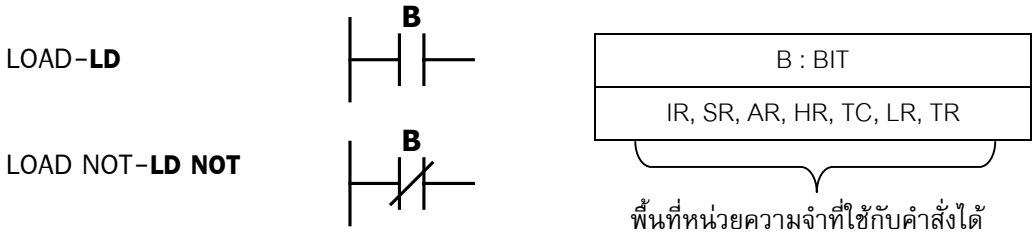
บทที่ 4
หลักการเขียน Ladder Diagram และคำสั่งพื้นฐาน

Ladder Diagram จัดเป็นภาษาสัญลักษณ์ที่สามารถดูตามโครงสร้างแล้วเข้าใจการทำงาน แต่เวลาที่ PLC ทำงานจะอาศัยชุดคำสั่ง (Instructions) ทำงานโดยวิธีการเขียนลงในส่วนหน่วยความจำ ข้อมูลในหน่วยความจำนั้น จะจัดเก็บเป็นรหัส (Code) ไม่สามารถจัดเก็บในลักษณะของ Ladder Diagram ได้โดยตรง ดังนั้นผู้ใช้จึงจำเป็นต้องเข้าใจชุดคำสั่ง เพราะชุดคำสั่งนั้น แปลงภาษามาจาก Ladder Diagram นั่นเอง

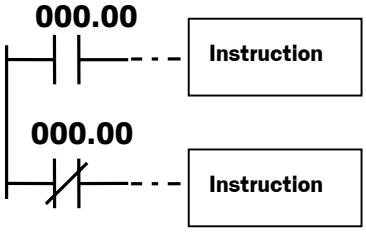
4.1 กลุ่มคำสั่งพื้นฐาน(Ladder Instruction & Output Control)

4.1.1 การใช้คำสั่ง Load (LD), Load Not (LD NOT)



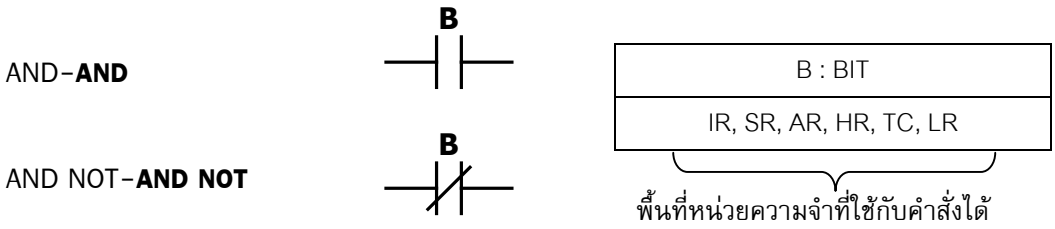
ตัวอย่างที่ 4.1

ชุดคำสั่งและการเขียน Ladder Diagram คำสั่ง LD และ LD NOT



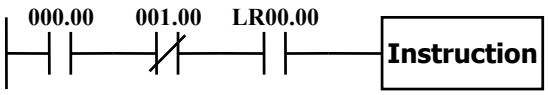
Address	Instruction	Operands
00000	LD	00000
00001	Instruction	
00002	LD NOT	00000
00003	Instruction	

4.1.2 การใช้คำสั่ง AND, AND NOT



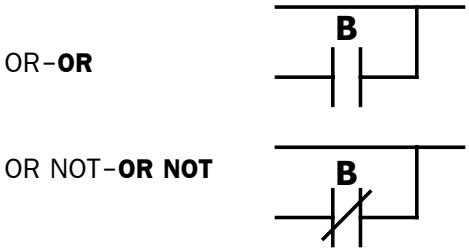
ตัวอย่างที่ 4.2

ชุดคำสั่งและการเขียน Ladder Diagram คำสั่ง AND, AND NOT



Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00100
00002	AND	LR 00000
00003	Instruction	

4.1.3 การใช้คำสั่ง OR, OR NOT

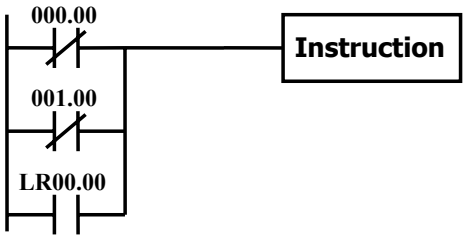


B : BIT
IR, SR, AR, HR, TC, LR

พื้นที่หน่วยความจำที่ใช้คำสั่งได้

ตัวอย่างที่ 4.3

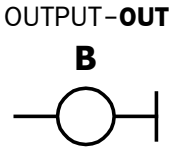
ชุดคำสั่งและการเขียน Ladder Diagram คำสั่ง OR, OR NOT



Address	Instruction	Operands
00000	LD NOT	00000
00001	OR NOT	00100
00002	OR	LR 00000
00003	Instruction	

4.1.4 การใช้คำสั่ง OUT, OUT NOT

เป็นคำสั่งที่สั่งขับให้ OUTPUT ภายนอกทำงานหรือไม่ทำงานตามคำสั่ง

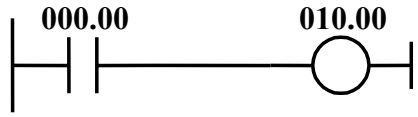


B : BIT
IR, SR, AR, HR, LR, TR

พื้นที่หน่วยความจำที่ใช้กับคำสั่งได้

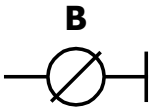
ตัวอย่างที่ 4.4

รูปแบบชุดคำสั่งจาก Ladder Diagram



Address	Instruction	Operands
00000	LD	00000
00001	OUT	01000

OUTPUT NOT–**OUT NOT:** การทำงานของคำสั่งนี้จะตรงกันข้ามกับคำสั่ง OUT

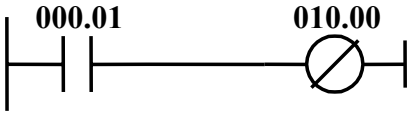


B : BIT
IR, SR, AR, HR, LR, TR

พื้นที่หน่วยความจำที่ใช้กับคำสั่งได้

ตัวอย่างที่ 4.5

จงเขียนชุดคำสั่งจาก Ladder Diagram



Address	Instruction	Operands
00000	LD	00001
00001	OUT NOT	01000

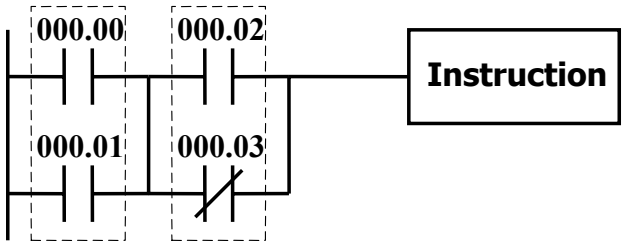
4.1.5 การใช้คำสั่ง END(FUN 01)

การเขียนโปรแกรมทุกครั้ง เมื่อสิ้นสุดการเขียนโปรแกรมแล้วจะต้องจบด้วยคำสั่ง END (01) เสมอ ถ้าไม่มีคำสั่งนี้ เมื่อผู้ใช้สั่ง Run โปรแกรมที่เขียนขึ้น PLC จะเกิด Error โดยสังเกตที่ PLC ไฟ Error/Alarm สีแดงจะติดค้าง และจะมีข้อความ “NO END INST” ปรากฏอยู่ที่หน้าจอ LCD นั่นหมายถึงว่าไม่มีคำสั่ง END (01)

ในกรณีนี้โปรแกรมจะไม่สามารถ RUN ได้ เพราะฉะนั้นเมื่อเขียนโปรแกรมจบทุกครั้งควรใส่คำสั่ง END (01) ด้วย

ตัวอย่างที่ 4.6

ทดลองป้อนโปรแกรมให้กับ PLCโดยไม่มีคำสั่ง END (01)



Address	Instruction	Operands
00000	LD	00000
00001	OR	00001
00002	LD	00002
00003	OR NOT	00003
00004	AND LD	---

หมายเหตุ การเรียกคำสั่งบางคำสั่งด้วยการกำหนดจากหมายเลขฟังก์ชัน Function (FUN)

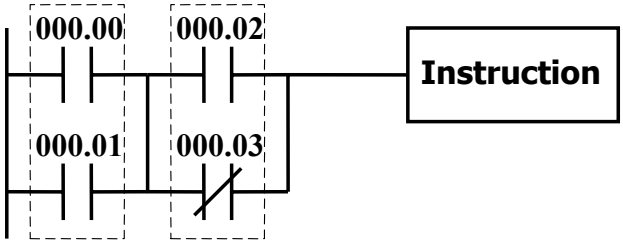
4.1.6 การใช้คำสั่ง AND LOAD (AND LD), OR LOAD (OR LD)

คำสั่งทั้งสองจะทำหน้าที่เชื่อมต่อกลุ่ม Ladder Diagram ในกรณีที่ต่ออนุกรม หรือขนานกันมากกว่า 1 หน้าสัมผัส ซึ่งการใช้คำสั่ง AND หรือ OR นั้น จะกระทำทีละ 1 หน้าสัมผัสเท่านั้น จึงต้องใช้ AND LD หรือ OR LD

ในการเขียน Ladder Diagram นั้น ไม่มีสัญลักษณ์ของ AND LD และ OR LD

ตัวอย่างที่ 4.7

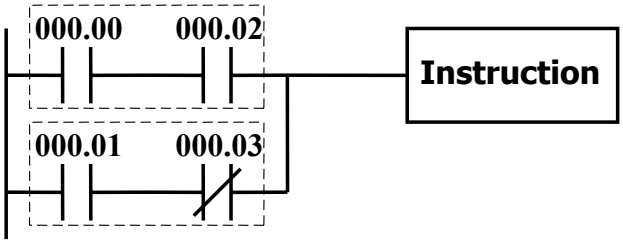
ชุดคำสั่งในรูปการเชื่อมแบบอนุกรมจะใช้คำสั่ง AND LD



Address	Instruction	Operands
00000	LD	00000
00001	OR	00001
00002	LD	00002
00003	OR NOT	00003
00004	AND LD	

ตัวอย่างที่ 4.8

ชุดคำสั่งในรูปการเชื่อมแบบขนานจะใช้คำสั่ง OR LD

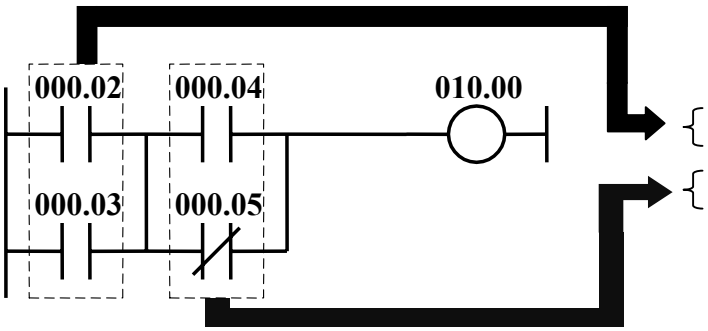


Address	Instruction	Operands
00000	LD	00000
00001	AND	00002
00002	LD	00001
00003	AND NOT	00003
00004	OR LD	

ตัวอย่างที่ 4.9

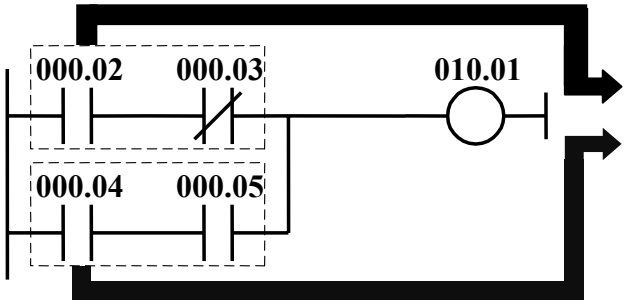
การเขียนโปรแกรมโดยใช้คำสั่ง AND LD และ OR LD

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



Address	Instruction	Data
00000	LD	00002
00001	OR	00003
00002	LD	00004
00003	OR NOT	00005
00004	AND LD	- - -
00005	OUT	01000

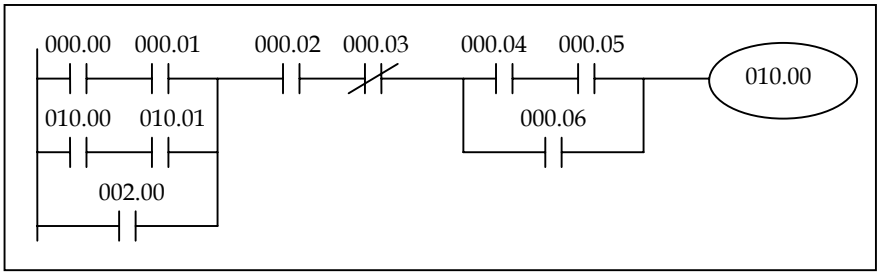
- **OR LD** คือ การเชื่อมโปรแกรม 2 block ในแบบขนาน



Address	Instruction	Data
00000	LD	00002
00001	AND NOT	00003
00002	LD	00004
00003	AND	00005
00004	OR LD	- - -
00005	OUT	01001

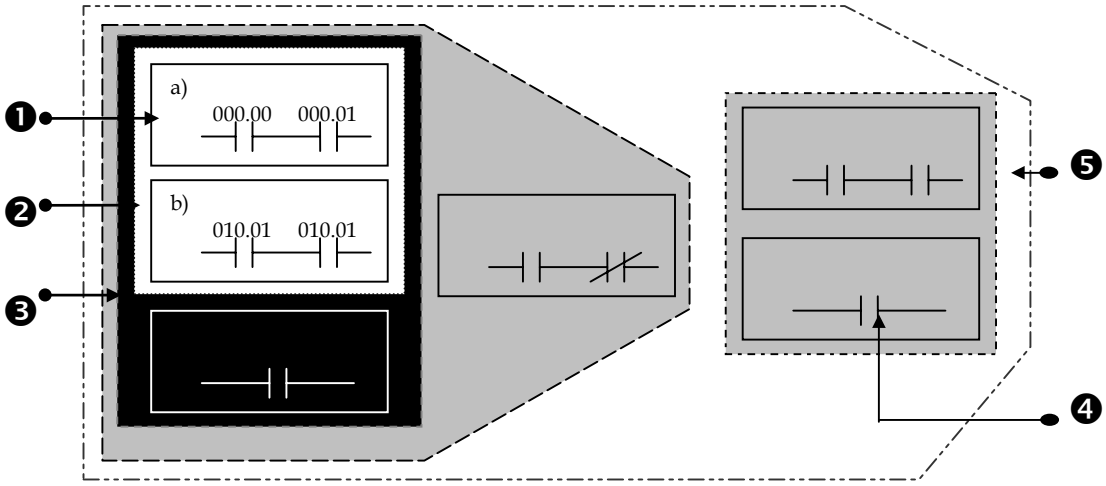
ตัวอย่างที่ 4.10

การแปลง Ladder Diagram เป็นภาษา Instruction List (Mnemonic Code)

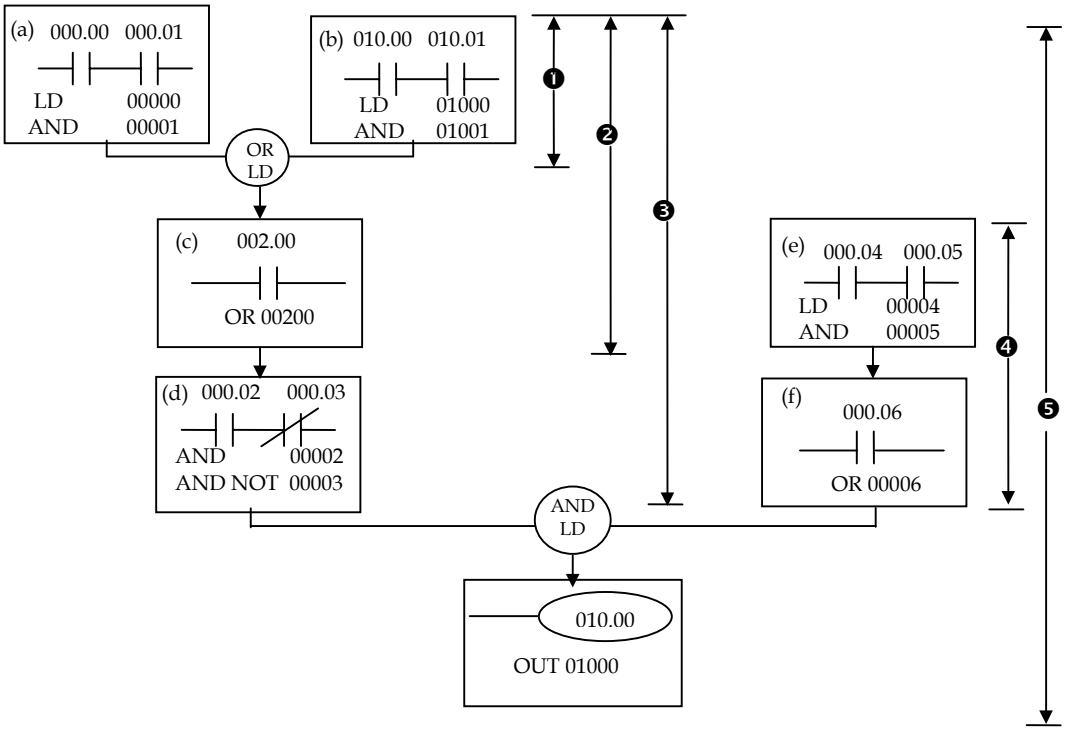


จากรูปข้างบน เราสามารถแบ่งเป็น Block ได้ดังนี้

1) แยกวงจรเป็น block เล็ก ๆ [a] ถึง [f]

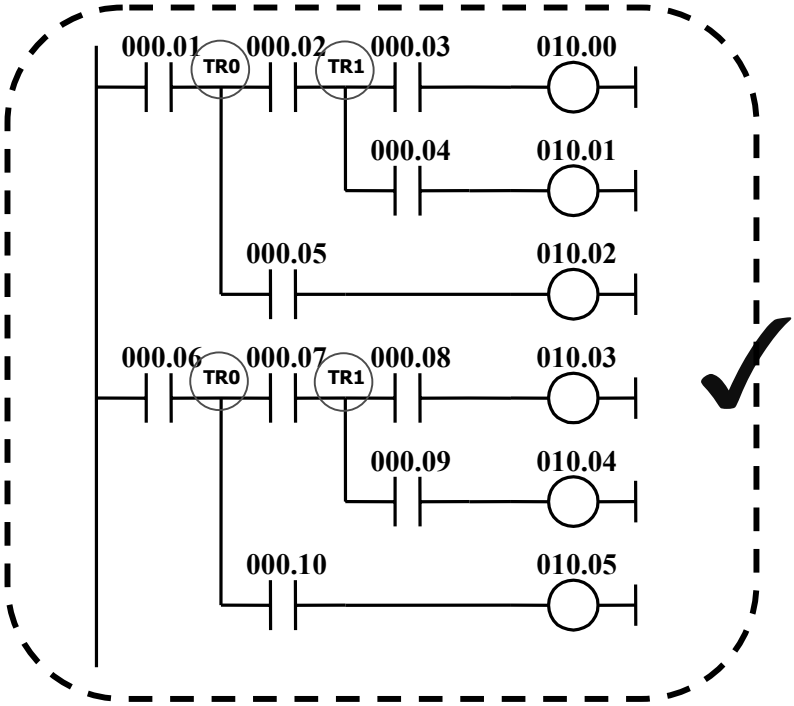


2) ป้อนโปรแกรมแต่ละ block จากบนสู่ล่าง ต่อจากนั้นจากซ้ายไปขวา

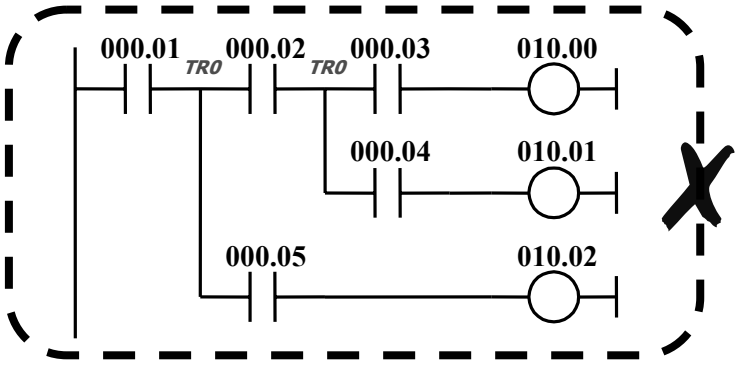


4.2 การใช้คำสั่ง TR (Temporary Relay)

คำสั่งนี้ใช้สำหรับการเขียน Ladder Diagram ที่มีการขับคอยล์เอาต์พุต (OUT Coil) อยู่หลายๆ สาขาโดยที่สาขาหนึ่งๆ ประกอบไปด้วยคำสั่ง TR นี้ โดยที่ Ladder Diagram สามารถแยกสาขาได้ถึง 8 สาขาย่อย (TR0-TR7)



Ladder Diagram ที่สามารถใช้งานได้



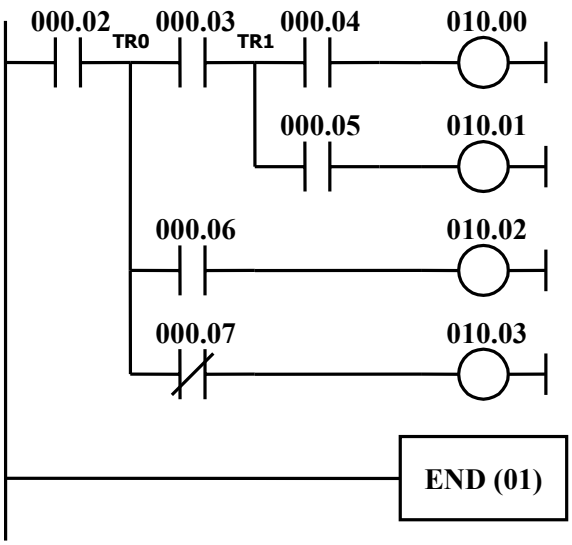
Ladder Diagram ชุดนี้ที่ใช้งานไม่ได้เพราะใช้ TR0 ซ้ำ

หมายเหตุ TR จะถูกใช้ก็ต่อเมื่อ Ladder Diagram มีการแยกสาขาย่อย

ตัวอย่างที่ 4.11

ลักษณะ Ladder Diagram ที่ต้องการใช้ TR มาช่วย

Ladder Diagram



ชุดคำสั่ง

Address	Instruction List	Operand
00000	LD	00002
00001	OUT	TR0
00002	AND	00003
00003	OUT	TR1
00004	AND	00004
00005	OUT	01000
00006	LD	TR1
00007	AND	00005
00008	OUT	01001
00009	LD	TR0
00010	AND	00006
00011	OUT	01002
00012	LD	TR0
00013	ANDNOT	00007
00014	OUT	01003
00015	END(01)	

4.3 คำสั่งที่สามารถเรียกใช้เมื่อกดปุ่ม FUN

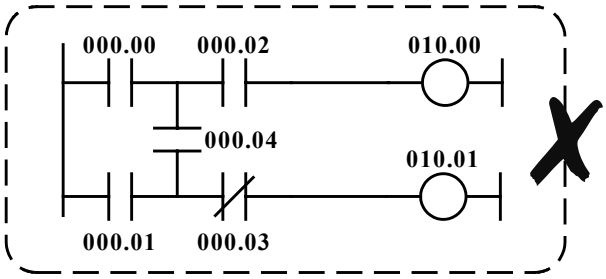
" FUN xx" เมื่อ xx คือตัวเลขที่บอกถึงคำสั่งต่างๆ ภายในของ PLC เช่น

FUN 01	หมายถึงคำสั่ง	END (End Instruction)
FUN 02	,,	IL (Interlock)
FUN 03	,,	ILC (Interlock Clear)
FUN 04	,,	JMP (Jump End)
FUN 05	,,	JME (Jump End)
FUN 10	,,	SFT (Shift Register)
FUN 11	,,	KEEP (Latching Relay)
FUN 12	,,	CNTR (Reversible Counter)
FUN 13	,,	DIFU (Differentation – Up)
FUN 14	,,	DIFD (Differentation –Down)

การเรียกคำสั่งใดออกมาใช้งาน ถ้าเป็นการเขียนโปรแกรมด้วย Programming Console จำเป็นจะต้องกดปุ่ม **FUN** แล้วตามด้วยหมายเลขของคำสั่งนั้นๆ จึงจะเป็นการเรียกคำสั่งนั้นออกมาใช้งาน

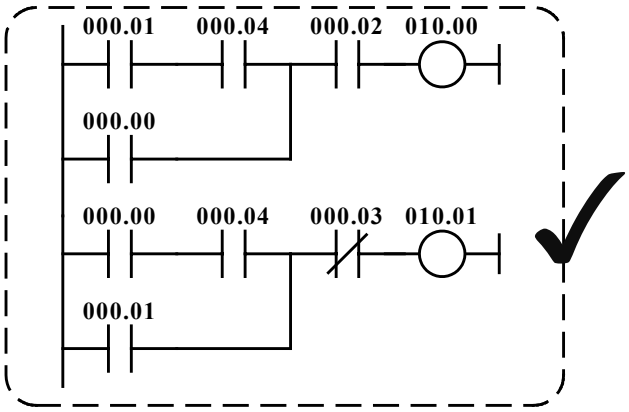
4.4 ข้อกำหนดในการเขียน Ladder Diagram

4.4.1 จาก Ladder Diagram ข้างล่าง จะไม่สามารถเขียนโปรแกรมได้ จำเป็นต้องแปลงชุด Ladder Diagram ก่อน



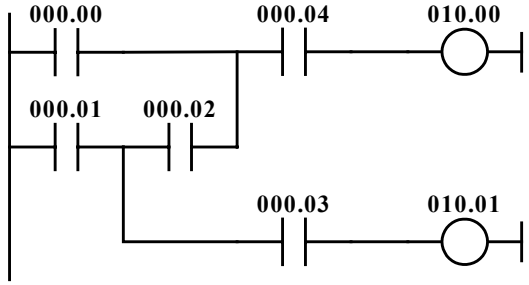
Ladder Diagram ที่ผิด

สามารถเขียนใหม่ได้ และวงจรทำงานเหมือนเดิม คือ



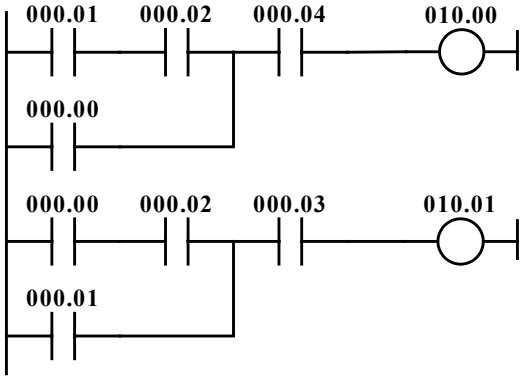
Ladder Diagram ที่ถูก

4.4.2 สำหรับ Ladder Diagram จะพิจารณาการทำงานจากซ้ายไปขวาเท่านั้น ดังตัวอย่างเช่น



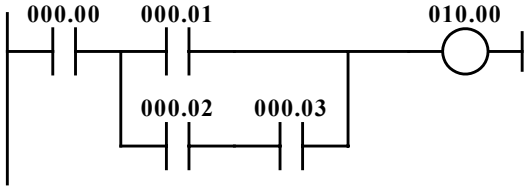
Ladder Diagram A

จาก Ladder Diagram A ถ้าหน้าสัมผัส 000.00, 000.02 และ 000.03 มีสภาวะ “ON” ก็ไม่สามารถทำให้ เอาต์พุต 010.01 นั้น “ON” ได้เลย ดังนั้นผู้ใช้จะต้องทำการจัดโปรแกรมเสียใหม่เพื่อให้การทำงาน กระทำจากซ้ายไปขวา ดังรูป Ladder Diagram B



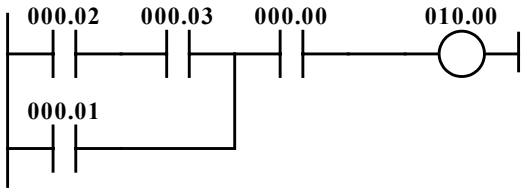
Ladder Diagram B

4.4.3 จำนวนหน้าคอนแทคทั้ง NO และ NC ของ อินพุต/เอาต์พุต,รีเลย์ และไทม์เมอร์ (TIM)/เคาน์เตอร์ (CNT) จะมีการโหลดเพื่อนำมาเขียนโปรแกรมเป็นจำนวนเท่าใดก็ได้ตามความประสงค์ของผู้ใช้ แต่ถึงอย่างไรก็ตามการเขียนโปรแกรมที่ดีจะต้องพยายามประหยัดขนาดของโปรแกรมให้มากเท่าที่จะสามารถทำได้ ซึ่งจะเปรียบเทียบให้เห็นใน Ladder Diagram A และ Ladder Diagram B จะสังเกตเห็นได้ว่าการเขียนใน Ladder Diagram B จะประหยัดคำสั่งได้ 2 คำสั่ง ในขณะที่โปรแกรมทำงานได้เหมือนกัน



Ladder Diagram A

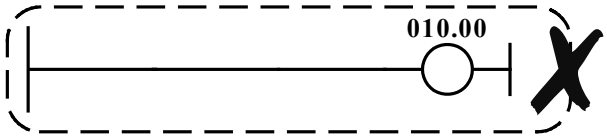
Address	Instruction	Operands
00000	LD	00000
00001	LD	00001
00002	LD	00002
00003	AND	00003
00004	OR LD	
00005	AND LD	
00006	OUT	01000



Ladder Diagram B

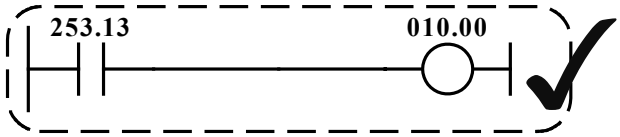
Address	Instruction	Operands
00000	LD	00002
00001	AND	00003
00002	OR	00001
00003	AND	00000
00004	OUT	01000

4.4.4 เมื่อต้องการให้เอาต์พุต ON ตลอดเวลาเราจะใช้แฟลค (Flag) ที่เป็นแบบ “ALWAYS ON” (253.13) ใน SR area มาเป็นตัวสร้างเงื่อนไขเพราะไม่สามารถต่อคอยล์เอาต์พุตได้โดยตรงกับ Bus Bar แต่ก็มีข้อยกเว้นเป็นบางคำสั่ง เช่น INTERLOCK CLEAR, JUMP END และ STEP



Ladder Diagram ที่ผิด

```
OUT     01000
END(01)
```



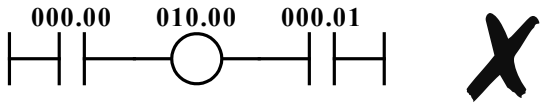
Ladder Diagram ที่ถูก

```
LD      25313
OUT     01000
END(01)
```

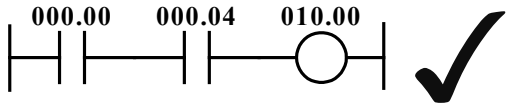
4.4.5 จำนวนหน้าสัมผัสที่ใช้ในการต่ออนุกรม หรือขนานไม่มีขีดจำกัดจะใช้เท่าใดก็ได้ ขึ้นอยู่กับความต้องการของผู้ใช้

4.4.6 เอาต์พุตทุกๆ ตัวจะมี Auxiliary Contact เพื่อใช้งานในโปรแกรมได้ และสามารถใช้จำนวนไม่จำกัด

4.4.7 ไม่สามารถเขียนโปรแกรมให้หน้าสัมผัสอยู่ตำแหน่งหลังจากคอยล์ได้

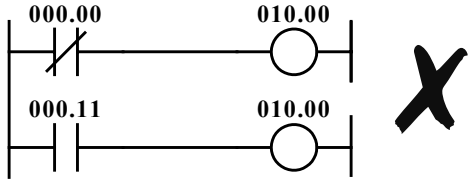


Ladder Diagram ที่ผิด

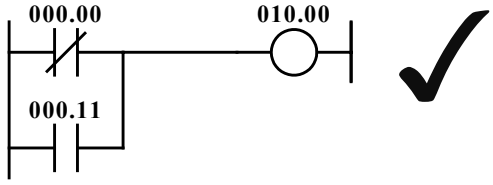


Ladder Diagram ที่ถูก

4.4.8 ไม่สามารถเขียนโปรแกรมให้มีเอาต์พุตเบอร์เดียวกันซ้ำหลายๆ ครั้งได้ ต้องจัดรูปเสียใหม่

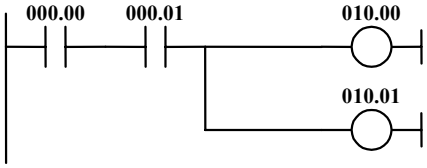


Ladder Diagram ที่ผิด



Ladder Diagram ที่ถูก

4.4.9 เอาต์พุตคอยล์ สามารถเขียนโปรแกรมให้ต่อขนานได้เลย กรณีรับเงื่อนไขของหน้าสัมผัสชุดเดียวกัน



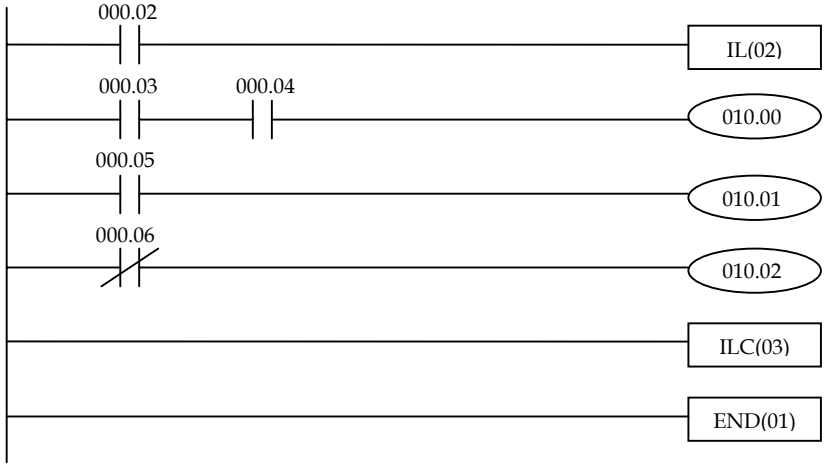
4.4.10 PLC จะเริ่มประมวลผลโปรแกรมจาก Address 000.00 จนกระทั่งถึงคำสั่ง END ตำแหน่งแรก โดยที่ คำสั่ง END อาจจะมีหลายตำแหน่งในโปรแกรมที่เป็นเช่นนี้เพื่อจุดประสงค์สำหรับการทดสอบโปรแกรม กรณีแยกโปรแกรมออกเป็นส่วนๆ และง่ายต่อการตรวจสอบแก้ไขโปรแกรม

4.5 กลุ่มคำสั่ง Program Control Instruction

4.5.1 การใช้คำสั่ง IL(02), ILC(03)

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

ตัวอย่างที่ 4.1 การใช้คำสั่ง

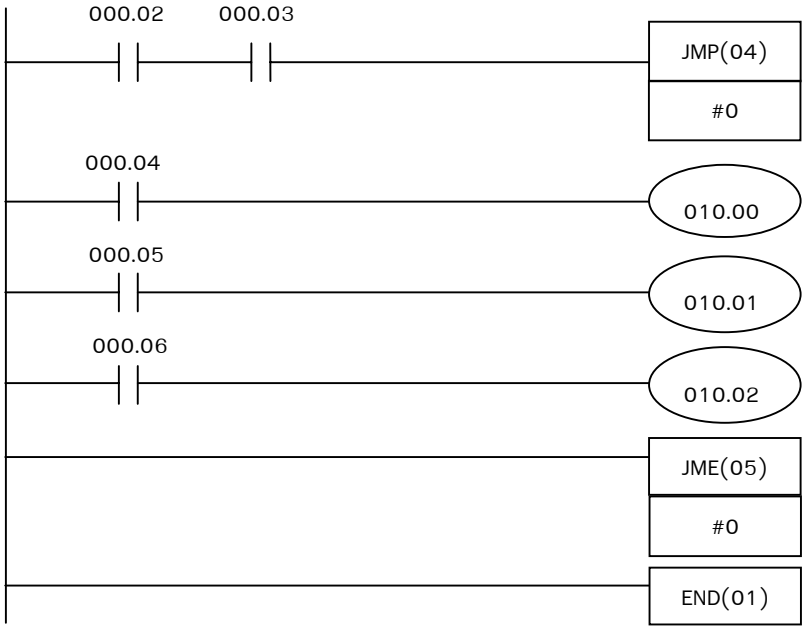


Address	Instruction	Data
00000	LD	00002
00001	IL (02)	-
00002	LD	00003
00003	AND	00004
00004	OUT	01000
00005	LD	00005
00006	OUT	01001
00007	LD NOT	00006
00008	OUT	01002
00009	ILC(03)	-
00010	END(01)	-

4.5.2 การใช้คำสั่ง JMP (04) และ JME (05)

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

Ladder Diagram

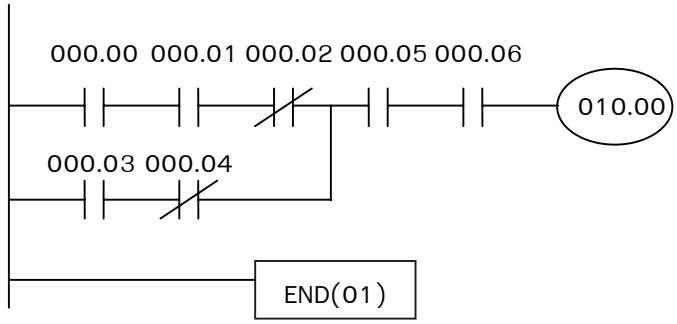


ชุดคำสั่ง

Address	Instruction	Operand
0000	LD	00002
0001	AND	00003
0002	JMP(04)	#0
0003	LD	00004
0004	OUT	01000
0005	LD	00005
0006	OUT	01001
0007	LD	00006
0008	OUT	01002
0009	JME(05)	#0
0010	END(01)	

แบบฝึกหัดทดสอบความเข้าใจเกี่ยวกับ Basic

1. จงเขียนโปรแกรมจาก Ladder Diagramให้เป็นรูป Mnemonic Code

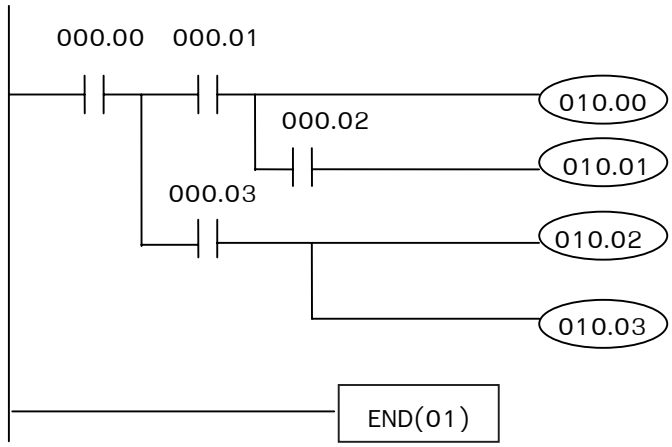


Address	*Instruction*	*Operands*
00000		
00001		
0002		
00003		
00004		
00005		
00006		
00007		
00008		
00009		

2. จงเขียนโปรแกรมจาก Mnemonic ให้เป็นรูปLadder Diagram

Address	*Instruction*	*Operands*
00000	LD	00000
00001	AND	00001
00002	LD NOT	00002
00003	AND	00003
00004	OR LD	
00005	LD	00004
00006	AND NOT	00005
00007	LD NOT	00006
00008	AND	00007
00009	OR LD	
00010	AND LD	
00011	OUT	01000
00012	END (01)	

3. จงเขียนโปรแกรมจาก Ladder Diagram ให้เป็นรูป Mnemonic Code

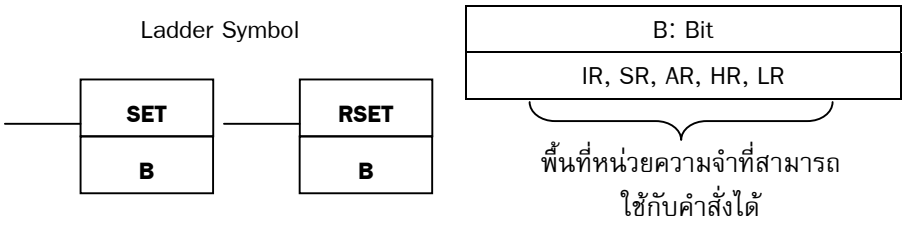


Address	*Instruction*	*Operands*
00000		
00001		
00002		
00003		
00004		
00005		
00006		
00007		
00008		
00009		
00010		

4.6 คำสั่งในกลุ่ม Bit Control Instruction

4.6.1 การใช้คำสั่งเซต(SET) และรีเซต (RESET)

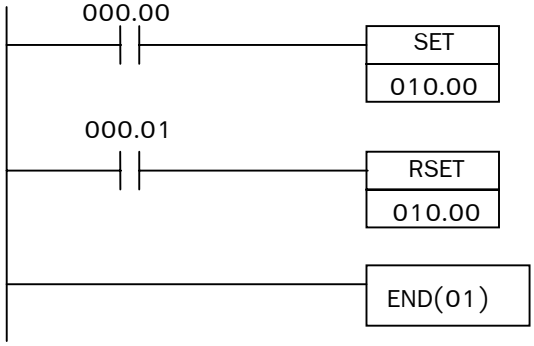
คำสั่ง SET เมื่อ ON แล้วบิตที่ถูกสั่งจะยังคงค้างอยู่จนกว่าคำสั่ง RSET ที่บิตเดียวกัน บิตนั้นจึงจะ “OFF”



ตัวอย่างที่ 4.2

ต้องการให้หลอดไฟที่เอาต์พุต 010.00 ติดตลอดเวลา หลังจาก ON ที่อินพุต 000.00 แล้วครั้งเดียวโดยไม่ต้อง Hold คำสั่งเอาต์พุต จนกว่าจะมีการ Reset อินพุต 000.01 หรือไฟดับเท่านั้น

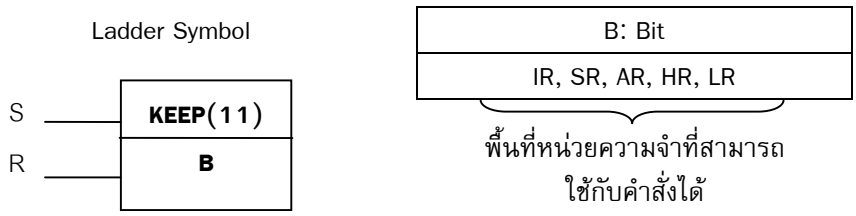
Ladder Diagram **ชุดคำสั่ง**



00000	LD	000.00
00001	SET	010.00
00002	LD	000.01
00003	RSET	010.00
00004	END	

4.6.2 การใช้คำสั่ง KEEP – KEEP(11)

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

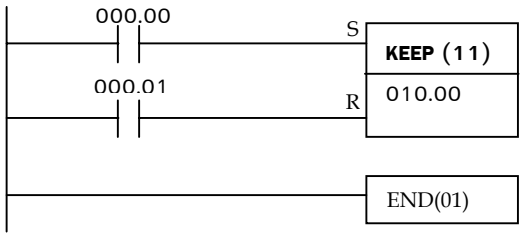


เมื่อขา S ถูกตั้ง ON บิตที่ B จะทำงานจนกว่าขา R จะถูกตั้ง ON บิต B ถึงจะเลิกทำงาน

ตัวอย่างที่ 4.3

ต้องการให้เอาต์พุต 010.00 ON ตลอดเวลาโดยการ ON อินพุต 000.00 ไม่ว่าจะ OFF แล้วก็ตาม จนกว่าอินพุต 000.01 จะ ON (RESET)

Ladder Diagram



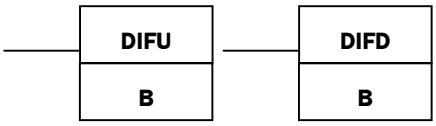
ชุดคำสั่ง

00000	LD	000.00
00001	LD	000.01
00002	KEEP	010.00
00003	END	

4.6.3 การใช้คำสั่ง DIFFERENTIATE UP and DOWN–DIFU(13), DIFD(14)

คำสั่งนี้ DIFU(13) และ DIFD(14) จะเป็นคำสั่งที่ทำงานเพียงขอบขาขึ้น หรือขอบขาลงของอินพุตเท่านั้น และจะทำงานเพียงช่วง One Cycle Time เท่านั้น

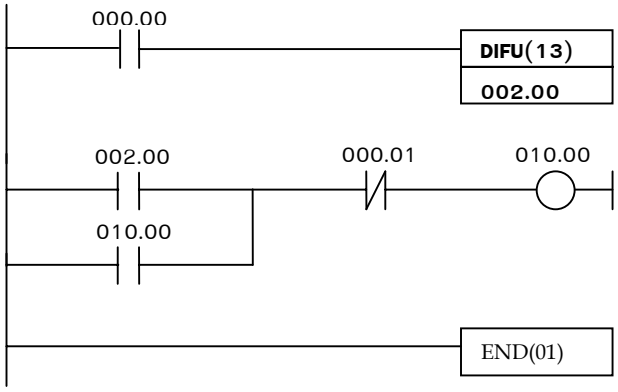
Ladder Symbols



ตัวอย่างที่ 4.4

ต้องการให้อินพุต 000.00 ที่มีความไวในการ ON-OFF สามารถ ON Output Lamp 010.00 ให้ติดได้โดยอินพุต 000.01 เป็นตัวสั่ง OFF

Ladder Diagram



ชุดคำสั่ง

00000	LD	000.00
00001	DIFU	002.00
00002	LD	002.00
00003	OR	010.00
00004	AND NOT	000.01
00005	OUT	010.00
00006	END	

4.7 กลุ่มคำสั่ง Timer/Counter

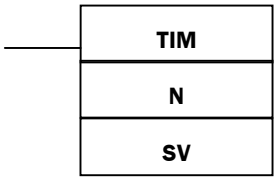
สำหรับ PLC บางรุ่น Timer และ Counter จะใช้พื้นที่เดียวกันซึ่งเรียกใช้ได้ทั้งหมด 256 ตัว ตั้งแต่ตัวที่ 000 ถึง 255 ภายใน 256 ตัวนี้สามารถกำหนดให้เป็น Timer หรือ Counter ก็ได้โดยที่หากตัวใดถูกกำหนดให้เป็น Timer แล้วจะนำไปใช้เป็น Counter อีกไม่ได้ ดังนั้นต้องดู Manual ของ PLC รุ่นนั้นประกอบด้วย ถ้า Timer/Counter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

แต่ PLC บางรุ่น Timer/Counter จะอยู่คนละพื้นที่ ดังนั้นจึงสามารถใช้ Timer และ Counter เบอร์เดียวกันได้

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

4.7.1 การใช้คำสั่ง TIMER: TIM

ใช้ในการจับเวลา,ตั้งเวลา โดยพื้นฐานแล้วต้องเข้าไปกำหนดค่า 2 ค่าคือ N และ SV ตามตัวอย่างข้างล่าง



N: TC Number (หมายเลขของ Timer/Counter)
000-255

SV: Set Value (ค่าตั้งเวลา)
ค่าคงที่ (#) หรือการอ้างถึง IR, SR, AR, DM, HR, LR

N = Timer Number (เบอร์ 000 - 255) เลือกว่าจะใช้ Timer ตัวที่เท่าใด

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

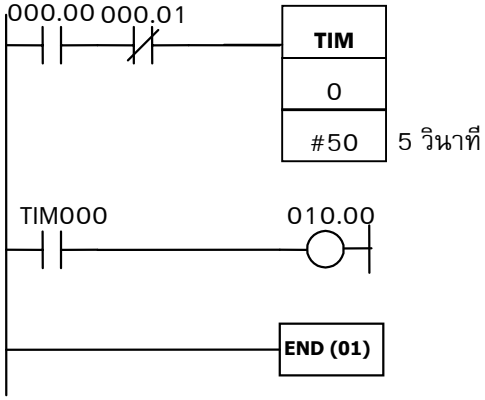
1. กำหนด SV เป็นค่าคงที่ #0000-9999 (000.0-999.9 วินาที คูณด้วย 0.1 วินาที)
2. กำหนด SV เป็น แอดเดรส IR, SR, AR, DM, HR, LR โดยใส่ค่าตั้งเวลาที่เป็นค่าคงที่ 0000-9999 ไว้ใน แอดเดรส ที่อ้างถึงอีกทีหนึ่ง (ค่าที่กำหนดจะคูณด้วย 0.1 วินาทีเช่นเดียวกับการกำหนดแบบค่าคงที่)

หมายเหตุ *ในที่นี้ยกตัวอย่างหมายเลข Timer ของ PLC รุ่น CPM2A เท่านั้นสำหรับ PLC รุ่นอื่นๆ สามารถใช้ Timer ได้มากกว่าที่กำหนด

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

ตัวอย่างที่ 4.5

การใช้งานของคำสั่ง Timer เมื่อ อินพุต 000.00 ON ไปได้ 5 Sec. เอาต์พุต 010.00 จะ ON และ เอาต์พุต 010.01 จะ OFF



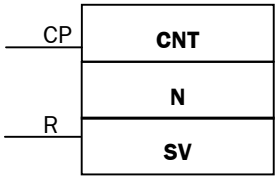
a) Ladder Diagram

Address	Instruction	Operands
00000	LD	00000
00001	AND NOT	00001
00002	TIM 000	#050
00003	LD	TIM000
00004	OUT	01000
00005	LD NOT	TIM000
00006	OUT	01001
00007	END (01)	

b) ชุดคำสั่ง

4.7.2 การใช้คำสั่ง COUNTER – CNT

เป็นคำสั่งที่ใช้นับจำนวนครั้งของสัญญาณ อินพุต ที่ ON แต่ละครั้ง ซึ่งเป็นคำสั่งที่นับลงจากค่าที่ตั้งไว้ (Set Value)



N: TC Number (หมายเลขของ Timer/Counter)
000–255

SV: Set Value (ค่าตั้งจำนวนนับ)
ค่าคงที่(#) หรือการอ้างถึง IR, SR, AR, DM, HR, LR

N = Counter Number(เบอร์ 000 - 255) เลือกว่าจะใช้ Counter ตัวที่เท่าใด

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

1. กำหนด SV เป็นค่าคงที่ #0000-9999
2. กำหนด SV เป็น แอดเดรส IR, SR, AR, DM, HR, LR โดยใส่ค่าตั้งจำนวนนับที่เป็นค่าคงที่ 0000-9999 ไว้ใน แอดเดรส ที่อ้างถึงอีกทีหนึ่ง

CP = ขานับ เมื่อมีสัญญาณอินพุตในช่วงที่เปลี่ยนสถานะจาก OFF เป็น ON เข้ามาที่ขานี้ Counter จะนับถอยหลังลง 1

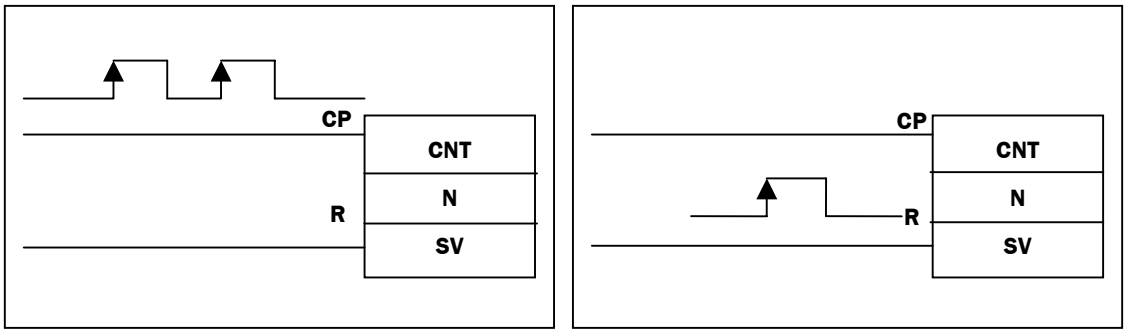
R = ขา Reset เมื่อมีสัญญาณอินพุตเข้ามาที่ขานี้ เอาต์พุตของ Counter จะหยุดทำงานและค่านับของ Counter จะถูก Reset กลับไปเท่ากับค่าตั้งจำนวนนับ (SV)

หมายเหตุ *ในที่นี้ยกตัวอย่างหมายเลข Counter ของ PLC รุ่น CPM2A เท่านั้นสำหรับ PLC รุ่นอื่นๆ สามารถใช้ Counter ได้มากกว่าที่กำหนด

*Memory Area ของ Timer และ Counter จะใช้พื้นที่เดียวกันจึงจะใช้คำสั่งทั้ง Timer และ Counter กับพื้นที่เดียวกันไม่ได้

ตัวอย่างที่ 4.6

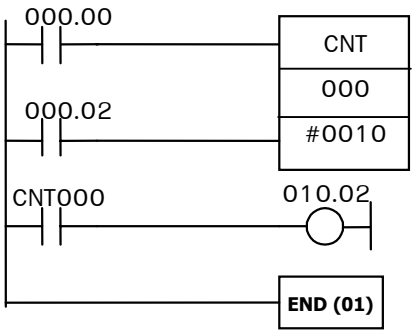
เราใช้คำสั่ง Timer หมายเลข 000 แล้ว จะใช้คำสั่ง Counter ที่หมายเลข 000 อีกไม่ได้ ต้องใช้หมายเลขอื่นๆ



จังหวะการนับ จังหวะการรีเซต

ตัวอย่างที่ 4.7

การใช้งานของคำสั่ง Counter เมื่อ อินพุต 000.00 ON 1 ครั้ง Counter จะนับ 1 ครั้ง ถ้า อินพุต 000.00 ON ครบ 10 ครั้ง จะทำให้คำสั่ง Counter ทำงานพร้อมกับ Contact ของ Counter (CNT000) จะทำงานด้วย และจะถูก Reset ด้วยอินพุต 000.02



a) Ladder Diagram

Address	Instruction	Operands
00000	LD	00000
00001	LD	00002
00002	CNT 001	#0010
00003	LD	CNT001
00004	OUT	01002
00005	END (01)	

b) ชุดคำสั่ง

✪ การประยุกต์ใช้งานของ TIMER และ COUNTER

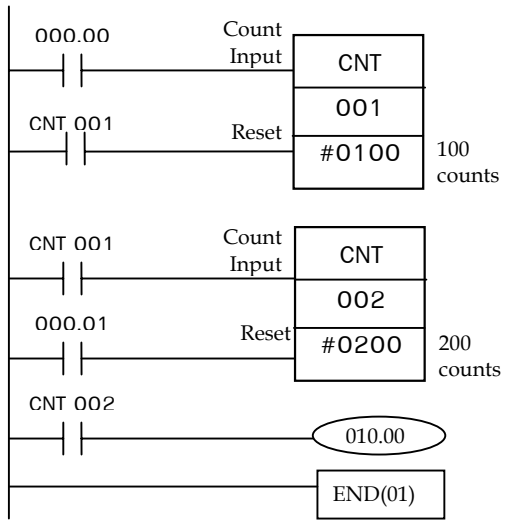
ตัวอย่างที่ 4.8

ต้องการนับจำนวนคนดูคอนเสิร์ต สถานที่จัดสามารถจุคนดูได้ 20,000 คน ใช้ Photo Switch นับจำนวนคน หลังจาก 20,000 คนแล้วให้ OUTPUT LAMP 010.00 ทำงาน เพื่อแสดงว่าคนเต็มแล้ว

I/O ที่กำหนด

PHOTO SWITCH	000.00
PB RESET	000.01
OUTPUT LAMP	010.00

Ladder Diagram



ชุดคำสั่ง

Address	Instruction	Data
00000	LD	00000
00001	LD	CNT 001
00002	CNT	001
		#0100
00003	LD	CNT 001
00004	LD	00001
00005	CNT	002
		#0200
00006	LD	CNT 002
00007	OUT	01000
00010	END (01)	

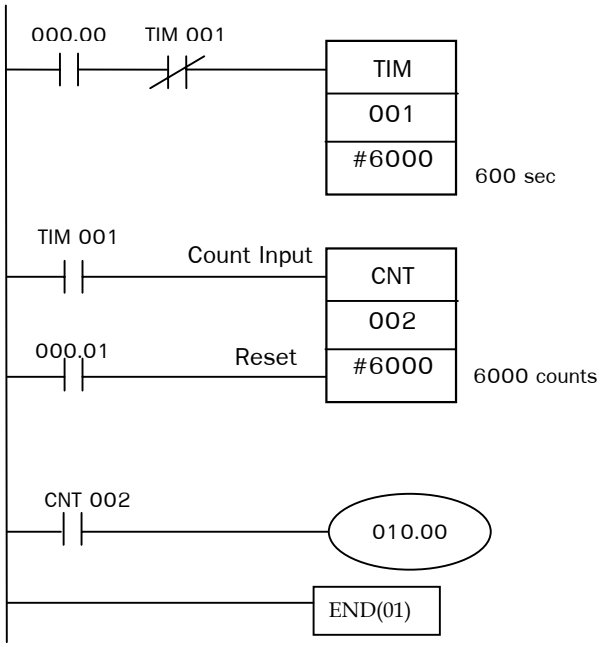
ตัวอย่างที่ 4.9

เครื่องจักรเครื่องหนึ่งต้องการอัดจารบีหลังจากใช้งานไปแล้ว 1,000 ชั่วโมง

I/O ที่กำหนด

PB START	000.00
PB RESET	000.01
VALUE LUBRICATE	01000

Ladder Diagram

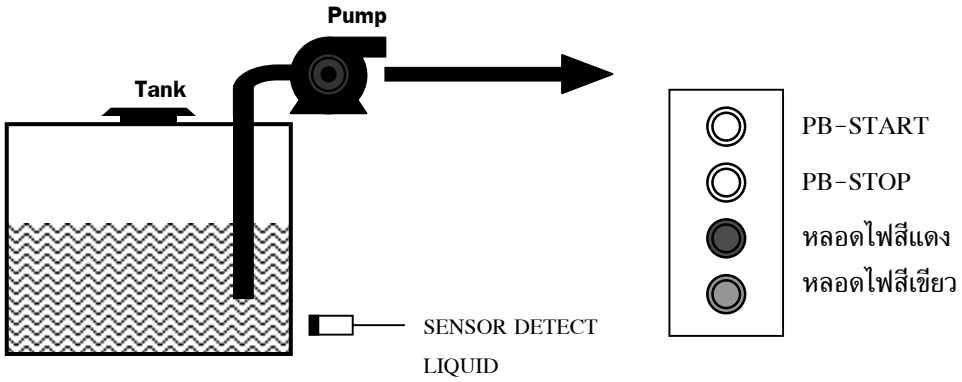


ชุดคำสั่ง

Address	*Instruction*	*Data*
00000	LD	00000
00001	AND-NOT	TIM 001
00002	TIM	001
		# 6000
00003	LD	TIM 001
00004	LD	00001
00005	CNT	002
		# 6000
00006	LD	CNT 002
00007	OUT	01000
00008	END (01)	

ตัวอย่างที่ 4.10

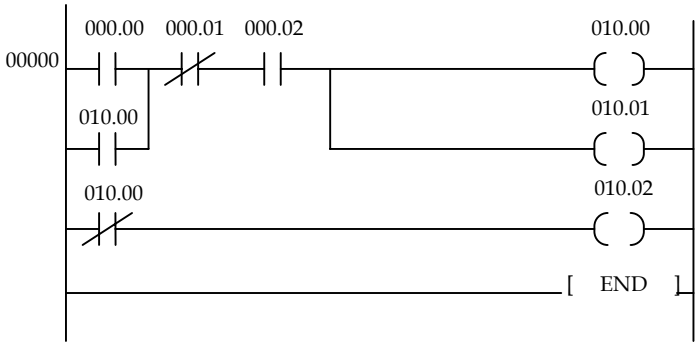
มอเตอร์ปั้มน้ำ ถ้ากดปุ่ม PB-START หลอดไฟสีแดงจะติดเพื่อแสดงว่าปั้มทำงาน และปั๊มน้ำจะหยุดการทำงานพร้อมกับหลอดไฟสีแดงดับ ก็ต่อเมื่อกด PB-STOP หลอดไฟสีเขียวจะติดแทน (ปั๊มจะทำงานได้ต้องมีน้ำในแทงค์เท่านั้น)



I/O ที่กำหนด

INPUT	00000	PB-START
	00001	PB-STOP
	00002	SENSOR DETECT LIQUID
OUTPUT	01000	MOTOR PUMP
	01001	หลอดไฟสีแดง
	01002	หลอดไฟสีเขียว

Ladder Diagram



ชุดคำสั่ง

Address	Instruction	Operand
00000	LD	00000
00001	OR	01000
00002	AND NOT	00001
00003	AND	00002
00004	OUT	01000
00005	OUT	01001
00006	LD NOT	01000
00007	OUT	01002
00008	END (01)	

แบบฝึกหัด

จงเขียนโปรแกรมตั้งเวลา 24 ชั่วโมง โดยเมื่อครบตามเวลาที่กำหนดต้องการให้มอเตอร์ทำงาน

Ladder Diagram

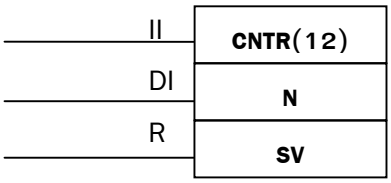
Mnemonic Code

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

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

ตัวนับ CNTR ใช้อินพุต 3 อินพุตควบคุมการทำงานคือ II (Increment Input), DI (Decrement Input) และ Reset Input (R) การใช้งานต้องระบุเบอร์ของตัวนับ และกำหนดค่าการนับ (Set Value) เป็นจำนวนเท่าใดด้วย

Ladder Symbol



N: TC Number(หมายเลขของ Timer/Counter)
000-255
SV: Set Value (word, BCD)
IR, SR, AR, DM, HR, LR, #

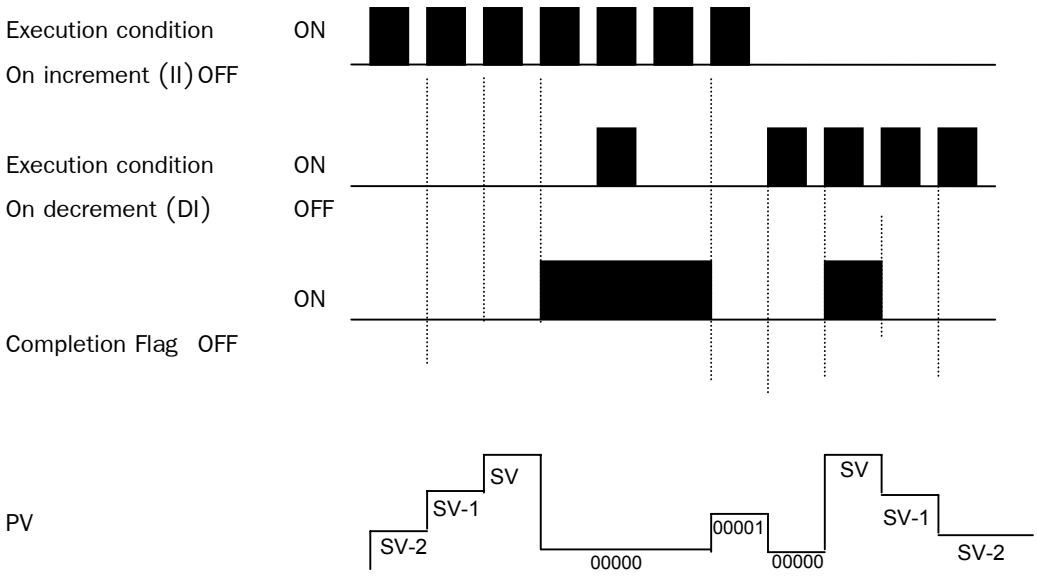
พื้นที่หน่วยความจำที่ใช้กับคำสั่งได้

II-คือขานับสัญญาณแบบนับขึ้น

DI-คือขานับสัญญาณแบบนับลง

R-คือขา Reset

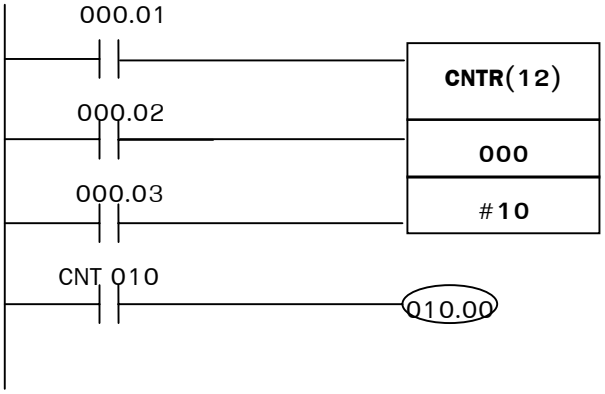
การกำหนดค่า SV (ค่าที่จะตั้งให้นับ) สามารถกำหนดเป็นค่าคงที่เลย หรือกำหนดผ่านอุปกรณ์ต่างๆ ที่กำหนดไว้ให้ แต่ถึงอย่างไรต้องตั้งอยู่ในย่าน 1-9,999 ลักษณะการทำงาน CNT เขียนได้ดังนี้



ตัวอย่างที่ 4.12

การใช้งาน Counter ชนิด UP/DOWN counter

Ladder Diagram



ชุดคำสั่ง

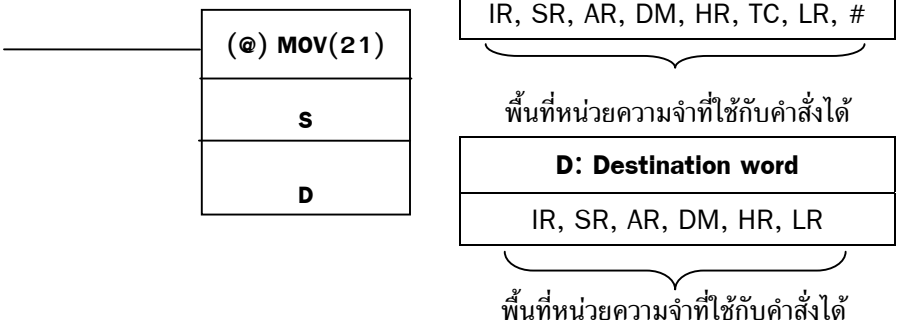
Address	Operand	Data	
00000	LD		0001
00001	LD		0002
00002	LD		0003
00003	CNTR(12)		010
00004		#	10
00005	LD	CNT	010
00006	OUT		1000

Reversible Counter หรือเรียกอีกชื่อว่า UP-DOWN Counter ทั้งนี้เพราะสามารถทำการนับขึ้น ในกรณีที่มีสัญญาณเข้าที่ II และมีการนับลงเมื่อมีสัญญาณเข้าที่ DI แต่ถ้ามีสัญญาณพร้อมกันทั้งที่ UP และ DOWN Input จะไม่ทำให้เกิดการนับในทิศทางใดๆ การนับจะนับเป็นลักษณะวนหรือต่อเนื่องกันไปเรื่อยๆ กล่าวคือ เมื่อนับครบตามจำนวนที่กำหนดไว้แล้ว จะวนกลับมาเป็น “0” หรือ “10” ใหม่ เป็นเช่นนี้ไปเรื่อย และการแสดงค่า Output และ CNTR จะมีสภาวะ “ON” เพียง 1 ครั้งเท่านั้น ในจังหวะที่เป็นการวนค่าเช่นจาก “10” เป็น “0” ในกรณีที่มีการเพิ่มหรือลดถึงค่าที่กำหนดเอาไว้ ส่วนสัญญาณ Reset ถือว่าเป็นอีกอินพุตหนึ่งของ CNTR ถ้ามีสภาวะ “ON” เมื่อใดจะทำให้ค่าจากการนับมีค่าเริ่มต้นที่ “0000” ทันที

4.8 กลุ่มคำสั่ง Data Movement

4.8.1 การใช้คำสั่ง MOVE – MOV(21)

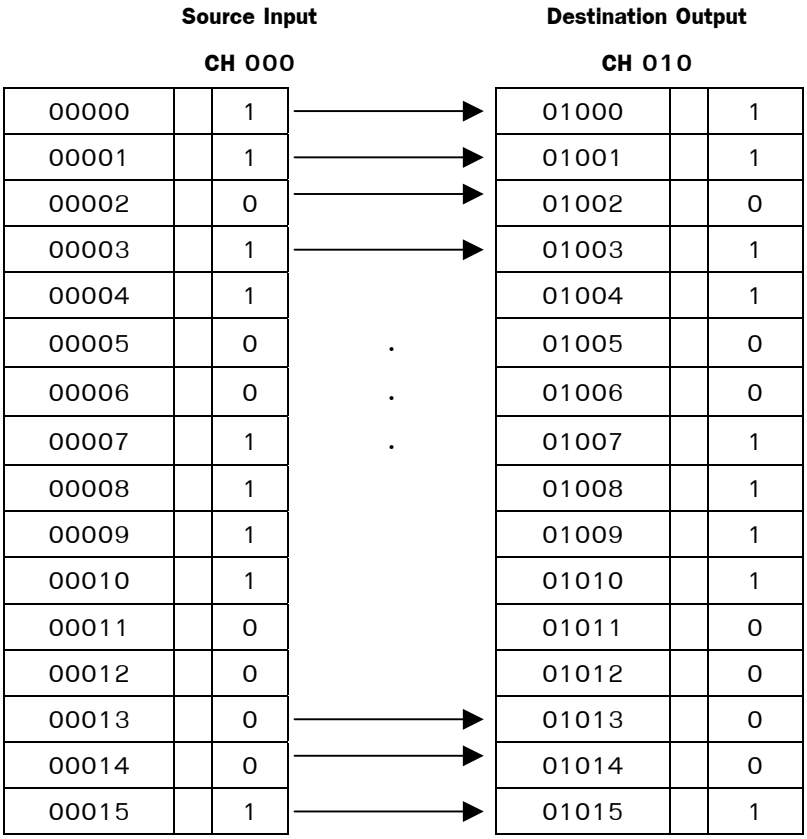
Ladder Symbol



S = เวิร์ดต้นฉบับที่ต้องการ Copy (Source word)

D = เวิร์ดปลายทางที่ Copy (Destination word)

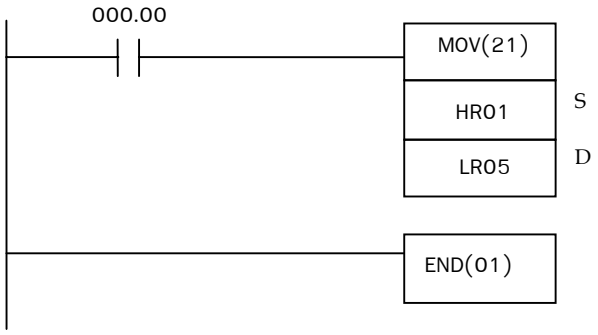
เมื่อคำสั่งนี้ทำงาน คือ MOV (21) จะ Copy ย้ายข้อมูลจาก S ไปยัง D โดยที่ S ยังมีข้อมูลเดิมอยู่ ส่วนสัญลักษณ์ @ หมายถึง คำสั่งจะทำงานเพียงแค่ 1 Scan time เท่านั้น



ตัวอย่างที่ 4.13

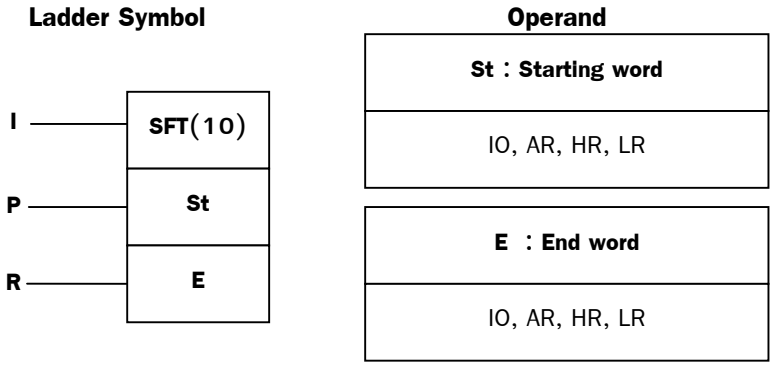
ค่าที่ S = HR 01 (มีค่าข้อมูลคือ 1500) และ D = LR 05 (มีค่าข้อมูลคงที่ 0050) เมื่อคำสั่ง MOV(21) ทำงาน ค่าที่ D จะมีข้อมูลใหม่คือ 1500

Ladder Diagram



4.9 กลุ่มคำสั่ง Data Shifting Instructions

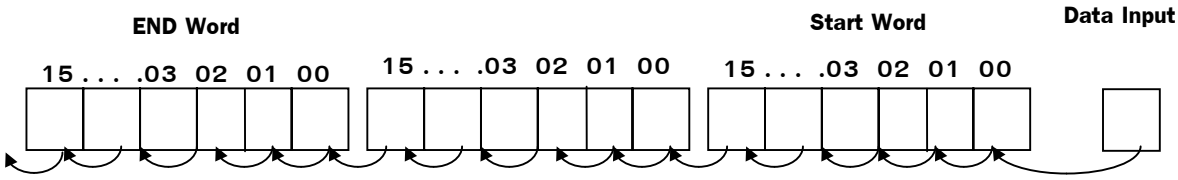
4.9.1 การใช้คำสั่ง SHIFT REGISTER – SFT(10)



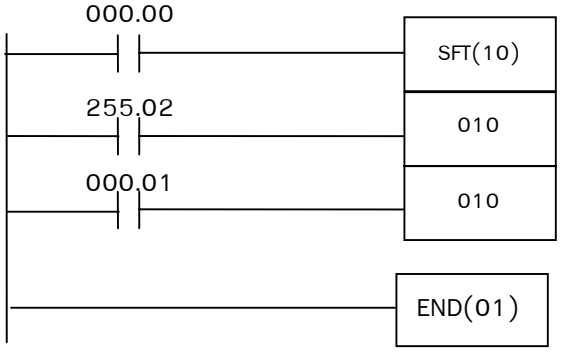
I คือ ขา Data Input กำหนดค่าที่ใช้ในการเลื่อนข้อมูล
P คือขา Pulse Input ใช้ป้อนสัญญาณพัลส์เพื่อเลื่อนข้อมูล
R คือ ขา Reset ใช้กำหนดค่าข้อมูลใน Starting Word ถึง End Word ให้มีค่าเป็น 0

การทำงานจะเลื่อนข้อมูลทีละบิต จากบิต 0 ของ Starting word ไปจนถึงบิต 15 ของ End word ถ้ามีการ ON ที่ขา P แต่ละครั้ง

ตัวอย่างที่ 4.14



ทดลองเขียนคำสั่ง SFT(10) โดยมี 1 Second clock bit (25502) ที่ขา P และมี (000.00) เป็นตัวสั่ง Shift โดย Output word 010 จะ ON ตั้งแต่บิตที่ 010.00 แล้วเลื่อนไป ON ที่ 010.01



Address	Instruction	Operands
00000	LD	00000
00001	LD	25502
00002	LD	00001
00003	SFT(10)	
		010
		010
00004	END(01)	