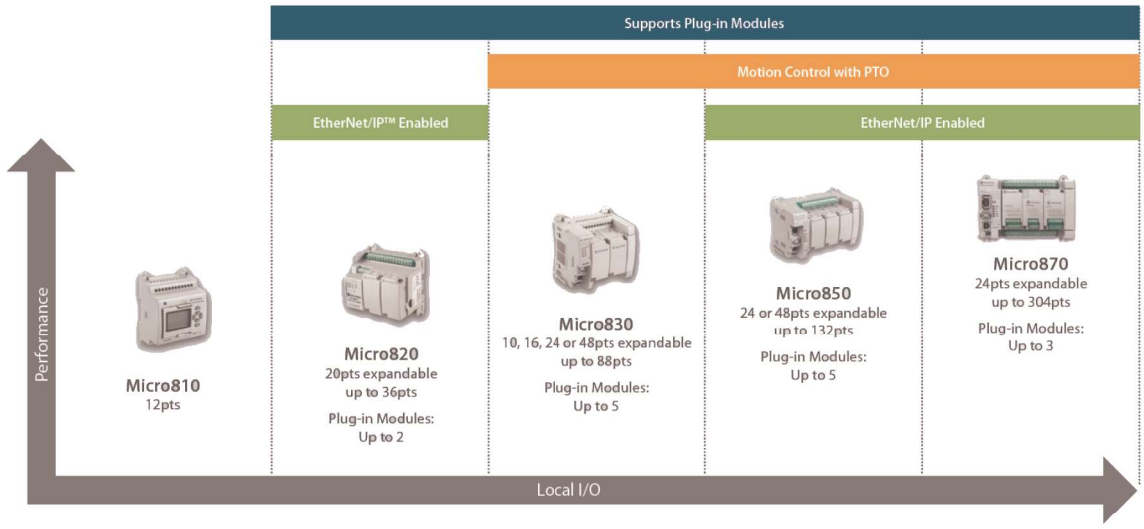
7

รายละเอียดและองค์ประกอบของ Micro800

Micro 800 เป็น PLC จากค่าย Allen Bradley ที่มาพร้อมฟังก์ชั่นการควบคุมเซอร์โวมอเตอร์ การควบคุมแบบลอจิก และการทำงานด้านระบบเครือข่ายในตัว เหมาะสำหรับงานควบคุมเครื่องจักรขนาดเล็กและต้องการเชื่อมต่อเข้ากับ Ethernet Micro 800 ใช้งานควบคู่กับซอฟต์แวร์ Connected Components Workbench (CCW) สำหรับใช้ในการเขียนโปรแกรม ที่สามารถดาวน์โหลดได้ฟรีจากเว็บไซต์ของ Rockwell Automation หรือ QR code ด้านล่างนี้ ซึ่งซอฟต์แวร์ CCW สามารถใช้งานกับ Micro 800 ได้ทุกรุ่น

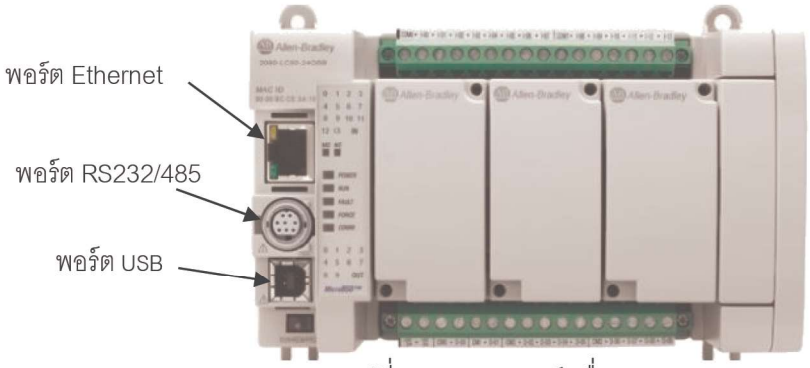
7.1 คุณสมบัติของ Micro800

Micro800 ประกอบด้วย Controller จำนวน 5 รุ่นด้วยกัน โดยแบ่งตามจำนวน I/O และคุณสมบัติพิเศษ เริ่มต้นจาก Micro810 ที่มี I/O เพียง 12 จุดเท่านั้น ส่วนรุ่นใหญ่สุดคือ Micro870 ในที่นี้จะใช้ Micro850 เป็นตัวอย่างในการอธิบายคุณสมบัติของฮาร์ดแวร์และการเขียนโปรแกรม



คุณสมบัติทางด้านฮาร์ดแวร์ Micro850

Micro850 เป็นหนึ่งในตระกูล Micro800 ที่มีฟังก์ชั่นการใช้งานครบครันเหมาะสำหรับระบบควบคุมอัตโนมัติขนาดเล็กหรือเครื่องจักร Stand-Alone หรือใช้งานเป็นส่วนหนึ่งของเครือข่ายสำหรับการควบคุมในระบบที่ใหญ่ขึ้น โดยมีคุณสมบัติเบื้องต้นดังนี้



รูปที่ 7.1 แสดงพอร์ตสื่อสารต่างๆ

- **รองรับการเชื่อมต่อระบบ Ethernet ที่เป็นมาตรฐาน**

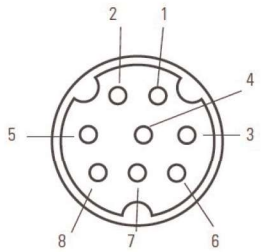
Micro850 มีพอร์ต Ethernet อยู่ 1 พอร์ต ที่ติดตั้งมาพร้อมกับคอนโทรลเลอร์ โดยพอร์ตนี้สามารถนำไปเชื่อมต่อกับอุปกรณ์ที่รองรับเครือข่าย Ethernet ได้ พอร์ตดังกล่าวรองรับการใช้งาน EtherNet/IP และ Modbus/TCP ด้วย

- **รองรับการใช้งาน USB 2.0**

พอร์ต USB ที่ให้มาพร้อม Micro850 เป็นชนิด non-isolated จำนวน 1 พอร์ต ใช้สำหรับเขียนโปรแกรมเพื่อโหลดเข้าสู่คอนโทรลเลอร์หรือทำงานอื่นๆที่เกี่ยวข้อง

- **พอร์ท RS232/RS485**

พอร์ตนี้ใช้สำหรับสื่อสารกับอุปกรณ์ต่างๆที่ต้องการการสื่อสารแบบอนุกรมในระยะสั้นไม่เกิน 3 เมตร

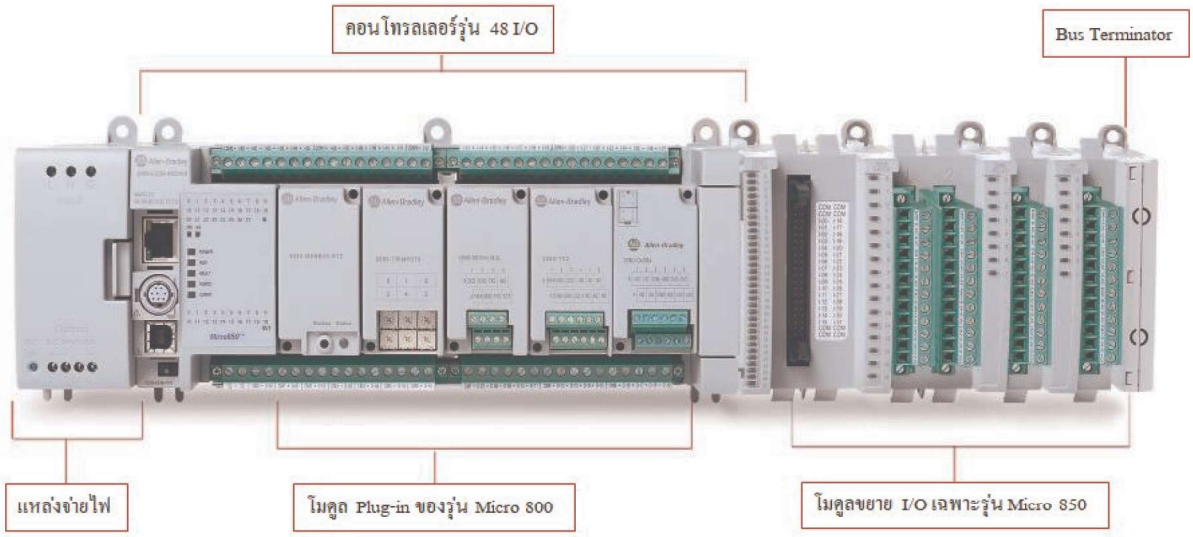


Pinout table

Pin	Definition	RS-485 Example	RS-232 Example
1	RS-485+	B(+)	(not used)
2	GND	GND	GND
3	RS-232 RTS	(not used)	RTS
4	RS-232 RxD	(not used)	RxD
5	RS-232 DCD	(not used)	DCD
6	RS-232 CTS	(not used)	CTS
7	RS-232 TxD	(not used)	TxD
8	RS-485-	A(-)	(not used)

- **รองรับการต่อขยายยูนิต I/O ต่าง ๆได้**

ผู้ใช้งานสามารถที่จะนำยูนิตขยาย I/O มาต่อเข้ากับยูนิต CPU ได้ อย่างไรก็ดี ให้ตรวจสอบก่อนว่ากำลังไฟของยูนิต CPU เพียงพอที่จะจ่ายให้กับยูนิตขยาย I/O หรือไม่



รูปที่ 7.2 การต่อยูนิตขยาย I/O

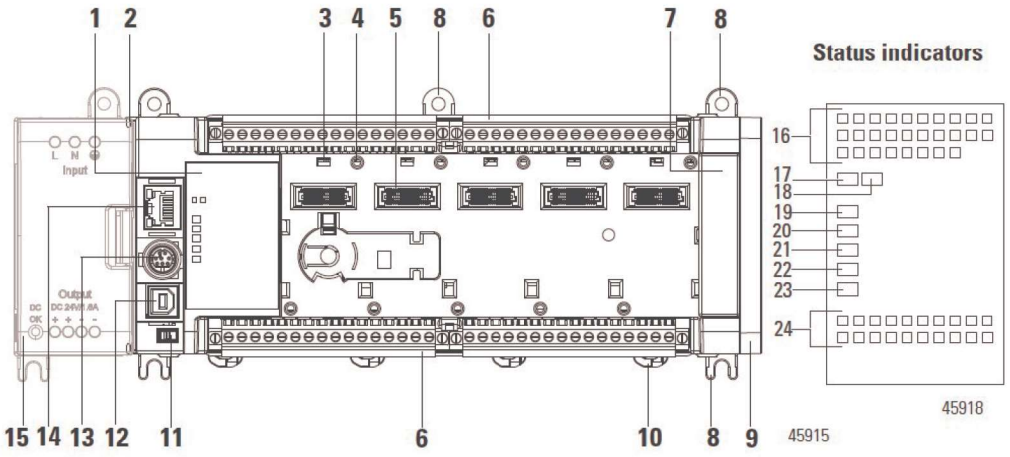
- **รองรับการใช้งาน SD Memory Card**

Micro850 มีช่องใส่ SD Memory Card มาให้ 1 ช่อง สามารถนำไปใช้ประโยชน์ได้หลายอย่างเช่น Backup/Restore Program, Data Collection เป็นต้น

7.2 ข้อมูลที่สำคัญต่าง ๆของ Micro850

คุณสมบัติ	**รุ่น Micro850 24 I/O**	**รุ่น Micro850 48 I/O**
ความจุโปรแกรม (Step)	10 K	10 K
Data bytes	20 KB	20 KB
ความเร็วคำสั่งพื้นฐาน	0.3 uS ต่อคำสั่งพื้นฐาน	
เวลาสแกนสูงสุด	<0.25 mS	
จำนวนอินพุตเอาต์พุตในตัว	14 in/ 10 out	28 in/ 20 out
จำนวนโมดูล Plug-in	3	5
การเขียนโปรแกรม	IEC 61131-3 (LD, FB, ST)	
Floating point	32-Bit และ 64-Bit	
Serial port protocol	Modbus RTU Master/Slave, ASCII/Binary, CIP serial	
Communication port	USB, RS232/RS485 และ Ethernet 10/100 Base	

7.3 โครงสร้างและส่วนประกอบของ Micro850



รูปที่ 7.3 รายละเอียดของ Micro 850 (48 I/O)

1	ไฟแสดงผล	9	ฝาปิดสล๊อตขยาย I/O
2	สล็อตสำหรับต่อแหล่งจ่ายไฟ	10	ตัวล็อกราง DIN
3	ตัวล็อกโมดูล Plug-in	11	สวิตช์โหมด
4	รูสกรูสำหรับยึดโมดูล Plug-in	12	พอร์ท USB
5	คอนเน็กเตอร์สำหรับโมดูล Plug in	13	พอร์ท RS232/RS485
6	เทอร์มินอลสำหรับอินพุตเอาท์พุต(ถอดได้)	14	พอร์ท EtherNet/IP
7	ฝาปิดด้านขวา	15	แหล่งจ่ายไฟ AC (อุปกรณ์เสริม)
8	รูยึด		

7.3.1 ความหมายของหลอดไฟแสดงสถานะการทำงานของ CPU

รูปที่ 7.4 หลอดไฟแสดงสถานะการทำงานของ CPU

	สีแสดงผล	สถานะ	ความหมาย
POWER	เขียว	สว่าง	แหล่งจ่ายไฟปกติ
		ไม่สว่าง	ไม่มีการจ่ายไฟ
RUN	เขียว	สว่าง	โปรแกรมทำงานปกติในโหมด RUN
		กระพริบ	โมดูลหน่วยความจำกำลังกำลังโอนถ่ายข้อมูล
		ไม่สว่าง	โปรแกรมไม่ทำงาน

FAULT	แดง	สว่าง	มี Fault เกิดขึ้นไม่สามารถทำงานได้
		กระพริบ	มี Fault เกิดขึ้นที่สามารถแก้ไขได้เกิดขึ้น
		ไม่สว่าง	ทำงานปกติ ไม่มีสิ่งใดผิดปกติ
FORCE	ส้ม	สว่าง	มีการบังคับข้อมูลในหน่วยความจำ
		ไม่สว่าง	ทำงานปกติ
COMM	เขียว	สว่าง	มีการสื่อสารข้อมูล
		ไม่สว่าง	ไม่มีการสื่อสารข้อมูล

7.4 โมดูล Plug-in และยูนิตขยาย I/O

Micro800 สามารถเพิ่มโมดูล Plug-in บนตัว CPU ได้ แต่จะทำได้เพียงบางรุ่นเท่านั้น เช่น Micro820, Micro830, Micro850 และ Micro870 ซึ่งจะสามารถเสียบโมดูล Plug-in ได้ ส่วน Micro850 และ Micro870 เท่านั้นที่ต่อยูนิตขยาย I/O เพิ่มได้

ตารางที่ 7.1 แสดงชื่อรุ่นของโมดูล Plug-in

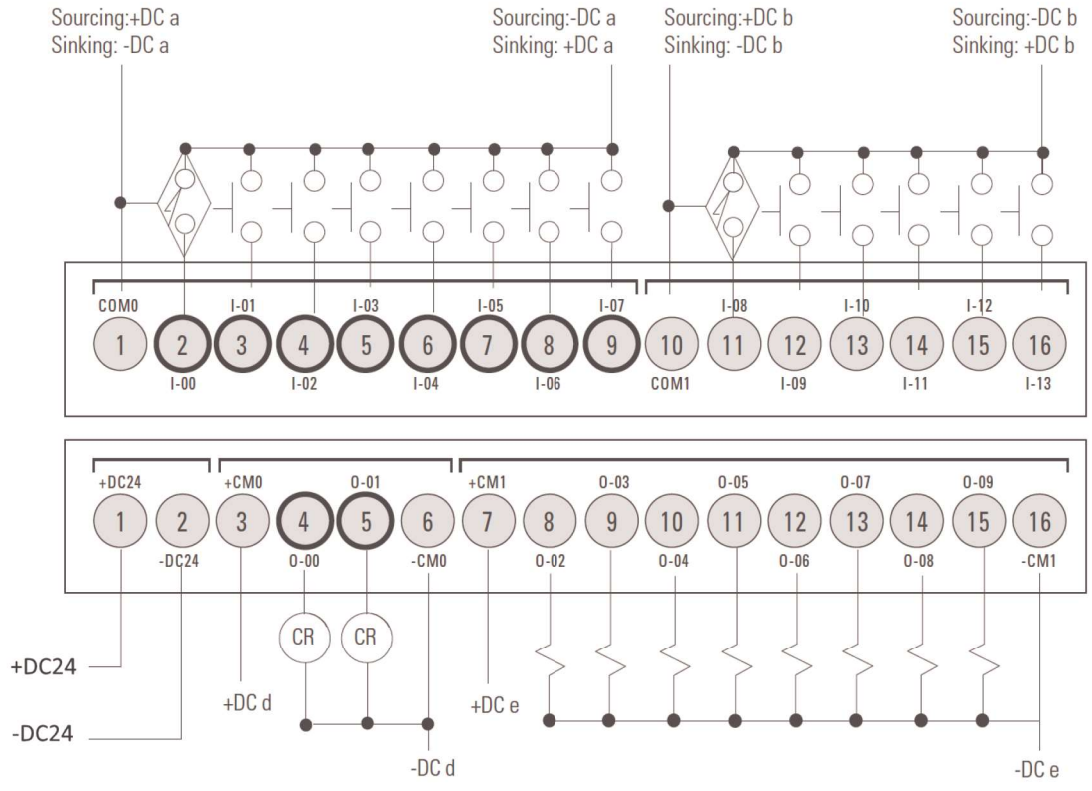
รูปร่าง	ชื่อรุ่น	รายละเอียด
	2080-IQ4	อินพุตดิจิตอล 4 จุด, 12/24VDC (Sink/Source, Type3)
	2080-OB4	เอาต์พุตดิจิตอล 4 จุด, 12/24VDC (Source)
	2080-OV4	เอาต์พุตดิจิตอล 4 จุด, 12/24VDC (Sink)
	2080-OW4I	เอาต์พุตรีเลย์ 4 จุด, (2A ต่อจุด)
	2080-IQ4OB4	อินพุตดิจิตอล 4 จุด, 12/24VDC (Sink/Source,Type3) และ เอาต์พุตดิจิตอล 4 จุด SRC (Source)
	2080-IQ4OV4	อินพุตดิจิตอล 4 จุด, 12/24VDC (Sink/Source, Type3) และ เอาต์พุตดิจิตอล 4 จุด, 12/24VDC (Sink)
	2080-IF2	อินพุตอนาล็อก 2 จุด, 0-20 mA, 0-10V, (non-isolated 12-bit)
	2080-IF4	อินพุตอนาล็อก 4 จุด, 0-20 mA, 0-10V, (non-isolated 12-bit)
	2080-OF2	เอาต์พุตอนาล็อก 2 จุด, 0-20 mA, 0-10V,(non-isolated 12-bit)
	2080-SERIALISOL	พอร์ตอนุกรม RS232/485 (isolated)
	2080-TRIMPOT6	อินพุตอนาล็อกสำหรับต่อตัวต้านทานปรับค่าได้(Trimpot) 6 จุด
	2080-RTD2	RTD 2 จุด, (non-isolated, ±1.0 °C)
	2080-TC2	TC 2 จุด, (non-isolated, ±1.0 °C)
	2080-MOT-HSC	High Speed Counter, 250kHz, Differential Line Receiver, เอาต์พุตดิจิตอล 1 จุด
	2080-DNET20	DeviceNet Scanner, 20 Nodes

ตารางที่ 7.2 แสดงชื่อรุ่นของยูนิตขยาย I/O

รูปร่าง	ชื่อรุ่น	รายละเอียด
	2085-IQ16	อินพุตดิจิตอล 16 จุด, 12/24VDC, Sink/Source
	2085-IQ32T	อินพุตดิจิตอล 32 จุด, 12/24VDC, Sink/Source
	2085-OV16	เอาต์พุตดิจิตอล 16 จุด, 12/24VDC, Sink
	2085-OB16	เอาต์พุตดิจิตอล 16 จุด, 12/24VDC, Source
	2085-OW8, 2085-OW16	เอาต์พุตรีเลย์รุ่น 8 จุด และรุ่น 16 จุด, (2A ต่อจุด)
	2085-IM8	อินพุต 8 จุด, 240 VAC
	2085-OA8	เอาต์พุต 8 จุด, 120/240 VAC
	2085-IF4, 2085-IF8	อินพุตอนาล็อกรุ่น 4 จุด และรุ่น 8 จุด, 0 ~ 20mA, -10V ~ +10V isolated,14บิต
	2085-OF4	เอาต์พุตอนาล็อก 4 จุด, 0 ~ 20mA, -10V ~ +10V, isolated, 12 บิต
	2085-IRT4	RTD และ TC 4 จุด, isolated, ±0.5 °C
	2085-ECR	End Cap Terminator

7.5 การเดินสายไฟสำหรับ Micro850

ในหัวข้อนี้เราจะกล่าวถึงการเดินสายไฟสำหรับ PLC โดยใช้ตัวอย่าง Micro850 รุ่น 24 I/O ซึ่งจะคำนึงถึงการลดสัญญาณรบกวนทางไฟฟ้าและแสดงตัวอย่างการต่อสายวงจรอินพุตเอาต์พุต



รูปที่ 7.5 แสดงการต่อสายไฟของ Micro850 (24 I/O)

จากรูปที่ 7.5 แหล่งจ่ายไฟที่ต่อให้กับ PLC จะเป็นไฟ DC ขนาด 24V ส่วนวงจรอินพุตจะแยกออกเป็น 2 กลุ่ม โดยมีคอมมอนแยกกัน คือ COM0 และ COM1 ซึ่งสามารถต่ออินพุตเป็น Sinking และ Sourcing แยกประเภทกันได้ แต่ถ้าอุปกรณ์อินพุตเป็นประเภทเดียวกันเราสามารถนำคอมมอนมาต่อร่วมกันได้ ส่วนเอาต์พุตจะแยกเป็น 2 กลุ่มเช่นกัน คือ CM0 และ CM1

7.5.1 การลดสัญญาณรบกวน

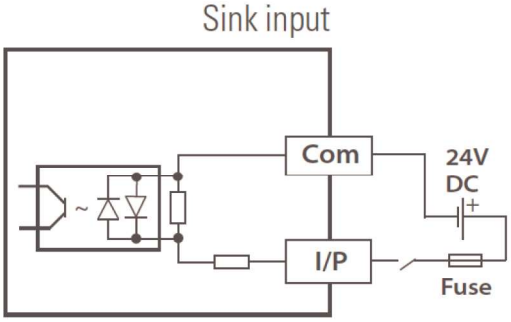
เนื่องด้วย PLC ถูกใช้งานในสิ่งแวดล้อม (สิ่งแวดล้อม หมายถึง อุปกรณ์ต่างๆที่ติดตั้งในบริเวณที่ PLC ติดตั้งอยู่ เช่น มอเตอร์) ที่หลากหลาย มันเป็นไปได้ยากที่สัญญาณรบกวน (noise) จะถูกกำจัดได้หมด เพื่อช่วยลดผลกระทบของสัญญาณรบกวนที่อยู่แวดล้อมเราควรติดตั้ง Micro800 ในตู้ที่มีมาตรฐาน เช่น NEMA เพื่อให้แน่ใจว่ามันถูกต่อกราวด์อย่างเหมาะสม ควรตรวจสอบเสมอระบบเมื่อสิ่งแวดล้อมในการติดตั้งมีการเปลี่ยนแปลง เช่น มีเครื่องจักรใหม่ติดตั้งเพิ่มเติม หรือแหล่งสัญญาณรบกวนใหม่ที่ติดตั้งใกล้ระบบ เช่น มีการติดตั้ง Drives หรือ Inverter ในบริเวณใกล้เคียง

ข้อแนะนำการเดินสายไฟสัญญาณอนาลอก

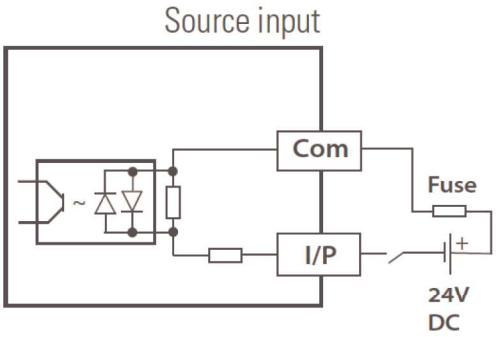
- ขั้วคอมมอน (COM) ของอนาล็อกไม่ได้แยกออกทางไฟฟ้าจากระบบและถูกต่อเข้ากับคอมมอนของแหล่งจ่ายไฟ
- ควรใช้สาย Belden #8761 หรือเทียบเท่าและเดินสายให้ห่างจากสายไฟ AC
- ในภาวะปกติสายชิลด์(Shield) ควรต่อเข้ากับโครงตู้ (earth ground) และระยะการต่อต้องสั้นที่สุดเท่าที่จะเป็นไปได้
- กรณีอินพุตแรงดันไฟฟ้า ถ้าต้องการสัญญาณที่มีความเที่ยงตรงสูงสุด ควรเดินสายให้สั้นที่สุด

7.5.2 การต่อวงจรอินพุตและเอาต์พุต

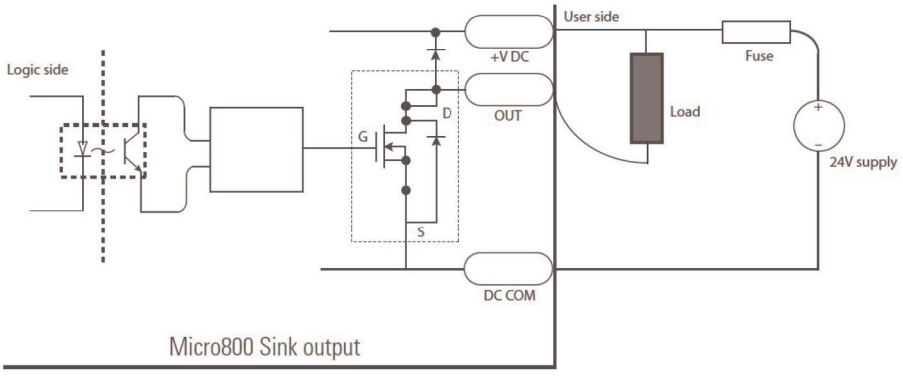
รูปต่อไปนี้แสดงการต่ออินพุตเอาต์พุตทั้งประเภท Sinking และ Sourcing



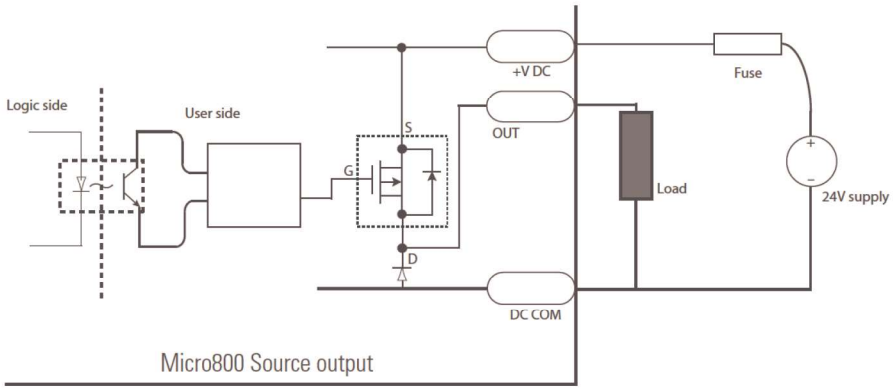
รูปที่ 7.6 แสดงการต่อสายไฟอินพุต Sinking ของ Micro800



รูปที่ 7.7 แสดงการต่อสายไฟอินพุต Sourcing ของ Micro800



รูปที่ 7.8 แสดงการต่อสายไฟเอาต์พุตพุต Sinking ของ Micro800



รูปที่ 7.9 แสดงการต่อสายไฟเอาต์พุตพุต Sourcing ของ Micro800

สรุปท้ายบท

บางครั้งการติดตั้งที่ไม่เหมาะสมจะส่งผลกระทบทำให้ PLC เกิดความเสียหายก่อนกำหนด การติดตั้งในสภาพแวดล้อมที่ไม่ดี เช่น มีความชื้นสูง มี Corrosive gas เป็นต้น ซึ่งสิ่งเหล่านี้จะทำให้วงจรอีเลคทอรนิคส์ของ PLC เสียหายได้ ดังนั้นการติดตั้งระบบ PLC เพื่อให้มีอายุการใช้งานที่ยาวนานควรคำนึงถึงสิ่งที่กล่าวมาทั้งหมดในบทนี้

Chapter

8

การเขียนโปรแกรมมาตรฐาน IEC

IEC61131-3 เป็นมาตรฐานเปิดสำหรับ PLC ที่ประกาศใช้ครั้งตั้งแต่ปลายปี 1993 โดย International Electrotechnical Commission(IEC) ปัจจุบันเป็นการแก้ไขครั้งที่ 3 แล้ว ซึ่งได้ประกาศใช้เมื่อต้นปี 2013 จุดประสงค์ของ IEC ในการกำหนดมาตรฐานนี้ขึ้นมาเพื่อให้ผู้ผลิต PLC ใช้เป็นมาตรฐานในการสร้างสถาปัตยกรรมของซอฟต์แวร์และภาษาที่ใช้เขียนโปรแกรม PLC ที่มีมาตรฐานเดียวกัน โดยมีภาษาที่เป็นกราฟฟิกและภาษาเขียนดังนี้

1.Ladder diagram (LD)

2.Function block diagram (FBD)

3.Structured text (ST)

4.Instruction list (IL)

5.Sequential function chart (SFC)

การใช้มาตรฐาน IEC จะทำให้การเขียนโปรแกรม PLC ไม่ต้องเรียนรู้ภาษาเฉพาะของ PLC แต่ละยี่ห้อและจดจำหมายเลขอินพุตเอาท์พุตของ PLC ยี่ห้อนั้นๆ อีกต่อไป เช่น ไม่ต้องใช้ X0, Y0, I0 และ Q0 เป็นต้น เพราะ PLC ที่รองรับมาตรฐานนี้จะมีหลักการเขียนและคำสั่งๆ ที่เหมือนกันและสามารถกำหนดชื่ออินพุตและเอาท์พุตได้เอง ซึ่งจะคล้ายๆ กับภาษา C ไม่ว่าเขียนบน OS ใดมันก็ใช้หลักการเดียวกัน แต่ความแตกต่างของ PLC จะอยู่ที่คุณสมบัติของฮาท์ดแวร์และซอฟ์ตแวร์มากกว่า รวมถึงคุณภาพและความน่าเชื่อถือ

8.1 โครงสร้างของโปรแกรม (Program)

โปรแกรมการทำงานของ PLC ตระกูล Micro 800 จะถูกเรียกว่า **POU** ย่อมาจาก **P**rogram **O**rganization **U**nit ซึ่งถูกกำหนดขึ้นมาตามมาตรฐาน IEC 61131-3 ประกอบด้วยส่วนของ Local variable และ Algorithm เราสามารถสร้าง POUs ได้ 3 ลักษณะคือ

- **Program**

- เป็นโปรแกรมหลักที่สามารถสร้างขึ้นมาโดยใช้ส่วนประกอบจาก Instruction, Function หรือ Function Block
- POUs ต้องมี Program เป็นส่วนประกอบอย่างน้อย 1 Program เสมอ

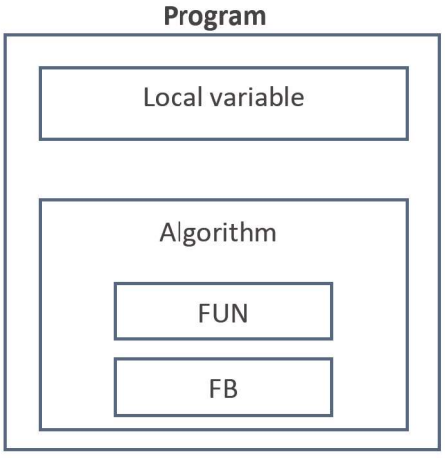
- **Function Blocks (FBs)**

- ทำงานเมื่อเงื่อนไขด้านอินพุตเป็นจริง
- สามารถให้ตัวแปรภายใน Function Block จดจำค่าหรือสถานะการทำงานคงค้างไว้ได้ เช่น TON และ CTU เป็นต้น (TON คือ Timer On Delay, CTU คือ Up Counter)
- ถูกเรียกใช้งานจาก Program หรือ Function Block อื่นได้

- **Functions (FUNs)**

- ทำงานเมื่อตัวแปรอินพุต Enable เป็นจริง
- ให้ค่าเอาท์พุตเหมือนเดิมในขณะที่อินพุตเป็นค่าเดิม
- ไม่สามารถจดจำค่าสถานะการทำงานเอาไว้ในตัวแปรภายในได้
- ไม่สามารถเรียกใช้งาน Function Block ภายใน Function ได้
- ถูกเรียกใช้ได้จาก Program, Function หรือจาก Function Block อื่นได้

รูปที่ 8.1 แสดงโครงสร้างของ Program ที่ประกอบด้วยตาราง Local variable และ Algorithm

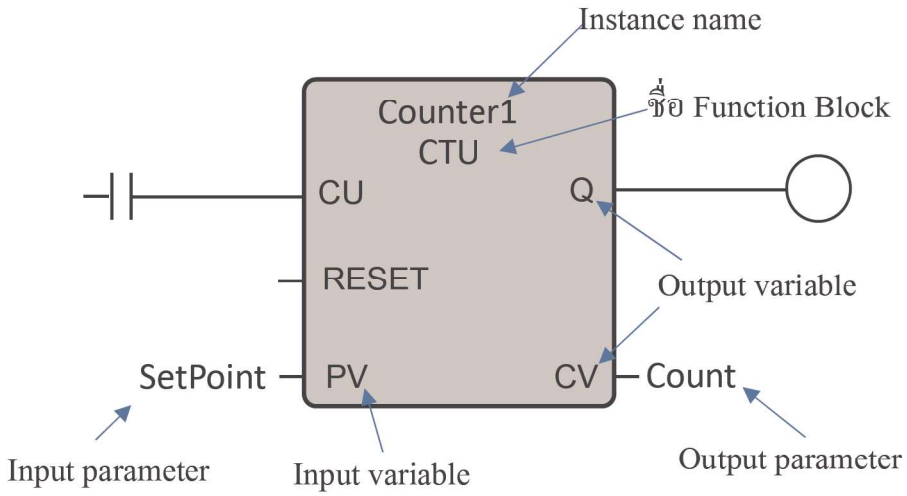


รูปที่ 8.1 แสดงโครงสร้างของ Program

8.2 โครงสร้างของ Function Block

Function Block(FB) คือ การเขียนโปรแกรมเพื่อการทำงานบางอย่างโดยสามารถสร้างได้จากภาษา Ladder(LD), Function Block Diagram(FBD) หรือ Structure Text(ST) ก็ได้ขึ้นอยู่กับความถนัดของผู้เขียน ซึ่ง Function Block Diagram(FBD) คือ ภาษาที่ใช้เขียนโปรแกรม แต่ Function Block คือ โปรแกรมที่ถูกสร้างขึ้นจากภาษา LD, FBD หรือ ST มันต่างกันนะครับ

โดยที่ Function Block จะประกอบด้วยพารามิเตอร์ Input/Output และส่วนของโปรแกรม คือ Algorithm ในการเรียกใช้งาน Function Block จะต้องกำหนด Instance name ขึ้นมา ซึ่งมันคือ ชื่อที่ใช้เป็นตัวแทน Function Block นั้นๆ และถ้ามีการเรียกใช้งาน Function Block เดียวกันมากกว่า 1 จุด จะต้องกำหนด Instance name ให้แตกต่างกันไปทุกครั้ง รูปข้างล่างนี้แสดงองค์ประกอบต่างๆของ Function Block

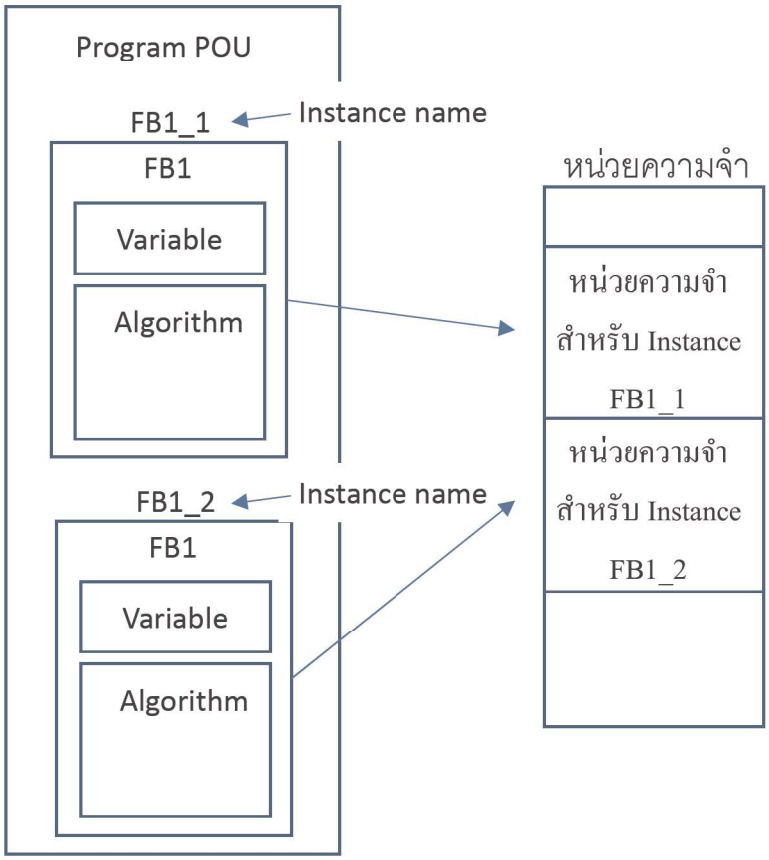


รูปที่ 8.2 แสดงองค์ประกอบต่างๆ ของ Function Block

จากรูปที่ 8.2 สามารถอธิบายรายละเอียดองค์ประกอบต่างๆ ของ FB ได้ดังนี้

- *Function Block Name* คือ ชื่อของ FB ที่เรียกใช้งาน เช่น TON และ CTU
- *Instance Name* คือ ชื่อที่ใช้อ้างแทน FB นั้นๆ เราสามารถเรียกใช้ TON ได้หลายตำแหน่งภายในโปรแกรมแต่เราต้องตั้งชื่อ Instance ให้ TON ในแต่ละตำแหน่งที่มันถูกเรียกใช้งานด้วย
- *Input variable* คือ ตัวแปรของ FB ที่ใช้รับค่าอินพุตเพื่อการประมวลผล
- *Input parameter* คือ ค่าที่ส่งให้กับตัวแปรอินพุตของ FB อาจเป็นค่าคงที่หรือตัวแปรก็ได้
- *Output variable* คือ ตัวแปรที่ใช้สำหรับส่งค่าจากการประมวลผลออกไปภายนอก
- *Output parameter* คือ ใช้สำหรับรับค่าจากตัวแปรเอาต์พุต

การเรียกใช้งาน FB ในโปรแกรมหลักนั้น ทุกครั้งที่มีการเรียกใช้งาน Instance name ของ FB จะใช้พื้นที่บนหน่วยความจำหลัก ดังนั้นการใช้ FB ไม่ได้ช่วยลดหน่วยความจำลง แต่จะช่วยอำนวยความสะดวกในการเขียนโปรแกรมมากกว่า



รูปที่ 8.3 แสดงการสำรองหน่วยความจำของ Function block

เราสามารถอ้างถึงตัวแปรของ Function Block ใน Structure Text ได้โดยมีรูปแบบการเรียกใช้ได้ดังนี้ *InstanceName.VariableName*

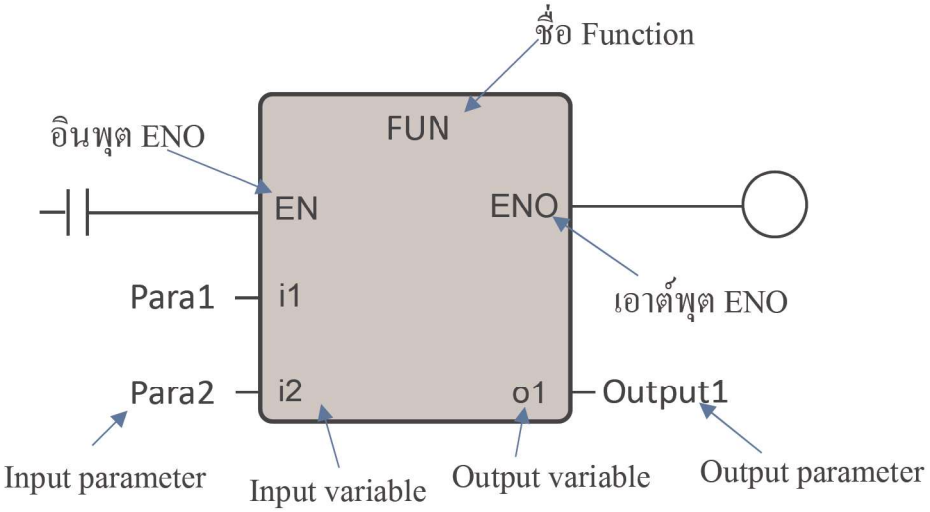
การอ้างถึงตัวแปร (Variable) ของ FB จะใช้รูปแบบ **InstanceName.VariableName** และสามารถกระทำได้ภายใน Program เดียวกันเท่านั้น แต่ไม่สามารถอ้างถึงภายในของ FB อื่นหรือโปรแกรมอื่นๆ ได้ ดังแสดงในรูปข้างล่างนี้

```
Output:= TON_1.Q;
```

TON_1 คือ Instance Name และ Q คือ Variable Name

8.3 โครงสร้างของ Function

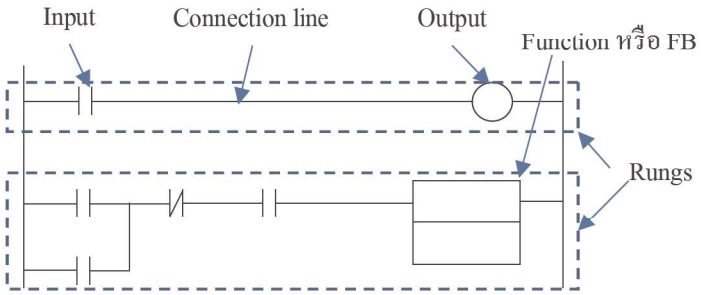
Function สามารถสร้างได้จากการเขียน Ladder หรือ Structure Text โดยเอา Function มาตรฐานมาร่วมใช้งานได้ โดยที่ Function จะประกอบด้วยพารามิเตอร์ Input/Output และส่วนของรายละเอียดโปรแกรมคือ Algorithm มันมีข้อแตกต่างจาก Function Block ตรงที่ Function จะต้องมีขาอินพุต EN และเอาต์พุต ENO แต่ไม่ต้องกำหนด Instance name เหมือนใน FB



รูปที่ 8.4 แสดงองค์ประกอบต่างๆ ของ Function

8.4 ภาษาที่ใช้ในซอฟต์แวร์ CCW

ภาษาที่ใช้ในการเขียน Algorithms เพื่อสร้าง POU (Program, Function หรือ Function Block) สำหรับ Micro 800 คือ การเขียนโปรแกรมแบบ Ladder (LD), Function Block Diagram(FBD) และ Structure Text (ST)

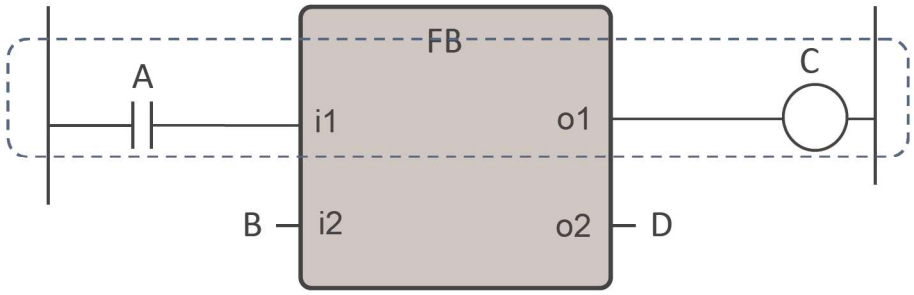


รูปที่ 8.5 โครงสร้างของ Ladder diagram ของ Micro 800

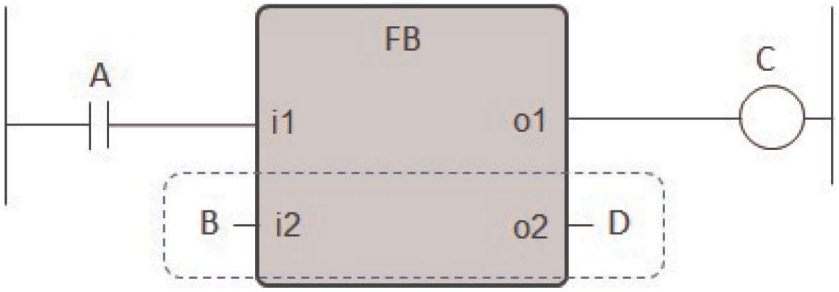
8.5 การใช้งาน Function และ Function Block ในโปรแกรมแลดเดอร์

ในการใช้งาน Function และ Function Block จะทำการลากเส้นจาก bus bar ทางด้านซ้ายมือผ่านเงื่อนไขด้าน input ไปเข้า i1 ของ Function Block เพียงตำแหน่งเดียวเท่านั้น และลากเส้นด้าน output คือ o1 ไปหา bus bar ทางด้านขวาโดยผ่านเอาท์พุตบิตเพียงบิตเดียวเช่นกัน

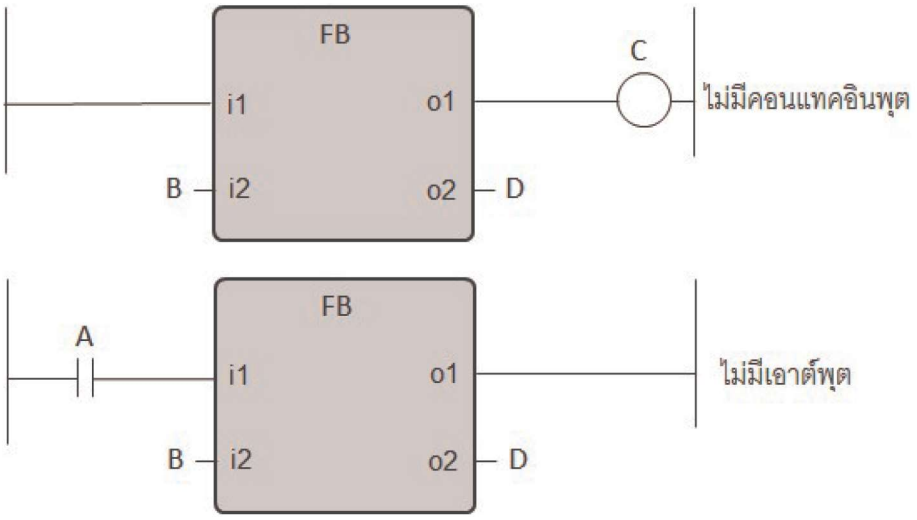
สำหรับ Function จะใช้ EN แทน i1 และ ENO แทน o1 แต่สำหรับ Function Block ตัวแปร i1/o1 จะมีชื่อตัวแปรที่แตกต่างกันไปตามลักษณะของ Function Block นั้นๆ



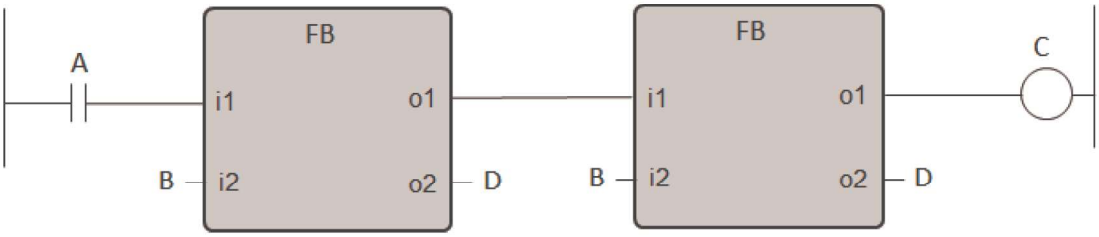
สำหรับ Input และ Output ตัวอื่นๆ ที่เหลือ ไม่ต้องลากเส้นไปที่ bus bar แต่ให้ป้อนเป็นพารามิเตอร์แทนซึ่งอาจเป็นค่าคงที่หรือตัวแปรก็ได้



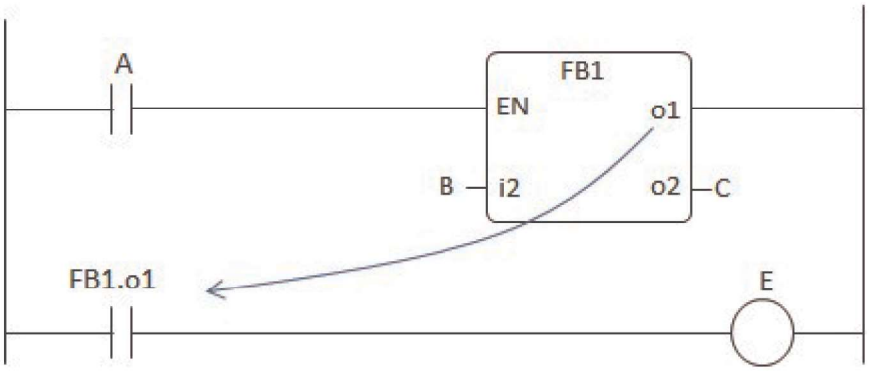
ถ้าไม่ต้องการใส่ Input หรือ Output ก็สามารถทำได้ แต่ต้องลากเส้นไปหา bus bar โดยตรง



การต่อ Cascade สามารถทำได้ เฉพาะ Input(In1) และ Output(Out1) เท่านั้นที่สามารถต่อถึงกันได้



สามารถนำเอา Input Variable หรือ Output Variable ไปใช้เขียนโปแกรมได้ โดยต้องอ้างชื่อ Instance ของ FB นั้นๆ แล้วตามด้วยชื่อของ Input Variable หรือ Output Variable ของ FB นั้นๆ ดังแสดงในตัวอย่างข้างล่างนี้จะเห็นว่า Ou1 ของ FB1 ถูกนำไปเขียนในโปรแกรมในจุดอื่นได้



8.6 ชนิดข้อมูล (Data Types)

ชนิดของข้อมูลที่ใช้ในการเขียนโปรแกรมบนซอฟแวร์ CCW

การจัดประเภท	ชนิดข้อมูล	ขนาดข้อมูล	ช่วงค่า
Bit strings	BYTE	1 byte	16#00 ถึง FF
	WORD	2 bytes	16#0000 ถึง FFFF
	DWORD	4 bytes	16#00000000 ถึง FFFFFFFF
	LWORD	8 bytes	16#0000000000000000 ถึง FFFFFFFFFFFFFFFF
Integers	SINT	1 byte	-128 ถึง +127
	INT	2 bytes	-32768 ถึง +32767
	DINT	4 bytes	-2147483648 ถึง +2147483647
	LINT	8 bytes	-9223372036854775808 ถึง +9223372036854775807
	USINT	1 byte	0 ถึง +255
	UINT	2 bytes	0 ถึง +65535
	UDINT	4 bytes	0 ถึง +4294967295
	ULINT	8 bytes	0 ถึง +18446744073709551615

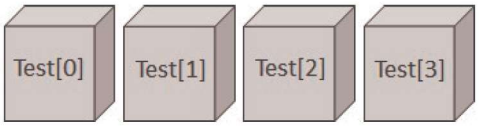
การจัดประเภท	ชนิดข้อมูล	ขนาดข้อมูล	ช่วงค่า
Real numbers	REAL	4 bytes	-3.402823e+38 ถึง -1.175495e-38 0 -1.175495e-38 ถึง 3.402823e+38 +∞ /-∞
	LREAL	8 bytes	-1.79769313486231e+308 ถึง -2.22507385850721e-308 0 2.22507385850721e-308 ถึง 1.79769313486231e+308 +∞ /-∞
Durations	LTIME	8 bytes	T#-9223372036854.775808ms (T#-106751d_23h_47m_16s_854.775808ms) ถึง T#+9223372036854.775807ms (T#+106751d_23h_47m_16s_854.775807ms)
Date	DATE	8 bytes	D#1970-01-01 ถึง D#2106-02-06 (January 1, 1970 ถึง February 6, 2106)
Character/ Character strings	CHAR WCHAR STRING WSTRING	1 byte 2 bytes	'abc'

8.6.1 ชนิดข้อมูลอาร์เรย์ (Array)

Array คือกลุ่มของข้อมูลที่เรียงลำดับกัน มีจำนวนแน่นอนซึ่งข้อมูลจะเป็นประเภทเดียวกัน ข้อมูลแต่ละตัวของอาร์เรย์จะเรียกว่า อีลีเมนต์(Element) และข้อมูลแต่ละอีลีเมนต์จะมีหมายเลขเพื่อใช้ในการอ้างอิงถึงเรียกตัวเลขนี้ ว่า เลขดัชนี (Index) จะเป็นตัวแปรที่ชื่อ เหมือนกัน แต่จะแตกต่างกันตรงหมายเลขดัชนี

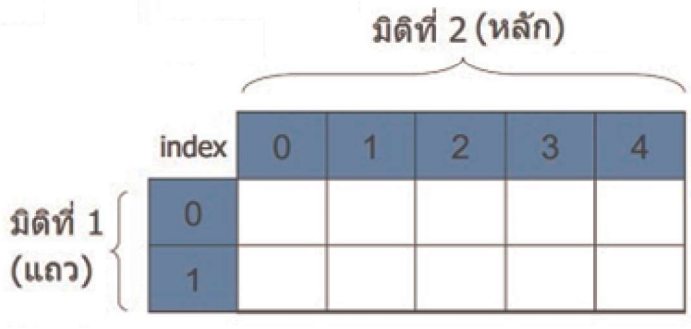
ตัวแปร Array 1 มิติ

การใช้ตัวแปร Array มีรูปแบบดังนี้ ประเภทตัวแปร ชื่อตัวแปร[จำนวนสมาชิก] เช่น ตัวแปรชื่อ Test[0..3]



ตัวแปร Array 2 มิติ

อาร์เรย์ 2 มิติ เป็นตัวแปรชุดที่มีการจัดการข้อมูล Row (แถว) , Column (หลัก) ซึ่งอยู่ในรูปแบบตาราง ที่มีแสดงตำแหน่ง 2 ตัว กล่าวคือ array 2 มิติ เป็น array ของ array 1 มิติ นั่นเอง ตัวตัวแปรอาร์เรย์ 2 มิติ Test[0..3 , 1..4]



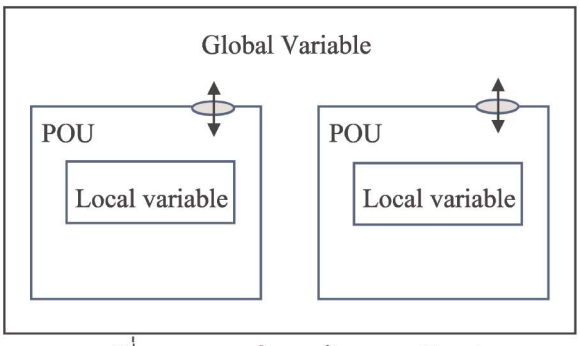
8.7 ประเภทตัวแปร (Variables)

ตัวแปร (Variables) ของ Micro800 สามารถแบ่งออกได้เป็น 2 ประเภทคือ

- User-defined Variables คือ ตัวแปรที่ผู้ใช้สร้างขึ้นเพื่อใช้งานในโปรแกรม ผู้ใช้สามารถกำหนดคุณสมบัติ (Attribute) ของตัวแปรได้ทั้งหมด
- System Variables คือตัวแปรที่ระบบกำหนดมาให้ใช้งานในฟังก์ชั่นต่างๆ

8.7.1 ตัวแปร User-defined Variables

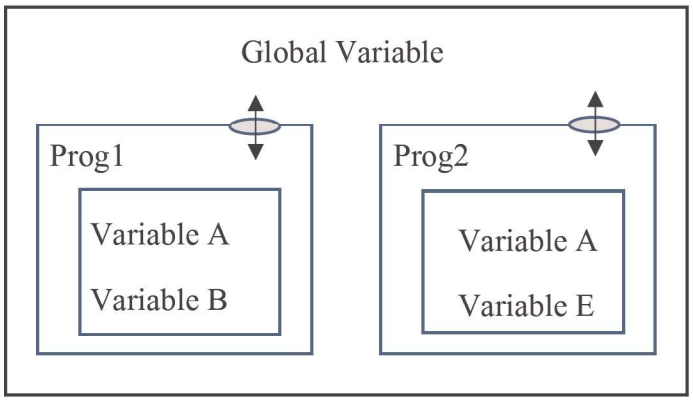
ตัวแปรที่เกี่ยวข้องกับ POU ประกอบด้วย Local Variables และ Global Variable



รูปที่ 8.6 แสดงโครงสร้างของตัวแปร

8.7.1.1 Local Variable คือตัวแปรที่สามารถใช้งานได้เฉพาะภายใน POU (Program, Function หรือ Function Block) เท่านั้น สมมุติว่าถ้าเราสร้าง Program1 และ Program2 ขึ้นมาเพื่อการใช้งาน แล้วประกาศตัวแปร Local ใน Program1 เราจะไม่สามารถอ้างถึงตัวแปรตัวนี้จาก Program2 ได้ เช่น มีตัวแปรชื่อ “Level_01” ใน Program1 จะไม่สามารถเชื่อมโยงตัวแปรชื่อ “Level_01” ใน Program2 ได้ (ถึงแม้จะเป็นชื่อเดียวกันก็ตาม)

ในรูปที่ 8.7 แสดงให้เห็นว่าเราสามารถตั้งชื่อ Local variable ใน Program1 และ Program2 เหมือนกันได้ แต่ตัวแปรทั้ง 2 ตัวนี้จะไม่มีความสัมพันธ์ต่อกัน ถ้าค่าตัวแปร A ใน Program1 เปลี่ยนแปลงจะไม่ทำให้ค่าของตัวแปร A ใน Program2 เปลี่ยนตามไปด้วย



รูปที่ 8.7 แสดงความเชื่อมโยงของตัวแปร Local

8.7.1.2 Global Variables

ตัวแปร Global เป็นตัวแปรที่สามารถเรียกใช้งานได้ทุกๆ POUs ไม่ว่าจะเป็น Program, Function และ FB นอกจากนั้นบางตัวแปรเช่น อินพุตเอาต์พุตในตัว (Built-in I/O) รวมทั้งยูนิตขยาย (Expansion I/O) ต่างๆ ด้วยจะถูกกำหนดชื่อมาจากโรงงาน รูปข้างล่างนี้แสดงตัวอย่าง Global variable ของอินพุตเอาต์พุตในตัว (Built-in I/O) ที่มาพร้อมกับ Micro800

Name	Alias	Data Type
_IO_EM_DO_00		BOOL
_IO_EM_DO_01		BOOL
_IO_EM_DO_02		BOOL
_IO_EM_DO_03		BOOL
_IO_EM_DO_04		BOOL
_IO_EM_DO_05		BOOL

รูปที่ 8.8 แสดงตารางตัวแปร Global ของอินพุตเอาต์พุต

8.7.2 System Variables

System Variables คือ ตัวแปรที่ระบบกำหนดมาให้ใช้งานกับ Micro800 ในฟังก์ชั่นต่างๆ มีลักษณะดังนี้

- ตัวแปรประเภทนี้ไม่สามารถเปลี่ยนชื่อได้
- ตัวแปรนี้ส่วนใหญ่จะขึ้นต้นด้วย “_” เช่น _SYSVA_FIRST_SCAN
- ตัวแปรบางตัวเป็น read-only หรือบางตัวเป็น read/write

8.8 ค่าคงที่ (Constant)

การระบุค่าคงที่ใน POU มีวิธีการระบุตามชนิดของข้อมูล (Data type) ดังนี้

- *Boolean Data*

รูปแบบ	ความหมาย
TRUE	1 (BOOL)
FALSE	0 (BOOL)

- *Bit Strings*

เป็นข้อมูลเลขฐานสิบหก ที่มีขนาดข้อมูล BYTE, WORD, DWORD และ LWORD ส่วน base ให้ระบุเป็นตัวเลข 2, 8, 10 และ16 ตัวอย่างเช่น *16#64 คือ ค่าคงที่ฐาน 16 มีค่าเท่ากับ 64(Hex)*

รูปแบบ	ตัวอย่าง	ความหมาย
{base}#{numeric_value}	16#0064	0064 (Hex)
{numeric_value}	100	

- *Integer Data*

เป็นข้อมูลเลขฐานสิบแบบเลขจำนวนเต็ม มีขนาดข้อมูล SINT, USINT, INT, UINT, DINT, UDINT, LINT และ ULINT

รูปแบบ	ตัวอย่าง	ความหมาย
{numeric_value}	-1	-1 (Dec) หรือ FFFF (Hex)

- *Real Numbers*

เป็นข้อมูลเลขฐานสิบที่มีทศนิยม มีขนาดข้อมูล REAL และ LREAL

base ระบุเป็นตัวเลข 10 เท่านั้น

รูปแบบ	ตัวอย่าง	ความหมาย
{numeric_value}	-5.43	-5.43 (REAL)

- *Durations*

เป็นข้อมูลช่วงเวลา ใช้อักษรนำหน้าเป็นแบบ TIME หรือ T

รูปแบบ	ตัวอย่าง	ความหมาย
T#{day}d{hour}h{minutes}m{seconds}s{milliseconds}ms	T#36m5s	36 นาที 5 วินาที

- *Dates*

เป็นข้อมูลวันที่ ใช้อักษรนำหน้าเป็น DATEหรือ D

รูปแบบ	ตัวอย่าง	ความหมาย
DATE#{year}-{month}-{day}	DATE#2019-1-12	วันที่ 12 มกราคม 2019
D#{year}-{month}-{day}	D#2019-1-12	

- *Text Strings*

เป็นข้อมูลตัวอักษร โดยใช้สัญลักษณ์ (‘) คร่อมตัวอักษรที่ต้องการ

รูปแบบ	ตัวอย่าง	ความหมาย
‘{String}’	‘I love dog’	I love dog

ในกรณีที่ใช้ตัวอักษรพิเศษต้องใส่อักษร $ นำหน้าตัวอักษรพิเศษ ตัวอย่างของการใส่ตัวอักษรพิเศษแสดงดังตารางข้างล่างนี้

ลำดับตัวอักษร	ความหมาย	ASCII (hex)	ตัวอย่าง
$$	'$' character	16#24	'I paid $$5 for this'
$'	apostrophe	16#27	'Enter $'Y$' for YES'
$L	line feed	16#0a	'next $L line'
$R	carriage return	16#0d	' llo $R He'
$N	new line	16#0d0a	'This is a line$N'
$P	new page	16#0c	'lastline $P first line'
$T	tabulation	16#09	'name$Tsize$Tdate'
$hh 1	any character	16#hh	'ABCD = $41$42$43$44'

"hh" คือ ค่า hexadecimal ของรหัส ASCII

สรุปท้ายบท

จากที่กล่าวมาในบทนี้เป็นส่วนหนึ่งของหลักการเขียนโปรแกรมตามมาตรฐาน IEC 61131-3 รวมถึงชนิดของข้อมูลและการนำไปประยุกต์ใช้งาน นอกจากนั้นการเขียนโปรแกรมตามมาตรฐานนี้จะให้ความสำคัญกับตัวแปรซึ่งผู้ใช้งานสามารถกำหนดขึ้นได้เอง เราจึงไม่จำเป็นต้องจดจำหมายเลขอินพุตเอาต์พุตและหน่วยความจำอีกต่อไป

Chapter

9

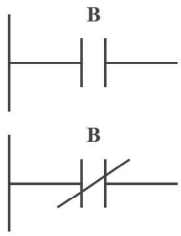
แลดเดอร์ไดอะแกรมและคำสั่งพื้นฐาน

แลดเดอร์ไดอะแกรม (Ladder Diagram) จัดเป็นภาษารูปภาพหรือสัญลักษณ์ที่เทียบเคียงมาจากวงจรรีเลย์ แต่เวลาที่ PLC ทำงานจะอาศัยชุดคำสั่ง (Instructions) ต่างๆโดยคำสั่งเหล่านี้จะถูกบันทึกลงในหน่วยความจำข้อมูลและในหน่วยความจำที่ว่านี้จะจัดเก็บเป็นรหัส (Code) ไม่สามารถจัดเก็บในสัญลักษณ์หรือรูปภาพของแลดเดอร์ไดอะแกรมได้โดยตรง

ดังนั้นผู้ใช้จึงจำเป็นต้องเรียนรู้และทำความเข้าใจชุดคำสั่งต่างๆ เพื่อนำมาประยุกต์ใช้งานและสร้างเงื่อนไขการควบคุมให้ได้ตามที่ต้องการ

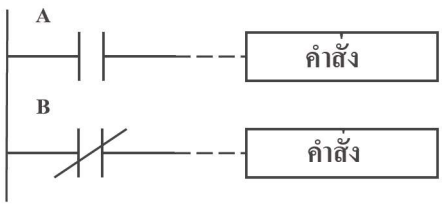
9.1 กลุ่มคำสั่งพื้นฐาน (Ladder Instruction & Output Control)

9.1.1 การใช้ Direct Contact และ Reverse Contact

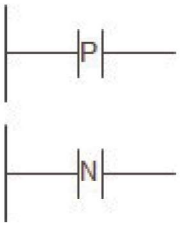


คำสั่ง Direct Contact จะทำงาน (หน้าคอนแทคต่อถึงกัน) เมื่อลอจิกของบิตนั้นเป็น True ส่วน Reverse Contact จะทำงาน (หน้าคอนแทคต่อถึงกัน) เมื่อลอจิกของบิตนั้นเป็น False

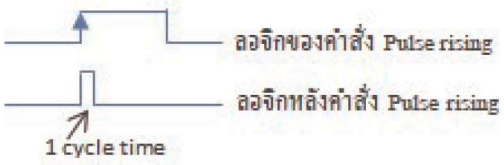
ตัวอย่างที่ 9.1 การเขียนแลดเดอร์ไดอะแกรมคำสั่ง Direct Contact และ Reverse Contact



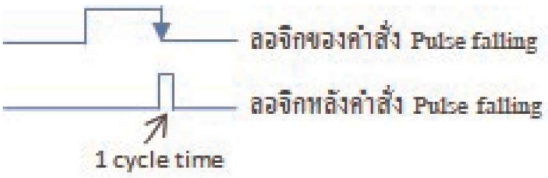
9.1.2 การใช้คอนแทค Pulse Rising Edge และ Pulse Falling Edge



รูปข้างล่างนี้แสดงการทำงานของคอนแทค Pulse rising edge จากรูปจะเห็นว่าเมื่อลอจิกของคำสั่งเปลี่ยนจาก False ไปเป็น True มันจะทำให้ลอจิกที่อยู่หลังคำสั่งหรือคอนแทคนี้มีลอจิก True เป็นเวลา 1 วงรอบการสแกน (cycle time)



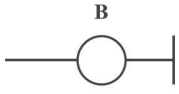
ส่วนรูปข้างล่างนี้แสดงการทำงานของคอนแทค Pulse falling edge จากรูปจะเห็นว่าเมื่อลอจิกของคำสั่งเปลี่ยนจาก True ไปเป็น False มันจะทำให้ลอจิกที่อยู่หลังคำสั่งหรือคอนแทคนี้มีลอจิก True เป็นเวลา 1 วงรอบการสแกน (cycle time)



9.1.3 การใช้คำสั่ง Direct Coil กับ Reverse Coil

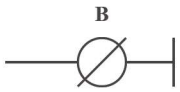
เป็นคำสั่งที่สั่งให้บิตนั้นๆทำงานซึ่งอาจเป็นเอาต์พุตภายนอกหรือบิตภายในก็ได้

สัญลักษณ์ Direct Coil



ตัวอย่างที่ 9.2 รูปแบบของแลดเดอร์ไดอะแกรม เมื่อลอจิก A เป็น True จะทำให้บิตของ Direct Coil เป็น True ด้วย

สัญลักษณ์ Reverse Coil



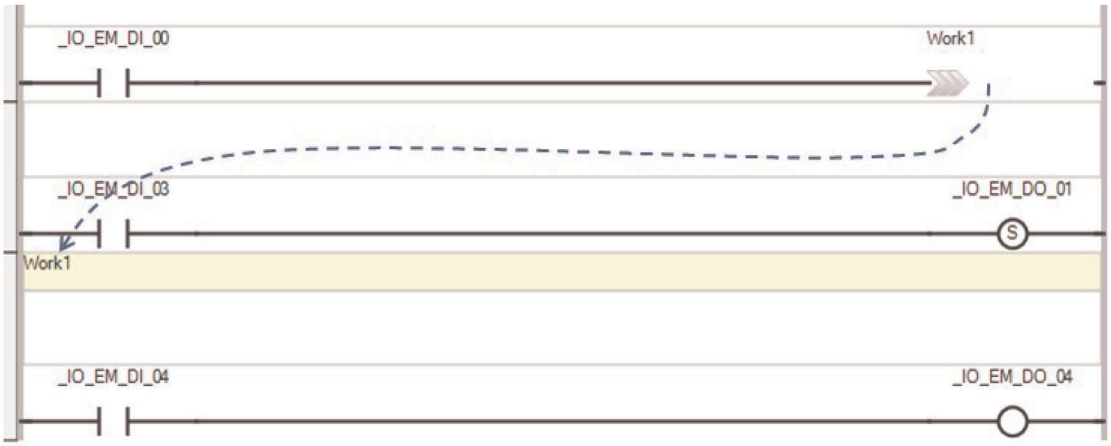
ตัวอย่างที่ 9.3 แลดเดอร์ไดอะแกรมของ Reverse Coil ถ้าลอจิกของ A เป็น False ลอจิกของ B จะเป็น True ในทางตรงกันข้ามถ้าลอจิกของ A เป็น True ลอจิกของ B จะเป็น False

9.2 กลุ่มคำสั่ง Program Control Instruction

9.2.1 การใช้งาน Label

ถ้าต้องการให้โปรแกรมข้ามหรือกระโดดไปทำงานในจุดอื่นของโปรแกรม เราสามารถทำได้ด้วยการใช้ Label จากตัวอย่างข้างล่างนี้ เมื่อเงื่อนไขของ >>Work1 มีสถานะเป็น True โปรแกรมจะข้ามไปทำงานในบรรทัดที่มี label “Work1” กำกับอยู่ ส่วนหน้าคอนแทค _IO_EM_DI_03 จะไม่ทำงานและอยู่ในสภาวะคงสถานะ(Retained)

แต่ถ้าเงื่อนไขของ >>Work1 มีสถานะเป็น False โปรแกรมจะทำงานตามปกติ โดยจะประมวลผลโปรแกรมของวงจรที่มี _IO_EM_DI_03 และ _IO_EM_DO_01 ด้วย

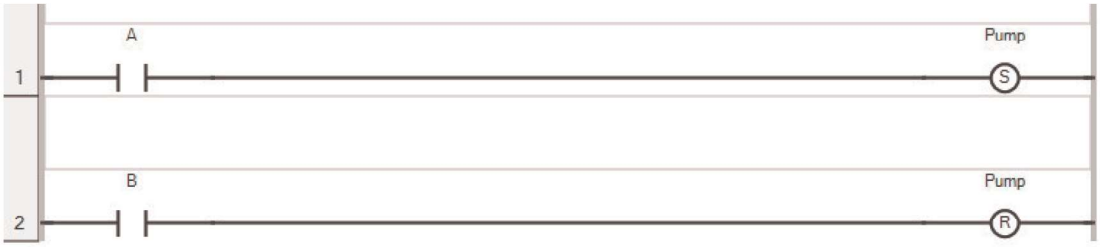


9.3 คำสั่งในกลุ่ม Sequence Output

9.3.1 การใช้คำสั่งเซต (S) และรีเซต (R)

คำสั่ง S จะทำให้บิตที่ถูกสั่งมีสถานะเป็น True (ON) และคงสถานะอยู่เช่นนั้นจนกว่าจะมีคำสั่ง R ที่บิตเดียวกัน บิตนั้นจึงจะสถานะเป็น False (OFF) หรือมีคำสั่งอื่นที่มีผลต่อบิตนั้นๆ

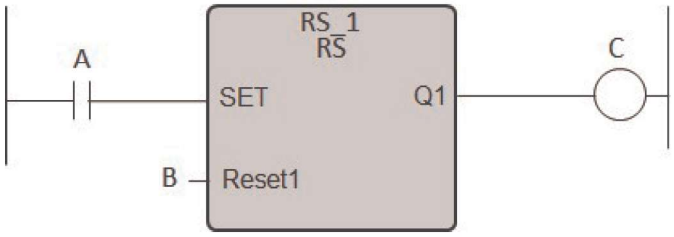
ตัวอย่างที่ 9.4 แสดงการใช้งานคำสั่ง S และ R ถ้าลอจิกของ A เป็น True ลอจิกของ Pump จะเป็น True และยังคงเป็น True แม้ลอจิกของ A จะเป็น False แล้วก็ตาม ถ้าลอจิกของ B เป็น True ลอจิกของ Pump จะเป็น False เพราะคำสั่ง R จะรีเซ็ทบิต Pump

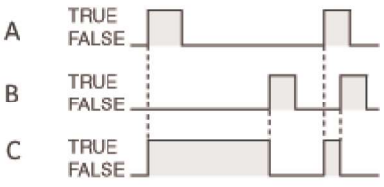


9.3.2 การใช้คำสั่ง RS และ SR

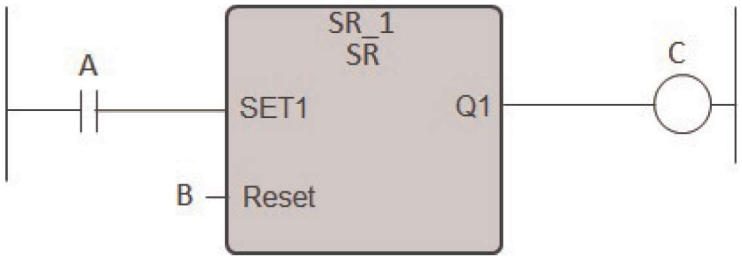
การทำงานของคำสั่ง RS และ SR จะคล้ายๆ กับคำสั่ง S(Set) และ R(Reset) เพียงแต่ขาสัญญาณ Set/Reset จะรวมอยู่ในคำสั่งเดียวกันเพื่อความสะดวกในการใช้งาน แต่คำสั่ง RS จะให้ความสำคัญกับ Reset มากกว่า Set ในทางตรงข้ามกันคำสั่ง SR จะให้ความสำคัญของ Set มากกว่า Rcsct ลองพิจารณา Timing Chart ประกอบ

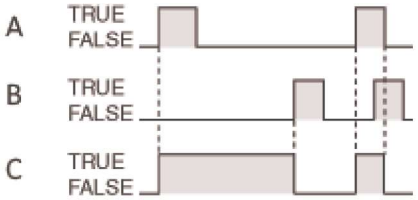
คำสั่ง RS





คำสั่ง SR



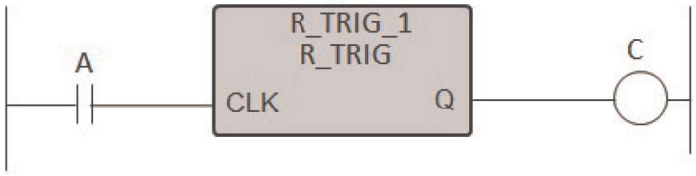


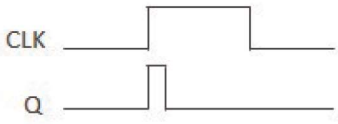
9.4 คำสั่งในกลุ่ม Sequence Input

9.4.1 การใช้คำสั่ง R_TRIG (UP) และ F_TRIG (DOWN)

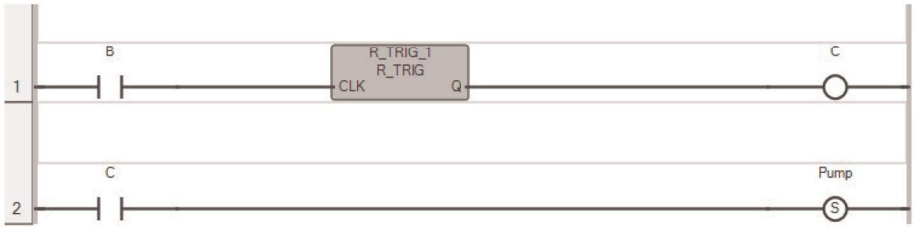
คำสั่ง R_TRIG (UP) และ F_TRIG (DOWN) จะเป็นคำสั่งที่ทำงานเมื่อเกิดขอบขาขึ้นหรือขอบขาลงของสัญญาณอินพุตเท่านั้น และจะทำงานเพียงช่วงเวลา 1 Cycle time หรือ Task period เท่านั้น (R=Rising, F=Falling) โดยไม่สนใจว่าสัญญาณอินพุตจะยังคงมีสถานะเป็น True อยู่หรือไม่ก็ตาม

คำสั่ง R_TRIG และ Timing chart

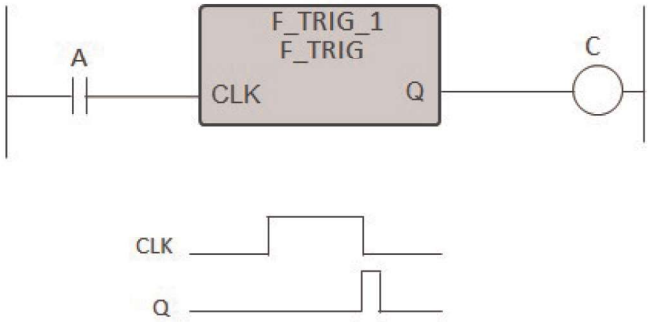




ตัวอย่างที่ 9.5 การเขียนแลดเดอร์ของคำสั่ง R_TRIG



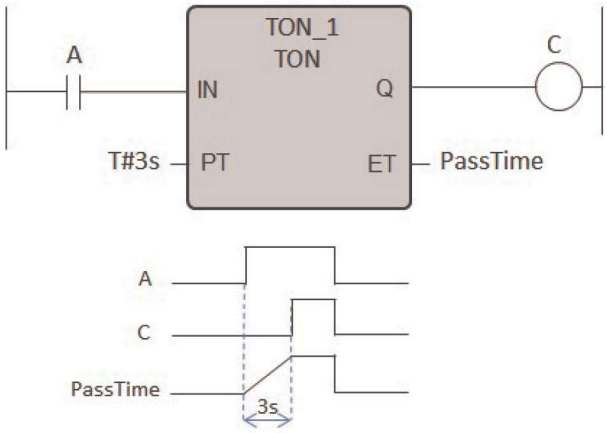
คำสั่ง F_TRIG และ Timing chart



9.5 คำสั่งในกลุ่ม Timer/Counter

9.5.1 การใช้คำสั่ง TON

จากรูปเมื่อลอจิก IN ของคำสั่ง TON เป็น True ไทม์เมอร์จะเริ่มนับเวลา และเอาต์พุต (Q)จะ ON เมื่อนับเวลาครบ



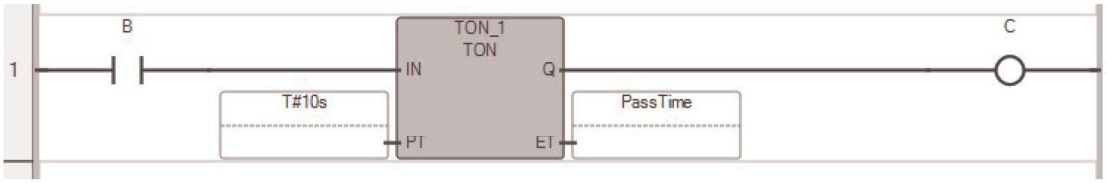
IN = Timer Input คือ สัญญาณเริ่มทำงาน จะเริ่มนับเวลาเมื่อเงื่อนไขเป็น True (ON)

PT = Preset Time คือ ค่าตั้งเวลาที่ต้องการนับ มีช่วงเวลาที่กำหนดได้ดังนี้

T#0ms ถึง T#9d17h

Q = Timer Output คือ เอาท์พุตจะเป็น True (ON) เมื่อ timer นับครบเวลาตามที่ให้ค่าไว้ใน PT

ET = Elapsed time คือ เวลาที่ผ่านไปหลังจาก timer เริ่มนับเวลา

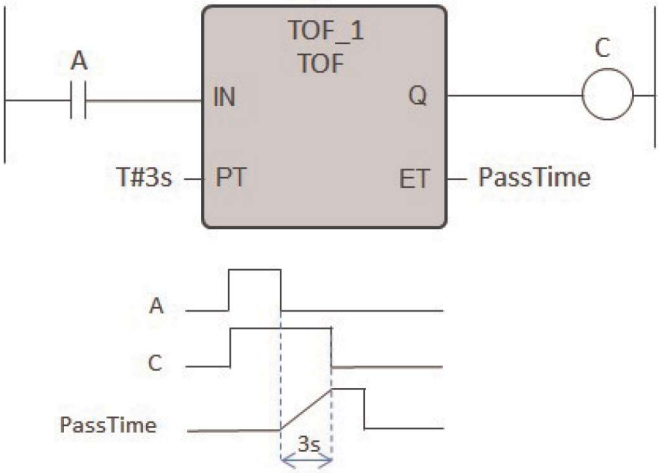


จากรูปข้างบน เมื่อมีสัญญาณสั่งให้ Timer ทำงาน (B มีสถานะ “ON”) คำสั่ง Timer จะเริ่มนับเวลาตามค่าที่ตั้งไว้ใน PT ในที่นี้คือ 10 วินาที เมื่อนับครบเวลาสัญญาณ Q หรือ Timer

Output ก็จะ “ON” ทำให้ลอจิกของ C เป็น True แต่ถ้าสัญญาณที่สั่งให้ Timer เปลี่ยนเป็น False (Contact B มีสถานะ OFF) Timer จะถูกรีเซท

9.5.2 การใช้คำสั่ง TOF

เอาท์พุต(Q) ของคำสั่ง TOF จะ ON ทันทีที่ลอจิกของ IN เป็น True และจะเริ่มนับเวลาเมื่อลอจิก IN (A) เปลี่ยนจาก True เป็น False และเอาต์พุต(Q) จะ OFF เมื่อนับเวลาครบ



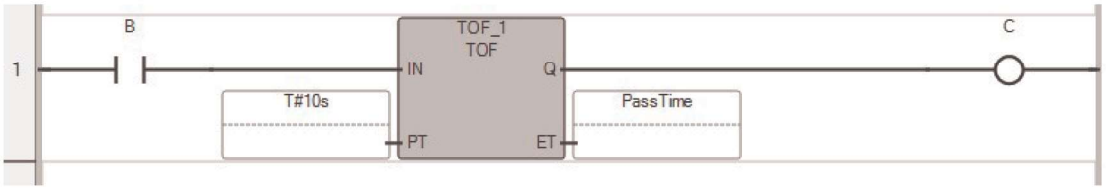
IN = Timer Input คือ สัญญาณเริ่มทำงาน จะนับเวลาเมื่อเปลี่ยนจาก True เป็น False

PT = Preset Time คือ ข้อมูลเวลาที่ต้องการนับ มีช่วงเวลาที่กำหนดได้ดังนี้

T#0ms ถึง T#9d17h

Q = Timer Output คือ สัญญาณเอาท์พุตจะ ON ทันทีเมื่อ In = True และยังคง ON ต่อไปจนกระทั่งมีการเริ่มนับเวลาจนครบตามเวลาที่ตั้งไว้ใน PT

ET = Elapsed time คือ เวลาที่ผ่านไปหลังจาก timer เริ่มนับเวลา

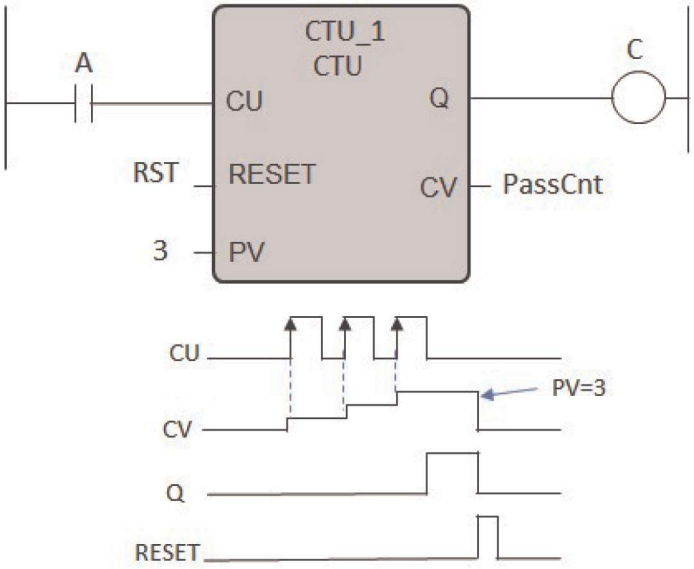


จากรูปข้างบน เมื่อมีสัญญาณสั่งให้ Timer ทำงาน (B มีสถานะ “ON”) สัญญาณ Q หรือ Timer Output ก็จะ “ON” ทันที เมื่อสัญญาณอินพุต(IN)หายไป (B มีสถานะ “OFF”) Timer จะเริ่มนับเวลาตามค่าที่ตั้งไว้ใน PT เมื่อนับครบเวลาแล้ว สัญญาณ Q หรือ Timer Output ก็จะ “OFF”

ถ้าสัญญาณอินพุต “ON” อีกครั้ง (B มีสถานะ “ON”) Timer จะถูกรีเซทพร้อมกับทำงานในรอบถัดไป

9.5.3 การใช้คำสั่ง COUNTER-CTU (นับขึ้น)

เป็นคำสั่งที่ใช้นับจำนวนครั้งของสัญญาณอินพุตที่ ON ซึ่งคำสั่งนี้จะนับขึ้นจากศูนย์ไปจนถึงค่าที่ตั้งไว้ที่ PV (Preset Value)



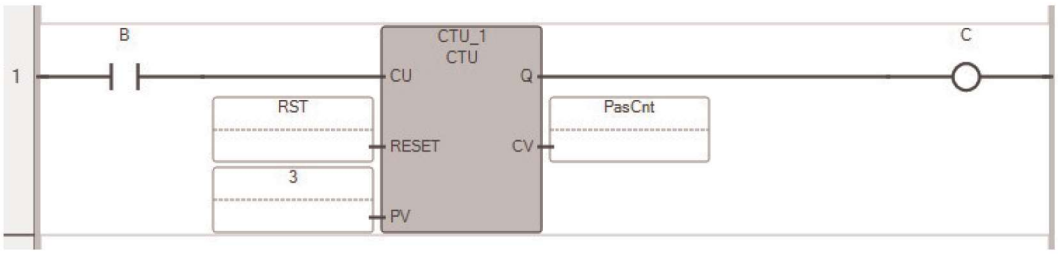
CU = UP Counter Input คือ จะนับเวลาเมื่อสัญญาณเปลี่ยนจาก False เป็น True

PV = Preset Value สามารถตั้งค่าจำนวนนับได้ 0 ถึง 32767 ครั้ง

Q = Counter Output เมื่อ Counter นับครบตามจำนวนที่ตั้งค่าไว้เอาต์พุตจะทำงาน

CV = Counter Value คือจำนวนครั้งที่นับได้

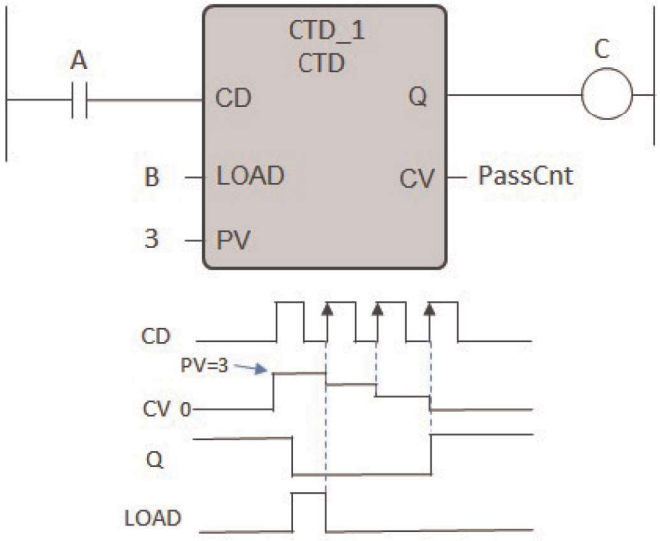
RESET = ใช้สำหรับรีเซ็ตค่านับและเอาต์พุตของ Counter



จากรูปข้างบนแสดงตัวอย่างคำสั่ง Counter-CTU ที่ตั้งค่า PV=3 ถ้า B มีสถานะเป็น True (ON) และ False (OFF) ครบ 3 ครั้ง จะทำให้ C มีสถานะเป็น True (ON)

9.5.4 การใช้คำสั่ง COUNTER-CTD (นับลง)

คำสั่ง CTD เป็น Counter แบบนับลงได้ในตัว สัญญาณ CD (Down Counter Input) เป็นสัญญาณแบบนับลง



CD = Down Counter Input คือขานับสัญญาณแบบนับลง

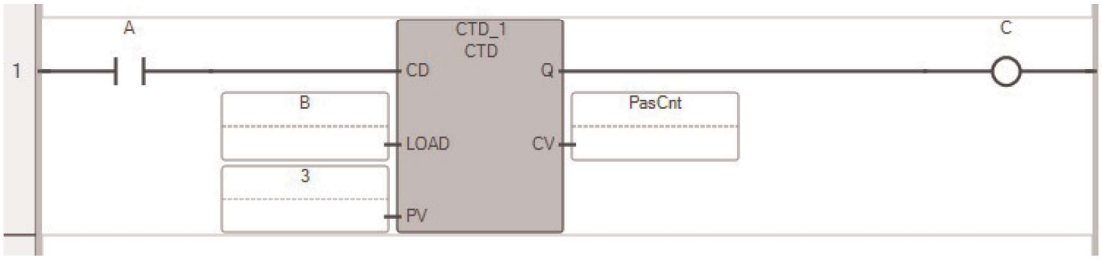
PV = (Preset Value) ค่าตั้งจำนวนนับ ใช้กำหนดว่าจะให้ Counter นับสัญญาณอินพุตเป็นจำนวนกี่ครั้ง เอาต์พุตของ Counter จะทำงานเมื่อนับครบ ซึ่งสามารถตั้งค่าจำนวนนับได้ 0 ถึง 32767 ครั้ง

Q = Counter Output Down จะทำงานเมื่อ Counter นับลงไปจนถึงศูนย์

CV = Counter Value แสดงค่าจำนวนนับได้

LOAD = สถานะเปลี่ยนจาก False เป็น True จะรีเซ็ตค่านับและเอาต์พุตของ Counter
สถานะเปลี่ยนจาก True เป็น False จะทำให้ Counter เริ่มนับใหม่

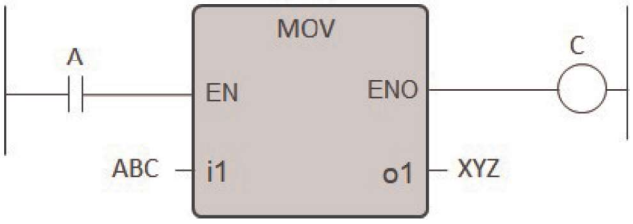
จากรูปข้างล่างนี้แสดงตัวอย่างคำสั่ง Counter-CTD ที่ตั้งค่า PV=3 ถ้า A มีสถานะเป็น True (ON) และ False (OFF) ครบ 3 ครั้ง จะทำให้ C มีสถานะเป็น True (ON)



9.6 คำสั่งในกลุ่ม Data Movement

9.6.1 การใช้คำสั่ง MOVE

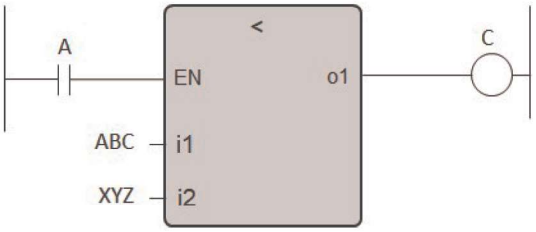
เมื่อคำสั่งนี้ทำงานมันจะทำการสำเนา (Copy) ข้อมูลจาก i1 ไปยัง o1 โดยที่ i1 ยังคงมีข้อมูลเดิมอยู่



เมื่อ EN เป็น TRUE คำสั่ง Move จะสำเนาข้อมูลจาก Input ไปที่ Output

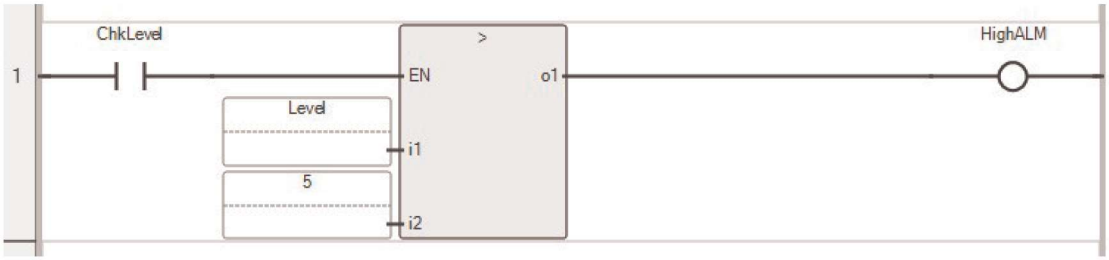
9.7 คำสั่งในกลุ่ม Comparison

9.7.1 การใช้คำสั่ง <, <=, >, และ >=



คำสั่งเหล่านี้ทำหน้าที่เปรียบเทียบข้อมูลของ Input โดยใช้ Input ตัวแรกเปรียบเทียบกับตัวถัดไปตัวอย่างเช่น การเปรียบเทียบน้อยกว่า (<) ของ i1และ i2 ถ้า i1 น้อยกว่า i2 คำสั่งนี้จะให้ลอจิกเอาท์พุตเป็น True

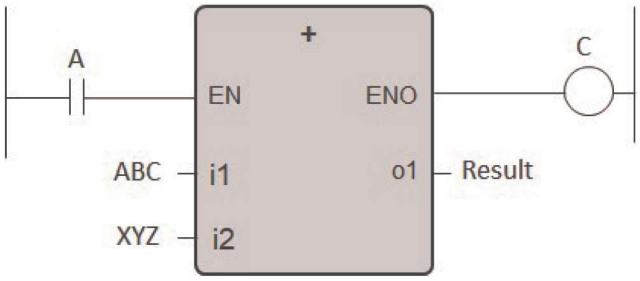
ตัวอย่างที่ 9.6 การใช้คำสั่ง > จากรูปข้างล่างนี้ข้อมูลที่ i1 คือ Level เป็นข้อมูลที่ได้รับมาจากอนาลอกอินพุตที่ใช้วัดระดับน้ำในถัง และนำมาเปรียบกับ i2 ซึ่งเป็นค่าคงที่จำนวนเต็ม (INT) ซึ่งมีค่าเท่า 5 ถ้าค่าของ Level มากกว่า 5 คำสั่งนี้จะสั่งให้เอาต์พุต (o1) เป็น True และส่งผลให้เอาท์พุต HighALM ทำงาน (ON)



9.8 คำสั่งในกลุ่มคำนวณคณิตศาสตร์ (Math)

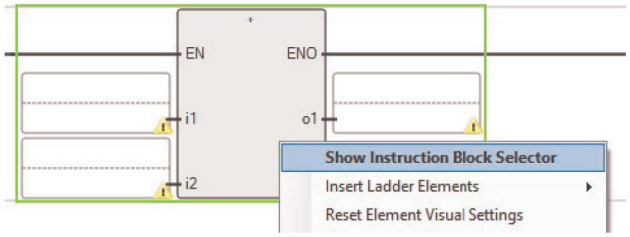
คำสั่งคำนวณคณิตศาสตร์มีให้เลือกใช้งานหลากหลายคำสั่งเช่น +, -, *, / และ ABS เป็นต้น การคำนวณทางคณิตศาสตร์สามารถทำได้เหมือนการคำนวณทางคณิตศาสตร์ปกติทั่วไปที่สามารถบวกเลขที่เป็นจำนวนจริง และจำนวนเต็มได้เลย หรือเอาข้อมูลประเภท String มาบวกกันก็ได้ ในที่นี้เราจะขอยกตัวอย่างเพียง 2 คำสั่งเท่านั้น ถ้าต้องการศึกษาเพิ่มเติมให้ดูได้จากคู่มือ

9.8.1 การใช้คำสั่ง + (บวก)

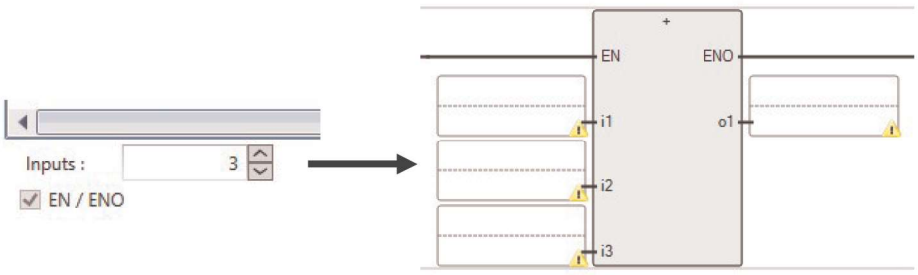


คำสั่งนี้ทำหน้าที่บวกเลข 2 จำนวนหรือมากกว่า โดยคำสั่งจะนำตัวเลขที่ i1 และ i2 มาบวกกัน แต่ถ้ามีมากกว่า 2 จำนวน เราสามารถเพิ่ม i3 หรือ i4 ได้ ทำให้ไม่ต้องใช้คำสั่ง + หลายๆ คำสั่ง

เราสามารถเพิ่มข้อมูลที่ใช้ในการบวกโดยการคลิกขวาที่คำสั่ง + จากนั้นให้เลือก Show Instruction Block Selector ดังแสดงในรูปข้างล่างนี้

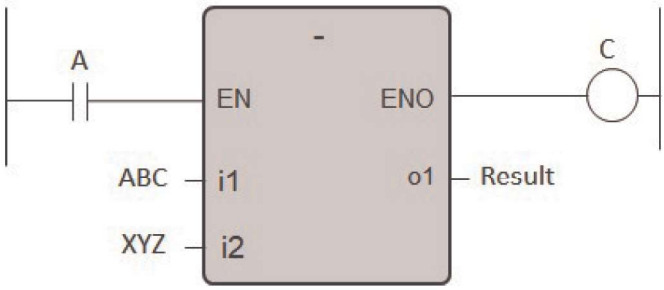


จะปรากฏหน้าต่าง Instruction Block Selector ขึ้นมา ให้เปลี่ยนจำนวน Inputs ตามที่ต้องการ ซึ่งจะอยู่ด้านล่างซ้ายมือของหน้าต่าง Instruction Block Selector ในตัวอย่างนี้จะป้อนเลข 3 จากนั้นคลิก OK เราก็จะได้คำสั่ง + ที่สามารถบวกได้ครั้งละ 3 จำนวน

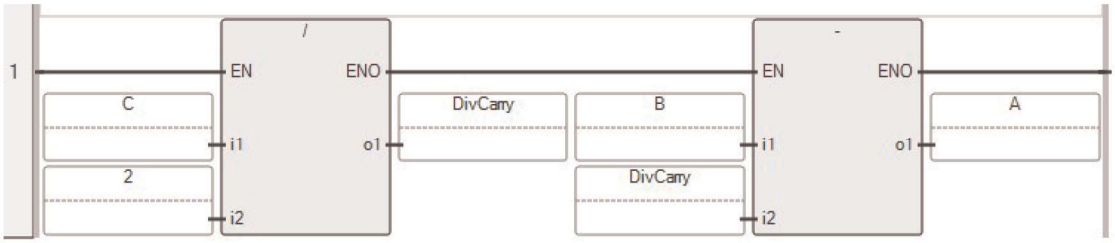


9.8.2 การใช้คำสั่ง – (ลบ)

คำสั่งนี้ทำหน้าที่ลบเลข 2 จำนวน โดยคำสั่งจะนำตัวเลขที่ i1 ตั้งและลบด้วย i2

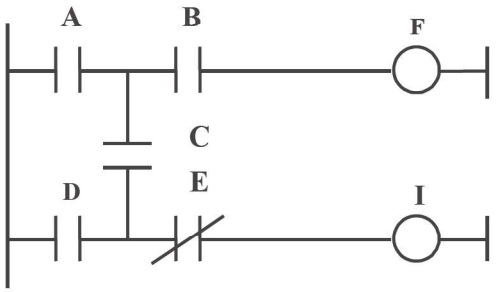


ตัวอย่างที่ 9.7 การใช้คำสั่ง Math จากรูปข้างล่างนี้เป็นตัวอย่างแลดเดอร์ไดอะแกรมในการคำนวณ โดยใช้สูตร A = (B – C/2)

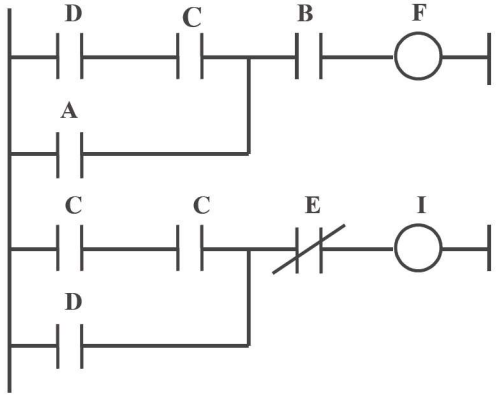


9.9 ข้อกำหนดในการเขียนแลดเดอร์ไดอะแกรม

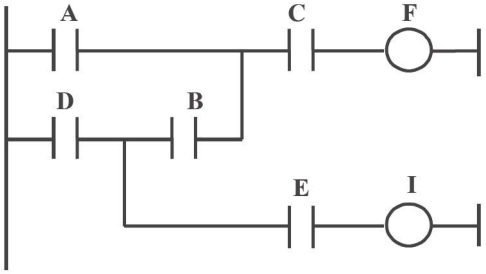
9.9.1 ไม่สามารถเขียนโปรแกรมตามแลดเดอร์ไดอะแกรมข้างล่างนี้ได้ จำเป็นต้องแปลงเป็นแลดเดอร์ไดอะแกรมใหม่เสียก่อน



เราสามารถเขียนแลดเดอร์ไดอะแกรมได้ใหม่ดังรูปข้างล่างนี้และยังคงทำงานเหมือนเดิม

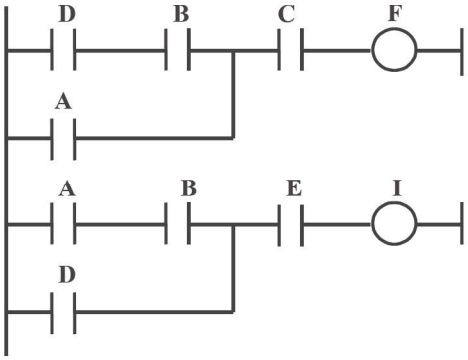


9.9.2 การอ่านแลดเดอร์ไดอะแกรมของ PLC จะพิจารณาการทำงานจากซ้ายไปขวาเท่านั้น ดังตัวอย่างเช่น



แลดเดอร์ไดอะแกรม A

จากแลดเดอร์ไดอะแกรม A ถ้าหน้าสัมผัส A, B และ E มีสถานะเป็น True(ON) ก็ไม่สามารถทำให้เอาต์พุต I นั้นเป็น True (ON) ได้เลย ดังนั้นถ้าต้องการใช้มันใช้งานได้จะต้องทำการจัดโปรแกรมเสียใหม่เพื่อให้การทำงานกระทำจากซ้ายไปขวาดังรูปแลดเดอร์ไดอะแกรม B



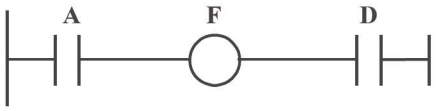
แลดเดอร์ไดอะแกรม B

9.9.3 จำนวนหน้าคอนแทคทั้ง NO และ NC ของอินพุต/เอาต์พุต, รีเลย์และไทม์เมอร์(TON)/เคาน์เตอร์ (CTU) จะนำมาเขียนโปรแกรมเป็นจำนวนเท่าใดก็ได้ตามความประสงค์ของผู้ใช้ทั้งนี้จะขึ้นอยู่กับหน่วยความจำโปรแกรมของ PLC ตัวนั้นๆ

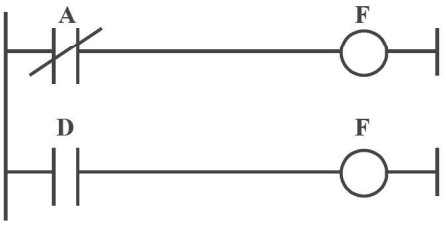
9.9.4 เมื่อต้องการให้เอาต์พุต ON ตลอดเวลาเราสามารถต่อคอยล์เอาต์พุตได้โดยตรงกับบัสบาร์

9.9.5 จำนวนหน้าสัมผัสที่ใช้ในการต่ออนุกรม หรือขนานไม่มีขีดจำกัดจะใช้เท่าใดก็ได้ขึ้นอยู่กับความต้องการของผู้ใช้ทั้งนี้จะขึ้นอยู่กับหน่วยความจำโปรแกรม

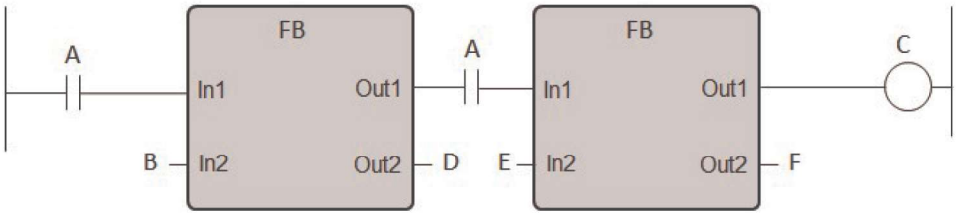
9.9.6 ไม่สามารถเขียนโปรแกรมให้หน้าสัมผัสอยู่ตำแหน่งหลังจากคอยล์ได้



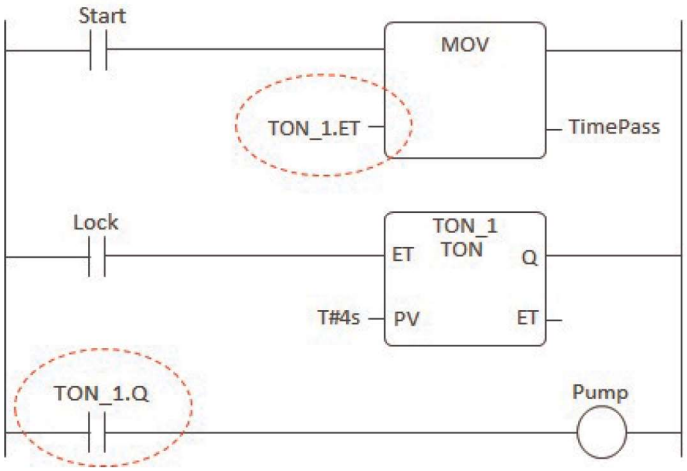
9.9.7 ไม่ควรเขียนโปรแกรมให้มีเอาต์พุตเดียวกันซ้ำหลายๆ ตำแหน่ง



9.9.8 สามารถเขียนโปรแกรมให้มี Function หรือ FB ในลักษณะอนุกรมกันได้



9.9.9 สามารถนำเอา Input Variable และ Output Variable ของ FB มาเขียนโปรแกรมได้ จากรูปข้างล่างนี้ได้นำเอา TON_1.Q (Boolean) กับ TON_1.ET (Time) มาเขียนในโปรแกรมแลดเดอร์ ในกรณีของ TON_1.ET จะถูกสำเนาค่าด้วยคำสั่ง MOV ไปที่ TimePass ดังนั้น TimePass จึงต้องเป็นตัวแปรชนิด Time เช่นเดียวกับ TON_1.ET



สรุปท้ายบท

การเรียนรู้การทำงานของคำสั่งแต่ละคำสั่งนั้นสำคัญมาก เปรียบเสมือนการเรียนวิชาภาษาอังกฤษที่เราต้องรู้คำศัพท์ก่อนจากนั้นจึงรวบรวมเป็นประโยคได้ ในทำนองเดียงกันการเขียนโปรแกรม PLC จึงจำเป็นต้องนำเอาคำสั่งต่างๆ มาเขียนร่วมกันเพื่อให้เครื่องจักรทำงานตามเงื่อนไข

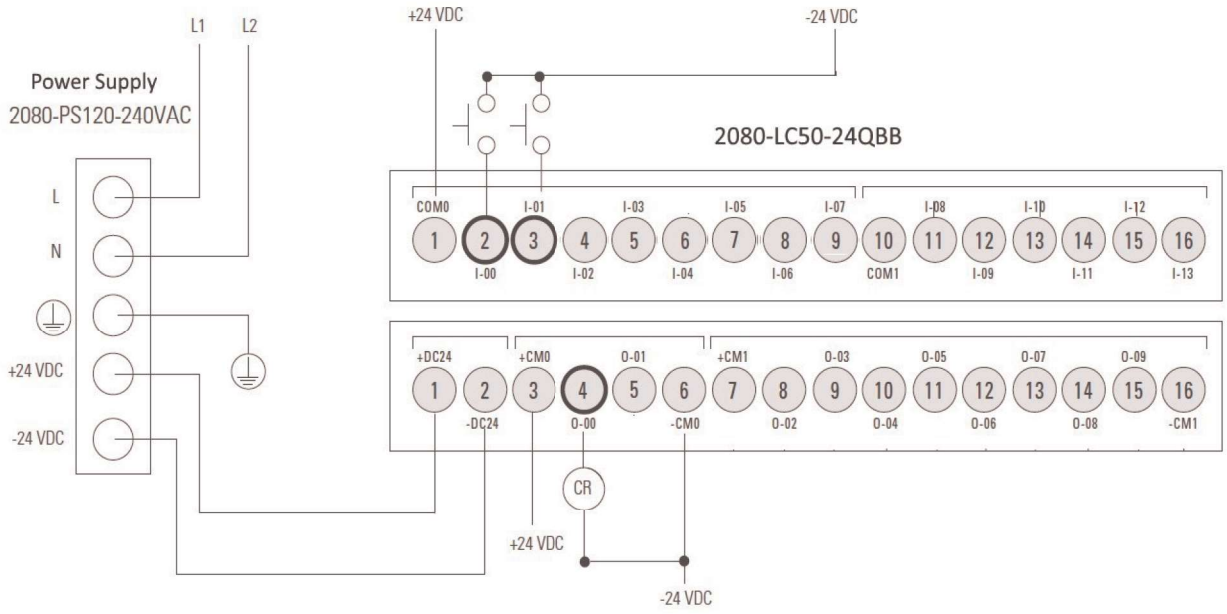
Chapter

10

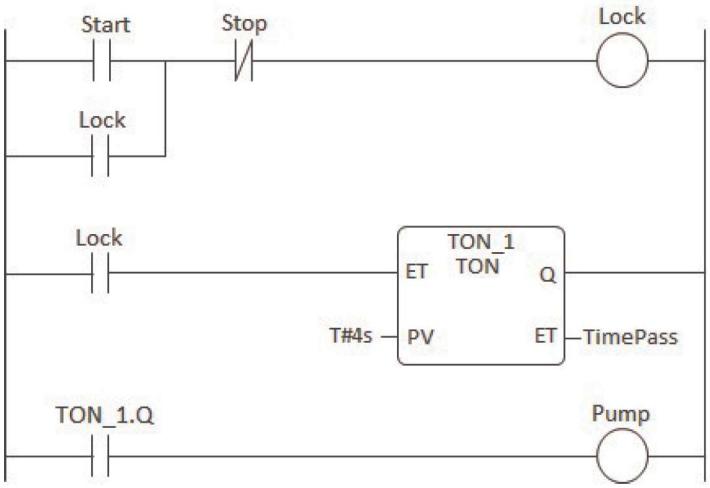
การใช้งานซอฟต์แวร์ CCW

ในบทที่ผ่านมาได้กล่าวถึงหลักการเขียนโปรแกรมและคำสั่งพื้นฐานต่างๆ เพื่อนำไปประยุกต์ใช้งานจริงได้อย่างเหมาะสม ต่อจากนี้ไปถึงเวลาเริ่มต้นการเขียนโปรแกรมโดยการใช้ซอฟต์แวร์ Connected Components Workbench(CCW)

ในรูปที่ 10.1 แสดงตัวอย่างวงจรการต่อของ Micro850 รุ่น 24 I/O ที่มีอินพุตเป็นสวิตช์ 2 ตัว ซึ่งต่อกับอินพุตหมายเลข I-00 (สวิตช์ Start) และ I-01(สวิตช์ Stop) และมีรีเลย์ 1 ตัว(CR) ต่อกับเอาต์พุต O-00 (Pump)



รูปที่ 10.1 แสดงตัวอย่างวงจรการต่ออินพุตเอาต์พุตของ Micro850



Global Variable (I/O)	Alias	Data Type
_IO_EM_DO_00	Pump	BOOL
_IO_EM_DI_00	Start	BOOL
_IO_EM_DI_01	Stop	BOOL

Local Variable	Data Type
Lock	BOOL
TimePass	INT

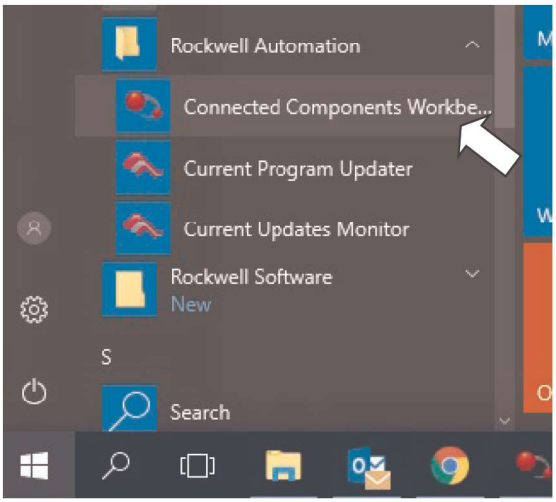
รูปที่ 10.2 แสดงตัวอย่างแลดเดอร์ไดอะแกรม

รูปที่ 10.2 เป็นแลดเดอร์ไดอะแกรมและตาราง I/O ที่เราจะใช้เป็นตัวอย่างในการเขียนโปรแกรมด้วยซอฟต์แวร์ CCW โดยในตัวอย่างนี้จะมีการใช้คำสั่งไทมเมอร์ร่วมด้วย

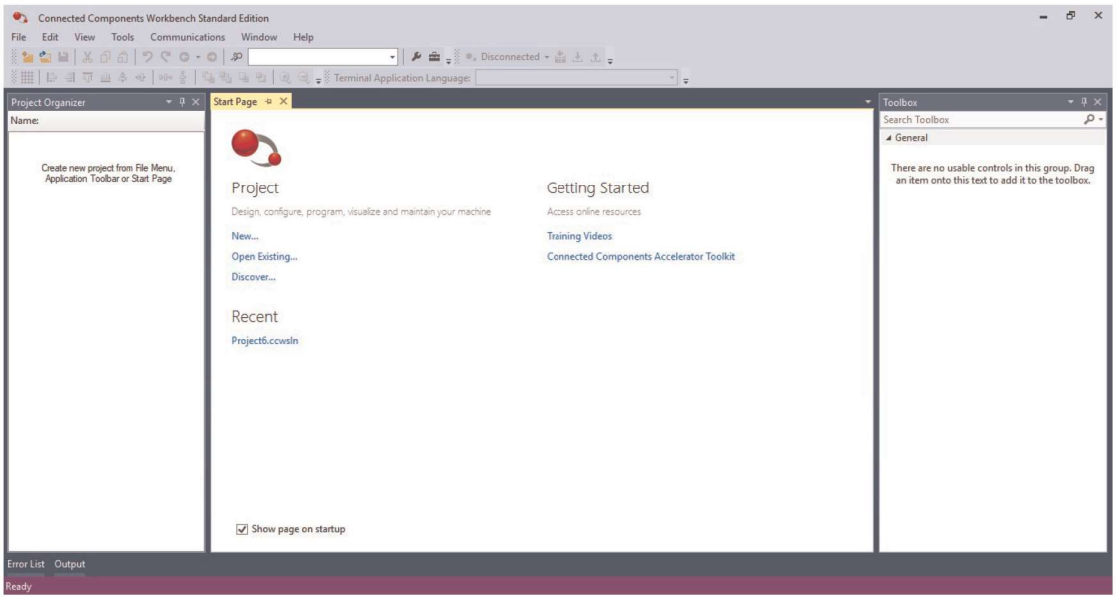
10.1 การเรียกใช้งานซอฟต์แวร์ CCW

การเปิดใช้งานซอฟต์แวร์ CCW นั้นสามารถทำได้โดยเลือก

Start-> Programs-> Rockwell Automation-> Connected Components Workbench

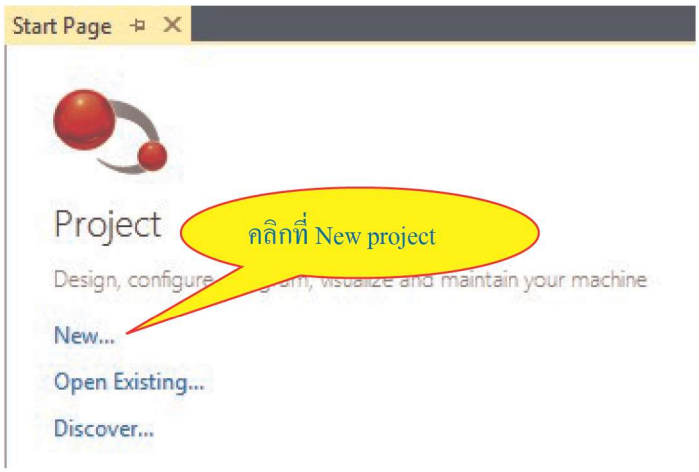


หลังจากนั้นซอฟต์แวร์ CCW จะถูกเรียกใช้งาน เมื่อเข้าสู่ซอฟต์แวร์แล้วจะมีหน้าต่างหลักปรากฏขึ้นมา ดังรูปด้านต่อไปนี้

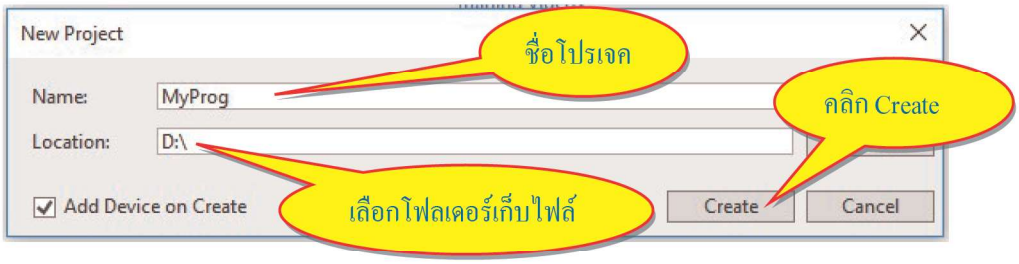


10.2 ขั้นตอนสำหรับการสร้าง Project ใหม่ (Create New Project)

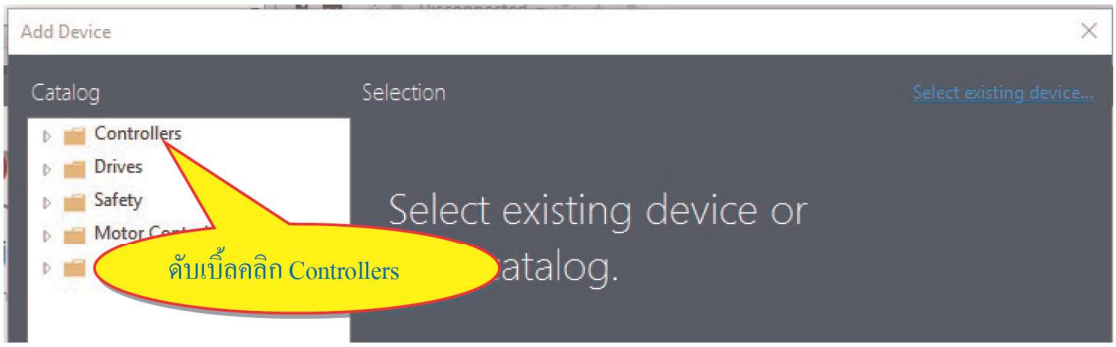
1. คลิกที่ New Project เพื่อสร้างโปรเจคใหม่



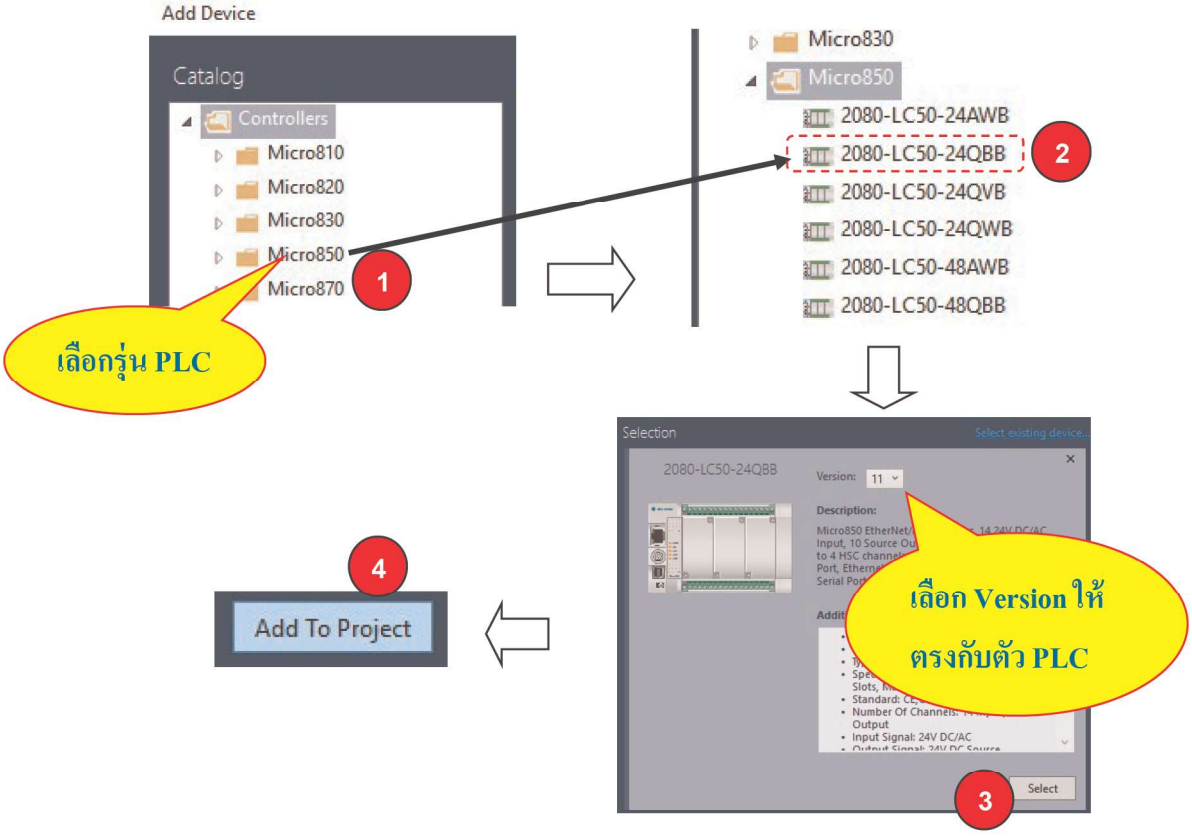
2. จะมีหน้าต่าง New Project ขึ้นมาให้ป้อนรายละเอียดดังต่อไปนี้ ใส่ชื่อโปรเจคและเลือกโฟลเดอร์เก็บไฟล์ตามที่ต้องการ จากนั้นคลิก Create



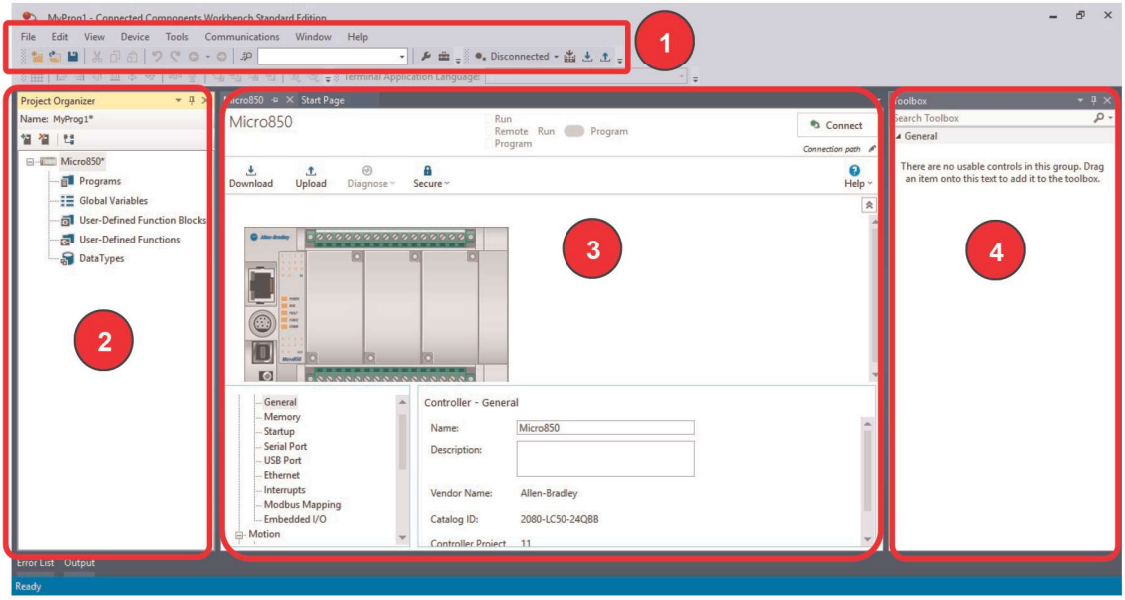
3. จากนั้นจะปรากฏหน้าต่าง Add Device ให้ดับเบิ้ลคลิกที่ Controllers



4. ต่อจากนั้นให้เลือกรุ่น PLC ที่ต้องการใช้งาน โดยให้ดับเบิ้ลคลิกที่รุ่นนั้นๆ ในตัวอย่างนี้คือ Micro850 จากนั้นจะมีรายการรุ่นให้เลือกใช้งาน ในที่นี้เลือก 2080-LC50-24QBB จากนั้นคลิกที่ Select และคลิก Add To Project

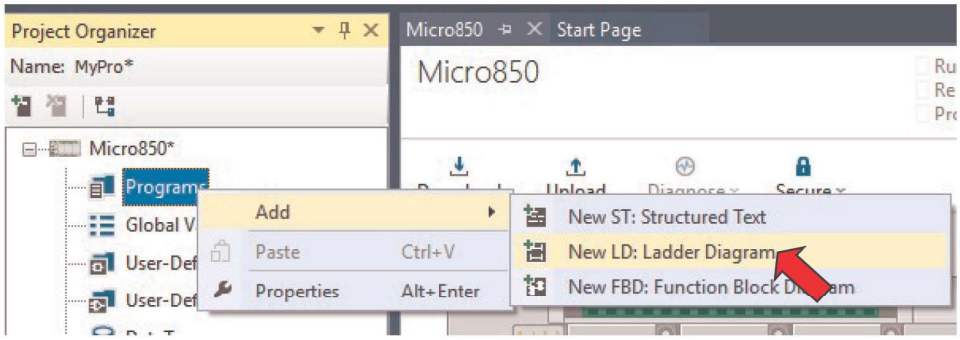


จากนั้นจะปรากฏหน้าต่างที่แสดงรายละเอียด PLC รุ่น Micro850 ดังรูปข้างล่างนี้

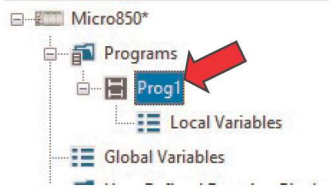


หมายเลข	ชื่อ	รายละเอียด
1	Main Menu	เมนูหลัก
2	Project Organizer	เป็นส่วนที่ใช้สำหรับการเข้าถึงข้อมูลต่างๆ ใน CCW
3	Start Page	ใช้สำหรับ Setting ค่าต่างๆของ Controller รวมถึงการเชื่อมต่อกับ Controller เพื่อการ Upload/Download เป็นต้น
4	Toolbox	เป็นส่วนที่แสดงรายการคำสั่งทั้งหมดที่สามารถนำมาเขียนโปรแกรมได้

5. สร้างโปรแกรมใหม่โดยไปที่ Project Organizer ให้คลิกขวาที่ Program>Add>New LD



จากนั้นที่หน้าต่าง Project Organizer จะมีการเพิ่มส่วนของ Prog1 และ Local Variables ขึ้นมาโดยอยู่ภายใต้ Programs ดังแสดงในรูปข้างล่างนี้

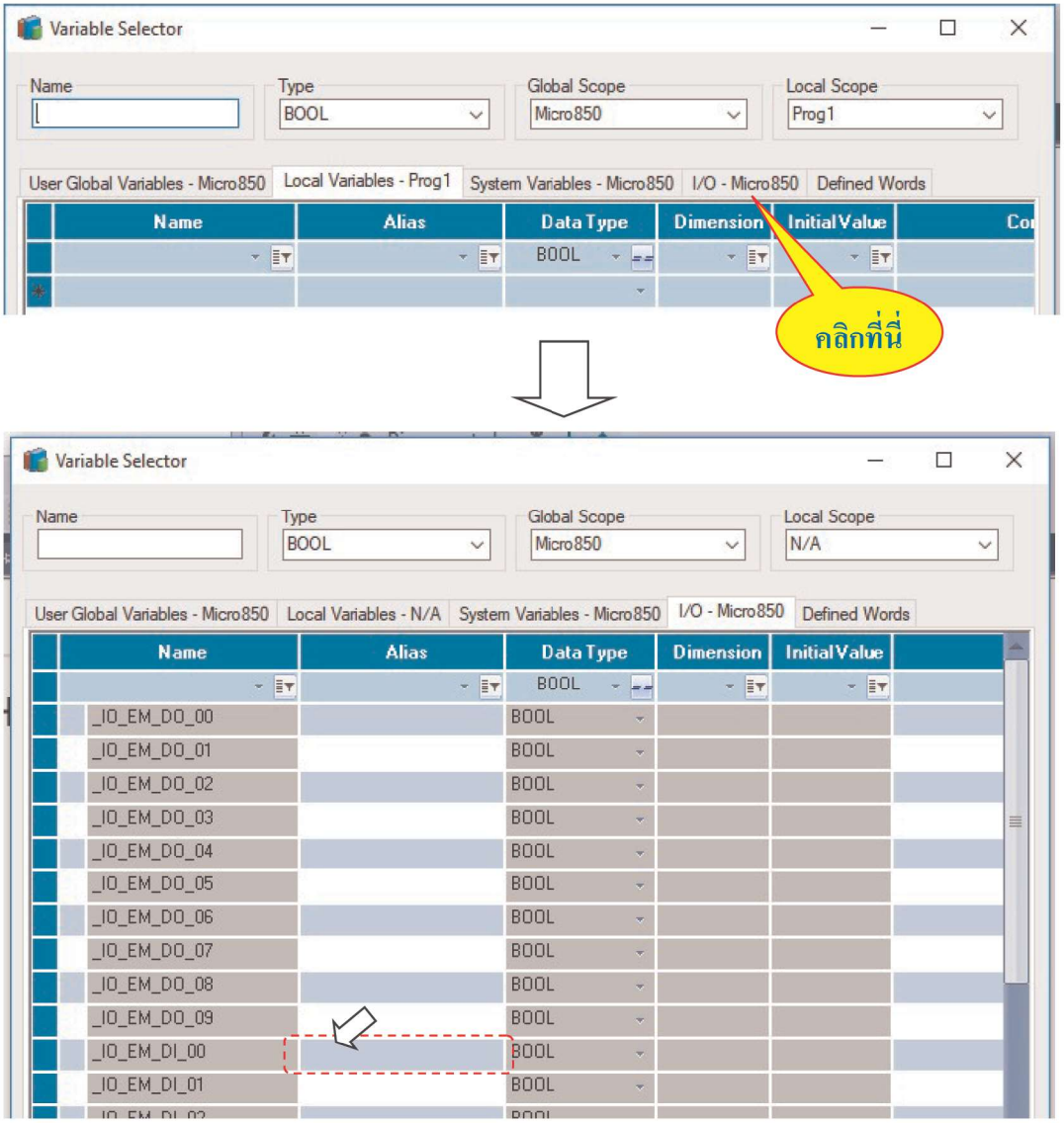


ดับเบิ้ลคลิกที่ Prog1 หรือคลิกขวาและเลือก Open จะปรากฏหน้าต่างสำหรับเขียนแลดเดอร์ไดอะแกรมขึ้นมา

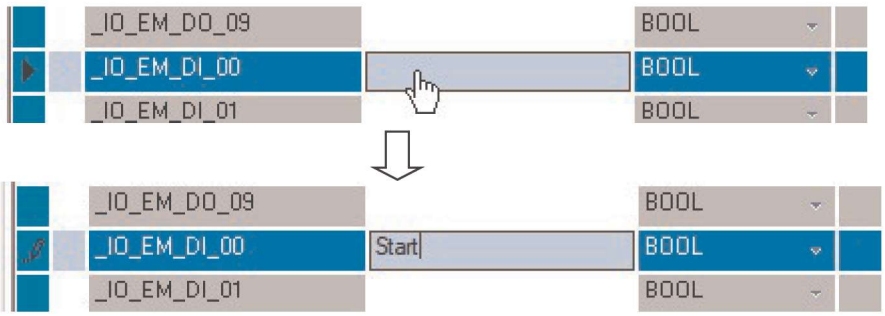
6. การสร้างหน้าคอนแทค NO

ที่ Toolbox ให้ใช้เมาส์คลิกที่ ⊣⊢ (Direct Contact) ค้างไว้แล้วลากเมาส์ไปยังตำแหน่งที่ต้องการวางหน้าคอนแทคดังรูปข้างล่างนี้ จะสังเกตุเห็นเครื่องหมายวงกลมล้อมรอบเครื่องหมายบวกเพื่อแสดงว่าหน้าคอนแทคจะถูกวางต่อจากเครื่องหมายนี้

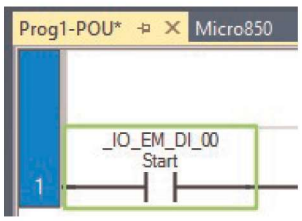
เมื่อปล่อยเมาส์จะปรากฏหน้าต่าง Variable Selector เพื่อใช้กำหนดตัว Variable ของหน้าคอนแทค NO ดังกล่าว ในที่นี้เราจะกำหนดให้เป็นอินพุต I-00 (สวิตช์ Start) ดังนั้นให้คลิกที่แท็บ I/O – Micro850 จะปรากฏหน้าต่างตัวแปรของอินพุตเอาต์พุตที่เป็นฮาร์ดแวร์จริงของ Micro850



จากนั้นให้ใช้เมาส์คลิกที่ _IO_EM_DI_00 ซึ่งเป็นอินพุตบิตที่ 00 (I-00) แล้วป้อนคำว่า Start ลงไป มันเป็นวิธีการตั้งชื่อให้กับอินพุต I-00 และที่ช่อง Data Type จะมีค่าเป็น BOOL ซึ่งหมายถึงตัวแปรที่มีค่าเป็น 0 กับ 1 เท่านั้น

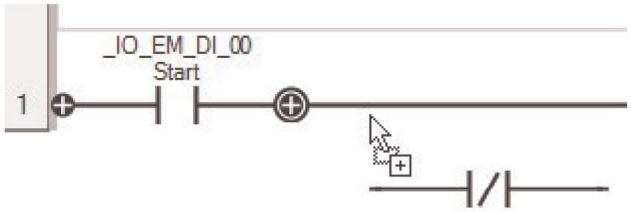


จากนั้นให้คลิกที่ OK จะปรากฏหน้าคอนแทค NO ชื่อ Start ในโปรแกรม

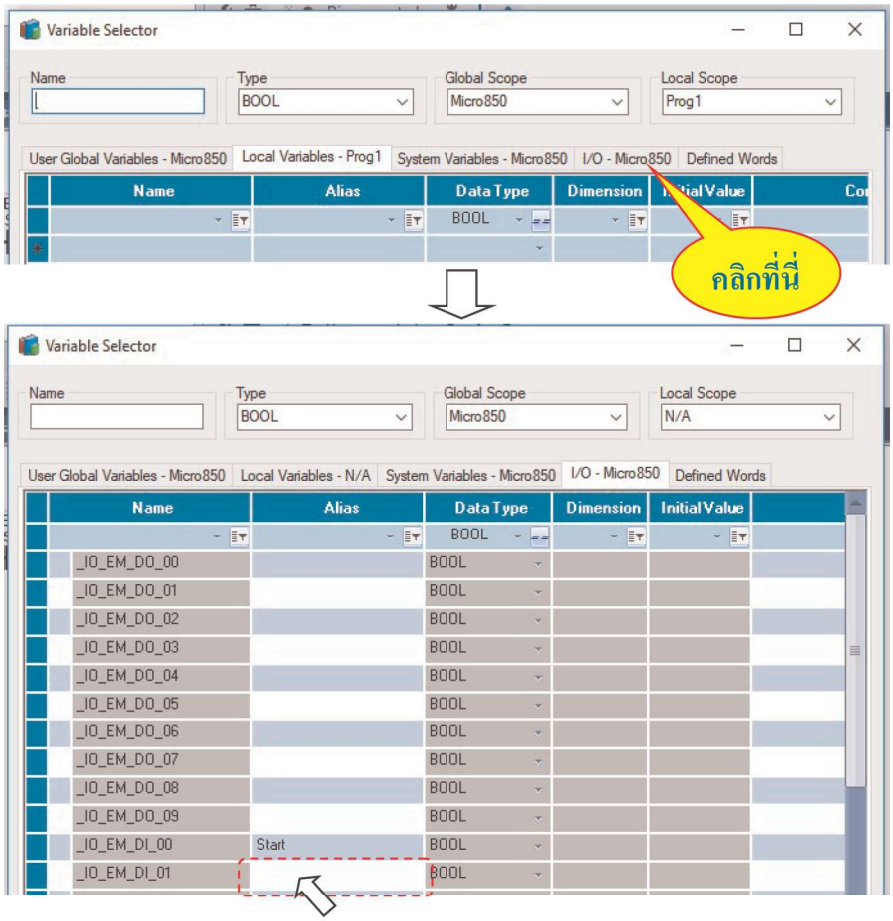


7. การสร้างหน้าคอนแทค NC

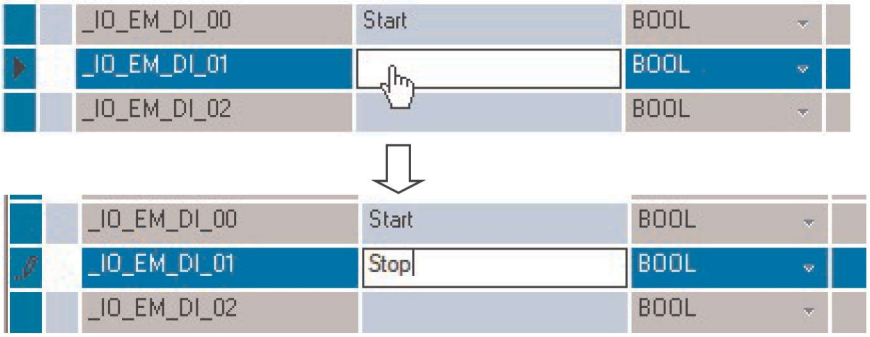
ที่ Toolbox ให้ใช้เมาส์คลิกที่ Reverse Contact ค้างไว้แล้วลากเมาส์ไปยังตำแหน่งหลังหน้าคอนแทค NO ดังรูป



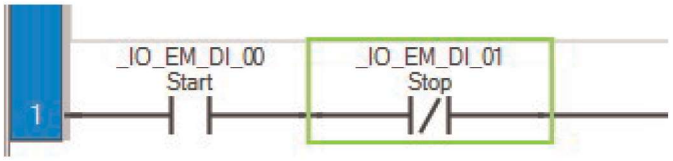
เมื่อปล่อยเมาส์จะปรากฏหน้าต่าง Variable Selector เพื่อใช้กำหนดตัว Variable ของหน้าคอนแทค NC ดังกล่าว ในที่นี้เราจะกำหนดให้เป็นอินพุต I-01 (สวิตช์ Stop) ดังนั้นให้คลิกที่แท็บ I/O – Micro850 จะปรากฏหน้าต่างตัวแปรของอินพุตเอาต์พุตที่เป็นฮาร์ดแวร์จริงของ Micro850



จากนั้นให้ใช้เมาส์คลิกที่ _IO_EM_DI_01 ซึ่งเป็นอินพุตบิตที่ 01 (I-01) แล้วป้อนคำว่า Stop ลงไป มันเป็นวิธีการตั้งชื่อให้กับอินพุต I-01 และที่ช่อง Data Type จะมีค่าเป็น BOOL ซึ่งหมายถึงตัวแปรที่มีค่าเป็น 0 กับ 1

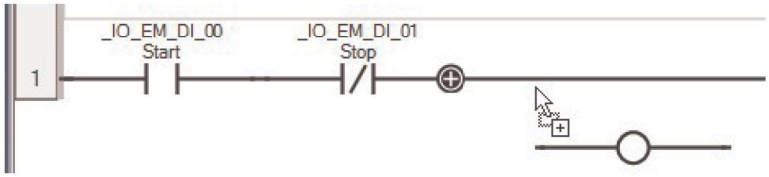


จากนั้นให้คลิกที่ OK จะปรากฏหน้าคอนแทค NC ชื่อ Stop ในโปรแกรม

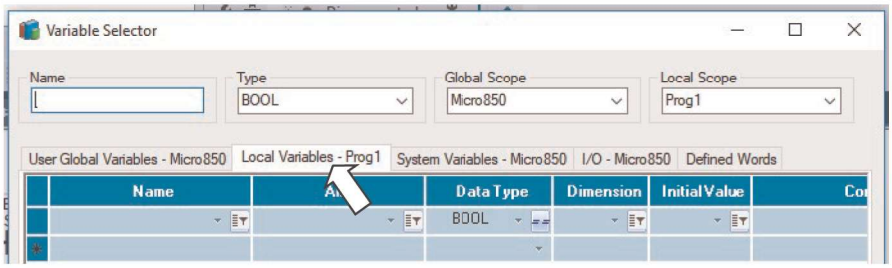


8. การสร้างเอาต์พุต

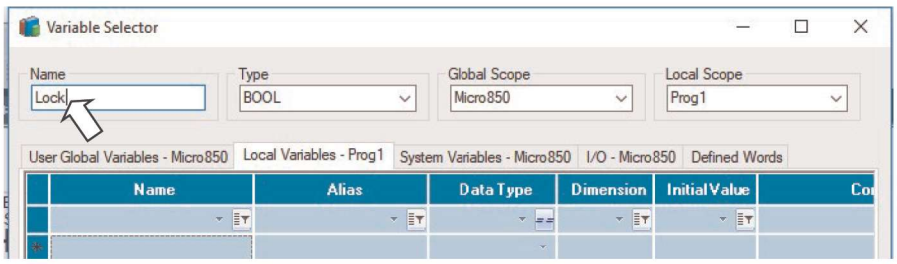
ที่ Toolbox ให้ใช้เมาส์คลิกที่ (Output) ค้างไว้แล้วลากเมาส์ไปยังตำแหน่งหลังหน้าคอนแทค NC ดังรูป



เมื่อปล่อยเมาส์จะปรากฏหน้าต่าง Variable Selector เพื่อใช้กำหนดตัว Variable ของเอาต์พุตดังกล่าว ในที่นี้เราจะกำหนดให้เป็นเอาต์พุตที่เป็น Local Variable จากหน่วยความจำ



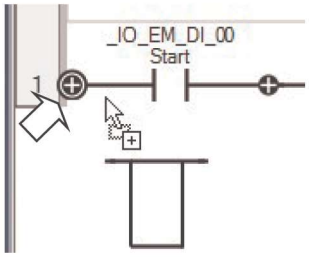
ตั้งชื่อ Variable โดยป้อนคำว่า Lock ที่ Name และ Type ต้องเป็น BOOL



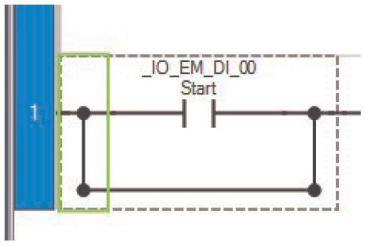
จากนั้นให้คลิกที่ OK จะปรากฏเอาต์พุต ชื่อ Lock ในโปรแกรม

9. การสร้างหน้าคอนแทคขนาน

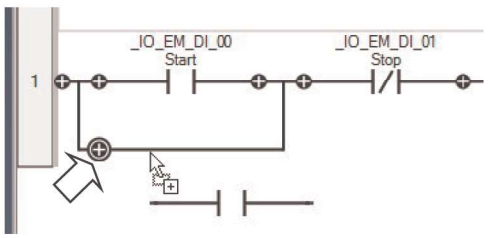
ไปที่ Toolbox ให้ใช้เมาส์คลิกที่ Branch ค้างไว้แล้วลากเมาส์ไปยังตำแหน่งที่ขนานกับหน้าคอนแทค NO โดยให้สังเกตุว่ามีวงกลมล้อมรอบเครื่องหมายบวก เป็นสิ่งแสดงให้รู้ว่าจะสร้างเส้นขนานตรงจุดนี้



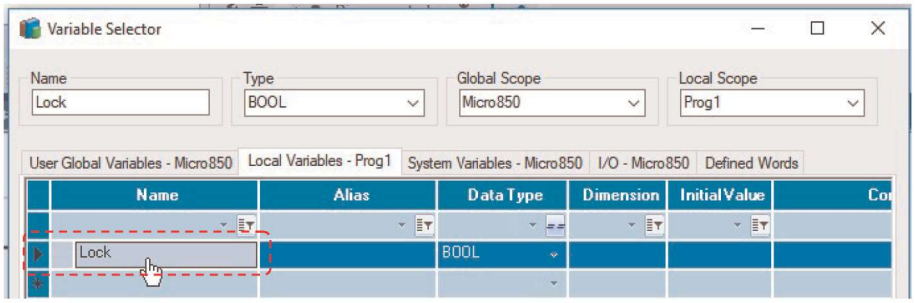
เมื่อปล่อยเมาส์จะมีเส้นขนานกับหน้าคอนแทค Start ดังรูปข้างล่างนี้



กลับไปที่ Toolbox ให้ใช้เมาส์คลิกที่ (Direct Contact) ค้างไว้แล้วลากเมาส์ไปยังตำแหน่งที่เส้นขนานถูกวางไว้ก่อนหน้านี้ โดยให้สังเกตุว่ามีวงกลมล้อมรอบเครื่องหมายบวก



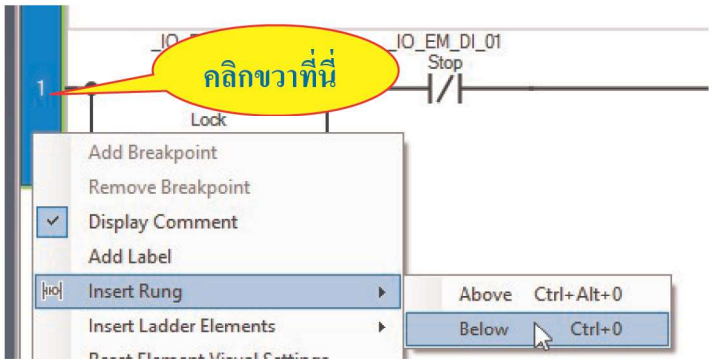
เมื่อปล่อยเมาส์จะปรากฏหน้าต่าง Variable Selector ขึ้นมา ให้คลิกที่ Local Varlable ชื่อ Lock ดังรูป เพื่อเลือกชื่อ Variable สำหรับหน้าคอนแทคที่นำมาขนาน แล้วคลิก OK



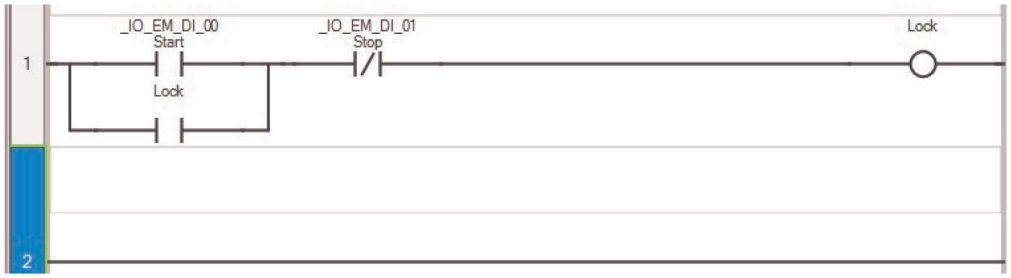
จากนั้นหน้าจอจะแสดงโปรแกรมแลดเดอร์ตามรูปข้างล่างนี้

10. การเขียนคำสั่งไทมเมอร์ (Timer)

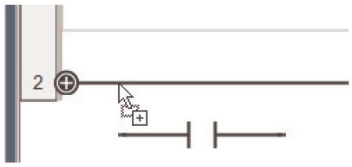
เพิ่ม Rung ใหม่โดยการคลิกขวาที่ด้านซ้ายมือสุดของ Rung 1 ดังแสดงในรูป จากนั้นเลื่อนเมาส์ไปที่ Insert Rung และ Below ตามลำดับ และคลิกที่ Below



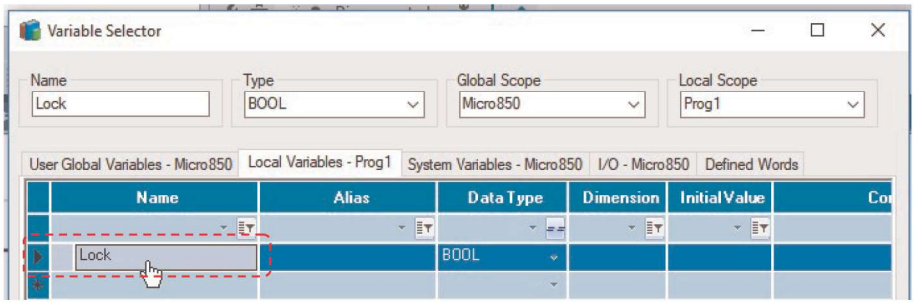
จะปรากฏ Rung ที่ 2 อยู่ใต้ Rung 1 ดังแสดงในรูปข้างล่างนี้



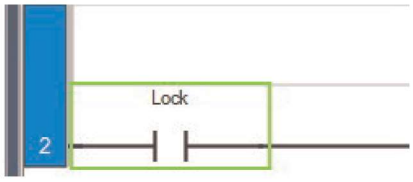
ไปที่ Toolbox ให้ใช้เมาส์คลิกที่ (Direct Contact) ค้างไว้แล้วลากเมาส์ไปยังตำแหน่งแรกของ Rung 2



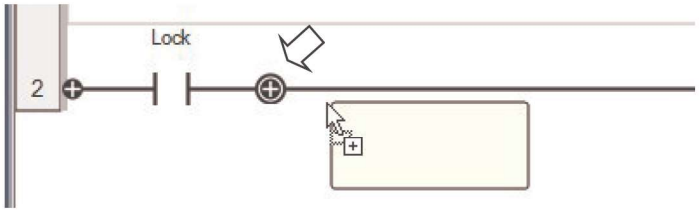
เมื่อปล่อยเมาส์จะปรากฏหน้าต่าง Variable Selector ขึ้นมา ให้คลิกที่ Local Varlable ชื่อ Lock ดังรูป เพื่อเลือกชื่อ Variable สำหรับหน้าคอนแทคนี้ แล้วคลิก OK



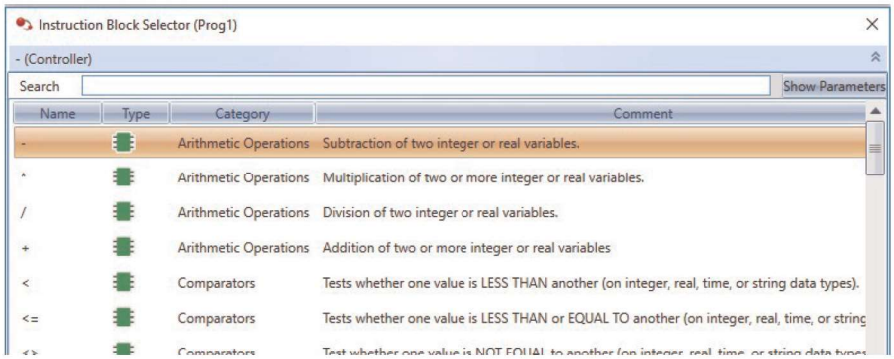
จะปรากฏหน้าคอนแทคชื่อ Lock ใน Rung 2



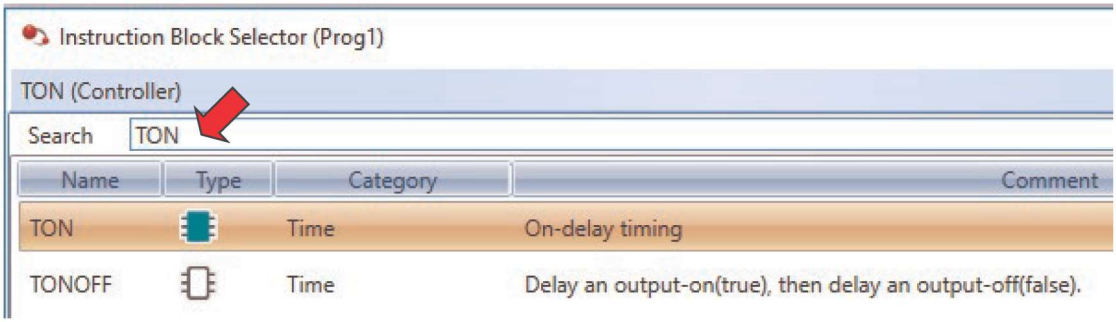
กลับไปที่ Toolbox คลิกที่ (Instruction Box) ค้างไว้แล้วลากเมาส์ไปยังตำแหน่งที่อยู่ต่อจากหน้าคอนแทค Lock จะมีวงกลมล้อมรอบเครื่องหมายบวกดังแสดงในรูปข้างล่างนี้



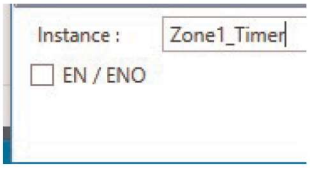
เมื่อปล่อยเมาส์จะปรากฏหน้าต่าง Instruction Block Selector ขึ้นมา



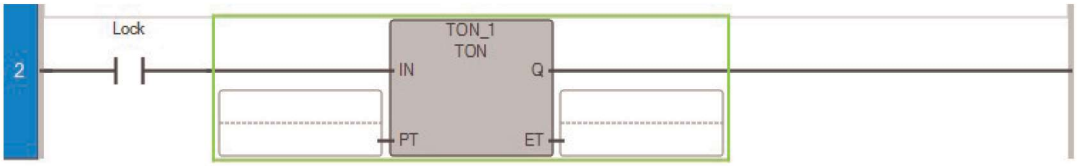
ให้เลือกคำสั่งโดยป้อนชื่อคำสั่งที่ Search ด้วย TON (Timer on delay) จากนั้นคลิก OK



หมายเหตุ: ในกรณีที่ต้องการตั้งชื่อ Instance ของ Timer สามารถป้อนชื่อได้ที่ช่อง Instance ซึ่งอยู่ตรงมุมด้านล่างซ้ายมือของหน้าต่าง Instruction Block Selector ก่อนที่จะคลิก OK ดังตัวอย่างรูปข้างล่างนี้ตั้งชื่อเป็น Zone1_Timer แต่ในตัวอย่างของเราจะใช้ชื่อ Instance ตามที่ซอฟต์แวร์กำหนดให้อัตโนมัติ

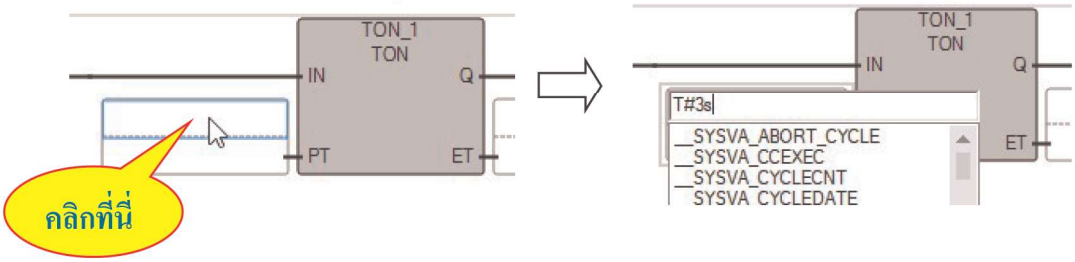


คำสั่ง Timer ที่ได้จะปรากฏใน Rung 2 ดังรูปข้างล่างนี้

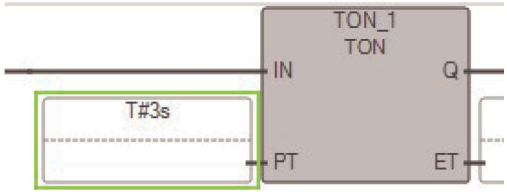


ต่อจากนั้นให้ป้อนค่าตั้งเวลา (Preset Time) ที่ขา PT ในที่นี้จะป้อนเป็นค่าคงที่โดยการคลิกที่ช่องด้านบนของพารามิเตอร์ PT จะปรากฏ Dropdown จากนั้นให้ป้อนค่า T#3s แล้วกด Enter (T# หมายถึงค่าคงที่ Timer ส่วน 3s คือ 3 วินาที ถ้าใส่ 3m จะหมายถึง 3 นาที)

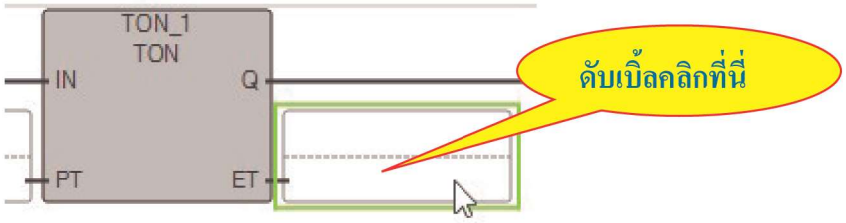
ในกรณีที่ต้องการตั้งค่า PT จาก Variable เราสามารถป้อนชื่อ Variable ตัวนั้นได้เลย แต่ Data Type ของ Variable ต้องเป็น Time และมีการสร้างไว้แล้วล่วงหน้า



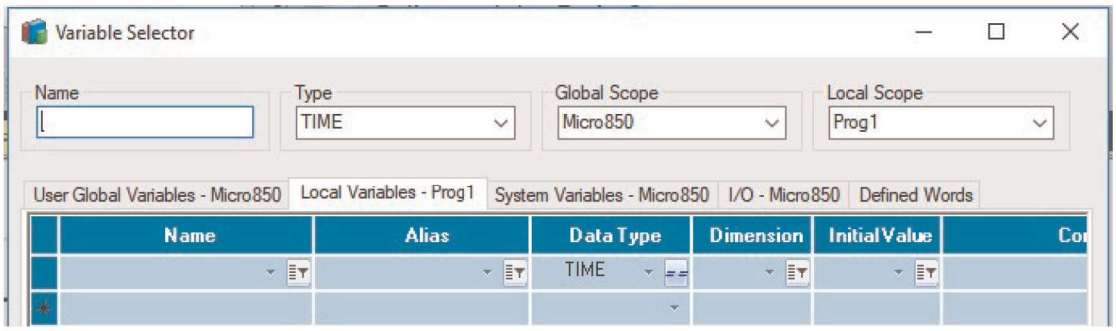
หลังจากกด Enter ค่า T#3s จะปรากฏที่พารามิเตอร์ PT ของ Timer



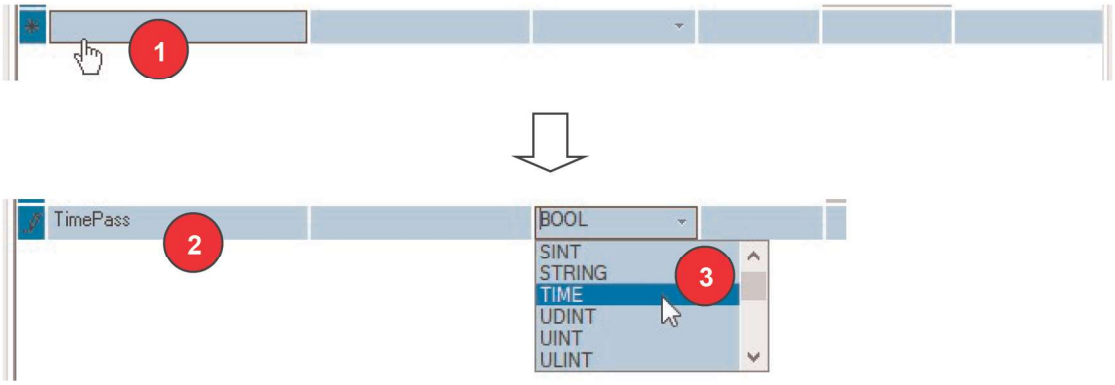
ต่อไปเป็นการกำหนดตัว Variable ที่จะใช้แสดงผลของเวลาที่ผ่านไปขณะที่ Timer ทำงาน แต่เนื่องจากไม่มีการสร้าง Variable ไว้ก่อน เราจึงต้องสร้างตัวแปรใหม่ขึ้นมา ให้ดับเบิ้ลคลิกที่ช่องด้านล่างของพารามิเตอร์ ET ดังรูป



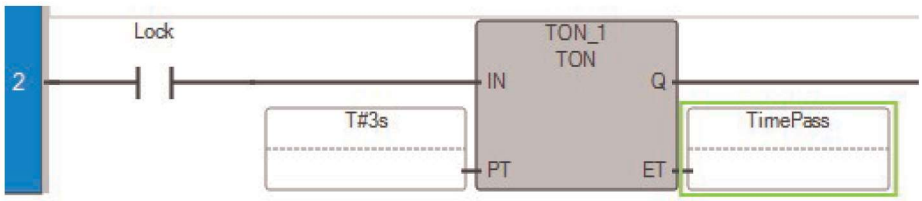
จะปรากฏหน้าต่าง Variable Selector ขึ้นมาเพื่อให้เราเลือกตัวแปร (Variable) แต่ในที่นี้ไม่มีจึงต้องสร้างขึ้นใหม่



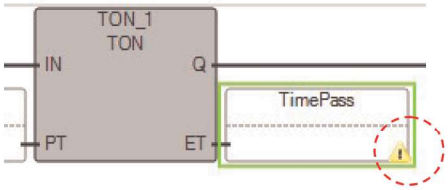
ให้คลิกที่คอลัมน์ Name แถวที่มี * จากนั้นให้ป้อนชื่อ TimePass และที่ Data Type ให้เลือก Time เพราะเป็นตัวแปรเวลา จากนั้นให้คลิก OK



หน้าตาของโปรแกรมที่ได้จะเป็นตามรูปข้างล่างนี้

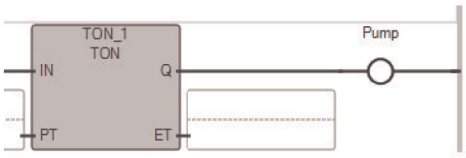


หมายเหตุ ถ้าการป้อนค่าของพารามิเตอร์ PT หรือ ET ไม่ถูกต้องจะมีเครื่องหมาย ! ขึ้นที่ขาพารามิเตอร์นั้นๆ ในตัวอย่างข้างล่างนี้พารามิเตอร์ ET มีการป้อนที่ผิดพลาด เพราะ Data Type ที่เลือกเป็น BOOL แทนที่จะเป็น Time ถ้าต้องการรู้ Data Type ของแต่ละพารามิเตอร์ของคำสั่งให้คลิกที่คำสั่งนั้นๆแล้วกด F1

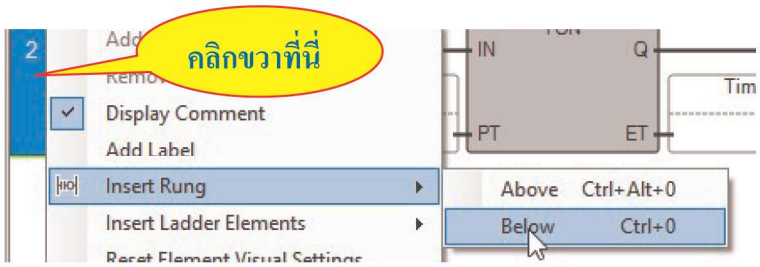


12. การใช้งาน Variable ของไทมเมอร์ (Timer)

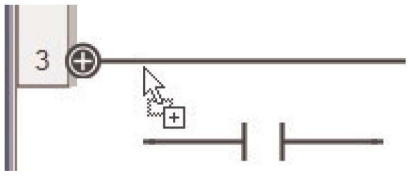
ปกติแล้วเราจะเขียนคำสั่งเอาต์พุตต่อที่ขา Q ของคำสั่ง Timer ดังตัวอย่างในรูปข้างล่างนี้ แต่เราไม่จำเป็นต้องเขียนโปรแกรมแบบนี้ก็ได้ โดยเราสามารถเอา Q ของ Timer ไปใช้งานได้โดยตรง ในขั้นตอนนี้เราจะแสดงถึงวิธีการนำเอา Variable(เช่น Q) ของ Timer ไปใช้งาน



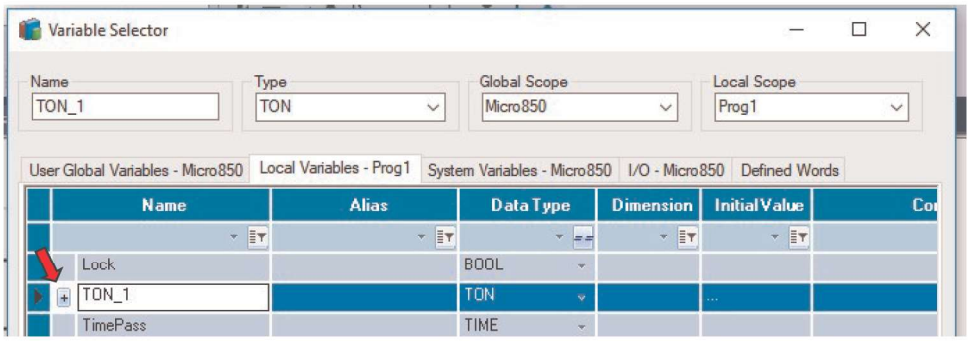
ให้เพิ่ม Rung 3 โดยคลิกขวาที่ด้านซ้ายมือสุดของ Rung 2 ดังแสดงในรูป จากนั้นเลื่อนเมาส์ไปที่ Insert Rung และ Below ตามลำดับ และคลิกที่ Below



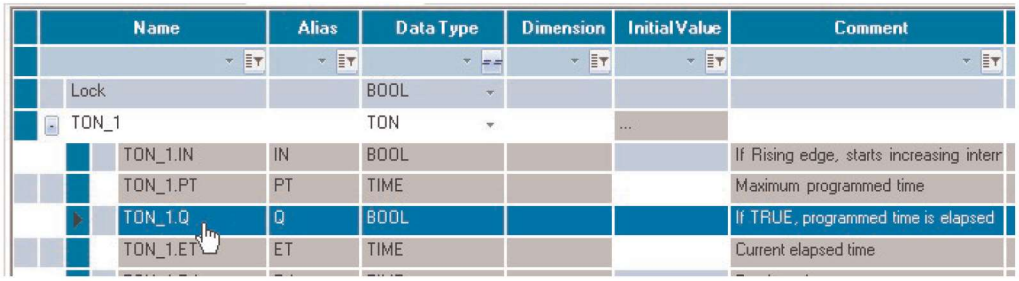
ไปที่ Toolbox ให้ใช้เมาส์คลิกที่ (Direct Contact) ค้างไว้แล้วลากเมาส์ไปยังตำแหน่งแรกของ Rung 3



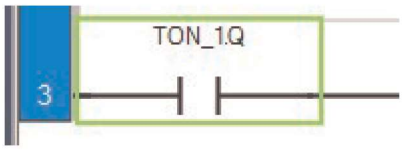
เมื่อปล่อยเมาส์จะปรากฏหน้าต่าง Variable Selector ขึ้นมา ให้คลิกที่เครื่องหมาย + ของ FB ชื่อ TON_1 เพื่อขยายดู Variable ต่างๆ



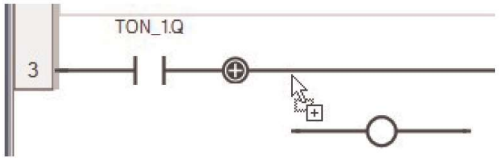
ให้คลิกเลือก TON_1.Q ซึ่งเป็นตัวแปรเอาต์พุต(Output Variable)ของ Timer จากนั้นคลิก OK



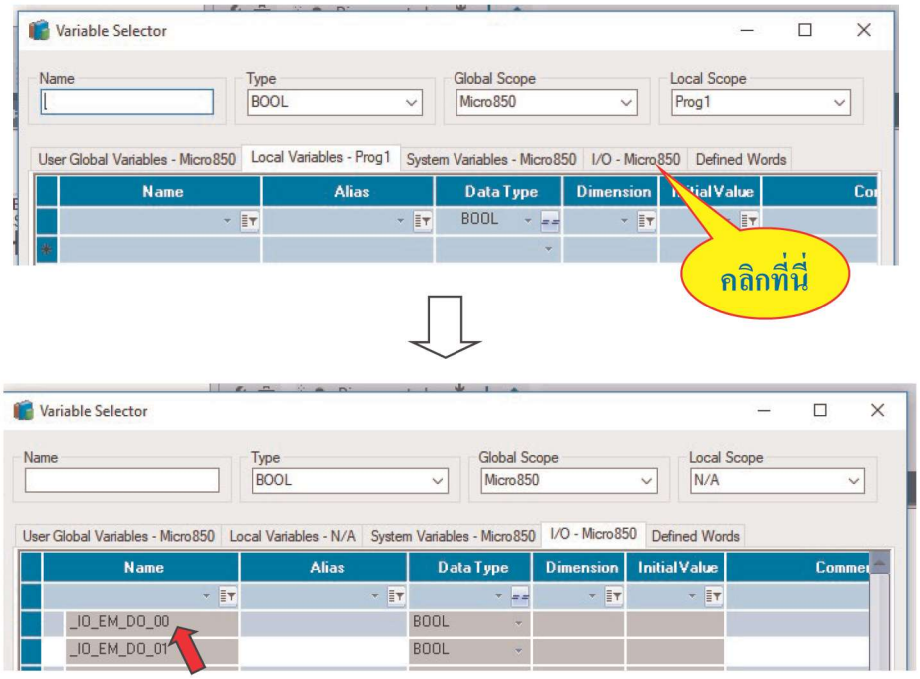
หน้าคอนแทค TON_1.Q จะปรากฏใน Rung 3 ดังรูปข้างล่างนี้



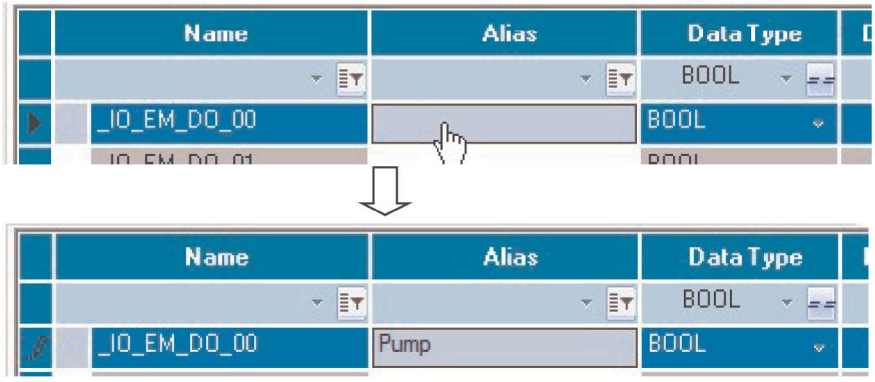
ไปที่ Toolbox ให้ใช้เมาส์คลิกที่ (Output) ค้างไว้แล้วลากเมาส์ไปยังตำแหน่งหลังหน้าคอนแทค TON_1.Q ดังรูป



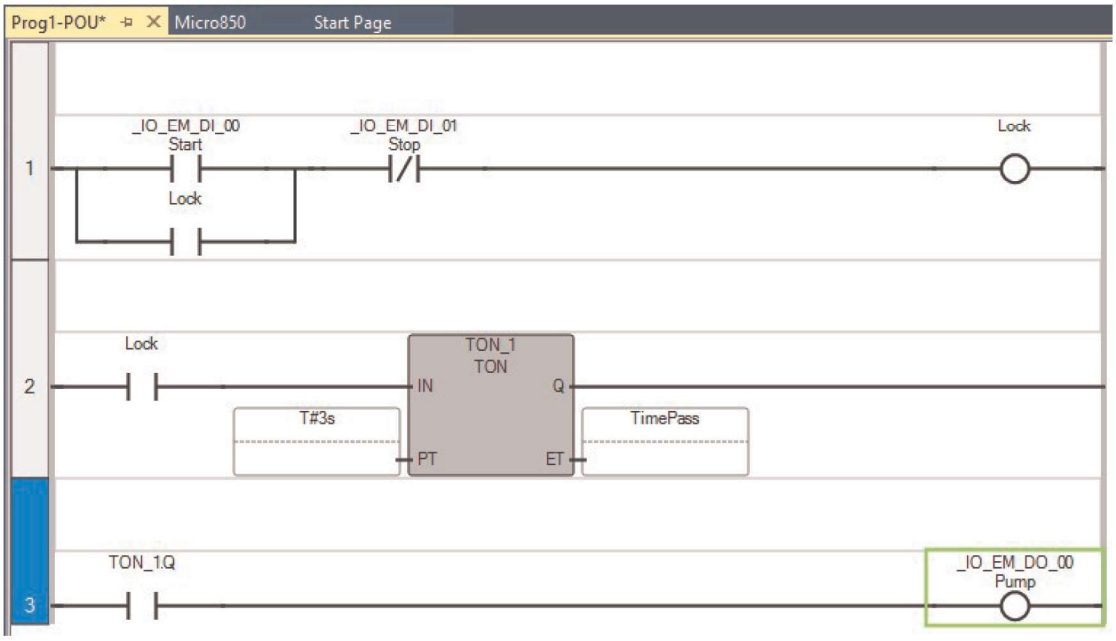
เมื่อปล่อยเมาส์จะปรากฏหน้าต่าง Variable Selector เพื่อใช้กำหนดตัว Variable ของเอาต์พุตดังกล่าว ในที่นี้เราจะกำหนดให้เป็นเอาต์พุต O-00 (Pump) ดังนั้นให้คลิกที่แท็บ I/O – Micro850 จะปรากฏหน้าต่างตัวแปรของอินพุตเอาต์พุตที่เป็นฮาร์ดแวร์จริงของ Micro850



จากนั้นให้ใช้เมาส์คลิกที่ _IO_EM_DO_00 ซึ่งเป็นเอาต์พุตบิตที่ 00 (I-00) แล้วป้อนคำว่า Pump ลงไป เพื่อตั้งชื่อให้กับเอาต์พุต O-00

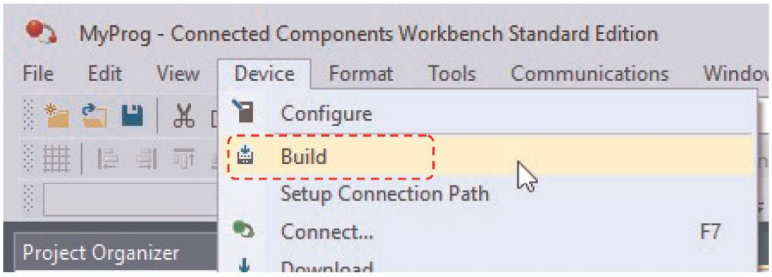


จากนั้นให้คลิกที่ OK จะปรากฏเอาต์พุตชื่อ Pump ใน Rung 3 และโปรแกรมที่สร้างขึ้นตามตัวอย่างจะสมบูรณ์ตามรูปในหน้าถัดไป

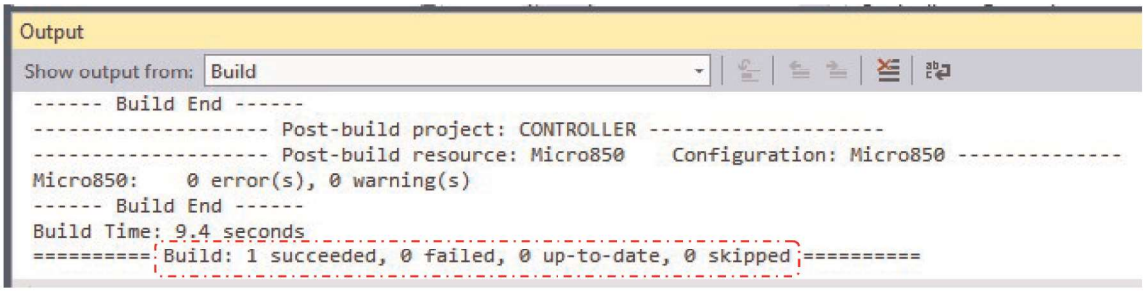


10.3 ขั้นตอนการ Build โปรแกรม

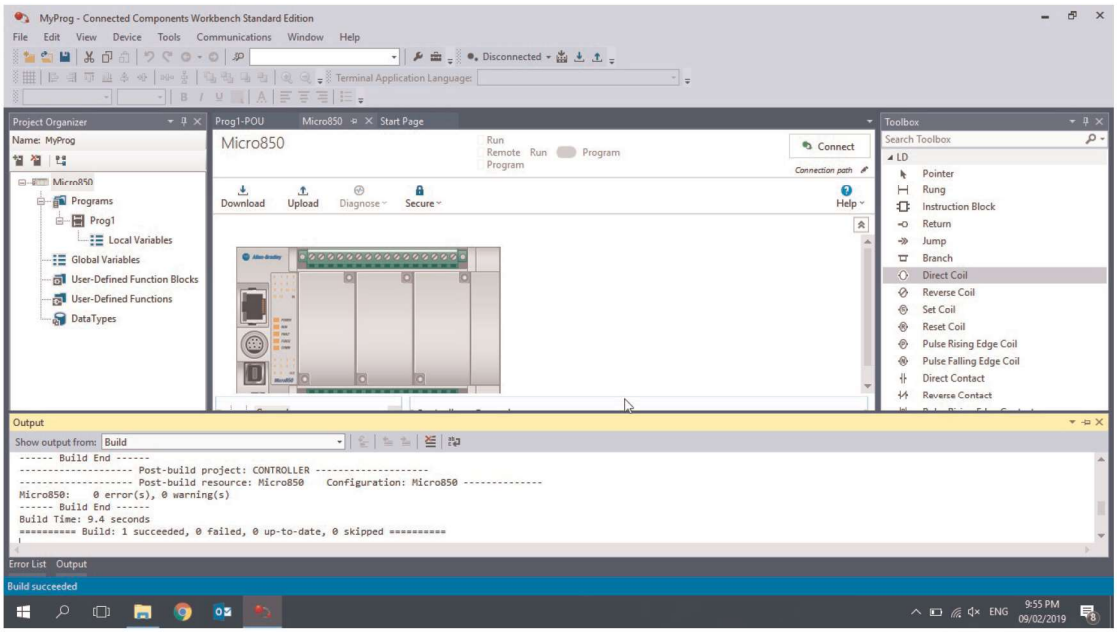
ทำการ Build โดยคลิกที่ Device ของเมนูและเลือก Build



เมื่อทำการ Build เสร็จเรียบร้อยแล้ว หน้าต่าง Output ที่อยู่ด้านล่างของหน้าจอจะแสดงผลลัพธ์ที่ได้จากการ Build จากรูปข้างล่างนี้แสดงว่าการ Build ประสบความสำเร็จ ไม่พบข้อผิดพลาดใดๆ

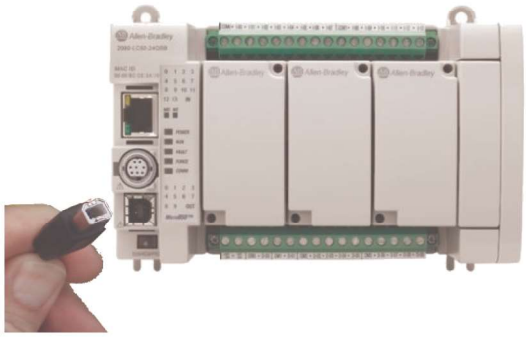


หลังจากการ Build หน้าจอแสดงผลจะรีเฟรสและปรากฏดังรูปข้างล่างนี้ เพื่อให้เราสามารถเชื่อมต่อ(Connect) และ Download โปรแกรมสู่ Micro850 ได้

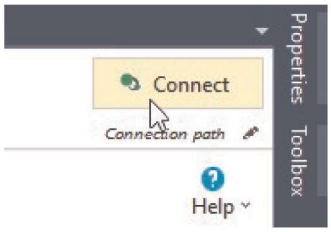


10.4 ขั้นตอนการ Download โปรแกรม

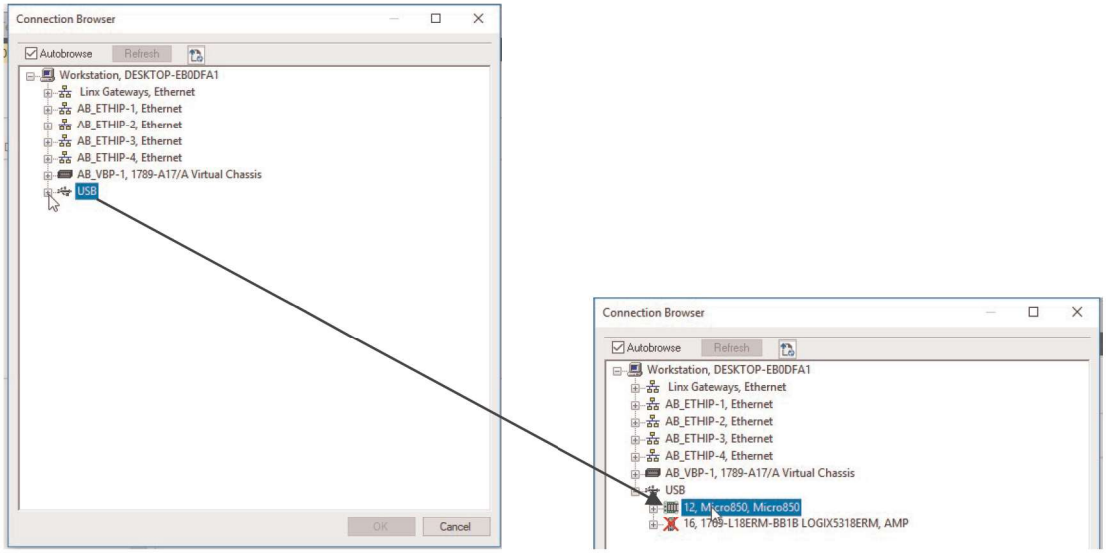
หลังจากการทำการ Build โปรแกรมแล้วไม่พบข้อผิดพลาดใดๆ ต่อมาก็ถึงขั้นตอนการ Download โปรแกรมลงตัว Micro850 ก่อนอื่นเราต้องต่อสาย USB ระหว่างคอมพิวเตอร์กับ Micro850 เสียก่อน



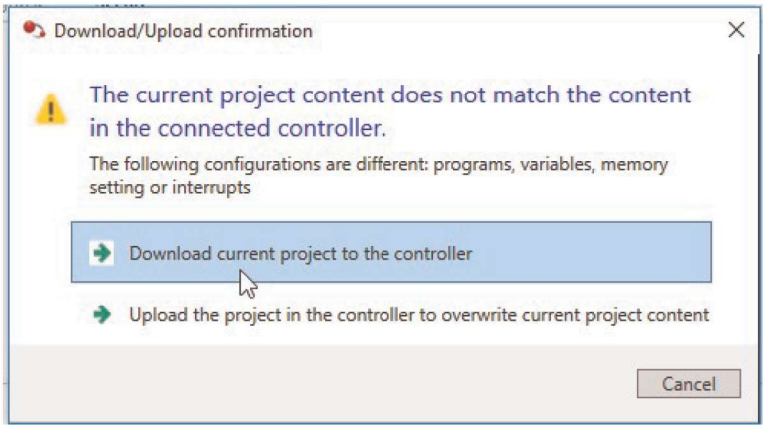
ต่อไปให้คลิก Connect ซึ่งอยู่ที่มุมขวาบนของหน้าจอ และรอสักครู่



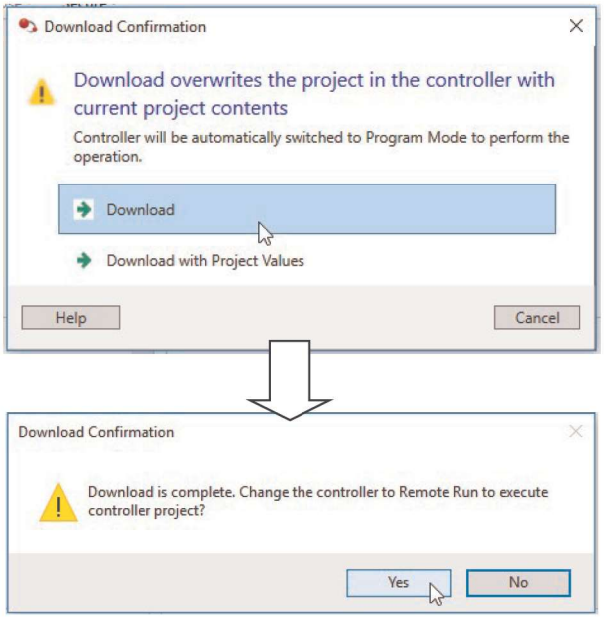
จากนั้นจะปรากฏหน้าต่าง Connection Browser ซึ่งใช้สำหรับเลือกประเภทการเชื่อมต่อ ตามตัวอย่างนี้คลิกที่ USB แล้วจะมีตัวเลือก Micro850 ปรากฏขึ้น ให้คลิกเลือก Micro850 จากนั้นให้คลิกที่ OK



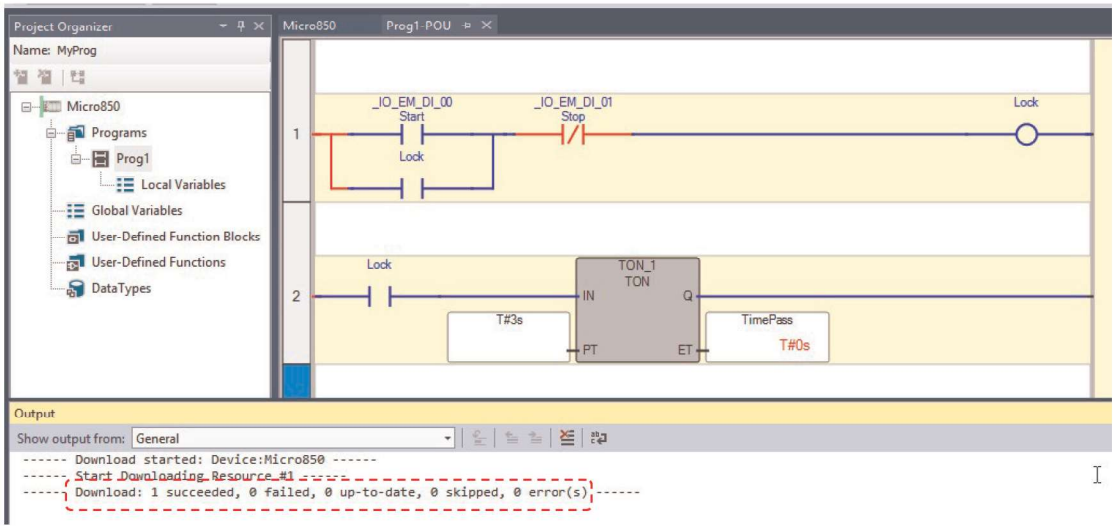
จากนั้นจะปรากฏหน้าต่าง Download/Upload confirmation ซึ่งเราต้องเลือกที่ Download current project to the conttroler ซึ่งหมายถึงการ Download โปรแกรมที่กำลังเขียนอยู่ลงไปที่ Mocro850



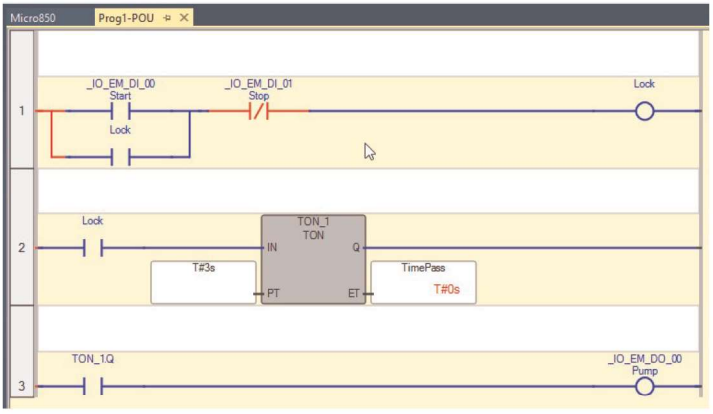
จากนั้นจะปรากฏหน้าต่าง Download Confirmation เพื่อยืนยันการ Download ให้เราคลิกที่ Download ได้เลย จากนั้นให้คลิก Yes



เมื่อ Download สำเร็จจะปรากฏหน้าจอดังรูปข้างล่างนี้ ในขณะที่หน้าต่าง Output แสดงข้อความว่าการ Download ประสบความสำเร็จ จากรูปหน้าคอนแทค NC มีสีแดงแสดงว่ามันปิดวงจรอยู่ คือมีสภาวะที่พร้อมให้กระแสไฟไหลผ่าน แต่ในความเป็นจริงไม่มีกระแสไฟนะครับ

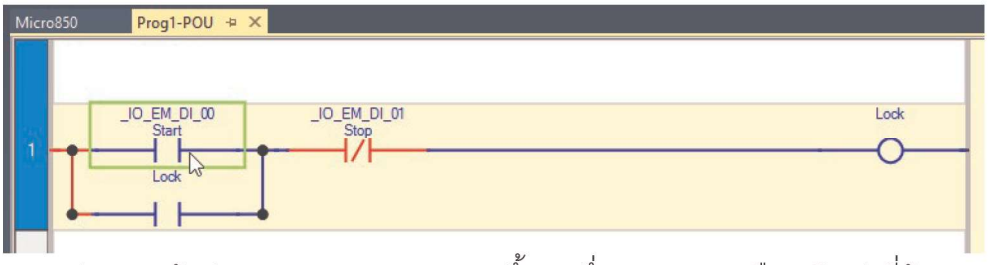


ถ้าเราคลิกไปที่วงจรแลดเดอร์จะทำให้หน้าจอเปิดกว้างขึ้น คราวนี้เราก็สามารถทดลองการทำงานดูโดยเปิด/ปิด สวิทช์ Start และ Stop ได้เลยครับ

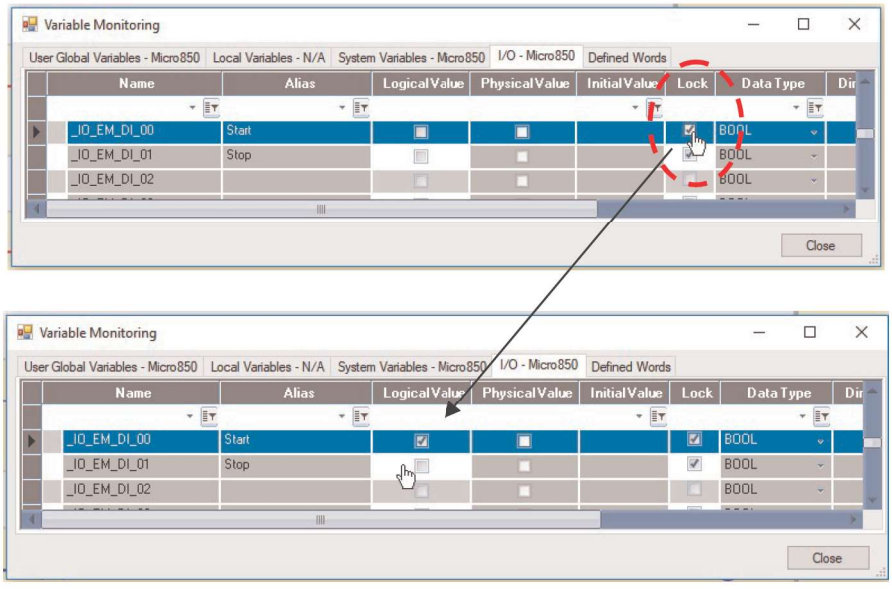


10.5 การทดสอบการทำงานด้วยซอฟต์แวร์

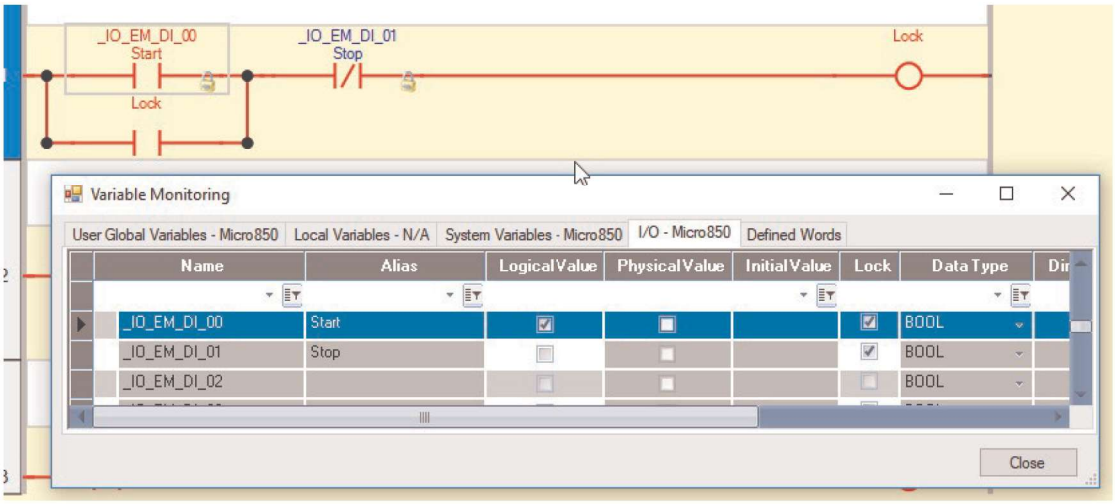
เราสามารถใช้ CCW เพื่อสั่งงานไปที่บิต Start (หรือบิตอื่น)ได้โดยตรง ด้วยการคลิกขวาที่หน้าคอนแทค Start ดังรูป



จะปรากฏหน้าต่าง Variable Monitoring ขึ้นมา ซึ่งเราสามารถเลือกตัวแปรที่ต้องควบคุมสถานะของลอจิกได้ ในตัวอย่างข้างล่างนี้คือการควบคุมลอจิกของตัวแปร Start และ Stop แต่เนื่องจากตัวแปรทั้ง 2 เป็นฮาร์ดแวร์จริงจึงมีสถานะตามสัญญาณอินพุตเอาต์พุตที่ต่ออยู่ในขณะนั้น ดังนั้นการที่จะเปลี่ยนสถานะมันได้ต้องคลิกที่ Lock ก่อนการคลิกที่ LogicalValue



จากรูปข้างล่างนี้แสดงวงจรแลดเดอร์ขณะสั่งให้ลอจิกของ Start เป็น True ซึ่งจะทำให้ลอจิก Lock เป็น True ด้วย เราจึงเห็นวงจรกลายเป็นสีแดงทั้งหมด

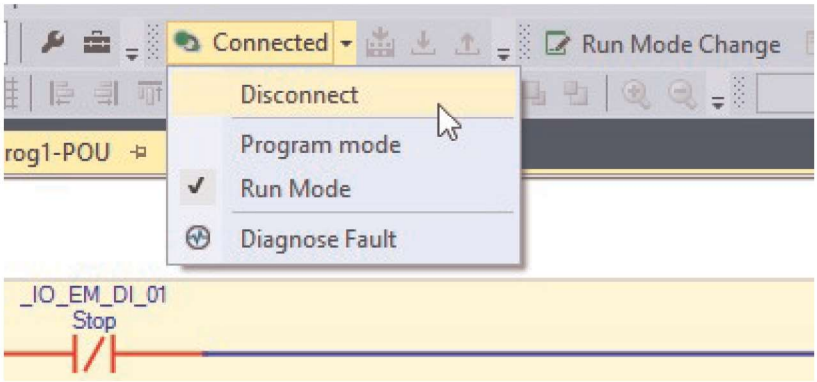


10.6 การ Disconnect/Connect

การเชื่อมต่อและยกเลิกการเชื่อมต่อระหว่างคอมพิวเตอร์กับคอนโทรลเลอร์ตระกูล Micro800 สามารถทำได้ 2 แบบ ดังนี้

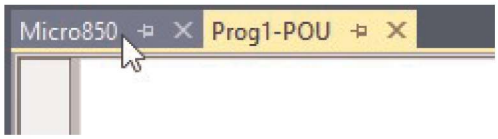
<u>การใช้ Tool Bar</u>

จากรูปข้างล่างจะแสดงการเชื่อมต่อและยกเลิกการเชื่อมต่อ (Connect/Disconnect) โดยการคลิกที่ Tool Bar ซึ่งจะมีดรอปดาวเมนู ให้เลือกว่าจะ Connect หรือ Disconnect นอกจากนั้นเรายังสามารถเปลี่ยนโหมดการทำงานของคอนโทรลเลอร์ Micro800 ได้อีกด้วย โดยเลือกเป็น Run Mode หรือ Program Mode ได้



การใช้แท็บ Controller

จากรูปข้างล่างให้คลิกที่แท็บ Micro850



จะแสดงผลดังรูปข้างล่างนี้ จากนั้นให้คลิก Disconnect หรือ Connect ขึ้นอยู่กับว่าคุณต้องการเชื่อมต่อหรือยกเลิกการเชื่อมต่อ

สรุปท้ายบท

ในบทนี้เราได้เรียนรู้ขั้นตอนต่างๆ ในการใช้ซอฟต์แวร์ CCW ในการเขียนโปรแกรมแลดเดอร์และสามารถดาวน์โหลดใส่คอนโทรลเลอร์ Micro850 นอกจากนั้นเรายังได้ทดสอบการสั่งงานบิตต่างๆให้ทำงานตามที่ต้องการได้ เพียงแค่นี้คุณก็สามารถสร้างโปรแกรม PLC อย่างง่ายได้แล้วครับ

Chapter

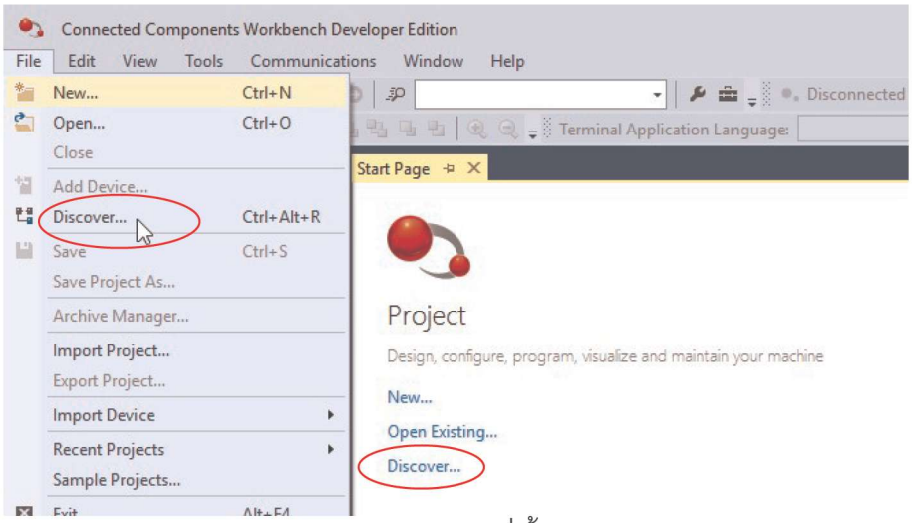
11

การใช้งานซอฟต์แวร์ CCW (2)

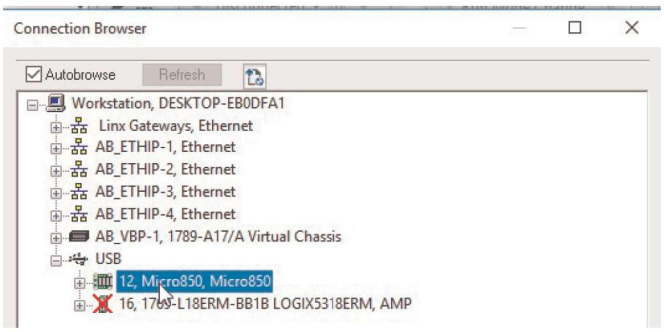
ในบทที่ผ่านมาเราได้แสดงวิธีการใช้งานซอฟต์แวร์ CCW เพื่อเขียนโปรแกรมแลดเดอร์และทดสอบการใช้งานเบื้องต้นไปแล้ว ต่อไปเราจะแสดงวิธีการใช้งานซอฟต์แวร์ CCW เพื่อวัตถุประสงค์อื่นๆ เพิ่มเติม

11.1 การเชื่อมต่อโดยไม่ต้องรู้รุ่นคอนโทรลเลอร์

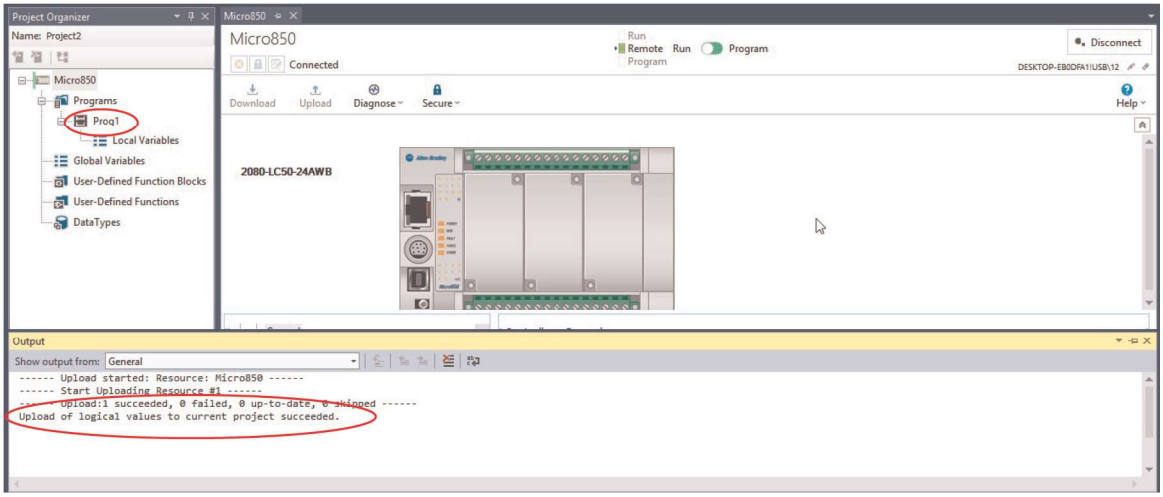
กรณีที่คุณต้องการเชื่อมต่อกับคอนโทรลเลอร์ตระกูล Micro800 มีขั้นตอนง่ายๆ ดังนี้ ให้ไปที่ File>Discover… หรือไปที่หน้า Start Page แล้วคลิกที่ Discover…



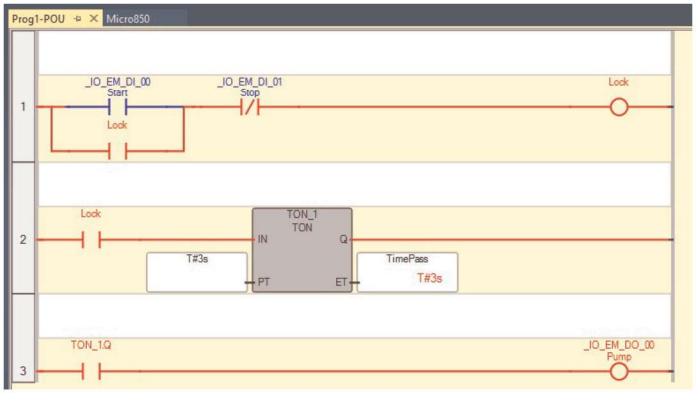
จากนั้นจะปรากฏหน้าต่าง Connection Browser ในที่นี้ให้เลือก USB และ Micro850



จากนั้นซอฟต์แวร์จะทำการ Upload โปรแกรมแลดเดอร์และค่า Setting ต่างๆ จากคอนโทรลเลอร์ขึ้นมาไว้ที่คอมพิวเตอร์จะปรากฏข้อความในหน้าต่าง Output ว่า 'Upload of logical valves to current project suceeded'



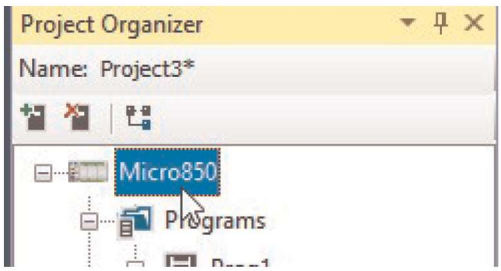
ให้คลิกที่แท็บ Prog1 มันจะแสดงโปรแกรมแลดเดอร์ที่เพิ่ง Upload ขึ้นมา



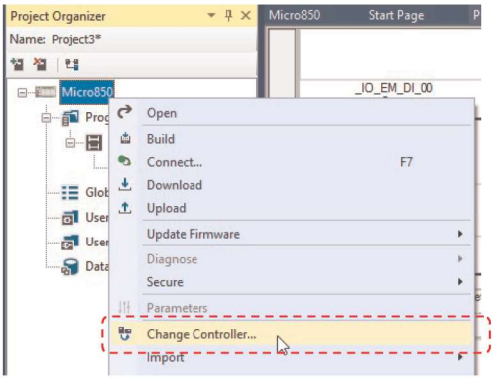
11.2 การเปลี่ยนรุ่นคอนโทรลเลอร์

บางครั้งเราอาจเลือกรุ่นคอนโทรลเลอร์ของ Micro800 ผิด หรือบางกรณีเรามีโปรแกรมที่เขียนไว้แล้วแต่ต้องการนำโปรแกรมไปใช้กับคอนโทรลเลอร์รุ่นอื่น เราสามารถทำได้ดังนี้

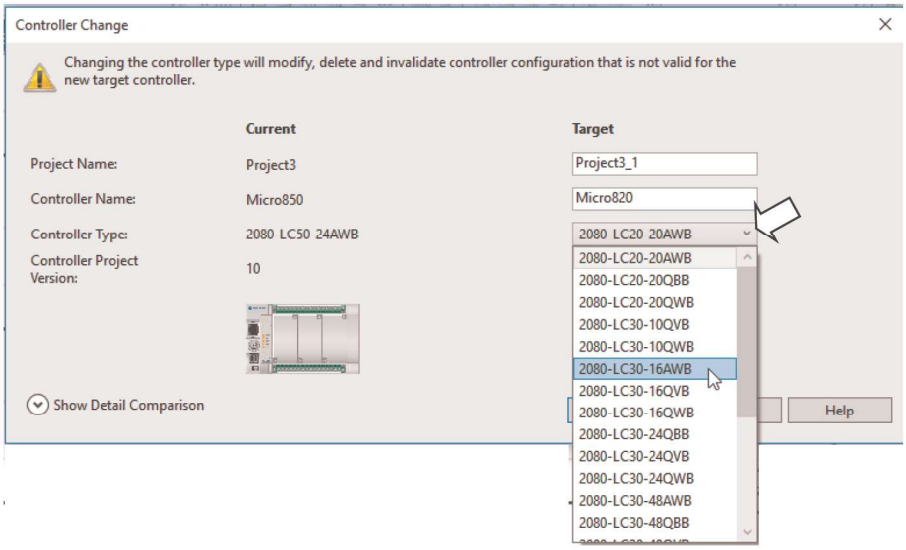
ซอฟต์แวร์ CCW ต้องไม่ Connected กับ Micro800 จากนั้นไปที่ Project Organizer ให้คลิกขวาที่รูปคอนโทรลเลอร์ ตามตัวอย่าง คือ Micro850



จะปรากฏเมนูให้เลือก จากนั้นเลือก Change Controller ตามรูปข้างล่าง

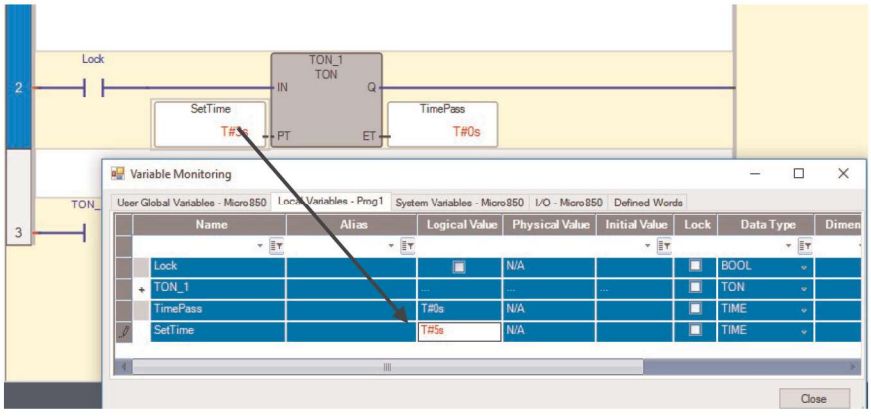


จากนั้นจะปรากฏหน้าต่าง Controller Change ขึ้นมา ที่หน้าต่างด้านขวามือให้คุณเลือกรุ่นคอนทโทรลเลอร์ที่ต้องการใช้งาน หรืออาจเปลี่ยน Project Name และ Controller Name ได้ตามที่คุณต้องการ จากนั้นให้คลิก OK



11.3 การแสดงและเปลี่ยนค่าตัวแปร

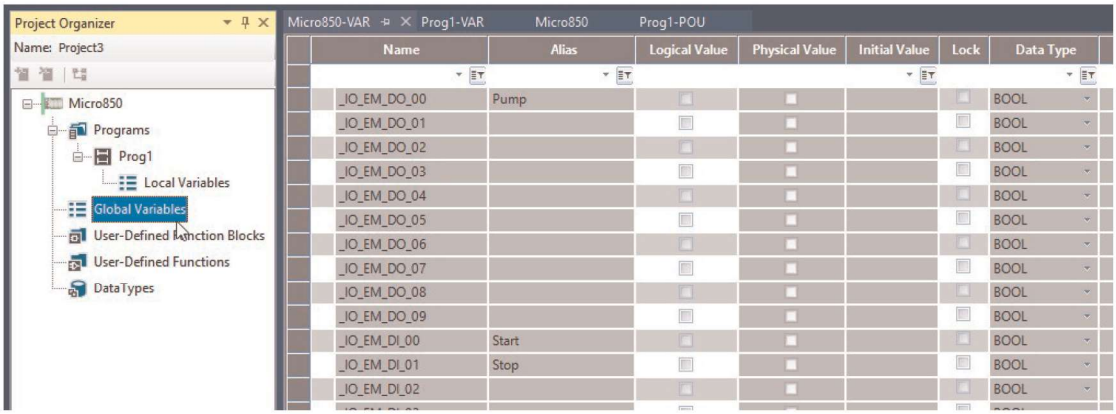
ในขณะที่กำลังเชื่อมต่อ(Connected) ระหว่างคอมพิวเตอร์กับคอนโทรลเลอร์อยู่นั้น เราสามารถดูสถานะหรือค่าข้อมูลของตัวแปรต่างๆได้



จากรูปในหน้าที่ 11-3 คำสั่ง TON มีค่า Preset (PT) เป็นตัวแปรชื่อ SetTime ซึ่งเราสามารถเปลี่ยนค่าในตัวแปรนี้โดยใช้ซอฟต์แวร์ CCW การเปลี่ยนค่านี้จะส่งผลต่อค่าตั้งเวลาของคำสั่ง TON เมื่อต้องการเปลี่ยนค่าให้คลิกที่ตัวแปร SetTime ซึ่งมีค่าเท่ากับ T#3s (3 วินาที) จากนั้นจะปรากฏหน้าต่าง Variable Monitoring ขึ้นมา เราจะเห็นชื่อตัวแปร SetTime โดยสามารถเปลี่ยนค่า SetTime ด้วยการป้อน T#5s (5 วินาที) ซึ่งจะทำให้ค่า SetTime ในโปรแกรมแลดเดอร์เปลี่ยนตามด้วย นั่นเท่ากับว่าค่าตั้งเวลาเปลี่ยนเป็น 5 วินาที

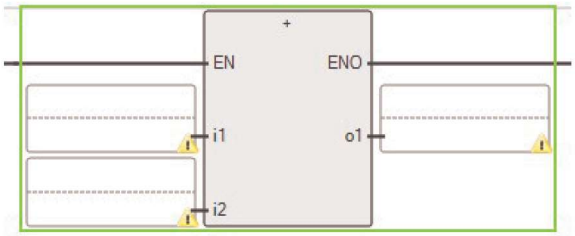
นอกจากนั้นเรายังสามารถแสดงค่าและสถานะของตัวแปร Local ได้ด้วยการคลิกที่ Local Variable ในหน้าต่าง Project Organizer ดังรูปข้างล่างนี้ ถ้าคลิกที่ช่อง Lock ซึ่งจะทำให้ค่า Logical Value ค้างค่าตามสถานะที่เราคลิกในช่อง Logical Value ถ้าไม่ได้คลิกช่อง Lock ค่าของตัวแปรจะเปลี่ยนไปตามการทำงานของโปรแกรมแลดเดอร์เมื่ออยู่ในโหมด RUN

ในทำนองเดียวกันการแสดงค่าและสถานะของตัวแปร Global ทำได้โดยการคลิกที่ Global Variable ในหน้าต่าง Project Organizer ดังรูปข้างล่างนี้

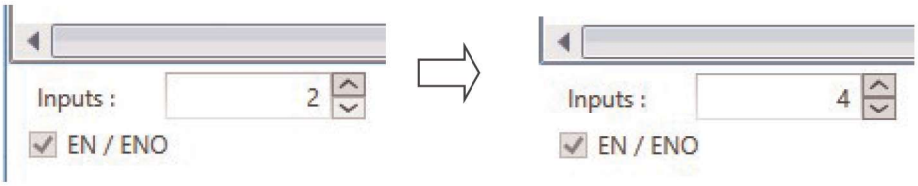


11.4 การเพิ่ม Input Variable ให้กับฟังก์ชั่นคำนวณคณิตศาสตร์

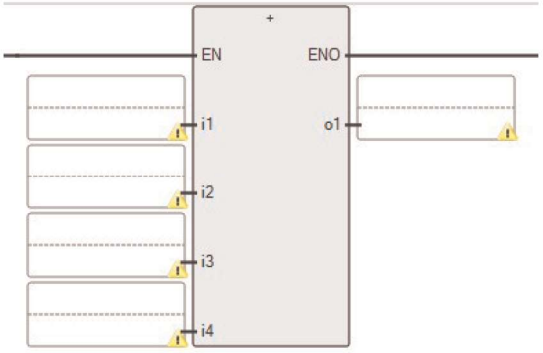
โดยปกติฟังก์ชั่นที่ใช้คำนวณค่าทางคณิตศาสตร์จะมีตัวแปรอินพุตเพียง 2 ตัว ดังแสดงในรูปข้างล่างนี้ สมมุติว่าเราต้องการบวกเลขหลายๆจำนวน จะต้องใช้ฟังก์ชั่นหลายตามจำนวนตัวเลขที่ต้องการบวก แต่ Micro800 สามารถเพิ่มจำนวน Input Variable ได้ทำให้ประหยัดคำสั่งในการเขียนโปรแกรม



เราสามารถเพิ่มจำนวนของ Input Variable ได้โดยการดับเบิ้ลคลิกที่ฟังก์ชั่นนั้นๆ เช่น ฟังก์ชั่นบวก (+) จากนั้นจะปรากฏหน้าต่าง Instruction Block Selector ให้สังเกตุที่มุมด้านล่างซ้ายมือจะเห็น Inputs ซึ่งปัจจุบันมีค่าเป็น 2 ให้เปลี่ยนเป็นเลข 4 หรือเลขอื่นๆตามที่คุณต้องการใช้งาน จากนั้นคลิก OK

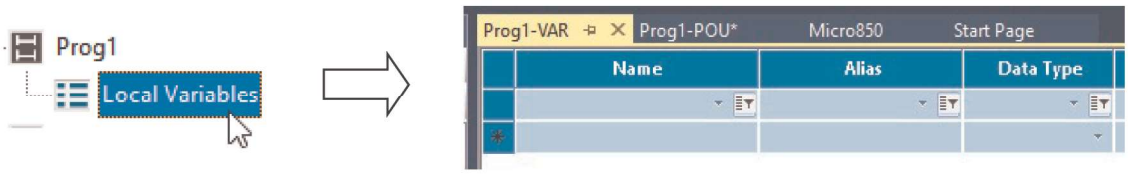


หลังจากคลิก OK แล้ว หน้าตาของคำสั่งบวกจะเปลี่ยนไปโดยมี Input เพิ่มเป็น 4 อินพุต

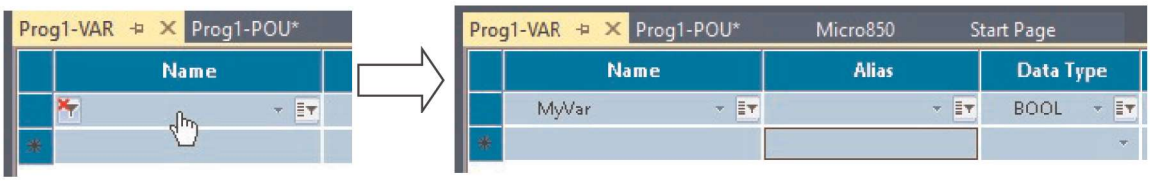


11.5 การสร้างตัวแปร (Variable)

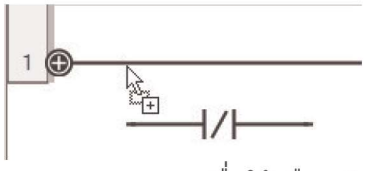
การสร้างตัวแปรทั้ง Local Variable และ Global Variable สามารถทำได้ 2 วิธี คือ การสร้างก่อนที่จะเขียนโปรแกรม หรือสร้างระหว่างที่เขียนโปรแกรม เรามาดูวิธีแรกกันก่อนดีกว่า ให้ดับเบิ้ลคลิกที่ Local Variable ที่อยู่ภายใต้ Prog1 จะทำให้หน้าต่าง Prog1-VAR ปรากฏขึ้นมา



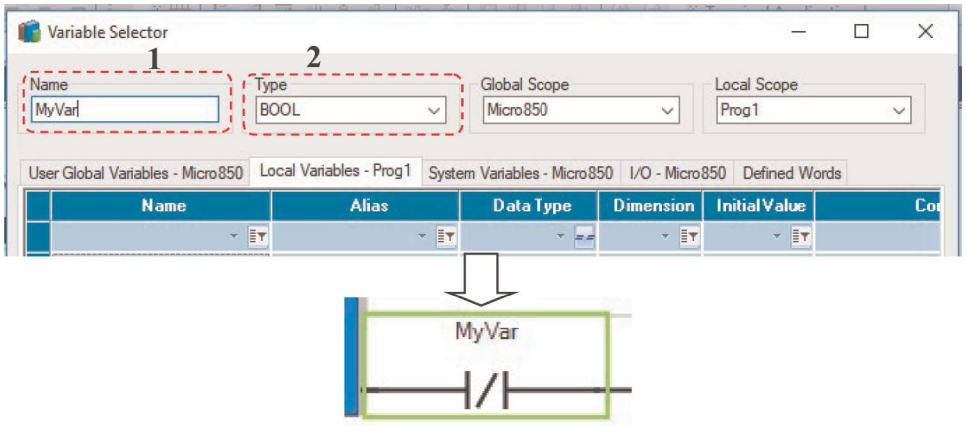
จากนั้นให้คลิก Row ที่ว่าง แล้วคลิกที่ Name เพื่อใส่ชื่อตัวแปร จากนั้นคลิกที่ Data Type เพื่อใส่ประเภทข้อมูลในที่นี้ให้ใส่ BOOL จากนั้นให้ใช้เมาส์คลิกที่ตำแหน่งอื่นเพียงเท่านี้เราก็จะได้ตัวแปรชื่อ MyVar แล้วครับ



ส่วนอีกวิธีหนึ่งคือทำระหว่างเขียนโปรแกรม สมมุติว่าต้องการเขียนหน้าคอนแทค NC ลงในโปรแกรม เมื่อเราลากหน้าคอนแทคมาวางในโปรแกรมแล้วปล่อยเมาส์

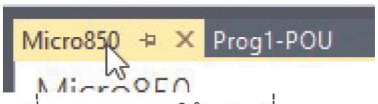


มันจะปรากฏหน้าต่าง Variable Selector เพื่อให้เลือกตัวแปร แต่ถ้าต้องสร้างใหม่ให้พิจารณาก่อนว่ามันเป็นตัวแปร Global หรือ Local โดยปกติมันจะถูกเลือกเป็นตัวแปร Local เริ่มสร้างตัวแปรโดยไปที่ Name เพื่อใส่ชื่อตัวแปรที่ต้องการสร้าง จากนั้นไปที่ Type เพื่อเลือกประเภทตัวแปร จากนั้นให้คลิก OK เราก็จะได้ตัวแปรชื่อ MyVar ในโปรแกรมที่สร้างขึ้น

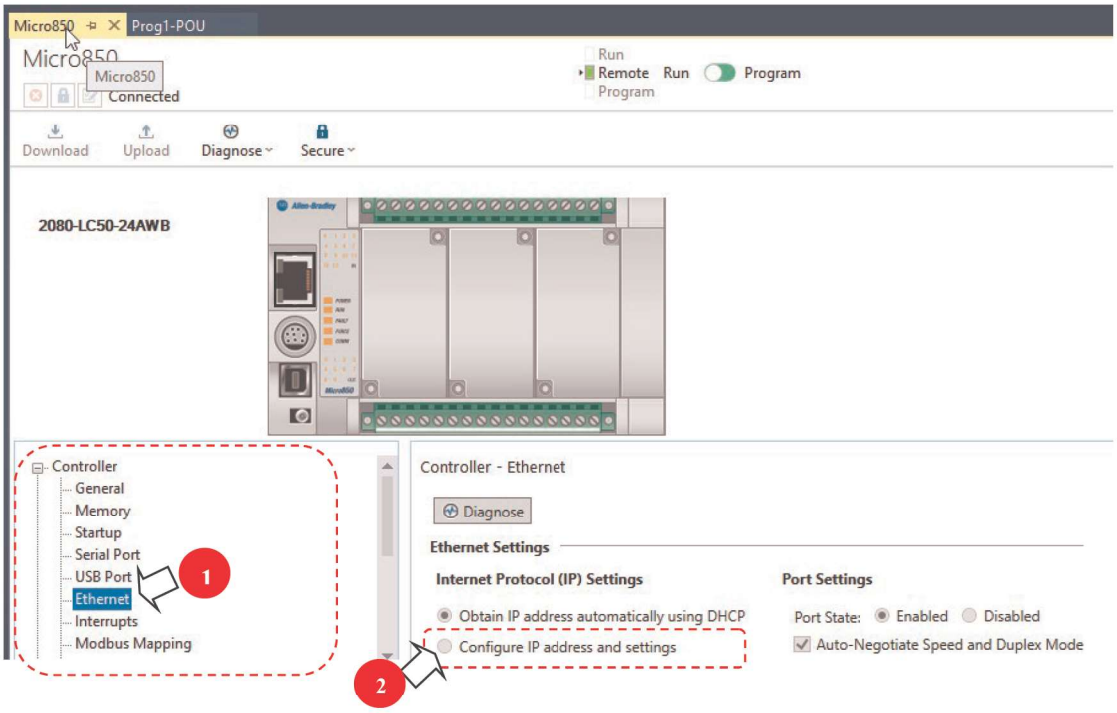


11.6 การเปลี่ยนค่า IP address

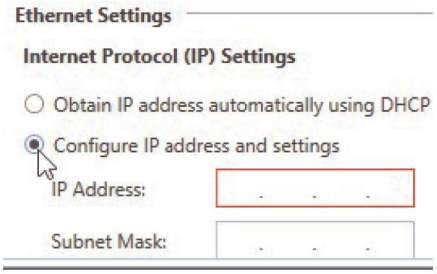
การเปลี่ยนค่า IP address ต้องทำขณะ Disconnect เริ่มต้นด้วยการคลิกที่แท็บคอนโทรลเลอร์ Micr850 ดังรูป



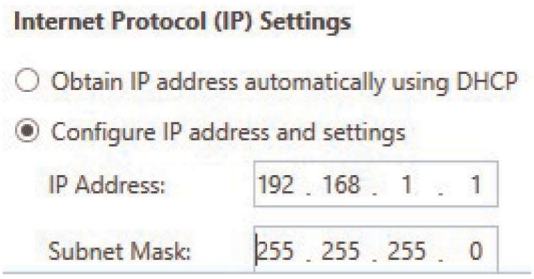
จะปรากฏหน้าต่างดังรูปข้างล่าง ที่ Controller ให้คลิกที่ Ethernet จะปรากฏค่าตั้งของ Ethernet ขึ้นมา โดยปกติค่าที่ตั้งมาจากโรงงานจะเป็น Obtain IP address automatically using DHCP



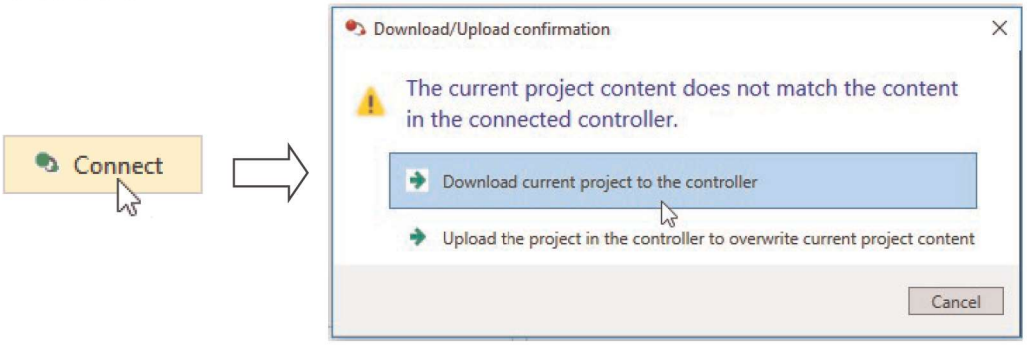
ให้คลิกที่ Configure IP address and setting จากปรากฏช่องให้ตั้งค่า IP Address



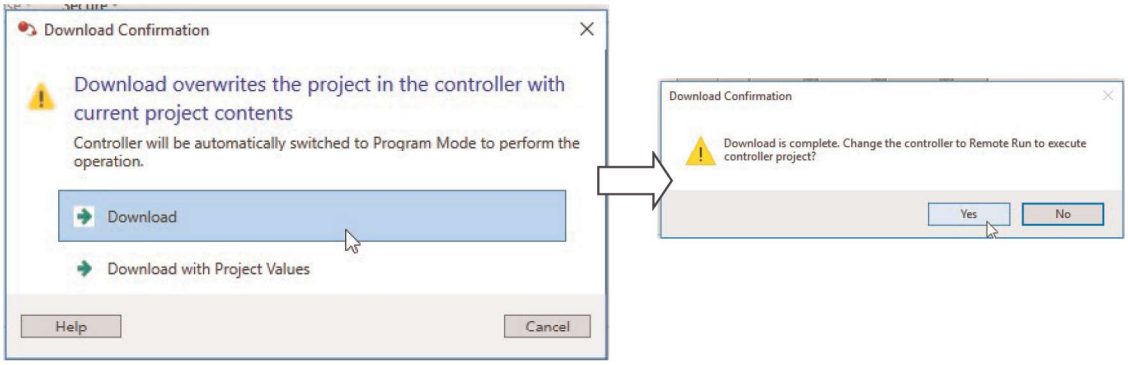
จากนั้นให้ป้อนค่า IP address ที่ต้องการ ในตัวอย่างนี้ คือ 192.168.1.1 และ Subnet Mask คือ 255.255.255.0



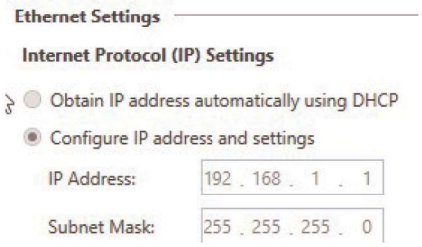
เมื่อตั้งค่าเสร็จแล้วให้คลิกที่ Connect เพื่อเชื่อมต่อกับคอนโทรลเลอร์ จากนั้นซอฟต์แวร์แจ้งว่าข้อมูลของคอมพิวเตอร์ไม่เหมือนกับคอนโทรลเลอร์ ให้เลือก Download current project to the controller



จากนั้นจะปรากฏหน้าต่างเพื่อให้เรายอมรับการเขียนโปรแกรมทับโปรแกรมเดิมที่อยู่ในคอนโทรลเลอร์ ให้เลือก Download และคลิก Yes จากนั้นกระบวนการดาวน์โหลดจะเริ่มขึ้น

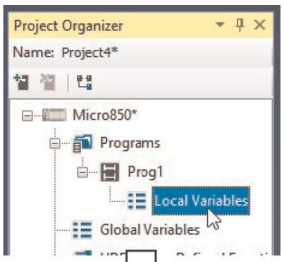


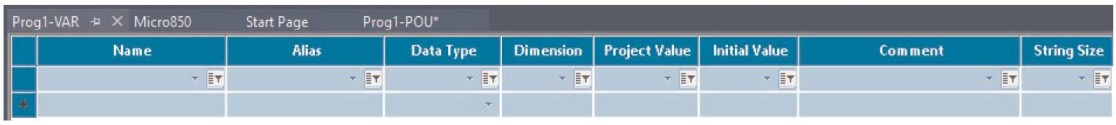
ถ้าการดาวน์โหลดเสร็จสมบูรณ์จะสังเกตุเห็นว่า IP Address จะมีเทาและเปลี่ยนแปลงค่าไม่ได้ ดังนั้นเมื่อนำ PLC ตัวนี้ไปเชื่อมต่อในเครือข่าย Ethernet ต้องอ้าง IP ตามที่ตั้งไว้



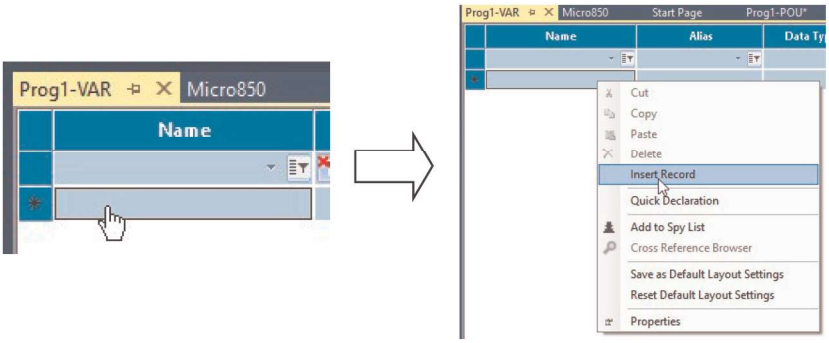
11.7 การสร้างตัวแปร Array

การสร้างและการเรียกใช้งานตัวแปร Array สามารถทำได้โดยดับเบิลคลิกที่ Local Variables ใน Project Organizer จะปรากฏหน้าต่าง Prog1-VAR ขึ้นมา

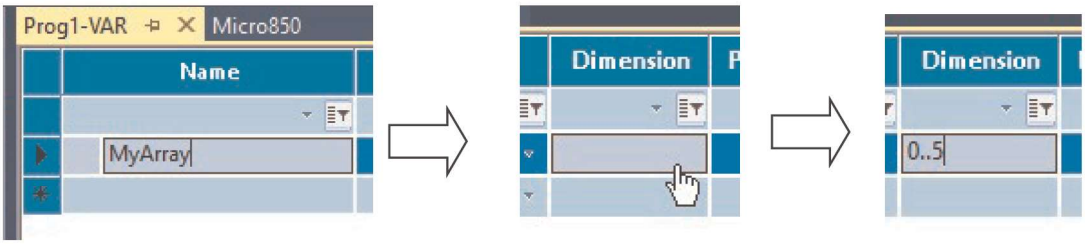




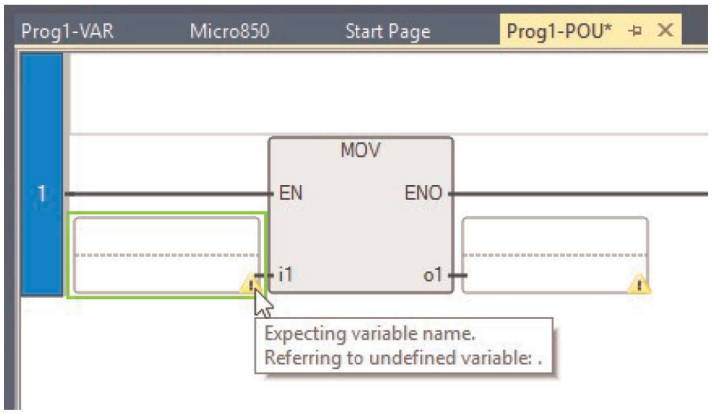
ให้คลิกขวาที่ Name ที่เราต้องการสร้างตัวแปร จากนั้นเลือก Insert Record



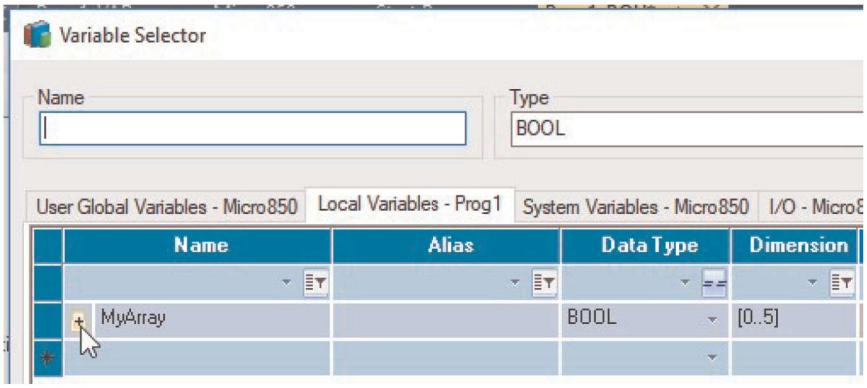
จากนั้นให้พิมพ์ชื่อตัวแปรที่ต้องการตั้ง ในที่นี้ คือ MyArray ส่วน Data Type ให้เลือกตามที่ต้องการใช้งาน จากนั้นให้ไปที่ช่อง Dimension แล้วใส่ขนาด Array ที่ต้องการในตัวอย่างมีขนาด 0..5



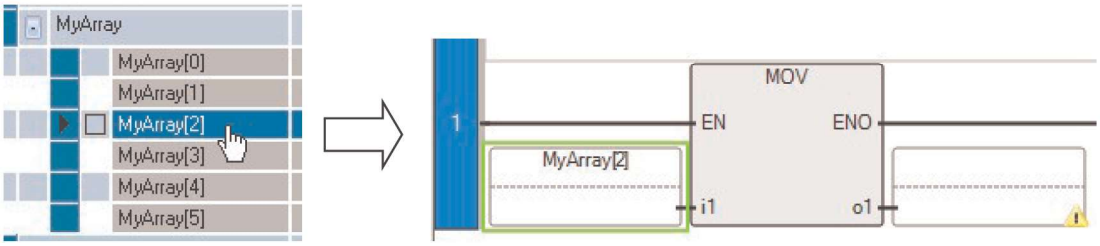
การใช้งานตัวแปร Array ในโปรแกรมแลดเดอร์ ให้คลิกที่ Prog1-POU ในตัวอย่างนี้เป็นคำสั่ง MOV เราจะแสดงการใส่ตัวแปรอินพุตโดยการดับเบิลคลิกที่เครื่องหมาย ! ดังรูปข้างล่างนี้



จะปรากฏหน้าต่าง Variable Selector ขึ้นมาให้คลิกที่เครื่องหมาย + ตามรูป



จากนั้นจะปรากฏตัวแปร Array แต่ละตัวของ MyArray ในตัวอย่างนี้เลือก MyArray[2] แล้วคลิก OK จากนั้นจะปรากฏตัวแปร MyArray[2] ในโปรแกรมแลดเดอร์ตามรูป



สรุปท้ายบท

ในบทนี้เราได้แนะนำการใช้งานซอฟต์แวร์ CCW เพิ่มเติม ซึ่งเป็นการใช้งานทั่วไปของซอฟต์แวร์ จริงแล้วซอฟต์แวร์ CCW ยังมีคุณสมบัติอื่นๆ อีกหลายอย่าง เช่น ใช้สำหรับตั้งค่าพารามิเตอร์ของไดร์ฟ ตั้งค่าออปชั่นบอร์ด เป็นต้น แต่เราจะไม่ขอกล่าวถึงในที่นี้

Chapter

12

การออกแบบลอจิกเชิงโครงสร้าง

โปรแกรมลอจิกแลดเดอร์หรือแลดเดอร์ไดอะแกรมแบบทั่วไป มักเขียนขึ้นโดยการพิจารณาจากกระบวนการทำงานของระบบหรือเครื่องจักร จากนั้นจึงเริ่มต้นเขียนโปรแกรมทันทีซึ่งจะทำให้โปรแกรมที่เขียนขึ้นต้องมีการแก้ไข (Debugging) ค่อนข้างมาก ปกติแล้วผู้เขียนโปรแกรมหน้าใหม่จะเขียนโปรแกรมโดยไม่ได้ให้ความสำคัญกับการออกแบบโปรแกรม บ่อยครั้งจะใช้ประสบการณ์ที่ผ่านมาในการเขียนโปรแกรมหรือคิดอะไรได้ก็เขียนขึ้นก่อนโดยไม่คำนึงถึงลำดับเหตุการณ์ต่างๆ ซึ่งวิธีการนี้ใครเป็นคนเขียนคนนั้นต้องเป็นคนแก้ไข เพราะคนอื่นอ่านโปรแกรมไม่เข้าใจ

ดังนั้นการเขียนโปรแกรม PLC เปรียบเสมือนการเขียนเรียงความหรือเขียนหนังสือสักเล่มหนึ่ง เราจะต้องออกแบบโครงสร้าง ลำดับของเนื้อหาต่าง ๆ และเขียนไม่วกไปวนมา ซึ่งจะทำให้หนังสืออ่านได้ง่ายและกระชับ การเขียนโปรแกรมแลดเดอร์ก็เช่นเดียวกัน เทคนิคการออกแบบลอจิกเชิงโครงสร้างจึงมีความสำคัญมากซึ่งจะกล่าวถึงในลำดับถัดไป

ระบบควบคุมส่วนใหญ่จะทำงานเป็นลำดับขั้น (Sequential) ระหว่างการทำงานปกติระบบจะมีการทำงานหลายขั้นตอน (Step) ในแต่ละขั้นตอนระบบจะกระทำการหรือทำงานที่แตกต่างกันไป ซึ่งรวมถึงขั้นตอนการ Start-up, Shut-Down และการทำงานปกติอื่นๆ ลองพิจารณาสัญญาณไฟจราจรซึ่งไฟแต่ละดวงมีรูปแบบสัญญาณไฟตามช่วงของเวลาที่เปลี่ยนไปในสภาวะหนึ่งๆ เช่น ไฟอาจเป็นสีเขียวหรือเหลืองที่ทิศทางหนึ่ง ในขณะที่ทิศทางอื่นๆ เป็นสีแดงทั้งหมดและสัญญาณไฟจะเปลี่ยนแปลงเป็นลำดับขั้นเช่นนี้ไปเรื่อยๆ บางครั้งระบบสัญญาณไฟจราจรอาจมีอุปกรณ์พิเศษเพิ่มเติม เช่น ปุ่มกดขอข้ามถนน (Cross walk) ซึ่งจะเปลี่ยนสัญญาณไฟเพื่อให้คนข้ามถนนได้

ระบบควบคุมซีเควนซ์ที่มีความยุ่งยากซับซ้อนในการออกแบบ การใช้แผนภูมิเวลาและบิตซีเคว้นซ์อย่างง่ายไม่อาจตอบโจทย์ ดังนั้น การออกแบบจะต้องใช้เทคนิคมากขึ้นสำหรับระบบที่ง่ายเราอาจใช้เทคนิคของบิตซีเคว้นซ์และโฟล์ชาร์ท ในระบบที่ซับซ้อนขึ้น เช่น ไฟจราจรอาจมีสภาวะการทำงานหลายสภาวะซึ่งการใช้สเตทไดอะแกรม (State diagram) จะเหมาะสม

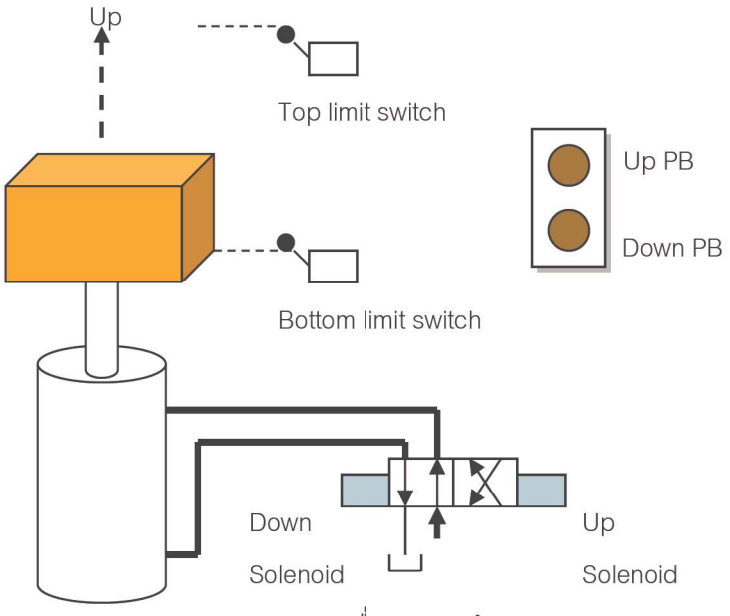
กว่า แต่ถ้ามีไฟจราจรหลายๆ ชุดทำงานร่วมกัน เช่น มีไฟจราจร 2 ชุด สำหรับทางแยก 2 จุด ที่ต้องทำงานประสานกัน การใช้ SFC อาจเหมาะสมกว่า

12.1 การเขียนโปรแกรมบิตซีเควนซ์ (Sequence Bit)

บิตซีเควนซ์เป็นวิธีการออกแบบโปรแกรมที่ง่ายที่สุดและเหมาะกับระบบควบคุมที่ไม่ซับซ้อนมากนัก เครื่องจักรโดยทั่วไปจะมีลำดับของขั้นตอน (step) การทำงานที่สามารถแยกแยะได้ชัดเจนซึ่งเราสามารถเขียนโปรแกรมแลดเดอร์ตามลำดับขั้นตอนได้ โดยมีขั้นตอนในการออกแบบ ดังนี้

1. ทำความเข้าใจกระบวนการทำงาน (Process)
2. เขียนขั้นตอนการทำงานเป็นลำดับขั้น (Sequence) และกำหนดหมายเลขขั้นตอนการทำงานหรือหมายเลขสเตป (Step)
3. กำหนดบิตใช้งานของแต่ละสเตป (Step)
4. เขียนโปรแกรมแลดเดอร์เพื่อให้บิตเหล่านั้น ON/OFF ตามกระบวนการในแต่ละขั้นตอนของมัน
5. เขียนโปรแกรมแลดเดอร์เพื่อสั่งให้เครื่องจักรทำงานสำหรับสเตป (Step) นั้นๆ
6. ถ้ากระบวนการทำงานซ้ำ ให้สเตป (Step) สุดท้ายกลับไปที่สเตป (Step) แรก

ตัวอย่างที่ 12.1 ต่อไปลองดูตัวอย่างการออกแบบโปรแกรมบิตซีเควนซ์ ในรูปที่ 12.1 จะแสดงเครื่องยกสินค้าขึ้นและลงโดยใช้สวิตช์กด (Push Button) เริ่มต้นการทำงานเราต้องเขียนรายละเอียดกระบวนการทำงานของเครื่องก่อน จากนั้นแปลงเป็นขั้นตอนหรือสเตป(Step) และแต่ละสเตปให้แปลงเป็นวงจรแลดเดอร์ ดังแสดงในรูปที่ 12.2



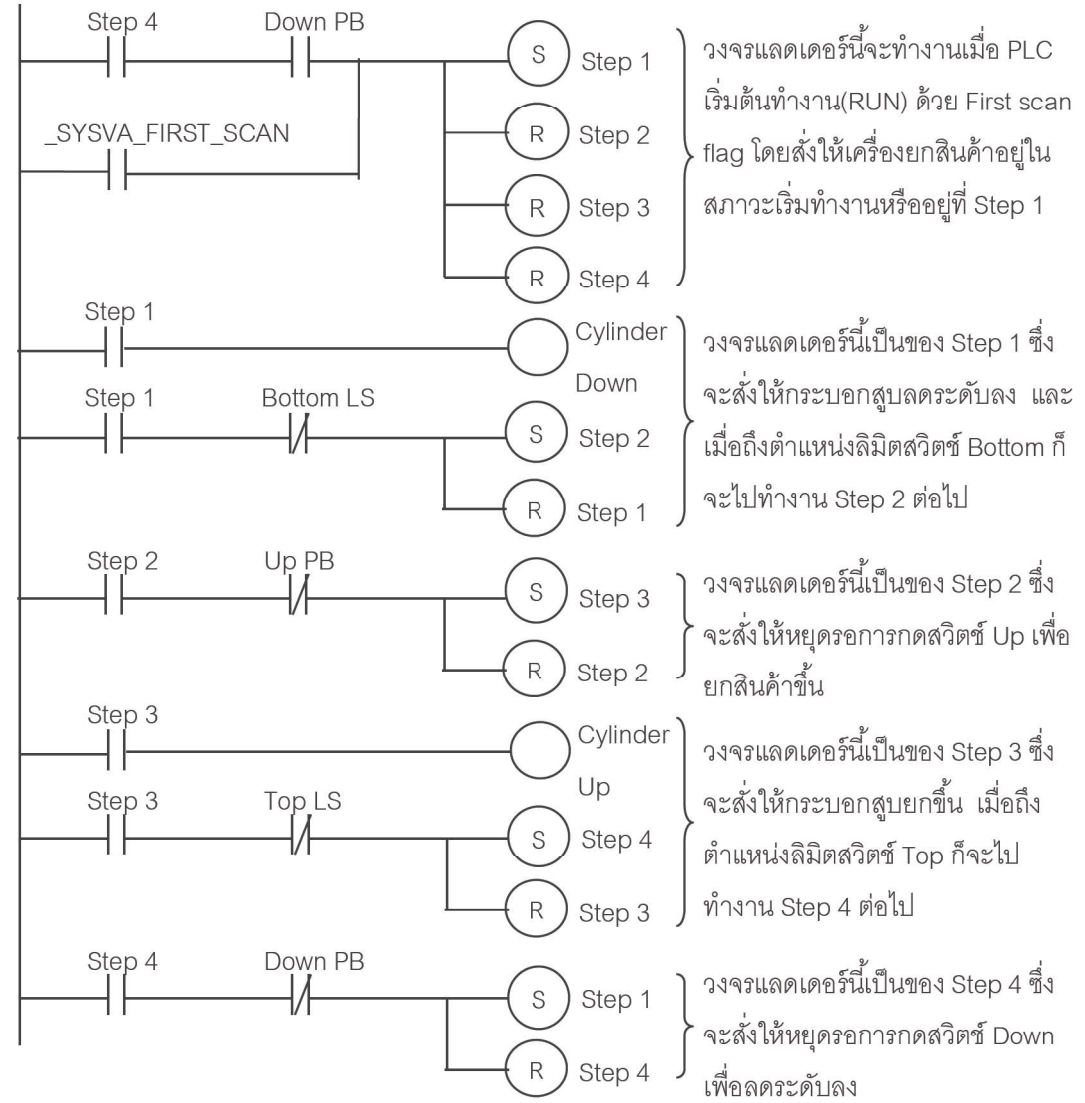
รูป 12.1 เครื่องยกสินค้า

รายละเอียดการทำงาน

เครื่องยกสินค้าจะยกขึ้นเมื่อสวิทซ์ Up ถูกกด และลดระดับลงเมื่อสวิตซ์ Down ถูกกด สวิตซ์ทั้งสองเป็นแบบกดติดปล่อยดับ นอกจากนั้นมีลิมิตสวิตซ์ 2 ตัว เพื่อตรวจสอบว่าเครื่องอยู่ในตำแหน่งยกขึ้นหรือลง เมื่อเปิดไฟจ่ายเครื่องยกสินค้าจะต้องลดระดับลงจนกระทั่งอยู่ที่ตำแหน่งลงสุดหรือที่ลิมิตสวิตซ์ Bottom ถูกกด

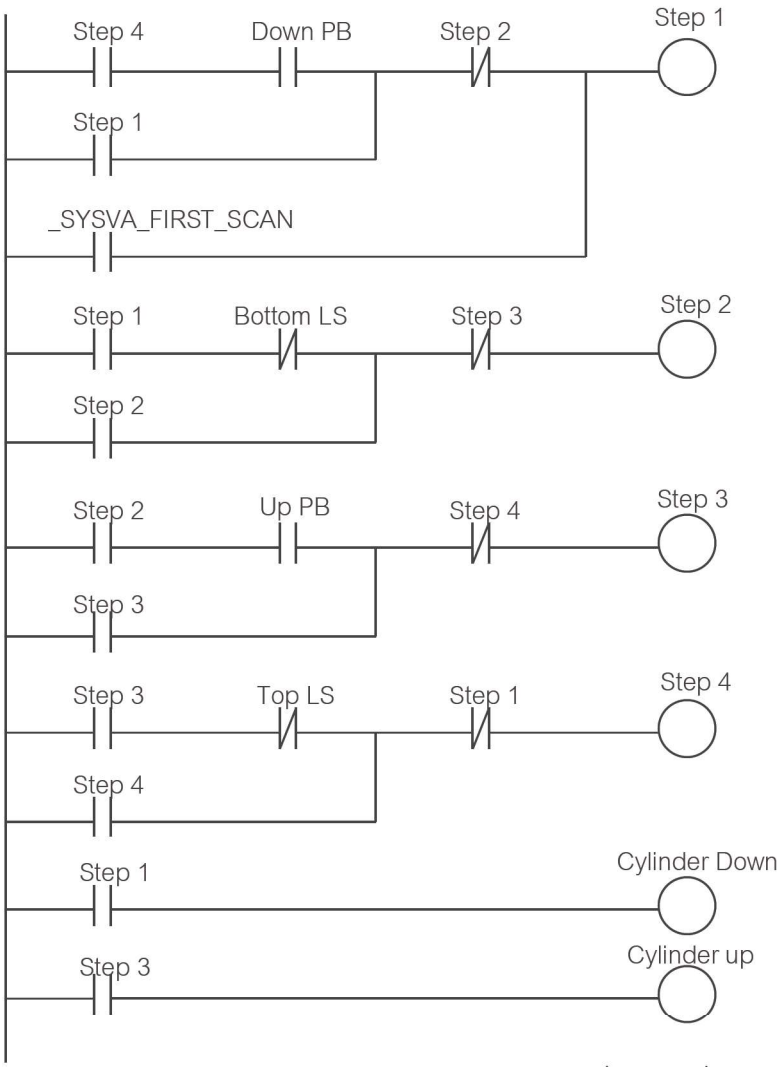
ขั้นตอนการทำงาน (Step)

1. เครื่องยกสินค้าลดระดับลงจนกระทั่งอยู่ในตำแหน่งลิมิตสวิตซ์ Bottom
2. เครื่องยกสินค้าจะหยุดรอที่ตำแหน่งลงสุด เพื่อรอการกดสวิตซ์ Up
3. เครื่องยกสินค้ายกขึ้นจนถึงตำแหน่งลิมิตสวิตซ์ Top
4. เครื่องยกสินค้าหยุดรอที่ตำแหน่งสูงสุด เพื่อรอการกดสวิตซ์ Down



รูป 12.2 ตัวอย่างการออกแบบบิตซีเควนซ์โดยใช้คำสั่ง SET/RESET

วิธีการของบิตซีเควนซ์ในรูปที่ 12.2 จะใช้คำสั่งบิต SET(S) และ RESET(R) ซึ่งบางครั้งทำให้รู้สึกว่ายาก แต่ยังมีวิธีที่นิยมใช้ทั่วไปโดยไม่ต้องใช้คำสั่ง SET/RESET ดังแสดงในรูป 12.3 ลำดับการทำงานของซีเควนซ์ยังเหมือนเดิมเพียงแต่แลดเดอร์ในแต่ละเตปจะใช้ Self interlock เพื่อให้แต่ละสเตปทำงานค้างอยู่ได้



รูป 12.3 ตัวอย่างการออกแบบบิตซีเควนซ์โดยใช้คำสั่งลอจิกทั่วไป

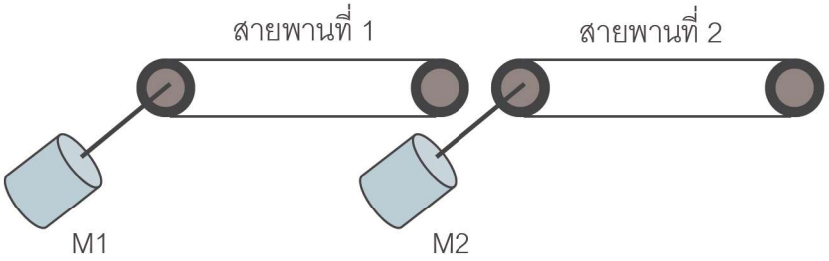
12.2 แผนภูมิเวลา (Timing Diagram)

แผนภูมิเวลาอาจเป็นประโยชน์มากกับการออกแบบโปรแกรมแลดเดอร์สำหรับกระบวนการทำงานที่ขึ้นอยู่กับเวลา แผนภูมิเวลาจะถูกวาดด้วยการระบุเวลาเริ่มต้น(Start) และเวลาสิ้นสุด (Stop) โปรแกรมแลดเดอร์จะประกอบด้วยไทม์เมอร์ที่ใช้เพื่อสั่งให้เอาท์พุตทำงาน (ON) และหยุดทำงาน (OFF) ตามเวลาที่กำหนด วิธีการเบื้องต้นประกอบด้วย

1. ทำความเข้าใจกระบวนการ
2. แยกแยะเอาต์พุตที่ทำงานขึ้นอยู่กับเวลา
3. วาดแผนภูมิเวลาของเอาต์พุตเหล่านั้น
4. กำหนดไทม์เมอร์หนึ่งตัวของแต่ละช่วงเวลาสำหรับสั่งให้เอาต์พุตเหล่านั้น ON และ OFF

5. เขียนโปรแกรมแลดเดอร์เพื่อทดสอบค่าเวลาไทม์เมอร์และการทำงานของเอาต์พุต ว่า ON และ OFF ตามที่กำหนดหรือไม่

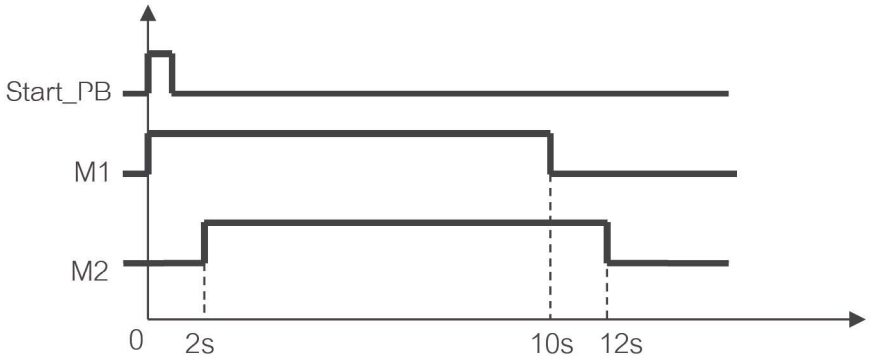
ตัวอย่างที่ 12.2 เราลองมาดูตัวอย่างง่าย ๆ ในการควบคุมระบบสายพานลำเลียง 2 ชุด ที่เริ่มต้นทำงานไม่พร้อมกัน และหยุดทำงานพร้อมกันเมื่อเวลาผ่านไป เริ่มต้นด้วยการอธิบายรายละเอียดการทำงานดังรายละเอียดข้างล่างนี้



รูป 12.4 ตัวอย่างสายพานลำเลียง

รายละเอียดการทำงาน

เมื่อกดสวิตช์ Start สายพานลำเลียงชุดที่ 1 จะเริ่มทำงาน จากนั้นอีก 2 วินาที สายพานชุดที่ 2 จะเริ่มทำงาน เมื่อสายพานชุดที่ 1 หยุดทำงานเมื่อครบ 10 วินาที สายพานทั้ง 2 ชุด จะหยุดทำงานหลังสายพานชุดที่ 1 หยุดทำงาน 2 วินาที แต่ถ้ามีการกดสวิตช์ Stop สายพานทั้ง 2 จะหยุดทำงานทันที



รูป 12.5 แผนภูมิเวลาของระบบสายพานลำเลียง

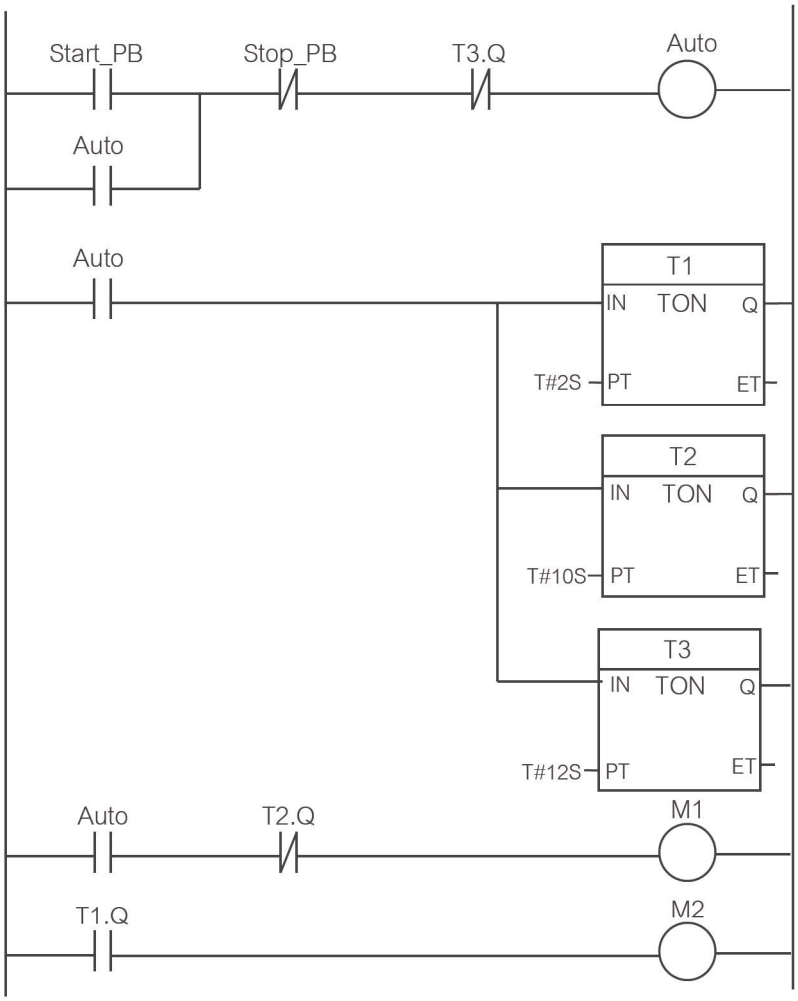
จากรูป 12.5 เป็นแผนภูมิเวลาแสดงการทำงานระบบสายพานลำเลียง โดยแกนตั้งของแผนภูมิเวลาคือ ลอจิกของการทำงาน ‘0’ กับ ‘1’ ส่วนแกนนอนคือ เวลา จากแผนภูมิเวลาทำให้ทราบว่าอุปกรณ์แต่ละตัวทำงานเมื่อใด จากนั้นนำไปแปลงเป็นโปรแกรมแลดเดอร์ดังแสดงในรูปที่ 12.6

โปรแกรมแลดเดอร์วงจรแรก จะใช้เพื่อ Start และ Stop สายพานลำเลียงโดยมีบิต Auto (เอาต์พุต) ทำหน้าที่ล็อกการทำงานไว้หลังจากกดปุ่ม Start_PB ขณะเดียวกันมันยังใช้เพื่อสั่งให้สายพานชุดที่ 1 ทำงานด้วย

ไทม์เมอร์ T1 จะนับเวลา 2 วินาที เมื่อนับเวลาครบจะสั่งให้สายพานชุดที่ 2 ทำงาน

ไทม์เมอร์ T2 จะนับเวลา 10 วินาที เมื่อนับเวลาครบจะสั่งให้สายพานชุดที่ 1 หยุดทำงาน

ไทม์เมอร์ T3 จะนับเวลา 12 วินาที เมื่อนับเวลาครบจะสั่งให้สายพานชุดที่ 2 หยุดทำงาน พร้อมทั้งยกเลิกการทำงานทั้งหมด ส่วนบิตเอาต์พุต Auto ในแลดเดอร์ไดอะแกรมวงจรแรกจะหยุดทำงาน(OFF) เมื่อ T3 ทำงาน(ON) เช่นกัน

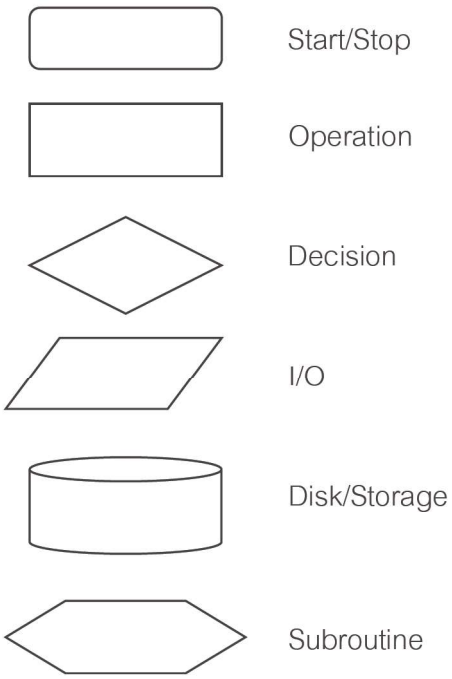


รูป 12.6 ไดอะแกรมแลดเดอร์ของสายพานลำเลียง

แผนภูมิเวลาอาจนำไปใช้ร่วมกับการออกแบบชนิดอื่นๆ เพื่อใช้ประกอบความเข้าใจการทำงานของระบบให้ง่ายขึ้น

12.3 โฟลว์ชาร์ต (Flow Chart)

โฟลว์ชาร์ตเป็นอีกวิธีการหนึ่งที่ช่วยในการออกแบบซีเควนซ์ของระบบควบคุม โดยการถ่ายทอดความเข้าใจของระบบให้อยู่ในรูปภาพหรือสัญลักษณ์ ซึ่งจะทำงานภายใต้การจัดลำดับขั้นตอนอย่างง่ายๆ และสัญลักษณ์แต่ละอันจะถูกเชื่อมโยงเข้าด้วยกันโดยใช้ลูกศร ตัวอย่างสัญลักษณ์ของโฟลว์ชาร์ตแสดงในรูปที่ 12.7 ปกติแล้วโฟลว์ชาร์ตจะต้องขึ้นต้นด้วยสัญลักษณ์ 'Start' เสมอ และสิ้นสุดด้วย 'Stop' เนื่องจากโปรแกรม PLC จะไม่หยุดทำงาน ดังนั้น สัญลักษณ์ 'Stop' จึงไม่ได้ใช้งาน ส่วนสัญลักษณ์ที่สำคัญ ได้แก่ 'Operation' และ 'Decision' ในขณะที่สัญลักษณ์อื่นๆ อาจถูกใช้งานบ้างแต่ไม่จำเป็นมากสำหรับการสร้างโปรแกรม PLC

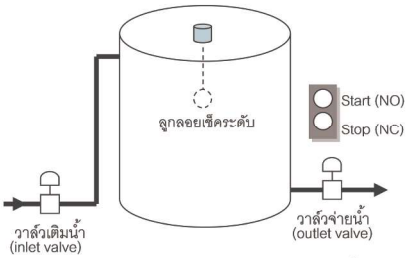


รูป 12.7 สัญลักษณ์โฟลว์ชาร์ต

ในรูปที่ 12.8 แสดงตัวอย่างระบบควบคุมถังเก็บน้ำ ส่วนรูปที่ 12.10 แสดงโฟลว์ชาร์ตการทำงานของระบบ เมื่อกดสวิตช์ Start จะเริ่มเติมน้ำเข้าถังและจะหยุดเติมน้ำเมื่อน้ำเต็มหรือกดสวิตช์ Stop ซึ่งจะทำให้ท่อเติมน้ำจะถูกปิดและน้ำในถังจะถูกจ่ายออกไปได้ตามปกติ

ในโฟลว์ชาร์ตจะเริ่มจาก 'START' ที่อยู่ด้านบนสุด การทำงานอย่างแรกของระบบ คือ เปิดวาล์วจ่ายน้ำ (Outlet valve) และปิดวาล์วเติมน้ำ (Inlet valve) ถัดไปเป็นสัญลักษณ์ตัดสินใจ (Decision) ถูกใช้เพื่อรอการกดสวิทช์ Start เมื่อสวิตช์ถูกกดนั่นคือเส้นทาง Yes ของสัญลักษณ์ตัดสินใจ จะทำให้วาล์วเติมน้ำเปิดและวาล์วจ่ายน้ำจะปิด ดังนั้น เส้นทางจะไปที่บล็อกตัดสินใจอีก 2 บล็อก เพื่อตรวจสอบว่าถังเต็มหรือมีการกดสวิทช์ Stop (ใช้หน้าคอนแทค NC) หรือไม่ ถ้า

เหตุการณ์อย่างใดอย่างหนึ่งเกิดขึ้น วาล์วเติมน้ำจะปิดและวาล์วจ่ายน้ำจะเปิด ระบบก็จะกลับไปรอการกดปุ่ม Start ใหม่อีกครั้งหนึ่ง โปรแกรมจะทำงานเช่นนี้ตลอดเวลาในขณะที่ PLC ยังคงทำงาน ดังนั้น สัญลักษณ์ Start จะถูกใช้เพียงครั้งเดียวเท่านั้นตอนเริ่มจ่ายไฟ

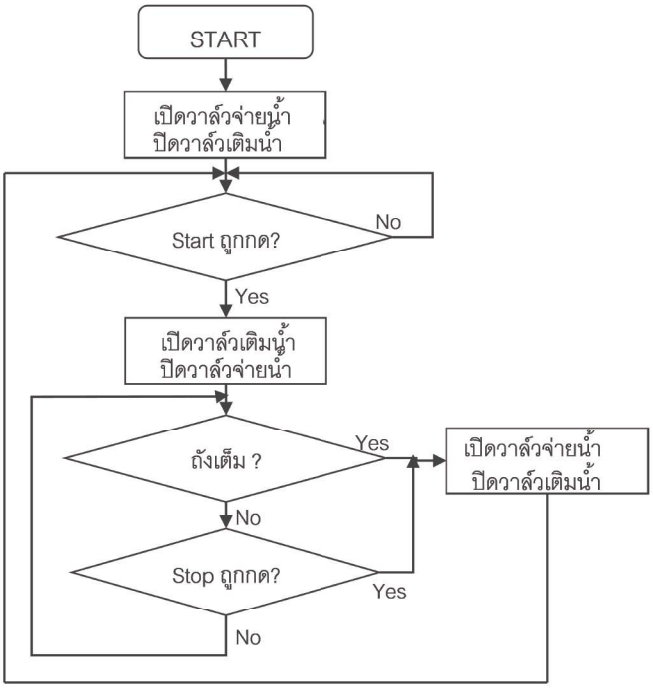


รูป 12.8 ระบบควบคุมถังเก็บน้ำ

วิธีการสร้างโฟลว์ชาร์ต ประกอบด้วย

1. ทำความเข้าใจกระบวนการทำงาน
2. กำหนดฟังก์ชั่นการทำงานหลักและวาดสัญลักษณ์
3. กำหนดลำดับขั้นการทำงาน แล้วลากลูกศรเชื่อมโยง
4. ถ้าลำดับขั้นการทำงาน มีการเปลี่ยนแปลงจะใช้สัญลักษณ์ตัดสินใจ (Decision) เพื่อแยกเส้นทาง

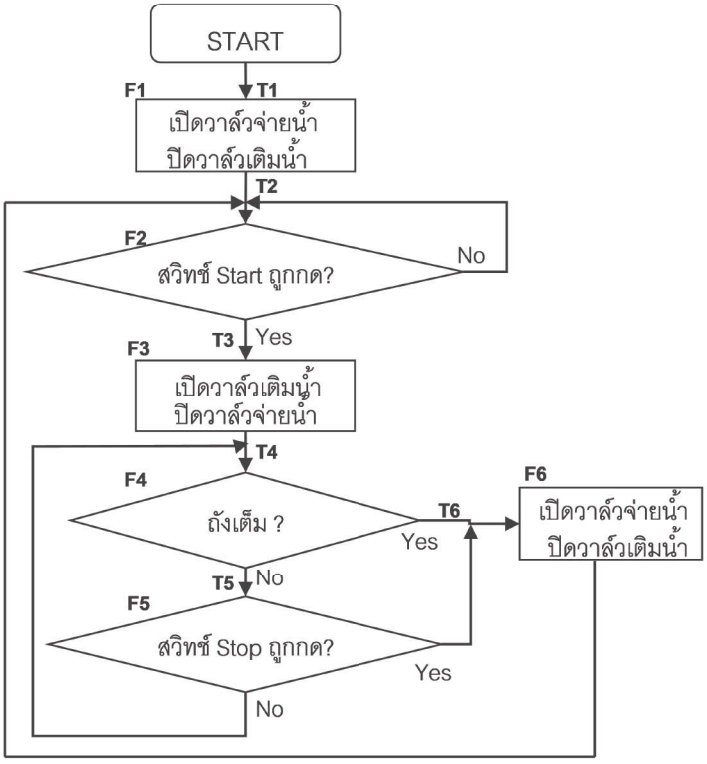
เมื่อสร้างโฟลว์ชาร์ตเสร็จแล้วจึงเริ่มต้นเขียนโปรแกรมแลดเดอร์ จากนั้นเขียนโปรแกรมแลดเดอร์ด้วยวิธีการปกติหรือบิตซีเคว้นซ์นั่นเอง



รูป 12.9 โฟลว์ชาร์ตของระบบควบคุมถังเก็บน้ำ

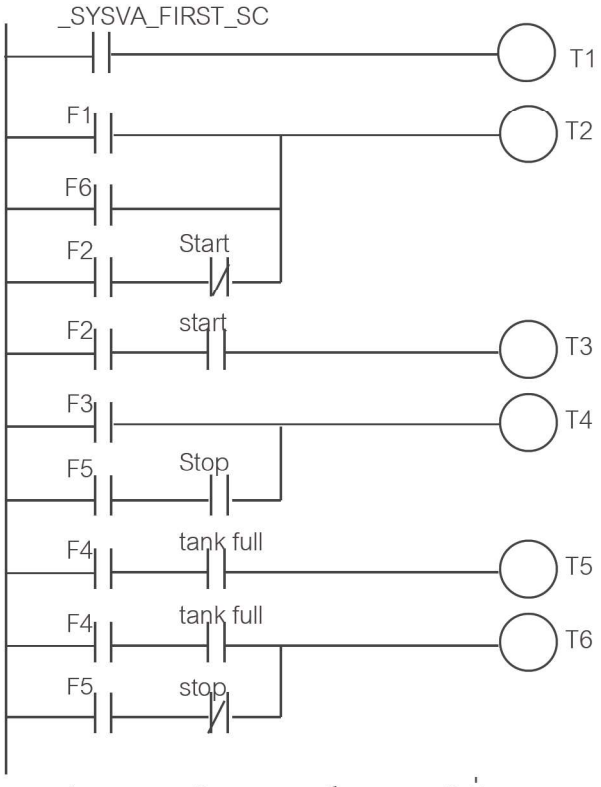
การสร้างโปรแกรมจากโฟลว์ชาร์ตด้วยบิตซีเคว้นซ์

ขั้นตอนแรกให้นำโฟลว์ชาร์ตก่อนหน้านี้ (รูป 12.9) ที่มีชื่อกำกับในแต่ละบล็อก จากนั้นให้ใส่ชื่อกำกับเส้นของลูกศรหรือทรานซิชั่น(Transition) ที่เชื่อมต่อระหว่างบล็อกดังแสดงในรูป 12.10 ซึ่งทรานซิชั่น(Transition) นี้จะใช้เพื่อบอกว่าเมื่อใดแลดเดอร์อีกบล็อกหนึ่งจะทำงาน



รูป 12.10 โฟลว์ชาร์ตที่กำหนดชื่อทรานซิชั่น

วงจรแลดเดอร์ส่วนแรกในรูปที่ 12.11 เขียนขึ้นเพื่อกำหนดว่าเมื่อไรจะมีการทรานซิชั่นหรือการส่งผ่านระหว่างบล็อก วงจรแลดเดอร์ของทรานซิชั่นทั้งหมดควรอยู่ด้วยกันและวางไว้ก่อนหน้าแลดเดอร์ส่วนอื่นๆ ที่จะตามมา

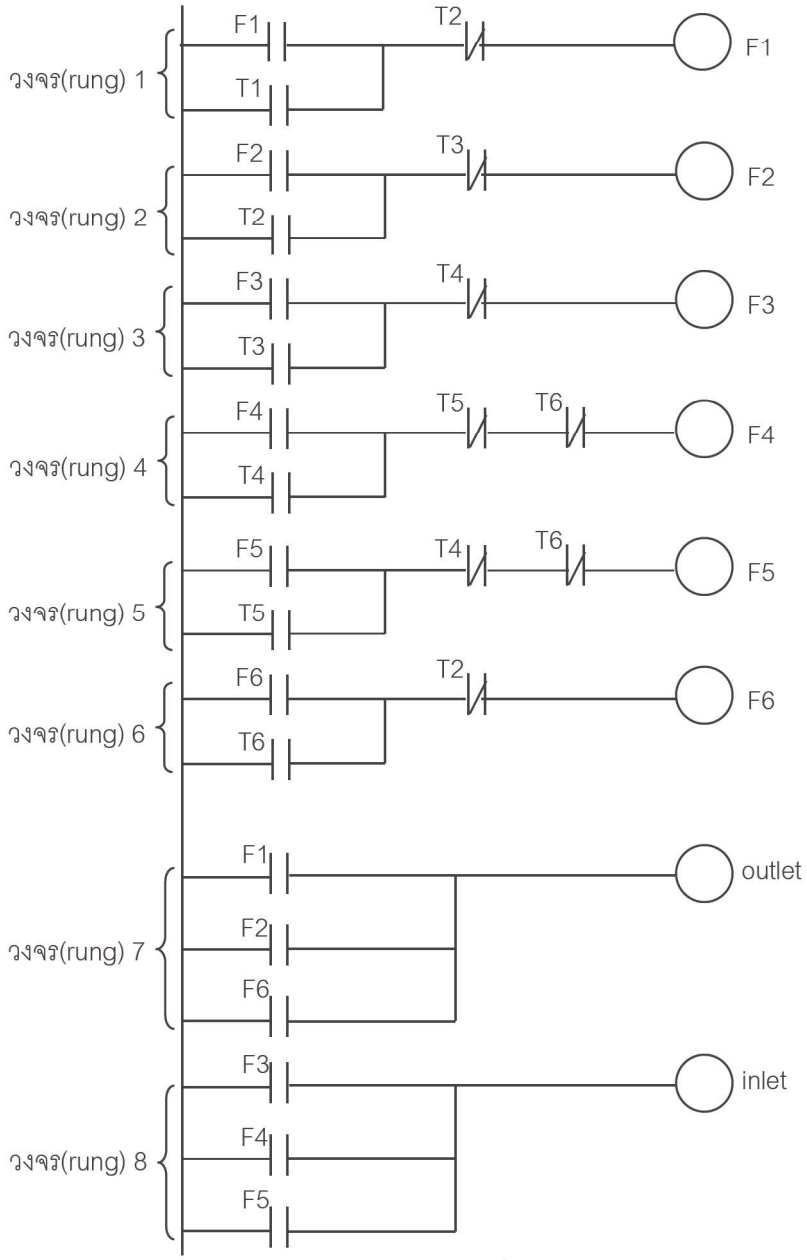


รูป 12.11 ลอจิกแลดเดอร์ของทรานซิชั่น

<u>หมายุเหตุ</u> สวิทช์ stop ใช้ Normally closed (NC) ซึ่งลอจิกจะตรงกันข้ามกับสวิตช์จริง

ลอจิกแลดเดอร์ในรูปที่ 12.12 แสดงให้เห็นว่าฟังก์ชั่นใดทำงาน หรือกำลังเปลี่ยนไปยังฟังก์ชันถัดไป ลองพิจารณาแลดเดอร์ในวงจรแรกของ F1 มันจะถูกสั่งให้ทำงานโดยทรานซิชั่น T1 และเมื่อฟังก์ชั่น F1 ทำงาน(ON) มันจะรักษาการทำงานไว้ด้วยหน้าคอนแทคของตัวมันเอง จนกว่าทรานซิชั่น T2 จะทำงานและสั่งให้ F1 หยุดทำงาน ในขณะเดียวจะสั่งให้ฟังก์ชั่น F2 ทำงาน

ส่วนลอจิกแลดเดอร์ในวงจรที่ 7 และ 8 เป็นส่วนของเอาต์พุตเพื่อควบคุมวาล์วเติมน้ำและวาล์วจ่ายน้ำ เอาต์พุตจะทำตามเงื่อนไขของฟังก์ชั่น เช่น วาล์วจ่ายน้ำจะเปิดเมื่อฟังก์ชั่น F1, F2 และ F6 ทำงาน



รูป 12.12 ลอจิกแลดเดอร์ของฟังก์ชั่นและเอาต์พุต

12.4 **สเตทไดอะแกรม** (State Diagram)

สเตทไดอะแกรม (State Diagram) เป็นวิธีการช่วยในการออกแบบลอจิกแลดเดอร์อีกแบบหนึ่งที่สามารถใช้กับงานที่ซับซ้อนมากขึ้น ส่วนใหญ่แล้วไม่มีการกล่าวถึงวิธีการนี้ในบ้านเราทั้งที่วิธีการนี้จะช่วยพัฒนาโปรแกรมได้ดีกว่าการใช้บิตซีเควนซ์ทั่วไป อีกทั้งมันยังช่วยให้การแก้ปัญหาระบบทำได้ง่ายด้วยการอ่านสเตทไดอะแกรม แทนที่จะอ่านวงจรแลดเดอร์ที่ยาวเยียด

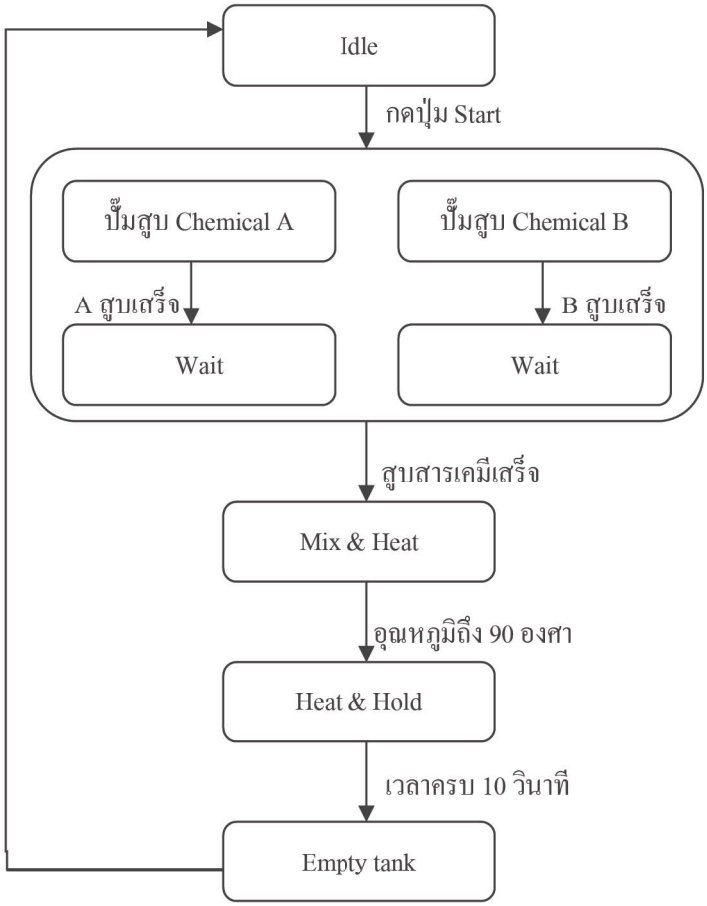
นอกจากนั้นยังมีความยืดหยุ่นสูงกว่าโฟร์ชาร์ตที่มีเส้นทางการไหลของกระบวนการเพียงทางเดียว ในขณะที่สเตทไดอะแกรมจะคำนึงถึงสภาวะ(State) การทำงานของระบบเป็นหลัก โดยสภาวะหนึ่งๆ ของระบบก็คือโหมดการทำงานนั่นเอง ซึ่งต่อไปเราจะใช้คำว่า “สเตท” (state) แทนโหมดการทำงานนี้ เราลองพิจารณาดูตัวอย่างง่ายๆ ของระบบการทำงานของเครื่อง ATM ซึ่งจะประกอบด้วยหลายสเตท โดยมีลำดับขั้นตอนของแต่ละสเตทดังนี้

1. เครื่องว่าง หรือรอการทำงาน (Idle)
2. สแกนบัตร
3. ป้อนรหัส
4. เลือกประเภทธุรกรรม
5. รับการใส่จำนวนเงิน
6. นับเงิน
7. ส่งมอบเงิน พร้อมคืนบัตร
8. กลับไปที่ สเตทเครื่องว่าง

เราจะเห็นว่าเครื่อง ATM จะแบ่งสเตทการทำงานเป็นหลายสเตทและการเปลี่ยนสเตทจากสเตทหนึ่งไปยังอีกสเตทหนึ่งจะต้องมีตัวกระตุ้นหรือเรียกว่า ทรานซิชั่น(Transition) จากตัวอย่างเครื่อง ATM เมื่อสอดบัตรจะมีเซ็นเซอร์ตรวจสอบซึ่งตัวสั่งให้เปลี่ยนจาก ‘สเตทว่าง’ ไปเป็นสเตท ‘สแกนบัตร’

ลองมาดูอีกตัวอย่างหนึ่งที่ใช้ในงานอุตสาหกรรมเป็นระบบควบคุม “ถังผสมเคมี” ซึ่งจะประกอบด้วยสเตทต่าง ๆ ดังนี้

1. เครื่องว่าง(Idle) รอโอเปอร์เรเตอร์กดปุ่ม Start
2. ปั๊มสูบเคมีและวัดการไหลของสารเคมี A และ B เข้าถังผสม
3. หลังจากเติมสารเคมีด้วยการสูบเสร็จแล้ว ตัวผสมจะทำงานและอุณหภูมิในถังจะสูงขึ้น เมื่ออุณหภูมิถึง 90 °c
4. ตัวผสมจะหยุดทำงานและรักษาอุณหภูมิไว้ 10 วินาที
5. สูบสารเคมีที่ผสมแล้ว เข้าถังเก็บ (Empty Tank)
6. กลับไปที่สเตท เครื่องว่าง(Idle)

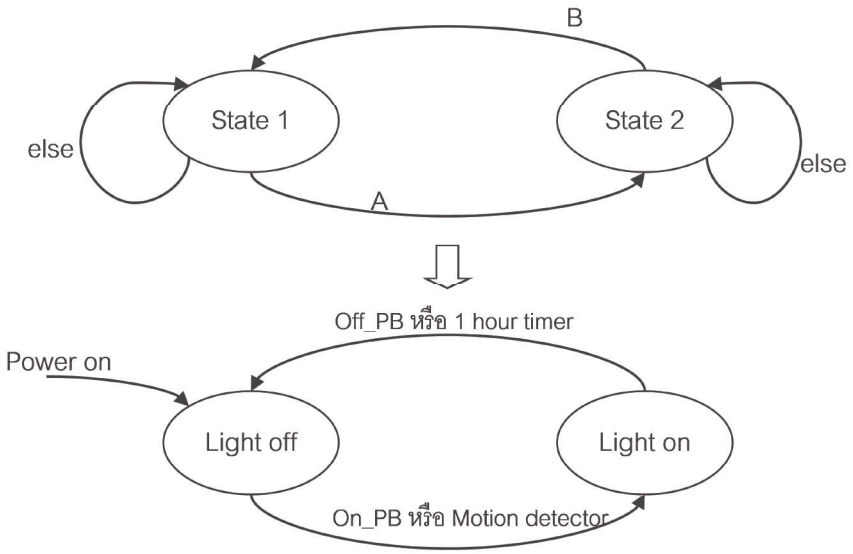


รูป 12.13 แสดงสเตทการทำงานของระบบผสมเคมี

การออกแบบสเตทไดอะแกรม

การออกแบบวงจรแลดเดอร์ด้วยสเตทไดอะแกรมจะต้องอธิบายระบบการทำงานได้ด้วย สเตท (State) และการทรานซิชั่น (Transition) ระหว่างสเตทเหล่านั้น ตัวอย่างสเตทไดอะแกรมที่แสดงในรูปที่ 12.14 จะมี 2 สเตท คือ สเตท 1 และ สเตท 2 ถ้าระบบทำงานอยู่ในสเตท 1 ทรานซิชั่น A เกิดทำงานขึ้นมาระบบจะทรานซิชั่นไปที่สเตท 2 ถ้าระบบอยู่ใน สเตท 2 และทรานซิชั่น B ทำงานระบบจะทรานซิชั่นกลับไปยัง สเตท 1 ส่วน Else Loop ใช้แสดงว่า สเตทนั้นๆ จะยังคงทำงานอยู่ถ้าไม่มีการทรานซิชั่น (Transition) เกิดขึ้น ซึ่งโดยทั่วไปจะละไว้และไม่เขียนลงในสเตทไดอะแกรม

ลองพิจารณาตัวอย่างระบบควบคุมหลอดไฟอัตโนมัติเพื่อใช้ประหยัดพลังงานในห้องทำงาน เมื่อจ่ายไฟให้กับระบบ(Power on) จะเข้าสู่ สเตทปิดไฟ(Light off) ถ้าเซ็นเซอร์ตรวจจับการเคลื่อนไหวได้หรือมีการกดสวิตช์ On_PB ระบบจะทรานซิชั่นสู่ สเตทเปิดไฟ (Light on) ถ้าระบบอยู่ใน สเตทเปิดไฟ(Light on) เป็นเวลา 1 ชั่วโมง หรือมีการกดสวิตช์ Off_PB จะทำให้ระบบส่งผ่านสู่ สเตทปิดไฟ (Light off) ไดอะแกรมในรูปที่ 12.15 แสดงให้เห็นการทำงานดังกล่าว



รูป 12.14 ตัวอย่างสเตทไดอะแกรม (State Diagram)

ส่วนที่เป็นหัวใจสำคัญของการสร้างสเตทไดอะแกรม คือ การแบ่งแยกสเตทต่าง ๆ โดยคำถามหลักที่ต้องพิจารณา คือ

1. พิจารณาระบบ
   >ปกติระบบนั้นๆ ทำหน้าที่อะไร?
   >ลักษณะการทำงานของระบบมีการเปลี่ยนแปลงหรือไม่?
   >ถ้ามีบางสิ่งเปลี่ยนแปลง ระบบจะทำงานอย่างไร?
   >มีลำดับขั้นตอนการทำงานหรือไม่?
2. ให้ลำดับรายการ โหมดการทำงาน โดยแยกเป็นกิจกรรมหนึ่งๆ ที่จะทำงานและหยุดทำงานในกิจกรรมนั้น จำไว้ด้วยว่าบางกิจกรรมมันอาจเป็นการรอ (Wait หรือ Idle) เท่านั้นก็ได้

ตัวอย่าง รูปที่ 12.15 แสดงการสร้างสเตทไดอะแกรมของเครื่องขายกาแฟอัตโนมัติ ขั้นตอนแรกให้แยกแยะสเตทของเครื่องขายกาแฟเป็นสเตทต่างๆ ดังรายละเอียดข้างล่างนี้

เริ่มจาก **สเตทว่าง** (Idle State) จากนั้นก็ **สเตทการหยอดเหรียญ** (Inserting coin state) ซึ่งจะแสดงจำนวนเงินที่หยอด เมื่อเหรียญที่หยอดมีจำนวนเพียงพอแล้วผู้ซื้อจะสามารถเลือกประเภทของกาแฟที่ชอบได้ หลังจากนั้นก็เป็น **สเตทชงกาแฟ** (Make coffee state) จะทำงาน ขณะที่กำลังชงกาแฟ ถ้ามีข้อผิดพลาด(error) เกิดขึ้น **สเตทบริการฉุกเฉิน** (Service needed state) จะทำงาน

รายละเอียดสเตท

Idle – เครื่องว่าง

Inserting coin – เหรียญถูกหยอดและนับจำนวนเหรียญพร้อมการแสดงผล

User choose – เหรียญที่หยอดเพียงพอและผู้ซื้อเลือกกาแฟที่ต้องการ

Make coffee – ชงกาแฟที่ถูกเลือก

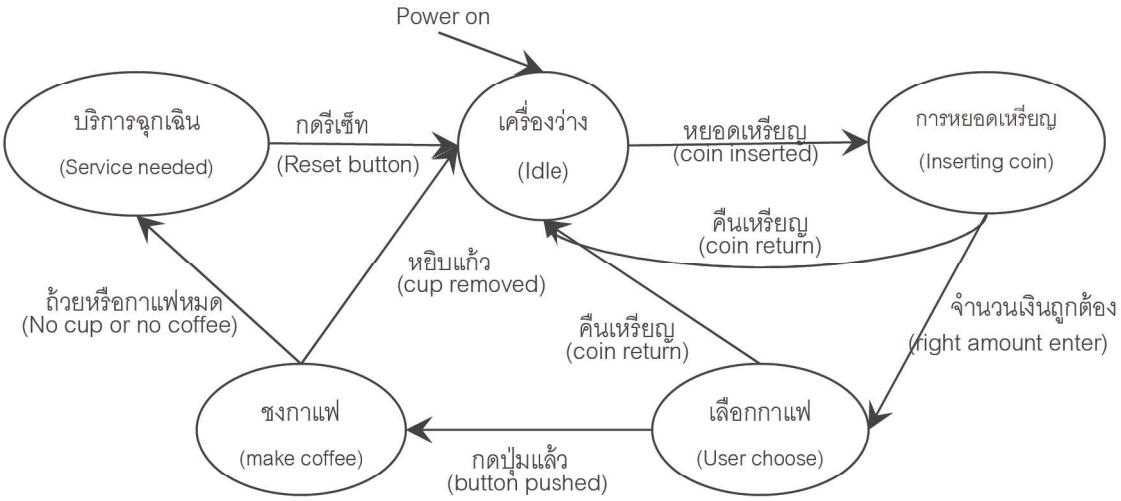
Service needed – เครื่องต้องการบริการฉุกเฉิน เช่น ถ้วยหรือกาแฟหมด เป็นต้น

สเตทต่างๆสามารถนำมาสร้างเป็นไดอะแกรมได้ดังรูปที่ 12.15 ส่วนทรานซิชั่น (Transition) จะถูกเพิ่มเข้าไปในระหว่างสเตท ดังนั้น เราจะเห็นว่าเมื่อจ่ายไฟเข้าเครื่อง (Power on) จะเริ่มต้นการทำงานในสเตทว่าง (Idle) ทรานซิชั่นของสเตทต่างๆ จะเป็นไปตามสัญญาณอินพุตและเซ็นเซอร์ของเครื่อง

สเตทไดอะแกรมจะแตกต่างกันไปตามความยากง่ายของระบบ ไดอะแกรมเหล่านี้จะแสดงให้เห็นกระบวนการทำงานของระบบควบคุมได้อย่างชัดเจนเมื่อเทียบกับการสร้างโปรแกรมด้วยบิตซีเควนซ์ จากรูปที่ 12.15 เราจะเห็นว่าเครื่องชงกระแฟมีทำงานอย่างไรและถ้าเปลี่ยนไปทำงานที่สเตทอื่นจะมีทรานซิชั่นอะไรเกี่ยวข้องบ้าง ลองพิจารณาดูสถานการณ์การต่อไปนี้

สมมุติว่าเครื่องชงกาแฟเสร็จแล้ว มันจะยังคงค้างอยู่ที่สเตทนี้จนกว่าจะมีคนหยิบแก้วกาแฟออก ถ้าคนไม่หยิบแก้วออกเครื่องจะไม่ว่างหรือถ้าคนหยิบแก้วออกแต่เซ็นเซอร์เสียเครื่องก็จะค้างที่สเตทนี้ต่อไปโดยไม่ยอมเข้าสู่สเตทเครื่องว่าง ในกรณีนี้ช่างซ่อมเครื่องจะรู้ทันทีว่าเซ็น-เซอร์เสีย

จากไดอะแกรมมีข้อด้อยที่สเตทบริการฉุกเฉิน สมมุติว่ากาแฟหมดเครื่องจะเข้าสู่สเตทบริการฉุกเฉิน ถ้าเราแก้ไขปัญหาด้วยการปิดไฟเข้าเครื่องและเปิดใหม่อีกครั้ง การแจ้งข้อผิด-พลาดของสเตทบริการฉุกเฉิน(Service needed) จะหายไปเพราะเมื่อเปิดไฟเครื่องจะเข้าสู่สเตทว่าง จนกว่าจะมีลูกค้าคนถัดไปหยอดเหรียญและพบว่าไม่ได้รับกาแฟมันจึงเข้าสู่ สเตทบริการฉุกเฉินอีกครั้งหนึ่ง ในการออกแบบจริงเราจึงต้องแก้ไขปัญหาดังกล่าวนี้



รูป 12.15 สเตทไดอะแกรมเครื่องชงกาแฟ

สรุปท้ายบท

บิตซีเควนซ์เป็นการออกแบบโปรแกรมแลดเดอร์ขั้นพื้นฐานที่ต้องทำความเข้าใจการทำงานของระบบและนำมาเขียนเป็นขั้นตอนการทำงาน จากนั้นจึงเขียนโปรแกรมตามขั้นตอนการทำงานนั้นๆ ส่วนแผนภูมิเวลาจะเหมาะกับระบบที่ทำงานขึ้นอยู่กับเวลามากกว่า ควรออกแบบแผนภูมิการทำงานก่อนที่จะเริ่มต้นเขียนโปรแกรม

ในขณะที่โฟลว์ชาร์ตและสเตทไดอะแกรมจะช่วยให้ผู้ออกแบบเข้าใจกระบวนการทำงานของระบบก่อนที่จะลงมือเขียนโปรแกรม นอกจากนั้นยังช่วยสร้างแลดเดอร์ให้สามารถแยกออกเป็นส่วนๆ ทำให้ง่ายต่อการเพิ่มเติมและแก้ไข โฟลว์ชาร์ตจะเหมาะกับโปรเซสที่มีเส้นทางการไหลของงานเพียงทางเดียวและมีซีเควนซ์การทำงานที่ชัดเจน ส่วนสเตทไดอะแกรมจะเหมาะกับโปรเซสที่มีการไหลของงานทางเดียวและมีโหมดการทำงานที่ชัดเจน

ภาคผนวก

Micro810

Rockwell Automation

Micro810 เป็นคอนโทรลเลอร์ราคาประหยัด เหมาะสำหรับผู้สร้างเครื่องจักรและผู้ใช้งานทั่วไปที่ต้องการควบคุมงานขนาดเล็ก

- เอาต์พุตรีเลย์ทนกระแสไฟสูงถึง 8 A ไม่ต้องใช้รีเลย์ภายนอก
- มีอนาล็อกอินพุต 4 จุด ขนาด 10 บิต 0-10V
- สามารถเพิ่มอุปกรณ์เสริมหน้าจอแสดงผลและเขียนโปรแกรมได้โดยไม่ต้องใช้ซอฟท์แวร์
- มี Real-Time Clock ในตัว
- เขียนโปรแกรมตามมาตรฐาน IEC61131-3 โดยใช้ซอฟท์แวร์ CCW

❑ ข้อมูลการสั่งซื้อ

ชื่อรุ่น	อินพุตดิจิตอล		เอาต์พุตดิจิตอล (Relay)	อินพุตอนาล็อก 0-10V* (ใช้ร่วมกับอินพุตดิจิตอล)	แหล่งจ่ายไฟ
	24VDC	240VAC			
2080-LC10-12QWB	8		4	4	24VDC
2080-LC10-12AWA		8	4	-	240VAC

หมายเหตุ *ถ้าตั้งค่าใช้งานอินพุตอนาล็อก 4 จุด จะเหลืออินพุตดิจิตอล 4 จุด

❑ อุปกรณ์เสริม

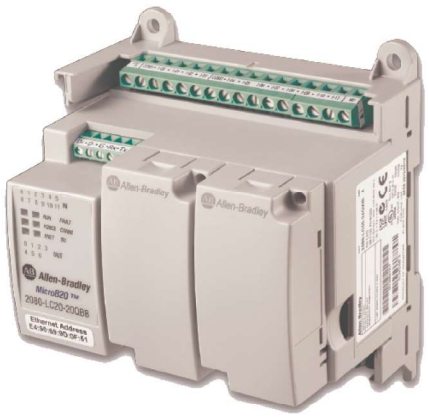
รูปร่าง	ชื่อรุ่น	รายละเอียด
	2080-USBADAPTER	อะแดปเตอร์สำหรับเชื่อมต่อกับพอร์ต USB ของคอมพิวเตอร์
	2080-LCD	จอแสดงผลขนาด 1.5 นิ้ว พร้อมคีย์แพดและหน่วยความจำสำหรับ Backup

❑ คุณสมบัติทางเทคนิค

รายการคุณสมบัติ	รายละเอียด
ขนาดหน่วยความจำ	2 KB
หน่วยความจำข้อมูล	4 KB
ภาษา IEC 61131-3	Ladder Diagram (LD), Function Block (FBD), Structured Text (ST)
Floating Point Math	32-bit และ 64-bit
PID Loop Control	รองรับ
ย่านอุณหภูมิใช้งาน	0°…55°C

Micro820

Rockwell Automation



Micro820 เหมาะสำหรับผู้สร้างเครื่องจักรและผู้ใช้งานทั่วไปที่ต้องการควบคุมเครื่องขนาดเล็กที่มาพร้อมพอร์ต Ethernet และ Serial

- มีพอร์ต EtherNet/IP ในตัว สามารถเชื่อมต่อกับซอฟต์แวร์เขียนโปรแกรมและ HMI ได้ รองรับ Modbus TCP
- มี Real Time Clock ในตัวแต่ไม่ต้องใช้แบตเตอรี่
- สามารถเสียบการ์ด Micro SD ได้ เพื่อถ่ายโอนโปรแกรม เก็บข้อมูลและสูตร

❑ ข้อมูลการสั่งซื้อ

ชื่อรุ่น	อินพุต		เอาต์พุต			แหล่งจ่ายไฟ
	ดิจิตอล (24VDC/VAC)	อนาล็อก 0-10V[1] (ใช้ร่วมกับอินพุตดิจิตอล)	รีเลย์	ดิจิตอล 24VDC (Source)	อนาล็อก 0-10V	
2080-LC20-20QWB(R)	12	4	7		1	24VDC
2080-LC20-20QBB(R)[2]	12	4		7	1	24VDC

หมายเหตุ *ถ้าตั้งค่าใช้งานอินพุตอนาล็อก 4 จุด จะเหลืออินพุตดิจิตอล 8 จุด, ถ้าต้องการใช้รุ่นที่ถอดเทอร์มินอลได้เวลาสั่งซื้อให้ระบุ R ต่อท้ายชื่อรุ่น

❑ อุปกรณ์เสริม

● โมดูล Plug-in

รูปร่าง	ชื่อรุ่น	รายละเอียด
	2080-IQ4	อินพุตดิจิตอล 4 จุด, 12/24VDC (Sink/Source, Type3)
	2080-OB4	เอาต์พุตดิจิตอล 4 จุด, 12/24VDC (Source)
	2080-OV4	เอาต์พุตดิจิตอล 4 จุด, 12/24VDC (Sink)
	2080-OW4I	เอาต์พุตรีเลย์ 4 จุด, (2A ต่อจุด)
	2080-IQ4OB4	อินพุตดิจิตอล 4 จุด, 12/24VDC (Sink/Source,Type3) และ เอาต์พุตดิจิตอล 4 จุด SRC
	2080-IQ4OV4	อินพุตดิจิตอล 4 จุด, 12/24VDC (Sink/Source, Type3) และ เอาต์พุตดิจิตอล 4 จุด, 12/24VDC (Sink)
	2080-IF2	อินพุตอนาล็อก 2 จุด, 0-20 mA, 0-10V, (non-isolated 12-bit)
	2080-IF4	อินพุตอนาล็อก 4 จุด, 0-20 mA, 0-10V, (non-isolated 12-bit)
	2080-OF2	เอาต์พุตอนาล็อก 2 จุด, 0-20 mA, 0-10V,(non-isolated 12-bit)
	2080-SERIALISOL	พอร์ตอนุกรม RS232/485 (isolated)
	2080-TRIMPOT6	อินพุตอนาล็อกสำหรับต่อตัวต้านทานปรับค่าได้(Trimpot) 6 จุด
	2080-RTD2	RTD 2 จุด, (non-isolated, ±1.0 °C)
	2080-TC2	TC 2 จุด, (non-isolated, ±1.0 °C)
	2080-MOT-HSC	High Speed Counter, 250kHz, Differential Line Receiver, เอาต์พุตดิจิตอล 1 จุด
	2080-DNET20	DeviceNet Scanner, 20 Nodes

● แหล่งจ่ายไฟและจอแสดงผล

รูปร่าง	ชื่อรุ่น	รายละเอียด
	2080-PS120-240VAC	แหล่งจ่ายไฟภายนอก 120/240V
	2080-REMLCD	จอแสดงผล LCD ขนาด 3.5นิ้ว ใช้กับแหล่งจ่ายไฟ 24VDC แสดงผลตัวอักษร(ASCII) ได้ 4 หรือ 8 บรรทัด ย่านอุณหภูมิใช้งาน 0°...50°C

❑ คุณสมบัติทางเทคนิค

รายการคุณสมบัติ	รายละเอียด
จำนวนอินพุต/เอาต์พุตดิจิตอล	12/7 (อินพุตอนาล็อก 4 จุด ใช้งานร่วมกับอินพุตปกติ)
จำนวนอินพุต/เอาต์พุตอนาล็อก	4/1
ขนาดหน่วยความจำ	10 Ksteps
หน่วยความจำข้อมูล	20 Kbytes
ขนาด MicroSD	สูงสุด 32 GB (Class 6)
พอร์ตสำหรับเขียนโปรแกรม	พอร์ต Ethernet (ใช้ซอฟต์แวร์ CCW)
ภาษา IEC 61131-3	Ladder Diagram (LD), Function Block (FBD), Structured Text (ST)
พอร์ตสื่อสาร Ethernet	EtherNet/IP Class 3, Modbus TCP
พอร์ตสื่อสาร Serial	RS232/485 non-isolated, CIP Serial, Modbus RTU, ASCII
เอาต์พุต PWM	5 KHz
คำสั่ง Motion	ไม่รองรับ PTO
Floating Point Math	32-bit และ 64-bit
PID Loop Control	รองรับ
จำนวนโมดูล Plug-in	2 (สูงสุด)
ย่านอุณหภูมิใช้งาน	-20°...65°C
แหล่งจ่ายไฟ	24 VDC

Micro850

Rockwell Automation

เหมาะสำหรับงานที่ต้องการความยืดหยุ่นในการออกแบบ สามารถต่อขยายเพิ่มอินพุตเอาต์พุตให้เหมาะสมกับความต้องการได้

- มีอินพุต/เอาต์พุตให้เลือกใช้งาน 24 และ 48 จุด
- มีพอร์ต EtherNet/IP ที่เชื่อมต่อกับซอฟต์แวร์ CCW และ HMI ได้ รองรับ Modbus TCP
- รุ่น 48 I/O ขยายอินพุตเอาต์พุตได้ถึง 132 จุด
- สามารถเพิ่มโมดูลขยาย I/O ได้ถึง 4 ตัว
- มีพอร์ต USB สำหรับเขียนโปรแกรม และพอร์ต Serial ที่รองรับ Modbus RTU

❑ ข้อมูลการสั่งซื้อ

ชื่อรุ่น	จำนวนอินพุต	จำนวนเอาต์พุต			Motion Axis #	HSC*
	(24VDC/VAC)	รีเลย์	24VDC Sink	24VDC Source		
2080-LC50-24QWB	14	10				4
2080-LC50-48QWB	28	20				6
2080-LC50-24QBB	14			10	2 PTO	4
2080-LC50-48QBB	28			20	3 PTO	6
2080-LC50-24QVB	14		10		2 PTO	4
2080-LC50-48QVB	28		20		3 PTO	6

หมายเหตุ: # PTO หนึ่งแกนจะต้องใช้ร่วมกับ HSC จำนวน 2 ชุด ถ้าใช้ PTO ทั้งหมดจำนวนของ HSC จะเหลือเป็นศูนย์

* จำนวน HSC ที่แสดงเป็นรุ่น 2 สาย ถ้าต้องการใช้กับ HSC รุ่น 4 สาย จำนวนจะลดลงครึ่งหนึ่ง

❑ อุปกรณ์เสริม

● แหล่งจ่ายไฟภายนอก

รูปร่าง	ชื่อรุ่น	รายละเอียด
	2080-PS120-240VAC	แหล่งจ่ายไฟภายนอก 120/240V

❑ อุปกรณ์เสริม

● โมดูล Plug-in

รูปร่าง	ชื่อรุ่น	รายละเอียด
	2080-IQ4	อินพุตดิจิตอล 4 จุด, 12/24VDC (Sink/Source, Type3)
	2080-OB4	เอาต์พุตดิจิตอล 4 จุด, 12/24VDC (Source)
	2080-OV4	เอาต์พุตดิจิตอล 4 จุด, 12/24VDC (Sink)
	2080-OW4I	เอาต์พุตรีเลย์ 4 จุด, (2A ต่อจุด)
	2080-IQ4OB4	อินพุตดิจิตอล 4 จุด, 12/24VDC (Sink/Source,Type3) และ เอาต์พุตดิจิตอล 4 จุด(SRC)
	2080-IQ4OV4	อินพุตดิจิตอล 4 จุด, 12/24VDC (Sink/Source, Type3) และ เอาต์พุตดิจิตอล 4 จุด, 12/24VDC (Sink)
	2080-IF2	อินพุตอนาล็อก 2 จุด, 0-20 mA, 0-10V, (non-isolated 12-bit)
	2080-IF4	อินพุตอนาล็อก 4 จุด, 0-20 mA, 0-10V, (non-isolated 12-bit)
	2080-OF2	เอาต์พุตอนาล็อก 2 จุด, 0-20 mA, 0-10V,(non-isolated 12-bit)
	2080-SERIALISOL	พอร์ตอนุกรม RS232/485 (isolated)
	2080-TRIMPOT6	อินพุตอนาล็อกสำหรับต่อตัวต้านทานปรับค่าได้(Trimpot) 6 จุด
	2080-RTD2	RTD 2 จุด, (non-isolated, ±1.0 °C)
	2080-TC2	TC 2 จุด, (non-isolated, ±1.0 °C)
	2080-MEMBAK-RTC	Memory Backup และ High Accuracy RTC
	2080-MOT-HSC	High Speed Counter, 250kHz, Differential Line Receiver, เอาต์พุตดิจิตอล 1 จุด
	2080-DNET20	DeviceNet Scanner, 20 Nodes

● โมดูลขยาย I/O

รูปร่าง	ชื่อรุ่น	รายละเอียด
	2085-IQ16	อินพุตดิจิตอล 16 จุด, 12/24VDC, Sink/Source
	2085-IQ32T	อินพุตดิจิตอล 32 จุด, 12/24VDC, Sink/Source
	2085-OV16	เอาต์พุตดิจิตอล 16 จุด, 12/24VDC, Sink
	2085-OB16	เอาต์พุตดิจิตอล 16 จุด, 12/24VDC, Source
	2085-OW8, 2085-OW16	เอาต์พุตรีเลย์รุ่น 8 จุด และรุ่น 16 จุด, (2A ต่อจุด)
	2085-IM8	อินพุต 8 จุด, 240 VAC
	2085-OA8	เอาต์พุต 8 จุด, 120/240 VAC
	2085-IF4, 2085-IF8	อินพุตอนาล็อกรุ่น 4 จุด และรุ่น 8 จุด, 0 ~ 20mA, -10V ~ +10V isolated,14บิต
	2085-OF4	เอาต์พุตอนาล็อก 4 จุด, 0 ~ 20mA, -10V ~ +10V, isolated, 12 บิต
	2085-IRT4	RTD และ TC 4 จุด, isolated, ±0.5 °C
	2085-ECR	End Cap Terminator

❑ คุณสมบัติทางเทคนิค

รายการคุณสมบัติ	รุ่น 24 I/O	รุ่น 48 I/O
จำนวนอินพุต/เอาต์พุต	14/10	28/20
จำนวนดิจิตอล I/O สูงสุด	132	
ขนาดหน่วยความจำ	10 Ksteps	
หน่วยความจำข้อมูล	20 Kbytes	
ภาษา IEC 61131-3	Ladder Diagram (LD), Function Block (FBD), Structured Text (ST)	
พอร์ตสำหรับเขียนโปรแกรม	USB 2.0 (ใช้ซอฟต์แวร์ CCW)	
พอร์ตสื่อสาร Ethernet	EtherNet/IP Class 3, Modbus TCP (10/100Mbps)	
พอร์ตสื่อสาร Serial	RS232/485 non-isolated (Modbus RTU Master/Slave, ASCII, CIP)	
HSC (100Kz)	4	6
Floating Point Math	32-bit และ 64-bit	
PID Loop Control/คำสั่ง Motion	รองรับ	
จำนวนโมดูล Plug-in	3	5
จำนวนโมดูลขยาย I/O	สูงสุด 4 โมดูล	
ย่านอุณหภูมิใช้งาน	-20°…65°C	
แหล่งจ่ายไฟ	24 VDC	

SONIC AUTOMATION

บริษัท โซนิค ออโตเมชั่น จำกัด
99/9 อาคารสำนักงานเซ็นทรัลแจ้งวัฒนะ ชั้นที่ 20
ห้องเลขที่ 2003 และ 2004 หมู่ 2 ถนนแจ้งวัฒนะ
ตำบลบางตลาด อำเภอปากเกร็ด จังหวัดนนทบุรี 11120
Fax: +66 (0) 2835 3935 www.sonicautomation.co.th

+66 (0) 2835 3933