PLC

# การประยุกต์ใช้โปรแกรม (Ladder Diagram/MELSEC iQ-R ซีรีส์)

หลักสูตรนี้จัดทำขึ้นสำหรับผู้ใช้ ที่เข้าใจพื้นฐาน PLC รุ่น iQ-R และต้องการเรียนรู้ขั้นตอนถัดไปของการเขียนโปรแกรม

L(CTS)00686THA

## บทนำ วัตถุประสงค์ของหลักสูตร

หลักสูตรนี้จัดทำขึ้นสำหรับผู้ใช้ที่ผ่านหลักสูตรพื้นฐานการเขียนโปรแกรม (Ladder Diagram) มาแล้ว หรือผู้ที่มีความรู้เทียบเท่า หลักสูตรนี้จะให้ความรู้เกี่ยวกับการเขียนโปรแกรมและการหาจุดบกพร่องอย่างมีประสิทธิภาพ สำหรับ PLC รุ่น iQ-R การใช้งานอุปกรณ์ชั้นสูง และการเขียนโปรแกรมลาเบล

คุณควรผ่านการฝึกอบรมหลักสูตรต่อไปนี้แล้ว หรือมีความรู้เทียบเท่า ซึ่งเป็นเงื่อนไขที่จำเป็นต้องมีสำหรับหลักสูตรนี้

- ข้อมูลเบื้องต้นเกี่ยวกับ MELSEC iQ-R ซีรีส์
- พื้นฐานการเขียนโปรแกรม

## บทนำ โครงสร้างของหลักสูตร

เนื้อหาของหลักสูตรนี้มีดังนี้

บทที่ 1 - การเขียนโปรแกรมอย่างมีประสิทธิภาพ

วิธีการและการตั้งค่าสำหรับการเขียนโปรแกรมอย่างมีประสิทธิภาพ

บทที่ 2 - การเขียนโปรแกรมขั้นสูง

การใช้อุปกรณ์และการเขียนโปรแกรมลาเบลชั้นสูง

บทที่ 3 - การหาจุดบกพร่องอย่างมีประสิทธิผล

ฟังก์ชันต่างๆ ของซอฟต์แวร์ที่ใช้เพื่อการหาจุดบกพร่องอย่างมีประสิทธิผล

แบบทดสอบประเมินผล

เกณฑ์การผ่าน: ต้องได้คะแนน 60% ขึ้นไป

## บทนำ วิธีการใช้งานเครื่องมือการเรียนรู้อิเล็กทรอนิกส์นี้

| | | |
|---|---|---|
| ไปที่หน้าถัดไป | > | ไปที่หน้าถัดไป |
| กลับไปยังหน้าที่แล้ว | < | กลับไปยังหน้าที่แล้ว |
| เลื่อนไปยังหน้าที่ต้องการ | TOC | ระบบจะแสดง "สารบัญ" ช่วยให้คุณสามารถไปยังหน้าต่างๆ ได้ |
| ออกจากการเรียนรู้ | X | ออกจากการเรียนรู้ |

## ข้อควรระวังด้านความปลอดภัย

เมื่อคุณเรียนรู้โดยการใช้งานผลิตภัณฑ์จริง โปรดอ่านข้อควรระวังด้านความปลอดภัยในคู่มือการใช้งานที่อย่างละเอียด

## ข้อควรระวังในหลักสูตรนี้

หน้าจอที่แสดงของเวอร์ชั่นที่คุณใช้อาจจะแตกต่างจากในหลักสูตรนี้ หลักสูตรนี้มีการใช้ซอฟต์แวร์เวอร์ชันต่อไปนี้:

- GX Works3 เวอร์ชัน 1.044W

# บทที่ 1 การออกแบบโปรแกรมอย่างมีประสิทธิภาพ

บทนี้อธิบายเกี่ยวกับการตั้งค่าซอฟต์แวร์เพื่อการออกแบบโปรแกรมอย่างมีประสิทธิภาพและพื้นฐานการออกแบบโปรแกรมด้วยเลเบิล

1.1 การใช้งานโปรแกรมในระบบต่างๆ ได้ง่ายขึ้น

1.2 การปรับพื้นที่หน่วยความจำตามสถานะการใช้งานอุปกรณ์

1.3 การใช้ชื่อเลเบิลที่สัมพันธ์กับการใช้งาน

1.4 การปรับปรุงความสะดวกในการอ่านโปรแกรม

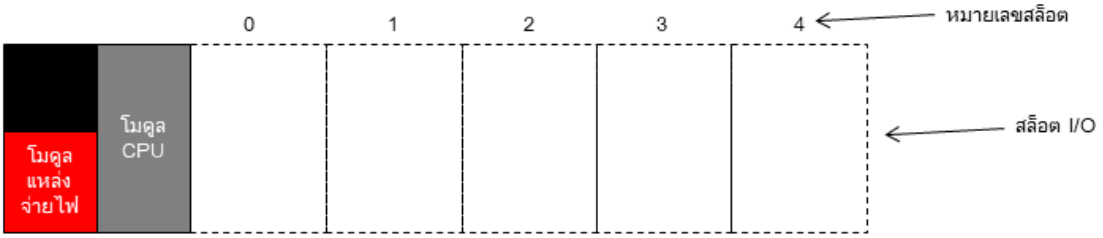
## 1.1 การใช้งานโปรแกรมในระบบต่างๆ ได้ง่ายขึ้น

หมวดนี้จะอธิบายเกี่ยวกับการกำหนดหมายเลข I/O อย่างมีประสิทธิผลเพื่อนำโปรแกรมไปใช้ในระบบต่างๆ ได้อย่างง่ายยิ่งขึ้น

### 1.1.1 การกำหนดหมายเลข I/O อัตโนมัติ

I/O จะถูกกำหนดอย่างเป็นลำดับให้กับโมดูล ที่ถูกติดตั้งบนฐานรองรับโมดูลโดยเริ่มจากสล็อตที่อยู่ใกล้กับโมดูล CPU มากที่สุด จำนวน I/O จะกำหนดเป็น 16 พอยต์ (0 ถึง F)

จำนวนของพอยต์ที่สามารถใช้ (จอง) ได้ จะแตกต่างกันไปตามชนิดของโมดูล (16, 32, 64 เป็นต้น)




ในตัวอย่างต่อไปนี้ มีการติดตั้งโมดูล 16 พอยต์ ห้าชุด

## 1.1.1 การกำหนดหมายเลข I/O อัตโนมัติ

เมื่อใช้โมดูลที่มีพอยต์สำหรับจอง 16, 32 และ 64 พอยต์พร้อมกัน การมอบหมายหมายเลข I/O จะเป็นดังนี้:

| | | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| | | 16 พอยต์ | 32 พอยต์ | 64 พอยต์ | 32 พอยต์ | 16 พอยต์ |
| โมดูลแหล่งจ่ายไฟ | โมดูล CPU | 00<br>ถึง<br>0F | 10<br>ถึง<br>2F | 30<br>ถึง<br>6F | 70<br>ถึง<br>8F | 90<br>ถึง<br>9F |

หากมีสล็อตว่างระหว่างโมดูลที่ติดตั้งไว้ จะมีการกำหนดหมายเลข I/O ให้กับสล็อตว่างเช่นกัน

| | | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| | | 16 พอยต์ | 32 พอยต์ | 64 พอยต์ | 16 พอยต์ | 16 พอยต์ |
| โมดูลแหล่งจ่ายไฟ | โมดูล CPU | 00<br>ถึง<br>0F | 10<br>ถึง<br>2F | 30<br>ถึง<br>6F | 70<br>ถึง<br>7F | 80<br>ถึง<br>8F |

สล็อตว่าง

ตามค่าเริ่มต้น สล็อตว่างจะได้รับการกำหนด16 พอยต์ จำนวนพอยต์ที่กำหนดสามารถเปลี่ยนแปลงได้โดยการตั้งค่าพารามิเตอร์ภายในช่วงระหว่าง 0 ถึง 1024 พอยต์(ในหน่วย 16 พอยต์)

## 1.1.1 การกำหนดหมายเลข I/O โดยอัตโนมัติ

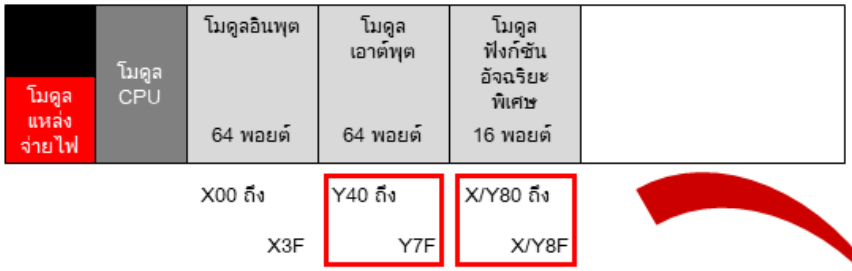
ฐานรองรับโมดูลต่อขยาย จะได้รับการกำหนดหมายเลข I/O โดยอัตโนมัติตามหมายเลข I/O สุดท้ายของเบสหลัก

## 1.1.2 การกำหนดตำแหน่งของหมายเลข I/O

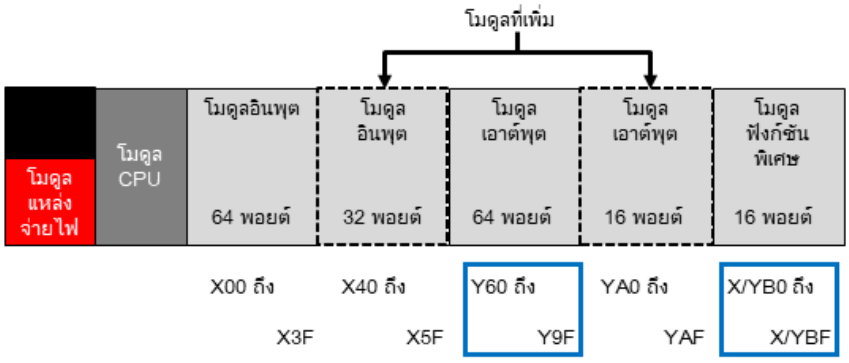
เมื่อกำหนดหมายเลข I/O ด้วยตนเอง หมายเลข I/O ที่กำหนดจะคงที่และไม่เปลี่ยนแปลงแม้ว่ารูปแบบการจัดวางโมดูลจะมีการเปลี่ยนแปลง ซึ่งหมายความว่าสามารถใช้โปรแกรมควบคุมโปรแกรมเดิมในการควบคุมแบบเดิมได้โดยที่ไม่ต้องพิจารณารูปแบบการจัดวางโมดูล

**การกำหนดอัตโนมัติ**

ก่อนเพิ่มโมดูล




หลังเพิ่มโมดูล
(เพิ่มโมดูลอินพุต32 พอยต์ และโมดูลเอาต์พุต16 พอยต์)

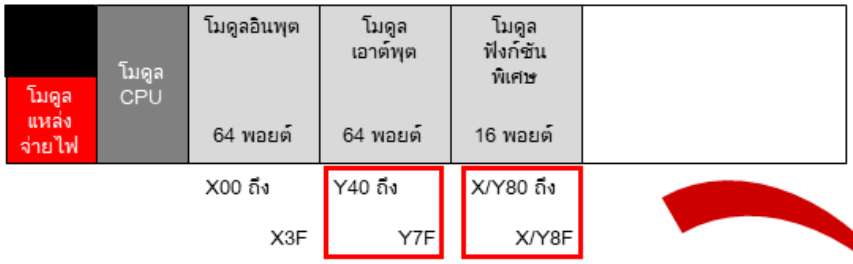



ระบบจะมอบกำหนดเลข I/O อีกครั้งหลังจากการเพิ่มโมดูล

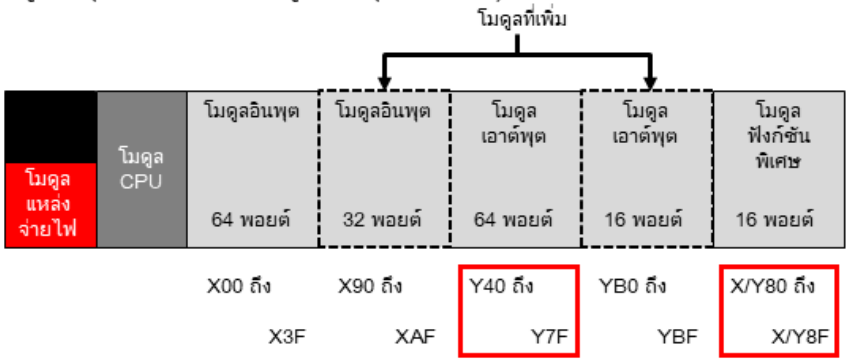
## 1.1.2 การกำหนดตำแหน่งของหมายเลข I/O

**การกำหนดตำแหน่งของหมายเลข I/O ด้วยตนเอง**

ก่อนเพิ่มโมดูล




หลังจากเพิ่มโมดูล
(เพิ่มโมดูลอินพุต32 พอยต์ และโมดูลเอาต์พุต16 พอยต์)

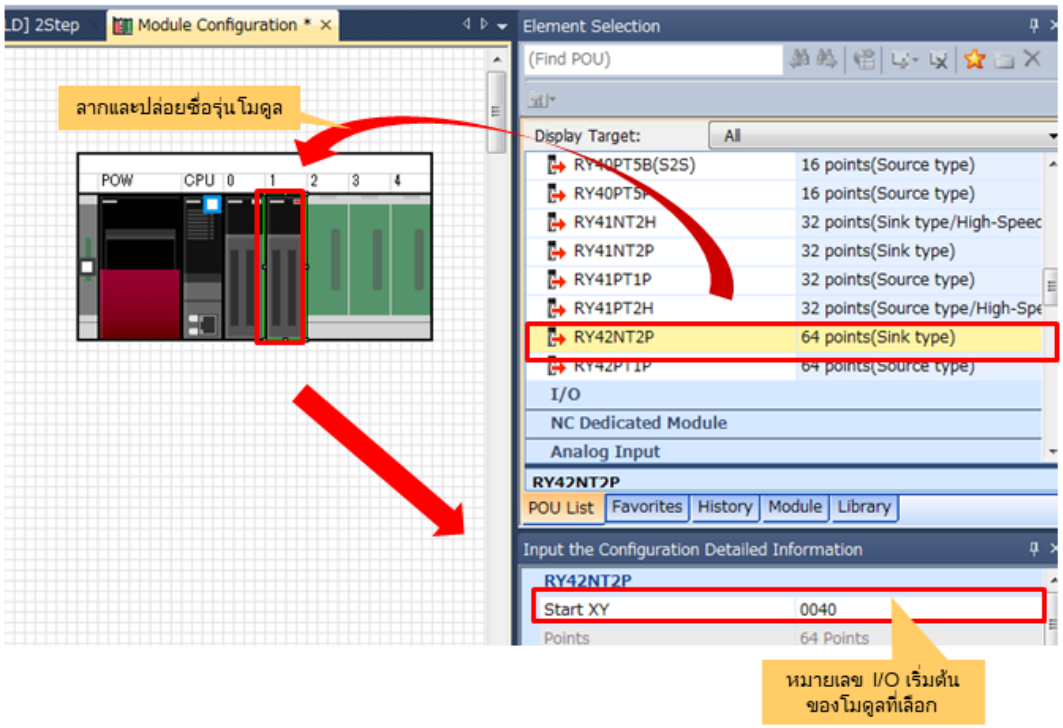



กำหนดตำแหน่งของหมายเลข I/O เดิมได้โดยที่ไม่ต้องพิจารณาถึงการเพิ่มโมดูล

เนื่องจากหมายเลข I/O ของโมดูลที่มีอยู่เดิมจะไม่เปลี่ยนแปลง จึงต้องเพิ่มหรือเปลี่ยนแปลงเฉพาะโปรแกรมที่เกี่ยวข้องกับโมดูลที่เพิ่มเข้ามาเท่านั้น

## 1.1.3 การกำหนดตำแหน่งของหมายเลข I/O อัตโนมัติโดยใช้แผนผังรูปแบบการจัดวางโมดูล

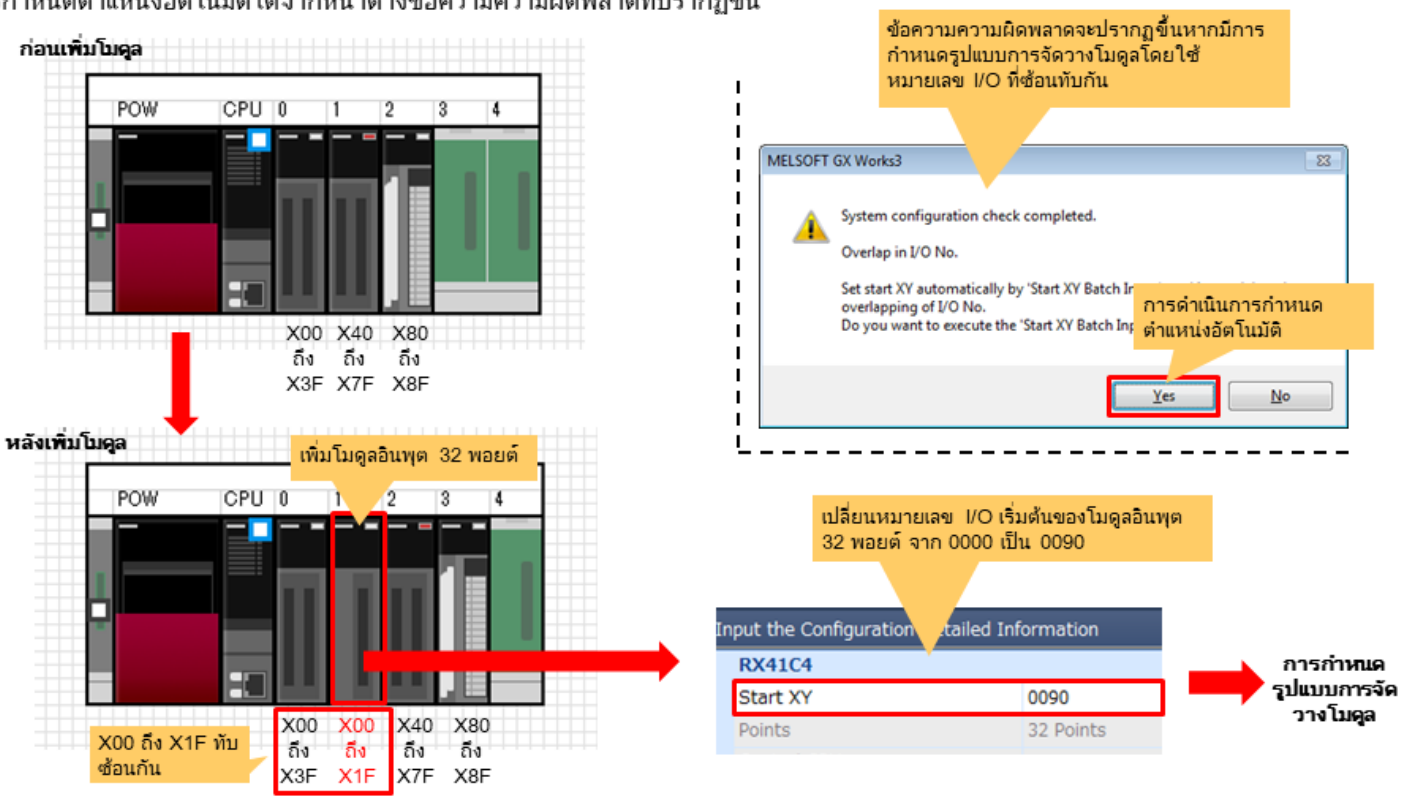
การกำหนดการตั้งค่าโมดูลสามารถทำได้โดยใช้แผนผังรูปแบบการจัดวางโมดูลของซอฟต์แวร์ GX Works3
เลือกชื่อรุ่นโมดูล จากนั้นลากแล้วปล่อยลงบนสล็อตเพื่อวางโมดูลไว้ที่ตำแหน่งดังกล่าว
โมดูลจะได้รับการกำหนดตำแหน่งของหมายเลข I/O ตามลำดับ โดยเริ่มจากโมดูลที่อยู่ใกล้กับโมดูล CPU มากที่สุด ดังที่แสดงในแผนผัง



## 1.1.4 การกำหนดตำแหน่งของหมายเลข I/O ด้วยตนเองโดยใช้แผนผังรูปแบบการจัดวางโมดูล

ตัวอย่างต่อไปนี้จะอธิบายวิธีการกำหนดตำแหน่งของหมายเลข I/O ด้วยตนเองโดยใช้แผนผังรูปแบบการจัดวางโมดูลของ GX Works3

เมื่อรูปแบบการจัดวางโมดูลมีการเปลี่ยนแปลงจากการเพิ่มโมดูลใหม่ลงในแผนผังรูปแบบการจัดวางโมดูล หมายเลข I/O ของโมดูลที่เพิ่มเข้ามาจะซ้อนทับกับหมายเลขที่มีอยู่ หากการวางตำแหน่งโมดูลที่มีอยู่เปลี่ยนแปลงไป ให้แก้ไขหมายเลข I/O เพื่อไม่ให้ซ้อนทับกัน และกำหนดรูปแบบการจัดวางโมดูล
ข้อความความผิดพลาดจะปรากฏขึ้นหากมีการกำหนดรูปแบบการจัดวางโมดูลโดยใช้หมายเลข I/O ที่ซ้อนทับกัน ในขณะนี้สามารถดำเนินการการกำหนดตำแหน่งอัตโนมัติได้จากหน้าต่างข้อความความผิดพลาดที่ปรากฏขึ้น



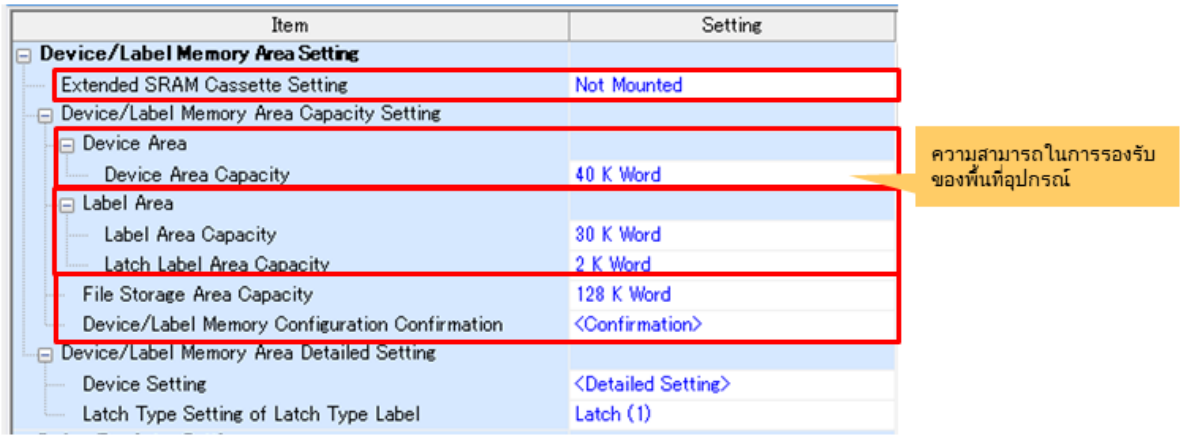
## 1.2 การปรับพื้นที่หน่วยความจำตามสถานะการใช้งานอุปกรณ์

### 1.2.1 การตั้งค่าพื้นที่หน่วยความจำอุปกรณ์/เลเบิล

จำนวนของพอยต์อุปกรณ์ที่ใช้สำหรับโมดูล CPU จะแตกต่างกันไปตามชนิดของโมดูล CPU จำนวนเริ่มต้นของจุดอุปกรณ์จะกำหนดตามความสามารถของโมดูล CPU

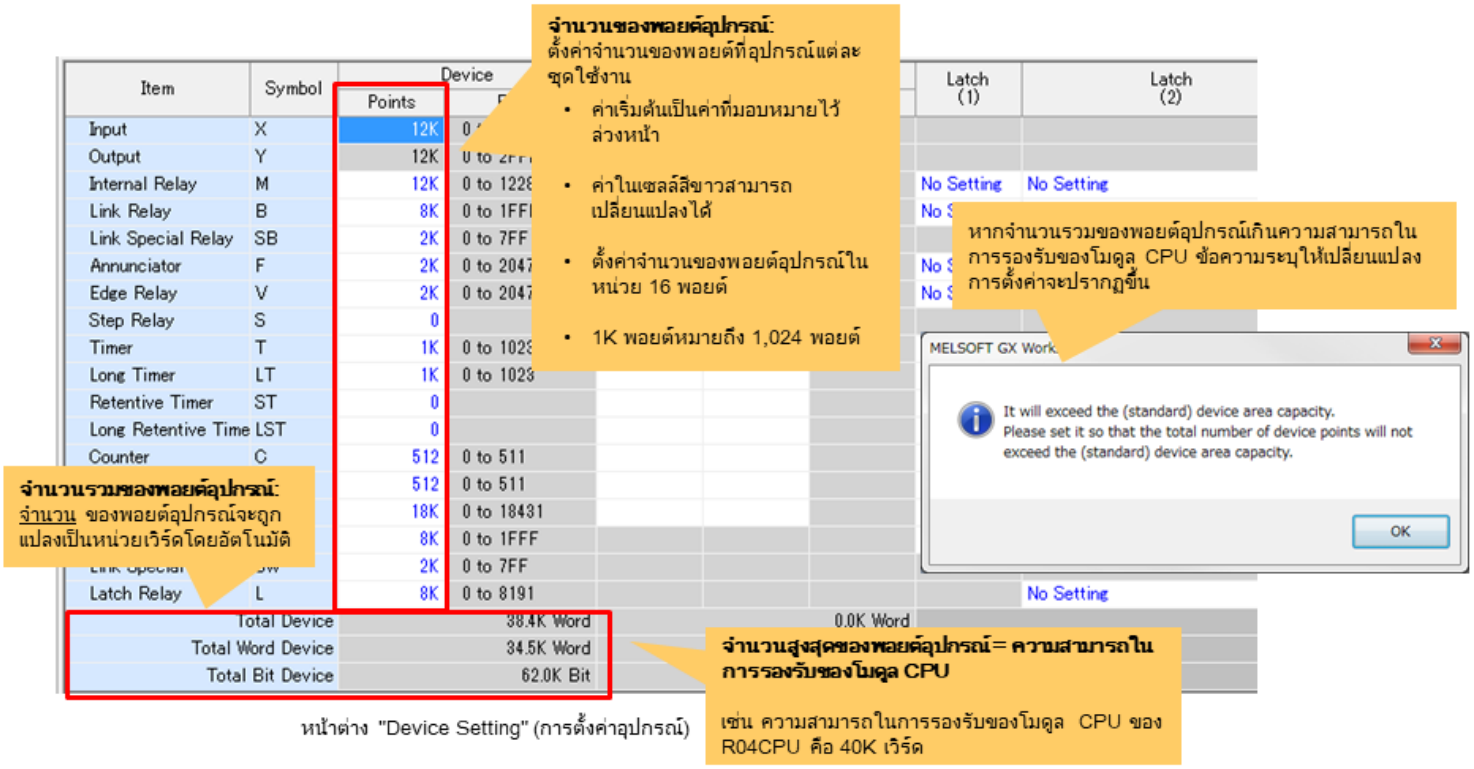
ความสามารถในการรองรับของพื้นที่อุปกรณ์ของ R04CPU คือ 40K เวิร์ด
การลดพื้นที่ที่ไม่ใช้งานจะทำให้จำนวนของพอยต์อุปกรณ์เพิ่มขึ้นจากค่าเริ่มต้น

รูปต่อไปนี้จะแสดงหน้าต่าง "Device/Label Memory Area Setting" (การตั้งค่าพื้นที่หน่วยความจำอุปกรณ์/เลเบิล) เป็นพารามิเตอร์สำหรับปรับพื้นที่หน่วยความจำ
ความสามารถในการรองรับของพื้นที่อุปกรณ์จะเพิ่มขึ้นเมื่อความสามารถในการรองรับของพื้นที่เลเบิลหรือพื้นที่การจัดเก็บไฟล์ลดลง
นอกจากนี้ ความสามารถในการรองรับของพื้นที่หน่วยความจำอุปกรณ์/เลเบิลทั้งหมดยังสามารถขยายได้โดยใช้ SRAM คาสเซท เพิ่มเติม



# 1.2.2 การตั้งค่าอุปกรณ์

จำนวนของพอยต์อุปกรณ์ที่มอบหมายให้กับอุปกรณ์แต่ละชุดสามารถเปลี่ยนแปลงได้ในหน้าต่าง "Device Setting" (การตั้งค่าอุปกรณ์) ค่าเริ่มต้นของอุปกรณ์บางชุดคือ 0 พอยต์ จะต้องกำหนดจำนวนของพอยต์เมื่อใช้อุปกรณ์ดังกล่าว



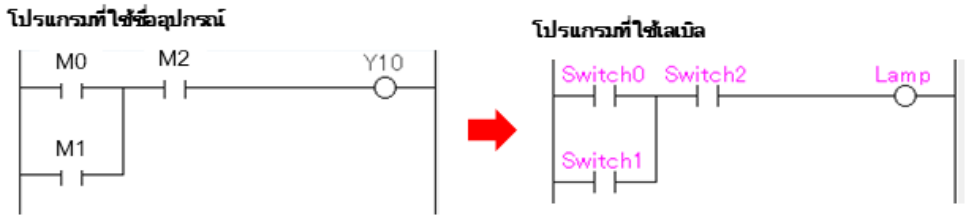

| Item | Symbol | Device Points | Latch (1) | Latch (2) |
|---|---|---|---|---|
| Input | X | 12K | | |
| Output | Y | 12K | | |
| Internal Relay | M | 12K | No Setting | No Setting |
| Link Relay | B | 8K | | |
| Link Special Relay | SB | 2K | | |
| Annunciator | F | 2K | | |
| Edge Relay | V | 2K | | |
| Step Relay | S | 0 | | |
| Timer | T | 1K | | |
| Long Timer | LT | 1K | | |
| Retentive Timer | ST | 0 | | |
| Long Retentive Time | LST | 0 | | |
| Counter | C | 512 | | |
| | | 512 | | |
| | | 18K | | |
| | | 8K | | |
| | | 2K | | |
| Latch Relay | L | 8K | | No Setting |
| Total Device | | 38.4K Word | 0.0K Word | |
| Total Word Device | | 34.5K Word | | |
| Total Bit Device | | 62.0K Bit | | |

หน้าต่าง "Device Setting" (การตั้งค่าอุปกรณ์)

# 1.3 การใช้ชื่อเลเบิลที่สัมพันธ์กับการใช้งาน

## 1.3.1 ข้อดีของการใช้เลเบิล

ชื่ออุปกรณ์ที่ใช้ในโปรแกรมควบคุมจะต้องประกอบด้วยตัวอักษรและหมายเลข เช่น "M0" และ "D5"
เมื่อชื่อเลเบิลสอดคล้องกับการใช้งาน เช่น "StartSwitch" จะทำให้ทราบเป้าหมายการประมวลผลได้อย่างชัดเจน
ชื่อเลเบิลสามารถตั้งค่าได้อย่างอิสระตามการใช้งาน ผู้ใช้ไม่จำเป็นต้องพิจารณาหมายเลขอุปกรณ์สำหรับพื้นที่ที่ใช้เลเบิล




เลเบิลสามารถแบ่งตามขอบเขตการใช้งานเป็นสองชนิดดังนี้

- **โกลเบิลเลเบิล**
  โกลเบิลเลเบิลใช้ได้กับทุกโปรแกรมในโปรเจ็กต์

- **โลเคิลเลเบิล**
  โลเคิลเลเบิลใช้ได้เฉพาะในโปรแกรมที่ได้ลงทะเบียนไว้

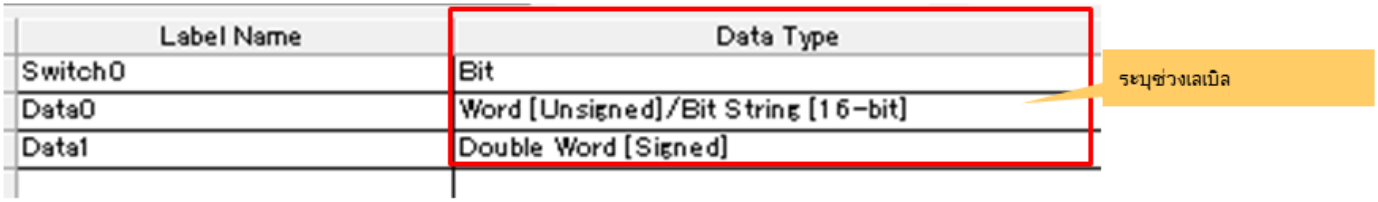
เมื่อใช้เลเบิลกับอุปกรณ์จริง (X, Y) จะต้องกำหนดชื่ออุปกรณ์ให้กับโกลเบิลเลเบิลโดยใช้ GX Works3

## 1.3.2 ชนิดข้อมูลของเลเบิล

ต้องระบุชนิดข้อมูลสำหรับแต่ละเลเบิลเพื่อกำหนดช่วงของค่าที่จะจัดการ
ชนิดข้อมูลประกอบด้วยบิตและจำนวนเต็ม ดังที่แสดงด้านล่าง

| ชนิดข้อมูล | | ช่วงข้อมูล |
|---|---|---|
| ชนิดบิต | | สถานะเปิด/ปิดของอุปกรณ์บิต และสถานะจริง/เท็จของผลการดำเนินการ |
| ชนิดจำนวนเต็ม | เวิร์ด (ไม่ระบุเครื่องหมาย) | 0 ถึง 65,535 |
| | เวิร์ด (ระบุเครื่องหมาย) | -32,768 ถึง 32,767 |
| | ดับเบิลเวิร์ด (ไม่ระบุเครื่องหมาย) | 0 ถึง 4,294,967,295 |
| | ดับเบิลเวิร์ด (ระบุเครื่องหมาย) | -2,147,483,648 ถึง 2,147,483,647 |

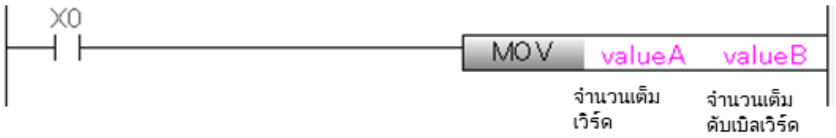
เมื่อใช้ชนิดจำนวนเต็ม ให้เลือกชนิดเวิร์ดหรือดับเบิลเวิร์ดตามช่วงข้อมูล และเลือกชนิดระบุเครื่องหมายหรือไม่ระบุเครื่องหมายตามความจำเป็นในการจัดการข้อมูล
ระบุชนิดข้อมูลของเลเบิลเมื่อทำการตั้งค่าชื่อเลเบิลโดยใช้ GX Works3




หน้าต่างการตั้งค่าเลเบิล

## 1.3.3 ชื่อเลเบิลที่แสดงถึงชนิดข้อมูล

การโอนถ่ายชนิดข้อมูลต้นทางและปลายทางที่แตกต่างกัน อาจก่อให้เกิดความผิดพลาดในการแปลงหรือได้ผลลัพธ์ที่ผิดพลาด
ด้านล่างนี้คือตัวอย่างโปรแกรมของกรณีดังกล่าว

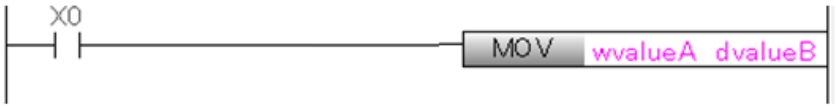



ค่าที่เป็นดับเบิลเวิร์ดจะไม่สามารถโอนถ่ายไปยังเลเบิลชนิดเวิร์ดได้ อย่างไรก็ตาม ไม่สามารถระบุชนิดข้อมูลได้จากชื่อเลเบิล

ดังนั้น สามารถเพิ่มส่วนนำหน้าซึ่งระบุชนิดข้อมูลให้กับชื่อเลเบิล เพื่อช่วยให้ทราบชนิดของข้อมูลจากการอ่านชื่อ
การตั้งชื่อเลเบิลลักษณะนี้มีชื่อเรียกว่า เทคนิคการตั้งชื่อตัวแปรแบบฮังกาเรียน

| ชนิดข้อมูล | | ช่วงข้อมูล | ส่วนนำหน้า | การขยายส่วนนำหน้า |
|---|---|---|---|---|
| ชนิดบิต | | สถานะเปิด/ปิดของอุปกรณ์บิต และ สถานะจริง/เท็จของผลการดำเนินการ | b | **b**it (บิต) |
| ชนิดจำนวนเต็ม | เวิร์ด (ไม่ระบุเครื่องหมาย) | 0 ถึง 65,535 | u | **u**nsigned word (เวิร์ดไม่ระบุเครื่องหมาย) |
| | เวิร์ด (ระบุเครื่องหมาย) | -32,768 ถึง 32,767 | w | signed **w**ord (เวิร์ดระบุเครื่องหมาย) |
| | ดับเบิลเวิร์ด (ไม่ระบุเครื่องหมาย) | 0 ถึง 4,294,967,295 | ud | **u**nsigned **d**ouble-word (ดับเบิลเวิร์ดไม่ระบุเครื่องหมาย) |
| | ดับเบิลเวิร์ด (ระบุเครื่องหมาย) | -2,147,483,648 ถึง 2,147,483,647 | d | signed **d**ouble-word (ดับเบิลเวิร์ดระบุเครื่องหมาย) |

ตัวอย่างโปรแกรมที่ส่วนบนสุดของหน้านี้สามารถเขียนได้ โดยใช้เทคนิคการตั้งชื่อตัวแปรแบบฮังกาเรียน:



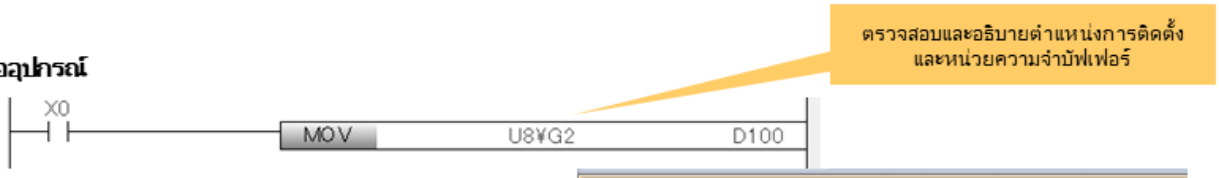

เมื่อใช้เทคนิคการตั้งชื่อตัวแปรแบบฮังกาเรียน จะช่วยให้ทราบเมื่อมีชนิดข้อมูลที่ไม่สอดคล้องกันในขั้นตอนการเขียนโปรแกรม
สำหรับส่วนที่เหลือของหลักสูตรนี้ ชื่อเลเบิลในตัวอย่างจะเขียนโดยใช้เทคนิคการตั้งชื่อตัวแปรแบบฮังกาเรียน

## 1.3.4 การใช้เลเบิลที่เตรียมไว้แล้ว

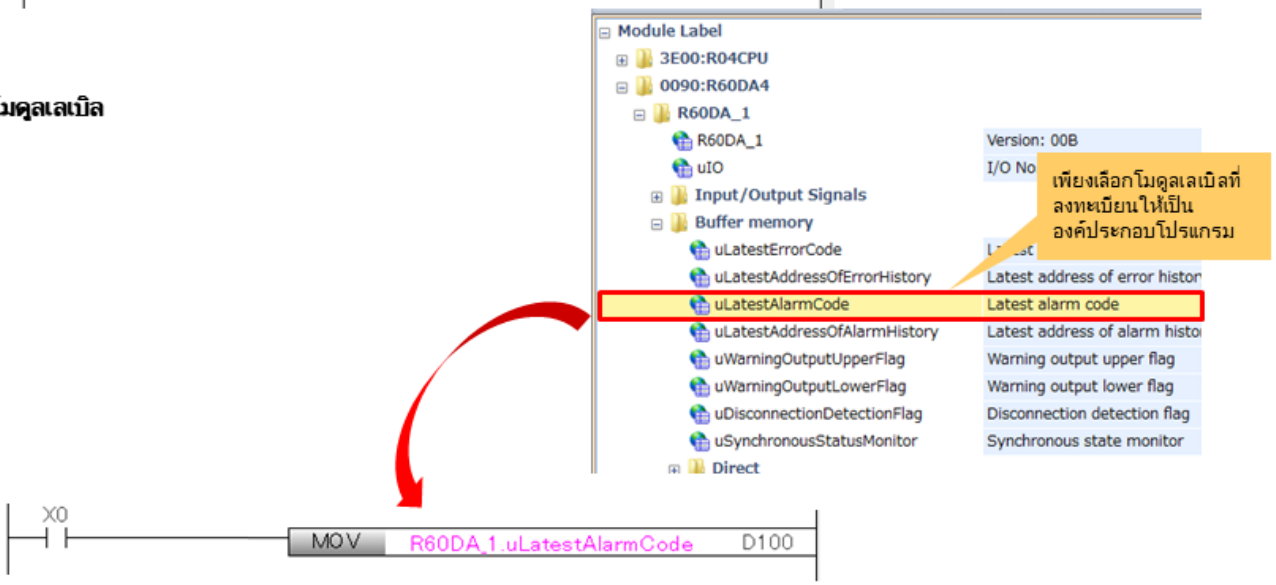
เมื่อตั้งค่ารูปแบบการจัดวางโมดูลในแผนผังรูปแบบการจัดวางโมดูล,เลเบิล (โมดูลเลเบิล) ที่สอดคล้องกับตำแหน่งการติดตั้งของโมดูลจะได้รับการลงทะเบียนโดยอัตโนมัติ

ในการออกแบบโปรแกรมโดยใช้ชื่ออุปกรณ์ จะต้องตรวจสอบหมายเลขอุปกรณ์และหน่วยความจำบัฟเฟอร์ที่สอดคล้องกับสัญญาณต่างๆ จากคู่มือ ซึ่งจะต้องใช้เวลา เวลาในการตั้งโปรแกรมสามารถลดลงได้โดยการใช้โมดูลเลเบิล เนื่องจากผู้ใช้เพียงแค่เลือกเลเบิลจากรายการเท่านั้น

**เมื่อใช้ชื่ออุปกรณ์**

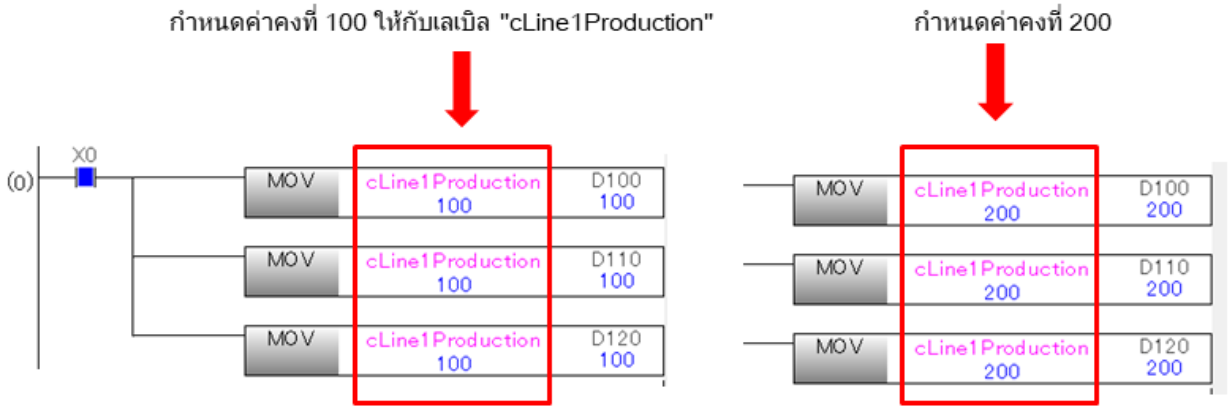



**เมื่อใช้โมดูลเลเบิล**

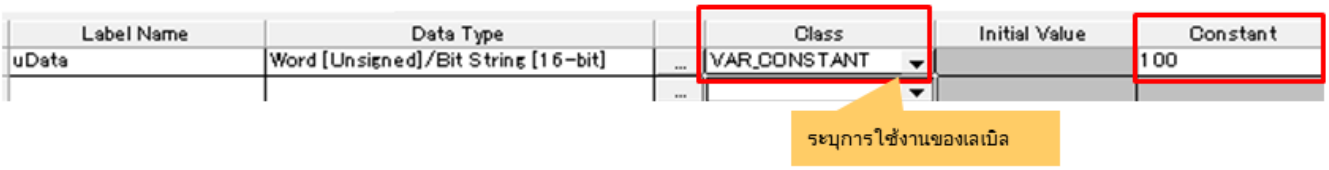


## 1.3.5 การกำหนดค่าคงที่ให้กับเลเบิล

ค่าคงที่สามารถกำหนดให้กับเลเบิลได้
เมื่อกำหนดค่าคงที่ให้กับเลเบิล จะสามารถเปลี่ยนแปลงค่าดังกล่าวได้โดยไม่ต้องแก้ไขโปรแกรม
ค่าคงที่เดียวกันที่ใช้สำหรับหลายเลเบิลจะสามารถเปลี่ยนแปลงพร้อมกันได้




ในการกำหนดค่าคงที่ให้กับเลเบิล ให้เปลี่ยนคลาสที่ระบุการใช้งานเลเบิลในหน้าต่างสำหรับการตั้งค่าเลเบิล
สำหรับโลเคิลเลเบิล ให้เลือก "VAR_CONSTANT"

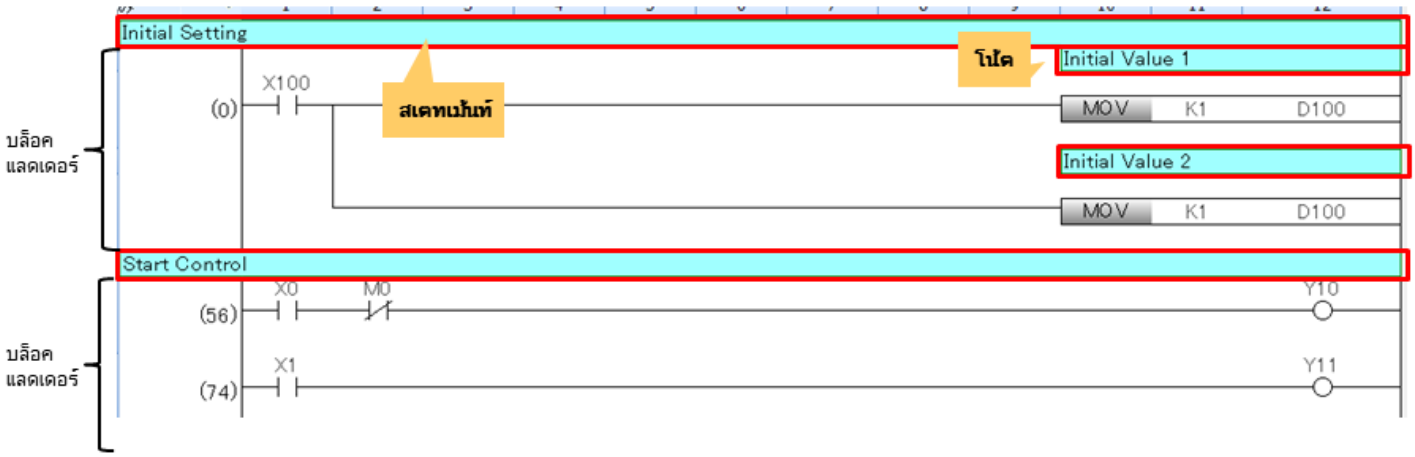


# 1.4 การปรับปรุงการอ่านโปรแกรม

สามารถเพิ่มคอมเม้นต์ เช่น รายละเอียดการประมวลผลและชื่ออุปกรณ์ ให้กับโปรแกรมได้
คอมเม้นต์ จะช่วยในการอธิบายลักษณะการทำงานของโปรแกรม


ชนิดของคอมเม้นต์ สามารถเลือกได้ตามองค์ประกอบหรือช่วงของโปรแกรม
ในตัวอย่างโปรแกรมด้านบน ได้มีการเพิ่ม "คอมเม้นต์ สำหรับอุปกรณ์/เลเบิล" เพื่ออธิบายถึงการใช้งานของอุปกรณ์หรือเลเบิล และชนิดของอุปกรณ์ I/O ที่เชื่อมต่อไว้

นอกจากนี้ ชนิดของคอมเม้นต์ ยังมี "ข้อความ" ซึ่งเพิ่มเข้ามายังบล็อคแลดเดอร์เพื่อช่วยให้ทราบถึงกระบวนขั้นตอนของการประมวลผล และ "โน้ต" ซึ่งช่วยให้ทราบถึงรายละเอียดของคอยล์และคำสั่งของการใช้งาน



## 1.5 สรุปสำหรับบทนี้

ในบทนี้คุณได้เรียนรู้เกี่ยวกับ:

- การกำหนดหมายเลข I/O
- การปรับพื้นที่หน่วยความจำ
- การออกแบบโปรแกรมโดยใช้เลเบิล
- คอมเม้นต์ในโปรแกรม

ประเด็นที่สำคัญ

| | |
|---|---|
| การกำหนดหมายเลข I/O | • สล็อตจะได้รับกำหนดหมายเลข I/O โดยอัตโนมัติ โดยเริ่มจากสล็อตที่อยู่ใกล้กับโมดูล CPU มากที่สุด<br>• เมื่อกำหนดหมายเลข I/O ให้กับโมดูลต่างๆ ด้วยตนเอง จะสามารถนำโปรแกรมไปใช้ในระบบที่แตกต่างกันได้ |
| การตั้งอุปกรณ์และการตั้งค่าพื้นที่หน่วยความจำ | • จำนวนของพอยต์อุปกรณ์จะแตกต่างกันไปตามโมดูล CPU<br>• จำนวนของพอยต์อุปกรณ์สามารถเพิ่มได้โดยการลดพื้นที่หน่วยความจำที่ไม่ใช้งาน<br>• จำนวนของพอยต์อุปกรณ์ สำหรับอุปกรณ์แต่ละชุดสามารถเปลี่ยนแปลงได้ตามสถานะการใช้งาน |
| การออกแบบโปรแกรมโดยใช้เลเบิล | การใช้เลเบิลจะช่วยให้ทราบเป้าหมายการประมวลผลได้อย่างชัดเจน |
| คอมเม้นต์ | การระบุคอมเม้นต์ จะช่วยให้เข้าใจถึงรายละเอียดและขั้นตอนของการประมวลผลได้ง่ายยิ่งขึ้น |

# บทที่ 2 การออกแบบโปรแกรมขั้นสูง

บทนี้จะอธิบายเกี่ยวกับการใช้อุปกรณ์ และการออกแบบโปรแกรมเลเบิลขั้นสูง

# 2.1 การใช้เวิร์ดดีไวซ์ในหน่วยบิต

โดยปกติแล้ว เวิร์ดดีไวซ์ เช่น ดาต้ารีจิสเตอร์ จะใช้ในหน่วยเวิร์ด แต่สามารถใช้ในหน่วยบิตได้เช่นกัน
หน่วยบิตใช้เพื่อระบุบิตหนึ่งๆ ในดาต้ารีจิสเตอร์(D)

ตัวอย่าง) ดาต้ารีจิสเตอร์(D)

| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

รูปแบบข้อมูลจำเพาะของบิต

D □.□

- สัญลักษณ์ของเวิร์ดดีไวซ์ (D, W, หรือ R)
- หมายเลขอุปกรณ์
- บิต (0 ถึง F)

เมื่อ b5 ของดาต้ารีจิสเตอร์"D0" เป็น 1

| | F | | | | | | | | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | **1** | 0 | 1 | 1 | 0 | 1 |

D0.5 เนื่องจาก D0.5 เป็น 1 หน้าสัมผัสจึงทำงาน

เมื่อ b5 ของดาต้ารีจิสเตอร์"D0" เป็น 0

| | F | | | | | | | | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | **0** | 0 | 1 | 1 | 0 | 1 |

D0.5 เนื่องจาก D0.5 เป็น 0 หน้าสัมผัสจึงไม่ทำงาน

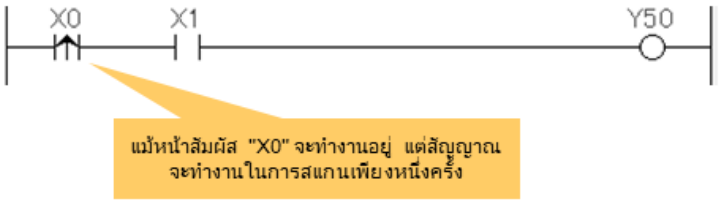

เนื่องจาก D10.2 เป็น 0 ค่าจึงกลับเป็น 1 (เปิด)

ในการใช้เลเบิล ให้ระบุคำอธิบายเป็น "uData.2" และ "uData.5"

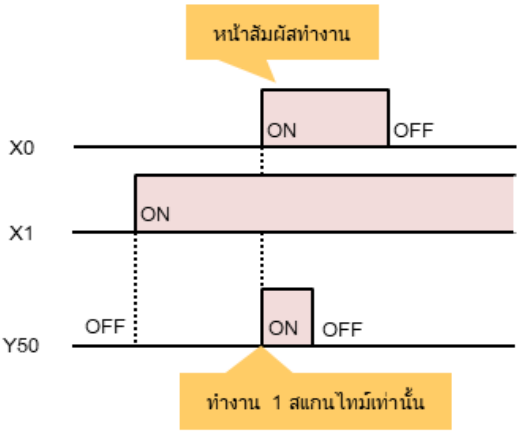
## 2.2 อุปกรณ์ทำงานเมื่อสถานะของหน้าสัมผัสมีการเปลี่ยนแปลง

สัญญาณจะทำงานที่ขอบขาขึ้น หรือที่ขอบลาลงของหน้าสัมผัสเป็นระยะเวลา 1 แสกนไทม์ ฟังก์ชั่นนี้มีประโยชน์สำหรับการควบคุมสัญญาณขาขึ้นและขาลงตามเงื่อนไขอินพุต

**ข้อมูลขอบขาขึ้นของหน้าสัมผัส**

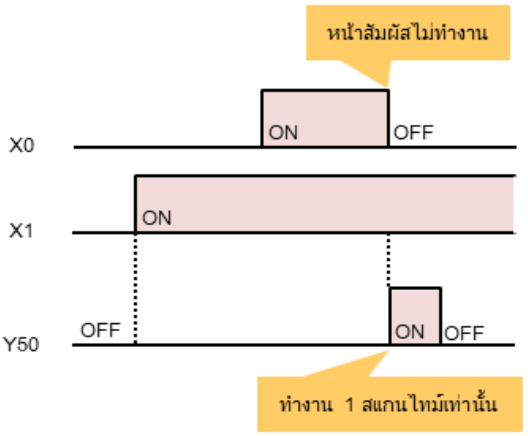



แม้หน้าสัมผัส "X0" จะทำงานอยู่ แต่สัญญาณจะทำงานในการสแกนเพียงหนึ่งครั้ง




**ข้อมูลขอบขาลงของหน้าสัมผัส**


แม้หน้าสัมผัส "X0" จะไม่ทำงานอยู่ แต่สัญญาณจะทำงานในการสแกนเพียงหนึ่งครั้ง



## 2.3 การคงค่าการวัดของไทม์เมอร์

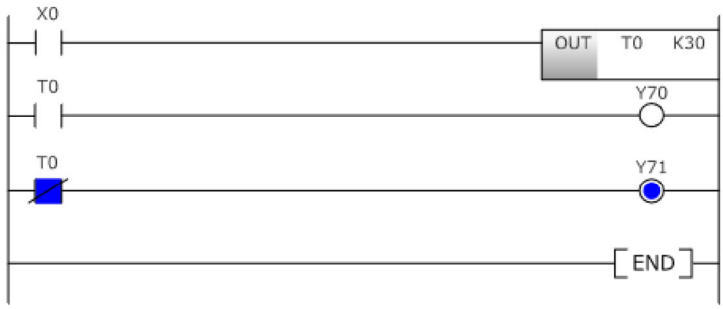
หมวดนี้จะอธิบายเกี่ยวกับไทม์เมอร์ชนิดหนึ่ง นั่นคือ รีเทนทีฟไทม์เมอร์ ซึ่งสามารถคงค่าเวลาการวัดได้

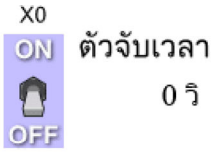
## 2.3.1 ความแตกต่างระหว่างไทม์เมอร์ และรีเทนทีฟไทม์เมอร์

ก่อนจะอธิบายถึงรีเทนทีฟไทม์เมอร์ โปรดดูวิธีการทำงานของไทม์เมอร์ก่อนว่าทำงานอย่างไร

ไทม์เมอร์จะเริ่มหน่วงเวลาเมื่อคอยล์ทำงาน เมื่อกำหนดเวลาเรียบร้อยแล้ว ไทม์เมอร์จะเริ่มจับเวลา และเมื่อครบเวลาสถานะหน้าสัมผัสจะทำงานขึ้นมา เมื่อคอยล์ไม่ทำงาน ไทม์เมอร์ที่จับเวลาไว้จะถูกรีเซ็ตเป็น "0" สัญลักษณ์อุปกรณ์ของไทม์เมอร์คือ "T"

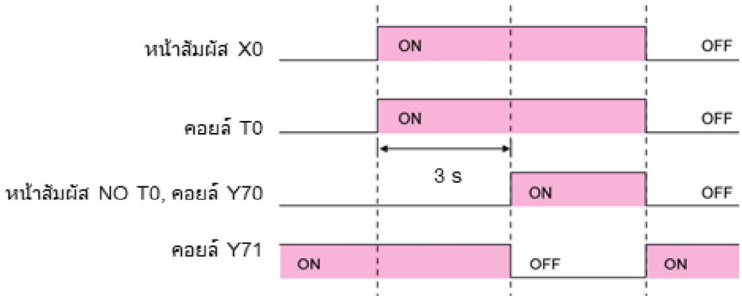
ใช้สวิตช์อินพุตที่ด้านขวาเพื่อดูวิธีการทำงานของ ไทม์เมอร์




ตัวจับเวลา:
0 วิ

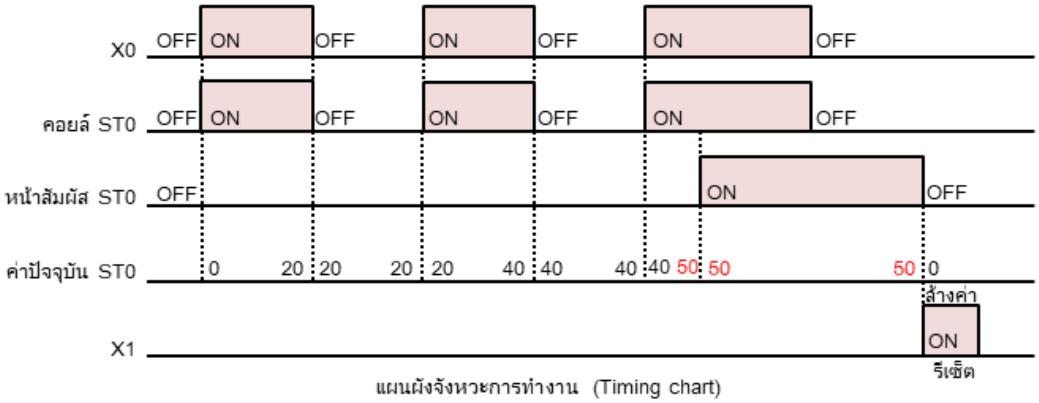

เมื่อผ่านไปสามวินาที หลังจากที่ X0 ทำงาน Y70 จะทำงานและ Y71 จะไม่ทำงาน




แผนผังจังหวะการทำงาน (Timing chart)

## 2.3.1 ความแตกต่างระหว่างไทม์เมอร์และรีเทนทีฟไทม์เมอร์

รีเทนทีฟไทม์เมอร์ เป็นประโยชน์ในการจับเวลารวมของการทำงาน รีเทนทีฟไทม์เมอร์จะเริ่มการจับเวลาเมื่อคอยล์ทำงาน เมื่อเวลาที่กำหนดผ่านไป เวลาจะหมดลง และสถานะหน้าสัมผัสจะเป็นเปิด เมื่อคอยล์ไม่ทำงาน รีเทนทีฟไทม์เมอร์ที่จับเวลาไว้จะไม่ถูกรีเซ็ต เมื่อคอยล์ทำงานอีกครั้ง การจับเวลาจะเริ่มขึ้นอีกครั้งต่อจากค่าที่คงไว้ สัญลักษณ์อุปกรณ์ของรีเทนทีฟไทม์เมอร์คือ "ST"


แผนผังจังหวะการทำงาน (Timing chart)

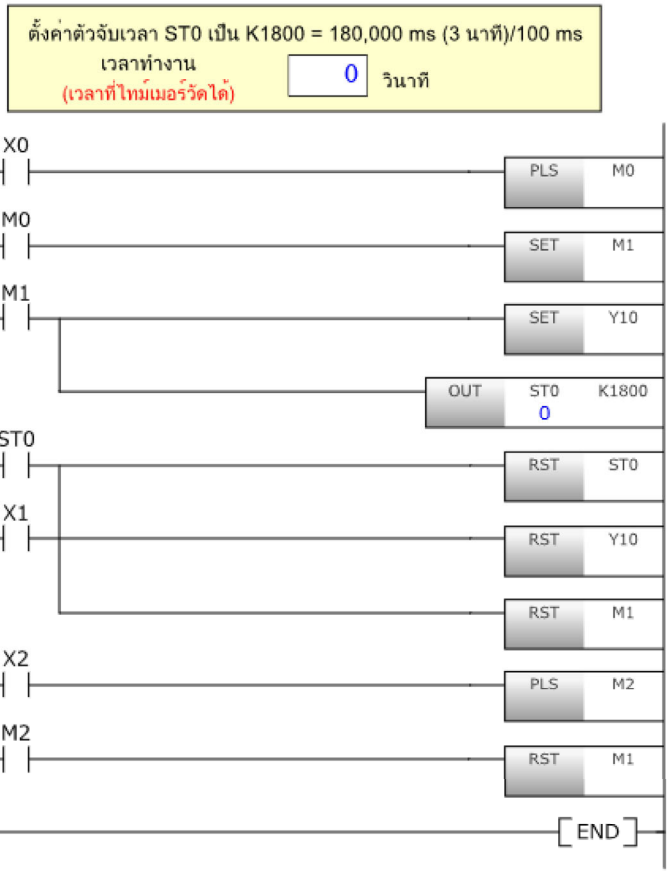
## 2.3.2 ตัวอย่างโปรแกรมของรีเทนทีฟไทม์เมอร์

มาดูวิธีการทำงานของรีเทนทีฟไทม์เมอร์ โดยการจำลองการทำงานของเครื่องจักรที่กำลังทำงานอยู่โดยใช้สวิตช์อินพุต (X0 ถึง X2)
*รีเทนทีฟไทม์เมอร์ (ST0) ถูกตั้งค่าให้นับเพิ่มครั้งละ 100 ms

1. เมื่อ X0 ทำงาน การทำงานจะเริ่มขึ้น
2. เมื่อ X2 ทำงาน การทำงานจะหยุดชั่วคราว และค่าปัจจุบันจะถูกคงไว้
3. เมื่อ X0 ทำงานอีกครั้ง การทำงานจะเริ่มขึ้นอีกครั้ง
4. เมื่อ X1 ทำงาน การทำงานจะหยุดลง และค่าปัจจุบันจะถูกรีเซ็ต

X0 ถึง X2: สวิตช์อินพุต

Y10: สัญญาณเริ่ม



## 2.3.3 การตั้งค่าสำหรับรีเทนทีฟไทม์เมอร์

ตามค่าเริ่มต้น จำนวนของพอยต์ที่รีเทนทีฟไทม์เมอร์ใช้คือ "0"
ก่อนที่จะใช้งาน รีเทนทีฟไทม์เมอร์ ให้ตั้งค่าจำนวนพอยต์ใน "Device Setting" (การตั้งค่าอุปกรณ์) ของ CPU Parameter โดยใช้ซอฟแวร์ GX Works3

ในตัวอย่างต่อไปนี้จะตั้งค่า 64 พอยต์ (ST0 ถึง ST63) สำหรับรีเทนทีฟไทม์เมอร์

| Item | Symbol | Device | | Local Device | | | Latch (1) | Latch (2) |
|---|---|---|---|---|---|---|---|---|
| | | Points | Range | Start | End | Points | | |
| Input | X | 12K | 0 to 2FFF | | | | | |
| Output | Y | 12K | 0 to 2FFF | | | | | |
| Internal Relay | M | 12K | 0 to 12287 | | | | No Setting | No Setting |
| Link Relay | B | 16K | 0 to 3FFF | | | | No Setting | No Setting |
| Link Special Relay | SB | 16K | 0 to 3FFF | | | | | |
| Annunciator | F | 2K | 0 to 2047 | | | | No Setting | No Setting |
| Edge Relay | V | 2K | 0 to 2047 | | | | No Setting | No Setting |
| Step Relay | S | 0 | | | | | | |
| Timer | T | 1K | 0 to 1023 | | | | No Setting | No Setting |
| Long Timer | LT | 1K | 0 to 1023 | | | | No Setting | No Setting |
| Retentive Timer | ST | 64 | 0 to 63 | | | | No Setting | No Setting |
| Long Retentive Time | LST | 0 | | | | | No Setting | No Setting |
| Counter | C | 512 | 0 to 511 | | | | No Setting | No Setting |
| Long Counter | LC | 512 | 0 to 511 | | | | No Setting | No Setting |
| Data Register | D | 18K | 0 to 18431 | | | | No Setting | No Setting |
| Link Register | W | 8K | 0 to 1FFF | | | | No Setting | No Setting |
| Link Special Registe | SW | 2K | 0 to 7FF | | | | | |
| Latch Relay | L | 8K | 0 to 8191 | | | | | No Setting |
| Total Device | | | 39.9K Word | | | 0.0K Word | | |
| Total Word Device | | | 34.6K Word | | | 0.0K Word | | |
| Total Bit Device | | | 84.2K Bit | | | 0.0K Bit | | |

## 2.3.4 การใช้เลเบิลในการกำหนดไทม์เมอร์

กำหนดประเภทข้อมูล (Data Type) ให้เป็น "Timer" เมื่อใช้เลเบิลแทนอุปกรณ์ไทม์เมอร์

| Label Name | Data Type |
|---|---|
| uTimer1 | Timer |
| uTimer2 | Retentive Timer |

ในการใช้งานรีเทนทีฟไทม์เมอร์ จำเป็นต้องทำการตั้งค่าอุปกรณ์ดังที่อธิบายไว้ในหมวดก่อนหน้านี้ อย่างไรก็ตาม ขั้นตอนดังกล่าวไม่จำเป็นสำหรับเลเบิล

## 2.4 การเปลี่ยนหน่วยการจับเวลาของไทม์เมอร์

หน่วยการจับเวลาและเวลาในการวัด จะแตกต่างกันไปตามชนิดของไทม์เมอร์

- ไทม์เมอร์ความเร็วสูง (High-speed timer)
- ไทม์เมอร์ความเร็วต่ำ (Low-speed timer)
- ลองไทม์เมอร์สามารถทำการจับเวลาระยะยาวได้

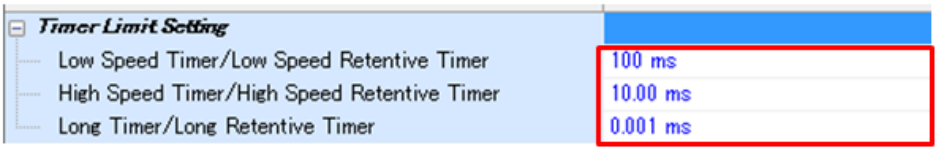
จากไทม์เมอร์ด้านบนนี้แต่ละชนิดจะมีฟังก์ชันตัวจับเวลาแบบรีเทนทีฟไทม์เมอร์

| ชนิด | หน่วยการวัด | ตัวอย่างโปรแกรม | การทำงาน | การคงค่าปัจจุบัน |
|---|---|---|---|---|
| ไทม์เมอร์ความเร็วต่ำ | 100 ms (ค่าเริ่มต้น) | OUT T0 K50 | ไทม์เมอร์ความเร็วต่ำ T0 จะจับเวลา 5 วินาที | 16 บิต |
| ไทม์เมอร์ความเร็วสูง | 10.00 ms (ค่าเริ่มต้น) | OUTH T1 K50 | ไทม์เมอร์ความเร็วสูง T1 จะจับเวลา 0.5 วินาที | |
| รีเทนทีฟไทม์เมอร์แบบความเร็วต่ำ | 100 ms (ค่าเริ่มต้น) | OUT ST0 K50 | รีเทนทีฟไทม์เมอร์ความเร็วต่ำ ST0 จะจับเวลา 5 วินาที | |
| รีเทนทีฟไทม์เมอร์ความเร็วสูง | 10.00 ms (ค่าเริ่มต้น) | OUTH ST1 K50 | รีเทนทีฟไทม์เมอร์ความเร็วสูง ST1 จะจับเวลา 0.5 วินาที | |
| ลองไทม์เมอร์ | 0.001 ms (ค่าเริ่มต้น) | OUT LT0 K50 | ลองไทม์เมอร์ LT0 จะจับเวลา 0.005 วินาที | 32 บิต |
| ลองรีเทนทีฟไทม์เมอร์ | | OUT LST0 K50 | ลองรีเทนทีฟไทม์เมอร์ LST0 จะจับเวลา 0.005 วินาที | |

หน่วยของการจับเวลาเริ่มต้นคือ 100 ms สำหรับไทม์เมอร์ความเร็วต่ำ, 10 ms สำหรับไทม์เมอร์ความเร็วสูง และ 0.001 ms สำหรับลองไทม์เมอร์
สำหรับวิธีเปลี่ยนหน่วยจับเวลา โปรดดูที่หน้าถัดไป

## 2.4 การเปลี่ยนหน่วยจับเวลา ของไทม์เมอร์

หน่วยจับเวลาของไทม์เมอร์สามารถเปลี่ยนแปลงได้ใน "Timer Limit Setting" (การตั้งค่าขีดจำกัดไทม์เมอร์) ของ CPU parameter



## 2.5 การจัดการอุปกรณ์จำนวนมาก (อินเด็กซ์รีจิสเตอร์)

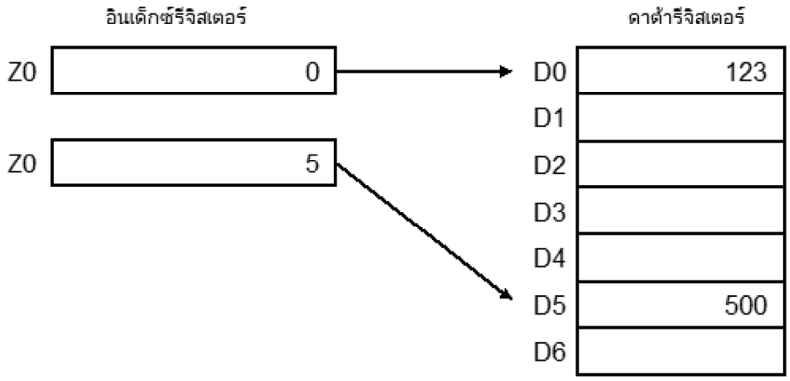
อินเด็กซ์รีจิสเตอร์ (Z) จะใช้ร่วมกับอุปกรณ์อื่นเพื่อกำหนด (แก้ไข) หมายเลขอุปกรณ์ของอุปกรณ์ที่เป็นเป้าหมายการควบคุม ทางอ้อม อินเด็กซ์รีจิสเตอร์มีประโยชน์สำหรับการลดความซับซ้อนของโปรแกรม เนื่องจากสามารถทำให้อุปกรณ์จำนวนมากรวมเป็นชุดเดียวกันได้

- อินเด็กซ์รีจิสเตอร์จะใส่ไว้หลังจากสัญลักษณ์อุปกรณ์และหมายเลขอุปกรณ
- หมายเลขอุปกรณ์ที่จะควบคุมเป้าหมายจริง = สัญลักษณ์อุปกรณ์ (หมายเลขอุปกรณ์ + อินเด็กซ์รีจิสเตอร์)
- โดยค่าเริ่มต้น ของจำนวนอุปกรณ์สำหรับอินเด็กซ์รีจิสเตอร์คือ 20 พอยต์ (Z0 ถึง Z19)

## 2.5.1 ตัวอย่างการใช้งานของอินเด็กซ์รีจิสเตอร์

เมื่ออธิบายอุปกรณ์เป็น "D0Z0" จะหมายถึง D(0+Z0)

ตัวอย่าง) เมื่อ Z0 เป็น 0 หมายเลขอุปกรณ์จะเป็น D0
เมื่อ Z0 เป็น 5 หมายเลขอุปกรณ์จะเป็น D5

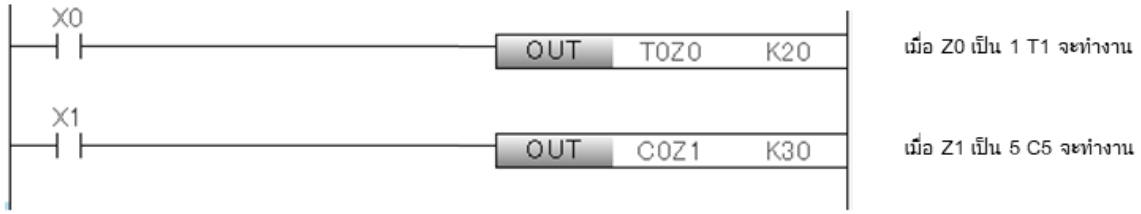


## 2.5.2 อุปกรณ์ที่สามารถใช้งานร่วมกับอินเด็กซ์รีจิสเตอร์ได้

อุปกรณ์ที่สามารถใช้งานร่วมกับอินเด็กซ์รีจิสเตอร์ได้มีดังต่อไปนี้:

| | |
|---|---|
| อุปกรณ์ประเภทบิต | X, Y, M, L, S, B, F |
| อุปกรณ์ประเภทเวิร์ดดีไวซ์ | T, C, D, R, W |
| ค่าคงที่ | K, H |
| พอยต์เตอร์ | P |

หมายเหตุ) สำหรับหน้าสัมผัสและคอยล์ที่ใช้ในไทม์เมอร์และเคาน์เตอร์ จะสามารถใช้ได้เฉพาะอินเด็กซ์รีจิสเตอร์"Z0" หรือ "Z1" เท่านั้น




เมื่อ Z0 เป็น 1 T1 จะทำงาน

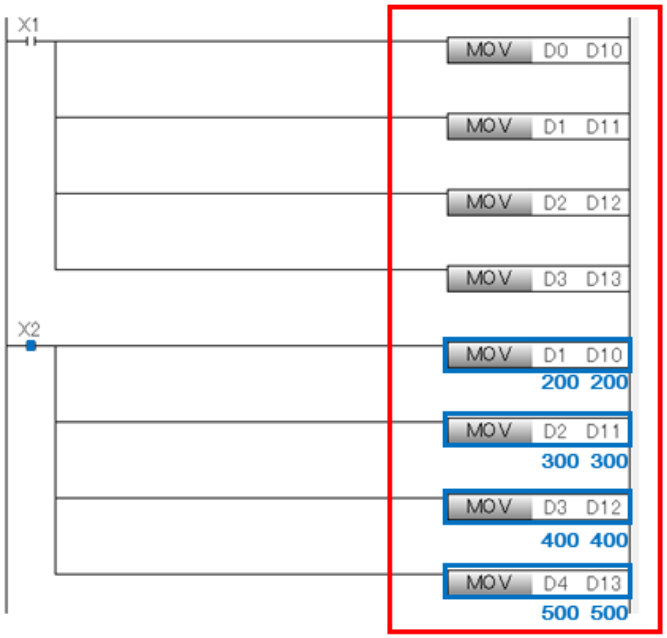
เมื่อ Z1 เป็น 5 C5 จะทำงาน

## 2.5.3 การลดความซับซ้อนของโปรแกรมโดยใช้อินเด็กซ์รีจิสเตอร์

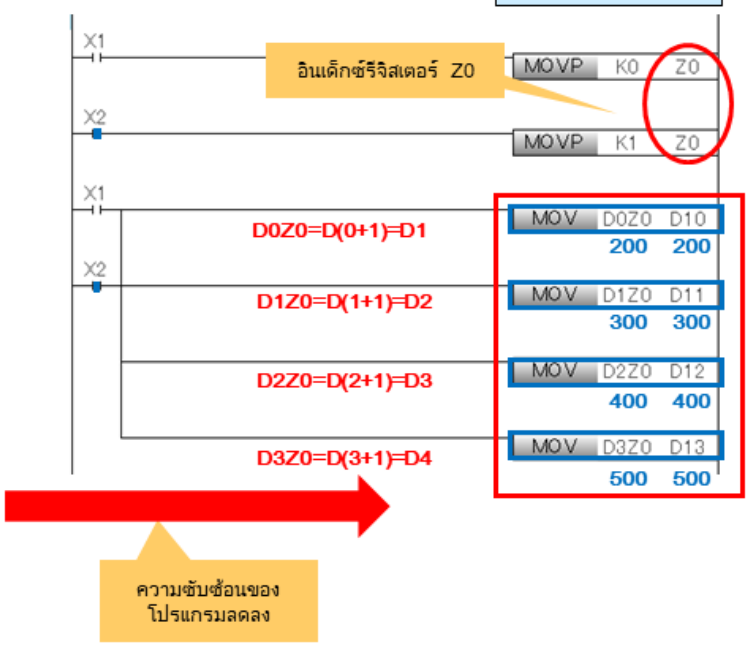
โปรแกรมที่แสดงด้านล่างจะส่งค่าใน "D0 ถึง D4" ไปยัง "D10 ถึง D13" เมื่อ X1 หรือ X2 ทำงาน โปรแกรม (1) และ (2) จะทำให้ได้ผลลัพธ์เดียวกัน ในโปรแกรม (1) จะส่งข้อมูลโดยตรง ในโปรแกรม (2) จะส่งข้อมูลโดยอ้อม โดยใช้อินเด็กซ์รีจิสเตอร์

**ค่าเริ่มต้นที่เก็บไว้**
D0=100
D1=200
D2=300
D3=400
D4=500

**(1) เมื่อไม่ใช้อินเด็กซ์รีจิสเตอร์**



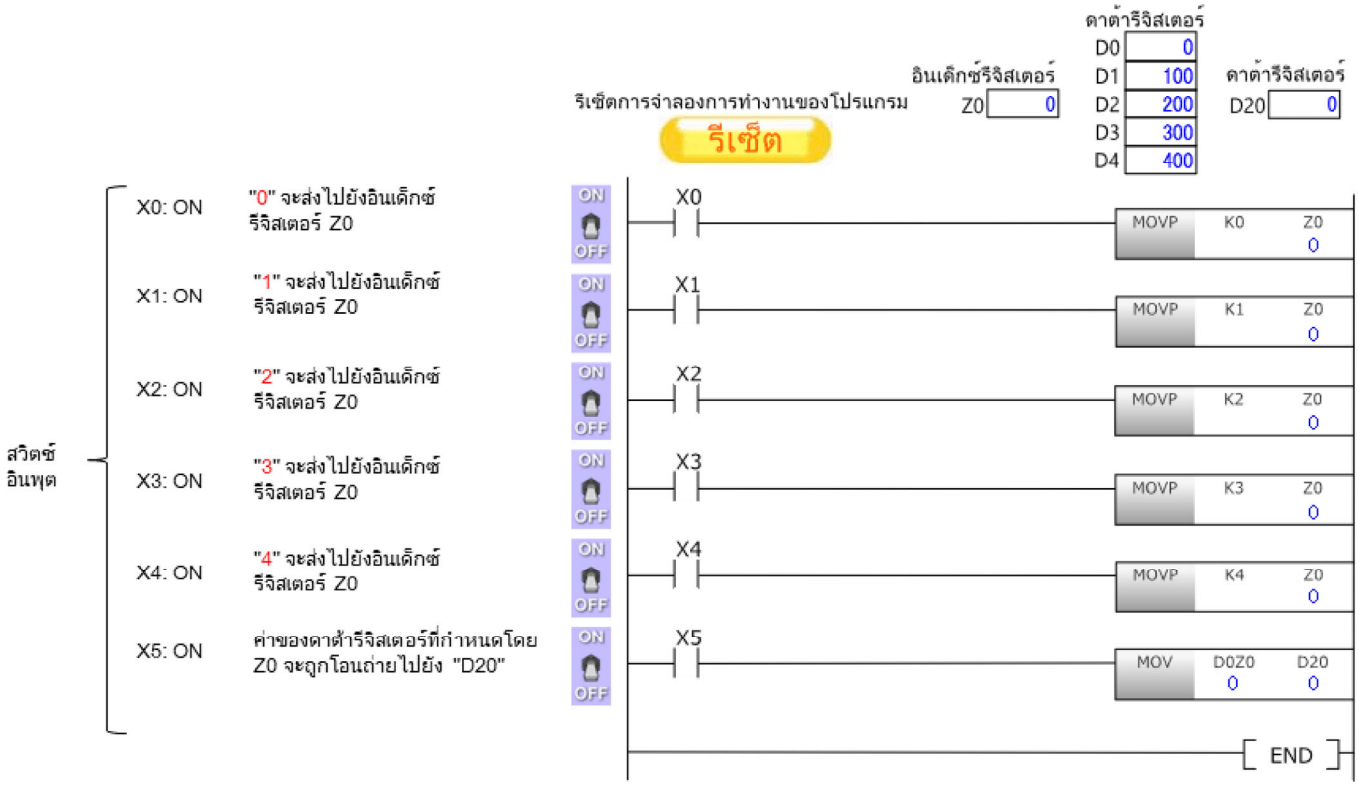

**(2) เมื่อใช้อินเด็กซ์รีจิสเตอร์**




ความซับซ้อนของโปรแกรมลดลง

## 2.5.4 ตัวอย่างโปรแกรมของอินเด็กซ์รีจิสเตอร์

ลักษณะการทำงานของอินเด็กซ์รีจิสเตอร์Z0 สามารถจำลองได้โดยการสั่งงานให้สวิตช์อินพุต X0 ถึง X5 ทำงาน
*K0 ถึง K400 ได้ถูกบันทึกข้อมูลไว้แล้วใน ดาต้ารีจิสเตอร์ D0 ถึง D4
กดสวิตช์อินพุท X0 ถึง X5 เพื่อตรวจสอบค่าที่จัดเก็บอุปกรณ์



# 2.6 การจัดการค่าจำนวนมาก (อาร์เรย์)

การใช้อาร์เรย์ช่วยให้สามารถจัดการค่าจำนวนมากได้โดยใช้ชื่อเลเบิลเดียว
ในตัวอย่างต่อไปนี้ ข้อมูลจำนวนการผลิตในโรงงานผลิตรถยนต์แห่งหนึ่งจะได้รับการจัดเก็บตามปลายทาง

| ปลายทาง | Country A | Country B | Country C |
|---|---|---|---|
| จำนวนการผลิต | 35 หน่วย | 75 หน่วย | 65 หน่วย |

ข้อมูลจำนวนการผลิตจากปลายทางจะถูกกำหนดไปที่เลเบิล
หากไม่ใช้อาร์เรย์ ชื่อเลเบิลจะต้องถูกสร้างสำหรับแต่ละปลายทาง
อย่างไรก็ตาม การใช้อาร์เรย์ จะสามารถกำหนดจำนวนการผลิตสำหรับหลายปลายทางให้กับเลเบิลเดียวและนำไปจัดเก็บไว้ในชื่อเลเบิลเดียวได้

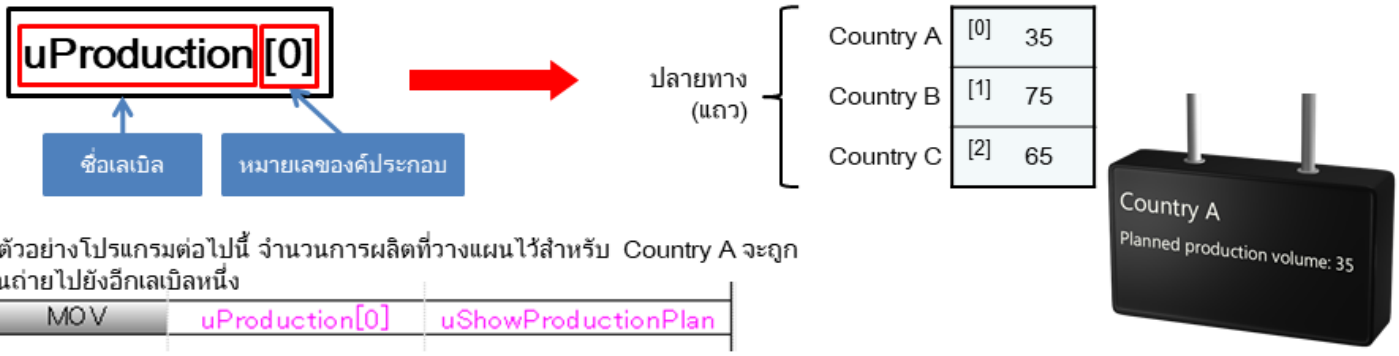
**เมื่อไม่ใช้อาร์เรย์**

uProductionA
uProductionB
uProductionC

**เมื่อใช้อาร์เรย์**

uProduction

การระบุแต่ละเลเบิลภายในอาร์เรย์ สามารถทำได้โดยใช้หมายเลของค์ประกอบ หมายเลของค์ประกอบจะเริ่มจาก [0]



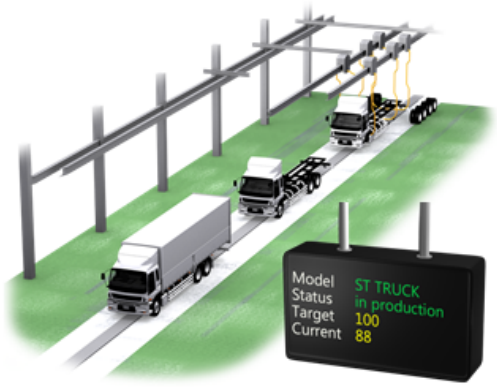

ในตัวอย่างโปรแกรมต่อไปนี้ จำนวนการผลิตที่วางแผนไว้สำหรับ Country A จะถูกโอนถ่ายไปยังอีกเลเบิลหนึ่ง

## 2.7 การจัดการค่าจำนวนมาก (โครงสร้าง)

การใช้โครงสร้างช่วยให้สามารถจัดการเลเบิลที่เกี่ยวข้องกันจำนวนมากได้ โดยใช้เลเบิลในชื่อเดียว
ในตัวอย่างต่อไปนี้ สถานะของสายการผลิตรถยนต์แห่งหนึ่งจะแสดงผลบน Andon (บอร์ดแสดงข้อมูล)

ตารางต่อไปนี้จะแสดงชื่อ ข้อมูล และชนิดข้อมูลของเลเบิล

| รายการ | ชื่อเลเบิล | ค่า | ชนิดข้อมูลของเลเบิล |
|---|---|---|---|
| รุ่น | sModel | 'ST TRUCK' | ชนิดสตริง |
| สถานะการทำงาน | bStatus | 'in production' | ชนิดบิต |
| จำนวนเป้าหมายการผลิตของวันนี้ | uPlanQty | '100' หน่วย | ชนิดจำนวนเต็ม(เวิร์ด ไม่ระบุเครื่องหมาย) |
| จำนวนการผลิตปัจจุบัน | uActualQty | '88' หน่วย | ชนิดจำนวนเต็ม (เวิร์ด ไม่ระบุเครื่องหมาย) |




หากไม่ใช้โครงสร้างสำหรับโรงงานที่มีไลน์การผลิตจำนวนมาก จะต้องเปลี่ยนชื่อเลเบิลให้กับแต่ละสายการผลิต
ต่อไปนี้คือตัวอย่างของชื่อเลเบิล ที่ถูกเพิ่มชื่อไลน์การผลิต

**ไลน์การผลิตแรก**

s1stLineModel
b1stLineStatus
u1stLinePlanQty
u1stLineActualQty

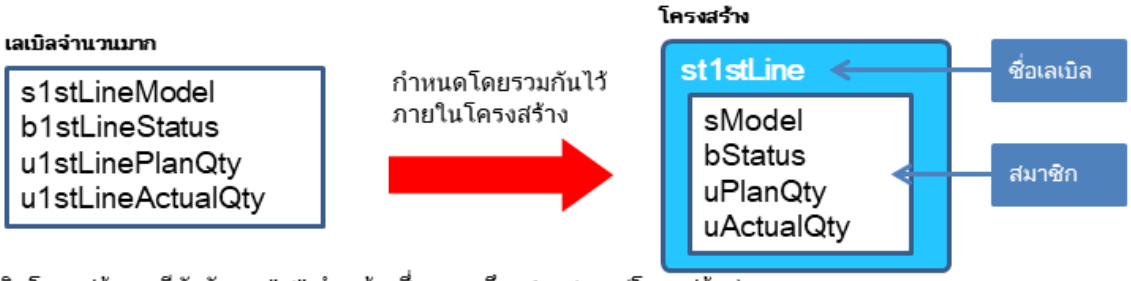
**ไลน์การผลิตที่สอง**

s2ndLineModel
b2ndLineStatus
u2ndLinePlanQty
u2ndLineActualQty

เมื่อจำนวนไลน์การผลิตเพิ่มขึ้น จำนวนของเลเบิลที่ต้องจัดการก็จะเพิ่มขึ้น ผลก็คือ โปรแกรมจะยาวขึ้นและอ่านยากขึ้น

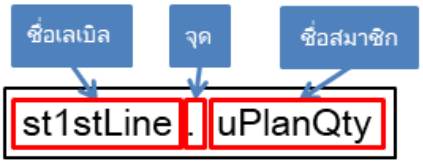
## 2.7 การจัดการค่าจำนวนมาก (โครงสร้าง)

การใช้โครงสร้างช่วยให้สามารถใช้ชื่อเลเบิลเดียว เป็นตัวแทนเลเบิลจำนวนมากที่เกี่ยวข้องกับไลน์การผลิตหนึ่ง
ดังนั้นจึงสามารถใช้โครงสร้างเพื่อจัดระเบียบ จัดเก็บ และจัดการเงื่อนไขและข้อมูลจำเพาะเกี่ยวกับวัตถุหรือสิ่งของทางกายภาพ เช่น อุปกรณ์ เครื่องมือ และชิ้นงานจำนวนมากรวมกันได้

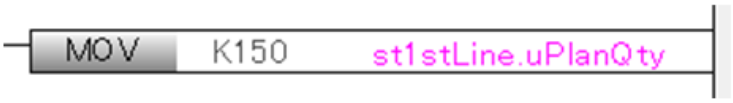



เลเบิลโครงสร้างจะมีตัวอักษร "st" นำหน้า ซึ่งหมายถึง **st**ructure (โครงสร้าง)
แต่ละเลเบิลที่กำหนดโดยโครงสร้างจะเรียกว่าสมาชิก ชนิดข้อมูลของสมาชิกแต่ละรายการสามารถแตกต่างกันได้

ในการกำหนดสมาชิกแต่ละรายการในโครงสร้าง ให้เพิ่มชื่อสมาชิกหลังชื่อเลเบิลโครงสร้าง โดยใช้จุด (.) เป็นอักขระตัวคั่น




ในตัวอย่างโปรแกรมต่อไปนี้ ค่าคงที่จะถูกกำหนดให้กับสมาชิกของเลเบิลชนิดโครงสร้างสำหรับไลน์การผลิตแรก



## 2.8 การคงค่าสถานะอุปกรณ์ (แลตช์)

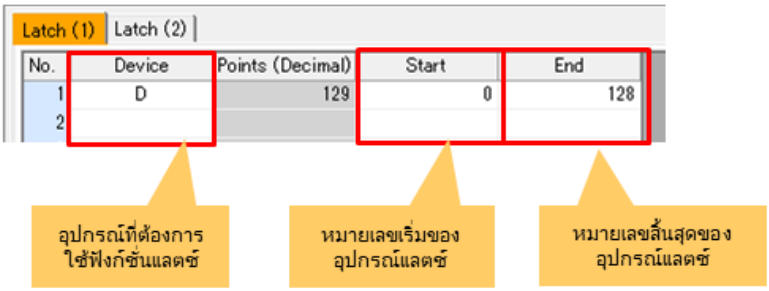
หมวดนี้จะอธิบายฟังก์ชันแลตช์ ซึ่งเป็นอุปกรณ์ที่จะคงค่าหรือสถานะไว้ เมื่อโมดูล CPU หยุดการทำงาน แม้ว่าเมื่อระบบไฟฟ้าล้มเหลว เกินค่าเวลาการล้มเหลวที่ยอมรับได้ PLC จะสามารถรีสตาร์ทการควบคุมลำดับอีกครั้ง โดยใช้ข้อมูลที่ถูกหยุดไว้ในขณะที่หยุดการดำเนินการ

หากไม่ใช้ฟังก์ชันแลตช์ ค่าอุปกรณ์จะถูกล้างและรีเซ็ตให้เป็นค่าเริ่มต้น (เป็น ปิด สำหรับอุปกรณ์บิต และเป็น "0" สำหรับเวิร์ดดีไวซ์) ในเหตุการณ์ต่อไปนี้:

- ปิดเครื่อง
- รีเซ็ตโดยใช้สวิตช์ RUN (ทำงาน)/STOP (หยุด)/RESET (รีเซ็ต)
- ระบบไฟฟ้าล้มเหลว เกินค่าเวลาการล้มเหลวที่ยอมรับได้

## 2.8.1 การตั้งค่าแลตช์ของอุปกรณ์

ตั้งค่าช่วงแลตช์ในหน้าต่าง "Device Setting" (การตั้งค่าอุปกรณ์) ของ CPU พารามิเตอร์
ต่อไปนี้คือตัวอย่างการตั้งค่าแลตช์ของดาต้ารีจิสเตอร์ D0 ถึง D128



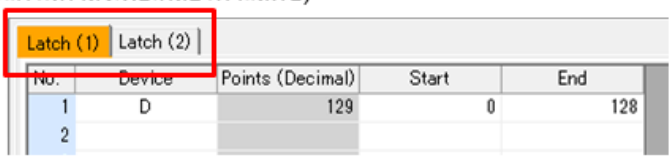
## 2.8.2 ชนิดของแลตช์และวิธีเคลียร์ค่า

แลตช์มีสองชนิด (แลตช์ (1) และแลตช์ (2)) ขึ้นอยู่กับวิธีเคลียร์ค่า

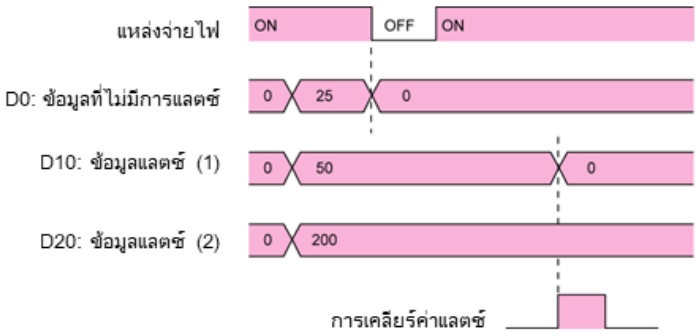
- Latch (1) (แลตช์ (1))

  ข้อมูลที่แลตช์สามารถเคลียร์ได้โดยใช้ฟังก์ชัน CPU memory operation ในซอฟต์แวร์ GX Works3
  ใช้แลตช์ 1 เมื่อต้องการเคลียร์ข้อมูลแลตช์ที่ไซต์การติดตั้ง (สามารถใช้ซอฟต์แวร์ GX Works3 เพื่อเคลียร์ค่าแลตช์ได้เลย)
- Latch (2) (แลตช์ (2))

  ข้อมูลที่แลตช์สามารถล้างได้โดยใช้คำสั่งในโปรแกรม
  ใช้แลตช์ 2 เมื่อไม่ได้ล้างข้อมูลแลตช์ที่ไซต์การติดตั้ง (ต้องเขียนโปรแกรมคำสั่งเพื่อเคลียร์ค่าแลตช์)



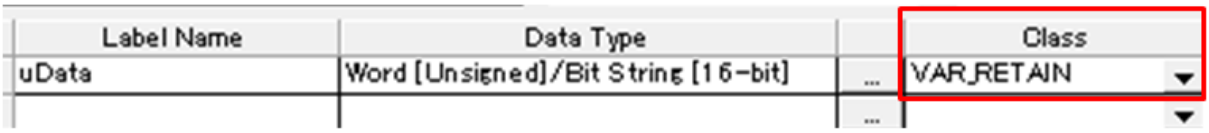

ต่อไปนี้คือแผนผังจังหวะการทำงาน (Timing chart) ของการเคลียร์ค่าแลตช์




*ดำเนินการฟังก์ชันนี้โดยเลือก [Online] (ออนไลน์) → [CPU Memory Operation] (การทำงานของหน่วยความจำ CPU) จากเมนูใน GX Works3

## 2.8.3 การตั้งค่าแลตช์ให้กับเลเบิล

ในการตั้งค่าแลตช์ให้กับเลเบิล ให้เลือกชื่อคลาสที่มี "RETAIN" ในหน้าต่างการตั้งค่าเลเบิล
สำหรับเลเบิลที่เป็นโลคอล ให้เลือก "VAR_RETAIN"



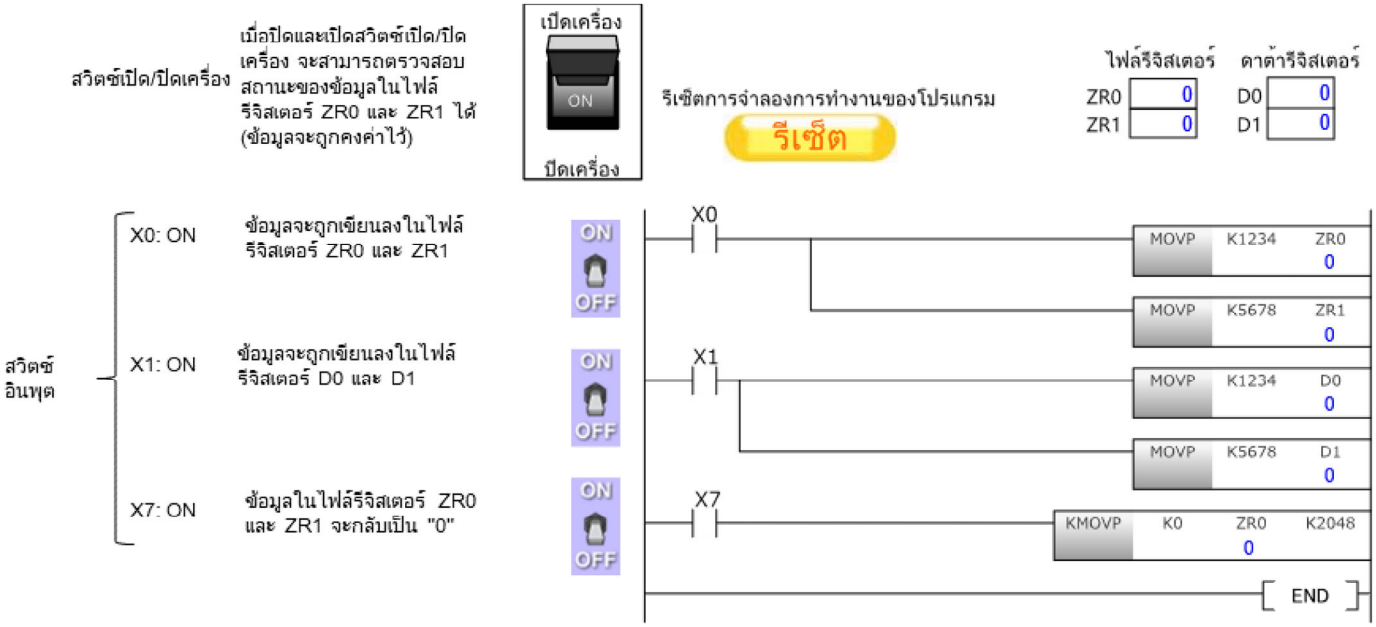
| Label Name | Data Type | | Class |
|---|---|---|---|
| uData | Word [Unsigned]/Bit String [16-bit] | ... | VAR_RETAIN |
| | | ... | |

## 2.9 การคงสถานะอุปกรณ์ (ไฟล์รีจิสเตอร์)

- ไฟล์รีจิสเตอร์เป็นอุปกรณ์ประเภทเวิร์ด ที่ใช้ขยายจำนวนดาต้ารีจิสเตอร์(D)
- เมื่อเปรียบเทียบกับดาต้ารีจิสเตอร์ ไฟล์รีจิสเตอร์สามารถจัดการข้อมูลได้จำนวนมากกว่า
- ไฟล์รีจิสเตอร์จะจัดเก็บไว้ในหน่วยความจำข้อมูลของโมดูล CPU หรือในการ์ดหน่วยความจำ
- ข้อมูลที่จัดเก็บไว้ในไฟล์รีจิสเตอร์จะถูกคงไว้ แม้ปิดระบบหรือเมื่อแม้แต่โมดูล CPU จะถูกรีเซ็ต
  ไฟล์รีจิสเตอร์มีประโยชน์สำหรับการเก็บข้อมูลที่กำหนดไว้ล่วงหน้า เช่น ค่ามาตรฐานสำหรับการผลิตต่างๆ
- ในการเคลียร์ข้อมูล ให้เขียนค่า 0 สำหรับช่วงไฟล์รีจิสเตอร์นั้นๆ หรือเคลียร์หน่วยความจำโดยใช้ซอฟต์แวร์ GX Works3
- สัญลักษณ์ของอุปกรณ์คือ "ZR"

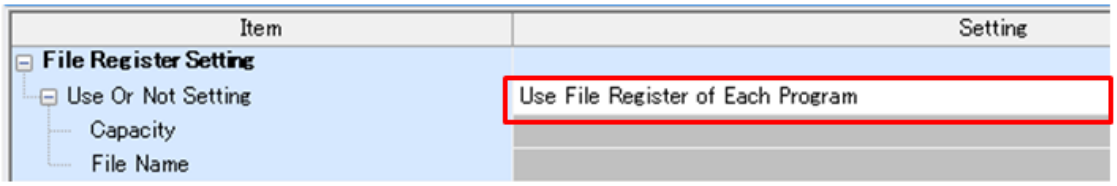
## 2.9.1 การทำงานของโปรแกรมแลดเดอร์

ลักษณะการทำงานของไฟล์รีจิสเตอร์สามารถจำลองได้โดยการเปิดสวิตช์อินพุทและสวิตช์เปิด/ปิดเครื่อง

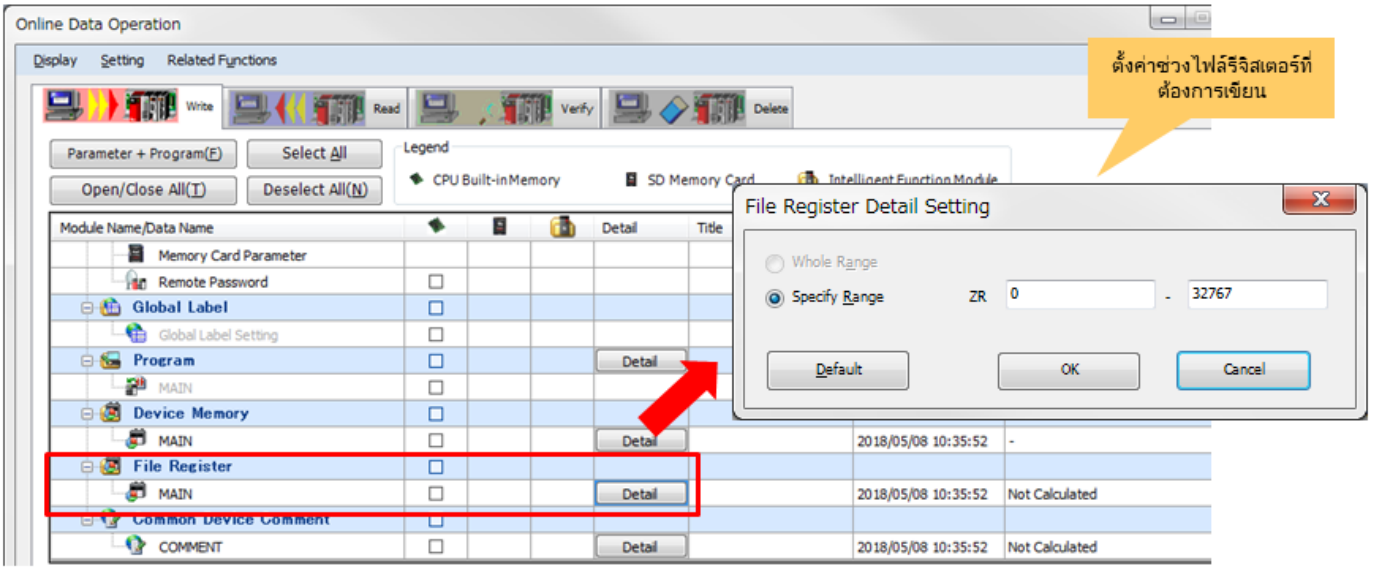


## 2.9.2 การตั้งค่าไฟล์รีจิสเตอร์

หมวดนี้จะอธิบายการตั้งค่าสำหรับการกำหนดให้ไฟล์รีจิสเตอร์ ที่เป็นโลเคิลเป็นปลายทางการเก็บข้อมูลของแต่ละโปรแกรม
เลือก "File Register Setting" (การตั้งค่าไฟล์รีจิสเตอร์) ใน CPU พารามิเตอร์ แล้วเลือก "Use File Register of Each Program"




ในการเขียนข้อมูลให้กับPLC ต้องเขียนการตั้งค่าไฟล์รีจิสเตอร์ให้กับแต่ละโปรแกรม



## 2.10 การใช้อุปกรณ์ที่มีฟังก์ชันและการทำงานที่ได้กำหนดไว้แล้ว

รีเลย์พิเศษและรีจิสเตอร์พิเศษที่ใช้ในโมดูล CPU จะมีฟังก์ชันและการทำงานที่ได้กำหนดไว้แล้ว
รีเลย์พิเศษ (SM) คือรีเลย์ภายในที่ใช้สำหรับข้อมูลประเภทบิต (เปิด/ปิด) และรีจิสเตอร์พิเศษ (SD) คือรีจิสเตอร์ภายในที่ใช้สำหรับข้อมูลประเภทเวิร์ด

### 2.10.1 ชนิดของรีเลย์พิเศษและรีจิสเตอร์พิเศษ

รีเลย์พิเศษและรีจิสเตอร์พิเศษสามารถแบ่งประเภทได้ตามชนิดของข้อมูล ชนิดหลักๆ มีในรายการด้านล่าง
ในโปรแกรม จะใช้รีเลย์พิเศษและรีจิสเตอร์พิเศษเป็นเงื่อนไขการกำหนดสำหรับการควบคุม
และยังใช้ในการตรวจสอบการทำงานอีกด้วย โดยสามารถสั่งงานได้จาก device monitor ในซอฟต์แวร์ GX Works3

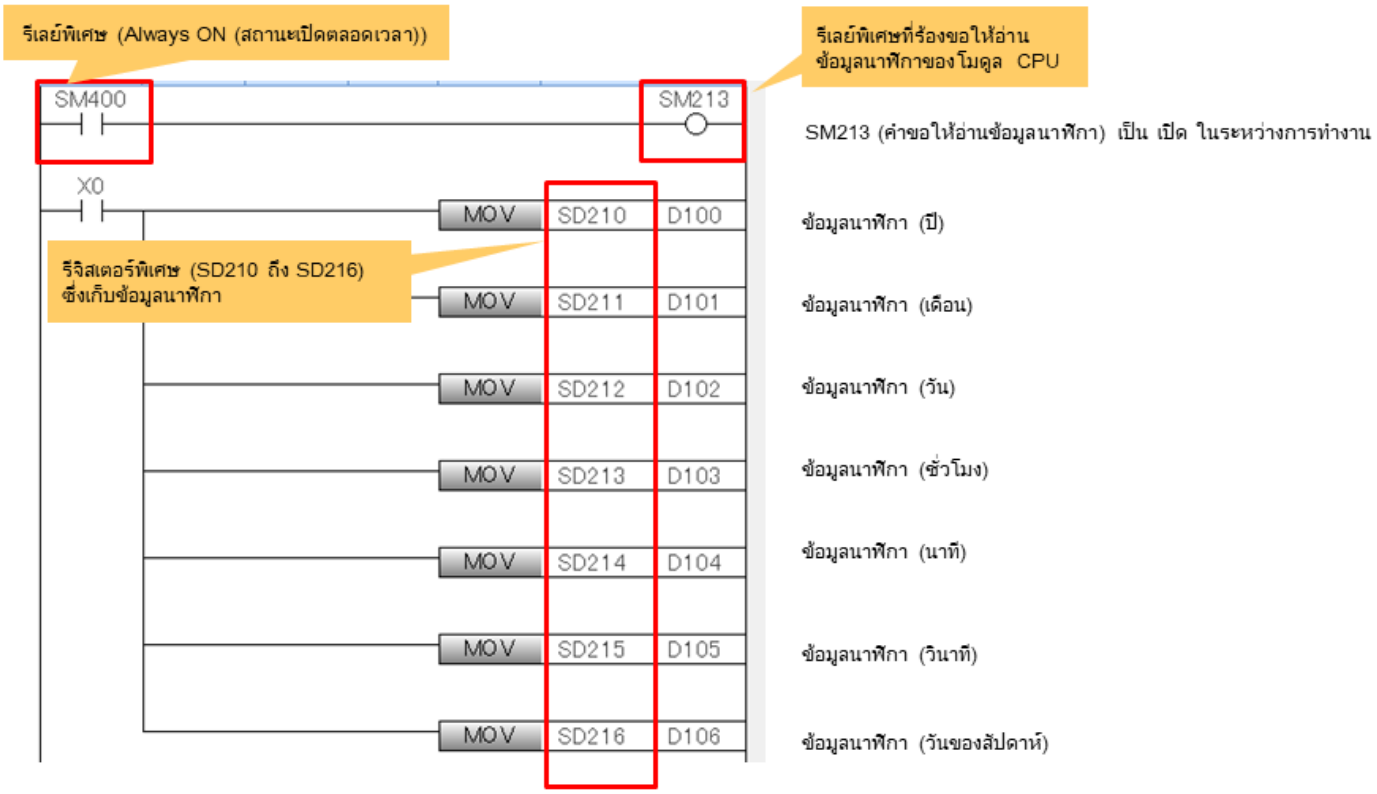
| ข้อมูลการวินิจฉัย (Diagnostic information) |
| --- |
| ผลการวินิจฉัยโมดูล CPU |
| การวินิจฉัยความผิดพลาดและรหัสความผิดพลาด |

| ข้อมูลระบบ (System information) |
| --- |
| ข้อมูลของโมดูล CPU |
| สถานะการทำงานของโมดูล CPU ข้อมูลนาฬิกา และข้อมูลอื่นๆ |

| นาฬิกา (System clock) |
| --- |
| สัญญาณนาฬิกาและค่าการนับที่ใช้เป็นองค์ประกอบจังหวะพื้นฐาน |
| สัญญาณนาฬิกาต่างๆ (ON (เปิด) ตลอดเวลา), ON/OFF (เปิด/ปิด) ตามช่วงที่กำหนด และสัญญาณอื่นๆ) |

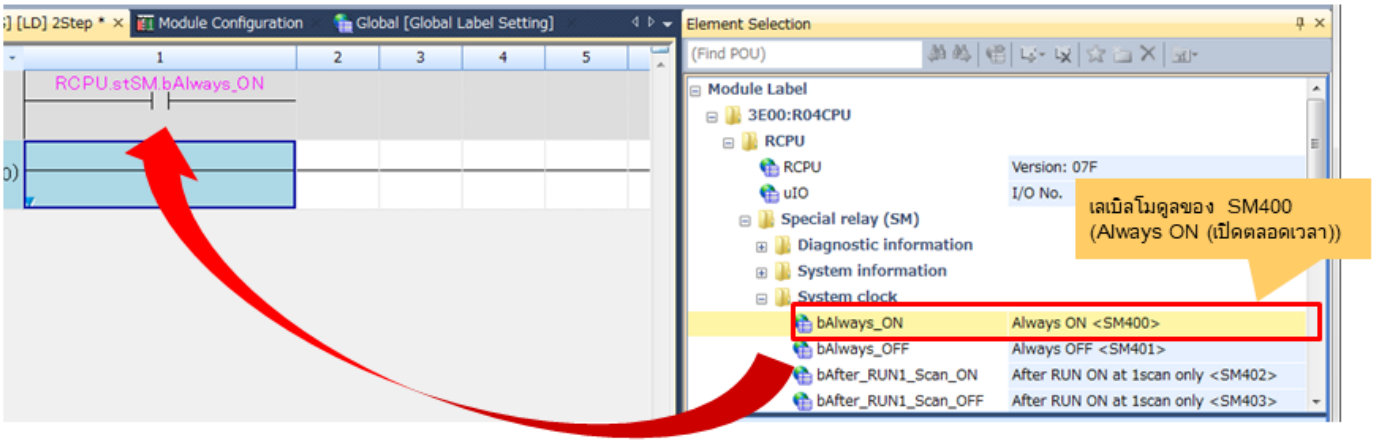
## 2.10.2 ตัวอย่างโปรแกรมของรีเลย์พิเศษและรีจิสเตอร์พิเศษ

ต่อไปนี้เป็นตัวอย่างโปรแกรมสำหรับการอ่านข้อมูลนาฬิกาของโมดูล CPU โดยใช้รีเลย์พิเศษและรีจิสเตอร์พิเศษ



## 2.10.3 การใช้เลเบิลสำหรับรีเลย์พิเศษและรีจิสเตอร์พิเศษ

รีเลย์พิเศษและรีจิสเตอร์พิเศษได้จัดเตรียมไว้เป็นเลเบิลโมดูลในโมดูล CPU ซึ่งสามารถใช้ได้อย่างง่ายดาย เพียงเลือกและวางชื่อเลเบิลที่เกี่ยวข้อง โดยไม่ต้องตรวจสอบหมายเลขอุปกรณ์ในคู่มือการใช้งาน



## 2.11 การคำนวณด้วยจำนวนจริง

## 2.11.1 การใช้งานจำนวนจริง

- "จำนวนจริง" คือค่าจำนวนที่มีจุดทศนิยม
- โดยปกติแล้วจะใช้จำนวนเต็มในโปรแกรมควบคุม อย่างไรก็ตาม จำนวนจริงจะใช้ในโปรแกรมสำหรับการควบคุมการทำงานขั้นสูง เช่น ฟังก์ชันตรีโกณมิติ และการทำงานที่เป็นเอกซ์โพเนนเชียล เนื่องจากในโปรแกรมดังกล่าวจำเป็นต้องจัดการค่าจำนวนที่มีจุดทศนิยม
- ข้อมูลทางตัวเลขของจำนวนจริงที่จัดการในโมดูล CPU จะเรียกว่า "floating-point data"

หมายเหตุ:

- จำนวนจริงหนึ่งจำนวนจะต้องใช้เวิร์ดดีไวซ์สองเวิร์ดต่อเนื่องกันเสมอ (กินพื้นที่หน่วยความจำ 32 บิต) โดยไม่คำนึงถึงค่า*

- ในโปรแกรมควบคุมคำสั่งการทำงานเฉพาะด้าน (เช่น การบวก การลบ การคูณ การหาร และฟังก์ชันพิเศษต่างๆ) สำหรับจัดการจำนวนจริง การจัดเตรียมคำสั่งแปลงระหว่างจำนวนเต็มและจำนวนจริงไว้แล้วเช่นกัน

*ถ้าต้องทำการคำนวณให้ถูกต้องมากยิ่งขึ้น โดยมีจำนวนจุดที่มากขึ้น ให้ใช้เวิร์ดดีไวซ์ 4 เวิร์ด

- จำนวนจริงที่ใช้เวิร์ดดีไวซ์ 2เวิร์ด จะเรียกว่า single-precision real numbers
- จำนวนจริงที่ใช้เวิร์ดดีไวซ์ 4เวิร์ด จะเรียกว่า double-precision real numbers

หลักสูตรนี้จะมุ่งเน้นเกียวกับ single-precision real numbers

## 2.11.2 การระบุข้อมูลสำหรับจำนวนจริง

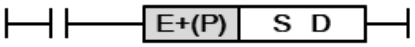
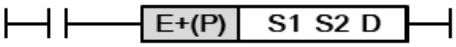
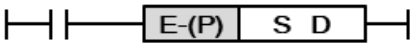
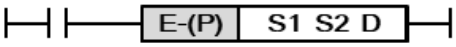
"E" ใช้เพื่อแสดงค่าจำนวนจริง

**การแสดงค่าคงที่ด้วยจำนวนจริง**

ในการเขียนค่าคงที่ ให้เริ่มด้วยตัว "E"

| | |
|---|---|
| การแสดงแบบปกติ | เขียนค่าจำนวน<br>(ตัวอย่าง) เขียน 10.2345 เป็น "E10.2345" |
| การแสดงแบบเอกซ์โพเนนเชียล | เขียนค่าจำนวนเป็น "(ค่าจำนวน) × $10^{n}$"<br>(ตัวอย่าง) เขียน 1234.0 เป็น "E1.234+3" |

## 2.11.3 คำสั่งการดำเนินการ (การบวกและการลบ)

| สัญลักษณ์ของคำสั่ง | ตัวอย่างแลดเดอร์ | |
|---|---|---|
| E+ (การบวก single-precision real numbers) |  เป็นการดำเนินการกับจำนวนจริง "D + S = D" |  เป็นการดำเนินการกับจำนวนจริง "S1 + S2 = D" |
| E- (การลบ single-precision real numbers) |  เป็นการดำเนินการกับจำนวนจริง "D - S = D" |  เป็นการดำเนินการกับจำนวนจริง "S1 - S2 = D" |

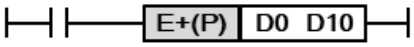
S (ต้นทาง): ข้อมูลก่อนการดำเนินการ (ค่าคงที่ หมายเลขอุปกรณ์)
D (ปลายทาง): ปลายทางของข้อมูลหลังการดำเนินการ (หมายเลขอุปกรณ์)
P: คำสั่งที่ต้องดำเนินการที่ขอบขาขึ้น (จากปิดเป็นเปิด)
S1 และ S2: ข้อมูลสองรายการที่จะดำเนินการ

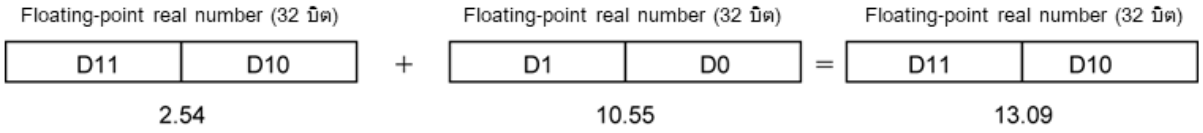
**หมายเหตุ:**

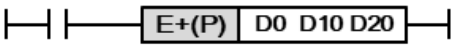
ไม่สามารถกำหนดจำนวนเต็มและจำนวนจริงผสมกันในการดำเนินการเดียวกันได้
สำหรับ single-precision real number S, S1 และ S2 ค่าตัวแปลจะต้องเป็น single-precision real number
Single-precision real numbers จะบันทึกไว้ใน D

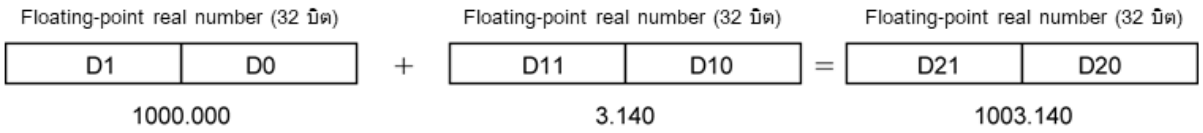
## 2.11.3 คำสั่งการดำเนินการ (การบวกและการลบ)

**ตัวอย่างการใช้งานของคำสั่งบวก**



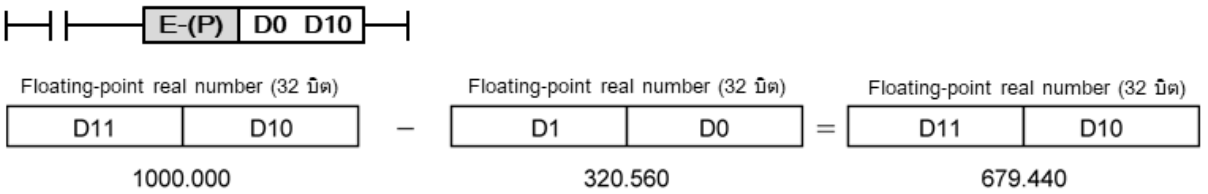



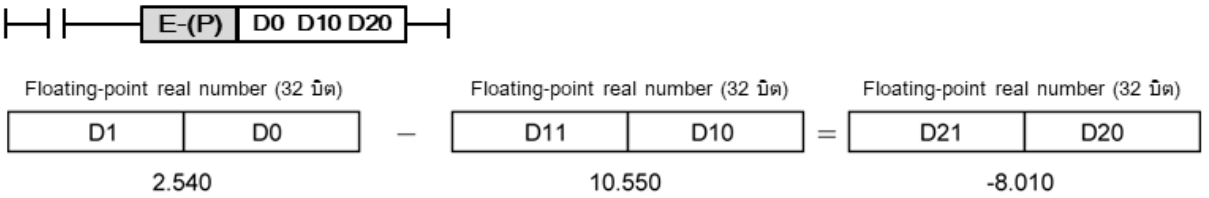



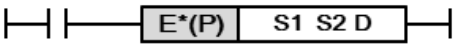
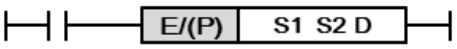
## 2.11.3 คำสั่งการดำเนินการ (การบวกและการลบ)

**ตัวอย่างการใช้งานของคำสั่งลบ**



## 2.11.4 คำสั่งการดำเนินการ (การคูณและการหาร)

| สัญลักษณ์ของคำสั่ง | ตัวอย่างแลดเดอร์ |
|---|---|
| E* (การคูณ single-precision real numbers) |  <br>เป็นการดำเนินการกับจำนวนจริง "S1 * S2 = D" |
| E/ (การหาร single-precision real numbers) |  <br>เป็นการดำเนินการกับจำนวนจริง "S1 / S2 = D" |

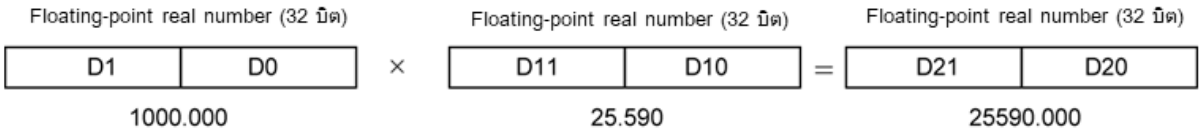
S1, S2 (ต้นทาง): ข้อมูลสองรายการที่จะดำเนินการ
D (ปลายทาง): ปลายทางของข้อมูลหลังการดำเนินการ (หมายเลขอุปกรณ์)
P: คำสั่งที่ต้องดำเนินการที่ขอบขาขึ้น (จากปิดเป็นเปิด)

**หมายเหตุ:**

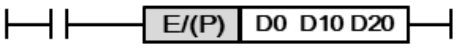
สำหรับsingle-precision real numbers S1 และ S2 ค่าตัวแปลจะต้องเป็น single-precision real number
Single-precision real numbers จะบันทึกไว้ใน D

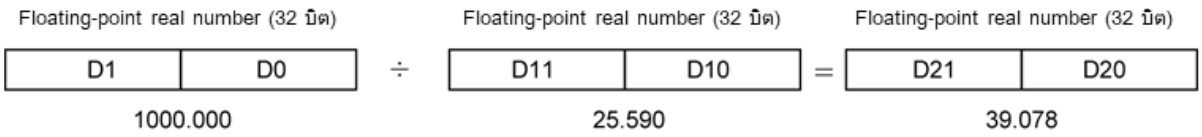
## 2.11.4 คำสั่งการดำเนินการ (การคูณและการหาร)

**ตัวอย่างการใช้งานของคำสั่งคูณ**


**ตัวอย่างการใช้งานของคำสั่งหาร**



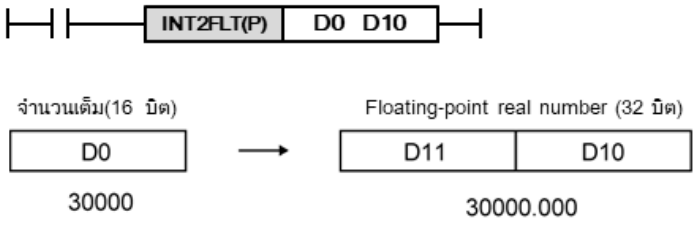
## 2.11.5 คำสั่งแปลงระหว่างจำนวนเต็มและจำนวนจริง

| สัญลักษณ์ของคำสั่ง | ตัวอย่างแลดเดอร์ | |
|---|---|---|
| INT2FLT (การแปลงจำนวนเต็มเป็นจำนวนจริงชนิด single-precision real number) | จำนวนเต็ม(16 บิต) จะถูกแปลงเป็นจำนวนจริง (32 บิต)<br>  <br>S (16 บิต) จะถูกแปลงและบันทึกไว้ใน D | จำนวนเต็ม (32 บิต) จะถูกแปลงเป็นจำนวนจริง (32 บิต)<br>  <br>S (32 บิต) จะถูกแปลงและบันทึกไว้ใน D |
| FLT2INT (การแปลงจำนวนจริงชนิด single-precision real numberเป็นจำนวนเต็ม) | จำนวนจริง (32 บิต) จะถูกแปลงเป็นจำนวนเต็ม (16 บิต)<br>  <br>S จะถูกแปลงและบันทึกไว้ใน D (16 บิต) | จำนวนจริง (32 บิต) จะถูกแปลงเป็นจำนวนเต็ม(32 บิต)<br>  <br>S จะถูกแปลงและบันทึกไว้ใน D (32 บิต) |

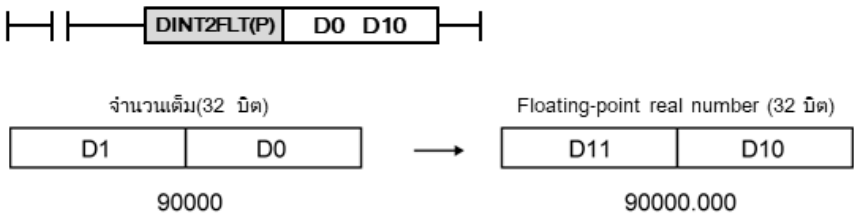
S (ต้นทาง): ข้อมูลก่อนการดำเนินการ (ค่าคงที่, หมายเลขอุปกรณ์)
D (ปลายทาง): ปลายทางของข้อมูลหลังการดำเนินการ (หมายเลขอุปกรณ์)

## 2.11.5 คำสั่งแปลงระหว่างจำนวนเต็มและจำนวนจริง

**ตัวอย่างการใช้งานคำสั่งแปลงจำนวนเต็ม(16 บิต) เป็นจำนวนจริง (32 บิต)**

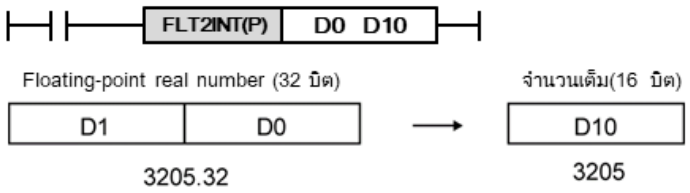



**ตัวอย่างการใช้งานคำสั่งแปลงจำนวนเต็ม (32 บิต) เป็นจำนวนจริง (32 บิต)**

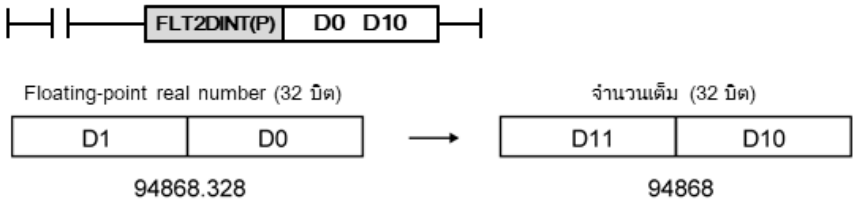


## 2.11.5 คำสั่งแปลงระหว่างจำนวนเต็มและจำนวนจริง

**ตัวอย่างการใช้งานคำสั่งแปลงจำนวนจริง (32 บิต) เป็นจำนวนเต็ม(16 บิต)**




**ตัวอย่างการใช้งานคำสั่งแปลงจำนวนจริง (32 บิต) เป็นจำนวนเต็ม(32 บิต)**



## 2.11.6 การใช้เลเบิลแสดงจำนวนจริง

ในการใช้เลเบิลสำหรับจำนวนจริง ให้ตั้งค่าชนิดข้อมูลเป็น "Single Precision" หรือ "Double Precision" ในหน้าต่างสำหรับการตั้งค่าเลเบิล

| Label Name | Data Type |
|---|---|
| eData | FLOAT [Single Precision] |
| leData | FLOAT [Double Precision] |

## 2.12 สรุปสำหรับบทนี้

ในบทนี้คุณได้เรียนรู้เกี่ยวกับ:

- ข้อมูลบิตของเวิร์ดดีไวซ์
- ข้อมูลการทำงานของขอบขาขึ้น และขอบขาลงของหน้าสัมผัส
- รีเทนทีฟไทม์เมอร์
- ข้อมูลหน่วยวัดของไทม์เมอร์
- อินเด็กซ์รีจิสเตอร์
- อาร์เรย์
- โครงสร้าง
- แลตช์
- ไฟล์รีจิสเตอร์
- รีเลย์พิเศษและรีจิสเตอร์พิเศษ
- การคำนวณด้วยจำนวนจริง

ประเด็นที่สำคัญ

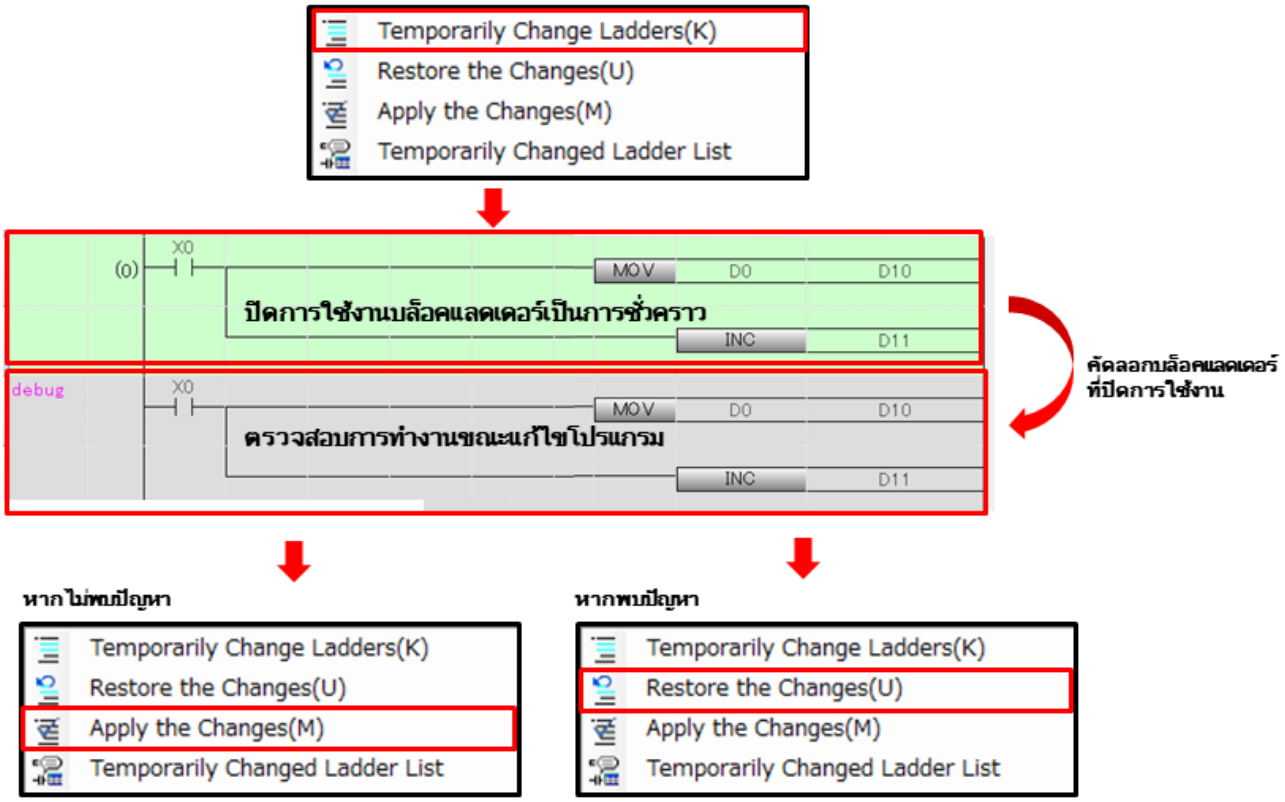
| | |
|---|---|
| รีเทนทีฟไทม์เมอร์ | เวลาที่จับเวลาได้จะถูกคงไว้แม้คอยล์ไม่ทำงาน และการจับเวลาจะกลับมาทำงานเมื่อคอยล์ทำงานอีกครั้ง |
| หน่วยการจับเวลาของไทม์เมอร์ | หน่วยการจับเวลาของไทม์เมอร์สามารถเปลี่ยนแปลงได้ในพารามิเตอร์ |
| อินเด็กซ์รีจิสเตอร์ | อุปกรณ์จำนวนมากสามารถอธิบายเป็นชุดได้ |
| อาร์เรย์ | สามารถจัดการค่าจำนวนมากได้โดยใช้ชื่อเลเบิลเดียว |
| โครงสร้าง | สามารถจัดการเลเบิลที่เกี่ยวข้องจำนวนมากได้ด้วยชื่อเลเบิลเดียว |
| แลตช์ | • ค่าของอุปกรณ์ประเภทแลตช์จะถูกคงไว้เมื่อโมดูล CPU หยุดการทำงาน<br>• ค่าของอุปกรณ์ประเภทแลตช์สามารถรีเซ็ทได้โดยการใช้งานฟังก์ชั่น CPU memory operation |
| ไฟล์รีจิสเตอร์ | • เมื่อเปรียบเทียบกับดาต้ารีจิสเตอร์ ไฟล์รีจิสเตอร์สามารถจัดการข้อมูลได้เป็นจำนวนมากกว่า<br>• ค่าภายในอุปกรณ์จะถูกคงค่าไว้เมื่อโมดูล CPU หยุดการทำงาน<br>• ค่าอุปกรณ์สามารถรีเซ็ทค่าได้โดยการใช้งานฟังก์ชั่น CPU memory operation หรือการเขียนค่า 0 ให้กับ ช่วงอุปกรณ์นั้นๆ |
| รีเลย์พิเศษและรีจิสเตอร์พิเศษ | สถานะภายในของโมดูล CPU เช่น ข้อมูลการวินิจฉัยและข้อมูลระบบ ได้บันทึกไว้ในอุปกรณ์เหล่านี้แล้ว |
| จำนวนจริง | • ใช้เวิร์ดดีไวซ์อย่างน้อยสองเวิร์ด (32 บิต)<br>• มีการเตรียมคำสั่งการคำเนินการไว้ให้แล้ว<br>• ไม่สามารถใช้จำนวนเต็มและจำนวนจริงผสมกันในการดำเนินการเดียวกันได้ |

# บทที่ 3 การแก้จุดบกพร่องอย่างมีประสิทธิภาพ

บทนี้จะอธิบายเกี่ยวกับฟังก์ชันของ GX Works3 สำหรับการแก้จุดบกพร่องอย่างมีประสิทธิภาพ

## 3.1 การเปลี่ยนช่วงของโปรแกรมเป็นการชั่วคราว

เมื่อแก้ไขโปรแกรมเป็นช่วงกว้างเพื่อการแก้จุดบกพร่อง เป็นการยากมากที่จะเลิกทำการเปลี่ยนแปลงทั้งหมด
เมื่อปิดการใช้งานบล็อคแลดเดอร์ที่ต้องการเป็นการชั่วคราว ผู้ใช้จะสามารถใช้สำเนาของโปรแกรมสำหรับการแก้จุดบกพร่องโดยไม่ต้องเปลี่ยนแปลงต้นฉบับ
(ฟังก์ชัน Temporarily change ladders (เปลี่ยนแลดเดอร์ชั่วคราว))
หลังเปลี่ยนบล็อคแลดเดอร์และตรวจสอบการทำงานด้วยฟังก์ชันนี้ การเปลี่ยนแปลงจะถูกนำไปใช้หากไม่พบปัญหา และการเปลี่ยนจะถูกกู้คืนหากพบปัญหา



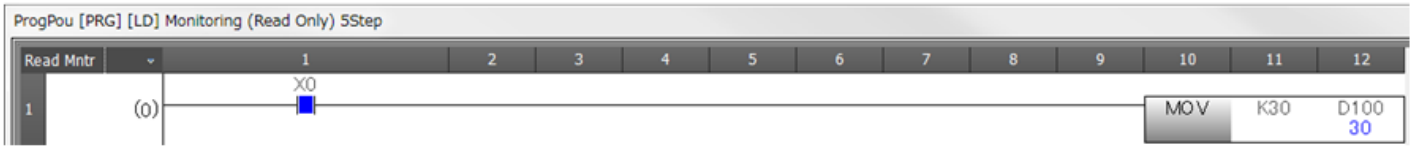

**หากไม่พบปัญหา**




**หากพบปัญหา**

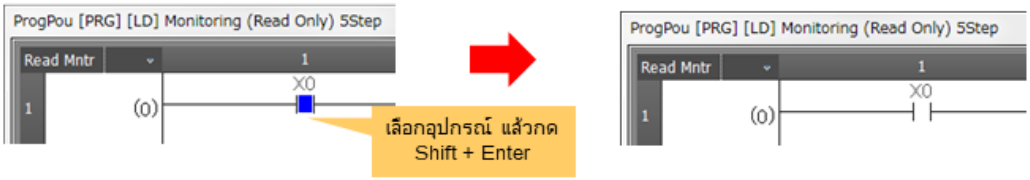
## 3.2 การตรวจสอบการทำงานขณะอุปกรณ์มีการเปลี่ยนแปลง

ขณะโปรแกรมทำงาน จะสามารถแสดงสถานะการทำงานของอุปกรณ์บิตและค่าที่บันทึกไว้ในเวิร์ดดีไวซ์ในหน้าต่างโปรแกรมได้ (ฟังก์ชัน Monitor)
ฟังก์ชันการตรวจสอบช่วยให้ผู้ใช้สามารถตรวจสอบสถานะการทำงานของโปรแกรมได้

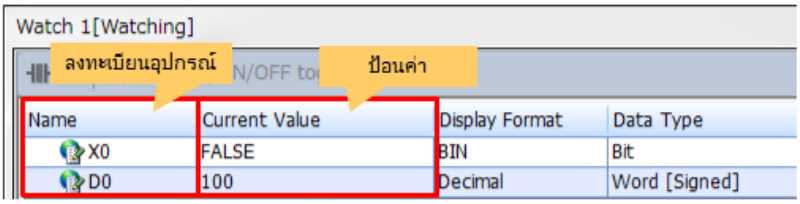



สามารถบังคับให้เปลี่ยนแปลงค่าปัจจุบันของอุปกรณ์ในระหว่างการตรวจสอบได้ (การเปลี่ยนค่าปัจจุบัน)
ฟังก์ชันเปลี่ยนค่าปัจจุบันช่วยให้สามารถทำการเปลี่ยนแปลงได้โดยที่ไม่ต้องแก้ไขโปรแกรมทั้งหมดหรือทำงานบนระบบจริงได้

สถานะของอุปกรณ์ประเภทบิตสามารถสลับได้ในหน้าต่างโปรแกรม

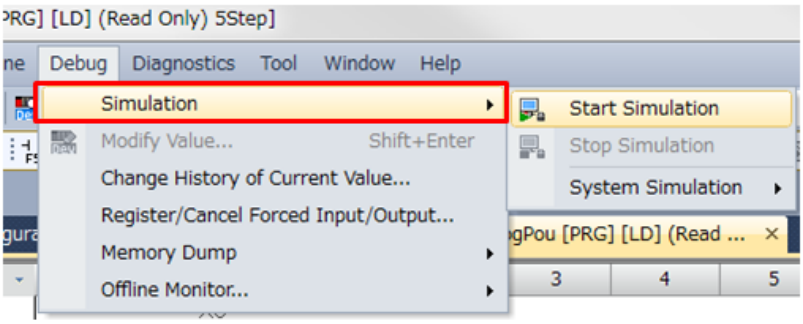



การใช้หน้าต่าง "Watch" ช่วยให้สามารถลงทะเบียนเวิร์ดดีไวซ์ที่ต้องการตรวจสอบได้ และสามารถเปลี่ยนแปลงค่าปัจจุบันได้



## 3.3 การจำลองการทำงานของโปรแกรม

หากโปรแกรมที่สร้างขึ้นใหม่เพื่อทำงานบนระบบจริง อาจเกิดความผิดพลาดที่ไม่คาดหมาย
สามารถจำลองการทำงานของโปรแกรมได้ โดยที่ไม่ต้องใช้ PLC จริง (ฟังก์ชัน Simulation)




ฟังก์ชัน Simulation ใช้ในการจำลอง ช่วยให้สามารถตรวจสอบการทำงานของโปรแกรม ได้ราวกับว่าโปรแกรมกำลังทำงานอยู่บนตัว PLC จริง

## 3.4 สรุปสำหรับบทนี้

ในบทนี้คุณได้เรียนรู้เกี่ยวกับ:

- การเปลี่ยนแปลงบล็อคแลดเดอร์เป็นการชั่วคราว
- การตรวจสอบโปรแกรมและการเปลี่ยนแปลงค่าปัจจุบัน
- การจำลองโปรแกรม

ประเด็นที่สำคัญ

| | |
|---|---|
| การเปลี่ยนแปลงบล็อคแลดเดอร์เป็นการชั่วคราว | บล็อคแลดเดอร์สามารถปิดการใช้งานเป็นการชั่วคราว และสามารถใช้สำเนาของโปรแกรมสำหรับการแก้จุดบกพร่องโดยที่ไม่ต้องเปลี่ยนแปลงโปรแกรมต้นฉบับ |
| การตรวจสอบ | • ช่วยให้สามารถมองเห็นลักษณะการทำงานของโปรแกรมได้<br>• สามารถตรวจสอบการทำงานของโปรแกรมได้ พร้อมกับเปลี่ยนแปลงค่าปัจจุบันของอุปกรณ์ |
| การจำลอง | สามารถจำลองการทำงานของโปรแกรมได้ โดยที่ไม่ต้องใช้ PLC จริง |

# 3.5 สรุปสำหรับหลักสูตรนี้

นี่คือส่วนสุดท้ายของหลักสูตร E-Learning ชุดนี้

ต่อไปนี้คือบทสรุปของหลักสูตรนี้

- **การออกแบบโปรแกรมอย่างมีประสิทธิภาพ**
  - กำหนดหมายเลข I/O ให้กับโมดูลต่างๆ เพื่อให้ใช้งานโปรแกรมในระบบต่างๆ ได้อย่างง่ายดาย
  - ปรับพื้นที่หน่วยความจำตามสถานะการใช้งานอุปกรณ์
  - ใช้เลเบิลเพื่อช่วยให้การตั้งโปรแกรมง่ายยิ่งขึ้นและเข้าใจการทำงานได้ง่ายยิ่งขึ้น
  - เพิ่มคอมเม้นต์ เพื่อปรับปรุงความสะดวกในการอ่านโปรแกรม

- **การออกแบบโปรแกรมขั้นสูง**
  - ใช้รีเทนทีฟไทม์เมอร์เพื่อคงค่าเวลาที่จับเวลาได้
  - ใช้อินเด็กซ์รีจิสเตอร์ อาร์เรย์ หรือโครงสร้าง เพื่อจัดการค่าจำนวนมาก
  - ใช้ฟังก์ชันแลตช์และไฟล์รีจิสเตอร์ในการคงค่าสถานะอุปกรณ์
  - ในระบบมีรีเลย์พิเศษและรีจิสเตอร์พิเศษซึ่งเก็บข้อมูลสถานะภายในของโมดูล CPU ซึ่งได้จัดเตรียมไว้แล้ว
  - จำนวนจริงจะแสดงด้วยเวิร์ดดีไวซ์ 2 หรือ 4 เวิร์ด ไม่สามารถกำหนดจำนวนเต็มและจำนวนจริงผสมกันในการดำเนินการเดียวกันได้

- **การแก้จุดบกพร่องอย่างมีประสิทธิภาพ**

  ผู้ใช้สามารถดำเนินการต่อไปนี้โดยใช้ซอฟแวร์ GX Works3:
  - แก้จุดบกพร่องโปรแกรมโดยไม่ต้องเปลี่ยนแปลงโปรแกรมต้นฉบับ
  - มองเห็นลักษณะการทำงานของโปรแกรม
  - จำลองการทำงานของโปรแกรม

เพื่อความพร้อมสำหรับการศึกษาในระดับถัดไป โปรดศึกษาหลักสูตรต่อไปนี้เกี่ยวกับ "การสร้างโครงสร้าง(Structuring)" ซึ่งจะแบ่งโปรแกรมออกเป็นเลเยอร์และส่วนประกอบต่างๆ เพื่อให้นำมาใช้ซ้ำได้อย่างง่าย

- การออกแบบโปรแกรมอย่างมีประสิทธิภาพ
- การออกแบบโปรแกรมอย่างมีประสิทธิภาพ (ฝึกหัด) (รอการเผยแพร่)

**แบบทดสอบ** แบบทดสอบประเมินผล

ขณะนี้คุณได้ผ่านบทเรียนทั้งหมดของหลักสูตร การประยุกต์ใช้โปรแกรม (Ladder Diagram/MELSEC iQ-R ซีรีส์) แล้ว คุณพร้อมที่จะทำแบบทดสอบประเมินผลแล้ว หากคุณยังไม่มั่นใจเกี่ยวกับหัวข้อต่างๆ ที่จะทดสอบ โปรดทบทวนหัวข้อเหล่านั้น

คำถามในแบบทดสอบประเมินผลนี้มีทั้งหมด 14 ข้อ (35 รายการ)

คุณสามารถทำแบบทดสอบประเมินผลได้หลายครั้งตามต้องการ

ผลคะแนน

จำนวนคำตอบที่ถูกต้อง จำนวนคำถาม เปอร์เซ็นต์คำตอบที่ถูกต้อง และผลลัพธ์ที่แสดงว่าผ่าน/ไม่ผ่านจะปรากฏบนหน้าผลคะแนน

**แบบทดสอบ** แบบทดสอบประเมินผล 1

ประโยคใดต่อไปนี้ถูกต้อง เกี่ยวกับการกำหนดหมายเลข I/O ของโมดูล?

ข้อ 1

- เราสามารถกำหนดหมายเลข I/O ให้กับแต่ละโมดูลด้วยตนเอง ดังนั้นไม่จำเป็นต้องแก้ไขโปรแกรมเมื่อการตั้งค่าของโมดูลมีการเปลี่ยนแปลง
- หมายเลข I/O ถูกกำหนดให้โดยอัตโนมัติจะไม่สามารถเปลี่ยนแปลงได้

**แบบทดสอบ** แบบทดสอบประเมินผล 2

ประโยคใดต่อไปนี้ถูกต้อง เกี่ยวกับการตั้งค่าจำนวนอุปกรณ์?

ข้อ 1

- อุปกรณ์แต่ละชุดจะต้องถูกกำหนดอย่างน้อยหนึ่งpoint แม้ว่าจะไม่มีการใช้งานอุปกรณ์ดังกล่าว
- สามารถกำหนดจำนวนอุปกรณ์ ได้ตามจำนวนที่ใช้

**แบบทดสอบ** แบบทดสอบประเมินผล 3

ประโยคใดต่อไปนี้ถูกต้อง เกี่ยวกับลาเบล? (ตอบได้หลายข้อ)

ข้อ 1

- [ ] การใช้ลาเบลจะช่วยระบุเป้าหมายของกระบวนการ และช่วยให้การออกแบบโปรแกรมง่ายขึ้น
- [ ] ลาเบลที่เป็นตัวแทนของสัญญาณในโมดูล จะถูกเตรียมไว้แล้ว
- [ ] สามารถเพิ่มคำอธิบายอุปกรณ์ (comment) ให้กับอุปกรณ์ต่างๆ เพื่อเพิ่มความสะดวกในการอ่านโปรแกรม
- [ ] เนื่องจากสามารถกำหนดค่าคงที่ให้กับลาเบลได้ จึงสามารถเปลี่ยนแปลงค่านี้ได้โดยไม่ต้องแก้ไขโปรแกรม

**แบบทดสอบ** แบบทดสอบประเมินผล 4

จงเติมข้อความบรรยาย retentive timer (ตัวจับเวลาแบบจำค่า)ต่อไปนี้ให้สมบูรณ์

retentive timer จะเริ่มหน่วงเวลาเมื่อ (ข้อ 1) ON (คอยล์ เปลี่ยนเป็น (ข้อ 2))
retentive timer จะเก็บค่าเวลาที่วัดได้เอาไว้แม้เงื่อนไขของอินพุทเปลี่ยนเป็น (ข้อ 3) และยังคงทำการหน่วงเวลาจากค่าที่เก็บไว้ เมื่อเงื่อนไขของอินพุทเปลี่ยนเป็น (ข้อ 4) อีกครั้ง
retentive timer สิ้นสุดการนับ เมื่อเวลาที่นับได้ถึงค่าที่กำหนด ในจุดที่ (ข้อ 5) เป็นสถานะเปิด

ข้อ 1 -- Select --

ข้อ 2 -- Select --

ข้อ 3 -- Select --

ข้อ 4 -- Select --

ข้อ 5 -- Select --

ข้อ 6 -- Select --

**แบบทดสอบ** แบบทดสอบประเมินผล 5

จงทำให้โปรแกรมควบคุมที่ดำเนินการการประมวลผลต่อไปนี้สมบูรณ์

- ใช้ retentive timer(ตัวจับเวลาแบบจำค่า) เบอร์ ST0 เพื่อวัดเวลาที่ ONสัญญาณอินพุต X0 หรือ X1
- เมื่อ ON X0 หรือ X1 เป็นเวลา 30 วินาที Y70 จะ ON ขึ้นมา และตัวแสดงการหมดเวลา
- เมื่อ ON X2 หน้าสัมผัสของ retentive timer จะ OFF (ST0) และเคลียร์เวลาที่วัดได้ (ค่าปัจจุบัน)

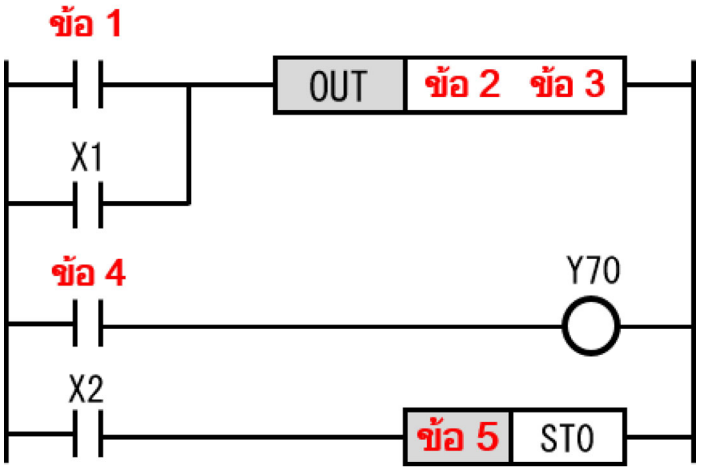
ข้อ 1 -- Select --

ข้อ 2 -- Select --

ข้อ 3 -- Select --

ข้อ 4 -- Select --

ข้อ 5 -- Select --




[ + ]

**แบบทดสอบ** แบบทดสอบประเมินผล 6

จงเลือกค่าที่ถูกเก็บไว้ในดาต้ารีจิสเตอร์ D20 เมื่อ ON X0 ภายใต้เงื่อนไขแต่ละข้อต่อไปนี้ในโปรแกรมควบคุมด้านล่าง

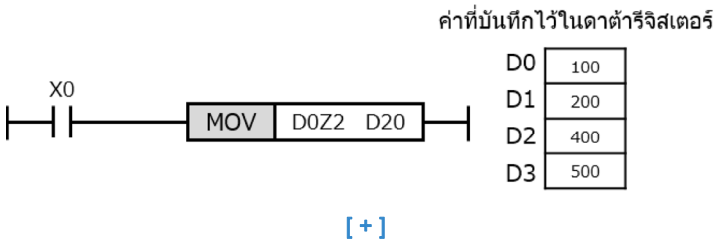
ข้อ 1) เมื่อค่าที่ถูกเก็บไว้ใน Z2 เป็น "0"
ข้อ 2) เมื่อค่าที่ถูกเก็บไว้ใน Z2 เป็น "1"
ข้อ 3) เมื่อค่าที่ถูกเก็บไว้ใน Z2 เป็น "2"
ข้อ 4) เมื่อค่าที่ถูกเก็บไว้ใน Z2 เป็น "3"

ข้อ 1 -- Select --

ข้อ 2 -- Select --

ข้อ 3 -- Select --

ข้อ 4 -- Select --




[ + ]

**แบบทดสอบ** แบบทดสอบประเมินผล 7

ประโยคใดต่อไปนี้ถูกต้อง เกี่ยวกับวิธีการกำหนดอาร์เรย์?

ข้อ 1

- เพิ่มหมายเลของค์ประกอบที่ท้ายชื่อลาเบล
- กำหนดหมายเลขอุปกรณ์โดยอ้อม

**แบบทดสอบ** แบบทดสอบประเมินผล 8

ประโยคเกี่ยวกับโครงสร้างใดต่อไปนี้ผิด?

ข้อ 1

- โครงสร้างที่ใช้เพื่อจัดระเบียบและจัดเก็บข้อมูล
- การใช้โครงสร้างช่วยให้สามารถอธิบายการประมวลผลข้อมูลจำนวนมากได้อย่างสะดวก
- สมาชิกที่ถูกกำหนดไว้ในโครงสร้างจะต้องเป็นข้อมูลชนิดเดียวกัน

**แบบทดสอบ** แบบทดสอบประเมินผล 9

ประโยคใดต่อไปนี้ถูกต้อง เกี่ยวกับฟังก์ชัน Latch?

ข้อ 1

- อุปกรณ์มีฟังก์ชันในการคงค่าอยู่แล้ว
- ต้องใช้ซอฟต์แวร์ในการตั้งค่าพารามิเตอร์เพื่อคงค่า

**แบบทดสอบ** แบบทดสอบประเมินผล 10

จงเติมข้อความอธิบายไฟล์รีจิสเตอร์ต่อไปนี้ให้สมบูรณ์

ไฟล์รีจิสเตอร์ คืออุปกรณ์ประเภทเวิร์ดที่ใช้เพื่อขยายดาต้ารีจิสเตอร์(D) และมีสัญลักษณ์อุปกรณ์เป็น (ข้อ 1)
file register มีความจุ (ข้อ 2) มากกว่าดาต้ารีจิสเตอร์และข้อมูลที่ถูกเก็บไว้ (ข้อ 3) แม้เมื่อปิดระบบแล้วหรือรีเซ็ตโมดูล CPU ไปแล้ว
ในการใช้ ไฟล์รีจิสเตอร์ การตั้งค่าพารามิเตอร์ (ข้อ 4) ต้องใช้ซอฟต์แวร์

ข้อ 1 -- Select --

ข้อ 2 -- Select --

ข้อ 3 -- Select --

ข้อ 4 -- Select --

**แบบทดสอบ** แบบทดสอบประเมินผล 11

ประโยคใดต่อไปนี้ถูกต้อง เกี่ยวกับรีเลย์พิเศษและ รีจิสเตอร์พิเศษ?

ข้อ 1

- สถานะภายในของโมดูล CPU ได้ถูกบันทึกไว้แล้วภายใน รีเลย์พิเศษและ รีจิสเตอร์พิเศษ ซึ่งอุปกรณ์เหล่านี้ ถูกใช้เป็นเงื่อนไขการคำนวนในโปรแกรมควบคุม
- ฟังก์ชันพิเศษสามารถถูกกำหนดให้กับ รีเลย์พิเศษและ รีจิสเตอร์พิเศษ ได้อย่างอิสระ

**แบบทดสอบ** แบบทดสอบประเมินผล 12

จงเติมข้อความอธิบายเกี่ยวกับจำนวนจริง (single-precision) ต่อไปนี้ให้สมบูรณ์

- จำนวนจริงหนึ่งค่าจะใช้อุปกรณ์ประเภทเวิร์ดจำนวน (ข้อ 1) และจะถูกเก็บไว้ในที่ว่างของหน่วยความจำขนาด (ข้อ 2) บิท
- ข้อมูลที่เป็นค่าตัวเลขของจำนวนจริงจะเรียกว่า (ข้อ 3) เช่น 2.035 อธิบายว่าเป็น (ข้อ 4) ในโปรแกรมควบคุม
- จำนวนเต็มและจำนวนจริง (ข้อ 5) ถูกรวมกันอยู่ในคำสั่งที่ใช้กับจำนวนจริง

ข้อ 1 -- Select --

ข้อ 2 -- Select --

ข้อ 3 -- Select --

ข้อ 4 -- Select --

ข้อ 5 -- Select --

**แบบทดสอบ** แบบทดสอบประเมินผล 13

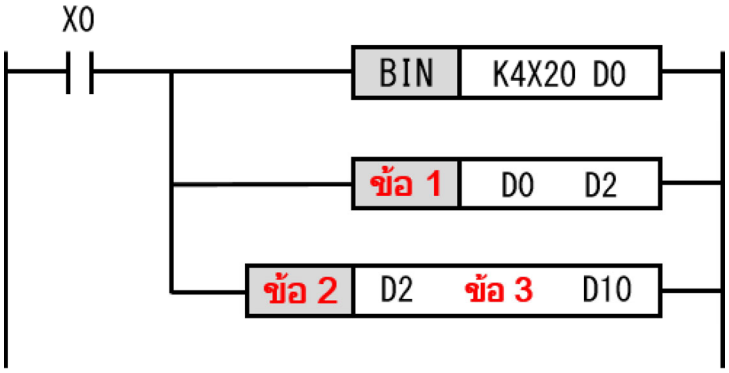
จงทำให้โปรแกรมควบคุมซึ่งดำเนินการการประมวลผลต่อไปนี้สมบูรณ์

- อ่านข้อมูลจาก X20 ถึง X2F (ข้อมูล BCD) เมื่อ ON X0 และเก็บข้อมูลไปยัง D0
- แปลงค่าใน D0 ให้เป็นจำนวนจริง และเก็บค่าที่แปลงไว้ใน D2
- คูณค่าใน D2 ด้วย 3.14 และเก็บผลลัพธ์ไว้ใน D10

ข้อ 1 -- Select --

ข้อ 2 -- Select --

ข้อ 3 -- Select --




[ + ]

**แบบทดสอบ** แบบทดสอบประเมินผล 14

ประโยคใดต่อไปนี้ถูกต้อง เกี่ยวกับการหาจุดบกพร่องของโปรแกรมควบคุม?

ข้อ 1

- การจำลองการทำงานของโปรแกรมสามารถทำได้อย่างปลอดภัยโดยใช้ฟังก์ชันของซอฟต์แวร์
- การหาจุดบกพร่องของโปรแกรมจะต้องดำเนินการในระบบจริง

**แบบทดสอบ** คะแนนการทดสอบ

คุณทำแบบทดสอบประเมินผลเสร็จสิ้นแล้ว ผลลัพธ์ของคุณมีดังต่อไปนี้
ในการสิ้นสุดแบบทดสอบประเมินผล ให้ไปยังหน้าถัดไป

| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | แบบทดสอบประเมินผล 1 | ✓ | | | | | | | | | |
| | แบบทดสอบประเมินผล 2 | ✓ | | | | | | | | | |
| | แบบทดสอบประเมินผล 3 | ✓ | | | | | | | | | |
| | แบบทดสอบประเมินผล 4 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | | |
| | แบบทดสอบประเมินผล 5 | ✓ | ✓ | ✓ | ✓ | ✓ | | | | | |
| | แบบทดสอบประเมินผล 6 | ✓ | ✓ | ✓ | ✓ | | | | | | |
| | แบบทดสอบประเมินผล 7 | ✓ | | | | | | | | | |
| | แบบทดสอบประเมินผล 8 | ✓ | | | | | | | | | |
| | แบบทดสอบประเมินผล 9 | ✓ | | | | | | | | | |
| | แบบทดสอบประเมินผล 10 | ✓ | ✓ | ✓ | ✓ | | | | | | |
| | แบบทดสอบประเมินผล 11 | ✓ | | | | | | | | | |
| | แบบทดสอบประเมินผล 12 | ✓ | ✓ | ✓ | ✓ | ✓ | | | | | |
| | แบบทดสอบประเมินผล 13 | ✓ | ✓ | ✓ | | | | | | | |
| | แบบทดสอบประเมินผล 14 | ✓ | | | | | | | | | |

จำนวนคำถามทั้งหมด: 35

คำตอบที่ถูกต้อง: 35

เปอร์เซ็นต์: 100 %

ล้าง

**คุณได้เรียนรู้หลักสูตร การประยุกต์ใช้โปรแกรม (Ladder Diagram/MELSEC iQ-R ซีรีส์) เสร็จสิ้นแล้ว**

ขอขอบคุณสำหรับการเรียนรู้หลักสูตรนี้

เราหวังว่าคุณจะเพลิดเพลินกับบทเรียน และข้อมูลที่คุณได้รับจากหลักสูตรนี้จะเป็นประโยชน์ในอนาคต

คุณสามารถทบทวนหลักสูตรได้หลายครั้งตามต้องการ

**ทบทวน**

**ปิด**