20 การเขียนโปรแกรมโดยใช้สคริปต์

(การเขียนโปรแกรมที่ไม่ใช้พาร์ท)

บทนี้จะอธิบายเกี่ยวกับหลักการพื้นฐานของ “การเขียนโปรแกรมโดยใช้สคริปต์” ใน GP-Pro EX และวิธีต่าง ๆ ในการสร้างสคริปต์
โปรดเริ่มต้นด้วยการอ่าน “20.1 เมนูการตั้งค่า” (หน้า 20-2) แล้วจึงไปอ่านหน้าที่เกี่ยวข้อง

20.1 เมนูการตั้งค่า

D-Scripts เป็นภาษาที่ไม่ซับซ้อน ซึ่งผู้ใช้สามารถนำไปเขียนโปรแกรมสำหรับใช้เองได้ คุณสามารถใช้คุณสมบัตินี้เพื่อทำงานในเครื่อง GP หรือเพื่อสื่อสารระหว่างเครื่อง GP กับอุปกรณ์ต่อพ่วงที่ไม่รองรับ

⚠ คำเตือน

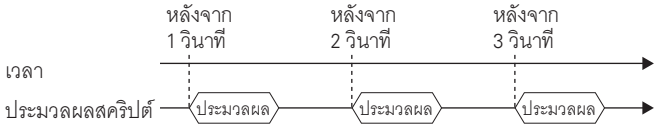
ห้ามใช้ D-Scripts/Global D-Scripts ควบคุมระบบการทำงานที่อาจก่อให้เกิดอันตรายถึงแก่ชีวิตหรือสร้างความเสียหายร้ายแรง

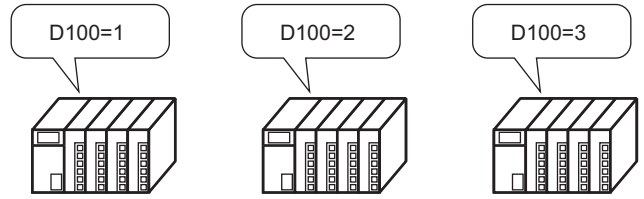
หมายเหตุ

- การตั้งค่า D-Scripts จะทำบนหน้าจอหลัก โดยหน้าจอหลักดังกล่าวจะพิจารณาเงื่อนไขขณะที่หน้าจอกำลังแสดงอยู่แล้วจึงเรียกใช้สคริปต์นั้น
- สำหรับ Global D-Scripts เมื่อเครื่อง GP ทำงาน ไม่ว่าหน้าจอนั้นจะแสดงขึ้นหรือไม่ โปรแกรมจะทำงานตามเงื่อนไข (ทริกเกอร์)
- Extended Scripts ใช้สำหรับโปรแกรมการสื่อสารระดับสูง

การทำงานตามเงื่อนไข

สร้างสคริปต์ซึ่งจะเปลี่ยนหน้าจอต่าง ๆ เป็นหน้าจอหมายเลข 7 โดยอัตโนมัติ หลังจากเวลาผ่านไป 3 วินาที





D100 ไม่ใช่ 3 ดังนั้น คำสั่งที่อยู่หลัง “if” จะไม่ทำงาน

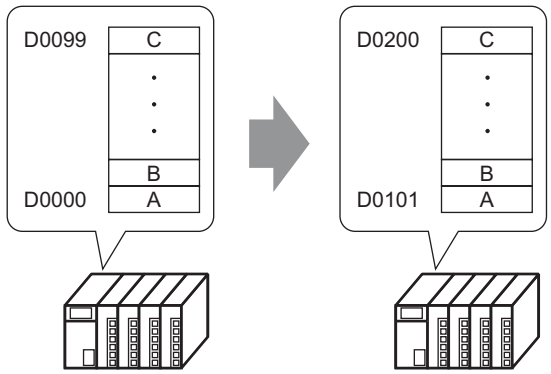
D100 ไม่ใช่ 3 ดังนั้น คำสั่งที่อยู่หลัง “if” จะไม่ทำงาน

D100 = 3 ดังนั้น เงื่อนไขเป็นจริงและ [w:LSD008]=7 จะทำงาน

☞ ขั้นตอนการตั้งค่า (หน้า 20-6)
☞ รายละเอียด (หน้า 20-5)

การคัดลอกข้อมูลในบล็อค

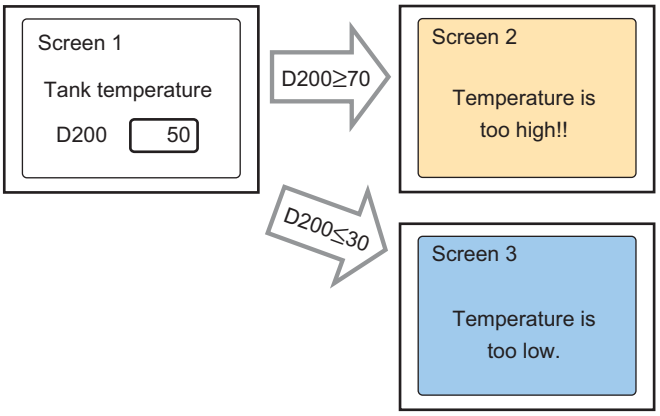
สร้างสคริปต์เพื่อตรวจหาขอบขาขึ้น (0 เปลี่ยนเป็น 1) ของตำแหน่งบิต M0100 แล้วคัดลอกข้อมูลที่จัดเก็บไว้ในอุปกรณ์เชื่อมต่อไปไว้ที่ตำแหน่งอื่น



☞ ขั้นตอนการตั้งค่า (หน้า 20-13)
☞ รายละเอียด (หน้า 20-12)

การแสดงการแจ้งเตือนเมื่อเกิดข้อผิดพลาด

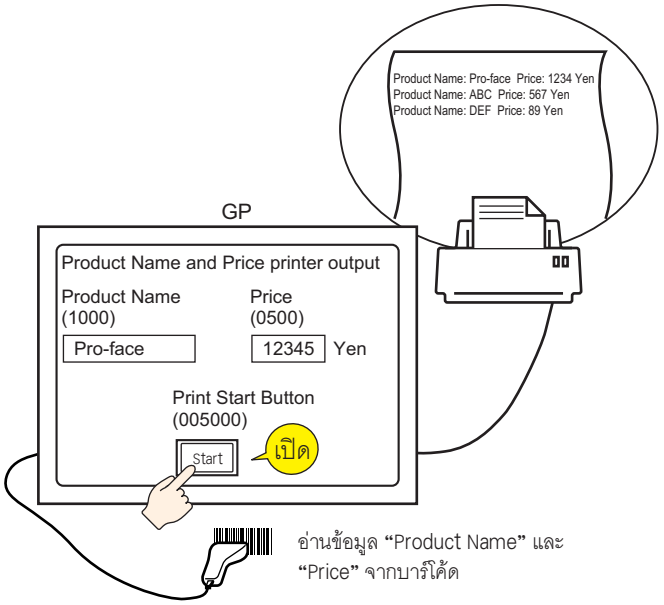
ระบบจัดการอุณหภูมิจะตรวจหาบิตที่มีข้อผิดพลาดจากอุปกรณ์ที่เชื่อมต่อไว้ และแสดงข้อความแจ้งเตือนเมื่อตำแหน่งจัดเก็บข้อมูลอุณหภูมิ (D200) มีค่าเพิ่มขึ้นถึงระดับตั้งแต่ 70°C ขึ้นไป หรือมีค่าลดลงถึงระดับตั้งแต่ 30°C ลงไป โดยสคริปต์นี้จะนับจำนวนข้อผิดพลาดที่ตรวจพบด้วย



☞ ขั้นตอนการตั้งค่า (หน้า 20-19)
☞ รายละเอียด (หน้า 20-18)

การสื่อสารกับอุปกรณ์ต่อพ่วงที่สคริปต์ปกติไม่รองรับ

สร้าง Extended Script เพื่อส่งข้อมูลที่อ่านได้จากบาร์โค้ดซึ่งเชื่อมต่อกับพอร์ต USB ไปยังเครื่องพิมพ์ที่ต่อกับพอร์ต COM1 แบบอนุกรม



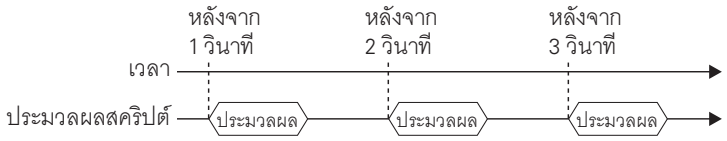
☞ ขั้นตอนการตั้งค่า (หน้า 20-35)
☞ รายละเอียด (หน้า 20-22)

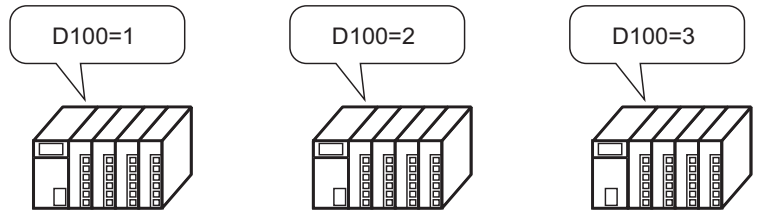
20.2 การทำงานตามเงื่อนไข

หมายเหตุ
- โปรดอ่านรายละเอียดจากคำแนะนำในการตั้งค่า
☞ "20.8.1 คำแนะนำในการตั้งค่าทั่วไป (D-Script)" (หน้า 20-50)

การทำงาน

สร้างสคริปต์ซึ่งจะเปลี่ยนหน้าจอต่าง ๆ เป็นหน้าจอหมายเลข 7 โดยอัตโนมัติ หลังจากเวลาผ่านไป 3 วินาที





D100 ไม่ใช่ 3 ดังนั้น คำสั่งที่อยู่หลัง "if" จะไม่ทำงาน

D100 ไม่ใช่ 3 ดังนั้น คำสั่งที่อยู่หลัง "if" จะไม่ทำงาน

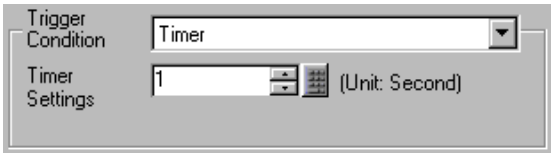
D100 = 3 ดังนั้น เงื่อนไขเป็นจริงและ [w:LSD008]=7 จะทำงาน

คำสั่งที่ใช้

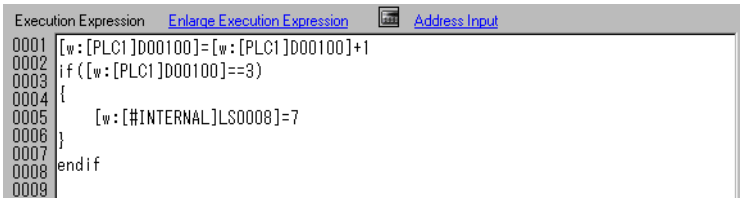
คำสั่ง	ข้อมูลสรุปของฟังก์ชัน
Assignment (=)	กำหนดค่าฝั่งขวาให้แก่ค่าฝั่งซ้าย
Addition (+)	เพิ่มค่าคงที่ให้แก่ข้อมูลของอุปกรณ์ชนิดเวิร์ด
if ( )	เมื่อเงื่อนไขภายในวงเล็บ "( )" ที่อยู่ตามหลัง "if" เป็นจริง ระบบจะดำเนินกระบวนการที่อยู่หลังข้อความคำสั่ง "if ( )"
Equivalent (==)	เปรียบเทียบค่าฝั่งซ้ายและขวา เงื่อนไขจะเป็นจริงหากฝั่งซ้ายเท่ากับฝั่งขวา
LS0008	เปลี่ยนเป็นหมายเลขหน้าจอที่จัดเก็บไว้ในค่านี้ ☞ "A.1.4.2 พื้นที่เก็บข้อมูลระบบ" (หน้า A-10)

เงื่อนไขการทริกเกอร์

เลือก Timer ตามภาพด้านล่างนี้ แล้วตั้งค่า [Timer Settings] เป็น 1 วินาที

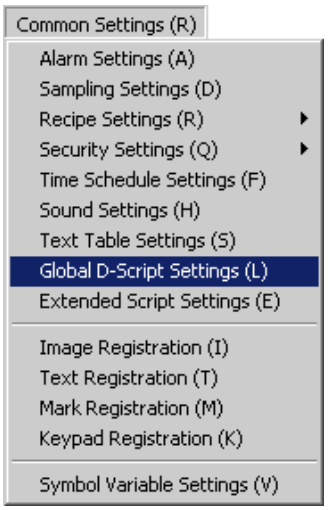


สคริปต์ที่เสร็จแล้ว

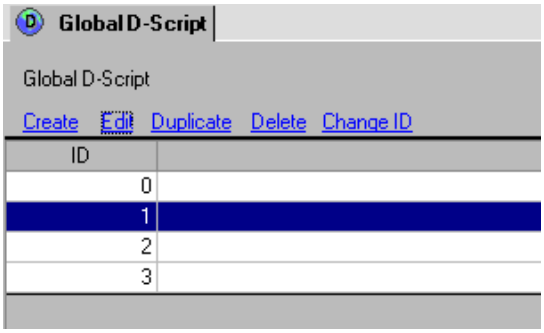


ขั้นตอนการสร้าง

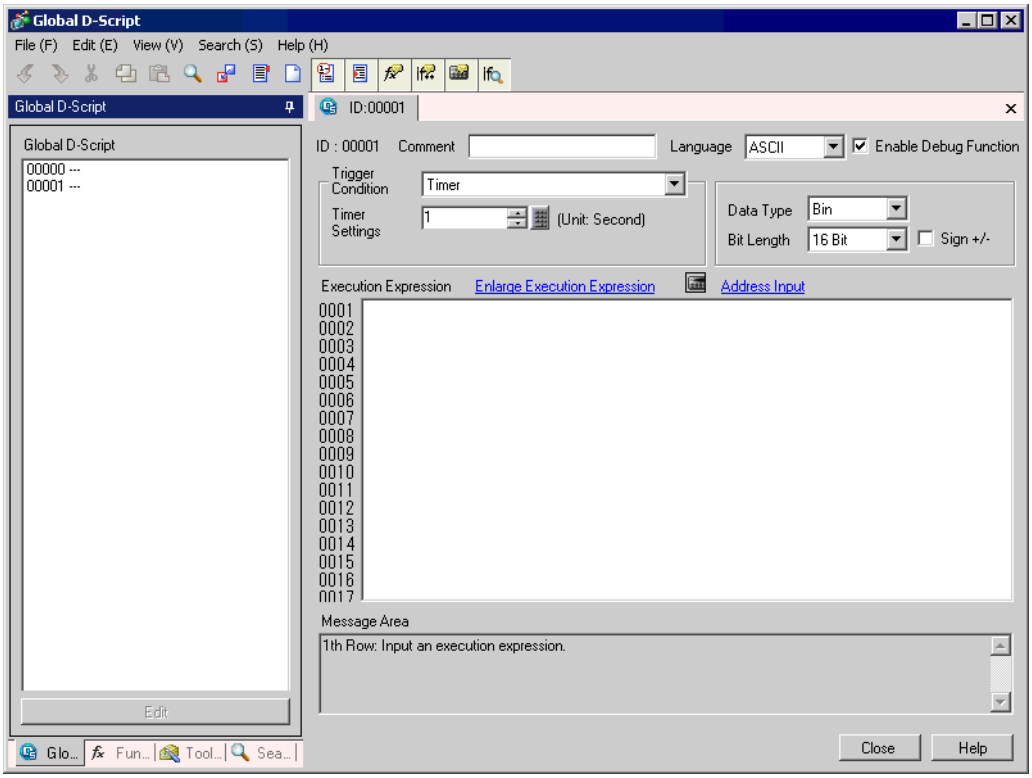
1 เลือกเมนู [Common Settings] - คำสั่ง [Global D-Script Settings]



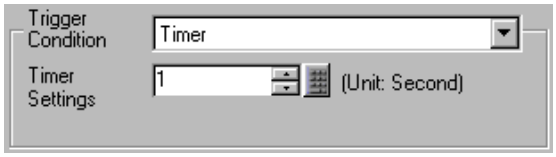
2 คลิก [Create] หากเป็นการเข้าใช้สคริปต์ที่ลงทะเบียนไว้ก่อนหน้านี้ ให้ระบุหมายเลข ID แล้วคลิก [Edit]



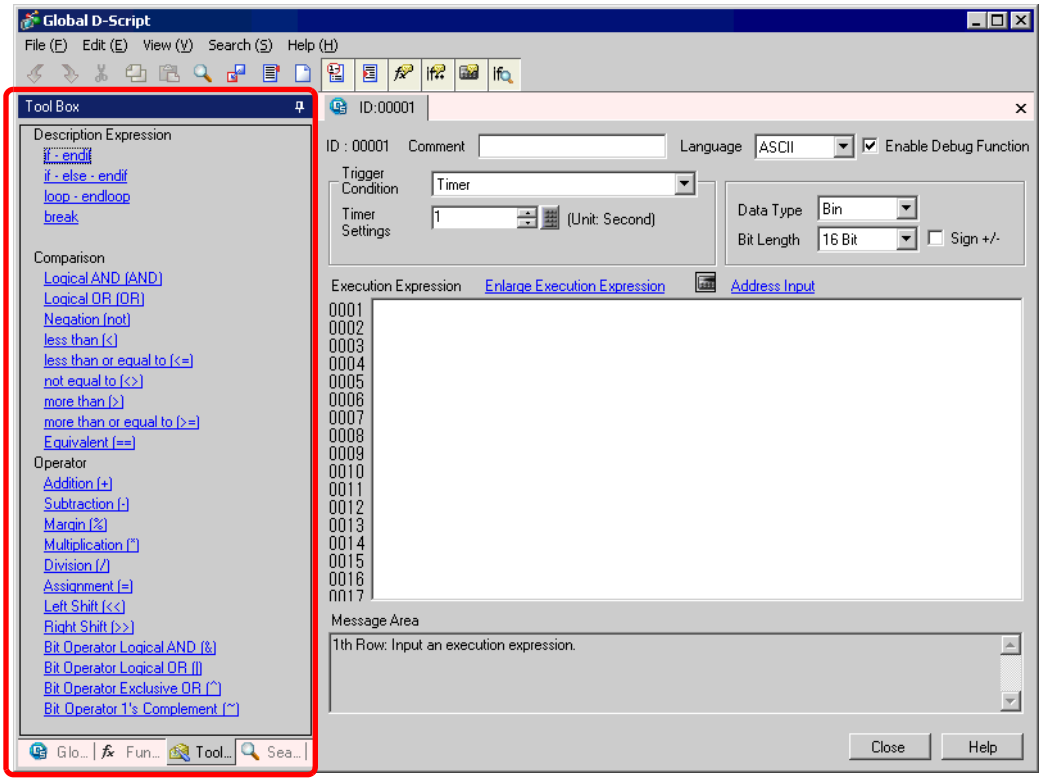
3 กล่องโต้ตอบ [Global D-Script] จะปรากฏขึ้น



4 ตั้งเงื่อนไขการทำงานของสคริปต์ (ทริกเกอร์) เลือก Timer ตามภาพด้านล่างนี้ แล้วตั้งค่า [Timer Settings] เป็น 1 วินาที

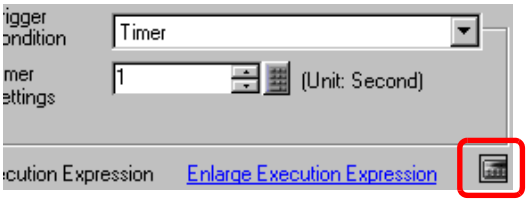


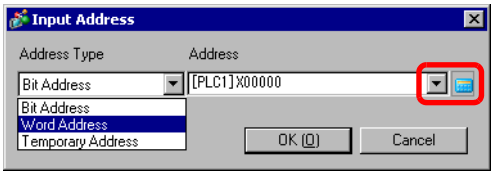
5 คลิกแท็บ [Tool Box] คุณสามารถใช้กล่องเครื่องมือนี้เพื่อใส่คำสั่งที่ใช้ได้ในสคริปต์ได้อย่างง่ายดาย เพียงแค่คลิกที่คำสั่งต้องการ



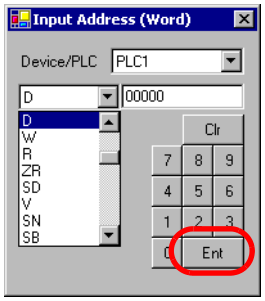
6 สร้างสคริปต์แถวแรก การทำงานของแถวแรกจะตั้งค่าเริ่มต้นของ D00100 เป็น 0 แล้วเพิ่มค่าขึ้นครั้งละ 1 ทุกครั้งที่มีการประมวลผลแถวนั้น

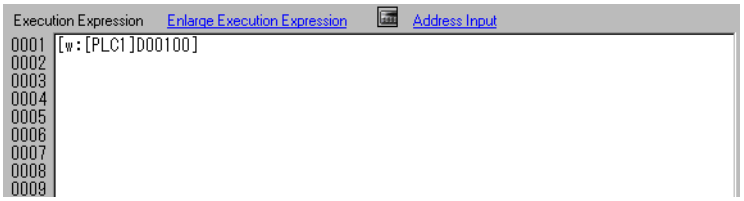
คลิก [Address Input Dialog] เลือก [Word Address] แล้วคลิก 



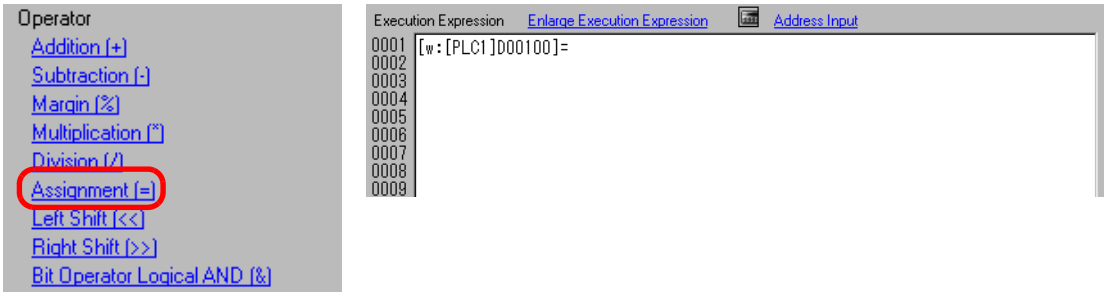


7 ป้อน D00100 แล้วคลิก [ENT]

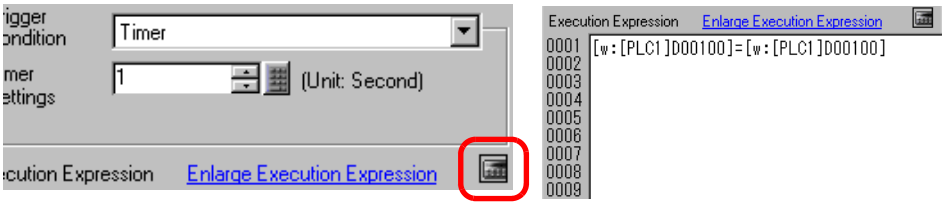




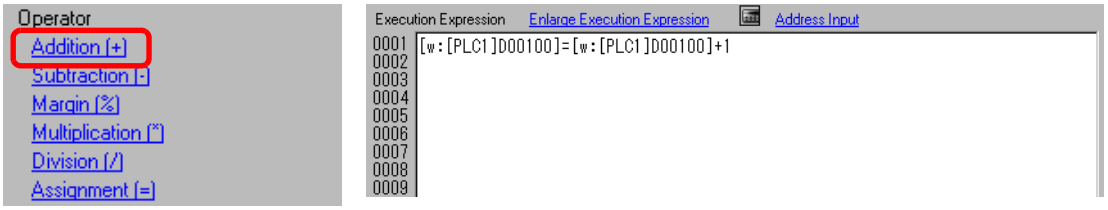
8 ใน [Tool Box] ให้คลิก [Assignment (=)]



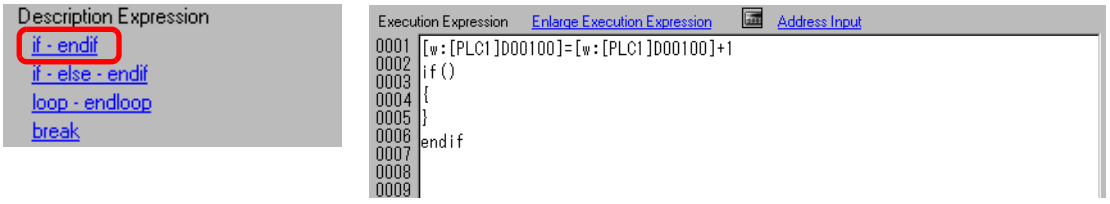
9 ทำซ้ำขั้นตอนที่ 6 ถึง 7 เพื่อใส่ D00100 อีกครั้งหนึ่ง



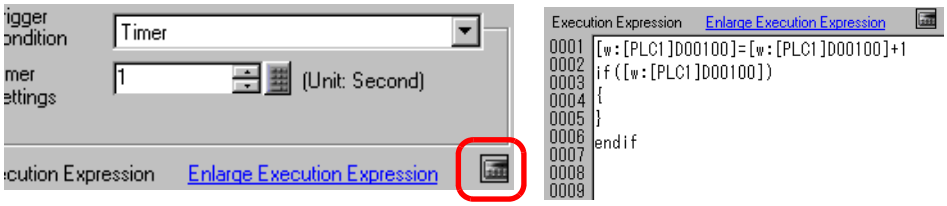
10 คลิก [Addition (+)] แล้วป้อน “1” แถวแรกเสร็จสมบูรณ์แล้ว



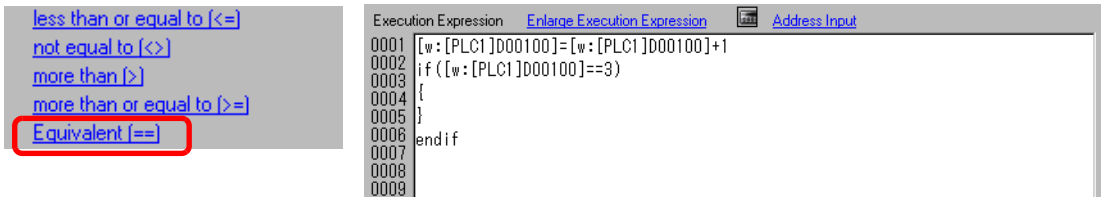
11 สร้างสคริปต์แถวที่สอง ในแถวที่สอง เมื่อเงื่อนไขภายในวงเล็บ “( )” ที่อยู่ตามหลัง “if” เป็นจริง ระบบจะดำเนินกระบวนการที่อยู่หลังข้อความคำสั่ง “if ( )”
คลิก [if - endif]



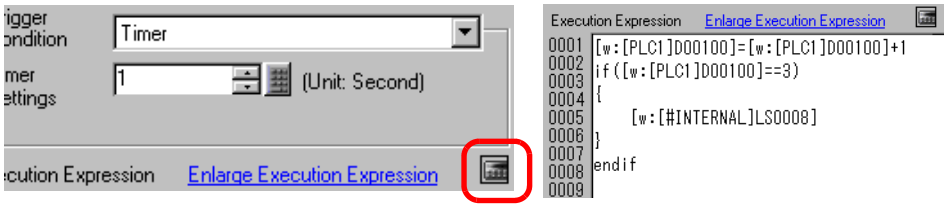
12 สร้างนิพจน์เงื่อนไขในวงเล็บ “( )” ที่อยู่หลัง “if” นิพจน์เงื่อนไขนี้จะเปรียบเทียบค่าที่เก็บไว้ใน D00100 กับ “3” และเงื่อนไขจะเป็นจริงหากค่าเท่ากัน
เลื่อนเคอร์เซอร์ให้อยู่ในวงเล็บ “( )” แล้วทำซ้ำขั้นตอนที่ 6 ถึง 7 เพื่อใส่ D00100 อีกหนึ่งครั้ง



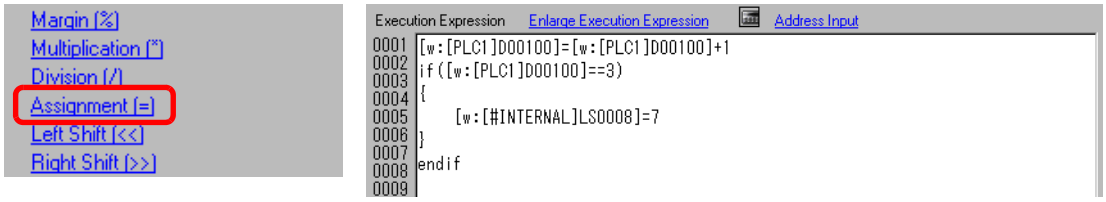
13 คลิก [Equivalent (==)] แล้วป้อน “3” แถวที่สองเสร็จสมบูรณ์แล้ว



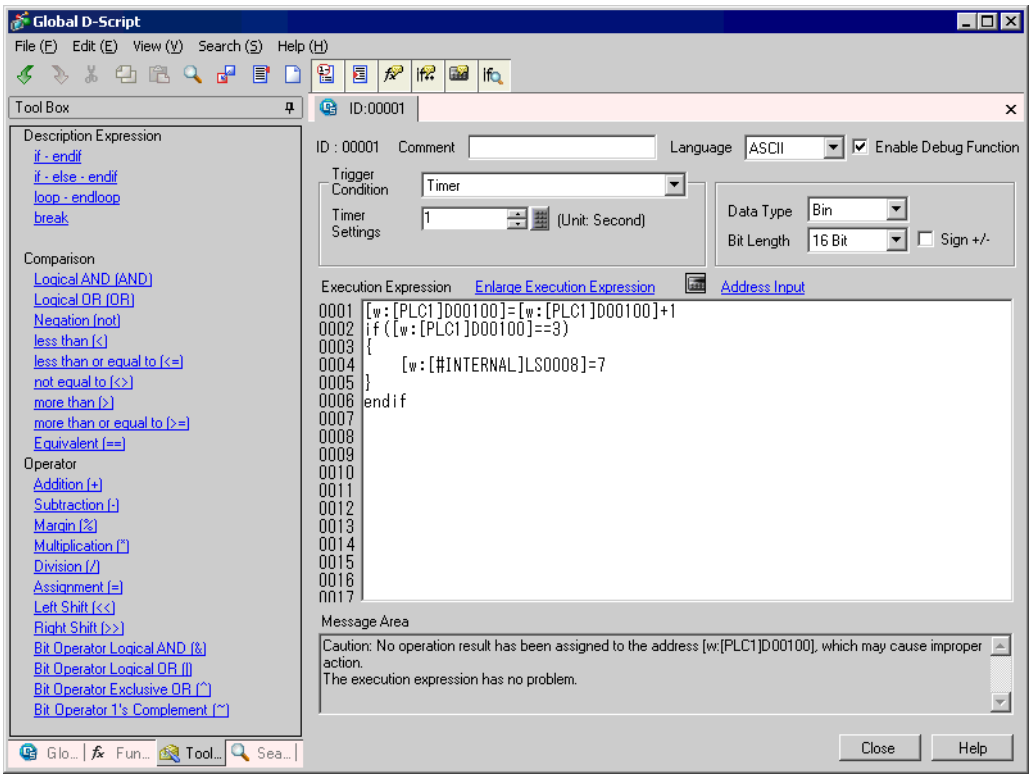
14 เลื่อนเคอร์เซอร์ให้อยู่ในวงเล็บปีกกา “{ }” แล้วขึ้นบรรทัดใหม่ ทำซ้ำขั้นตอนที่ 6 ถึง 7 เพื่อใส่ LS0008 อีกครั้งหนึ่ง



15 คลิก [Assignment (=)] แล้วป้อน “7”



16 สคริปต์นี้จะเสร็จสมบูรณ์

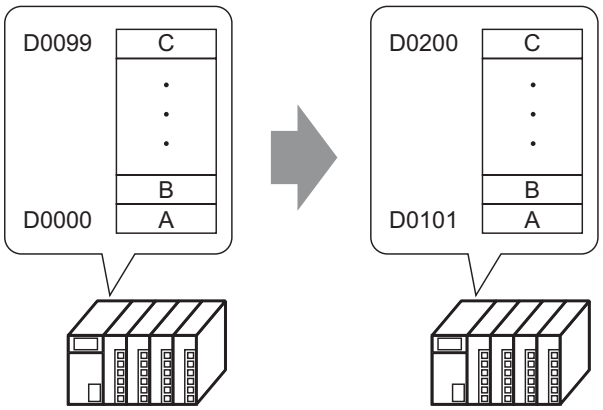


20.3 การคัดลอกข้อมูลในบล็อค

หมายเหตุ
- โปรดอ่านรายละเอียดจากคำแนะนำในการตั้งค่า
☞ “20.8.1 คำแนะนำในการตั้งค่าทั่วไป (D-Script)” (หน้า 20-50)

การทำงาน

สร้างสคริปต์เพื่อตรวจหาขอบขาขึ้น (0 เปลี่ยนเป็น 1) ของตำแหน่งบิต M0100 แล้วคัดลอกข้อมูลที่จัดเก็บไว้ในอุปกรณ์เชื่อมต่อไปไว้ที่ตำแหน่งอื่น

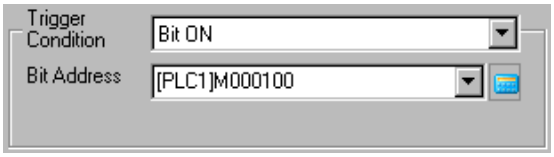


คำสั่งที่ใช้

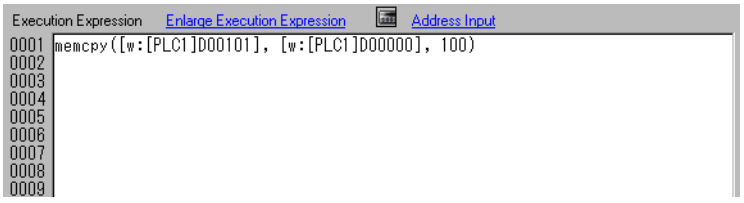
คำสั่ง	ข้อมูลสรุปของฟังก์ชัน
Copy Memory memcpy ( )	คัดลอกค่าที่เก็บไว้ลงในอุปกรณ์โดยทำเพียงครั้งเดียว ระบบจะคัดลอกข้อมูลตามหมายเลขตำแหน่งไปยังตำแหน่งเวิร์ดของปลายทางการคัดลอก โดยเริ่มจากตำแหน่งเวิร์ดแรกสุดของข้อมูลต้นทาง [Format] memcpy ([ตำแหน่งปลายทางการคัดลอก], [ตำแหน่งที่จะคัดลอก] จำนวนของเวิร์ด)

เงื่อนไขการทริกเกอร์

เลือกบิตขาขึ้นดังนี้ แล้วตั้งค่า [Bit Address] เป็น M000100

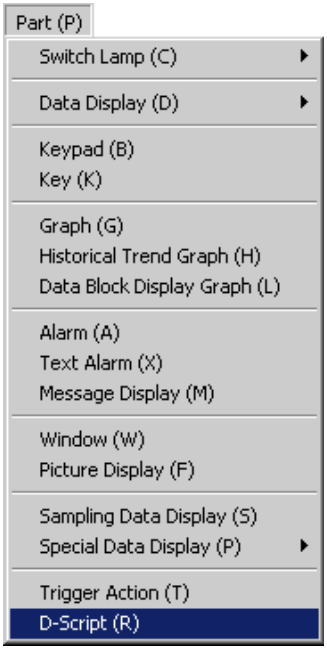


สคริปต์ที่เสร็จแล้ว

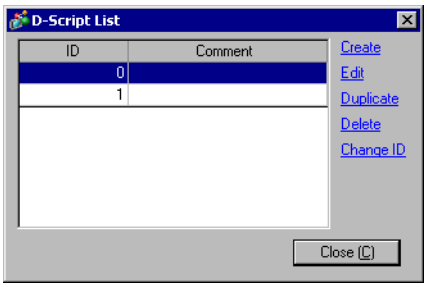


ขั้นตอนการสร้าง

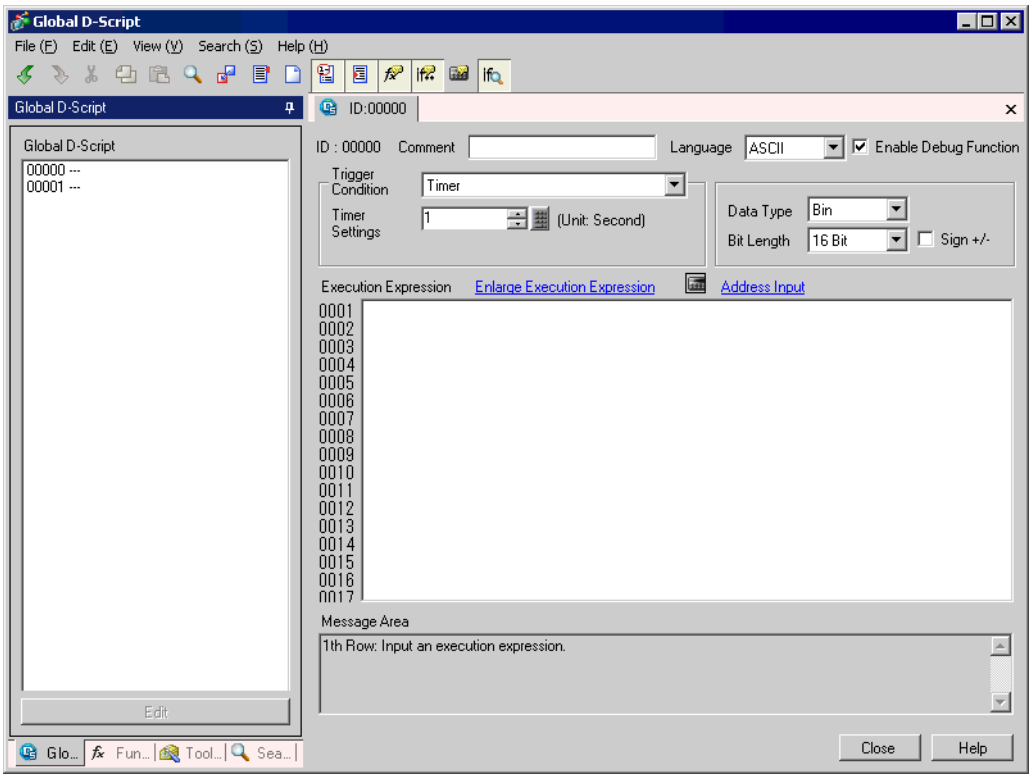
1 เลือกเมนู [Part] - คำสั่ง [D-Script] หรือคลิก 



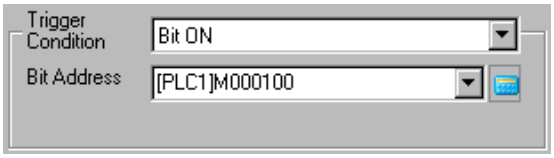
2 คลิก [Create] หากลงทะเบียนสคริปต์นี้ไว้แล้ว หมายเลข ID จะปรากฏขึ้น



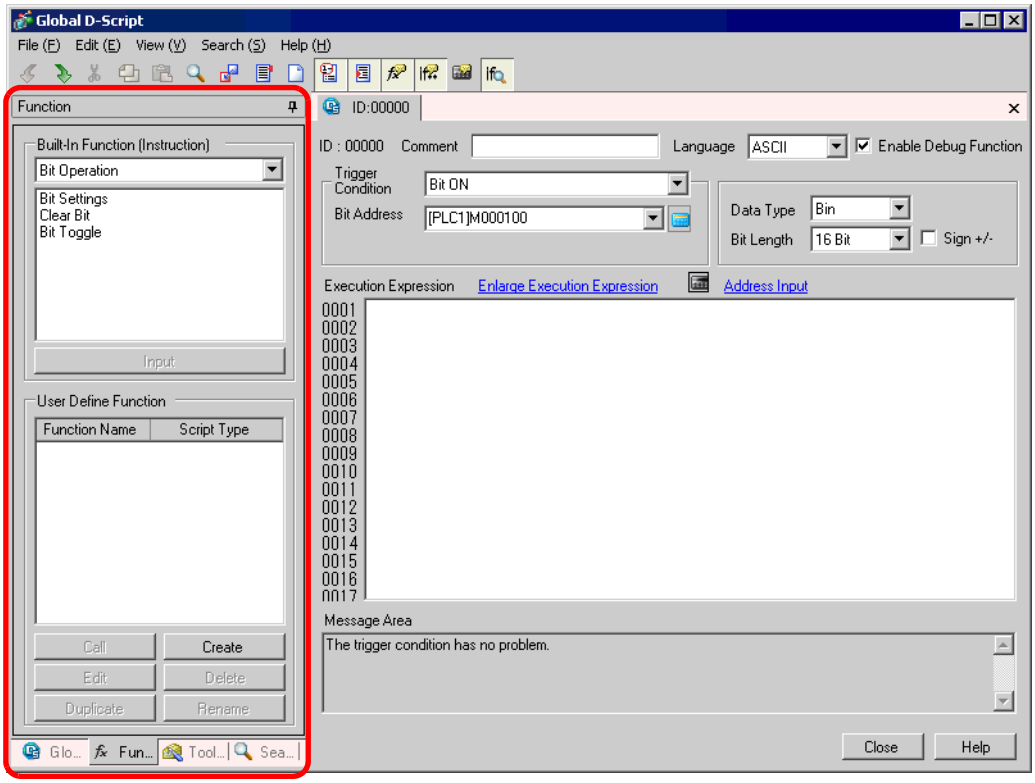
3 กล่องโต้ตอบ [D-Script] จะปรากฏขึ้น



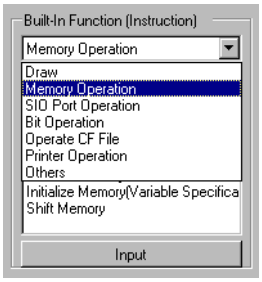
4 ตั้งเงื่อนไขการทำงานของสคริปต์ (ทริกเกอร์) เลือกบิตขาขึ้นดังนี้ แล้วตั้งค่า [Bit Address] เป็น M000100



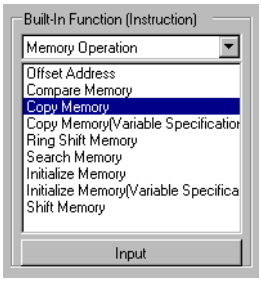
5 คลิกแท็บ [Function] คุณสามารถใส่คำสั่งที่ใช้ในสคริปต์ได้อย่างง่ายดายเพียงแค่คลิกที่คำสั่งที่ต้องการในฟังก์ชันต่าง ๆ ที่มีให้เลือก

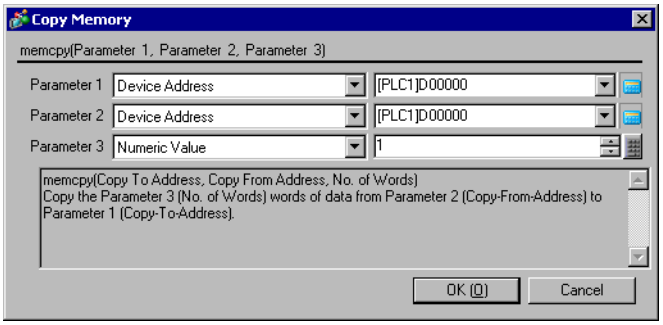


6 เลือก [Memory Operation] จากเมนูพูลดาวน์ [Built-in Function (Instruction)]

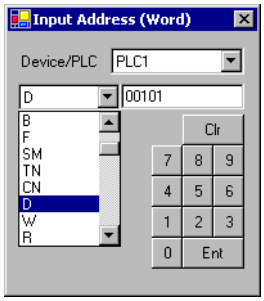


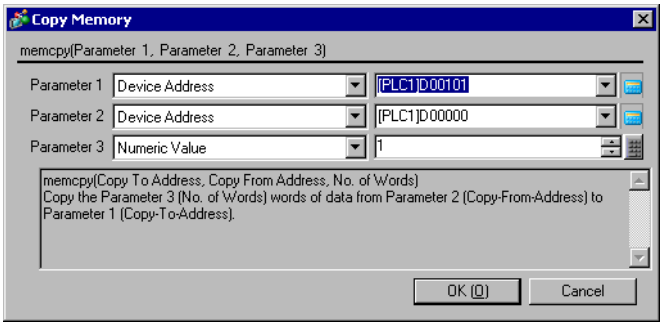
7 ดับเบิลคลิก [Copy Memory] จากนั้นให้ตั้งค่าตำแหน่งปลายทาง ตำแหน่งต้นทาง และจำนวนตำแหน่งในกล่องโต้ตอบ คลิก 



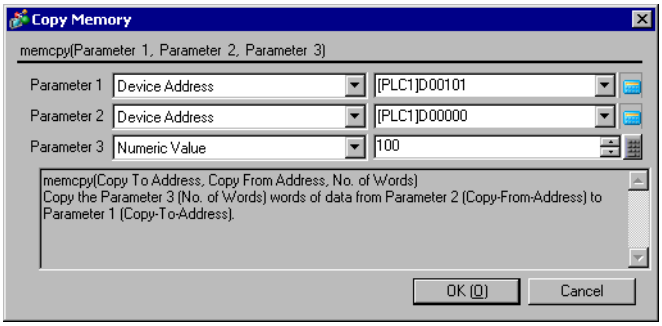


8 ป้อน D00101 แล้วคลิก [ENT]

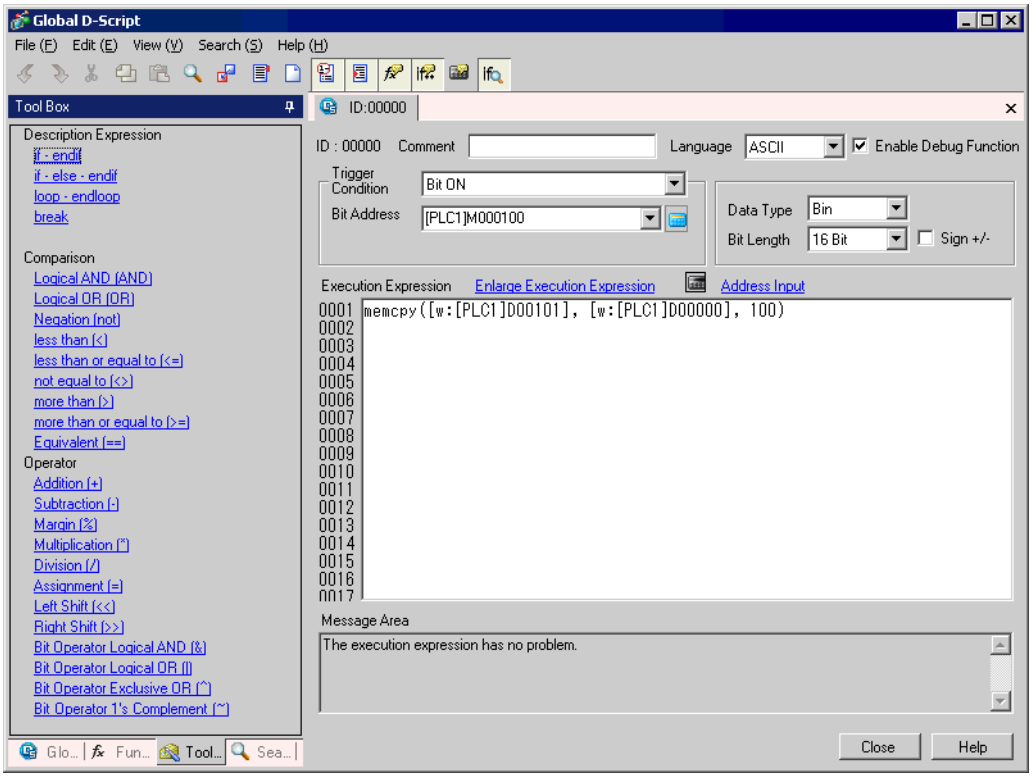




9 ป้อนจำนวนตำแหน่งเป็น 100 ทำซ้ำขั้นตอนที่ 8 เพื่อตั้งค่าตำแหน่งเวิร์ดต้นทางเป็น D00000 แล้วคลิก [OK]



10 สคริปต์นี้จะเสร็จสมบูรณ์

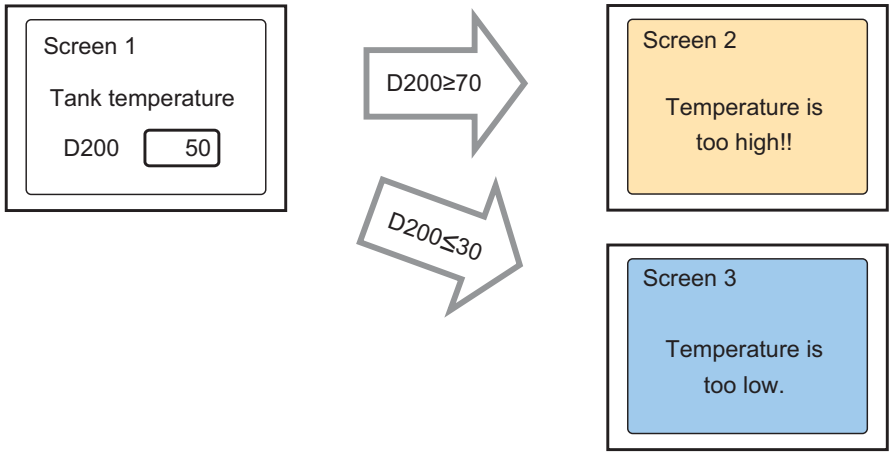


20.4 การแสดงการแจ้งเตือนเมื่อเกิดข้อผิดพลาด

หมายเหตุ
- โปรดอ่านรายละเอียดจากคำแนะนำในการตั้งค่า
☞ "20.8.1 คำแนะนำในการตั้งค่าทั่วไป (D-Script)" (หน้า 20-50)

การทำงาน

ระบบจัดการอุณหภูมิจะตรวจหาบิตที่มีข้อผิดพลาดจากอุปกรณ์ที่เชื่อมต่อไว้ และแสดงข้อความแจ้งเตือน เมื่อตำแหน่งจัดเก็บข้อมูลอุณหภูมิ (D200) มีค่าเพิ่มขึ้นถึงระดับตั้งแต่ 70°C ขึ้นไป หรือมีค่าลดลงถึงระดับตั้งแต่ 30°C ลงไป โดยสคริปต์นี้จะนับจำนวนข้อผิดพลาดที่ตรวจพบด้วย



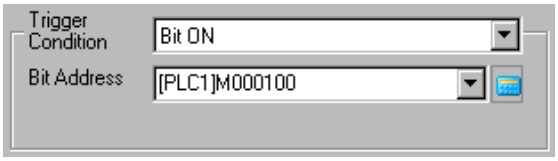
ตำแหน่งที่ทำการนับทุกครั้งที่อุณหภูมิที่ D200 เพิ่มขึ้นถึงระดับตั้งแต่ 70°C ขึ้นไป และบันทึกจำนวนครั้งที่เกิดขึ้น : LS0300
ตำแหน่งที่ทำการนับทุกครั้งที่อุณหภูมิที่ D200 ลดลงถึงระดับตั้งแต่ 30°C ลงไป และบันทึกจำนวนครั้งที่เกิดขึ้น : LS0301
ตำแหน่งที่จัดเก็บหมายเลขหน้าจอเพื่อแสดงหน้าจอแจ้งเตือน : LS0302

คำสั่งที่ใช้

คำสั่ง	ข้อมูลสรุปของฟังก์ชัน
if ( )	เมื่อเงื่อนไขภายในวงเล็บ "( )" ที่อยู่ตามหลัง "if" เป็นจริง ระบบจะดำเนินกระบวนการที่อยู่หลังข้อความคำสั่ง "if ( )"
more than or equal to (>=)	เป็นจริงหาก N1 มากกว่าหรือเท่ากับ N2 (N1 >= N2)
Assignment (=)	กำหนดค่าฝั่งขวาให้แก่ค่าฝั่งซ้าย
Addition (+)	เพิ่มค่าคงที่ให้แก่ข้อมูลของอุปกรณ์ชนิดเวิร์ด
less than or equal to (<=)	เป็นจริงหาก N1 น้อยกว่าหรือเท่ากับ N2 (N1 <= N2)

เงื่อนไขการทริกเกอร์

เลือกบิตขาขึ้นดังนี้ แล้วตั้งค่า [Bit Address] เป็น M000100



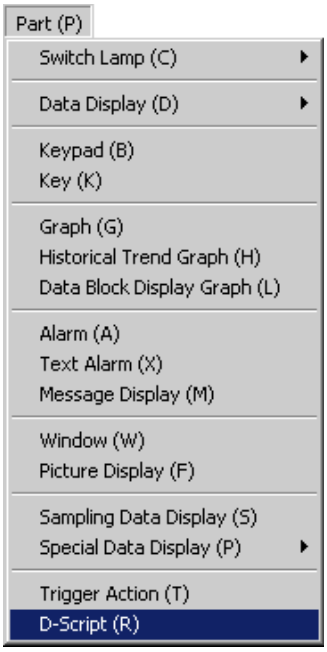
สคริปต์ที่เสร็จแล้ว

Execution Expression | Enlarge Execution Expression | Address Input

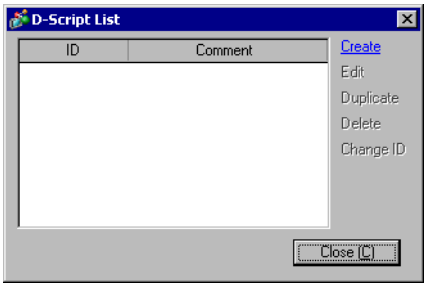
```
0001 if([w:[PLC1]D00200]>=70)                          //When temperature is 70( or more
0002 {
0003     [w:[#INTERNAL]LS0302]=100                     //Substitutes 100, the No. of the 70
0004                                                   //( or more warning message screen.
0005     [w:[#INTERNAL]LS0300]=[w:[#INTERNAL]LS0300]+1 //Increments error count.
0006 }
0007 endif
0008
0009 if([w:[PLC1]D00200]<=30)                          //When temperature is 30( or less
0010 {
0011     [w:[#INTERNAL]LS0302]=101                     //Substitutes 101, the No. of the 30
0012                                                   //( or less warning message screen.
0013     [w:[#INTERNAL]LS0301]=[w:[#INTERNAL]LS0301]+1 //Increments error count.
0014 }
0015 endif
0016
0017
```

ขั้นตอนการสร้าง

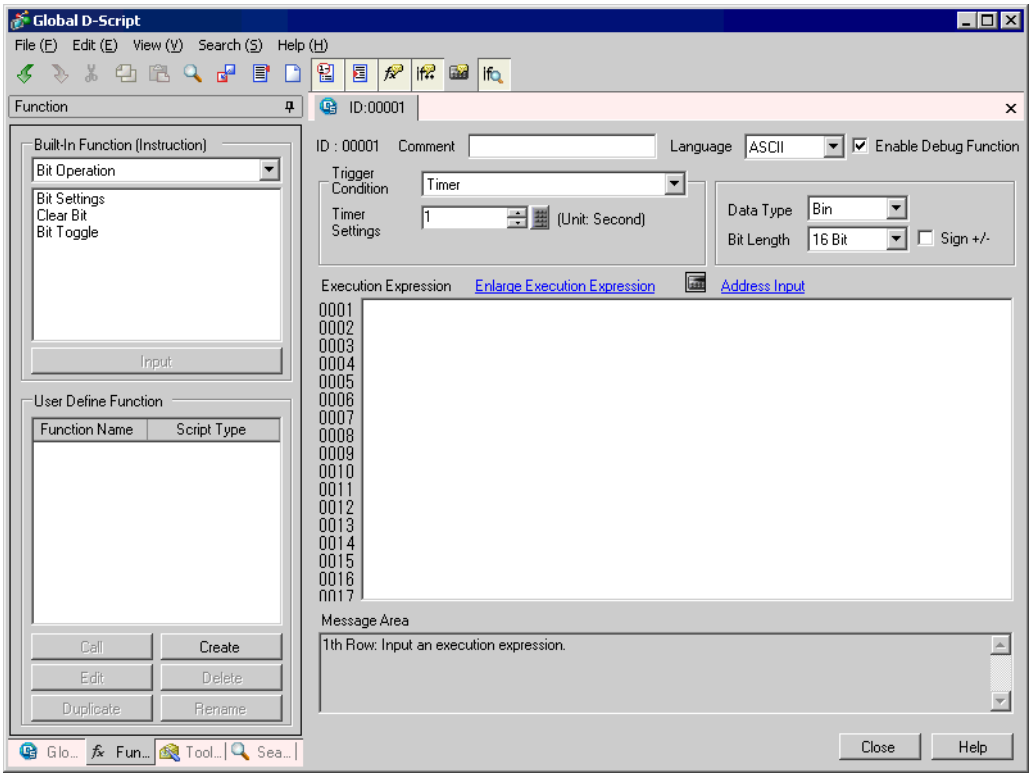
1 เลือกเมนู [Part] - คำสั่ง [D-Script] หรือคลิก 



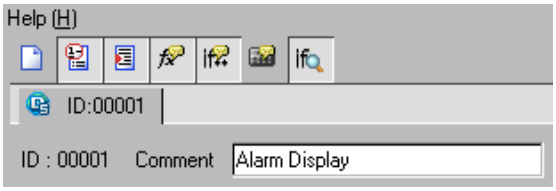
2 คลิก [Create] หากลงทะเบียนสคริปต์นี้ไว้แล้ว หมายเลข ID จะปรากฏขึ้น



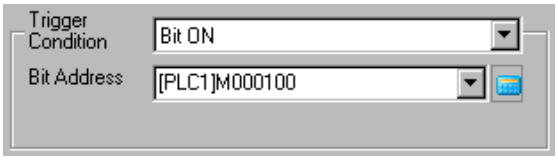
3 กล่องโต้ตอบ [D-Script] จะปรากฏขึ้น



4 ระบุคำอธิบาย ป้อน “Alarm Display”



5 ตั้งเงื่อนไขการทำงานของสคริปต์ (ทริกเกอร์) เลือกบิตขาขึ้นดังนี้ แล้วตั้งค่า [Bit Address] เป็น M000100



6 สร้างโปรแกรมในพื้นที่ทำงาน โดยป้อนฟังก์ชัน ข้อความคำสั่ง และนิพจน์ การตั้งค่าจะเสร็จสมบูรณ์

Execution Expression | Enlarge Execution Expression | Address Input

```
0001 if([w:[PLC1]D00200]>=70)                          //When temperature is 70( or more
0002 {
0003     [w:[#INTERNAL]LS0302]=100                     //Substitutes 100, the No. of the 70
0004                                                   //( or more warning message screen.
0005     [w:[#INTERNAL]LS0300]=[w:[#INTERNAL]LS0300]+1 //Increments error count.
0006 }
0007 endif
0008
0009 if([w:[PLC1]D00200]<=30)                          //When temperature is 30( or less
0010 {
0011     [w:[#INTERNAL]LS0302]=101                     //Substitutes 101, the No. of the 30
0012                                                   //( or less warning message screen.
0013     [w:[#INTERNAL]LS0301]=[w:[#INTERNAL]LS0301]+1 //Increments error count.
0014 }
0015 endif
0016
0017
```

20.5 การสื่อสารกับอุปกรณ์ต่อพ่วงที่สคริปต์ปกติไม่รองรับ

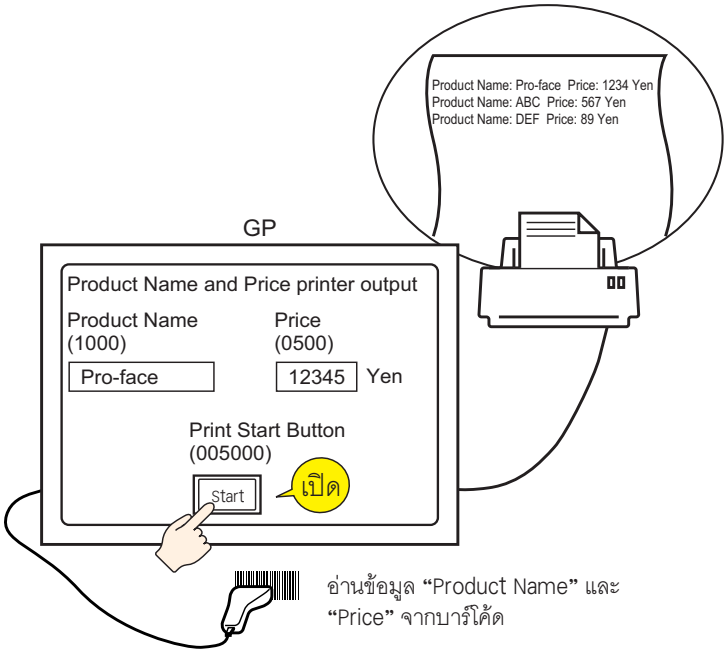
หมายเหตุ

- โปรดอ่านรายละเอียดจากคำแนะนำในการตั้งค่า

☞ “20.8.1 คำแนะนำในการตั้งค่าทั่วไป (D-Script)” (หน้า 20-50)

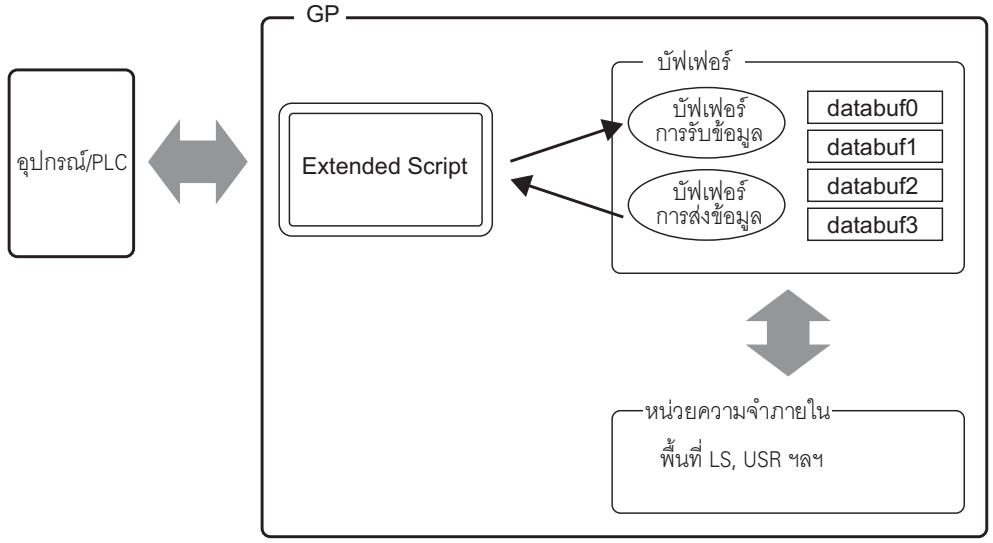
■ การดำเนินการ

สร้าง Extended Script เพื่อส่งข้อมูลที่อ่านได้จากบาร์โค้ดซึ่งเชื่อมต่อกับพอร์ต USB ไปยังเครื่องพิมพ์ที่ต่อกับพอร์ต COM1แบบอนุกรม



■ โครงสร้างของ Extended Script

Extended Script คือ สคริปต์ที่ใช้สื่อสารระหว่างพอร์ตอนุกรมภายในของ GP กับอุปกรณ์อินพุต/เอาต์พุตที่เชื่อมต่ออยู่โดยเฉพาะ
สำหรับการจัดการข้อมูล Extended Script ตามที่แสดงในรูปภาพด้านล่างนี้ ข้อมูลจะถูกเก็บไว้ใน databuf0 ถึง databuf3 โดยผ่านทางบัฟเฟอร์การส่ง/รับข้อมูล databuf จะไม่ถูกแบ่งตามตำแหน่ง ดังนั้น เมื่อจะแก้ไขข้อมูลจากอุปกรณ์/PLC โปรดเก็บข้อมูลดังกล่าวไว้ในหน่วยความจำภายในก่อนทำการแก้ไข



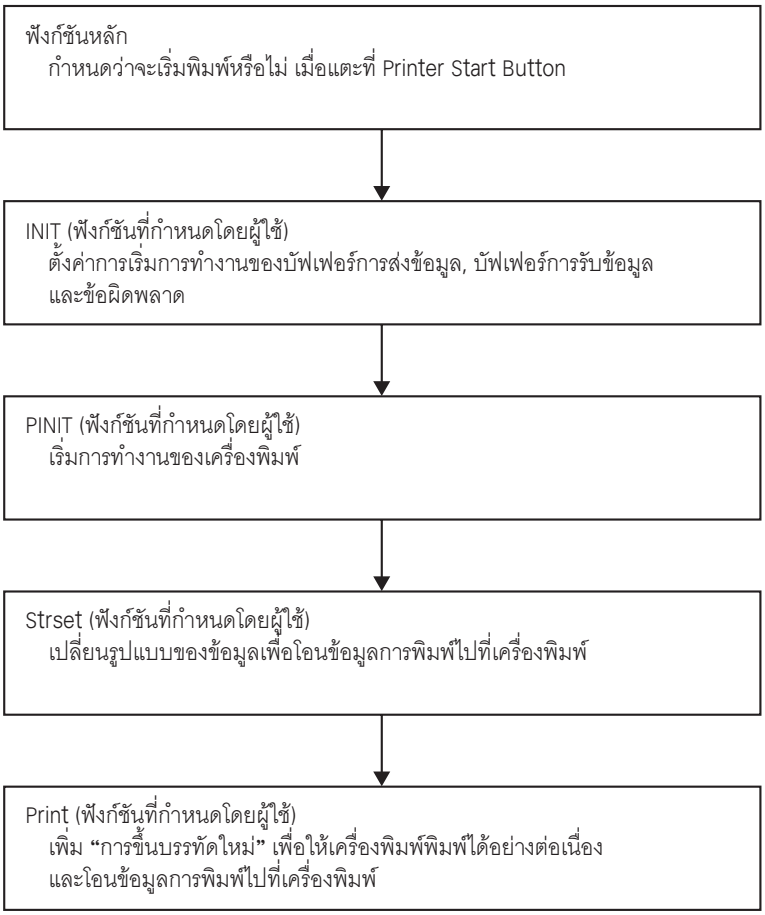
บัฟเฟอร์การรับข้อมูล/บัฟเฟอร์การส่งข้อมูล

สำหรับการสื่อสารกับอุปกรณ์/PLC บัฟเฟอร์นี้จะทำหน้าที่เป็นพื้นที่หน่วยความจำบิตซึ่งจะคอยแยกแยะข้อมูลที่ได้รับและข้อมูลที่ส่งในแบบเรียลไทม์

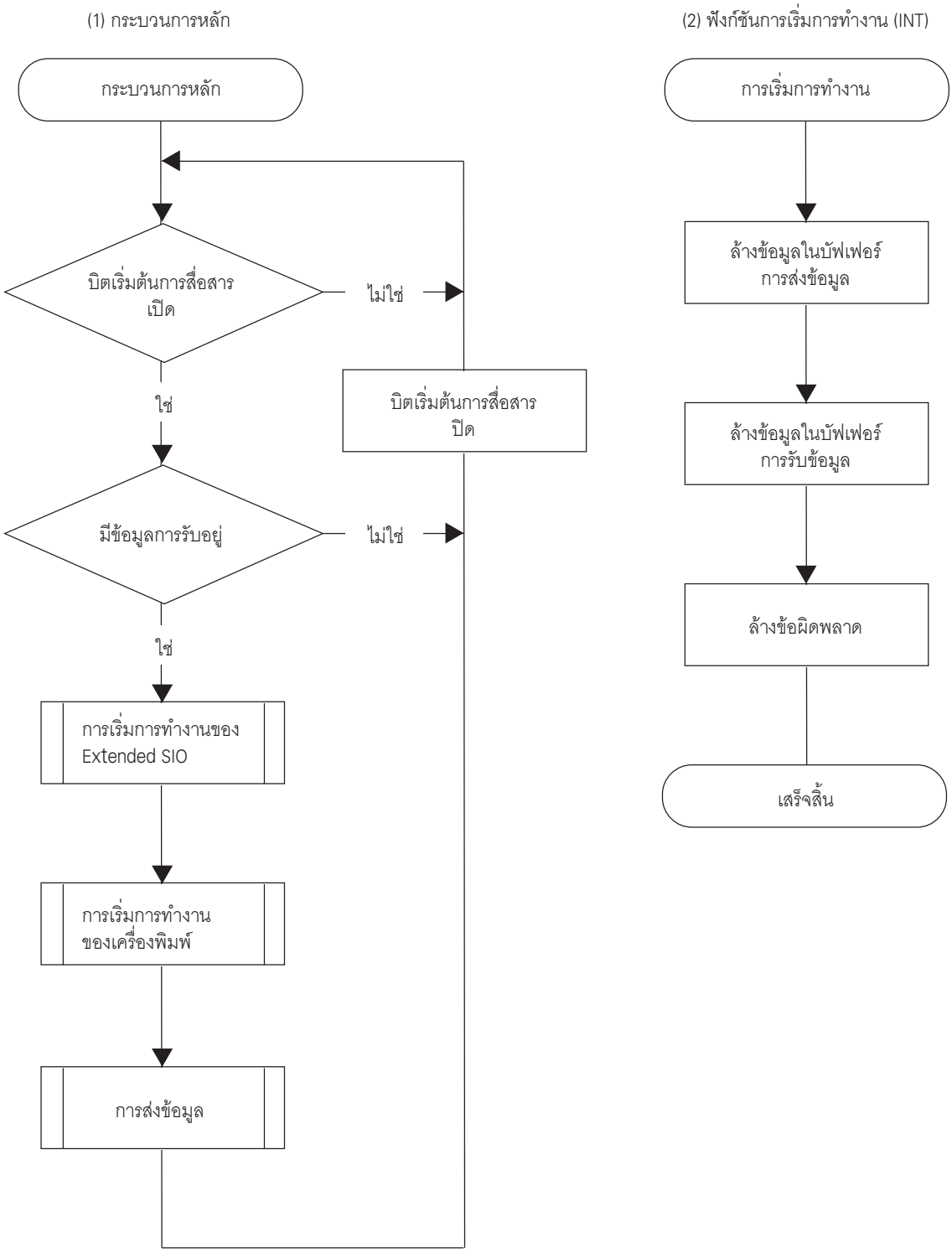
databuf0 ถึง databuf3

บัฟเฟอร์เหล่านี้คือพื้นที่หน่วยความจำแบบไบต์ (8 บิต) ซึ่งทำหน้าที่เป็นตำแหน่งจัดเก็บข้อมูล โดยแต่ละบัฟเฟอร์มีขนาดเท่ากับ 1 KB

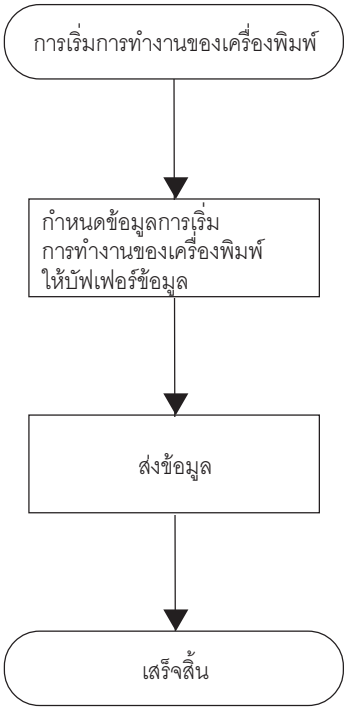
■ ขั้นตอนการสร้างสคริปต์



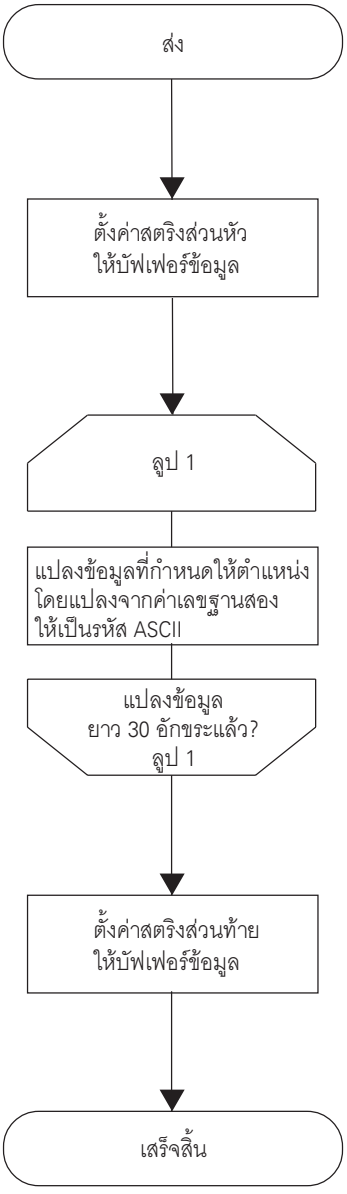
■ ผังการทำงาน


(1) กระบวนการหลัก
กระบวนการหลัก
บิตเริ่มต้นการสื่อสาร
เปิด
ไม่ใช่
ใช่
บิตเริ่มต้นการสื่อสาร
ปิด
มีข้อมูลการรับอยู่
ไม่ใช่
ใช่
การเริ่มการทำงานของ
Extended SIO
การเริ่มการทำงาน
ของเครื่องพิมพ์
การส่งข้อมูล
(2) ฟังก์ชันการเริ่มการทำงาน (INT)
การเริ่มการทำงาน
ล้างข้อมูลในบัฟเฟอร์
การส่งข้อมูล
ล้างข้อมูลในบัฟเฟอร์
การรับข้อมูล
ล้างข้อผิดพลาด
เสร็จสิ้น

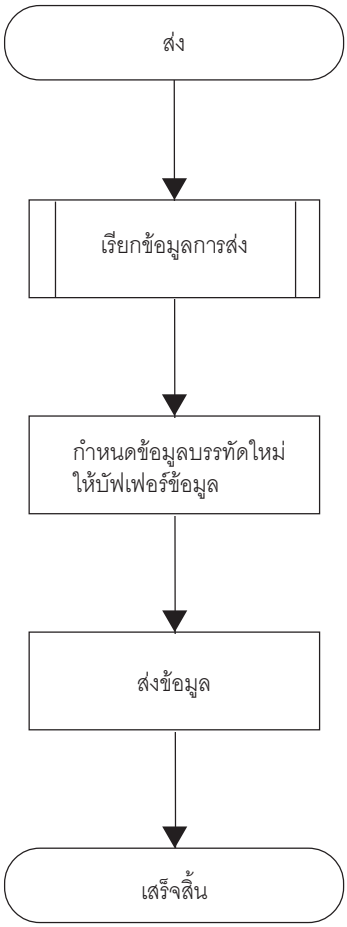
(3) ฟังก์ชันการเริ่มการทำงานของเครื่องพิมพ์ (PRINT)



(4) ฟังก์ชันสตริง (Strset)

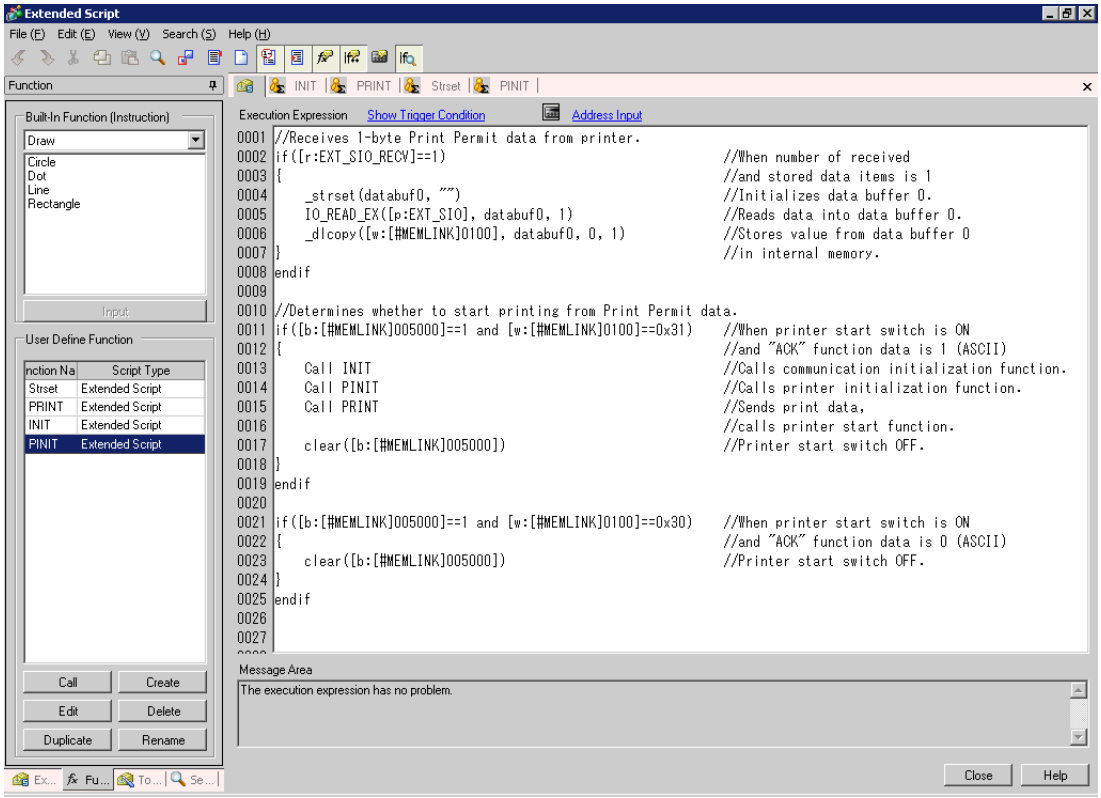


(5) ฟังก์ชันส่ง (Print)



◆ **ฟังก์ชันหลัก**

สคริปต์ที่เสร็จแล้ว



ข้อมูลสรุปของฟังก์ชัน

เมื่อปุ่ม Start ของเครื่องพิมพ์ (หน่วยความจำภายใน 005000) มีสถานะ ON ให้พิจารณาว่าจะเริ่มพิมพ์จากไบต์แรกของข้อมูล Print Permit หรือไม่

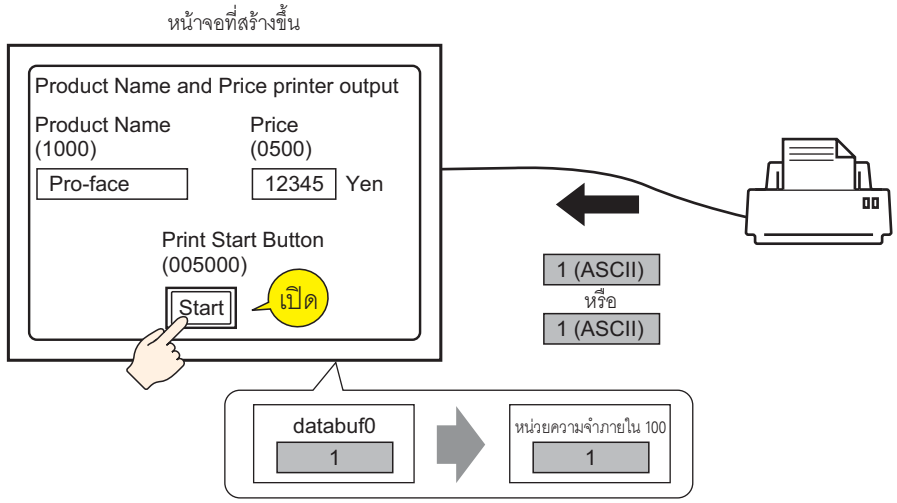
ข้อมูล Print Permit จะทำงานต่อไปนี้เพื่อเป็นตัวอย่างข้อมูลจำเพาะของเครื่องพิมพ์

Print Preparation OK: Send 0x31 (ASCII code “1”) to the device/PLC.

Print Preparation Invalid: Send 0x30 (ASCII code “0”) to the device/PLC.

GP จะรับข้อมูล Print Permit ใน databuf0 แล้วย้ายข้อมูลนี้ไปที่หน่วยความจำภายใน 100 ซึ่งเข้าถึงได้ง่ายกว่าด้วยการจัดการสคริปต์ดังต่อไปนี้

เมื่อหน่วยความจำภายใน 100 ได้รับข้อมูล 0x31 (ASCII code “1”) การพิมพ์จะเริ่มขึ้น เมื่อได้รับข้อมูล 0x30 (ASCII code “0”) GP จะกลับไปยังจุดเริ่มต้นแล้วปฏิบัติขั้นตอนนี้ซ้ำจนกว่าจะได้รับข้อมูล 0x31



◆ INIT (**ฟังก์ชันที่กำหนดโดยผู้ใช้**)

สคริปต์ที่เสร็จแล้ว

```
Execution Expression  Enlarge Execution Expression  Address Input
0001 [c:EXT_SIO_CTRL00]=1    //Clears send buffer.
0002 [c:EXT_SIO_CTRL01]=1    //Clears receive buffer.
0003 [c:EXT_SIO_CTRL02]=1    //Clears error.
0004
```

ข้อมูลสรุปของฟังก์ชัน

ตั้งค่าการเริ่มการทำงานของบัฟเฟอร์การส่งข้อมูล บัฟเฟอร์การรับข้อมูล และข้อผิดพลาด

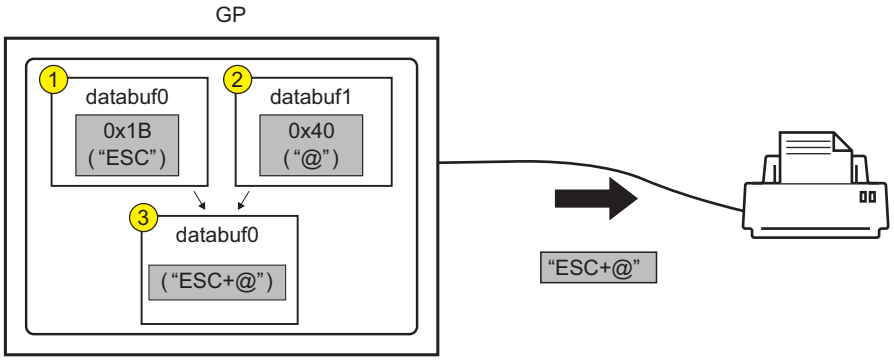
◆ PINIT (**ฟังก์ชันที่กำหนดโดยผู้ใช้**)

สคริปต์ที่เสร็จแล้ว

```
Execution Expression  Enlarge Execution Expression  Address Input
0001 //Printer initialization ( ESC/P [ESC + @] )
0002
0003 _strset(databuf0, "")           //Clears data buffer 0.
0004 _strset(databuf0, 0x1B)         //Sets ASCII code for "ESC".
0005 _strset(databuf1, "")           //Clears data buffer 1.
0006 _strset(databuf1, 0x40)         //Sets ASCII code for "@".
0007 _strcat(databuf0, databuf1)     //Concatenates data buffer 1 to end of data buffer 0.
0008 _strlen([t:0000], databuf0)     //Converts data length to numerical value
0009                                 //and stores it in temporary address.
0010
0011 //Sends data from serial port.
0012
0013 IO_WRITE_EX([p:EXT_SIO], databuf0, [t:0000])    //Sends amount of buffer 0 data specified by value
0014                                                 //of temporary address.
0015
```

ข้อมูลสรุปของฟังก์ชัน

เริ่มการทำงานของเครื่องพิมพ์ ส่งคำสั่ง ESC/P “ESC+@” ไปยังเครื่องพิมพ์



◆ Strset (**ฟังก์ชันที่กำหนดโดยผู้ใช้**)

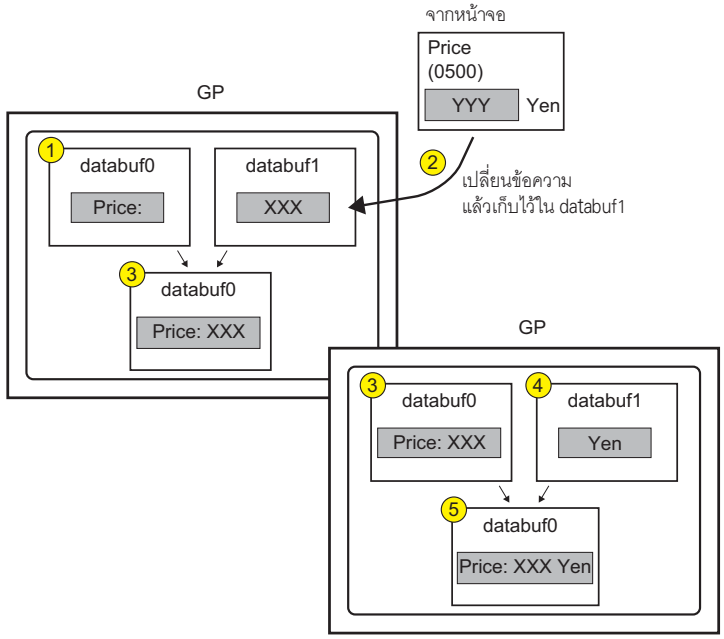
สคริปต์ที่เสร็จแล้ว

Execution Expression Show Trigger Condition Address Input

```
0001 //Appends "Price:" and "Yen" text strings.
0002 _strset(databuf0, "")                          //Initializes data buffer 0.
0003 _strset(databuf0, "価格 : ")                    //"Price:" Stores text string in data buffer 0.
0004 _bin2decasc(databuf0, [w:[#MEMLINK]0500])      //Converts numerical value to text string
0005                                                //and stores it in data buffer 1.
0006 _strcat(databuf0, databuf1)                    //Concatenates data buffer 1 to end of data buffer 0.
0007 _strset(databuf1, "")                          //Initializes data buffer 1.
0008 _strset(databuf1, "円")                         //"Yen" Stores text string in data buffer 1.
0009 _strcat(databuf0, databuf1)                    //Concatenates data buffer 1 to end of data buffer 0.
0010
0011 //Temporary address initialization
0012 [t:0001]=0
0013 [t:0002]=0
0014
0015 //Re-stores text string stored sequentially in word units in internal memory, in byte units (30 characters of data).
0016 loop()
0017 {
0018     [w:[#MEMLINK]2000]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] >> 8       //Stores higher-order byte in place of lower-order byte.
0019     [w:[#MEMLINK]2001]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] & 0xFF     //Erases higher-order byte and stores result
0020                                                                        //in next address.
0021     [t:0001]=[t:0001]+1                                                //Address offset + 1
0022     [t:0002]=[t:0002]+2                                                //Address offset + 2
0023     if([t:0001]==15)                                                   //Loop ends when process of storing each 2-byte value
0024     {                                                                  //in 2 words has repeated 15 times.
0025         break
0026     }
0027     endif
0028 }
0029 endloop
0030 _ldcopy(databuf2, [w:[#MEMLINK]2000], 30)      //Stores data in internal memory addresses 2000 to 2030 in data buffer
0031                                                //as text string.
0032 //Appends "Product Name:" text string.
0033 _strset(databuf1, "")                          //Initializes data buffer 1.
0034 _strset(databuf1, "品名 : ")                    //"Product Name:" Stores text string in data buffer 1.
0035 _strcat(databuf1, databuf2)                    //Concatenates data buffer 2 to end of data buffer 1.
0036
0037 //Appends price character string to product name.
0038 _strcat(databuf1, databuf0)                    //Concatenates value of data buffer 0 to data buffer 1.
```

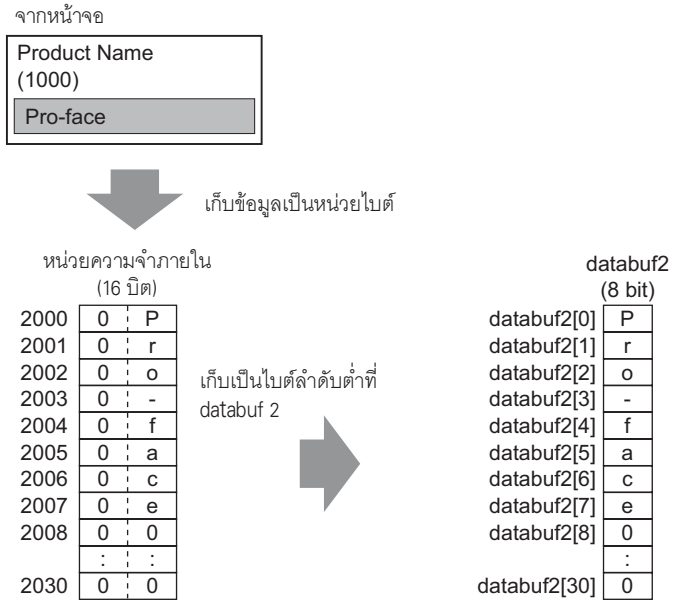
ข้อมูลสรุปของฟังก์ชัน

1 เพิ่มข้อความ “Price:” และ “Yen” ลงในข้อมูลราคาที่จัดเก็บอยู่ในตำแหน่ง 0500 ของหน่วยความจำภายใน

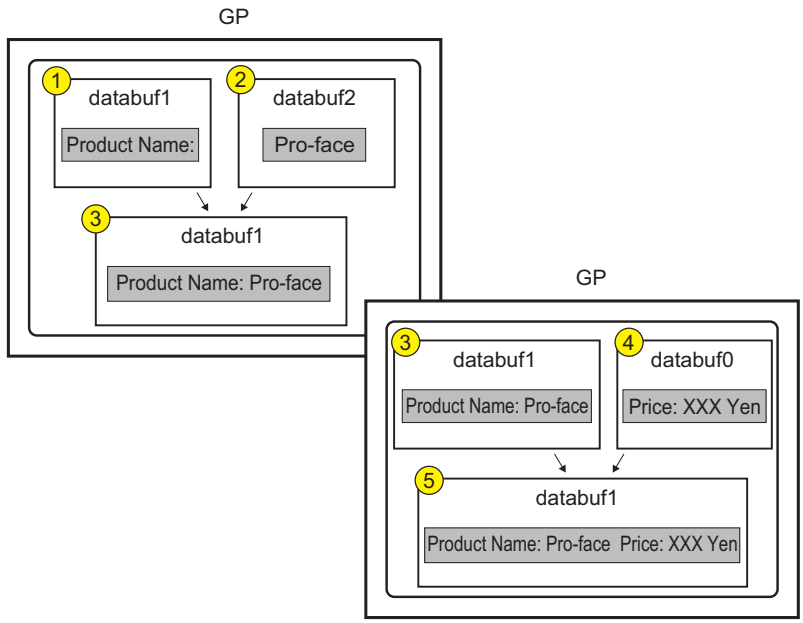


2 เปลี่ยนรูปแบบข้อมูลเพื่อส่งข้อมูลการพิมพ์ไปยังเครื่องพิมพ์ แบ่งข้อมูลสตริง (Product Name) ที่จัดเก็บไว้ตามลำดับในตำแหน่ง 1000 ของหน่วยความจำภายในออกเป็นหน่วยไบต์ แล้วเก็บข้อมูลดังกล่าวลงในตำแหน่ง 2000 ถึง 2030 ของหน่วยความจำภายใน โดยเก็บเป็นข้อมูลสตริงไบต์ล่าง
ใช้ฟังก์ชัน _ldcopy และเก็บข้อมูลใน databuf2 ตามลำดับไบต์ล่างสุดของตำแหน่งเวิร์ดแบบต่อเนื่อง

หมายเหตุ
- ฟังก์ชัน _ldcopy จะนำข้อมูลไปเก็บไว้เป็นเวิร์ด และเก็บเฉพาะไบต์ลำดับต่ำกว่าในบัฟเฟอร์ โดยจะไม่สนใจข้อมูลไบต์ที่มีลำดับสูงกว่า



3 เพิ่มข้อความ “Product Name:” และ “Price” ใน databuf2



◆ Print (**ฟังก์ชันที่กำหนดโดยผู้ใช้**)

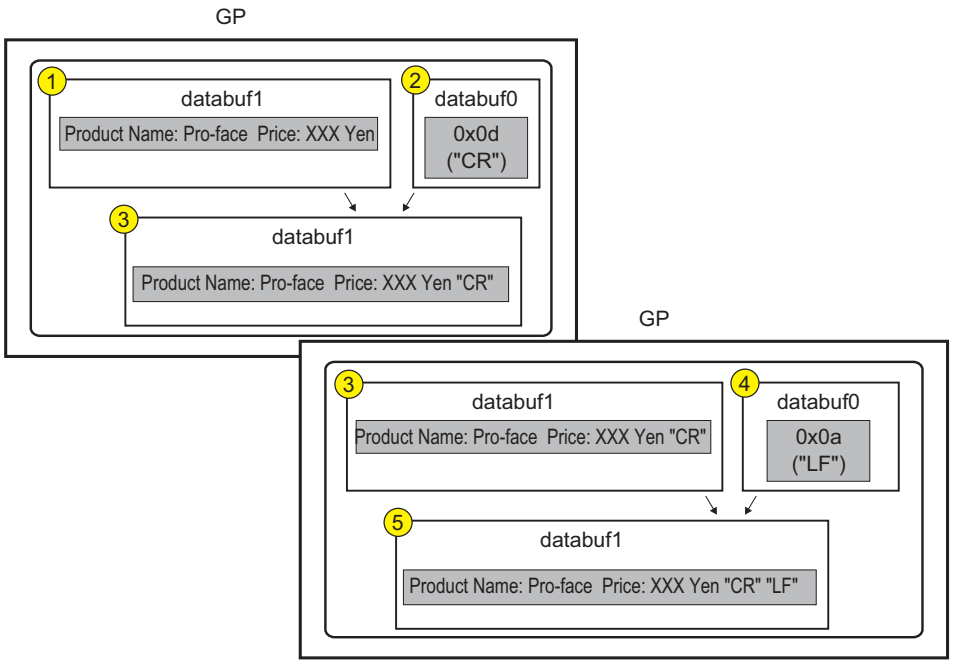
สคริปต์ที่เสร็จแล้ว

Execution Expression Enlarge Execution Expression Address Input

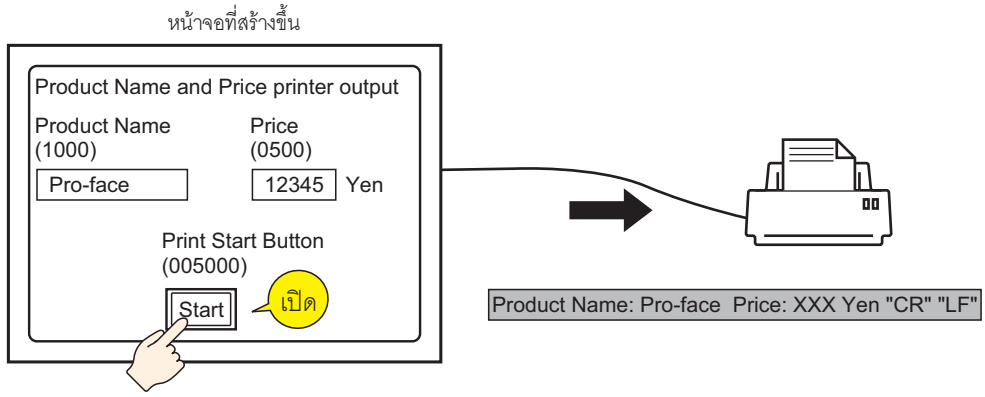
```
0001 Call Strset                          //Calls print data function.
0002 _strset(databuf0, "")                //Clears data buffer 1.
0003
0004 //Printing and delimiter (line feed + carriage return)
0005
0006 _strset(databuf0, 0x0d)              //Prints and returns to head of line.
0007 _strcat(databuf1, databuf0)          //Concatenates data buffer 1 to end of data buffer 0.
0008 _strset(databuf0, "")                //Clears data buffer 1.
0009 _strset(databuf0, 0x0a)              //Sends line feed to move to next line.
0010 _strcat(databuf1, databuf0)          //Concatenates data buffer 1 to end of data buffer 0.
0011
0012 _strlen([t:0000], databuf1)          //Converts data length to numerical value
0013                                      //and stores it in temporary address.
0014 //Sends data from serial port.
0015
0016 IO_WRITE_EX([p:EXT_SIO], databuf1, [t:0000])    //Sends amount of buffer 0 data
0017                                                 //specified by value of temporary address.
0018
```

ข้อมูลสรุปของฟังก์ชัน

1 **เพิ่ม “การขึ้นบรรทัดใหม่” เพื่อให้เครื่องพิมพ์พิมพ์ได้อย่างต่อเนื่อง**



2 ตั้งค่าข้อมูลการพิมพ์ให้เครื่องพิมพ์

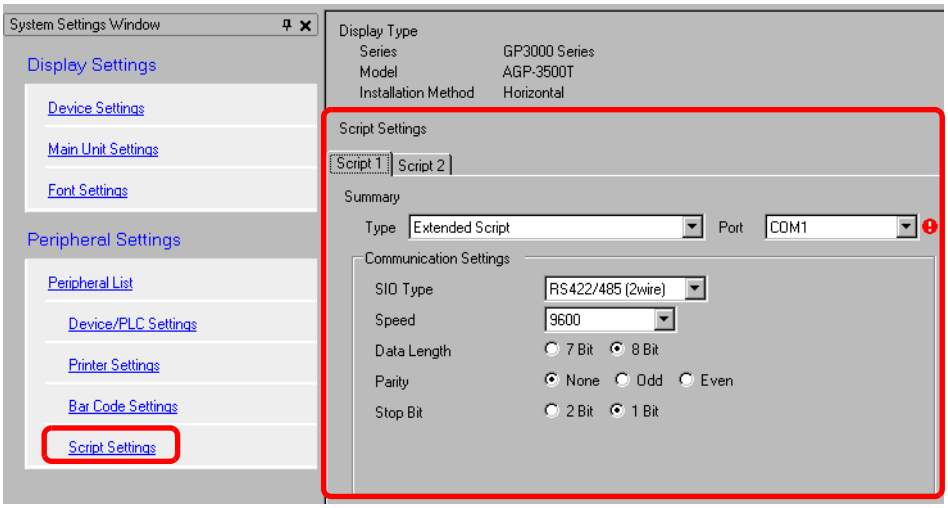


■ คำสั่งที่ใช้

คำสั่ง	ข้อมูลสรุปของฟังก์ชัน
if ( )	เมื่อเงื่อนไขภายในวงเล็บ “( )” ที่อยู่ตามหลัง “if” เป็นจริง ระบบจะดำเนินกระบวนการที่อยู่หลังข้อความคำสั่ง “if ( )”
Label Settings [r:EXT_SIO_RECV]	แสดงปริมาณข้อมูล (จำนวนไบต์) ที่ได้รับในคราวนั้น ๆ ขนาดข้อมูลที่ได้รับจะมีคุณสมบัติเป็นแบบอ่านอย่างเดียว
Equivalent (==)	เป็นจริงหาก N1 เท่ากับ N2 (N1 = N2)
Text Settings (_strset)	จัดเก็บสตริงแบบตายตัวในบัฟเฟอร์ข้อมูล
Extended Receive (IO_READ_EX)	รับข้อมูลตามขนาดที่ระบุในขนาดข้อมูลที่ได้รับ (ไบต์) จาก Extended SIO แล้วเก็บไว้ในบัฟเฟอร์ข้อมูล
From Data Buffer to Internal Device (_dlcopy)	ข้อมูลสตริงแต่ละไบต์ที่เก็บไว้ในออฟเซ็ตของบัฟเฟอร์ข้อมูล จะถูกคัดลอกไปยังพื้นที่ LS ตามจำนวนสตริง
Label Settings [c:EXT_SIO_CTRL**]	ตัวแปรควบคุมนี้ใช้เพื่อล้างข้อมูลในบัฟเฟอร์การส่งข้อมูล บัฟเฟอร์การรับข้อมูล และสถานะข้อผิดพลาด
Connect Text (_strcat)	เชื่อมสตริงอักขระหรือรหัสอักขระด้วยบัฟเฟอร์ข้อความ
Text Length (_strlen)	รับความยาวของสตริงที่จัดเก็บไว้
Extended Send (IO_WRITE_EX)	ส่งข้อมูลในบัฟเฟอร์ข้อมูลด้วย Extended SIO ตามขนาดของจำนวนไบต์ที่ส่ง
Assignment (=)	กำหนดค่าฝั่งขวาให้แก่ค่าฝั่งซ้าย
Addition (+)	เพิ่มค่าคงที่ให้แก่ข้อมูลของอุปกรณ์ชนิดเวิร์ด
Numeric Value Decimal String Conversion (_bin2decasc)	ใช้ฟังก์ชันนี้เพื่อแปลงจำนวนเต็มให้เป็นสตริงเลขฐานสิบ
From Internal Device To Data Buffer (_ldcopy)	ข้อมูลของสตริงที่จัดเก็บอยู่ในพื้นที่ LS จะถูกคัดลอกไปยังบัฟเฟอร์ข้อมูลตามจำนวนสตริงในการถ่ายโอนข้อมูลแบบไบต์ต่อไบต์

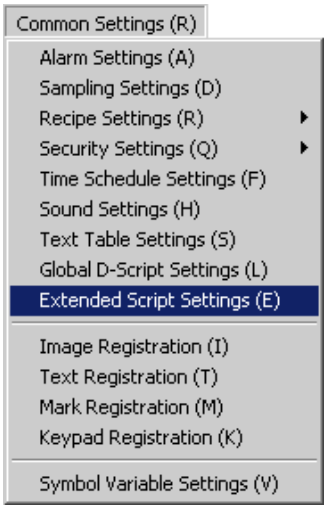
■ ขั้นตอนการสร้าง

1 กำหนด Extended Script ที่จะใช้ในการสื่อสาร คลิก [Project] - [System Settings] - [Script Settings] ตรวจสอบให้แน่ใจว่าได้ตั้งค่า [Type] เป็น [Extended Script] แล้ว

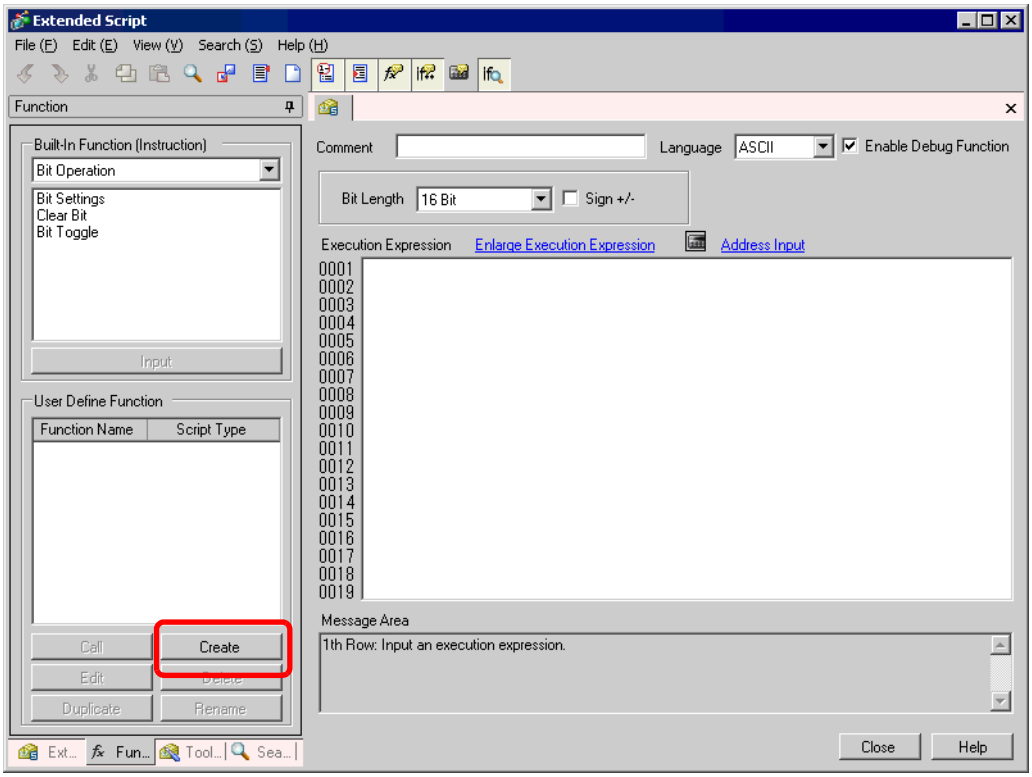


แท็บสำหรับตั้งค่าสคริปต์มีด้วยกัน 2 แท็บ รูปภาพด้านบนนี้ใช้แท็บ [Script 1]
ตั้งค่า [Port] เป็น COM1 หรือ COM2 แล้วตั้งค่า [Communication Settings] ให้ตรงกับ Extended SIO

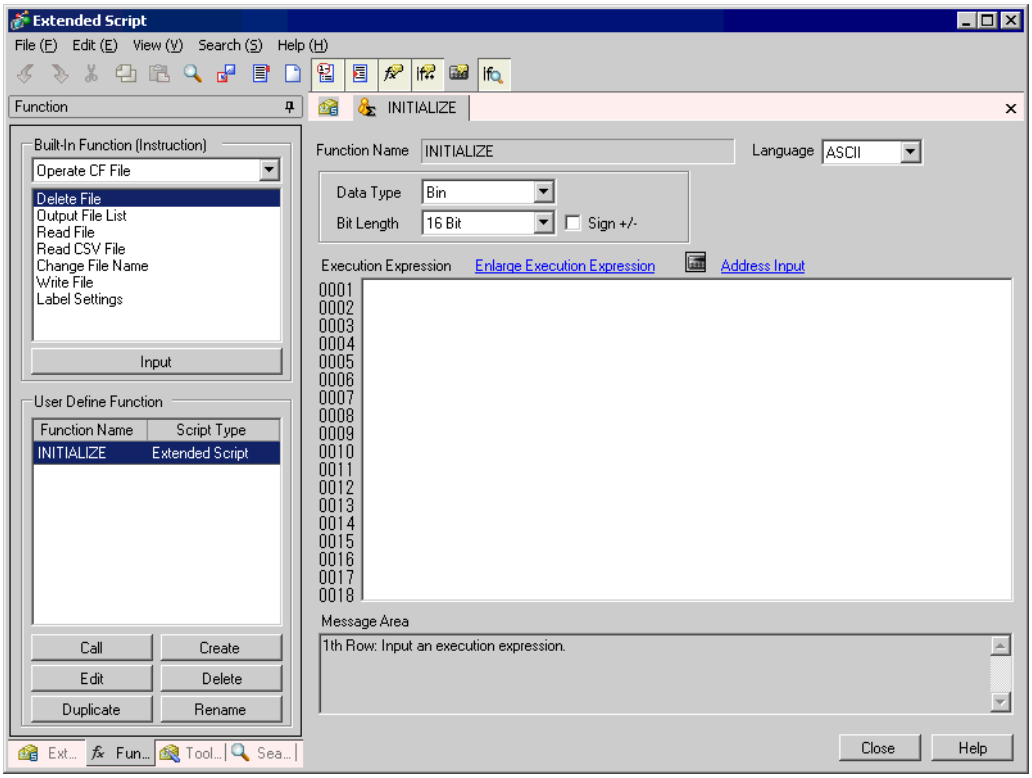
2 เลือก [Extended Script Settings] จากเมนู [Common Settings]



3 ลงทะเบียน “INIT” เป็นฟังก์ชันที่กำหนดโดยผู้ใช้ คลิกแท็บ [Function] แล้วคลิกปุ่ม [Create] ในกรอบ User Define Function



4 ป้อน [INIT] เป็นชื่อฟังก์ชัน คลิก [OK] หน้าจอต่อไปนี้จะปรากฏขึ้น



5 สร้างสคริปต์ใน Execution Expression ด้วยการป้อนคำสั่ง ข้อความคำสั่ง และค่าคงที่

Execution Expression | Enlarge Execution Expression | Address Input

```
0001 [c:EXT_SIO_CTRL00]=1    //Clears send buffer.
0002 [c:EXT_SIO_CTRL01]=1    //Clears receive buffer.
0003 [c:EXT_SIO_CTRL02]=1    //Clears error.
0004
```

6 ปฏิบัติตามขั้นตอนที่ผ่านมาเพื่อลงทะเบียน “PINIT” เป็นฟังก์ชันที่กำหนดโดยผู้ใช้ ป้อน [PINIT] เป็นชื่อฟังก์ชัน แล้วสร้างสคริปต์ต่อไปนี้ใน Execution Expression

Execution Expression | Enlarge Execution Expression | Address Input

```
0001 //Printer initialization ( ESC/P 「ESC + @」 )
0002
0003 _strset(databuf0, "")           //Clears data buffer 0.
0004 _strset(databuf0, 0x1B)         //Sets ASCII code for "ESC".
0005 _strset(databuf1, "")           //Clears data buffer 1.
0006 _strset(databuf1, 0x40)         //Sets ASCII code for "@".
0007 _strcat(databuf0, databuf1)     //Concatenates data buffer 1 to end of data buffer 0.
0008 _strlen([t:0000], databuf0)     //Converts data length to numerical value
0009                                 //and stores it in temporary address.
0010
0011 //Sends data from serial port.
0012
0013 IO_WRITE_EX([p:EXT_SIO], databuf0, [t:0000])    //Sends amount of buffer 0 data specified by value
0014                                                  //of temporary address.
0015
```

7 ปฏิบัติตามขั้นตอนที่ผ่านมาเพื่อลงทะเบียน “Strset” เป็นฟังก์ชันที่กำหนดโดยผู้ใช้ ป้อน [Strset] เป็นชื่อฟังก์ชัน แล้วสร้างสคริปต์ต่อไปนี้ใน Execution Expression

Execution Expression | Show Trigger Condition | Address Input

```
0001 //Appends "Price:" and "Yen" text strings.
0002 _strset(databuf0, "")                      //Initializes data buffer 0.
0003 _strset(databuf0, "価格 : ")               //"Price:" Stores text string in data buffer 0.
0004 _bin2decasc(databuf0, [w:[#MEMLINK]0500])  //Converts numerical value to text string
0005                                            //and stores it in data buffer 1.
0006 _strcat(databuf0, databuf1)                //Concatenates data buffer 1 to end of data buffer 0.
0007 _strset(databuf1, "")                      //Initializes data buffer 1.
0008 _strset(databuf1, "円")                    //"Yen" Stores text string in data buffer 1.
0009 _strcat(databuf0, databuf1)                //Concatenates data buffer 1 to end of data buffer 0.
0010
0011 //Temporary address initialization
0012 [t:0001]=0
0013 [t:0002]=0
0014
0015 //Re-stores text string stored sequentially in word units in internal memory, in byte units (30 characters of data).
0016 loop()
0017 {
0018     [w:[#MEMLINK]2000]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] >> 8       //Stores higher-order byte in place of lower-order byte.
0019     [w:[#MEMLINK]2001]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] & 0xFF     //Erases higher-order byte and stores result
0020                                                                        //in next address.
0021     [t:0001]=[t:0001]+1                                                //Address offset + 1
0022     [t:0002]=[t:0002]+2                                                //Address offset + 2
0023     if([t:0001]==15)                                                   //Loop ends when process of storing each 2-byte value
0024     {                                                                  //in 2 words has repeated 15 times.
0025         break
0026     }
0027     endif
0028 }
0029 endloop
0030 _ldcopy(databuf2, [w:[#MEMLINK]2000], 30)      //Stores data in internal memory addresses 2000 to 2030 in data buffer
0031                                                //as text string.
0032 //Appends "Product Name:" text string.
0033 _strset(databuf1, "")                      //Initializes data buffer 1.
0034 _strset(databuf1, "品名 : ")               //"Product Name:" Stores text string in data buffer 1.
0035 _strcat(databuf1, databuf2)                //Concatenates data buffer 2 to end of data buffer 1.
0036
0037 //Appends price character string to product name.
0038 _strcat(databuf1, databuf0)                //Concatenates value of data buffer 0 to data buffer 1.
```

(1) (2) (3)

8 ปฏิบัติตามขั้นตอนที่ผ่านมาเพื่อลงทะเบียน “Print” เป็นฟังก์ชันที่กำหนดโดยผู้ใช้ ป้อน [Print] เป็นชื่อฟังก์ชัน แล้วสร้างสคริปต์ต่อไปนี้ใน Execution Expression

Execution Expression | Enlarge Execution Expression | Address Input

```
0001 Call Strset                          //Calls print data function.
0002 _strset(databuf0, "")                //Clears data buffer 1.
0003
0004 //Printing and delimiter (line feed + carriage return)
0005
0006 _strset(databuf0, 0x0d)              //Prints and returns to head of line.
0007 _strcat(databuf1, databuf0)          //Concatenates data buffer 1 to end of data buffer 0.
0008 _strset(databuf0, "")                //Clears data buffer 1.
0009 _strset(databuf0, 0x0a)              //Sends line feed to move to next line.
0010 _strcat(databuf1, databuf0)          //Concatenates data buffer 1 to end of data buffer 0.
0011
0012 _strlen([t:0000], databuf1)          //Converts data length to numerical value
0013                                      //and stores it in temporary address.
0014 //Sends data from serial port.
0015
0016 IO_WRITE_EX([p:EXT_SIO], databuf1, [t:0000])    //Sends amount of buffer 0 data
0017                                                 //specified by value of temporary address.
0018
```

9 สร้างสคริปต์หลัก สร้างสคริปต์ต่อไปนี้ใน Execution Expression การตั้งค่าจะเสร็จสมบูรณ์

Execution Expression | Show Trigger Condition | Address Input

```
0001 //Receives 1-byte Print Permit data from printer.
0002 if([r:EXT_SIO_RECV]==1)                                    //When number of received
0003 {                                                          //and stored data items is 1
0004     _strset(databuf0, "")                                  //Initializes data buffer 0.
0005     IO_READ_EX([p:EXT_SIO], databuf0, 1)                   //Reads data into data buffer 0.
0006     _dlcopy([w:[#MEMLINK]0100], databuf0, 0, 1)            //Stores value from data buffer 0
0007 }                                                          //in internal memory.
0008 endif
0009
0010 //Determines whether to start printing from Print Permit data.
0011 if([b:[#MEMLINK]005000]==1 and [w:[#MEMLINK]0100]==0x31)    //When printer start switch is ON
0012 {                                                          //and "ACK" function data is 1 (ASCII)
0013     Call INIT                                              //Calls communication initialization function.
0014     Call PINIT                                             //Calls printer initialization function.
0015     Call PRINT                                             //Sends print data,
0016                                                            //calls printer start function.
0017     clear([b:[#MEMLINK]005000])                            //Printer start switch OFF.
0018 }
0019 endif
0020
0021 if([b:[#MEMLINK]005000]==1 and [w:[#MEMLINK]0100]==0x30)    //When printer start switch is ON
0022 {                                                          //and "ACK" function data is 0 (ASCII)
0023     clear([b:[#MEMLINK]005000])                            //Printer start switch OFF.
0024 }
0025 endif
```

หมายเหตุ

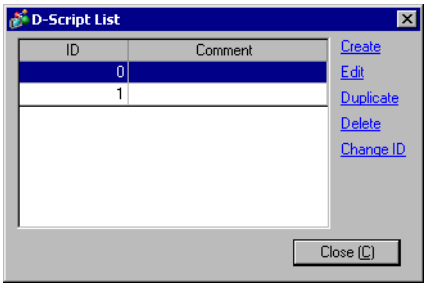
- เมื่อใส่ฟังก์ชันที่กำหนดโดยผู้ใช้ที่สร้างขึ้นในขั้นที่ 3 ถึงขั้นที่ 9 ลงในสคริปต์หลัก ให้เลือกฟังก์ชันที่จะใส่ แล้วคลิก [Call] ที่แท็บ [Function] ฟังก์ชันจะถูกใส่โดยใช้ “ชื่อฟังก์ชันที่เรียกใช้”

20.6 ขั้นตอนการสร้างสคริปต์

20.6.1 ขั้นตอนการสร้าง D-Scripts/Global D-Scripts

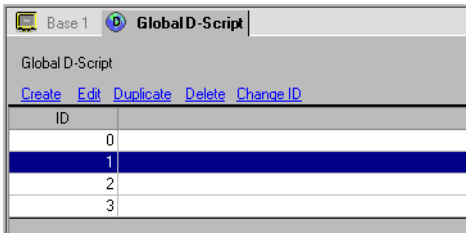
เปิด [D-Script] จากเมนู [Part]

คลิก [Create] หากคุณเข้าใช้งานสคริปต์ที่ลงทะเบียนไว้ก่อนหน้านี้ ให้ระบุหมายเลข ID แล้วคลิก [Edit] หรือดับเบิลคลิกที่แถวหมายเลข ID

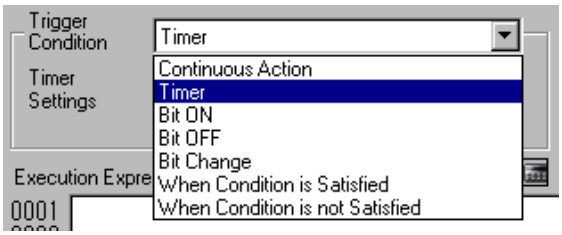


เปิด [Global D-Script Settings] จากเมนู [Common Settings]

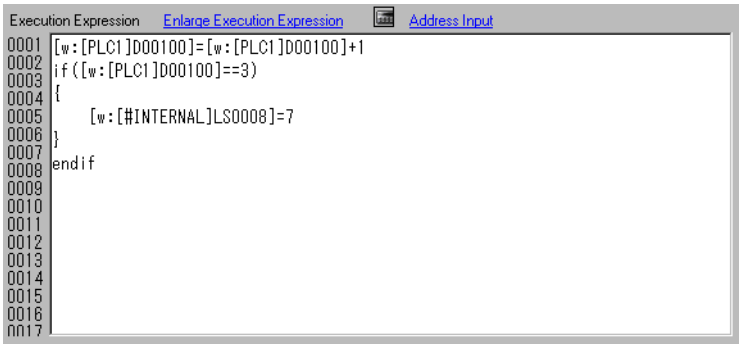
คลิก [Create] หากคุณเข้าใช้งานสคริปต์ที่ลงทะเบียนไว้ก่อนหน้านี้ ให้ระบุหมายเลข ID แล้วคลิก [Edit] หรือดับเบิลคลิกที่แถวหมายเลข ID



ตั้งค่าเงื่อนไขการทริกเกอร์ที่จะทำให้สคริปต์ทำงาน สำหรับข้อมูลเพิ่มเติมเกี่ยวกับฟังก์ชันนี้ โปรดดูที่ “20.7 การตั้งค่าเงื่อนไขการทริกเกอร์” (หน้า 20-44)

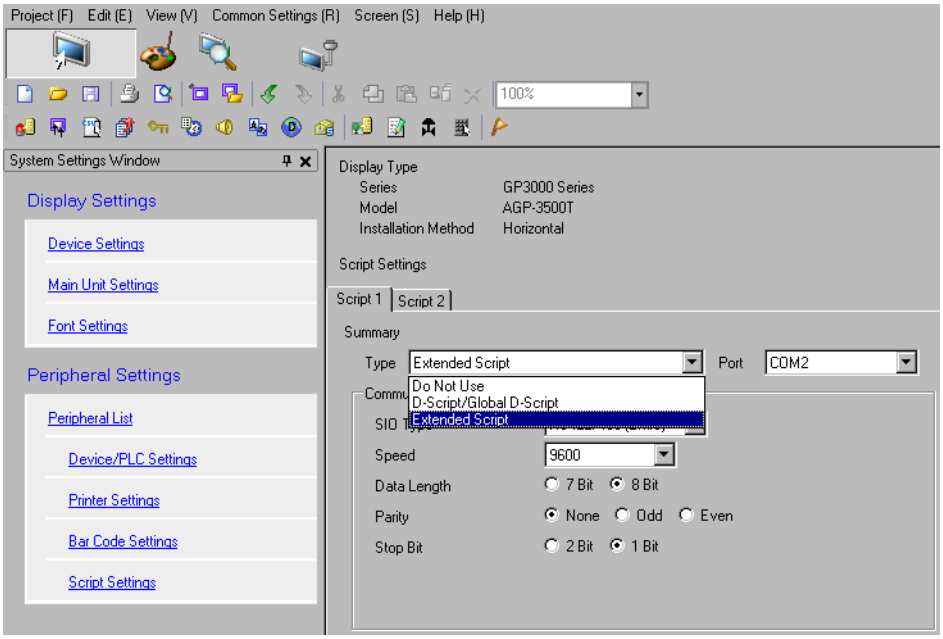


สร้างสคริปต์ (นิพจน์การดำเนินการ) หากต้องการทราบข้อมูลเพิ่มเติมเกี่ยวกับคำสั่งและฟังก์ชันต่าง ๆ โปรดดูที่ “21.13 รายการคำสั่ง” (หน้า 21-98)



20.6.2 ขั้นตอนการสร้าง Extended Script

เปิด [System Settings] จากเมนู [Project] คลิก [Script Settings] กล่องโต้ตอบต่อไปนี้จะปรากฏขึ้น
เมื่อใช้ Extended Script ให้เลือก [Type] เป็น [Extended Script] แล้วกำหนดพอร์ตเชื่อมต่อ

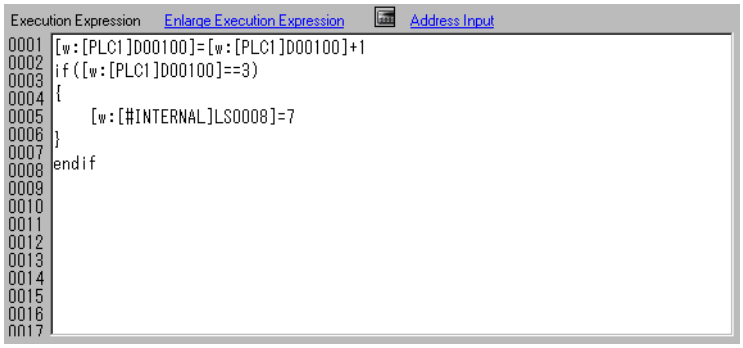


↓

เปิด [Extended Script Settings] จากเมนู [Common Settings]

↓

สร้างสคริปต์ (นิพจน์การดำเนินการ) หากต้องการทราบข้อมูลเพิ่มเติมเกี่ยวกับคำสั่งและฟังก์ชันต่าง ๆ โปรดดูที่ “บทที่ 21 คำสั่งและนิพจน์ ของโปรแกรม” (หน้า 21-1)



20.6.3 ขั้นตอนการตั้งค่าของฟังก์ชันที่กำหนดโดยผู้ใช้

ลงทะเบียนสคริปต์ที่สร้างขึ้นให้เป็นฟังก์ชันที่กำหนดโดยผู้ใช้ซึ่งสคริปต์อื่นสามารถใช้งานได้ โดย D-Script, Global D-Script, หรือ Extended Script สามารถใช้ฟังก์ชันที่ลงทะเบียนไว้ได้

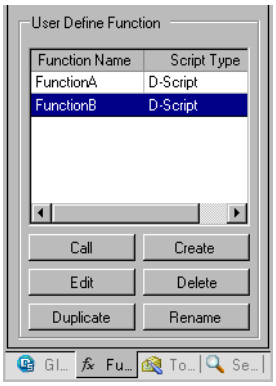
■ ขั้นตอนการตั้งค่า

เมื่อสร้างฟังก์ชันใหม่ที่กำหนดโดยผู้ใช้

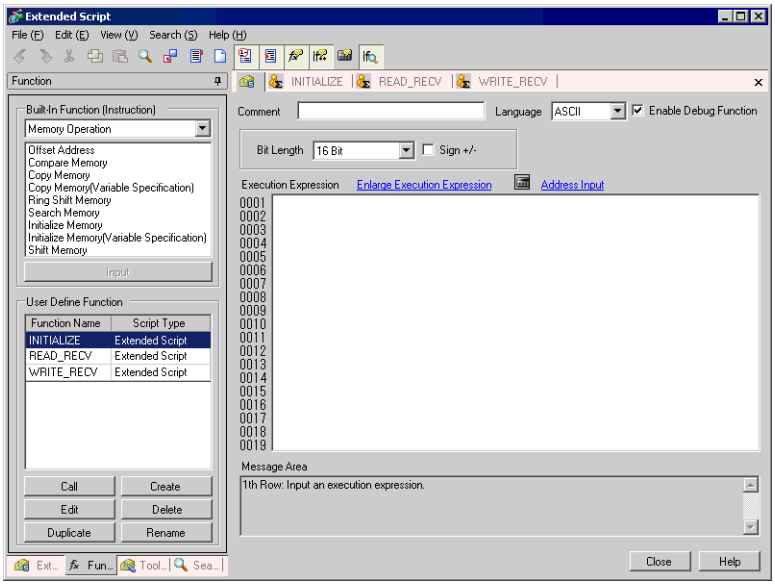
คลิกที่ [Create] กล่องโต้ตอบ User Defined Function จะปรากฏขึ้น

เมื่อแก้ไขฟังก์ชันที่กำหนดโดยผู้ใช้ที่ลงทะเบียนไว้

เลือกฟังก์ชันที่กำหนดโดยผู้ใช้ที่คุณต้องการแก้ไข แล้วคลิก [Edit]
ฟังก์ชันที่กำหนดโดยผู้ใช้ที่ลงทะเบียนไว้จะปรากฏขึ้น



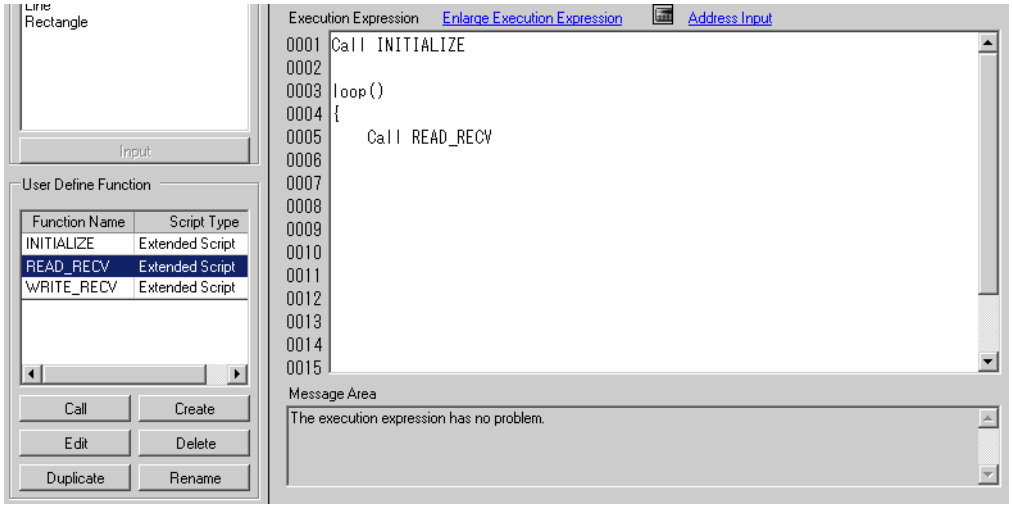
ป้อนชื่อฟังก์ชัน แล้วสร้างสคริปต์ในฟิลด์ Execution Expression คลิก [OK] ฟังก์ชันที่กำหนดโดยผู้ใช้ได้รับการลงทะเบียนแล้ว



หมายเหตุ

- มีข้อจำกัดเกี่ยวกับชื่อที่สามารถใช้เป็นชื่อฟังก์ชันได้ สำหรับข้อมูลเพิ่มเติม โปรดดูที่ “20.9.3 ข้อจำกัดของฟังก์ชันที่กำหนดโดยผู้ใช้” (หน้า 20-59)

เลือกฟังก์ชันที่กำหนดโดยผู้ใช้ที่จะเรียกใช้งาน คลิก [Call] “ชื่อฟังก์ชันที่เรียกใช้” จะถูกใส่ลงในฟิลด์ Execution Expression



ข้อสำคัญ

- เมื่อฟังก์ชันที่กำหนดโดยผู้ใช้เรียกใช้งานสคริปต์อื่น ฟังก์ชันดังกล่าวจะไม่สามารถเรียกฟังก์ชันต่างๆ ที่สร้างขึ้นใน Extended Script ใน D-Scripts หรือ Global D-Scripts ได้

20.7 การตั้งค่าเงื่อนไขการทริกเกอร์

สคริปต์ที่สร้างขึ้นสามารถใช้เงื่อนไขการทริกเกอร์ 7 ชนิดดังต่อไปนี้

การตั้งค่า		คำอธิบาย
Continuous Action		สคริปต์ถูกทริกเกอร์ตามปกติ
Timer		สคริปต์ถูกทริกเกอร์หลังจากครบระยะเวลาที่กำหนด
Bit	Bit ON	เมื่อ GP ตรวจพบบิตที่กำหนดมีค่าเพิ่มจาก 0 เป็น 1 สคริปต์จะถูกทริกเกอร์
	Bit OFF	เมื่อ GP ตรวจพบขอบขาลงของบิตที่กำหนด สคริปต์จะถูกทริกเกอร์
	Bit Change	เมื่อ GP ตรวจพบบิตที่กำหนดมีค่าเพิ่มจาก 0 เป็น 1 หรือลดลงจาก 1 เป็น 0 สคริปต์จะถูกทริกเกอร์
Condition Expression	When Condition is Satisfied	เมื่อ GP ตรวจพบว่านิพจน์ที่กำหนดไว้มีค่าเป็นจริง สคริปต์จะถูกทริกเกอร์
	When Condition is not Satisfied	เมื่อ GP ตรวจพบว่านิพจน์ที่กำหนดไว้มีค่าเป็นเท็จ สคริปต์จะถูกทริกเกอร์

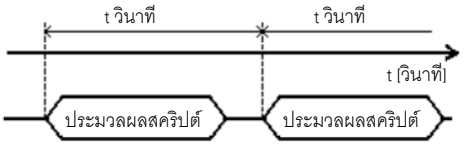
20.7.1 Continuous Action

ทำงานตามระยะเวลาสำหรับการแสดงผล

20.7.2 Timer

■ Timer

สคริปต์จะทำงานทุกครั้งที่ครบระยะเวลาที่กำหนดไว้ คุณสามารถตั้งระยะเวลาของตัวตั้งเวลานี้ได้ตั้งแต่ 1 ถึง 32,767 วินาที



หมายเหตุ

- ในการตั้งเวลาของฟังก์ชันตัวตั้งเวลา ค่าของเวลาจะประกอบด้วยเวลาที่กำหนดไว้ + ข้อผิดพลาดของเวลาที่ใช้สแกนจอแสดงผล นอกจากนี้ ฟังก์ชันตัวตั้งเวลาอาจทำงานได้ช้าขึ้นอยู่กับเวลาที่ใช้วาดรายการหน้าจอหรือเวลาที่ใช้พิมพ์ข้อมูลด้วย สำหรับข้อมูลเพิ่มเติมเกี่ยวกับเวลาสำหรับการแสดงผล โปรดดูที่ “ ■ ข้อจำกัดของทริกเกอร์บิต” (หน้า 20-47)
- เมื่อใช้ D-Script การเปลี่ยนหน้าจอจะทำให้ฟังก์ชันตัวตั้งเวลาเริ่มนับจาก 0 ใหม่

20.7.3 Bit

■ Bit ON

เมื่อ GP ตรวจพบว่าตำแหน่งบิตที่กำหนดไว้ (ทริกเกอร์บิต) มีค่าเพิ่มจาก 0 เป็น 1 สคริปต์จะถูกทริกเกอร์

หมายเหตุ
- โปรดกำหนดระยะเวลาในการเปิดหรือปิดทริกเกอร์บิตให้มีระยะเวลานานกว่าระยะเวลาของรอบการสื่อสารหรือเวลาสำหรับการแสดงผล ขึ้นกับว่าระยะเวลาใดนานกว่ากัน สำหรับข้อมูลเพิ่มเติมเกี่ยวกับฟังก์ชันนี้ โปรดดูที่ “ ■ ข้อจำกัดของทริกเกอร์บิต” (หน้า 20-47)

■ Bit OFF

เมื่อ GP ตรวจพบว่าตำแหน่งบิตที่กำหนดไว้ (ทริกเกอร์บิต) มีค่าลดลงจาก 1 เป็น 0 สคริปต์จะถูกทริกเกอร์

หมายเหตุ
- โปรดกำหนดระยะเวลาในการเปิดหรือปิดทริกเกอร์บิตให้มีระยะเวลานานกว่าระยะเวลาของรอบการสื่อสารหรือเวลาสำหรับการแสดงผล ขึ้นกับว่าระยะเวลาใดนานกว่ากัน สำหรับข้อมูลเพิ่มเติมเกี่ยวกับฟังก์ชันนี้ โปรดดูที่ “ ■ ข้อจำกัดของทริกเกอร์บิต” (หน้า 20-47)

■ Bit Change

เมื่อ GP ตรวจพบว่าตำแหน่งบิตที่กำหนดไว้ (ทริกเกอร์บิต) มีค่าเพิ่มจาก 0 เป็น 1 หรือลดลงจาก 1 เป็น 0 สคริปต์จะถูกทริกเกอร์

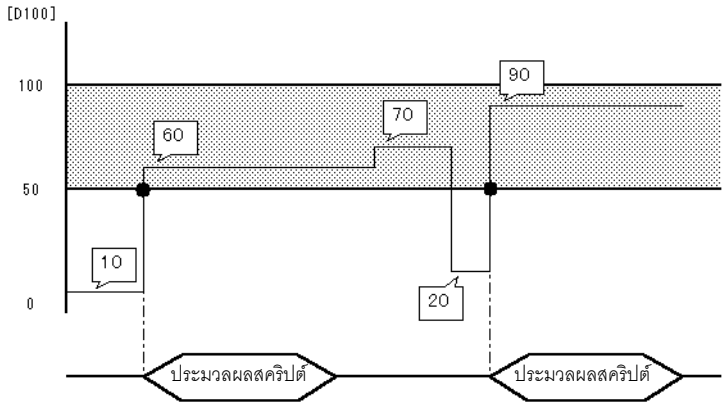
หมายเหตุ
- โปรดกำหนดระยะเวลาในการเปิดหรือปิดทริกเกอร์บิตให้มีระยะเวลานานกว่าระยะเวลาของรอบการสื่อสารหรือเวลาสำหรับการแสดงผล ขึ้นกับว่าระยะเวลาใดนานกว่ากัน สำหรับข้อมูลเพิ่มเติมเกี่ยวกับฟังก์ชันนี้ โปรดดูที่ “ ■ ข้อจำกัดของทริกเกอร์บิต” (หน้า 20-47)

20.7.4 Condition Expression

■ When Condition is Satisfied

สคริปต์จะทำงานหนึ่งครั้งเมื่อ GP ตรวจพบว่านิพจน์ที่กำหนดในโปรแกรมทริกเกอร์มีค่าเป็นจริง

ตัวอย่าง เมื่อตั้งค่าเงื่อนไขการทริกเกอร์เป็น 100>[D100]>50 สคริปต์จะทำงานตามเวลาต่อไปนี้
สคริปต์จะทำงานเมื่อ GP ตรวจพบว่าเงื่อนไขเปลี่ยนจาก [Not Satisfied]→[Satisfied] และกำหนดค่า 70 ให้ตำแหน่ง D100
หากเงื่อนไขเปลี่ยนแปลงแบบ [Satisfied]→[Satisfied] สคริปต์จะไม่ทำงาน

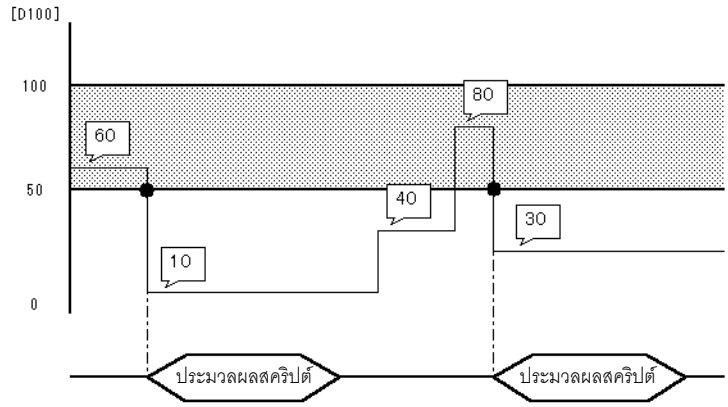


หมายเหตุ
- โปรดกำหนดระยะเวลาของเงื่อนไขการทริกเกอร์ให้มีระยะเวลานานกว่าระยะเวลาของรอบการสื่อสารหรือเวลาสำหรับการแสดงผล ขึ้นกับว่าระยะเวลาใดนานกว่ากัน สำหรับข้อมูลเพิ่มเติมเกี่ยวกับฟังก์ชันนี้ โปรดดูที่ “ ■ ข้อจำกัดของทริกเกอร์บิต” (หน้า 20-47)

■ When Condition is not Satisfied

สคริปต์จะทำงานหนึ่งครั้งเมื่อ GP ตรวจพบว่านิพจน์ที่กำหนดในโปรแกรมทริกเกอร์มีค่าเป็นเท็จ

ตัวอย่าง เมื่อตั้งค่าเงื่อนไขการทริกเกอร์เป็น 100>[D100]>50 สคริปต์จะทำงานตามเวลาต่อไปนี้
สคริปต์จะทำงานเมื่อ GP ตรวจพบว่าเงื่อนไขเปลี่ยนจาก
[Satisfied]→[Not Satisfied] และกำหนดค่า 20 ให้ตำแหน่ง D100
หากเงื่อนไขเปลี่ยนแปลงแบบ [Not Satisfied]→[Not Satisfied] สคริปต์จะไม่ทำงาน



หมายเหตุ
- โปรดกำหนดระยะเวลาของเงื่อนไขการทริกเกอร์ให้มีระยะเวลานานกว่าระยะเวลาของรอบการสื่อสารหรือเวลาสำหรับการแสดงผล ขึ้นกับว่าระยะเวลาใดนานกว่ากัน สำหรับข้อมูลเพิ่มเติมเกี่ยวกับฟังก์ชันนี้ โปรดดูที่ “ ■ ข้อจำกัดของทริกเกอร์บิต” (หน้า 20-47)

■ ข้อจำกัดของทริกเกอร์บิต

- โปรดกำหนดระยะเวลาในการเขียนข้อมูลลงในอุปกรณ์ที่เชื่อมต่อให้มีระยะเวลานานกว่าระยะเวลาของรอบการสื่อสาร การเขียนข้อมูลลงในอุปกรณ์ที่เชื่อมต่ออยู่โดยใช้ตัวนับจำนวนการสแกนของรีเลย์พิเศษภายในของ GP เป็นจำนวนบ่อยครั้ง อาจเกิดข้อผิดพลาดในการสื่อสารหรือข้อผิดพลาดของระบบได้
- เมื่อตั้งค่าบิตที่ใช้ในเงื่อนไขการทริกเกอร์ของ D-Script ให้เป็นการ“แตะ” แต่บิตนั้นปิดลงในขณะประมวลผล D-Script ระยะเวลาที่ใช้ในการแตะบริเวณสัมผัสซ้ำ ๆ หลายครั้งอาจทำให้ระบบตรวจไม่พบการเพิ่มขึ้นของบิตได้ ทริกเกอร์ D-Script จะเปรียบเทียบค่าที่อ่านในครั้งก่อนกับค่าที่อ่านได้ในปัจจุบัน เพื่อพิจารณาว่าทริกเกอร์ในขณะนี้มีค่าเป็น “จริง” หรือไม่ อย่างไรก็ตาม ในระหว่างการสแกนแต่ละครั้ง ค่าที่เก็บไว้ในตำแหน่งบิตที่ใช้ระหว่างการทริกเกอร์จะเป็นค่าเดียวกัน แม้ว่าค่านั้นจะเปลี่ยนไปในระหว่างที่สคริปต์ทำงานก็ตาม ระบบจะอ่านค่าใหม่หลังจากสแกนครั้งต่อไปเท่านั้น

*ระยะเวลาของรอบการสื่อสาร:

ระยะเวลาของรอบการสื่อสาร คือเวลาที่ใช้ในการร้องขอข้อมูลและนำข้อมูลจาก GP ไปที่ PLC โดยข้อมูลจะถูกจัดเก็บเป็นข้อมูลเลขฐานสองในตำแหน่ง LS2037 ของอุปกรณ์ภายใน มีหน่วยเป็นมิลลิวินาที (ms) โดยมีความคลาดเคลื่อนเท่ากับ ±10 มิลลิวินาที

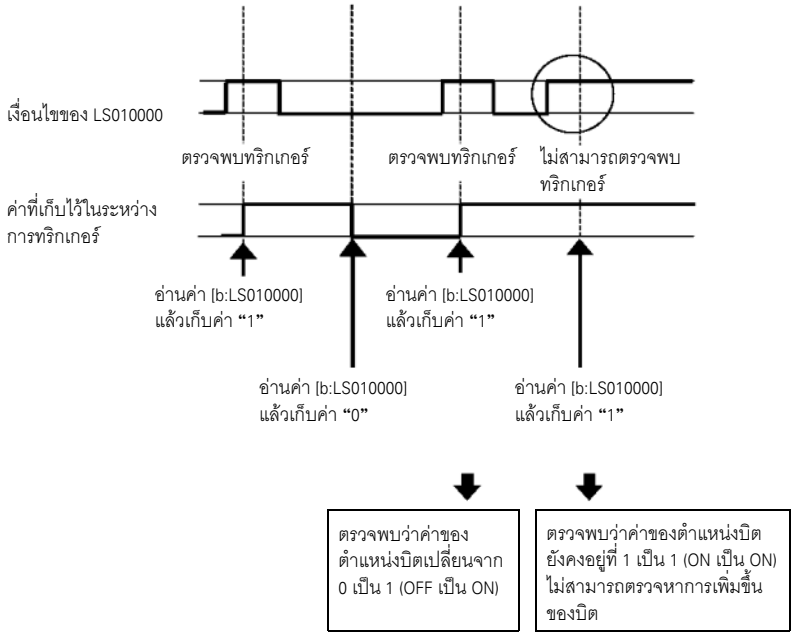
เวลาสำหรับการแสดงผล:

เวลาสำหรับการแสดงผล คือเวลาที่ใช้ในการแสดงผล/คำนวณค่าของหน้าจอ 1 หน้าจอ เวลานี้จะถูกจัดเก็บเป็นข้อมูลเลขฐานสองในตำแหน่ง LS2036 ของอุปกรณ์ภายใน มีหน่วยเป็นมิลลิวินาที (ms) โดยมีความคลาดเคลื่อนเท่ากับ ±10 มิลลิวินาที

ตัวอย่าง เมื่อกำหนดให้เปิดทริกเกอร์บิต (LS010000) ด้วยการแตะและปิดโดยใช้ D-Script

Trigger Condition: Bit ON [#INTERNAL] LS010000
Execution Expression: clear ([b:[#INTERNAL] LS010000])

◆ **ผังเวลาการประมวลผล** D-Script



ตามตัวอย่างนี้ หากไม่ใช้ระยะเวลาในการแตะของ D-Script และทำการตรวจหาเพียงอย่างเดียวเท่านั้น การประมวลผลจะเป็นดังนี้

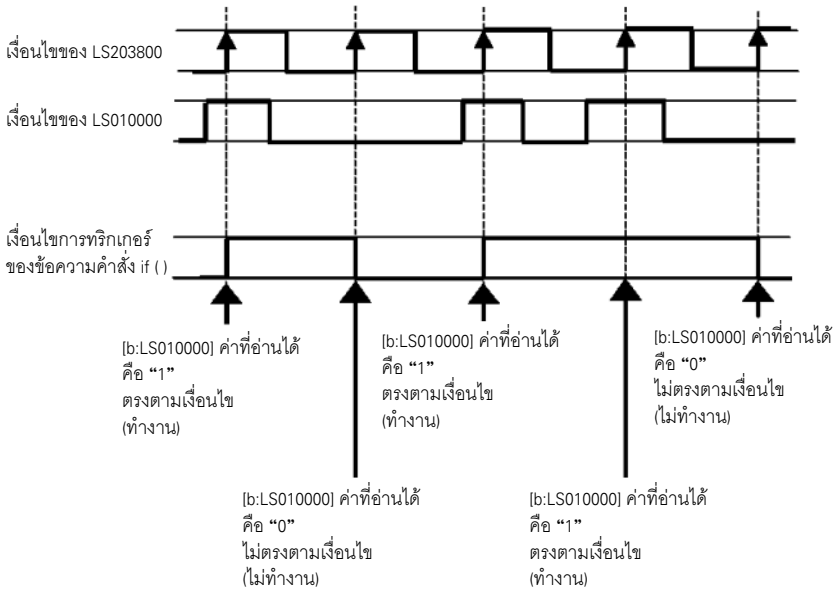
การใช้ข้อความคำสั่ง ( ) เพื่อตรวจหาทริกเกอร์
ใช้ข้อความคำสั่ง if ในการตรวจหาเมื่อตั้งค่าบิตให้ใช้การแตะ GP จะอ่านค่านี้และเปรียบเทียบค่าก่อนเริ่มประมวลผล

Trigger Condition: Bit ON ([#INTERNAL]LS203800 *1)
Execution Expression: if ([b:[#INTERNAL]LS010000]==1)
{
clear ([b:[#INTERNAL]LS010000])
:
:

*1 **ตัวนับภายในของ GP ตัวนับจะนับเพิ่มขึ้นทุกครั้งที่พาร์ทซึ่งตั้งค่าบนหน้าจอแสดงผลทำการประมวลผล**

เมื่อสร้าง D-Script ชนิดที่กล่าวถึงด้านบนนี้ ถึงแม้จะป้อนข้อมูลด้วยการแตะซ้ำ ๆ กันหลายครั้ง แต่เครื่องจะทำการสแกนแท็กดังที่แสดงในผังเวลาด้านล่างนี้ ในผังนี้ เครื่องจะอ่านค่าการสแกนแท็กแต่ละค่าและเปรียบเทียบเงื่อนไข หากตรงตามเงื่อนไขเครื่องจะประมวลผลไม่ว่าค่าก่อนหน้านี้จะเป็นเช่นไร

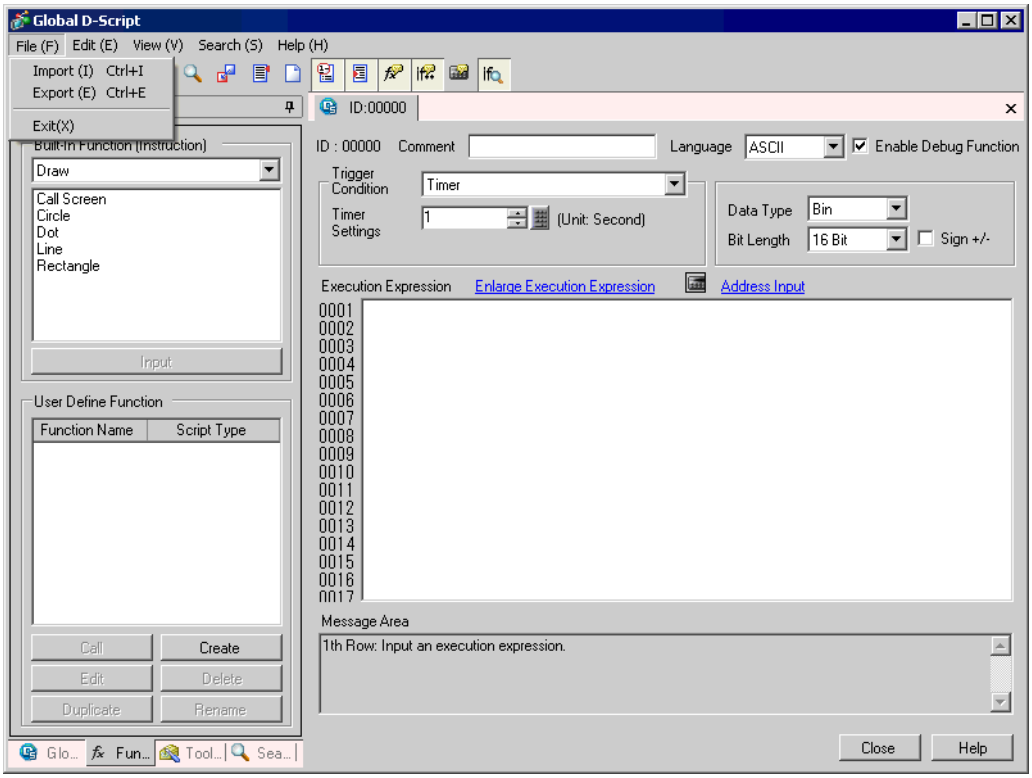
◆ **ผังเวลาการประมวลผล** D-Script

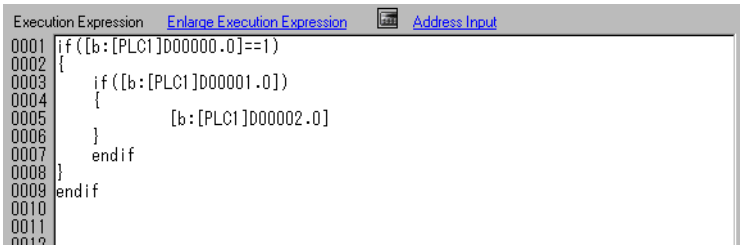


20.8 คำแนะนำในการตั้งค่า

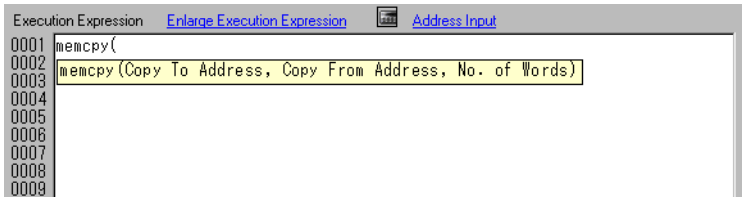
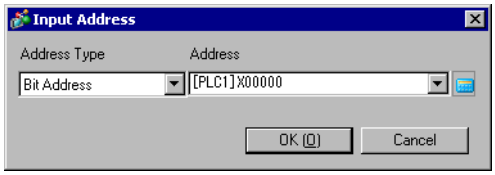
20.8.1 คำแนะนำในการตั้งค่าทั่วไป (D-Script)

กล่องโต้ตอบ D-Script Global D-Script จะมีลักษณะดังต่อไปนี้ สำหรับ Extended จะไม่มีหมายเลข ID และการตั้งค่าทริกเกอร์ แต่การตั้งค่าอื่น ๆ จะเหมือนกัน

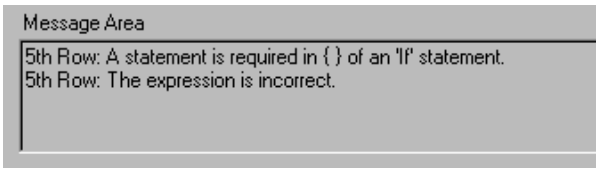


การตั้งค่า	คำอธิบาย
Export	คุณสามารถเลือกค่านี้ได้จากเมนู File Export จะเขียนสคริปต์ที่สร้างขึ้นเป็นไฟล์ข้อความ (.txt) ซึ่งสคริปต์อื่น ๆ สามารถนำเข้าไปใช้ได้
Import	คุณสามารถเลือกค่านี้ได้จากเมนู File Import จะอ่านข้อมูลในสคริปต์ที่ถูกนำออก (ไฟล์ข้อความ)
หมายเลขแถว	แสดงหมายเลขแถวทางด้านขวาของโปรแกรม
ตั้งย่อหน้าอัตโนมัติ	ย่อหน้าข้อความคำสั่งโดยอัตโนมัติดังเช่นในรูปภาพด้านล่างนี้ 

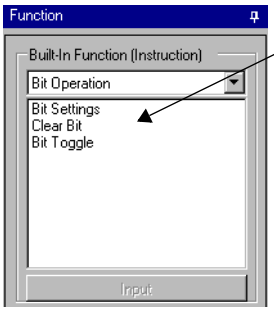
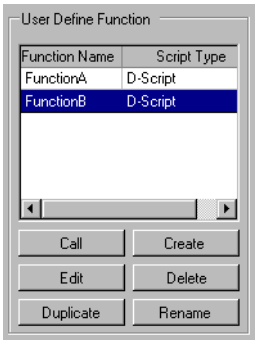
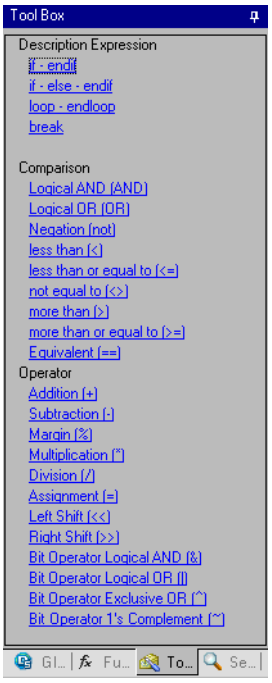
ต่อ

การตั้งค่า	คำอธิบาย
ตัวช่วยป้อนฟังก์ชัน	เมื่อป้อนฟังก์ชันและวงเล็บเปิด “(“ เช่นในรูปภาพด้านล่างนี้ ระบบจะแสดงรูปแบบที่มีอยู่ของฟังก์ชัน 
เติมคำสั่งให้สมบูรณ์อัตโนมัติ	เมื่อป้อน “if” หรือ “loop” จากแป้นคีย์ ระบบจะเติมคำสั่งที่เหลือให้ครบถ้วนโดยอัตโนมัติ
ตัวช่วยป้อนตำแหน่ง	เมื่อสร้างสคริปต์ หากคุณเปิดวงเล็บเหลี่ยม ([ ) กล่องโต้ตอบ [Input Address] จะปรากฏขึ้นโดยอัตโนมัติ
กล่องโต้ตอบสำหรับป้อนตำแหน่ง	แสดงกล่องป้อนข้อมูลต่อไปนี้เพื่อให้คุณป้อนข้อมูลตำแหน่ง  มีอุปกรณ์ 3 ชนิดที่สามารถเลือกได้ สำหรับข้อมูลเพิ่มเติมเกี่ยวกับอุปกรณ์ภายใน โปรดดูที่ “A.1.3 การสื่อสารกับอุปกรณ์/PLC ที่ GP ไม่รองรับ (วิธีการเชื่อมต่อผ่านหน่วยความจำ)” (หน้า A-5)หรือ “A.1.2 การสื่อสารกับอุปกรณ์/PLC โดยไม่มีผลต่อการทำงาน (วิธีการเชื่อมต่อโดยตรง)” (หน้า A-3) • เฉพาะการเชื่อมต่อผ่านหน่วยความจำ #INTERNAL : กำหนด USR ของอุปกรณ์ภายใน GP #MEMLINK : กำหนดอุปกรณ์ที่เชื่อมต่อผ่านหน่วยความจำ • เฉพาะอุปกรณ์ที่เชื่อมต่อ (ยกเว้นการเชื่อมต่อผ่านหน่วยความจำ) ชื่ออุปกรณ์แต่ละตัว (PLC1 ฯลฯ) : กำหนดอุปกรณ์แต่ละตัว #INTERNAL : กำหนดอุปกรณ์ภายใน GP • การเชื่อมต่อผ่านหน่วยความจำและอุปกรณ์ที่เชื่อมต่อ ชื่ออุปกรณ์แต่ละตัว (PLC1 ฯลฯ) : กำหนดอุปกรณ์แต่ละตัว #INTERNAL : กำหนดอุปกรณ์ภายใน GP #MEMLINK : กำหนดอุปกรณ์ที่เชื่อมต่อผ่านหน่วยความจำ

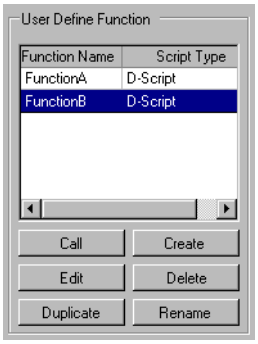
ต่อ

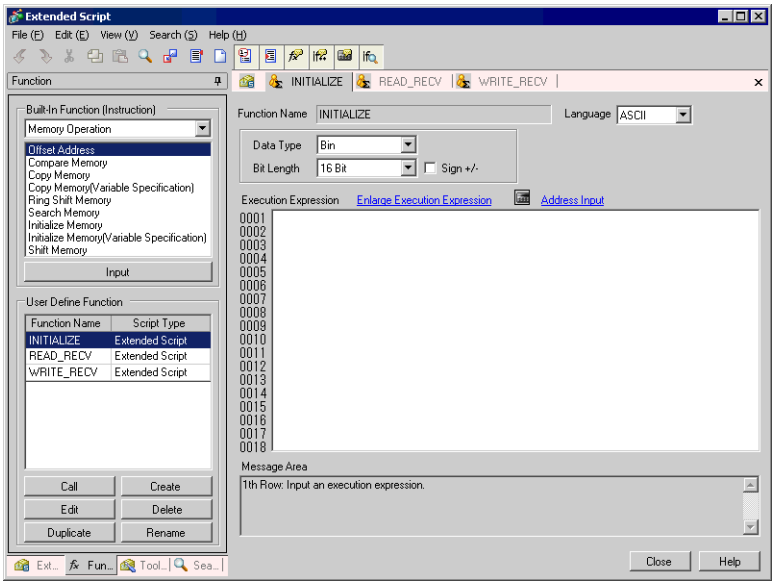
การตั้งค่า	คำอธิบาย
กล่องข้อความโต้ตอบการป้อนตำแหน่ง 	ข้อสำคัญ • ในสคริปต์ต่างๆ ห้ามตั้งค่ารหัสผ่านหรือค่าอื่นๆ โดยขึ้นต้นด้วย “0” เพราะระบบจะประมวลผลค่าตัวเลขที่ขึ้นต้นด้วย “0” เป็นข้อมูลชนิด Oct (ฐานแปด) • วิธีจำแนกข้อมูลที่ป้อนด้วยรูปแบบต่างๆ ตัวอย่าง DEC (ฐานสิบ) : ค่าเริ่มต้นที่ไม่ใช่ศูนย์ เช่น 100 Hex (ฐานสิบหก) : ค่าที่เริ่มต้นด้วย 0x เช่น 0x100 Oct (ฐานแปด) : ค่าที่เริ่มต้นด้วย 0 เช่น 0100 • ตัวอย่างการทำงานด้วยข้อมูลที่มีรูปแบบต่างกัน โดยใช้ตัวดำเนินการ AND (Hex และ BCD) Hex เท่านั้น 0x270F & 0xFF00 ผลลัพธ์: 0x2700 BCD และ Hex 9999 & 0xFF00 ผลลัพธ์: 0x9900
วิเคราะห์โปรแกรมอัตโนมัติ 	ตรวจสอบไวยากรณ์ระหว่างสร้างสคริปต์ ผลการตรวจสอบจะแสดงอยู่ที่ด้านล่างของหน้าต่าง 
ID	สคริปต์จะถูกจัดการตามหมายเลข ID เมื่อสร้างสคริปต์หลายสคริปต์โดยมีเงื่อนไขการทริกเกอร์ที่ต่างกัน ให้ตั้งค่าตั้งแต่ 0 ถึง 65,535
Comment	ใส่คำอธิบายสคริปต์
Language	เลือก [ASCII], [Japanese], [Taiwanese], [Chinese] หรือ [Korean]
Enable Debug Function	กำหนดว่าจะใช้งานฟังก์ชันดีบักหรือไม่ หากมีฟังก์ชัน _debug อยู่ในส่วนเนื้อหาของสคริปต์ ฟังก์ชัน _debug จะทำงาน สำหรับข้อมูลเพิ่มเติมเกี่ยวกับฟังก์ชันนี้ โปรดดูที่ “21.7.1 ฟังก์ชัน Debug” (หน้า 21-65)
เงื่อนไขการทริกเกอร์	ตั้งค่าเงื่อนไขการทริกเกอร์ที่จะทำให้สคริปต์ทำงาน สำหรับข้อมูลเพิ่มเติมเกี่ยวกับฟังก์ชันนี้ โปรดดูที่ “20.7 การตั้งค่าเงื่อนไขการทริกเกอร์” (หน้า 20-44) Extended Script จะไม่มีการตั้งค่าเงื่อนไขการทริกเกอร์
Data Type	ตั้งค่ารูปแบบข้อมูลสำหรับสคริปต์เป็น Bin หรือ BCD สำหรับ Extended Script จะกำหนดค่าไว้ตายตัวคือ Bin
Bit Length	ตั้งค่าความยาวข้อมูลสำหรับสคริปต์ระหว่าง 16 Bit หรือ 32 Bit
Sign +/–	เลือกช่องนี้เมื่อคุณต้องการแทรกจำนวนลบ คุณจะสามารถเลือกช่องนี้ได้เฉพาะเมื่อตั้งค่า Data Type เป็น Bin เท่านั้น

ต่อ

การตั้งค่า	คำอธิบาย
Execution Expression	รายละเอียดของสคริปต์
Built-in Function (Instruction)	คุณสามารถใช้ไอคอนนี้เลือกคำสั่งและฟังก์ชันต่าง ๆ ที่ใช้ได้ในสคริปต์ได้อย่างง่ายดาย ซึ่งจะช่วยลดเวลาในการป้อนคำสั่งและฟังก์ชันลง สำหรับข้อมูลเพิ่มเติมเกี่ยวกับคำสั่งและฟังก์ชันต่าง ๆ ที่ใช้ได้ โปรดดูที่ “21.13 รายการคำสั่ง” (หน้า 21-98) <Built-in Functions>  เลือกประเภทจากเมนูพูลดาวน์ที่ด้านบน ฟังก์ชันที่เกี่ยวข้องจะปรากฏขึ้นในพื้นที่ด้านล่าง คลิก [Input] หลังจากเลือกฟังก์ชันแล้ว กล่องโต้ตอบการตั้งค่าฟังก์ชันนั้นๆ จะปรากฏขึ้น
User Define Function	ลงทะเบียนสคริปต์ที่สร้างขึ้นให้เป็นฟังก์ชันที่กำหนดโดยผู้ใช้ ซึ่งสคริปต์อื่น ๆ สามารถใช้งานได้  **หมายเหตุ** • สำหรับรายละเอียดเพิ่มเติมเกี่ยวกับฟังก์ชันที่กำหนดโดยผู้ใช้ โปรดดูที่ “20.8.2 คำแนะนำในการตั้งค่าฟังก์ชันที่กำหนดโดยผู้ใช้” (หน้า 20-54)
Tool Box	คุณสามารถคลิกเลือกคำสั่งที่ใช้ได้ในสคริปต์ได้ซึ่งจะช่วยลดเวลาในการป้อนคำสั่งลง และยังสามารถเลือกคำสั่งเช่น ค้นหาและจัดตำแหน่งข้อความที่ใช้ในสคริปต์ได้อีกด้วย สำหรับข้อมูลเพิ่มเติมเกี่ยวกับคำสั่งที่ใช้ได้ โปรดดูที่ “บทที่ 21 คำสั่งและนิพจน์ ของโปรแกรม” (หน้า 21-1) 

20.8.2 คำแนะนำในการตั้งค่าฟังก์ชันที่กำหนดโดยผู้ใช้





การตั้งค่า	คำอธิบาย
Call	เรียกฟังก์ชันที่สร้างขึ้น เลือกฟังก์ชันที่จะเรียกใช้งาน คลิก [Call] “ชื่อฟังก์ชันที่เรียกใช้” จะถูกใส่ลงในฟิลด์ Execution Expression
Create	สร้างฟังก์ชันใหม่ คลิกที่ [Create] กล่องโต้ตอบของชื่อฟังก์ชันที่จะสร้างใหม่จะปรากฏขึ้น
Edit	แก้ไขฟังก์ชันที่สร้างไว้แล้ว เลือกฟังก์ชันที่จะแก้ไข คลิก [Edit] กล่องโต้ตอบ [D-Script Function] จะปรากฏขึ้น
Delete	ลบฟังก์ชันที่สร้างไว้แล้ว เลือกฟังก์ชันที่จะลบ แล้วคลิก [Delete]
Duplicate	คัดลอกฟังก์ชันที่สร้างไว้แล้ว เลือกฟังก์ชันที่จะคัดลอก คลิก [Duplicate] กล่องโต้ตอบจะปรากฏขึ้น เพื่อให้คุณบันทึกสำเนาฟังก์ชันด้วยชื่อใหม่
Rename	เปลี่ยนชื่อฟังก์ชันที่สร้างไว้แล้ว คลิก [Rename] กล่องโต้ตอบ Rename Function จะปรากฏขึ้น

20.9 ข้อจำกัด

20.9.1 ข้อจำกัดของ D-Script/Global D-Script

◆ **ข้อจำกัดในการทำงานของพอร์ต SIO**

- ตำแหน่งที่กำหนดในฟังก์ชัน Send/Receive จะไม่นับรวมอยู่ในจำนวนตำแหน่งของ D-Script
- Control เป็นตัวแปรแบบเขียนข้อมูลอย่างเดียว ส่วน Status และ Received Data เป็นตัวแปรแบบอ่านอย่างเดียว ห้ามอ่านข้อมูลตัวแปร Control หรือเขียนข้อมูลลงในตัวแปร Status เพราะจะทำให้การทำงานล้มเหลว
- ให้สร้าง D-Scripts (หรือฟังก์ชัน) ต่างหากสำหรับการส่งและการรับ
  สำหรับข้อมูลเพิ่มเติมเกี่ยวกับผังการถ่ายโอนข้อมูล โปรดดูที่

☞ “ ■ ผังการทำงาน” (หน้า 20-25)

- ช่วงที่ใช้ได้ของอุปกรณ์ LS ที่สามารถจัดเก็บข้อมูลของฟังก์ชัน Send/Receive ได้ คือพื้นที่สำหรับผู้ใช้ (LS20 ถึง LS2031 และ LS2096 ถึง LS8191)
- ใน [System Settings] - [Script Settings] หากไม่ได้ตั้งค่า [D-Script/Global D-Script] ไว้ บิตที่ 13 ของ LS2032 จะเปลี่ยนเป็นสถานะเปิด เมื่อมีการอ่านฟังก์ชัน Send, ฟังก์ชัน Receive, ตัวแปร Control, ตัวแปร Status และคุณสมบัติ Received Data Size
  สำหรับข้อมูลเกี่ยวกับโครงสร้างของอุปกรณ์ภายใน โปรดดูที่

☞ “A.1.4.3 รีเลย์พิเศษ” (หน้า A-16)

- เมื่อใช้ฟังก์ชัน Send/Receive ให้ตั้งค่าความยาวบิตของ D-Script เป็น 16 บิต
  โปรดทราบว่าการทำงานจะล้มเหลวหากตั้งค่าความยาวบิตเป็น 32 บิต
- บัฟเฟอร์การส่งข้อมูลมีขนาดเท่ากับ 2048 ไบต์ ส่วนบัฟเฟอร์การรับข้อมูลมีขนาด 8192 ไบต์ สัญญาณ ER (เอาต์พุต) สัญญาณ RS (เอาต์พุต) จะถูกปิด หากบัฟเฟอร์การรับข้อมูลรับข้อมูลเข้ามาแล้วอย่างน้อย 80%

◆ **ข้อจำกัดของการทำงานด้วยรูปแบบข้อมูล BCD**

หากโปรแกรมพบค่าที่ไม่สามารถแปลงเป็นรูปแบบ BCD ในระหว่างการทำงาน โปรแกรมจะหยุดทำงาน
ค่าเหล่านี้ได้แก่ A ถึง F ของข้อมูลแบบเลขฐานสิบหก ห้ามใช้ค่าเหล่านี้
หากโปรแกรมหยุดทำงานเนื่องจากค่าที่ไม่ใช่รูปแบบ BCD บิต 7 ในข้อมูลรีเลย์ร่วม (LS2032) ในเครื่อง GP จะเปิดขึ้น บิตนี้จะไม่ปิดจนกว่าจะปิดเครื่อง GP หรือเครื่อง GP เข้าสู่โหมดออฟไลน์
ตัวอย่าง

[w:[PLC1]D0200]=([w:[PLC1]D0300]<<2)+80
หาก D300 เท่ากับ 3 การเลื่อนบิตสองบิตไปทางซ้ายจะทำให้เกิดผลลัพธ์เป็น 0x000C ซึ่งไม่สามารถแปลงเป็นรูปแบบ BCD ได้ และจะขัดจังหวะการทำงานของโปรแกรม

[w:[PLC1]D0200]=[w:[PLC1]D0300]<<2
หาก D300 เท่ากับ 3 การเลื่อนบิตสองบิตไปทางซ้ายจะทำให้เกิดผลลัพธ์เป็น 0x000C แต่ต่างจากตัวอย่างด้านบนตรงที่ 0x000C เป็นผลลัพธ์จากการทำงานที่จะจัดเก็บไว้ในหน่วยความจำ และไม่ทำให้โปรแกรมหยุดทำงาน

◆ **ข้อจำกัดของการทำงานด้วยค่าศูนย์**

หากคุณใช้ศูนย์เป็นตัวหารใน division (/) และ remainder (%) โปรแกรมจะหยุดทำงาน ห้ามใช้ศูนย์เป็นตัวหาร หากโปรแกรมหยุดทำงานเนื่องจากค่าที่ไม่ใช่ BCD บิต 8 ในข้อมูลรีเลย์ร่วม (LS2032) ในเครื่อง GP จะเปิดขึ้น บิตนี้จะไม่ปิดจนกว่าจะปิดเครื่อง GP หรือเครื่อง GP เข้าสู่โหมดออฟไลน์

◆ **ข้อควรทราบเกี่ยวกับความล่าช้าในระหว่างการทำงานของคำสั่ง** Assign

การใช้ตำแหน่งอุปกรณ์ในคำสั่ง Assign อาจทำให้การเขียนข้อมูลล่าช้าได้ เนื่องจาก GP ต้องอ่านข้อมูลตำแหน่งจากอุปกรณ์ที่เชื่อมต่ออยู่ โปรดดูตัวอย่างด้านล่างนี้

ตัวอย่าง

[w:[PLC1]D0200]=([w:[PLC1]D0300]+1 ... (1)

[w:[PLC1]D0201]=([w:[PLC1]D0200]+1 ... (2)

ข้อความคำสั่ง (1) กำหนด (D0300+1) ลงใน D0200 แต่ในข้อความคำสั่ง (2) ผลลัพธ์ของข้อความคำสั่ง (1) ไม่ได้ถูกกำหนดลงใน D0200 เนื่องจากการสื่อสารกับอุปกรณ์/PLC ใช้เวลานาน ในกรณีเช่นนี้ ให้เขียนโปรแกรมให้จัดเก็บผลลัพธ์ของข้อความคำสั่ง (1) ในพื้นที่ LS ก่อนที่จะเรียกใช้ผลลัพธ์นั้น ดังเช่นตัวอย่างด้านล่างนี้

[w:[#INTERNAL]LS0100]=[w:[PLC1]D0300]+1

[w:[PLC1]D0200]=[w:[#INTERNAL]LS0100]

[w:[PLC1]D0201]=[w:[#INTERNAL]LS0100]+1

◆ ในการเขียนโปรแกรม D-Script โปรดจำไว้ว่าตำแหน่งสามตำแหน่งจะใช้หน่วยความจำเท่ากับพาร์ทหนึ่งพาร์ท จำนวนตำแหน่งสูงสุดที่สามารถใช้กับ D-Script ได้คือ 255 ตำแหน่ง แต่ควรใช้ตำแหน่งให้น้อยที่สุดเท่าที่ทำได้ เนื่องจากยิ่งใช้อุปกรณ์มากเท่าไร ก็จะยิ่งทำให้การตอบสนองช้าลงเท่านั้น

◆ คำสั่ง Convert Address ในเมนู Utility ของ Project Manager ไม่สามารถแปลงตำแหน่งที่ใช้ใน D-Script ได้ ให้เปิด D-Script Editor เพื่อแก้ไขตำแหน่งเหล่านี้

◆ เมื่อเปลี่ยนแปลงการตั้งค่า Connected Device Type จากหน้าต่าง Save As ของเมนู Project ใน Project Manager จะไม่สามารถแก้ไขตำแหน่งที่ D-Script ใช้ได้ โปรดแก้ไขตำแหน่งเหล่านี้โดยใช้ D-Script Editor

◆ ขนาดของ D-Script จะมีผลต่อเวลาสำหรับการแสดงผล โปรดทราบว่าการใช้ตำแหน่งเป็นจำนวนมากทำให้ประสิทธิภาพการทำงานของโปรแกรมลดลงอย่างเห็นได้ชัด

◆ เมื่อเรียกฟังก์ชันหนึ่งจากฟังก์ชันหนึ่ง สามารถเรียกได้สูงสุด 9 ระดับ (การซ้อนกัน) ห้ามสร้างระดับการซ้อนมากกว่านี้

◆ คุณสามารถเรียกฟังก์ชันซ้อนกันได้ไม่เกิน 9 ระดับ

◆ คุณสามารถสร้างฟังก์ชันได้ไม่เกิน 254 ฟังก์ชัน

◆ การทำงานของ D-Script ที่เปิดขึ้นโดยทริกเกอร์หลังจากหน้าจอเปลี่ยนแปลงจะเป็นดังนี้:

เงื่อนไขการทริกเกอร์ / ค่าปัจจุบันหรือเงื่อนไข	วิธีการเชื่อมต่อโดยตรง				วิธีการเชื่อมต่อผ่านหน่วยความจำ			
	Bit "0"	Bit "1"	Condition is not Satisfied	Condition is Satisfied	Bit "0"	Bit "1"	Condition is not Satisfied	Condition is Satisfied
ขอบขาขึ้นของบิต	ไม่รองรับ	รองรับ	—	—	ไม่รองรับ	ไม่รองรับ	—	—
ขอบขาลงของบิต	รองรับ	ไม่รองรับ	—	—	ไม่รองรับ	ไม่รองรับ	—	—
การเปลี่ยนแปลงของบิต	รองรับ	รองรับ	—	—	ไม่รองรับ	ไม่รองรับ	—	—
การตั้งค่าตัวตั้งเวลา	ไม่รองรับ	ไม่รองรับ	ไม่รองรับ	ไม่รองรับ	ไม่รองรับ	ไม่รองรับ	ไม่รองรับ	ไม่รองรับ
ตรวจพบว่าเป็นจริง	—	—	ไม่รองรับ	รองรับ	—	—	ไม่รองรับ	รองรับ
ตรวจพบว่าเป็นเท็จ	—	—	รองรับ	ไม่รองรับ	—	—	รองรับ	ไม่รองรับ

รองรับ : ทำงานทันทีหลังจากเปลี่ยนแปลงหน้าจอ หรือเมื่อเปิดเครื่อง

ไม่รองรับ : ไม่ทำงานทันทีหลังเปลี่ยนแปลงหน้าจอ หรือเมื่อเปิดเครื่อง

- เมื่อตัวตั้งเวลาทำงาน ตัวตั้งเวลาจะเริ่มนับเวลาทันทีหลังจากหน้าจอเปลี่ยนแปลง
- เมื่อใช้ Global D-Script จะมีการทำงานต่าง ๆ ดังที่กล่าวถึงด้านบนนี้เฉพาะเมื่อเปิดเครื่อง GP เท่านั้น อย่างไรก็ตาม เมื่อหน้าจอ GP เปลี่ยนแปลงจะไม่มีการทำงานต่าง ๆ ดังที่กล่าวถึงด้านบน และจอมอนิเตอร์จะทำงานโดยใช้เงื่อนไขการทริกเกอร์ที่ตั้งค่าไว้
- เมื่อมีตัวตั้งเวลาอยู่ใน Global D-Script ตัวตั้งเวลาจะเริ่มนับเวลาทันทีหลังจากเสียบปลั๊กไฟของเครื่อง GP

หมายเหตุ
- ห้ามใช้ปุ่มบนหน้าจอสัมผัสตั้งค่าทริกเกอร์บิตหรือสั่งงานบิตเริ่มต้นในโปรแกรม ระยะเวลาในการแตะเพื่อป้อนข้อมูลอาจไม่ถูกต้อง ทำให้บิตที่ป้อนไม่ถูกต้อง

◆ เมื่อกำหนดค่าให้ตำแหน่งสำหรับเปลี่ยนหน้าจอในขณะที่กำลังเรียกใช้คำสั่ง D-Script ระบบจะเปลี่ยนหน้าจอหลังจากประมวลผล D-Scripts ทั้งหมดแล้ว

ตัวอย่าง

```
ID              00000
Data Type:      Bin,        Data Length: 16 Bit,    Sign +/-:    None
Trigger:        Leading bit ([b:M0000])
[w:[PLC1]D0100]=0           // (1)
[w:[#INTERNAL]LS0008]=30// (2) Switches to Base screen No. 30
[w:[PLC1]D0101]=1           // (3)
[w:[PLC1]D0102]=2           // (4)
```

เมื่อเรียกใช้ D-Script ข้างต้น ระบบจะเปลี่ยนหน้าจอหลังจากประมวลผล (3) และ (4) แล้ว

◆ เมื่อตั้งค่าข้อมูลที่ใช้กับ D-Script ด้วยการแตะจากเครื่อง GP ให้ตรวจดูว่าได้เขียนข้อมูลทั้งหมดแล้ว แล้วจึงเรียกใช้ D-Script

◆ **ข้อจำกัดเฉพาะของ** Global D-Script

- เมื่อเปิดเครื่อง GP จะไม่มีการทำงานตามที่แสดงในตารางในหน้าที่แล้ว เมื่อหน้าจอเปลี่ยนแปลง จะไม่นำตารางดังกล่าวมาใช้ และเงื่อนไขการทริกเกอร์จะถูกตรวจสอบอย่างต่อเนื่อง
- Global D-Script จะถูกพักการทำงานชั่วคราวในระหว่างเปลี่ยนหน้าจอหรือมีการทำงานอื่น ๆ ของเครื่อง GP
- หลังจากเปิดเครื่อง GP Global D-Script จะยังไม่ทำงาน จนกว่าเครื่องจะอ่านข้อมูลทั้งหมดของหน้าจอเริ่มต้นเสร็จแล้ว
  อย่างไรก็ตาม หลังจากหน้าจอเริ่มต้นเปลี่ยนไป Global D-Script อาจทำงานก่อนที่จะอ่านข้อมูลเสร็จ
- จำนวนอุปกรณ์สูงสุดใน Global D-Script คือ 255 เมื่อมีจำนวนอุปกรณ์เท่ากับ 256 D-Script จะไม่ทำงาน เนื่องจากอุปกรณ์เหล่านี้จะอ่านข้อมูลเสมอโดยไม่คำนึงถึงหน้าจอ จึงควรตั้งค่าจำนวนอุปกรณ์ใน D-Script ให้มีจำนวนน้อยที่สุด มิฉะนั้น ประสิทธิภาพในการทำงานจะลดลง
- จำนวน Global D-Scripts สูงสุดที่สามารถใช้ได้คือ 32 ฟังก์ชันที่ใช้อยู่ในขณะนี้ก็นับเป็นหนึ่ง Global D-Script เช่นกัน เมื่อ Global D-Scripts มีจำนวนครบ 32 ระบบจะไม่สนใจ Global D-Scripts ใหม่

20.9.2 ข้อจำกัดของ Extended Script

- ตำแหน่งอุปกรณ์ต่าง ๆ สามารถใช้ได้เฉพาะพื้นที่ LS และพื้นที่ USR (พื้นที่เสริมสำหรับผู้ใช้) เท่านั้น
- ตำแหน่งชั่วคราวของ D-Scripts และ Global D-Scripts จะถูกแยกจัดการจากตำแหน่งชั่วคราวของ Extended Script ดังนั้น การเปลี่ยนแปลงต่าง ๆ ในตำแหน่งชั่วคราวของ D-Scripts และ Global D-Scripts จึงไม่ปรากฏในตำแหน่งชั่วคราวของ Extended Script
- คุณสามารถเรียกฟังก์ชันที่กำหนดโดยผู้ใช้ที่สร้างด้วย D-Script/Global D-Script ได้ แต่หากคุณเข้าใช้ตำแหน่งอุปกรณ์ที่อยู่นอกอุปกรณ์ภายในที่มีอยู่ในฟังก์ชัน ฟังก์ชันอาจทำงานไม่ถูกต้อง นอกจากนี้ เมื่อมีการถ่ายโอนข้อมูล (ในระหว่างสร้างข้อมูลสำหรับ GP) ระบบจะสร้างฟังก์ชันที่กำหนดโดยผู้ใช้ของ D-Scripts, Global D-Scripts และ Extended Script แยกต่างหากจากกัน
- เมื่อเรียกฟังก์ชันหนึ่งจากฟังก์ชันหนึ่ง สามารถเรียกได้สูงสุด 9 ระดับ (การซ้อนกัน)
- คุณสามารถเรียกฟังก์ชันได้ไม่เกิน 254 ฟังก์ชัน (จำนวนฟังก์ชันที่ใช้กับฟังก์ชัน “Call” ได้คือ 254)
- Extended Script ไม่มีผลต่อการนับจำนวนแท็ก
- ฟังก์ชันที่รองรับเฉพาะ Extended Script เช่น การทำงานของสตริง จะไม่ทำงานหากเรียกด้วย D-Script หรือ Global D-Script
- รูปแบบข้อมูลที่สามารถใช้ได้คือ Bin ข้อมูลที่เป็นรูปแบบ BCD ไม่สามารถใช้ได้
- บัฟเฟอร์การส่งข้อมูลมีขนาดเท่ากับ 2048 ไบต์ ส่วนบัฟเฟอร์การรับข้อมูลมีขนาด 8192 ไบต์ บรรทัด CTS จะถูกปิด หากบัฟเฟอร์การรับข้อมูลรับข้อมูลเข้ามาแล้วอย่างน้อย 80%
- ไม่สามารถเลือกโปรโตคอลทั่วไปและ Extended Script พร้อมกันได้ ตารางต่อไปนี้แสดงข้อมูลเพิ่มเติมเกี่ยวกับการใช้ร่วมกัน

การตั้งค่า Extended SIO	ฟังก์ชัน Extended SIO ของ D-Script/ Global D-Script	ฟังก์ชัน Extended SIO ของ Extended Script
โปรโตคอลทั่วไป	รองรับ: ทำงานได้	ไม่รองรับ: ทำงานไม่ได้
Extended Script	ไม่รองรับ: ทำงานไม่ได้	รองรับ: ทำงานได้

- หลักเกณฑ์ในการตั้งค่าสตริงอักขระ
  เมื่อใช้สตริงอักขระที่มี “_ strset ( )” และฟังก์ชันอื่นๆ ให้ใส่สตริงอักขระไว้ในเครื่องหมายอัญประกาศ (")
  เมื่อต้องการแสดงเครื่องหมายอัญประกาศในสตริงอักขระ ให้ใส่สัญลักษณ์ “\” และแสดงเป็น as [\"]
  ไม่มีวิธีใดที่จะเขียนแทนสัญลักษณ์ “\” ตัวเดียวได้ หากจำเป็น ให้ใช้การตั้งค่ารูปแบบรหัสอักขระ
  (_strset (databuf0, 92)
  ตัวอย่าง
  "ABC\"DEF" → ABC"DEF
  "ABC\DEF" → ABC\DEF
  "ABC\\"DEF" → ABC\"DEF
  "ABC\\DEF" → ABC\\DEF

◆ **ตารางต่อไปนี้จะแสดงขนาดของบัฟเฟอร์สำหรับ** Extended SIO, databuf0, databuf1, databuf2 **และ** databuf3

บัฟเฟอร์	ชื่อบัฟเฟอร์	ขนาด
บัฟเฟอร์ข้อมูล 0	databuf0	1 KB
บัฟเฟอร์ข้อมูล 1	databuf1	1 KB
บัฟเฟอร์ข้อมูล 2	databuf2	1 KB
บัฟเฟอร์ข้อมูล 3	databuf3	1 KB

20.9.3 ข้อจำกัดของฟังก์ชันที่กำหนดโดยผู้ใช้

- ส่วนของคำสั่งที่สามารถใช้ได้จะแตกต่างกันไปในแต่ละสคริปต์ เมื่อใช้คำสั่ง โปรดดูที่ “21.13 รายการคำสั่ง” (หน้า 21-98)
- สำหรับชื่อฟังก์ชัน คุณสามารถใช้ตัวอักษรภาษาอังกฤษตัวใดก็ได้หรือใช้เส้นใต้อักขระ “_” (คุณสามารถขึ้นต้นชื่อฟังก์ชันได้ด้วยอักขระตัวเลขผสมตัวอักษรเท่านั้น)
- ห้ามใช้ชื่อต่อไปนี้เป็นชื่อฟังก์ชัน

and	b_call	Bcall	_bom2hexase	break	Call
_CF_delete	_CF_dir	_CF_read	_CF_read_csv	_CF_rename	_CF_write
clear	databuf0	databuf1	databuf2	databuf3	_decasc2bin
_dlcopy	dsp_arc	dsp_corcle	dsp_dot	dsp_line	dsp_rectangle
else	endif	fall	_hexasc2bin	if	IO_READ
IO_READ_EX	IO_READ_WAIT	IO_WRITE	IO_WRITE_EX	loop	_memcmp
memcpy	_memcpy_EX	memring	_memsearch	memset	_memsetEX
_memshift	not	or	return	rise	rise_expr
set	_strcat	_strlen	_strmid	_strset	timer
toggle	_wait				

20.9.4 ข้อควรทราบเกี่ยวกับผลการทำงาน

■ Overflowing Digit

Overflowing Digit เกิดขึ้นจากการคำนวณที่ถูกปัดเศษ
เมื่อทำการคำนวณข้อมูล 16 บิตที่ไม่ได้ระบุเครื่องหมาย:

- 65535 + 1 = 0 (เกิด Overflowing Digit)
- (65534 * 2) / 2 = 32766 (เกิด Overflowing Digit)
- (65534 / 2) * 2 = 65534 (ไม่เกิด Overflowing Digit)

■ ความแตกต่างของการคำนวณเศษเหลือ

ผลลัพธ์ของการคำนวณเศษเหลือจะขึ้นอยู่กับว่ามีการใส่เครื่องหมายทางฝั่งซ้ายและขวาหรือไม่

- −9 % 5 = −4
- 9 % −5 = 4

■ การปัดจุดทศนิยม

จุดทศนิยมที่เป็นผลจากการหารจะถูกปัดเศษ

- 10 / 3 * 3 = 9
- 10 * 3 / 3 = 10

■ ข้อความทราบในการทำงานด้วยข้อมูล BCD

การทำงานกับข้อมูล BCD ที่ทำให้เกิด Overflowing Digit จะมีผลลัพธ์ที่ไม่ถูกต้อง

20.9.5 ข้อผิดพลาด

ข้อความแสดงข้อผิดพลาดต่อไปนี้จะปรากฏขึ้นเมื่อตั้งค่าสคริปต์ไม่ถูกต้อง
ข้อผิดพลาดจะแสดงอยู่ที่ด้านล่างของหน้าจอเครื่อง GP
รหัสข้อผิดพลาดจะถูกเขียนลงในตำแหน่ง LS91XX ตัวเลขที่เขียนในรหัสข้อผิดพลาดจะเป็นตัวเลขที่ตามหลัง RAAA ในตารางด้านล่างนี้ (ตัวอย่างเช่น เมื่อเกิดข้อผิดพลาด RAAA130 ขึ้น ระบบจะเขียนเลข '130' ไว้)

☞ "30.4 ข้อผิดพลาดที่แสดงบนเครื่อง GP" (หน้า 30-19)

รายการแสดงรหัสข้อผิดพลาดของสคริปต์

D-Script (ตำแหน่งข้อผิดพลาด = LS9120)	Global D-Script (ตำแหน่งข้อผิดพลาด = LS9110)	Extended Script (ตำแหน่งข้อผิดพลาด = LS9100)
—	RAAA130	RAAA140
ไม่ใช้	Global D-Script เกิดข้อผิดพลาด (จำนวน Global D-Scripts ทั้งหมดเกินจำนวนสูงสุดที่กำหนดไว้ที่ 32)	Extended Script เกิดข้อผิดพลาด (จำนวนฟังก์ชันทั้งหมดเกินจำนวนสูงสุดที่กำหนดไว้ที่ 255)
—	RAAA131	—
ไม่ใช้	Global D-Script เกิดข้อผิดพลาด (จำนวนอุปกรณ์ทั้งหมดเกินจำนวนสูงสุดที่กำหนดไว้ที่ 255)	ไม่ใช้
RAAA120	RAAA132	RAAA141
D-Script เกิดข้อผิดพลาด (ไม่มีฟังก์ชันตามที่ระบุหรือฟังก์ชันเกิดข้อผิดพลาด)	Global D-Script เกิดข้อผิดพลาด (ไม่มีฟังก์ชันตามที่ระบุหรือฟังก์ชันเกิดข้อผิดพลาด)	Extended Script เกิดข้อผิดพลาด (ไม่มีฟังก์ชันที่ระบุหรือฟังก์ชันเกิดข้อผิดพลาด)
RAAA121	RAAA133	RAAA142
D-Script เกิดข้อผิดพลาด (ฟังก์ชันเหล่านี้ซ้อนกันเกิน 10 ระดับ)	Global D-Script เกิดข้อผิดพลาด (ฟังก์ชันเหล่านี้ซ้อนกันเกิน 10 ระดับ)	Extended Script เกิดข้อผิดพลาด (ฟังก์ชันเหล่านี้ซ้อนกันเกิน 10 ระดับ)
RAAA122	RAAA134	RAAA143
D-Script เกิดข้อผิดพลาด (มีนิพจน์ที่ใช้ไม่ได้ในเวอร์ชันนี้)	Global D-Script เกิดข้อผิดพลาด (มีนิพจน์ที่ใช้ไม่ได้ในเวอร์ชันนี้)	Extended Script เกิดข้อผิดพลาด (มีนิพจน์ที่ใช้ไม่ได้ในเวอร์ชันนี้)
RAAA123	RAAA135	RAAA144
D-Script เกิดข้อผิดพลาด (ใช้ฟังก์ชันการทำงานของ SIO โดยที่ยังไม่ได้ตั้งค่าอุปกรณ์/PLC)	Global D-Script เกิดข้อผิดพลาด (ใช้ฟังก์ชันการทำงานของ SIO โดยที่ยังไม่ได้ตั้งค่าอุปกรณ์/PLC)	Extended Script เกิดข้อผิดพลาด (ใช้ฟังก์ชันการทำงานของ SIO โดยที่ยังไม่ได้ตั้งค่าอุปกรณ์/PLC)
RAAA124	RAAA136	RAAA145
D-Script เกิดข้อผิดพลาด	Global D-script เกิดข้อผิดพลาด	Extended D-Script เกิดข้อผิดพลาด

บันทึก