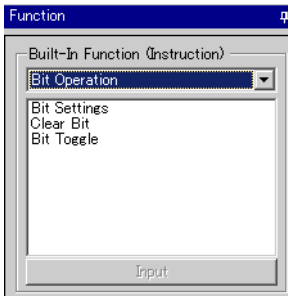
21 คำสั่งและคำอธิบาย

ในบทนี้จะอธิบายวิธีการใช้คำสั่งและคำอธิบายสคริปต์ใน GP-Pro EX
สำหรับข้อมูลเพิ่มเติมเกี่ยวกับการเขียนโปรแกรมโดยใช้สคริปต์ โปรดดูที่ “บทที่ 20 การใช้สคริปต์ (การเขียนโปรแกรมโดยไม่ใช้พาร์ท)” (หน้า 20-1)

21.1 การทำงานของบิต

Bit Operation	ข้อมูลสรุปของฟังก์ชัน
	Bit Settings ☞ "21.1.1 Bit Settings" (หน้า 21-2) เปลี่ยนตำแหน่งบิตที่ระบุจาก 0 → 1
	Clear Bit ☞ "21.1.2 Clear Bit" (หน้า 21-2) เปลี่ยนตำแหน่งบิตที่ระบุจาก 1 → 0
	Bit Toggle ☞ "21.1.3 Bit Toggle" (หน้า 21-2) เปลี่ยนตำแหน่งบิตที่ระบุจาก 1 → 0 หรือจาก 0 → 1

21.1.1 Bit Settings

รายการ	**คำอธิบาย**
ข้อมูลสรุป	เปลี่ยนตำแหน่งบิตที่ระบุจาก 0 → 1
รูปแบบ	set( )

ตัวอย่างนิพจน์

set ([b:[#INTERNAL]LS010000])

จากตัวอย่างข้างบน บิตที่ 00 ของ LS0100 ถูกเปลี่ยนจาก 0 → 1

21.1.2 Clear Bit

รายการ	คำอธิบาย
ข้อมูลสรุป	เปลี่ยนตำแหน่งบิตที่ระบุจาก 1 → 0
รูปแบบ	clear()

ตัวอย่างนิพจน์

clear ([b:[#INTERNAL]LS010000])

จากตัวอย่างข้างบน บิตที่ 00 ของ LS0100 ถูกเปลี่ยนจาก 1 → 0

21.1.3 Bit Toggle

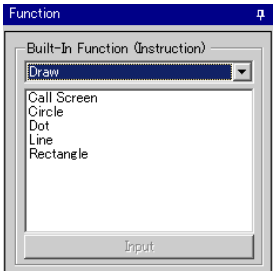
รายการ	คำอธิบาย
ข้อมูลสรุป	เปลี่ยนตำแหน่งบิตที่ระบุจาก 1 → 0 หรือาจาก 0 → 1
รูปแบบ	toggle ( )

ตัวอย่างนิพจน์

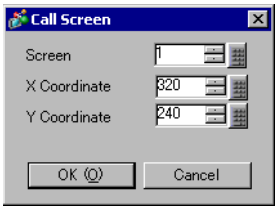
toggle([b:[#INTERNAL]LS010000])

จากตัวอย่างข้างบน บิตที่ 00 ของ LS0100 ถูกเปลี่ยนจาก 1 → 0 หรือจาก 0 → 1

21.2 การวาด

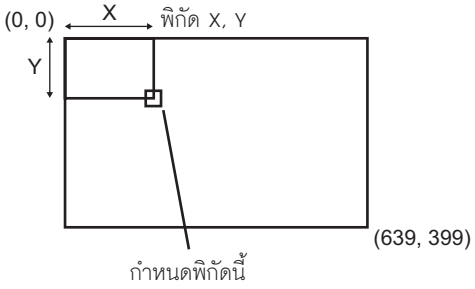
Draw	ข้อมูลสรุปของฟังก์ชัน
	Call Screen ☞ "21.2.1 Call Screen" (หน้า 21-3) **เรียกหน้าจอ (หน้าจอหลัก) ที่มีหมายเลขหน้าจอที่กำหนด** **ฟังก์ชันนี้ใช้ใน Extended Script ไม่ได้**
	Circle ☞ "21.2.2 Circle" (หน้า 21-4) **วาดวงกลมที่กำหนด**
	Dot ☞ "21.2.3 Dot" (หน้า 21-5) **วาดจุดที่กำหนด**
	Line ☞ "21.2.4 Line" (หน้า 21-5) **วาดเส้นที่กำหนด**
	Rectangle ☞ "21.2.5 Rectangle" (หน้า 21-6) **วาดสี่เหลี่ยมผืนผ้าที่กำหนด**

21.2.1 Call Screen

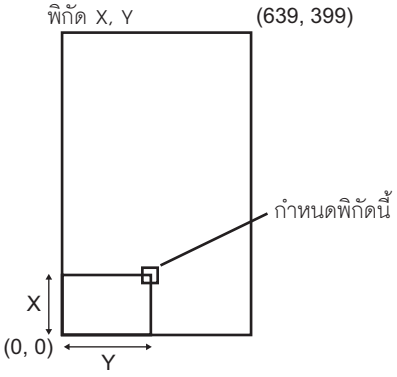
รายการ	คำอธิบาย
ข้อมูลสรุป	**ฟังก์ชันนี้จะเรียกรายการไลบรารีที่ลงทะเบียนไว้ หน้าจอที่กำหนด (หน้าจอหลัก) จะถูกเรียก ให้แสดงขึ้นที่พิกัด X,Y ที่ระบุ** **ฟังก์ชันนี้ใช้ใน Extended Script ไม่ได้**
รูปแบบ	b_call (หมายเลขหน้าจอ, พิกัด X, พิกัด Y)  • **กำหนดพิกัดกึ่งกลางด้วยพิกัด X และพิกัด Y ของหน้าจอที่ถูกเรียก**

ตำแหน่งพิกัด

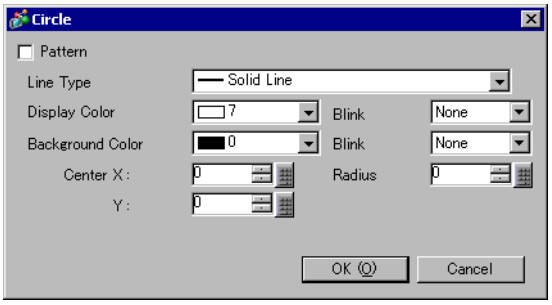
แบบแนวตั้ง



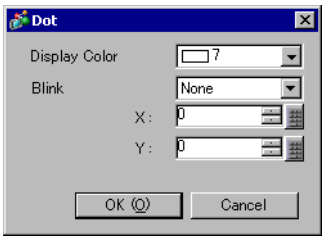
แบบแนวนอน



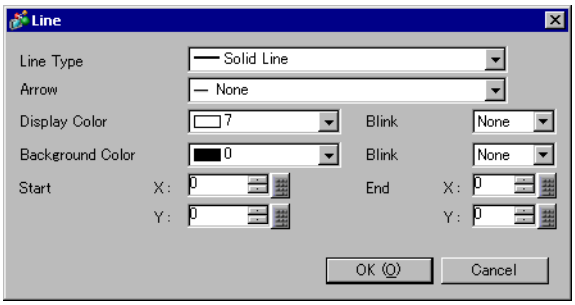
21.2.2 Circle

รายการ	คำอธิบาย
ข้อมูลสรุป	วาดวงกลมตรงตำแหน่งที่กำหนด หากคุณทำเครื่องหมายที่ช่อง [Pattern] จะเป็นการวาดวงกลมทึบ ให้เลือกและป้อนชนิดของเส้น (หรือรูปแบบการเติมเมื่อเลือกรูปแบบ), คุณสมบัติของสี, พิกัดกึ่งกลาง และค่ารัศมี คุณสามารถตั้งค่าพิกัดกึ่งกลาง และรัศมีได้โดยทางอ้อมได้อีกด้วย
รูปแบบ	dsp_circle (พิกัด X, พิกัด Y, รัศมี, การกะพริบของสีที่แสดงผล + สีที่แสดงผล, การกะพริบของสีพื้นหลัง + สีพื้นหลัง, ชนิดของเส้น)  • เมื่อตั้งค่าเป็นสีดำและให้มีการกระพริบด้วย สีพื้นหลังจะกลายเป็นโปร่งใส

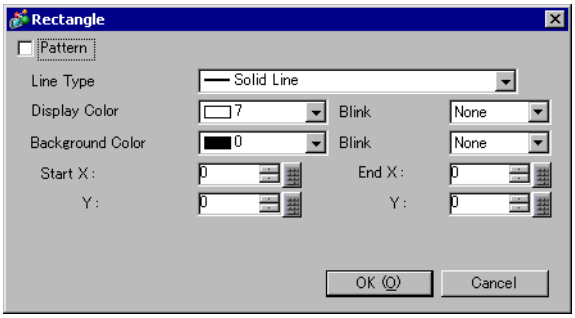
21.2.3 Dot

รายการ	คำอธิบาย
ข้อมูลสรุป	วาดจุดตรงตำแหน่งที่กำหนด กำหนดพิกัด X,Y และสีที่แสดงผล
รูปแบบ	dsp_dot (พิกัด X, พิกัด Y, การกะพริบ + สีที่แสดงผล)  • เมื่อตั้งค่าการกะพริบและสีดำพร้อมกัน สีพื้นหลังจะโปร่งใส

21.2.4 Line

รายการ	คำอธิบาย
ข้อมูลสรุป	วาดเส้นตรงตำแหน่งที่กำหนด กำหนดชนิดของเส้น, คุณสมบัติของสี พิกัดเริ่มต้นและพิกัดสิ้นสุด
รูปแบบ	dsp_line (พิกัด X ของจุดเริ่มต้น, พิกัด Y ของจุดเริ่มต้น, พิกัด X ของจุดสิ้นสุด, พิกัด Y ของจุดสิ้นสุด, การกะพริบของสีที่แสดงผล + สีที่แสดงผล, การกะพริบของสีพื้นหลัง + สีพื้นหลัง, ชนิดของเส้นและลูกศร)  • เมื่อตั้งค่าเป็นสีดำและให้มีการกระพริบด้วย สีพื้นหลังจะกลายเป็นโปร่งใส

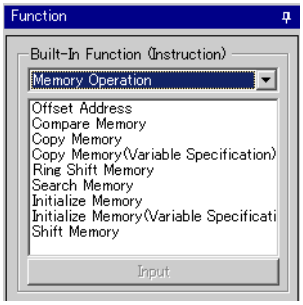
21.2.5 Rectangle

รายการ	คำอธิบาย
ข้อมูลสรุป	วาดสี่เหลี่ยมผืนผ้าตรงตำแหน่งที่กำหนด เมื่อคุณทำเครื่องหมายที่ช่อง [Pattern] จะเป็นการวาดสี่เหลี่ยมผืนผ้าทึบ ให้เลือกและป้อนชนิดของเส้น (หรือรูปแบบการเติมเมื่อเลือกรูปแบบ), คุณสมบัติของสี พิกัดเริ่มต้นและพิกัดสิ้นสุด
รูปแบบ	dsp_rectangle (พิกัด X ของจุดเริ่มต้น, พิกัด Y ของจุดเริ่มต้น, พิกัด X ของจุดสิ้นสุด, พิกัด Y ของจุดสิ้นสุด, การกะพริบของสีที่แสดงผล + สีที่แสดงผล, การกะพริบของสีพื้นหลัง + สีพื้นหลัง, รูปแบบและชนิดของเส้น)  • เมื่อตั้งค่าการกะพริบและสีดำพร้อมกัน สีพื้นหลังจะโปร่งใส

ข้อสำคัญ

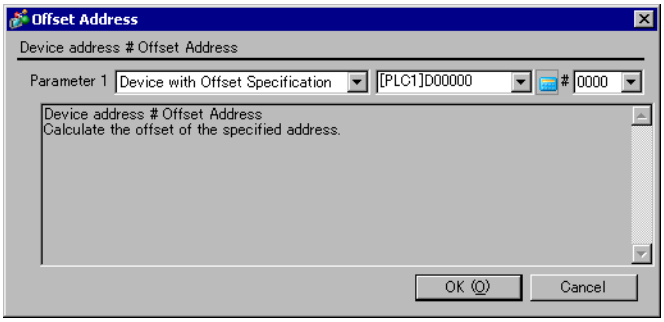
- เมื่อกำหนดรหัสสีในฟังก์ชันการวาด ให้ตั้งค่ารหัสสีระหว่าง 0 ถึง 255 หากคุณตั้งค่าเป็น E1 ถึง E12 และบันทึกสคริปต์ จะเกิดข้อผิดพลาดขึ้น

21.3 การทำงานของหน่วยความจำ

Memory Operation	ข้อมูลสรุปของฟังก์ชัน
	Offset Address ☞ **"**21.3.1 Offset Address**"** (หน้า 21-8) **กำหนดค่าออฟเซ็ตของตำแหน่ง**
	Compare Memory ☞ **"**21.3.2 Compare Memory**"** (หน้า 21-9) **เปรียบเทียบข้อมูลสองบล็อคตรงตำแหน่งที่ระบุ (ออฟเซ็ต) และเขียนผลการเปรียบเทียบลงในตำแหน่งจัดเก็บ**
	Copy Memory ☞ **"**21.3.3 Copy Memory**"** (หน้า 21-11) **คัดลอกหน่วยความจำของอุปกรณ์โดยทำเพียงครั้งเดียว**
	Copy Memory (Variable Specification) ☞ **"**21.3.4 Copy Memory (Variable)**"** (หน้า 21-14) **คัดลอกหน่วยความจำของอุปกรณ์โดยทำเพียงครั้งเดียว คุณสามารถแก้ไขตำแหน่งต้นทาง (ที่ถูกคัดลอก), ตำแหน่งปลายทาง (ที่คัดลอกแล้ว) และจำนวนตำแหน่งได้**
	Ring Shift Memory ☞ **"**21.3.5 Memory Ring**"** (หน้า 21-15) **เลื่อนข้อมูลในหน่วยความจำเป็นวงตามจำนวนเวิร์ดบล็อคที่กำหนด**
	Search Memory ☞ **"**21.3.6 Search Memory**"** (หน้า 21-18) **ค้นหาข้อมูลในบล็อคต่าง ๆ และส่ง (บันทึก) ผลการค้นหาไปที่ตำแหน่งจัดเก็บข้อมูลที่ระบุไว้**
	Initialize Memory ☞ **"**21.3.7 Initialize Memory**"** (หน้า 21-21) **เริ่มการทำงานของอุปกรณ์ทั้งหมดพร้อมกัน**
	Initialize Memory (Variable Specification) ☞ **"**21.3.8 Initialize Memory (Variable)**"** (หน้า 21-22) **เริ่มการทำงานของอุปกรณ์ทั้งหมดพร้อมกัน คุณสามารถแก้ไขตำแหน่งเริ่มต้น, ข้อมูลที่ตั้งค่าไว้ และจำนวนตำแหน่งได้**
	Shift Memory ☞ **"**21.3.9 Shift Memory**"** (หน้า 21-23) **เลื่อนบล็อคขึ้นข้างบน**

21.3.1 Offset Address

รายการ	คำอธิบาย
ข้อมูลสรุป	ฟังก์ชันนี้ใช้สำหรับกำหนดตำแหน่งออฟเซ็ต โดยตำแหน่งจัดเก็บค่าออฟเซ็ตสามารถกำหนดได้เฉพาะตำแหน่งเวิร์ดชั่วคราวเท่านั้น
รูปแบบ	[ตำแหน่งอุปกรณ์] # [ตำแหน่งออฟเซ็ต]



ช่วงการป้อนค่าคงที่

Data Type	Constant Input	
	ต่ำสุด	สูงสุด
Bin16	0	65535
Bin32	0	4294967295
Bin16+/-	−32768	32767
Bin32+/-	−2147483648	2147483647
BCD16	0	9999
BCD32	0	99999999

ตัวอย่างนิพจน์ 1

[w:[PLC1]D0200]=[w:[PLC1]D0100]#[t:0000]

จากตัวอย่างข้างบน เมื่อค่าของ [t:0000] คือ 2 ค่าที่จัดเก็บไว้ในตำแหน่ง D0102 จะถูกออฟเซ็ตให้ตำแหน่ง D0200

ตัวอย่างนิพจน์ 2

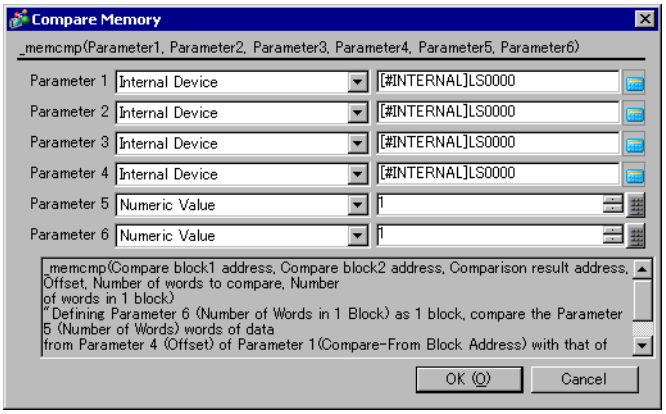
[w:[PLC1]D0100]#[t:0000]=30

จากตัวอย่างข้างบน เมื่อค่าของ [t:0000] คือ 8 จะออฟเซ็ตค่า 30 ให้ตำแหน่ง D0108

ข้อสำคัญ

- ตำแหน่งเวิร์ดที่ใช้ในรูปแบบตำแหน่งออฟเซ็ตจะไม่นับเป็นตำแหน่งของ D-Script
- อุปกรณ์ที่เชื่อมต่ออยู่จะไม่อ่านข้อมูลอย่างต่อเนื่องจากอุปกรณ์ที่กำหนดโดยตำแหน่งออฟเซ็ต โดยจะอ่านข้อมูลเมื่อใช้งาน D-Script เมื่อเกิดข้อผิดพลาดระหว่างการอ่าน ค่าที่อ่านได้จะถือเป็น “0” นอกจากนี้ บิต 12 ของ LS2032 ซึ่งเป็นรีเลย์พิเศษภายในของจอแสดงผลจะเปิดขึ้น เมื่ออ่านข้อมูลเสร็จสมบูรณ์ตามปกติ บิต 12 จะปิดลง
- หากผลการดำเนินการเกินกว่า 16 บิต (ค่าสูงสุด: 65535) ระบบจะถือว่าบิต 1 ถึงบิต 15 เป็นบิตที่ถูกต้อง โดยไม่สนใจบิต 16 และบิตอื่นๆ

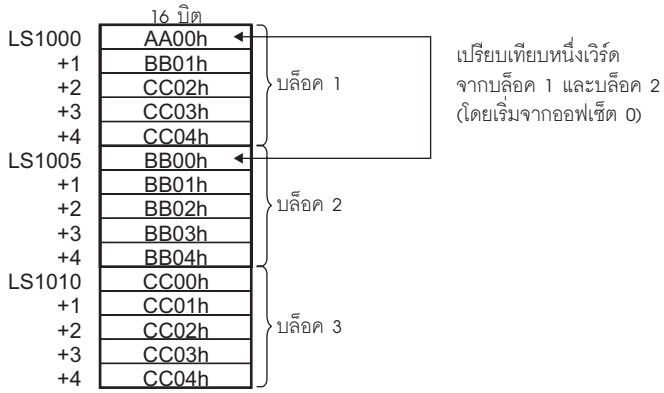
21.3.2 Compare Memory

รายการ	คำอธิบาย
ข้อมูลสรุป	เปรียบเทียบข้อมูลสองบล็อคตรงตำแหน่งที่ระบุ (ออฟเซ็ต) และเขียนผลการเปรียบเทียบลงในตำแหน่งจัดเก็บ ระบบจะจัดเก็บผลการเปรียบเทียบด้วยค่าต่อไปนี้ ได้แก่ “0” เมื่อค่าเท่ากัน, “1” เมื่อข้อมูลเป้าหมายมากกว่าข้อมูลเดิม, “2” เมื่อข้อมูลเป้าหมายน้อยกว่าข้อมูลเดิม เมื่อเกิดข้อผิดพลาด ค่าสถานะข้อผิดพลาดจะถูกเขียนลงใน LS9152
รูปแบบ	_memcmp ([ตำแหน่งบล็อคที่จะเปรียบเทียบ], [ตำแหน่งบล็อคที่ใช้เปรียบเทียบ], [ตำแหน่งจัดเก็บผลการเปรียบเทียบ], ออฟเซ็ตจากบล็อคส่วนบน, จำนวนเวิร์ดที่จะเปรียบเทียบ, จำนวนเวิร์ดใน 1 บล็อค)  Parameter 1: อุปกรณ์ภายใน Parameter 2: อุปกรณ์ภายใน Parameter 3: อุปกรณ์ภายใน Parameter 4: ค่าตัวเลข (0 ถึง 639), อุปกรณ์ภายใน, ตัวแปรชั่วคราว Parameter 5: ค่าตัวเลข (1 ถึง 640) Parameter 6: ค่าตัวเลข (1 ถึง 640) ข้อมูลที่จะจัดเก็บ 0: เหมือนกัน 1: ต้นทางน้อยกว่าเป้าหมาย (ต้นทาง < เป้าหมาย) 2: ต้นทางมากกว่าเป้าหมาย (ต้นทาง > เป้าหมาย)

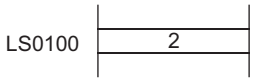
ตัวอย่างนิพจน์ 1

_memcmp ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1005], [w:[#INTERNAL]LS0100], 0, 1, 5)

(เปรียบเทียบเวิร์ดหนึ่งเวิร์ดจากบล็อค 1 และบล็อค 2 (โดยเริ่มจากออฟเซ็ต 0) และบันทึกผลการเปรียบเทียบใน LS0100)



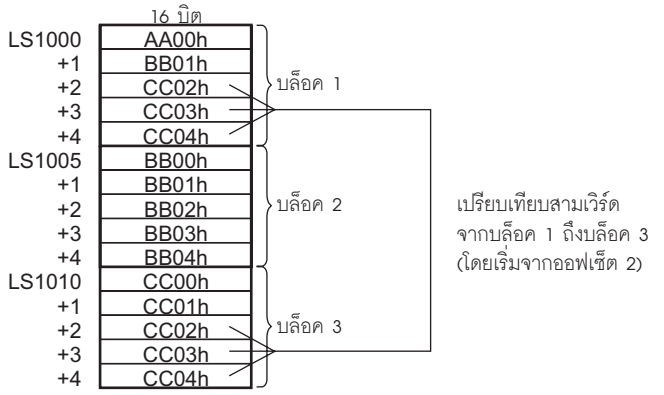
เนื่องจากค่าต้นทางน้อยกว่ากว่าค่าเป้าหมาย จึงจัดเก็บผลการเปรียบเทียบ “2” ไว้ใน LS0100



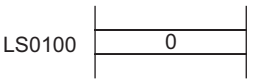
ตัวอย่างนิพจน์ 2

_memcmp ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1010], [w:[#INTERNAL]LS0100], 2, 3, 5)

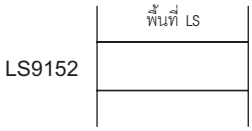
(เปรียบเทียบเวิร์ดสามเวิร์ดจากบล็อค 1 และบล็อค 3 (โดยเริ่มจากออฟเซ็ต 2) และบันทึกผลการเปรียบเทียบใน LS0100)



เนื่องจากค่าของข้อมูลต้นทางและข้อมูลเป้าหมายเหมือนกัน จึงจัดเก็บผลการเปรียบเทียบ “0” ไว้ใน LS0100



สถานะข้อผิดพลาด

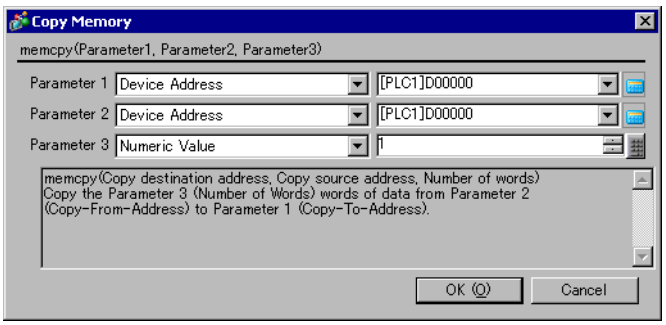


ชื่อฟังก์ชันตัวแก้ไข	พื้นที่ LS	สถานะข้อผิดพลาด	สาเหตุ
_memcmp ( )	LS9152	0000h	เสร็จสมบูรณ์
		0001h	พารามิเตอร์เกิดข้อผิดพลาด
		0003h	การเขียน/การอ่านเกิดข้อผิดพลาด

ข้อสำคัญ

- ช่วงอุปกรณ์ LS ที่สามารถระบุได้นั้นมีเฉพาะอุปกรณ์ในพื้นที่สำหรับผู้ใช้ที่กำหนดเท่านั้น (LS20 ถึง LS2031 และ LS2096 ถึง LS8191)
- หากกำหนดค่าออฟเซ็ตของตอนต้นบล็อคด้วยค่าที่มากกว่าจำนวนเวิร์ดในหนึ่งบล็อค คุณสมบัตินี้จะไม่ทำงาน
- หากกำหนดจำนวนเวิร์ดที่จะเปรียบเทียบมากเกินกว่าจำนวนเวิร์ดในหนึ่งบล็อค คุณสมบัตินี้จะไม่ทำงาน

21.3.3 Copy Memory

รายการ	คำอธิบาย
ข้อมูลสรุป	คัดลอกหน่วยความจำของอุปกรณ์โดยทำเพียงครั้งเดียว ระบบจะคัดลอกข้อมูลจำนวน ตำแหน่งไปยังตำแหน่งเวิร์ดของปลายทางการคัดลอก โดยเริ่มจากตำแหน่งเวิร์ดแรกสุด ของข้อมูลต้นทาง คุณสามารถป้อนจำนวนตำแหน่งได้ตั้งแต่ 1 ถึง 640
รูปแบบ	memcpy ([ตำแหน่งปลายทางการคัดลอก], [ตำแหน่งที่ถูกคัดลอก], เวิร์ด) 

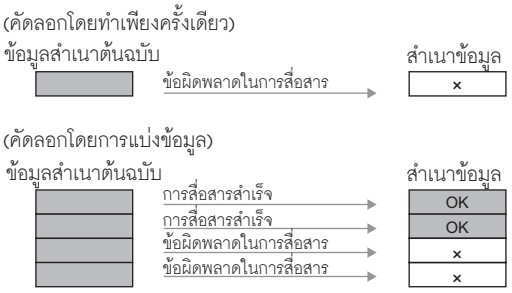
ตัวอย่างนิพจน์

memcpy ([w:[PLC1]D0200], [w:[PLC1]D0100], 10)

จากตัวอย่างข้างบน ข้อมูลถูกคัดลอกจากตำแหน่ง D0100-D0109 ไปยังตำแหน่ง D0200-D0209

ข้อสำคัญ

- เมื่อจำเป็น ข้อมูลสำเนาต้นฉบับจะถูกอ่านจากอุปกรณ์ที่เชื่อมต่อเพียงครั้งเดียวเท่านั้น หากเกิดข้อผิดพลาดในการสื่อสารระหว่างการอ่านข้อมูล บิต 12 ของ LS2032 ซึ่งเป็นรีเลย์พิเศษภายในของจอแสดงผลจะเปิดขึ้น เมื่ออ่านข้อมูลเสร็จสมบูรณ์ตามปกติ บิต 12 จะปิดลง
- การอ่านจากข้อมูลสำเนาต้นฉบับและการเขียนข้อมูลไปยังปลายทางจะทำเพียงครั้งเดียว หรือทำโดยแบ่งข้อมูลออกเป็นรายการต่างๆ ให้เท่ากับจำนวนตำแหน่งที่ใช้สำหรับข้อมูลสำเนาต้นฉบับ หากเกิดข้อผิดพลาดในการสื่อสารระหว่างการอ่านข้อมูล ผลการคัดลอกข้อมูลจะแตกต่างกันดังต่อไปนี้ โดยขึ้นอยู่กับว่าจะประมวลผลข้อมูลโดยทำเพียงครั้งเดียว หรือแบ่งข้อมูลเป็นรายการ: (ผลการคัดลอกข้อมูล OK: คัดลอกถูกต้อง, x: ไม่มีการคัดลอกข้อมูล)

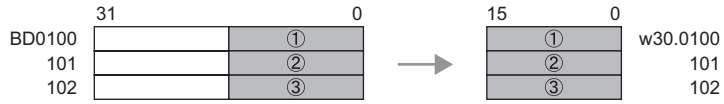


- เมื่อจำนวนตำแหน่งเพิ่มขึ้น การเขียนข้อมูลลงใน PLC จำเป็นต้องใช้เวลานานขึ้น โดยอาจใช้เวลาตั้งแต่ 20 วินาทีจนถึงหลายนาที ขึ้นอยู่กับจำนวนของตำแหน่ง
- หากข้อมูลที่จะเขียนมีมากกว่าช่วงอุปกรณ์ที่กำหนดไว้ จะเกิดข้อผิดพลาดในการสื่อสาร ในกรณีนี้ คุณต้องปิดเครื่อง GP แล้วเปิดอีกครั้งเพื่อตั้งค่า GP ใหม่หลังจากเกิดข้อผิดพลาด
- เมื่อเขียนข้อมูลลงในพื้นที่ LS ด้วยฟังก์ชัน Copy Memory (memcpy) จะสามารถเขียนข้อมูลได้เฉพาะในพื้นที่สำหรับผู้ใช้เท่านั้น คุณไม่สามารถเขียนข้อมูลลงในพื้นที่เก็บข้อมูลระบบ (LS0000 ถึง LS0019), พื้นที่พิเศษ (LS2032 ถึง LS2047) หรือพื้นที่สำรอง (LS2048 ถึง LS2095) แต่สามารถอ่านข้อมูลจากพื้นที่เหล่านี้ได้

ต่อ

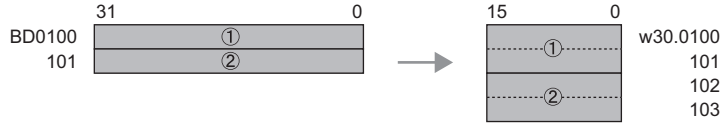
ข้อสำคัญ

- เมื่อคัดลอกข้อมูลของอุปกรณ์ชนิด 32 บิต → อุปกรณ์ชนิด 16 บิตโดยใช้ D-Script และกำหนดความยาวบิตเป็น 16 บิต ระบบจะคัดลอกเฉพาะข้อมูลของ 16 บิตล่างเท่านั้น
- ตัวอย่าง: memcpy ([w:[PLC1]w30.0100], [w:[PLC1]BD0100], 3)

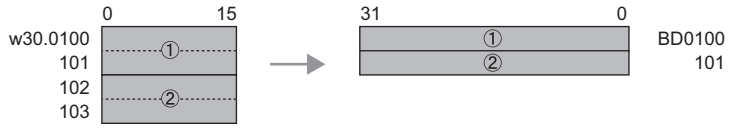


- นอกจากนี้ เมื่อคัดลอกข้อมูลของอุปกรณ์ชนิด 16 บิตไปยังอุปกรณ์ชนิด 32 บิต ระบบจะคัดลอกข้อมูลไปยัง 16 บิตล่างและกำหนดค่า "0" ให้กับข้อมูลของ 16 บิตบน
- ตัวอย่าง: memcpy ([w:[PLC1]BD0100], [w:[PLC1]w30.0100], 3)

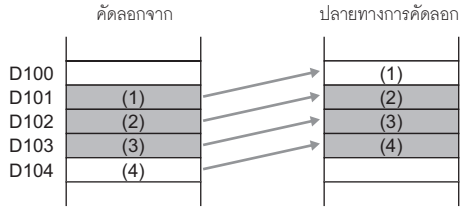
- เมื่อคัดลอกข้อมูลของอุปกรณ์ชนิด 32 บิต → อุปกรณ์ชนิด 16 บิต หรือเมื่อคัดลอกข้อมูลของอุปกรณ์ชนิด 16 บิต → อุปกรณ์ชนิด 32 บิต หากกำหนดความยาวบิตของ D-Script ไว้เท่ากับ 32 การคัดลอกจะเป็นดังต่อไปนี้ เมื่ออุปกรณ์หนึ่งเป็นอุปกรณ์ชนิด 32 บิต และอีกชิ้นหนึ่งเป็นอุปกรณ์ชนิด 16 บิต ให้ใช้จำนวน ตำแหน่งของอุปกรณ์ชนิด 16 บิต เพื่อกำหนดจำนวนตำแหน่งของฟังก์ชัน memcpy ( ) memcpy ( )
- ตัวอย่าง: memcpy ([w:[PLC1]w30.0100], [w:[PLC1]BD0100], 4)



- ตัวอย่าง: memcpy ([w:[PLC1]BD0100], [w:[PLC1]w30.0100], 4)



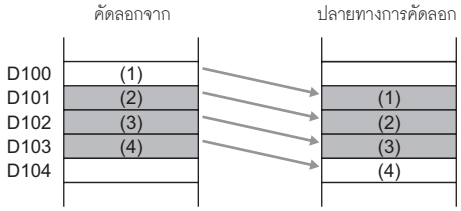
- หากข้อมูลต้นฉบับและข้อมูลปลายทางมีช่วงข้อมูลคาบเกี่ยวกัน ข้อมูลที่คาบเกี่ยวกันทั้งหมดจะถูกเขียนใหม่ดังนี้
- ตัวอย่าง: เมื่อคัดลอก D101-D104 ไปยัง D100-D103
- ข้อมูลถูกคัดลอกลงในตำแหน่งที่มีหมายเลขน้อยกว่า



ต่อ

ข้อสำคัญ

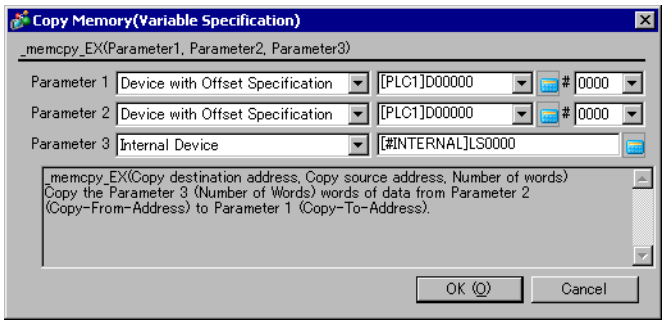
ตัวอย่าง: เมื่อคัดลอก D100-D103 ไปยัง D101-D104
ข้อมูลถูกคัดลอกลงในตำแหน่งที่มีหมายเลขมากกว่า



- แม้ว่าฟังก์ชันของตัวอย่างนี้จะกำหนดไว้ 2 ตำแหน่ง แต่ตำแหน่งดังกล่าวจะไม่นับเป็นตำแหน่งของ D-Script
- เมื่อใช้ตำแหน่งอุปกรณ์กับคำสั่ง Assign การสื่อสารกับอุปกรณ์/PLC อาจทำให้มีการหน่วงเวลาในการกำหนดค่า

•

21.3.4 Copy Memory (Variable)

รายการ	คำอธิบาย
ข้อมูลสรุป	คัดลอกหน่วยความจำของอุปกรณ์โดยทำเพียงครั้งเดียว ข้อมูลของตำแหน่งที่ระบุใน Parameter 3 จะถูกคัดลอกจากตำแหน่งเวิร์ดต้นทางที่ระบุใน Parameter 2 ไปยังตำแหน่งเวิร์ดปลายทางที่ระบุใน Parameter 1 คุณสามารถป้อนจำนวนตำแหน่งได้ตั้งแต่ 1 ถึง 640 ฟังก์ชัน “_memcpy_EX” ช่วยให้กำหนดตำแหน่งต้นทาง ตำแหน่งปลายทาง และจำนวนตำแหน่งโดยทางอ้อมได้
รูปแบบ	_memcpy_EX ([ตำแหน่งปลายทางการคัดลอก], [ตำแหน่งที่ถูกคัดลอก], เวิร์ด) Parameter 1: ตำแหน่งอุปกรณ์ + ตำแหน่งชั่วคราว Parameter 2: ตำแหน่งอุปกรณ์ + ตำแหน่งชั่วคราว Parameter 3: ค่าตัวเลข, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 3 คือตั้งแต่ 1 ถึง 640) 

ตัวอย่างนิพจน์

[t:0000]=10, [t:0001]=20

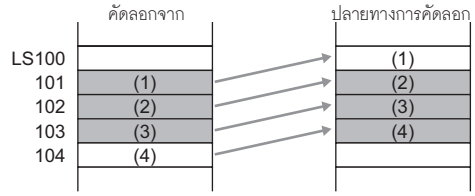
_memcpy_EX ([w:[#INTERNAL]LS 0100]#[t:0000], [w:[PLC1]D0100]#[t:0001], 5)

จากตัวอย่างข้างบน ระบบจะอ่านข้อมูลจำนวนห้าเวิร์ดจากตำแหน่ง D0120 และเขียนลงใน LS0110 ถึง LS0114

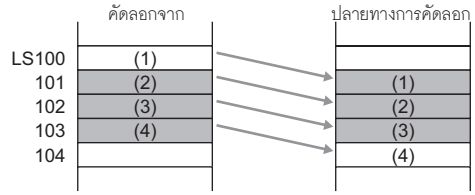
ข้อสำคัญ

- หากข้อมูลต้นฉบับและข้อมูลปลายทางมีช่วงข้อมูลคาบเกี่ยวกัน ข้อมูลที่คาบเกี่ยวกันทั้งหมดจะถูกเขียนใหม่ดังนี้

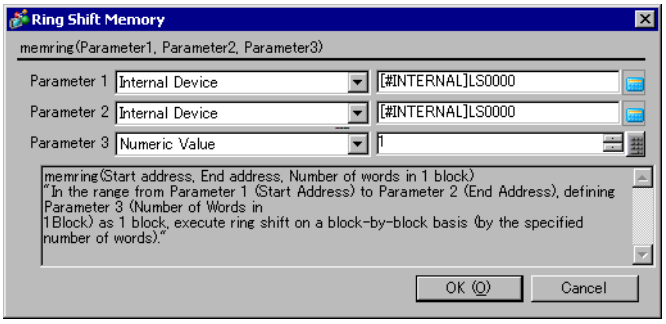
ตัวอย่าง: เมื่อคัดลอกจาก LS101-LS104 ไปยัง LS100-LS103
ข้อมูลถูกคัดลอกลงในตำแหน่งที่มีหมายเลขน้อยกว่า



ตัวอย่าง: เมื่อคัดลอกจาก LS100-LS103 ไปยัง LS101-LS104
ข้อมูลถูกคัดลอกลงในตำแหน่งที่มีหมายเลขมากกว่า



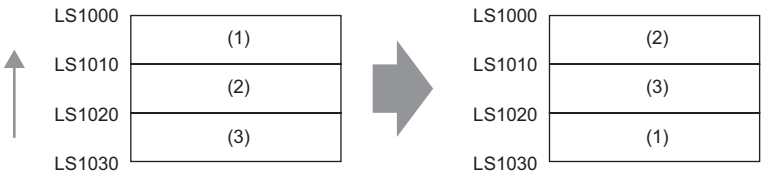
21.3.5 Memory Ring

รายการ	คำอธิบาย
ข้อมูลสรุป	**เลื่อนข้อมูลในหน่วยความจำในบล็อคให้เป็นวง** **ฟังก์ชันนี้จะเลื่อนข้อมูลให้เป็นวงระหว่างตำแหน่งเริ่มต้นและตำแหน่งสิ้นสุดในบล็อค (ตามจำนวนเวิร์ดที่กำหนด) เมื่อเกิดข้อผิดพลาด สถานะข้อผิดพลาดจะถูกเขียนลงใน LS9150**
รูปแบบ	memring ([ตำแหน่งเริ่มต้น], [ตำแหน่งสิ้นสุด], จำนวนเวิร์ดใน 1 บล็อค)  Parameter 1: **อุปกรณ์ภายใน** Parameter 2: **อุปกรณ์ภายใน** Parameter 3: **ค่าตัวเลข (1 ถึง 640)** **- เมื่อ Parameter 1 น้อยกว่า Parameter 2 (P1 < P2) ข้อมูลบล็อคจะเลื่อนขึ้นข้างบน** **- เมื่อ Parameter 1 มากกว่า Parameter 2 (P1 > P2) ข้อมูลบล็อคจะเลื่อนลงข้างล่าง** • **โปรดตรวจสอบให้แน่ใจว่าได้ตั้งค่าตำแหน่งเริ่มต้นและตำแหน่งสิ้นสุดเป็นอุปกรณ์ชนิดเดียวกัน (LS หรือ USR)**

ตัวอย่างนิพจน์ 1

memring ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1030], 10)

(เมื่อ Parameter 1 น้อยกว่า Parameter 2 (P1 < P2))

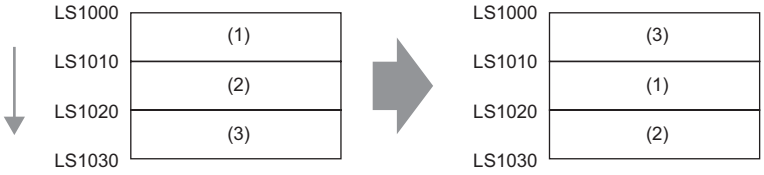


ข้อมูลจะเลื่อนขึ้นข้างบน 10 เวิร์ดบล็อค

ตัวอย่างนิพจน์ 2

memring ([w:[#INTERNAL]LS1030], [w:[#INTERNAL]LS1000], 10)

(เมื่อ Parameter 1 มากกว่า Parameter 2 (P1 > P2))

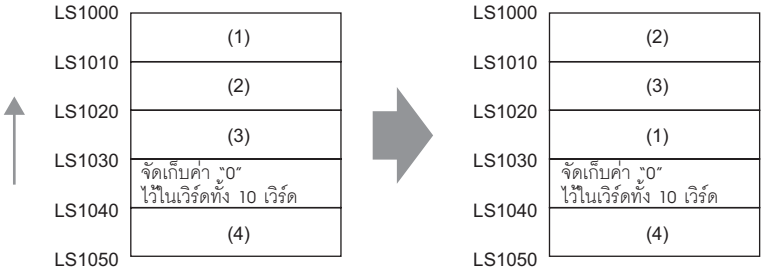


ข้อมูลจะเลื่อนลงข้างล่าง 10 เวิร์ดบล็อค

ตัวอย่างนิพจน์ 3

memring ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1050], 10)

(เมื่อมีบล็อคที่มีเวิร์ดทุกเวิร์ดมีข้อมูลเป็น “0” อยู่ภายในช่วง)

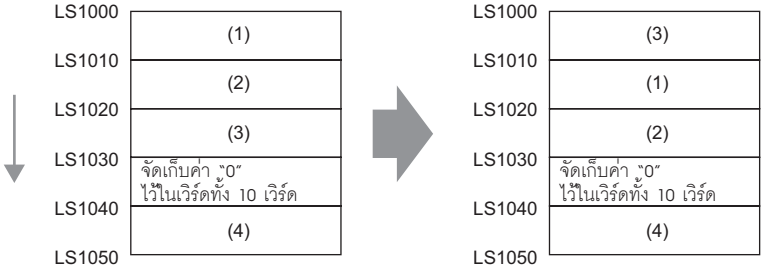


ข้อมูลจะเลื่อนขึ้นข้างบน 10 เวิร์ดบล็อคเท่านั้น โดยเริ่มจากบล็อคเริ่มต้นจนถึงบล็อคที่มีข้อมูลเป็น “0” หากมีข้อมูลอยู่ต่อจากบล็อคที่มีข้อมูลเป็น “0” ระบบจะไม่สนใจข้อมูลนั้น

ตัวอย่างนิพจน์ 4

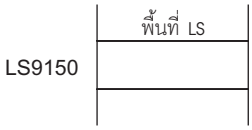
memring ([w:[#INTERNAL]LS1050], [w:[#INTERNAL]LS1000], 10)

(เมื่อมีบล็อคที่มีข้อมูลเป็น “0” อยู่ภายในช่วง)



ข้อมูลจะเลื่อนลงข้างล่าง 10 เวิร์ดบล็อคเท่านั้น โดยเริ่มจากบล็อคเริ่มต้นจนถึงบล็อคที่มีข้อมูลเป็น “0” หากมีข้อมูลอยู่ต่อจากบล็อคที่มีข้อมูลเป็น “0” ระบบจะไม่สนใจข้อมูลนั้น

สถานะข้อผิดพลาด



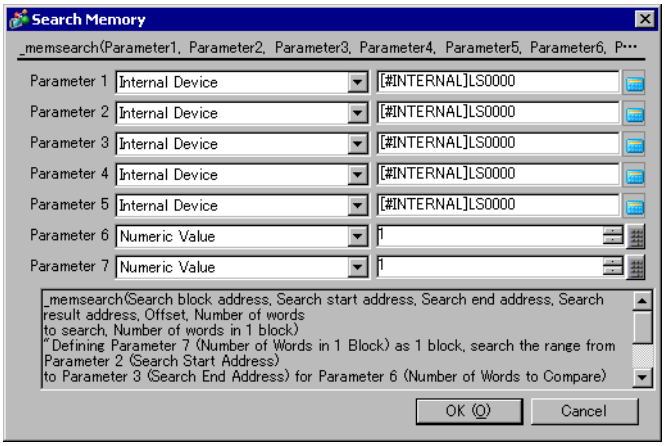
ชื่อฟังก์ชันตัวแก้ไข	พื้นที่ LS	สถานะข้อผิดพลาด	สาเหตุ
memring ( )	LS9150	0000h	เสร็จสมบูรณ์
		0001h	พารามิเตอร์เกิดข้อผิดพลาด
		0003h	การเขียน/การอ่านเกิดข้อผิดพลาด

ข้อสำคัญ

- เวลาที่ต้องใช้ในการประมวลผลจะเป็นสัดส่วนกับช่วงที่กำหนดโดยตำแหน่งเริ่มต้นและตำแหน่งสิ้นสุด ยิ่งกำหนดช่วงไว้ยาว ยิ่งต้องใช้เวลาในการประมวลผลนานขึ้น ระบบจะไม่รีเฟรชพาร์ทจนกว่าจะประมวลผลเสร็จสมบูรณ์
- ช่วงอุปกรณ์ LS ที่สามารถระบุได้นั้นมีเฉพาะอุปกรณ์ในพื้นที่สำหรับผู้ใช้ที่กำหนดเท่านั้น (LS20 ถึง LS2031 และ LS2096 ถึง LS8191)

21.3.6 Search Memory

รายการ	คำอธิบาย
ข้อมูลสรุป	ฟังก์ชันนี้จะค้นหาข้อมูลในบล็อคโดยเริ่มจากรายการแรกในช่วงที่ระบุ โดยการเปรียบเทียบบล็อคข้อมูล โดยเริ่มจากบล็อค (ออฟเซ็ต) ที่ระบุ และส่ง (บันทึก) ผลการค้นหาไปที่ตำแหน่งจัดเก็บข้อมูลที่ระบุไว้ เมื่อพบบล็อคที่ตรงกัน จะบันทึกค่าออฟเซ็ตของบล็อคไว้ (1 หรือมากกว่า) เมื่อไม่พบบล็อคที่ตรงกัน จะบันทึก “FFFFh” ไว้ เมื่อเกิดข้อผิดพลาด ค่าสถานะข้อผิดพลาดจะถูกเขียนลงใน LS9153
รูปแบบ	_memsearch ([ตำแหน่งบล็อคที่ค้นหา], [ตำแหน่งเริ่มต้นการค้นหา], [ตำแหน่งสิ้นสุดการค้นหา], [ตำแหน่งจัดเก็บผลการค้นหา], ออฟเซ็ตจากบล็อคเริ่มต้น, จำนวนเวิร์ดที่จะเปรียบเทียบ, จำนวนเวิร์ดใน 1 บล็อค)



Parameter 1: อุปกรณ์ภายใน
Parameter 2: อุปกรณ์ภายใน
Parameter 3: อุปกรณ์ภายใน
Parameter 4: อุปกรณ์ภายใน
Parameter 5: ค่าตัวเลข (0 ถึง 639), อุปกรณ์ภายใน, ตัวแปรชั่วคราว
Parameter 6: ค่าตัวเลข (1 ถึง 640)
Parameter 7: ค่าตัวเลข (1 ถึง 640)

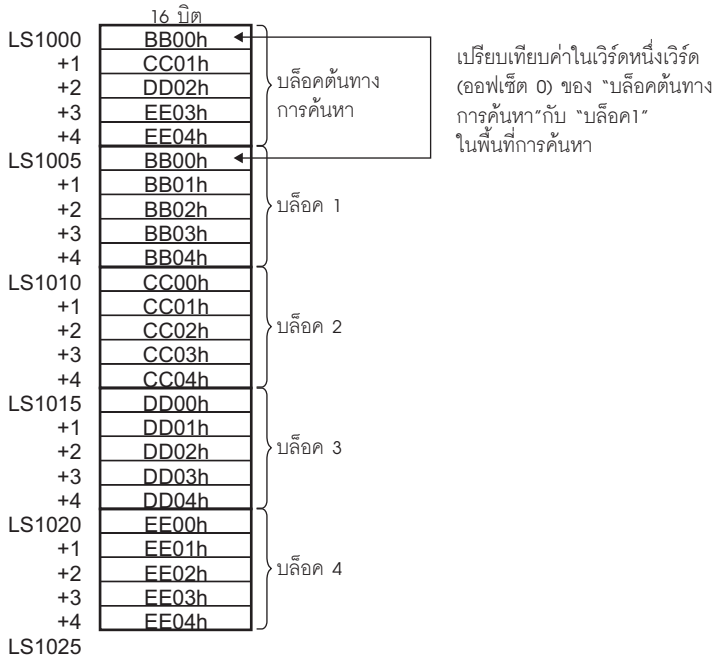
ข้อมูลที่จะถูกเขียน
เมื่อมีบล็อคที่ตรงกัน: ค่าออฟเซ็ตของบล็อค (“1” หรือสูงกว่า)
เมื่อไม่มีบล็อคที่ตรงกัน: “FFFFh”

- โปรดตรวจสอบให้แน่ใจว่าได้ตั้งค่าตำแหน่งเริ่มต้นการค้นหาและตำแหน่งสิ้นสุดการค้นหาเป็นอุปกรณ์ชนิดเดียวกัน (LS หรือ USR) อย่างไรก็ตาม สามารถตั้งค่า [Searched Block Address] และ [Search Result Storage Address] ให้เป็นอุปกรณ์ภายในได้
- ค่า [Parameter 2] ต้องน้อยกว่า [Parameter 3] (Parameter 2 < Parameter 3) มิฉะนั้นจะเกิดข้อผิดพลาด

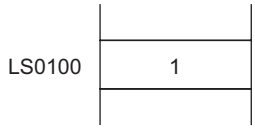
ตัวอย่างนิพจน์ 1

_memsearch ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1005], [w:[#INTERNAL]LS1025],[w:[#INTERNAL]LS0100], 0, 1, 5)

(ค้นหาระหว่าง LS1005 ถึง LS1025 เพื่อหาบล็อคที่มีค่าเดียวกัน โดยเริ่มต้นจากออฟเซ็ต 0 ของบล็อคต้นทางการค้นหา และจัดเก็บผลการค้นหาไว้ใน LS0100)



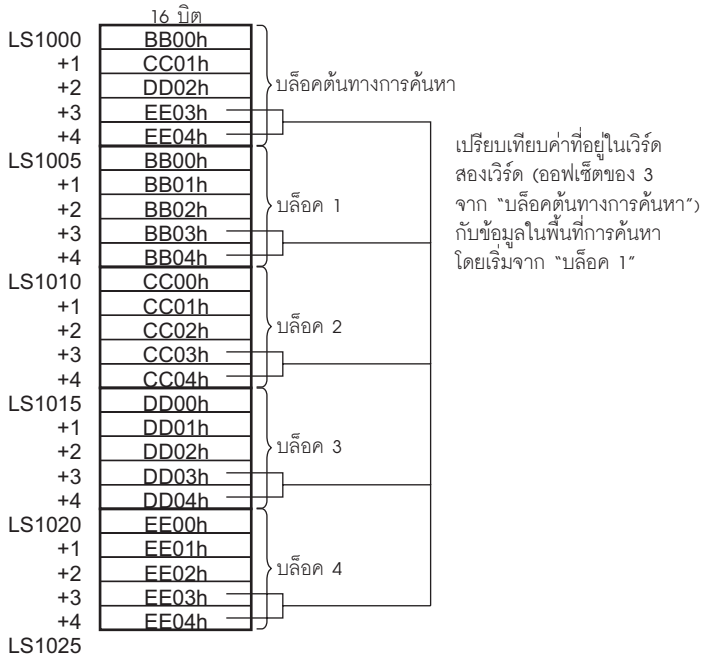
ในกรณีนี้ ค่าของ "บล็อค 1" ตรงกับค่าของ "บล็อคต้นทางการค้นหา" ดังนั้น จะจัดเก็บผลการค้นหา "1" ไว้ใน LS0100



ตัวอย่างนิพจน์ 2

_memsearch ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1005], [w:[#INTERNAL]LS1025],
[w:[#INTERNAL]LS0100], 3, 2, 5)

(ค้นหาระหว่าง LS1005 ถึง LS1025 เพื่อหาบล็อคที่มีค่าเดียวกัน โดยใช้เวิร์ดสองเวิร์ด โดยเริ่มค้นหาจากออฟเซ็ตของ 3 และจัดเก็บผลลัพธ์ไว้ใน LS0100)



ในกรณีนี้ ค่าของ "บล็อค 4" จะตรงกับค่าของ "บล็อคต้นทางการค้นหา" ดังนั้น จะจัดเก็บผลการค้นหา "4" ไว้ใน LS0100

สถานะข้อผิดพลาด

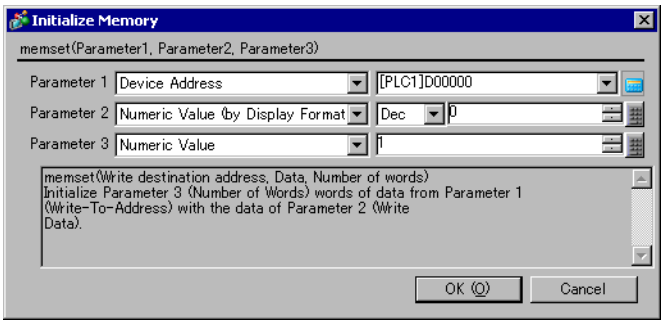
	พื้นที่ LS
LS9153	

ชื่อฟังก์ชันตัวแก้ไข	พื้นที่ LS	สถานะข้อผิดพลาด	สาเหตุ
_memsearch ( )	LS9153	0000h	**เสร็จสมบูรณ์**
		0001h	**พารามิเตอร์เกิดข้อผิดพลาด**
		0003h	**การเขียน/การอ่านเกิดข้อผิดพลาด**

ข้อสำคัญ

- เวลาที่ต้องใช้ในการประมวลผลจะเป็นสัดส่วนกับช่วงที่กำหนดโดยตำแหน่งเริ่มต้นและตำแหน่งสิ้นสุด ยิ่งกำหนดช่วงไว้ยาว ยิ่งต้องใช้เวลาในการประมวลผลนานขึ้น ระบบจะไม่รีเฟรชพาร์ทจนกว่าจะประมวลผลเสร็จสมบูรณ์
- ช่วงอุปกรณ์ LS ที่สามารถระบุได้นั้นมีเฉพาะอุปกรณ์ในพื้นที่สำหรับผู้ใช้ที่กำหนดเท่านั้น (LS20 ถึง LS2031 และ LS2096 ถึง LS8191)

21.3.7 Initialize Memory

รายการ	คำอธิบาย
ข้อมูลสรุป	**เริ่มการทำงานของอุปกรณ์ทั้งหมดพร้อมกัน ข้อมูลการตั้งค่าจำนวนตำแหน่ง จะได้มาจากตำแหน่งเวิร์ดที่ตั้งค่า คุณสามารถป้อนจำนวนตำแหน่งได้ตั้งแต่ 1 ถึง 640**
รูปแบบ	memset ([ตำแหน่งปลายทางการเขียน], ข้อมูลที่เขียน, เวิร์ด) 

ตัวอย่างนิพจน์

memset ([w:[PLC 1]D0100], 0, 10)

จากตัวอย่างข้างบน เป็นการตั้งค่า “0” ให้กับตำแหน่ง D0100 ถึง D0109

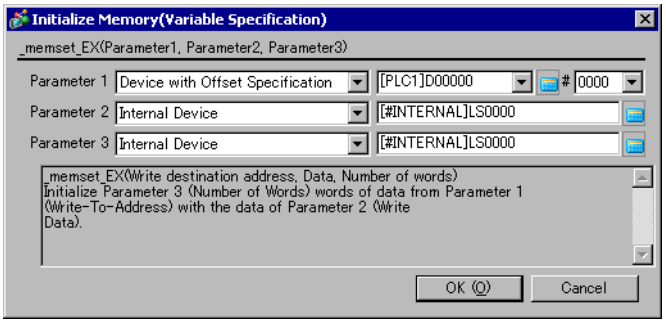
ข้อสำคัญ

- เมื่อจำนวนตำแหน่งเพิ่มขึ้น การเขียนข้อมูลลงใน PLC จำเป็นต้องใช้เวลานานขึ้น โดยอาจใช้เวลาตั้งแต่ 20 วินาทีจนถึงหลายนาที ขึ้นอยู่กับจำนวนของตำแหน่ง
- หากข้อมูลที่จะเขียนมีมากกว่าช่วงอุปกรณ์ที่กำหนดไว้ จะเกิดข้อผิดพลาดในการสื่อสาร ในกรณีนี้ คุณต้องปิดเครื่อง GP แล้วเปิดอีกครั้งเพื่อตั้งค่า GP ใหม่หลังจากเกิดข้อผิดพลาด
- แม้ว่าฟังก์ชันนี้จะกำหนดตำแหน่งไว้ แต่ตำแหน่งดังกล่าวจะไม่นับเป็นตำแหน่งของ D-Script
- เมื่อเขียนข้อมูลลงในพื้นที่ LS ด้วยฟังก์ชัน Memory Reset (memset) จะสามารถเขียนข้อมูลได้เฉพาะในพื้นที่สำหรับผู้ใช้เท่านั้น คุณไม่สามารถเขียนข้อมูลลงในพื้นที่เก็บข้อมูลระบบ (LS0000 ถึง LS0019), พื้นที่พิเศษ (LS2032 ถึง S2047) หรือพื้นที่สำรอง (LS2048 ถึง LS2095)
- เมื่อใช้ตำแหน่งอุปกรณ์กับคำสั่ง Assign ระบบจะไม่กำหนดค่าที่เขียนให้โดยทันที เนื่องจากต้องใช้เวลาในการส่งข้อมูลจาก GP ไปยัง PLC
  (ตัวอย่าง)

```
memset ([w:D0100], 0, 10)   // Initializes "D100 to D109" to 0
[w:D200] = [w:D100]         // Assigns D100 data to D200.
```

ในตัวอย่างนี้ ค่า 0 ซึ่งเป็นผลการดำเนินการที่เขียนลงใน D100 นั้นยังไม่ถูกกำหนดลงใน D200

21.3.8 Initialize Memory (Variable)

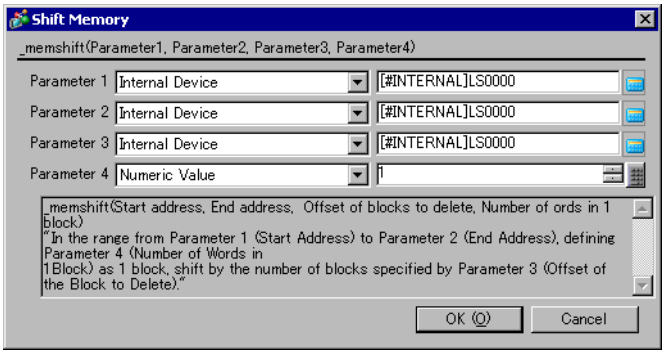
รายการ	คำอธิบาย
ข้อมูลสรุป	**เริ่มการทำงานของอุปกรณ์ทั้งหมดพร้อมกัน ข้อมูลตั้งค่าซึ่งระบุด้วย Parameter 2 จะถูกตั้งค่าจากตำแหน่งเวิร์ดที่ตั้งค่าซึ่งระบุด้วย Parameter 1 ลงในตำแหน่งซึ่งระบุด้วย Parameter 3 คุณสามารถป้อนจำนวนตำแหน่งได้ตั้งแต่ 1 ถึง 640 และยังสามารถกำหนดตำแหน่งปลายทางการเขียน, ข้อมูลที่เขียน และหมายเลขตำแหน่งได้โดยอ้อม**
รูปแบบ	_memset_EX ([ตำแหน่งปลายทางการเขียน], ข้อมูลที่เขียน, เวิร์ด)  Parameter 1: ตำแหน่งอุปกรณ์ + ตำแหน่งชั่วคราว Parameter 2: ค่าตัวเลข, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 2 คือตั้งแต่ 0 ถึง 65535 สำหรับ Dec และตั้งแต่ 0 ถึง FFFF สำหรับ Hex) Parameter 3: ค่าตัวเลข, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 3 คือตั้งแต่ 1 ถึง 640)

ตัวอย่างนิพจน์

[t:0000]=10
[w:LS0050]=0
[w:LS0051]=5
_memset_EX ([w:[#INTERNAL]LS0100]#[t:0000], [w:[#INTERNAL]LS0050], [w:[#INTERNAL]LS0051])

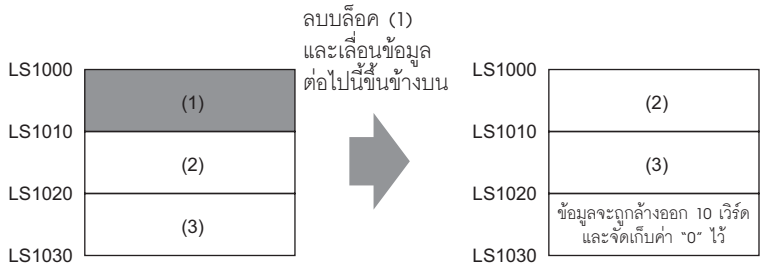
จากตัวอย่างข้างบน ระบบจะเขียนค่า “0” ลงในเวิร์ดห้าเวิร์ดตั้งแต่ LS0100 ถึง LS0114

21.3.9 Shift Memory

รายการ	คำอธิบาย
ข้อมูลสรุป	ลบบล็อคที่ระบุและเลื่อนบล็อคข้อมูลต่อไปนี้ขึ้นข้างบน บล็อคที่จะลบจะถูกกำหนดโดยใช้ออฟเซ็ต เมื่อเกิดข้อผิดพลาด สถานะข้อผิดพลาดจะถูกเขียนลงใน LS9151
รูปแบบ	_memshift ([ตำแหน่งเริ่มต้น], [ตำแหน่งสิ้นสุด], ออฟเซ็ตของบล็อคที่จะลบ, จำนวนเวิร์ด ใน 1 บล็อค)  Parameter 1: อุปกรณ์ภายใน Parameter 2: อุปกรณ์ภายใน Parameter 3: ค่าตัวเลข (1 ถึง 65,535), อุปกรณ์ภายใน, ตัวแปรชั่วคราว Parameter 4: ค่าตัวเลข (1 ถึง 640) • โปรดตรวจสอบให้แน่ใจว่าได้ตั้งค่าตำแหน่งเริ่มต้นและตำแหน่งสิ้นสุดเป็นอุปกรณ์ชนิดเดียวกัน (LS หรือ USR) • ค่า [Parameter 1] ต้องน้อยกว่า [Parameter 2] (Parameter 1 < Parameter 2) มิฉะนั้นจะเกิดข้อผิดพลาด

ตัวอย่างนิพจน์ 1

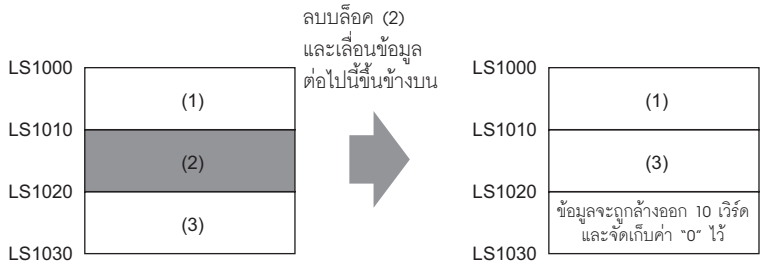
_memshift ([w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS1030], 1, 10)



ข้อมูลเลื่อนขึ้นข้างบนเป็นบล็อค (1 บล็อค = 10 เวิร์ด) และบล็อคสุดท้าย (10 เวิร์ด) ถูกล้างข้อมูลเป็นศูนย์

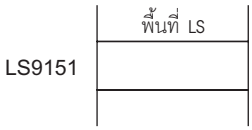
ตัวอย่างนิพจน์ 2

_memshift ([w:[#INTERNAL]LS 1000], [w:[#INTERNAL]LS1030], 2, 10)



ข้อมูลเลื่อนขึ้นข้างบนเป็นบล็อค (1 บล็อค = 10 เวิร์ด) โดยเริ่มจากตำแหน่งออฟเซ็ต 2 และบล็อคสุดท้าย (10 เวิร์ด) ถูกล้างข้อมูลเป็นศูนย์

สถานะข้อผิดพลาด

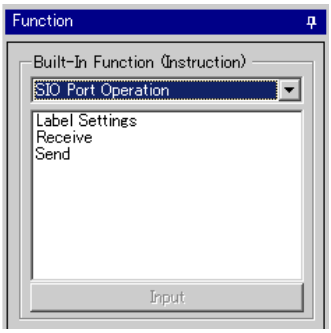


ชื่อฟังก์ชันตัวแก้ไข	พื้นที่ LS	สถานะข้อผิดพลาด	สาเหตุ
_memshift ( )	LS9151	0000h	เสร็จสมบูรณ์
		0001h	พารามิเตอร์เกิดข้อผิดพลาด
		0003h	การเขียน/การอ่านเกิดข้อผิดพลาด

ข้อสำคัญ

- เวลาที่ต้องใช้ในการประมวลผลจะเป็นสัดส่วนกับช่วงที่กำหนดโดยตำแหน่งเริ่มต้นและตำแหน่งสิ้นสุด ยิ่งกำหนดช่วงไว้ยาว ยิ่งต้องใช้เวลาในการประมวลผลนานขึ้น ระบบจะไม่รีเฟรชพาร์ทจนกว่าจะประมวลผลเสร็จสมบูรณ์
- หากกำหนดค่าออฟเซ็ตของบล็อคที่จะลบด้วยค่าที่เกินกว่าช่วงตำแหน่งเริ่มต้นและตำแหน่งสิ้นสุดที่ระบุไว้ คุณสมบัตินี้จะทำงานไม่ถูกต้อง
- ช่วงอุปกรณ์ LS ที่สามารถระบุได้นั้นมีเฉพาะอุปกรณ์ในพื้นที่สำหรับผู้ใช้ที่กำหนดเท่านั้น (LS20 ถึง LS2031 และ LS2096 ถึง LS8191)

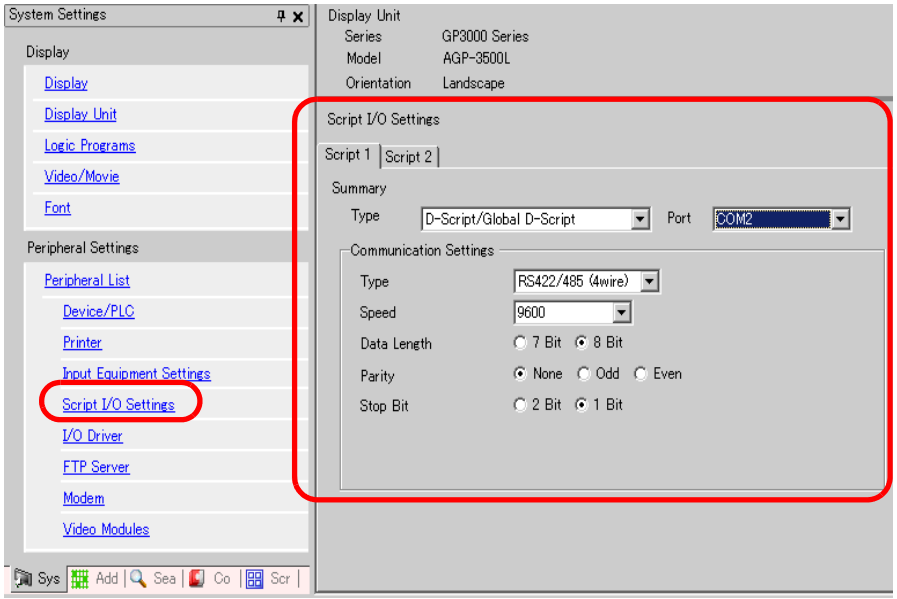
21.4 การทำงานของพอร์ต SIO

การทำงานของพอร์ต SIO	ข้อมูลสรุปของฟังก์ชัน
	Label Settings ☞ "21.4.1 Label Settings" (หน้า 21-27) กำหนดจากตัวแปรควบคุม, สถานะ, จำนวนข้อมูลที่ได้รับ, ฟังก์ชันรับและฟังก์ชันส่ง
	Receive ☞ "21.4.2 Receive" (หน้า 21-30) อ่านข้อมูลที่รับเข้ามาจากพอร์ตอนุกรมที่กำหนด (COM1 หรือ COM2)
	Send ☞ "21.4.3 Send" (หน้า 21-31) เขียนข้อมูลลงในพอร์ตอนุกรมที่กำหนด (COM1 หรือ COM2)
	Extended Receive ☞ "21.4.4 Extended Receive" (หน้า 21-31) อ่านข้อมูลที่รับเข้ามาจากพอร์ตอนุกรมที่กำหนด (COM1 หรือ COM2) ตัวแปรนี้ใช้ได้เฉพาะใน Extended Script เท่านั้น
	Extended Send ☞ "21.4.5 Extended Send" (หน้า 21-32) เขียนข้อมูลลงในพอร์ตอนุกรมที่กำหนด (COM1 หรือ COM2) ตัวแปรนี้ใช้ได้เฉพาะใน Extended Script เท่านั้น
	ฟังก์ชัน Standby Reception ☞ "21.4.6 ฟังก์ชัน Standby Reception" (หน้า 21-33) ระบบอยู่ในโหมดสแตนด์บายการรับจนกว่าจะได้รับสตริงที่ระบุ ตัวแปรนี้ใช้ได้เฉพาะใน Extended Script เท่านั้น
	ฟังก์ชัน Standby ☞ "21.4.7 ฟังก์ชัน Standby" (หน้า 21-34) ระบบหยุดรอ (พักการทำงานชั่วคราว) เป็นระยะเวลาตามที่ระบุไว้จนกว่าจะเริ่มทำงาน ตัวแปรนี้ใช้ได้เฉพาะใน Extended Script เท่านั้น

ข้อสำคัญ

- คุณสามารถใส่ตัวแปร Label Settings, Send และ Receive ไว้ใน D-Script/Global D-Script ได้อย่างง่ายดาย
- หากสื่อสารกับ D-Scripts/Global D-Scripts โปรดกำหนดการตั้งค่าสคริปต์ต่อไปนี้ หากไม่ได้กำหนดการตั้งค่าสคริปต์ จะดำเนินการไม่ได้

[ขั้นตอน I/O ของ D-Script/Global D-Script]
(1) ในหน้า [System Settings] [Script I/O] ให้ตั้งค่า [Type] เป็น [D-Script/Global D-Script]



ใน Script I/O จะมี 2 แท็บ คือ “Script 1” ดังแสดงที่ด้านบน
ตั้งค่า [Port] เป็น COM1 หรือ COM2 แล้วตั้งค่า [Communication Settings] ให้ตรงกับ Extended SIO

- เมื่อสร้างโปรแกรมการสื่อสารด้วยฟังก์ชันการทำงานที่สูงกว่าการทำงานของพอร์ต SIO ขอแนะนำให้ใช้ [Extended Script] สำหรับตัวอย่างที่ใช้ Extended Script โปรดดูที่ ☞“20.5 การสื่อสารกับอุปกรณ์ต่อพ่วงที่ไม่รองรับ” (หน้า 20-21)

21.4.1 Label Settings

◆ **ตัวแปรควบคุม**

เมื่อกำหนดบิต: [c:EXT_SIO_CTRL**] (เขียนอย่างเดียว)
เมื่อกำหนดเวิร์ด: [c:EXT_SIO_CTRL] (เขียนอย่างเดียว)

◆ **สถานะ**

เมื่อกำหนดบิต: [s:EXT_SIO_STAT**] (อ่านอย่างเดียว)
เมื่อกำหนดเวิร์ด: [s:EXT_SIO_STAT] (อ่านอย่างเดียว)

◆ **จำนวนข้อมูลที่ได้รับ**

[r:EXT_SIO_RCV] (อ่านอย่างเดียว)

◆ **ฟังก์ชันรับ**

IO_READ ([p:EXT_SIO], ตำแหน่งจัดเก็บของ LS, จำนวนไบต์)

◆ **ฟังก์ชันส่ง**

IO_WRITE ([p:EXT_SIO], ตำแหน่งจัดเก็บของ LS, จำนวนไบต์)

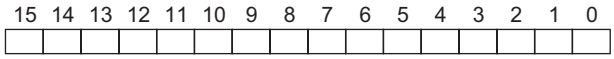
■ **ตัวแปรควบคุม**

รายการ	คำอธิบาย
ข้อมูลสรุป	ตัวแปรควบคุมนี้ใช้สำหรับล้างข้อมูลในบัฟเฟอร์การส่งข้อมูล บัฟเฟอร์การรับข้อมูล และสถานะข้อผิดพลาด ตัวแปรควบคุมนี้เป็นแบบเขียนอย่างเดียว
รูปแบบ	เมื่อกำหนดบิต: [c:EXT_SIO_CTRL**] (**: 00 ถึง 15) เมื่อกำหนดเวิร์ด: [c:EXT_SIO_CTRL]

ตัวอย่างนิพจน์

เมื่อกำหนดบิต: [c:EXT_SIO_CTRL00] = 1
เมื่อกำหนดเวิร์ด: [c:EXT_SIO_CTRL] = 0x0007

EXT_SIO_CTRL



บิต	รายละเอียด
15	สำรอง
14	
13	
12	
11	
10	
9	
8	
7	
6	
5	
4	
3	1: ล้างการรับที่เกิดไทม์เอาต์
2	1: ล้างข้อผิดพลาด
1	1: ล้างบัฟเฟอร์การรับข้อมูล
0	1: ล้างบัฟเฟอร์การส่งข้อมูล

หมายเหตุ

- **เมื่อกำหนดเวิร์ด (เมื่อตั้งค่าบิตพร้อมกันสองบิตขึ้นไป) จะทำการประมวลผลตามลำดับดังนี้: ล้างข้อผิดพลาด → ล้างข้อมูลในบัฟเฟอร์การรับข้อมูล → ล้างข้อมูลในบัฟเฟอร์การส่งข้อมูล**

■ สถานะ

รายการ	คำอธิบาย
ข้อมูลสรุป	สถานะจะรวมถึงข้อมูลต่อไปนี้ ตัวแปรสถานะนี้เป็นแบบเขียนอย่างเดียว
รูปแบบ	เมื่อกำหนดบิต: [s:EXT_SIO_STAT**] (** : 00 ถึง 15) เมื่อกำหนดเวิร์ด: [s:EXT_SIO_STAT]

ตัวอย่างนิพจน์

เมื่อกำหนดบิต: if ([s:EXT_SIO_STAT 00] == 1)

เมื่อกำหนดบิต: if (([s:EXT_SIO_STAT] & 0x0001) <> 0)

รายละเอียดของ EXT_SIO_STAT

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

บิต	รายละเอียด
15	0: ไม่มี D-Script/Global D-Script 1: มี D-Script/Global D-Script
14	0: ไม่มี extended script 1: มี Extended script
13	สำรอง
12	
11	
10	
9	
8	
7	
6	
5	
4	0: ปกติ 1: การรับเกิดไทม์เอาต์
3	0: ปกติ 1: การรับเกิดข้อผิดพลาด
2	0: ไม่มีข้อมูลการรับ 1: มีข้อมูลการรับ
1	0: ปกติ 1: การส่งเกิดข้อผิดพลาด
0	0: มีข้อมูลอยู่ในบัฟเฟอร์การส่งข้อมูล 1: ไม่มีข้อมูลอยู่ในบัฟเฟอร์การส่งข้อมูล

หมายเหตุ

- **บิตที่สำรองไว้อาจถูกกำหนดในภายหลัง ดังนั้น ให้กำหนดเฉพาะบิตที่จำเป็นเท่านั้น**
- **ข้อผิดพลาดในการส่งมีอยู่สองชนิด ได้แก่ ข้อผิดพลาดเนื่องจากการส่งเกิดไทม์เอาต์ และข้อผิดพลาดเนื่องจากบัฟเฟอร์การส่งเต็ม เมื่อเกิดข้อผิดพลาดใดขึ้น บิตข้อผิดพลาดในการส่งจะเปิดขึ้น ระยะเวลาไทม์เอาต์ของการส่งคือห้าวินาที**
- **ข้อผิดพลาดในการรับมีอยู่สี่ชนิด ได้แก่ ข้อผิดพลาดพาริตี้, ข้อผิดพลาดโอเวอร์รัน, ข้อผิดพลาดเฟรมมิ่ง และโอเวอร์โฟลว์ เมื่อเกิดข้อผิดพลาดใดขึ้น บิตข้อผิดพลาดในการรับจะเปิดขึ้น**
- **หากตรวจพบข้อผิดพลาดในการส่ง ข้อมูลการส่งจะยังคงอยู่ในบัฟเฟอร์การส่งข้อมูล หากตรวจไม่พบข้อผิดพลาดในการส่ง ข้อมูลการส่งจะถูกส่งออกจากบัฟเฟอร์การส่งข้อมูล**
- **เมื่อใช้พอร์ตอินเตอร์เฟซแบบอนุกรม COM2 ซึ่งเป็น RS-422 จะตรวจไม่พบสัญญาณ CS (CTS) ดังนั้น จึงไม่สามารถตรวจพบได้ว่ามีสายเคเบิลหลุด**

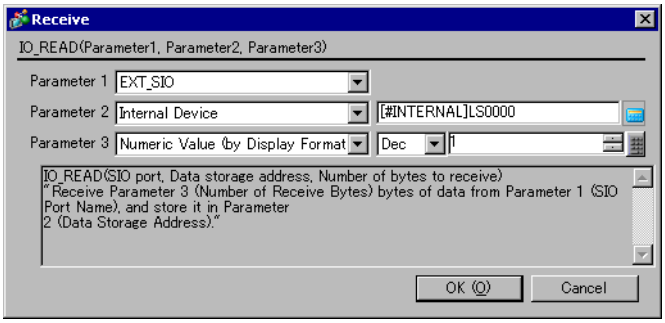
■ จำนวนข้อมูลที่ได้รับ

รายการ	คำอธิบาย
ข้อมูลสรุป	แสดงปริมาณข้อมูล (จำนวนไบต์) ที่ได้รับในคราวนั้น ๆ ขนาดข้อมูลที่ได้รับจะมีคุณสมบัติเป็นแบบอ่านอย่างเดียว
รูปแบบ	[r:EXT_SIO_RECV]

ข้อสำคัญ

- ชื่อป้ายชื่อของจำนวนข้อมูลที่ได้รับ (จำนวนไบต์)
  เมื่อใช้ GP-PRO/PB III V.6.0 และเวอร์ชันก่อนหน้า ชื่อป้ายชื่อซึ่งกำหนดไว้สำหรับขนาดข้อมูลที่ได้รับ คือ [r: EXT_SIO_RCV] อย่างไรก็ตาม คุณไม่จำเป็นต้องแก้ไขคำอธิบาย เพราะจะมีฟังก์ชันเหมือนกันไม่ว่าจะเลือก [r: EXT_SIO_RCV] หรือ [r: EXT_SIO_RECV].

21.4.2 Receive

รายการ	คำอธิบาย
ข้อมูลสรุป	เขียนข้อความคำสั่งต่อไปนี้ขณะอ่านข้อมูลที่ได้รับจาก Extended SIO
รูปแบบ	IO_READ ([p:EXT_SIO], ตำแหน่งจัดเก็บข้อมูล, จำนวนไบต์ที่ได้รับ)  Parameter 1: EXT_SIO Parameter 2: อุปกรณ์ภายใน Parameter 3: ค่าตัวเลข

ตัวอย่างนิพจน์

IO_READ ([p:EXT_SIO], [w:[#INTERNAL]LS0100], 10)

จากตัวอย่างข้างบน จำนวนไบต์ที่ได้รับจะจัดเก็บไว้ใน LS0100 โดยจัดเก็บข้อมูลจำนวน 10 ไบต์ เริ่มตั้งแต่ LS0101 ภาพด้านล่างแสดงข้อมูลที่ได้รับซึ่งจัดเก็บไว้

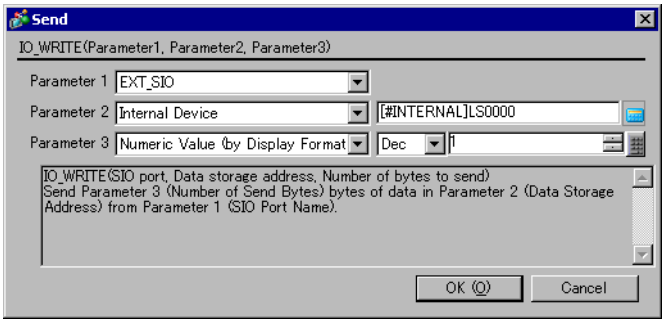
หมายเหตุ

- ในระหว่างการรับข้อมูลสามารถถ่ายโอนไบต์ได้เป็นจำนวนสูงสุด 2,011 ไบต์ ระบบจะเขียนข้อมูลหน่วยละ 1 ไบต์ลงในตำแหน่งเวิร์ดแต่ละตำแหน่ง

	ขนาดข้อมูลที่ได้รับ		
LS0100			... 10 ไบต์
LS0101	00	ไบต์ 1	
LS0102	00	ไบต์ 2	
LS0103	00	ไบต์ 3	
LS0104	00	ไบต์ 4	
LS0105	00	ไบต์ 5	
LS0106	00	ไบต์ 6	
LS0107	00	ไบต์ 7	
LS0108	00	ไบต์ 8	
LS0109	00	ไบต์ 9	
LS0110	00	ไบต์ 10	

วิธีการจัดเก็บข้อมูลที่ได้รับ

21.4.3 Send

รายการ	คำอธิบาย
ข้อมูลสรุป	เขียนข้อความคำสั่งต่อไปนี้เมื่อเขียนข้อมูลลงใน Extended SIO
รูปแบบ	IO_WRITE ([p:EXT_SIO], ตำแหน่งจัดเก็บข้อมูล, จำนวนไบต์ที่ส่ง)  Parameter 1: EXT_SIO Parameter 2: อุปกรณ์ภายใน Parameter 3: ค่าตัวเลข

ตัวอย่างนิพจน์

IO_WRITE ([p:EXT_SIO], [w:[#INTERNAL]LS0100], 10)

จากตัวอย่างข้างบน จะเป็นการส่งข้อมูล 10 ไบต์โดยเริ่มจาก LS0100 ภาพด้านล่างแสดงข้อมูลที่ส่งออกไปซึ่งจัดเก็บไว้

หมายเหตุ

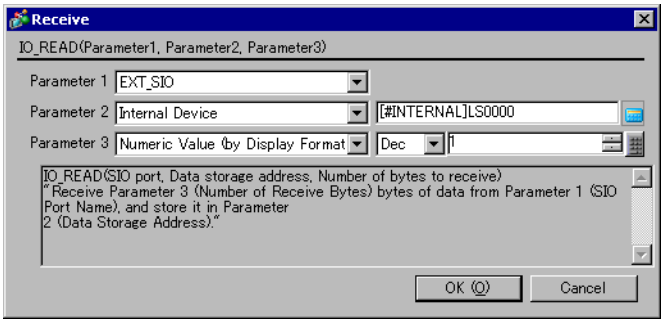
- ในระหว่างการรับข้อมูลสามารถถ่ายโอนไบต์ได้เป็นจำนวนสูงสุด 2,012 ไบต์
- เมื่อใช้อุปกรณ์ LS สำหรับบัฟเฟอร์การส่งข้อมูล ระบบจะเขียนข้อมูลเป็นไบต์เดี่ยวลงในตำแหน่งเวิร์ดแต่ละตำแหน่ง

LS0100	00	ไบต์ 1
LS0101	00	ไบต์ 2
LS0102	00	ไบต์ 3
LS0103	00	ไบต์ 4
LS0104	00	ไบต์ 5
LS0105	00	ไบต์ 6
LS0106	00	ไบต์ 7
LS0107	00	ไบต์ 8
LS0108	00	ไบต์ 9
LS0109	00	ไบต์ 10

วิธีการจัดเก็บข้อมูลที่ส่ง

21.4.4 Extended Receive

รายการ	คำอธิบาย
ข้อมูลสรุป	รับข้อมูลจาก Extended SIO ตามขนาดที่ระบุในขนาดข้อมูลที่ได้รับ (ไบต์) แล้วเก็บไว้ในบัฟเฟอร์ข้อมูล รับข้อมูลจาก Extended SIO ตามจำนวนไบต์ที่ระบุใน Parameter 3 และจัดเก็บไว้ในบัฟเฟอร์ข้อมูลที่ระบุใน Parameter 2 ตัวแปรนี้ใช้ได้เฉพาะใน Extended Script เท่านั้น

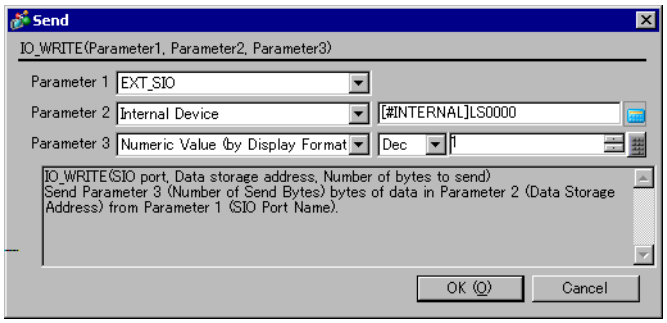
รูปแบบ	IO_READ_EX ([p:EXT_SIO], บัฟเฟอร์ข้อมูล, จำนวนไบต์ที่ได้รับ)  Parameter 1: [p:EXT_SIO] Parameter 2: บัฟเฟอร์ข้อมูล Parameter 3: ค่าตัวเลข, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 3 คือตั้งแต่ 1 ถึง 1,024)

ตัวอย่างนิพจน์

IO_READ_EX ([p:EXT_SIO], databuf 1, 10)

จากตัวอย่างข้างบน เป็นการรับข้อมูลจำนวน 10 ไบต์จาก Extended SIO แล้วจัดเก็บไว้ใน “databuf1”

21.4.5 Extended Send

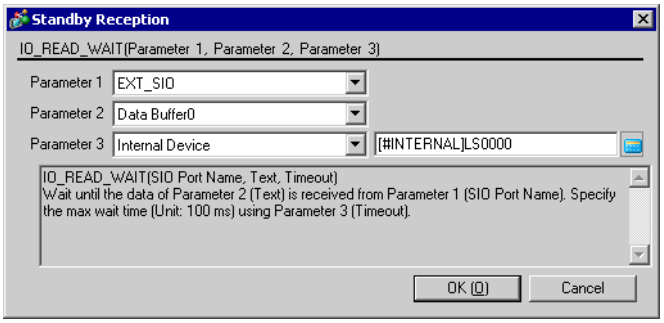
รายการ	คำอธิบาย
ข้อมูลสรุป	ส่งข้อมูลในบัฟเฟอร์ข้อมูลด้วย Extended SIO ตามขนาดจำนวนไบต์ที่ส่ง Extended SIO จะส่งรายละเอียดของบัฟเฟอร์ข้อมูลที่ระบุใน Parameter 2 โดยมีความยาวตามที่ระบุใน Parameter 3 ตัวแปรนี้ใช้ได้เฉพาะใน Extended Script เท่านั้น
รูปแบบ	IO_WRITE_EX ([p:EXT_SIO], บัฟเฟอร์ข้อมูล, จำนวนไบต์ที่ส่ง)  Parameter 1: [p:EXT_SIO] Parameter 2: บัฟเฟอร์ข้อมูล Parameter 3: ค่าตัวเลข, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 3 คือตั้งแต่ 1 ถึง 1,024)

ตัวอย่างนิพจน์

IO_WRITE_EX ([p:EXT_SIO], databuf 0, 10)

จากตัวอย่างข้างบน เป็นการส่งข้อมูลใน “databuf0” 10 ไบต์จาก Extended SIO

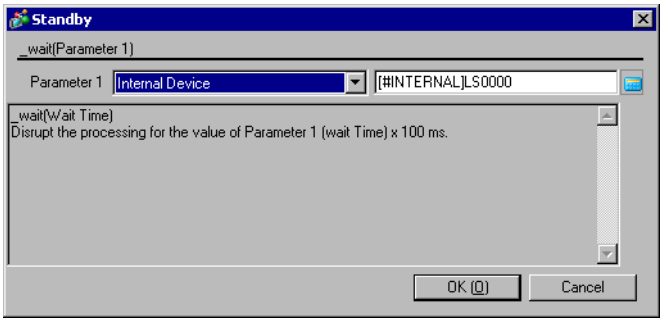
21.4.6 ฟังก์ชัน Standby Reception

รายการ	คำอธิบาย
ข้อมูลสรุป	ระบบอยู่ในโหมดสแตนด์บายการรับจนกว่าจะได้รับข้อความที่ระบุ เมื่อระยะเวลาไทม์เอาต์สิ้นสุดลง ระบบจะตั้งค่าให้บิต 4 (ข้อผิดพลาดเนื่องจากการรับเกิดไทม์เอาต์) ของสถานะ [s: EXT_SIO_STAT] คุณสามารถตั้งระยะเวลาไทม์เอาต์ได้โดยเพิ่มขึ้นครั้งละ 100 มิลลิวินาที ระบบจะอยู่ในโหมดสแตนด์บายการรับจนกว่าจะได้รับสตริงอักขระหรือรหัสอักขระที่ระบุใน Parameter 2 ระยะเวลาไทม์เอาต์จะกำหนดด้วย Parameter 3 ตัวแปรนี้ใช้ได้เฉพาะใน Extended Script เท่านั้น
รูปแบบ	IO_READ_WAIT([p:EXT_SIO], ข้อความ, ไทม์เอาต์)  Parameter 1: [p:EXT_IO] Parameter 2: ค่าตัวเลข, ข้อความ, บัฟเฟอร์ข้อมูล Parameter 3: ค่าตัวเลข, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 3 คือตั้งแต่ 1 ถึง 600)

ข้อสำคัญ

- ข้อมูลที่ได้รับไม่สามารถใช้ได้จนกว่าจะได้รับข้อความที่ระบุ (มิฉะนั้น ระบบจะยกเลิกข้อมูลดังกล่าว)
- สามารถระบุอักขระได้สูงสุด 128 ตัว (ไบต์) โปรดทราบว่าหากระบุสตริงมากกว่าที่กำหนดไว้ จะไม่สามารถสแตนด์บายการรับได้

21.4.7 ฟังก์ชัน Standby

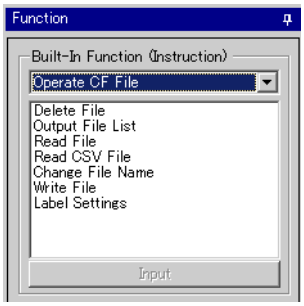
รายการ	คำอธิบาย
ข้อมูลสรุป	ระบบหยุดรอเป็นระยะเวลาตามที่ระบุไว้ คุณสามารถตั้งเวลาได้โดยเพิ่มขึ้นครั้งละ 100 มิลลิวินาที ตัวแปรนี้ใช้ได้เฉพาะใน Extended Script เท่านั้น
รูปแบบ	_wait (เวลารอ)  Parameter 1: อุปกรณ์ภายใน, ตำแหน่งชั่วคราว, ค่าตัวเลข (ช่วงที่ใช้ได้สำหรับ Parameter 1 คือตั้งแต่ 1 ถึง 600)

ตัวอย่างนิพจน์

_wait (10)

จากตัวอย่างข้างบน ระบบจะรอเป็นเวลาหนึ่งวินาที

21.5 การใช้งานไฟล์ในการ์ด CF

Operate CF File	ข้อมูลสรุปของฟังก์ชัน
	Label Settings ☞ "21.5.1 Label Settings" (หน้า 21-35) **ตั้งค่าจากจำนวนไฟล์ตามที่แสดง, จำนวนไบต์ที่อ่าน และสถานะข้อผิดพลาดของการ์ด CF**
	Write File ☞ "21.5.2 Write to File" (หน้า 21-44) **เขียนข้อมูลตามจำนวนไบต์ที่ระบุจากตำแหน่งต้นทางไปยังไฟล์ที่ระบุ**
	Change File Name ☞ "21.5.3 Change File Name" (หน้า 21-49) **แก้ไขชื่อไฟล์**
	Read CSV File ☞ "21.5.4 Read CSV File" (หน้า 21-51) **อ่านข้อมูลในเซลล์จากไฟล์ CSV และเขียนลงในตำแหน่งเวิร์ด**
	Read File ☞ "21.5.5 Read File" (หน้า 21-54) **อ่านข้อมูลที่อยู่ต่อจากออฟเซ็ตที่ระบุไว้ตามจำนวนไบต์ที่ระบุ และเขียนข้อมูลที่อ่านได้ลงในตำแหน่งปลายทาง**
	Output File List ☞ "21.5.6 Output File List" (หน้า 21-56) **เขียนรายการไฟล์ที่มีอยู่ในโฟลเดอร์ที่ระบุลงในอุปกรณ์ภายใน**
	Delete File ☞ "21.5.7 Delete File" (หน้า 21-57) **ลบไฟล์ทิ้ง**

21.5.1 Label Settings

สถานะของการ์ด CF จะมีสถานะต่าง ๆ ต่อไปนี้

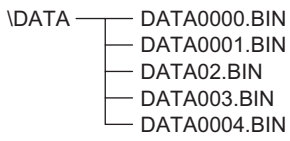
ชื่อสถานะ	ชื่อป้ายชื่อ	คำอธิบาย
จำนวนไฟล์ที่แสดง	[s:CF_FILELIST_NUM]	จัดเก็บจำนวนไฟล์ตามที่แสดงจริง เมื่อมีการเรียกใช้ฟังก์ชันส่งรายการไฟล์ "_CF_dir ( )"
จำนวนไบต์ที่อ่าน	[s:CF_READ_NUM]	จัดเก็บจำนวนไบต์ที่สามารถอ่านได้เมื่อมีการเรียกใช้ฟังก์ชันอ่านไฟล์ "_CF_read ( )"
สถานะข้อผิดพลาดของการ์ด CF	[s:CF_ERR_STAT]	จัดเก็บสถานะข้อผิดพลาดที่สร้างขึ้นเมื่อมีการเข้าใช้ข้อมูลในการ์ด CF

■ จำนวนไฟล์ที่แสดง

เมื่อเรียกใช้ฟังก์ชันส่งรายการไฟล์ "_CF_dir ( )" จำนวนรายการไฟล์ที่ถูกเขียนจริงในพื้นที่ LS จะถูกจัดเก็บไว้ใน "จำนวนไฟล์ที่แสดง [s:CF_FILELIST_NUM]"

ตัวอย่างการใช้

```
_CF_dir ("\DATA\*.*", [w:[#INTERNAL]LS0100], 10, 0)
[w:LS0200] = [s:CF_FILELIST_NUM]
```



เมื่อขอรับรายการไฟล์ที่ประกอบด้วยไฟล์จำนวน 10 ไฟล์ แต่โฟลเดอร์ที่ระบุไว้มีไฟล์อยู่เพียง 5 ไฟล์
ระบบจะจัดเก็บค่า “5” ไว้ใน [s:CF_FILELIST_NUM]

ข้อสำคัญ

- เมื่อไม่มีไฟล์สำหรับเขียน ระบบจะเขียนจำนวนไฟล์ทั้งหมดที่มีอยู่ในโฟลเดอร์ที่ระบุลงใน [s:CF_FILELIST_NUM]

■ จำนวนไบต์ที่อ่าน

เมื่อเรียกใช้ฟังก์ชันอ่านไฟล์ “_CF_read ( )” จำนวนไบต์ที่อ่านจริงจะถูกจัดเก็บไว้ใน “จำนวนไบต์ที่อ่าน [s:CF_READ_NUM]”

ตัวอย่างการใช้

```
_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 16, 16)
[w:[#INTERNAL]LS0200] = [s:CF_READ_NUM]
```

เมื่อมีความพยายามขออ่านข้อมูลจำนวน 16 ไบต์ แต่อ่านได้สำเร็จเพียง 12 ไบต์
ระบบจะจัดเก็บค่า “12” ไว้ใน [s:CF_READ_NUM]

■ สถานะข้อผิดพลาดของการ์ด CF

จัดเก็บสถานะข้อผิดพลาดที่สร้างขึ้นเมื่อมีการเข้าใช้ข้อมูลในการ์ด CF

ตำแหน่งบิต	ชื่อข้อผิดพลาด	คำอธิบาย
15	สำรอง	สำรอง
14		
13		
12		
11		
10		
9		
8		
7		
6	ข้อผิดพลาดในการเปลี่ยนชื่อไฟล์	• การ์ด CF ถูกดึงออกขณะทำงานอยู่ • ไม่มีไฟล์ที่ระบุ • มีความพยายามที่จะเปลี่ยนชื่อไฟล์ที่เป็นไฟล์แบบอ่านอย่างเดียว
5	ข้อผิดพลาดในการลบไฟล์	• การ์ด CF ถูกดึงออกขณะทำงานอยู่ • ไม่มีไฟล์ที่ระบุ • มีความพยายามที่จะลบไฟล์ที่เป็นไฟล์แบบอ่านอย่างเดียว
4	ข้อผิดพลาดในการเขียนไฟล์	• การ์ด CF ถูกดึงออกขณะทำงานอยู่ • ไม่มีพื้นที่ว่างเหลือในการ์ด CF • มีความพยายามที่จะเขียนข้อมูลลงในไฟล์ที่เป็นไฟล์แบบอ่านอย่างเดียว • ไม่มีไฟล์ตามที่กำหนดให้ “เขียนทับ”
3	ข้อผิดพลาดในการอ่านไฟล์	• การ์ด CF ถูกดึงออกขณะทำงานอยู่ • ไม่มีไฟล์ที่ระบุ
2	ข้อผิดพลาดในรายการไฟล์	• การ์ด CF ถูกดึงออกขณะทำงานอยู่ • ไม่มีโฟลเดอร์ที่ระบุ
1	ข้อผิดพลาดของการ์ด CF	• การ์ด CF ใช้ไม่ได้ • สื่อบันทึกข้อมูลที่เสียบไว้ไม่ใช่การ์ด CF
0	ไม่มีการ์ด CF	• ไม่ได้เสียบการ์ด CF • ฝาครอบเปิดอยู่

- แม้จะเกิดข้อผิดพลาดในการ์ด CF แต่การทำงานยังคงดำเนินต่อไป ทุกครั้งที่คุณใช้ฟังก์ชันการใช้งานไฟล์ในการ์ด CF ต้องเขียนสคริปต์สำหรับตรวจสอบข้อผิดพลาดด้วย
  ตัวอย่าง)

```
_CF_dir ("\DATA\*.*", [w:[#INTERNAL]LS0100], 2, 1) ส่งข้อมูลรายการไฟล์
if ([s:CF_ERR_STAT02] <> 0)          // ตรวจสอบสถานะข้อผิดพลาด
{
    set ([b:[#INTERNAL]LS 005000]) // กำหนดตำแหน่งบิตสำหรับแสดงข้อผิดพลาด
}
endif
```

รายละเอียดสถานะข้อผิดพลาดของการ์ด CF - พื้นที่จัดเก็บข้อมูล

หากเกิดข้อผิดพลาดขึ้น ระบบจะตั้งค่าบิตตามที่เหมาะสม คุณสามารถตรวจสอบสาเหตุของข้อผิดพลาดนั้นได้โดยดูที่รายละเอียดสถานะ รายละเอียดสถานะสำหรับแต่ละฟังก์ชันจะจัดเก็บไว้ในพื้นที่ LS9132 จนถึง LS9137 ของพื้นที่เสริมของระบบ พื้นที่เหล่านี้เป็นแบบอ่านอย่างเดียว

	พื้นที่ LS	
LS0000		
:		
LS9132		สถานะรายการไฟล์ในการ์ด CF
LS9133		สถานะการอ่านข้อมูลในการ์ด CF
LS9134		สถานะการเขียนข้อมูลในการ์ด CF
LS9135		สถานะการลบข้อมูลในการ์ด CF
LS9136		สถานะการเปลี่ยนชื่อในการ์ด CF
LS9137		สถานะการอ่านข้อมูล CSV
:		
LS9999		

รายการข้อผิดพลาดสำหรับแต่ละฟังก์ชัน

ชื่อฟังก์ชันตัวแก้ไข		สถานะข้อผิดพลาด	สาเหตุ
_CF_dir ( )	LS9132	0010h	ข้อมูล D-Script ไม่ถูกต้อง (เกิดข้อผิดพลาดในการค้นชื่อโฟลเดอร์ที่ระบุด้วยสตริงแบบตายตัว)
		0012h	ชื่อไฟล์ (ชื่อพาธ) มีข้อผิดพลาด
		0018h	ช่วงการเขียนของพื้นที่ LS เกิดข้อผิดพลาด
		0020h	ไม่มีการ์ด CF
		0021h	การ์ด CF ใช้ไม่ได้
		0100h	การเปิดไดเร็กทอรีเกิดข้อผิดพลาด
_CF_read ( )	LS9133	0010h	ข้อมูล D-Script ไม่ถูกต้อง (เกิดข้อผิดพลาดในการค้นชื่อโฟลเดอร์/ชื่อไฟล์ที่ระบุด้วยสตริงแบบตายตัว)
		0011h	ช่วงการอ่านของพื้นที่ LS เกิดข้อผิดพลาด
		0010002h	ชื่อไฟล์ (ชื่อพาธ) มีข้อผิดพลาด
		0018h	ช่วงการเขียนของพื้นที่ LS เกิดข้อผิดพลาด
		0020h	ไม่มีการ์ด CF
		0021h	การ์ด CF ใช้ไม่ได้
		0101h	การหาไฟล์เกิดข้อผิดพลาด (ข้อผิดพลาดของออฟเซ็ต)
		0102h	จำนวนไบต์ที่อ่านมีข้อผิดพลาด
		0110h	การสร้าง (เปิด) ไฟล์เกิดข้อผิดพลาด

ชื่อฟังก์ชันตัวแก้ไข		สถานะข้อผิดพลาด	สาเหตุ
_CF_write ( )	LS9134	0010h	ข้อมูล D-Script ไม่ถูกต้อง (เกิดข้อผิดพลาดในการค้นชื่อโฟลเดอร์/ชื่อไฟล์ที่ระบุด้วยสตริงแบบตายตัว)
		0011h	ช่วงการอ่านของพื้นที่ LS เกิดข้อผิดพลาด
		0010002h	ชื่อไฟล์ (ชื่อพาธ) มีข้อผิดพลาด
		0020h	ไม่มีการ์ด CF
		0021h	การ์ด CF ใช้ไม่ได้
		0101h	การหาไฟล์เกิดข้อผิดพลาด (ข้อผิดพลาดของออฟเซ็ต)
		010h	การสร้างไฟล์เกิดข้อผิดพลาด
		0108h	โหมดการเขียนเกิดข้อผิดพลาด
		0110h	การสร้าง (เปิด) ไฟล์เกิดข้อผิดพลาด
		0111h	การเขียนไฟล์เกิดข้อผิดพลาด (เช่น การ์ด CF มีพื้นที่ว่างไม่เพียงพอ)
_CF_delete ( )	LS9135	0010h	ข้อมูล D-Script ไม่ถูกต้อง (เกิดข้อผิดพลาดในการค้นชื่อโฟลเดอร์/ชื่อไฟล์ที่ระบุด้วยสตริงแบบตายตัว)
		0011h	ช่วงการอ่านของพื้นที่ LS เกิดข้อผิดพลาด
		0010002h	ชื่อไฟล์ (ชื่อพาธ) มีข้อผิดพลาด
		0020h	ไม่มีการ์ด CF
		0021h	การ์ด CF ใช้ไม่ได้
		0112h	การลบไฟล์เกิดข้อผิดพลาด (เช่น ไม่มีไฟล์ที่ระบุ ไฟล์ที่ระบุเป็นแบบอ่านอย่างเดียว)
_CF_rename ( )	LS9136	0010h	ข้อมูล D-Script ไม่ถูกต้อง (เกิดข้อผิดพลาดในการค้นชื่อโฟลเดอร์/ชื่อไฟล์ที่ระบุด้วยสตริงแบบตายตัว)
		0011h	ช่วงการอ่านของพื้นที่ LS เกิดข้อผิดพลาด
		0010002h	ชื่อไฟล์ (ชื่อพาธ) มีข้อผิดพลาด
		0020h	ไม่มีการ์ด CF
		0021h	การ์ด CF ใช้ไม่ได้
		0114h	การเปลี่ยนชื่อไฟล์เกิดข้อผิดพลาด (เช่น ไม่มีไฟล์ที่ระบุ ไฟล์ที่ระบุเป็นแบบอ่านอย่างเดียว มีชื่อไฟล์นั้นแล้ว)

ชื่อฟังก์ชันตัวแก้ไข		สถานะข้อผิดพลาด	สาเหตุ
_CF_read_csv()	LS9137	0001h	พารามิเตอร์เกิดข้อผิดพลาด
		000h	ข้อผิดพลาดของการ์ด CF (ไม่มีการ์ด CF, การเปิดไฟล์เกิดข้อผิดพลาด, การอ่านไฟล์เกิดข้อผิดพลาด)
		0003h	การเขียนเกิดข้อผิดพลาด

โหมดการจัดเก็บข้อมูล

เมื่อมีการอ่าน/เขียนข้อมูลจาก/ไปยังตำแหน่งอุปกรณ์ ขณะเรียกใช้ฟังก์ชันอ่านไฟล์/เขียนไฟล์ คุณสามารถระบุลำดับการจัดเก็บของข้อมูลที่เขียน (อ่าน) ได้

การตั้งค่าโหมดการจัดเก็บข้อมูลในพื้นที่ LS9130 สามารถเปลี่ยนลำดับการจัดเก็บได้ โดยสามารถเลือกโหมดได้จากตัวเลือกได้แก่ โหมด 0, 1, 2 และ 3

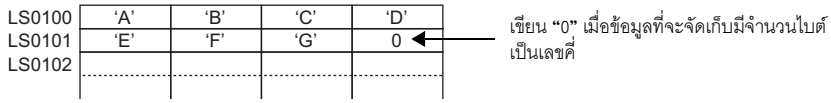
◆ **โหมด** 0

ตัวอย่าง: การใช้ฟังก์ชันอ่านไฟล์เขียนสตริง “ABCDEFG” ลงในตำแหน่งอุปกรณ์

```
[w:[#INTERNAL]LS9130] = 0
_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 0, 7)
```

- เมื่อตำแหน่งอุปกรณ์ยาว 16 บิต

- เมื่อตำแหน่งอุปกรณ์ยาว 32 บิต



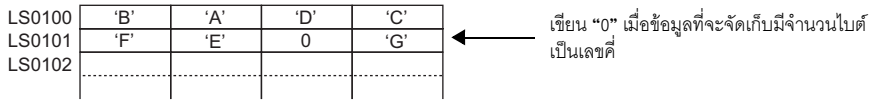
◆ **โหมด** 1

ตัวอย่าง: การใช้ฟังก์ชันอ่านไฟล์เขียนสตริง “ABCDEFG” ลงในตำแหน่งอุปกรณ์

```
[w:[#INTERNAL]LS9130] = 1
_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 0, 7)
```

- เมื่อตำแหน่งอุปกรณ์ยาว 16 บิต

- เมื่อตำแหน่งอุปกรณ์ยาว 32 บิต

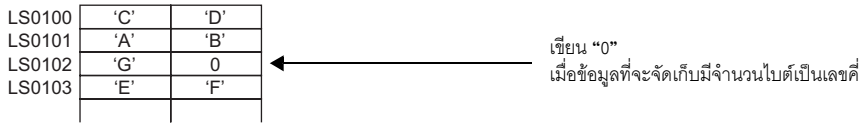


◆ **โหมด 2**

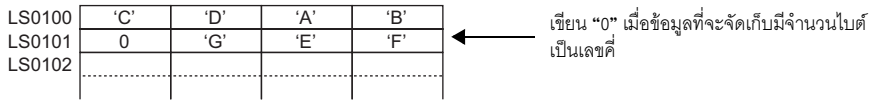
ตัวอย่าง: การใช้ฟังก์ชันอ่านไฟล์เขียนสตริง “ABCDEFG” ลงในตำแหน่งอุปกรณ์

```
[w:[#INTERNAL]LS9130] = 2
_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 0, 7)
```

- **เมื่อตำแหน่งอุปกรณ์ยาว 16 บิต**



- **เมื่อตำแหน่งอุปกรณ์ยาว 32 บิต**

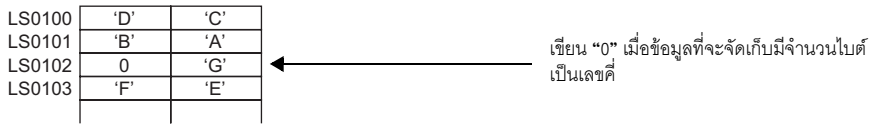


◆ **โหมด 3**

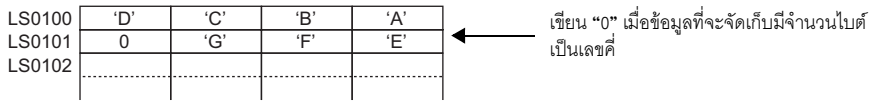
ตัวอย่าง: การใช้ฟังก์ชันอ่านไฟล์เขียนสตริง “ABCDEFG” ลงในตำแหน่งอุปกรณ์

```
[w:[#INTERNAL]LS9130] = 3
_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 0, 7)
```

- **เมื่อตำแหน่งอุปกรณ์ยาว 16 บิต**



- **เมื่อตำแหน่งอุปกรณ์ยาว 32 บิต**



ข้อสำคัญ

- โหมดการจัดเก็บข้อมูลไม่เหมือนกับโหมดข้อมูลสตริงในการตั้งค่าระบบ ความสัมพันธ์ระหว่างโหมดการจัดเก็บข้อมูลกับโหมดข้อมูลสตริงจะแสดงอยู่ในตารางด้านล่าง

ลำดับการจัดเก็บของอุปกรณ์ข้อมูล	ลำดับการจัดเก็บไบต์ในเวิร์ดแบบ LH หรือ HL	ลำดับการจัดเก็บในดับเบิลเวิร์ดแบบ LH หรือ HL	โหมดการจัดเก็บข้อมูล D-Script	โหมดข้อมูลตัวอักษร
จัดเก็บจากข้อมูลเริ่มต้น	ลำดับ HL	ลำดับ HL	0	1
	ลำดับ LH		1	2
	ลำดับ HL	ลำดับ LH	2	5
	ลำดับ LH		3	4
จัดเก็บจากข้อมูลท้ายสุด	ลำดับ HL	ลำดับ HL	-	3
	ลำดับ LH		-	7
	ลำดับ HL	ลำดับ LH	-	8
	ลำดับ LH		-	6

- การเขียนข้อมูลในการ์ด CF มีข้อจำกัดเกี่ยวกับจำนวนครั้งที่สามารถเขียนซ้ำได้ โปรดสำรองข้อมูลทั้งหมดในการ์ด CF เก็บไว้ในสื่อเก็บข้อมูลอื่นอยู่เป็นประจำ สามารถเขียนทับข้อมูลในรูปแบบ DOS ขนาด 500KB ได้ 100,000 ครั้ง
- หากเกิดข้อผิดพลาดขึ้นระหว่างประมวลผลการ์ด CF ระบบจะเขียนข้อผิดพลาดลงในสถานะข้อผิดพลาดของการ์ด CF [s:CF_ERR_STAT] โปรดดูรายละเอียดเพิ่มเติมได้ที่ ☞“ ■ สถานะข้อผิดพลาดของการ์ด CF” (หน้า 21-37)
- ชื่อโฟลเดอร์หรือชื่อไฟล์ไม่สามารถใช้สัญลักษณ์และอักขระต่อไปนี้ หากใช้สัญลักษณ์และอักขระเหล่านี้ในชื่อโฟลเดอร์หรือชื่อไฟล์จะทำให้เกิดข้อผิดพลาด

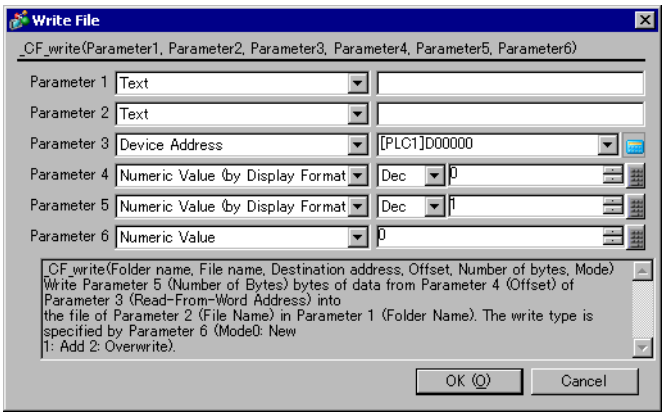
:	,	=	+	/	"	[
]	|	<	>	(พื้นที่ว่าง)	?	

- หากต้องการระบุโฟลเดอร์ราก (ไดเร็กทอรี) ให้ระบุชื่อโฟลเดอร์เป็น “ ” (สตริงเปล่า)

21.5.2 Write to File

รายการ	คำอธิบาย
ข้อมูลสรุป	**เขียนข้อมูลตามจำนวนไบต์ที่ระบุจากตำแหน่งต้นทางไปยังไฟล์ที่ระบุ โดยสามารถเลือกโหมดใดโหมดหนึ่งในสามโหมดนี้ได้แก่ “New”, “Add” หรือ “Overwrite” โปรดดูรายละเอียดเพิ่มเติมเกี่ยวกับลำดับการจัดเก็บข้อมูลได้ในหัวข้อ “โหมดการจัดเก็บข้อมูล” ข้างล่าง**

รูปแบบ	_CF_write (ชื่อโฟลเดอร์, ชื่อไฟล์, ตำแหน่งที่ถูกอ่าน, ออฟเซ็ต, จำนวนไบต์, โหมด)



Parameter 1

ชื่อโฟลเดอร์: สตริงแบบตายตัว (ความยาวสูงสุด: อักขระแบบไบต์เดี่ยว 32 อักขระ)

Parameter 2

ชื่อไฟล์: สตริงแบบตายตัว, อุปกรณ์ภายใน (ความยาวสูงสุด: อักขระแบบไบต์เดี่ยว 32 อักขระ), อุปกรณ์ภายใน + ตำแหน่งชั่วคราว

Parameter 3

ตำแหน่งที่ถูกอ่าน: ตำแหน่งอุปกรณ์, ตำแหน่งอุปกรณ์ + ตำแหน่งชั่วคราว

Parameter 4

ออฟเซ็ต: ค่าตัวเลข, ตำแหน่งอุปกรณ์, ตำแหน่งชั่วคราว (จำนวนสูงสุดที่สามารถระบุได้: 65,535 สำหรับความยาว 16 บิต, 4,294,967,295 สำหรับความยาว 32 บิต)

Parameter 5

จำนวนไบต์: ค่าตัวเลข, ตำแหน่งอุปกรณ์, ตำแหน่งชั่วคราว (ความยาวสูงสุด: 1,280)

Parameter 6

โหมด: ค่าตัวเลข, ตำแหน่งอุปกรณ์, ตำแหน่งชั่วคราว (ค่าที่ใช้ได้: 0, 1, 2)

ข้อมูลทั่วไปเกี่ยวกับรูปแบบการจัดเก็บข้อมูล

Mode	ชื่อ	คำอธิบาย
0	New	สร้างไฟล์ใหม่ หากมีไฟล์ที่มีชื่อเหมือนกันอยู่ ระบบจะลบไฟล์นั้นทิ้ง
1	Add	เพิ่มข้อมูลลงในไฟล์ที่ระบุ หากไม่มีไฟล์ที่ระบุ ระบบจะสร้างไฟล์ใหม่ขึ้นมา
2	Overwrite	เขียนทับบางส่วนของไฟล์นั้น หากระบุออฟเซ็ตใหญ่กว่าขนาดไฟล์ ระบบจะเติม 0 ลงในพื้นที่ที่เกินมาและเขียนข้อมูลต่อจากพื้นที่นั้น หากระบุออฟเซ็ตไว้ตอนท้ายของข้อมูลไฟล์ จะเหมือนกับการเพิ่มข้อมูลลงในไฟล์ หากไม่มีไฟล์อยู่ จะเกิดข้อผิดพลาดขึ้น สำหรับข้อมูลเพิ่มเติมเกี่ยวกับข้อผิดพลาดนี้ โปรดดูที่ “ ■ สถานะข้อผิดพลาดของการ์ด CF” (หน้า 21-37)

ตัวอย่างนิพจน์

```
[w:[#INTERNAL]LS0200] = 0       //Offset ("0" when the mode is "New")
[w:[#INTERNAL]LS0202] = 100     // Number of Bytes (100 bytes)
[w:[#INTERNAL]LS0204] = 0       //Mode (New)
_CF_write ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100],
[w:[#INTERNAL]LS0200],
[w:[#INTERNAL]LS0202], [w:[#INTERNAL]:LS0204])[#INTERNAL]LS0202],
[w:[#INTERNAL]:LS0204])
```

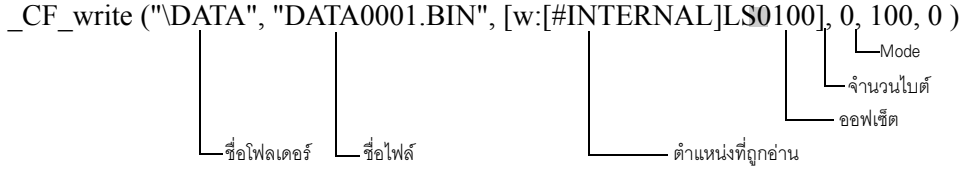
ตัวอย่างข้างบนเป็นการสร้างไฟล์ใหม่ชื่อ DATA0001.BIN ไว้ในโฟลเดอร์ \DATA และจัดเก็บข้อมูลจำนวน 100 ไบต์ที่อ่านจากพื้นที่ LS0100
เมื่อระบุอุปกรณ์ภายในให้กับออฟเซ็ต จำนวนไบต์ หรือโหมด คุณสามารถกำหนดออฟเซ็ต จำนวนไบต์ หรือโหมดโดยทางอ้อมได้

ข้อสำคัญ

- การตั้งค่าออฟเซ็ตจะมีผลเฉพาะในโหมด “Overwrite” เท่านั้น ส่วนในโหมด “New” และโหมด “Add” จะตั้งค่าออฟเซ็ตไม่ได้ ให้ตั้งค่าออฟเซ็ตเป็น “0” ในโหมดอื่นที่ไม่ใช่โหมด “Overwrite”
- เมื่อระบุโหมด “New” และมีไฟล์ที่มีชื่อเหมือนกันอยู่แล้ว ไฟล์นั้นจะถูกเขียนทับ
- เมื่อระบุพื้นที่ LS ไว้สำหรับ “ชื่อไฟล์” แล้ว “ตำแหน่งที่ถูกอ่าน” จะไม่นับเป็นตำแหน่งของ D-Script
- เมื่อระบุอุปกรณ์ PLC ไว้สำหรับ “ตำแหน่งที่ถูกอ่าน” เมื่อมีการเรียกใช้ฟังก์ชัน ระบบจะอ่านข้อมูลจาก PLC เพียงครั้งเดียวเท่านั้น หากเกิดข้อผิดพลาดระหว่างการอ่านข้อมูล ระบบจะกำหนดข้อผิดพลาดไว้ในสถานะข้อผิดพลาดของการ์ด CF [s:CF_ERR_STAT] และจะล้างข้อผิดพลาดออกเมื่ออ่านข้อมูลเสร็จสมบูรณ์
- ระบบจะแบ่งข้อมูลออกเป็นรายการ และอ่านจากต้นฉบับ แม้ว่าจะขึ้นอยู่กับจำนวนไบต์ที่จะอ่านด้วยก็ตาม ดังนั้น ถึงแม้จะเกิดข้อผิดพลาดในการสื่อสารขึ้นระหว่างที่อ่านข้อมูล แต่อาจมีข้อมูลบางส่วนถูกเขียนลงในไฟล์ที่ระบุแล้ว
- หากต้องการระบุพาธแบบครบถ้วนเป็นชื่อไฟล์ ให้ระบุ “*” (เครื่องหมายดอกจัน) เป็นชื่อโฟลเดอร์ ตัวอย่าง: _CF_read ("*", "\DATA\DATA0001.BIN" [w:[#INTERNAL]LS0100], 0, 10)

ตัวอย่างนิพจน์ของรูปแบบการจัดเก็บข้อมูล

◆ **เมื่อระบุโหมด** “New”



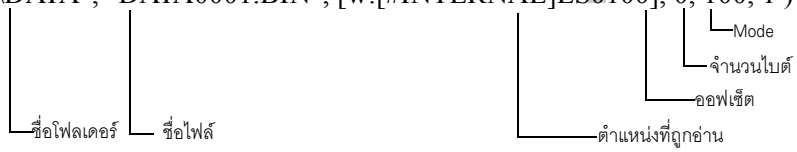
เมื่อมีการเรียกใช้ข้อความคำสั่งข้างบน ระบบจะอ่านข้อมูลจำนวน 100 ไบต์จากพื้นที่ LS0100 และพื้นที่ถัดไป แล้วเขียนลงในไฟล์ DATA0001.BIN ที่สร้างขึ้นใหม่ในโฟลเดอร์ \DATA

ข้อสำคัญ

- สามารถใช้ชื่อไฟล์ได้เฉพาะรูปแบบ 8.3 เท่านั้น (สูงสุด 12 อักขระ โดยแบ่งเป็นชื่อไฟล์ เครื่องหมายมหัพภาค 8 อักขระ และนามสกุลอีก 3 อักขระ) ชื่อไฟล์ที่ยาวกว่ารูปแบบนี้ ไม่สามารถใช้ได้

◆ **เมื่อระบุโหมด** “Add”

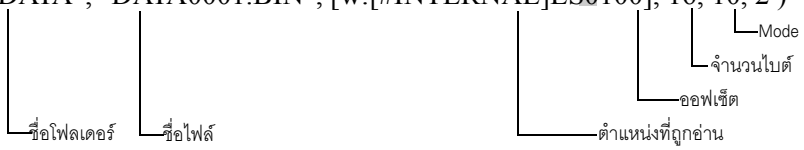
_CF_write ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 0, 100, 1)



หากมีไฟล์ที่ระบุอยู่แล้ว (เช่น DATA0001.BIN) และเรียกใช้ข้อความคำสั่งข้างบน ระบบจะอ่านข้อมูลจำนวน 100 ไบต์จากพื้นที่ LS0100 และพื้นที่ถัดไป แล้วเพิ่มลงในไฟล์ DATA0001.BIN ในโฟลเดอร์ \DATA

◆ **เมื่อระบุโหมด** “Overwrite” (1)

_CF_write ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 16, 10, 2)



หากมีไฟล์ที่ระบุอยู่แล้ว (เช่น DATA0001.BIN) และเรียกใช้ข้อความคำสั่งข้างบน ระบบจะอ่านข้อมูลจำนวน 10 ไบต์ที่จัดเก็บไว้ในพื้นที่ LS0100 และพื้นที่ถัดไป แล้วเขียนทับข้อมูลจำนวน 10 ไบต์ที่จัดเก็บอยู่ในไบต์ที่ 17 และไบต์ถัดไปต่อจากออฟเซ็ตในไฟล์ DATA0001.BIN ในโฟลเดอร์ \DATA

◆ **เมื่อระบุโหมด** “Overwrite” (2)

(ไฟล์ที่จะเขียนทับมีน้อยกว่ายอดรวมของค่าออฟเซ็ตและจำนวนไบต์)

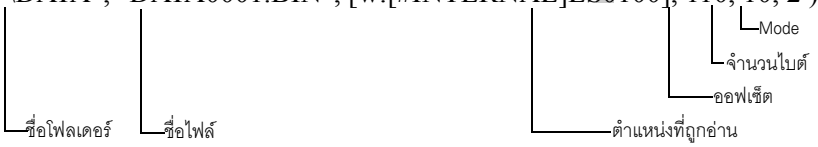
_CF_write ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 96, 10, 2)

มีไฟล์ที่ระบุอยู่แล้ว (เช่น DATA0001.BIN) และขนาดไฟล์เท่ากับ 100 ไบต์ เมื่อตั้งค่าออฟเซ็ตเป็น 96 ไบต์ และตั้งค่าจำนวนไบต์สำหรับการเขียนทับไว้ 10 ไบต์ ระบบจะอ่านข้อมูลจำนวน 10 ไบต์ที่จัดเก็บอยู่ในพื้นที่ LS0100 และพื้นที่ถัดไป จากนั้น ข้อมูลที่อ่านได้ 4 ไบต์แรกจะเขียนทับข้อมูล 4 ไบต์ที่จัดเก็บไว้ในไบต์ที่ 97 และไบต์ถัดไปในไฟล์ ส่วนข้อมูลที่เหลืออีก 6 ไบต์จะถูกเพิ่มลงในตอนท้ายของข้อมูลในไฟล์ ไฟล์ผลลัพธ์ที่ได้จะมีข้อมูลเท่ากับ 106 ไบต์

◆ **เมื่อระบุโหมด** “Overwrite” (3)

(ไฟล์ที่จะถูกเขียนทับเล็กกว่าค่าออฟเซ็ต)

_CF_write ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 110, 10, 2)



มีไฟล์ที่ระบุอยู่แล้ว (เช่น DATA0001.BIN) และขนาดไฟล์เท่ากับ 100 ไบต์ เมื่อตั้งค่าออฟเซ็ตเป็น 110 ไบต์ และตั้งค่าจำนวนไบต์สำหรับการเขียนทับไว้ 10 ไบต์ ระบบจะเติม 0 ลงในพื้นที่ระหว่างไบต์ที่ 101 และไบต์ที่ 110 และเขียนข้อมูลจำนวน 10 ไบต์ที่อ่านจากพื้นที่ LS0100 และพื้นที่ถัดไปลงในไบต์ที่ 111 และไบต์ถัดไป ไฟล์ผลลัพธ์ที่ได้จะมีข้อมูลเท่ากับ 120 ไบต์

ข้อสำคัญ

- พารามิเตอร์แรก **“ชื่อโฟลเดอร์”** และพารามิเตอร์ที่สอง **“ชื่อไฟล์”** สามารถใช้อักขระแบบไบต์เดี่ยวได้สูงสุด 32 ตัว
- สามารถระบุอุปกรณ์ภายในเป็น **“ชื่อไฟล์”** ของพารามิเตอร์ที่สองได้ การระบุอุปกรณ์ภายในทำให้สามารถอ้างตำแหน่งของชื่อไฟล์โดยอ้อมได้ โดยสามารถระบุชื่อไฟล์ด้วยอักขระแบบไบต์เดี่ยวได้สูงสุด 32 ตัว
  ตัวอย่าง: _CF_write ("\DATA" [w:LS0100], [w:[#INTERNAL]LS0200], 0, 100, 0)
  การจัดเก็บชื่อไฟล์ในพื้นที่ LS0100 ช่วยอ้างตำแหน่งชื่อไฟล์โดยอ้อมได้ ในตัวอย่างนี้ ชื่อไฟล์จะถูกจัดเก็บไว้ในพื้นที่ LS0100 จนถึง LS0106 ดังต่อไปนี้

16 บิต

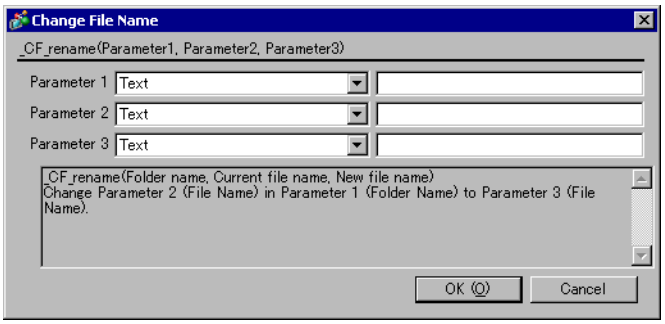
LS0100	‘D’	‘A’
LS0101	‘T’	‘A’
LS0102	‘0’	‘0’
LS0103	‘0’	‘1’
LS0104	‘.’	‘B’
LS0105	‘I’	‘N’
LS0106	‘\0’	‘\0’
	:	

ตอนท้ายของชื่อไฟล์ต้องเป็นอักขระค่าศูนย์ อุปกรณ์แสดงผลรับรู้ข้อมูลที่อยู่ก่อนหน้าอักขระค่าศูนย์ว่าเป็นชื่อไฟล์

จากตัวอย่างข้างบน ระบบจะอ่านข้อมูลจำนวน 100 ไบต์จากพื้นที่ LS0200 และพื้นที่ถัดไป และสร้างไฟล์ใหม่ชื่อ **“**\DATA\DATA0001.BIN**”** เพื่อใช้จัดเก็บข้อมูล

- ชื่อไฟล์สามารถใช้ได้เฉพาะ **“รูปแบบ 8.3”** เท่านั้น (อักขระสูงสุด 12 อักขระ โดยเป็นชื่อไฟล์, เครื่องหมายมหัพภาค 8 อักขระ และนามสกุลอีก 3 อักขระ) ไม่สามารถใช้ชื่อไฟล์ยาวกว่านี้ได้

21.5.3 Change File Name

รายการ	คำอธิบาย
ข้อมูลสรุป	แก้ไขชื่อไฟล์ Parameter 1 จะกำหนดโฟลเดอร์ข้อมูลการ์ด CF Parameter 2 จะกำหนดชื่อไฟล์ต้นฉบับ Parameter 3 จะกำหนดชื่อใหม่
รูปแบบ	_CF_rename (ชื่อโฟลเดอร์, ชื่อไฟล์, ชื่อไฟล์ใหม่) คุณสามารถกำหนดชื่อไฟล์โดยอ้อมด้วยตำแหน่ง LS ได้ด้วย  Parameter 1 ชื่อโฟลเดอร์: ข้อความที่กำหนดไว้ตายตัว Parameter 2 ชื่อไฟล์: ข้อความที่กำหนดไว้ตายตัว, อุปกรณ์ภายใน, อุปกรณ์ภายใน + ตำแหน่งชั่วคราว Parameter 3 ชื่อไฟล์: ข้อความที่กำหนดไว้ตายตัว, อุปกรณ์ภายใน, อุปกรณ์ภายใน + ตำแหน่งชั่วคราว

ตัวอย่างนิพจน์

_CF_rename ("\DATA","DATA0001.BIN","DATA1234.BIN")

ตัวอย่างข้างบนเป็นการเปลี่ยนชื่อไฟล์จาก “\DATA\DATA0001.BIN” เป็น “\DATA\DATA1234.BIN”

ข้อสำคัญ

- ชื่อไฟล์สามารถใช้ได้เฉพาะ **“รูปแบบ 8.3”** เท่านั้น (อักขระสูงสุด 12 อักขระ โดยเป็นชื่อไฟล์, เครื่องหมายมหัพภาค 8 อักขระ และนามสกุลอีก 3 อักขระ) ไม่สามารถใช้ชื่อไฟล์ยาวกว่านี้ได้
- พารามิเตอร์แรก **“ชื่อโฟลเดอร์”** และพารามิเตอร์ที่สอง **“ชื่อไฟล์”** สามารถใช้อักขระแบบไบต์เดี่ยวได้สูงสุด 32 ตัว
- สามารถระบุอุปกรณ์ภายในเป็น **“ชื่อไฟล์”** ของพารามิเตอร์ที่สองหรือสามได้ การระบุอุปกรณ์ภายในทำให้สามารถอ้างตำแหน่งของชื่อไฟล์โดยอ้อมได้ โดยสามารถระบุชื่อไฟล์ด้วยอักขระแบบไบต์เดี่ยวได้สูงสุด 32 ตัว
  ตัวอย่าง:

  _CF_rename ("\DATA", [w:[#INTERNAL]LS0100], [w:[#INTERNAL]LS0200])
  การจัดเก็บชื่อไฟล์ในพื้นที่ LS0100 และ LS0200 ทำให้อ้างตำแหน่งชื่อไฟล์ได้โดยอ้อม

- จัดเก็บชื่อไฟล์ในพื้นที่ LS0100 จนถึง LS0106 ดังต่อไปนี้

16 บิต

LS0100	‘D’	‘A’
LS0101	‘T’	‘A’
LS0102	‘0’	‘0’
LS0103	‘0’	‘1’
LS0104	‘.’	‘B’
LS0105	‘I’	‘N’
LS0106	‘\0’	‘\0’
	:	

ตอนท้ายของชื่อไฟล์ต้องเป็นอักขระค่าศูนย์ GP
รับรู้ข้อมูลที่อยู่ก่อนหน้าอักขระค่าศูนย์ว่าเป็นชื่อไฟล์

16 บิต

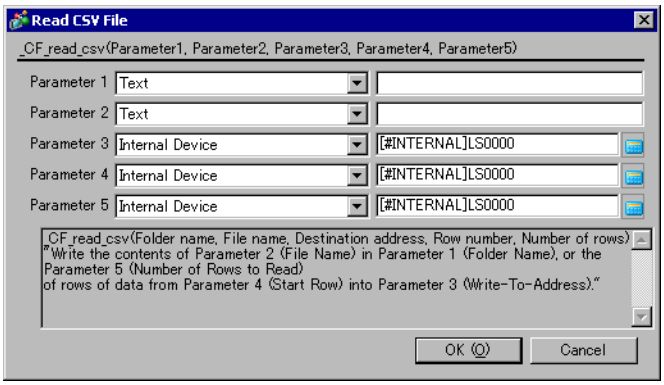
LS0200	‘D’	‘A’
LS0201	‘T’	‘A’
LS0202	‘1’	‘2’
LS0203	‘3’	‘4’
LS0204	‘.’	‘B’
LS0205	‘I’	‘N’
LS0206	‘\0’	‘\0’
	:	

จากตัวอย่างข้างบน ชื่อของไฟล์ "\DATA\DATA0001.BIN" ถูกเปลี่ยนเป็น "\DATA\DATA1234.BIN"

- เมื่อระบุพื้นที่ LS เป็น **“ชื่อไฟล์”** พื้นที่นั้นจะไม่นับเป็นตำแหน่งของ D-Script
- หากต้องการระบุโฟลเดอร์ราก (ไดเร็กทอรี) ให้ระบุชื่อโฟลเดอร์เป็น **“ ”** (สตริงเปล่า)
- ระบบจะแบ่งข้อมูลออกเป็นส่วนๆ และอ่านจากต้นฉบับ แม้ว่าจะขึ้นอยู่กับจำนวนไบต์ที่จะอ่านด้วยก็ตาม

21.5.4 Read CSV File

รายการ	คำอธิบาย
ข้อมูลสรุป	อ่านข้อมูลในเซลล์จากไฟล์ CSV (ซึ่งสร้างจากภาพเซลล์ที่คั่นด้วย “,”) และเขียนลงในตำแหน่งเวิร์ด
รูปแบบ	_CF_read_csv (ชื่อโฟลเดอร์, ชื่อไฟล์, [ตำแหน่งปลายทางการเขียน], แถวเริ่มต้น, จำนวนแถวที่จะอ่าน)



Parameter 1: ข้อความ (อักขระแบบไบต์เดี่ยวสูงสุด 32 อักขระ
Parameter 2: ข้อความ (อักขระแบบไบต์เดี่ยวสูงสุด 32 อักขระ), อุปกรณ์ภายใน, อุปกรณ์ภายใน + ตำแหน่งชั่วคราว
Parameter 3: อุปกรณ์ภายใน, อุปกรณ์ภายในที่กำหนดด้วยออฟเซ็ต
Parameter 4: ค่าตัวเลข (1 ถึง 65,535), อุปกรณ์ภายใน, ตัวแปรชั่วคราว
Parameter 5: ค่าตัวเลข (1 ถึง 65,535), อุปกรณ์ภายใน, ตัวแปรชั่วคราว

ตัวอย่างนิพจน์

_CF_read_csv ("\CSV", "SAMPLE.CSV", [w:[#INTERNAL]LS1000], 1, 2)

(เมื่อใช้ฟังก์ชัน “_CF_read_csv ( )” อ่านข้อมูลในการ์ดหน่วยความจำ CF จำนวนสองบรรทัด เริ่มจากบรรทัดแรกของไฟล์ [\CSV\SAMPLE.CSV])

SAMPLE.CSV

```
001, "DAT01-01", "DAT01-2"
002, "DAT02-01", "DAT02-2"
```

อ่านข้อมูลของไฟล์ CSV จำนวนสองบรรทัดโดยเริ่มจากบรรทัดแรก เมื่ออักขระตัวแรกเป็นค่าตัวเลข ("0" ถึง "9" หรือ "-") ข้อมูลจะถูกจัดเก็บเป็นค่าตัวเลข เมื่ออักขระตัวแรกเป็น ["] ระบบจะถือว่าข้อมูลเป็นอักขระและจัดเก็บค่า “00h” ไว้ที่ตอนท้ายของสตริงข้อความ เช่น เมื่อจัดเก็บ “DAT01-01” ขนาดข้อมูลคือ 8 อักขระซึ่งเป็นจำนวนคู่ และใช้เวิร์ดทั้งหมด 5 เวิร์ด โดย 4 เวิร์ดใช้สำหรับจัดเก็บสตริงข้อความ และอีก 1 เวิร์ดสำหรับจัดเก็บ “00h” ไว้ตอนท้าย เช่น เมื่อจัดเก็บ “DAT01-2” ขนาดข้อมูลคือ 7 อักขระซึ่งเป็นจำนวนคี่ และใช้เวิร์ดทั้งหมด 4 เวิร์ด เพื่อจัดเก็บข้อความโดยจะจัดเก็บ “00h” ไว้ตอนท้าย

16 บิต

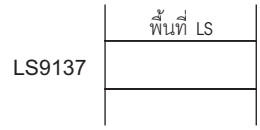
LS1000	1	
+1	‘D’	‘A’
+2	‘T’	‘0’
+3	‘1’	‘-’
+4	‘0’	‘1’
+5	00h	00h
+6	‘D’	‘A’
+7	‘T’	‘0’
+8	‘1’	‘-’
+9	‘2’	00h
LS1010	2	
+1	‘D’	‘A’
+2	‘T’	‘0’
+3	‘2’	‘-’
+4	‘0’	‘1’
+5	00h	00h
+6	‘D’	‘A’
+7	‘T’	‘0’
+8	‘2’	‘-’
+9	‘2’	00h

เมื่อโหมดการจัดเก็บข้อมูลคือโหมด 0

หมายเหตุ

- เมื่ออักขระตัวแรกในเซลล์เป็นค่าตัวเลข (“0” ถึง “9”, “_”) ระบบจะแปลงค่าเป็นข้อมูลตัวเลข แล้วเขียนข้อมูลนั้นลงในอุปกรณ์ LS ช่วงที่สามารถใช้ได้คือตั้งแต่ -32,768 ถึง 32,767
- เมื่ออักขระแรกในเซลล์เป็น [″] ระบบจะเขียนช่วงที่มีอักขระ [″] ลงในอุปกรณ์ LS เป็นข้อมูลสตริงข้อความ เมื่อจำนวนไบต์ของข้อมูลสตริงข้อความเป็นเลขคี่ ระบบจะเติม “0x00” ไว้ตอนท้าย เมื่อจำนวนไบต์ของข้อมูลตัวอักษรเป็นเลขคู่ ระบบจะเขียน “0x0000” ลงในตำแหน่งต่อจากตำแหน่งสุดท้าย ในหนึ่งเซลล์สามารถป้อนอักขระแบบไบต์เดี่ยวได้สูงสุด 32 อักขระ
- เมื่อไฟล์ CSV มีข้อมูล 2 บรรทัดขึ้นไป สามารถอ่านข้อมูลเป็นจำนวนบรรทัดที่ต้องการได้โดยเริ่มจากบรรทัดที่ระบุไว้ ในหนึ่งบรรทัดสามารถป้อนอักขระแบบไบต์เดี่ยวได้สูงสุด 200 อักขระ และในไฟล์ CSV หนึ่งไฟล์สามารถป้อนข้อมูลได้สูงสุด 65,535 บรรทัด
- เมื่อเกิดข้อผิดพลาด ระบบจะเขียนสถานะข้อผิดพลาดลงในพื้นที่ LS9137
- ขณะเขียนข้อมูลตัวอักษรของไฟล์ CSV ลงในอุปกรณ์ LS ลำดับการจัดเก็บข้อมูลจะขึ้นอยู่กับโหมดการจัดเก็บข้อมูล

สถานะข้อผิดพลาด



ชื่อฟังก์ชันตัวแก้ไข	พื้นที่ LS	สถานะข้อผิดพลาด	สาเหตุ
_CF_read_csv ( )	LS9137	0000h	เสร็จสมบูรณ์
		0001h	พารามิเตอร์เกิดข้อผิดพลาด
		000h	ข้อผิดพลาดของการ์ด CF (ไม่มีการ์ด CF/การเปิดไฟล์เกิดข้อผิดพลาด/การอ่านไฟล์เกิดข้อผิดพลาด)
		0003h	การเขียน/การอ่านเกิดข้อผิดพลาด

ข้อสำคัญ

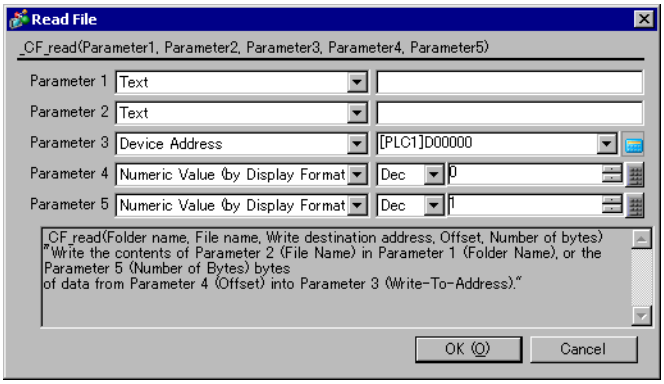
- เมื่อระบุชื่อโฟลเดอร์เป็น " * " คุณสามารถใช้พาธแบบครบถ้วนเป็นชื่อไฟล์ได้
- สามารถใช้ชื่อไฟล์ได้เฉพาะรูปแบบ 8.3 เท่านั้น (สูงสุด 12 อักขระ โดยแบ่งเป็นชื่อไฟล์, เครื่องหมายมหัพภาค 8 อักขระ และนามสกุลอีก 3 อักขระ) ชื่อไฟล์ที่ยาวกว่ารูปแบบนี้ไม่สามารถใช้ได้
- พื้นที่อุปกรณ์ LS ที่สามารถใช้จัดเก็บข้อมูลที่นำเข้ามาจากไฟล์ CSV ได้ มีเฉพาะพื้นที่สำหรับผู้ใช้ที่กำหนดไว้เท่านั้น (LS20 ถึง LS2031 และ LS2096 ถึง LS8191)
- เวลาที่ต้องใช้ในการประมวลผลสำหรับการนำเข้าข้อมูลจะเป็นสัดส่วนกับปริมาณข้อมูลของไฟล์ CSV ที่จะอ่าน ระบบจะไม่รีเฟรชพาร์ทจนกว่าจะประมวลผลเสร็จสมบูรณ์ (การอ่านข้อมูลในไฟล์ CSV ซึ่งมี 100 บรรทัด บรรทัดละ 40 อักขระ ตั้งแต่บรรทัดแรกจนถึงบรรทัดที่ 100 จะใช้เวลาประมาณ 10 วินาที)
- สถานะจะไม่ถูกบันทึกลงใน [s:CF_ERR_STAT] ในทันทีหลังจากเรียกใช้ฟังก์ชัน ซึ่งจะแตกต่างจากฟังก์ชัน [_CF_read ( )] (ในบางกรณี อาจจัดเก็บค่าที่ไม่ได้กำหนดไว้ด้วย)
- สตริงข้อความที่ขึ้นด้วยตัวเลขต้องใส่ ["] ไว้หน้าและหลังสตริงด้วย

(ตัวอย่าง)

[123, 2-D4EA] [123, "2-D4EA"]

X O

21.5.5 Read File

รายการ	คำอธิบาย
ข้อมูลสรุป	อ่านข้อมูลที่อยู่ต่อจากออฟเซ็ตที่ระบุไว้ตามจำนวนไบต์ที่ระบุ และเขียนข้อมูลที่อ่านได้ลงในตำแหน่งปลายทาง โปรดดูรายละเอียดเพิ่มเติมเกี่ยวกับลำดับการจัดเก็บข้อมูลได้ในส่วน “โหมดการจัดเก็บข้อมูล” ข้างล่าง
รูปแบบ	_CF_read (ชื่อโฟลเดอร์, ชื่อไฟล์, ตำแหน่งปลายทางการเขียน, ออฟเซ็ต, จำนวนไบต์)  Parameter 1 ชื่อโฟลเดอร์: สตริงแบบตายตัว (ความยาวสูงสุด: อักขระแบบไบต์เดี่ยว 32 อักขระ) Parameter 2 ชื่อไฟล์: สตริงที่กำหนดไว้ตายตัว, อุปกรณ์ภายใน, อุปกรณ์ภายใน + ตำแหน่งชั่วคราว (ความยาวสูงสุด: อักขระแบบไบต์เดี่ยว 32 ตัว) Parameter 3 ตำแหน่งปลายทางการเขียน: ตำแหน่งอุปกรณ์, ตำแหน่งอุปกรณ์ + ตำแหน่งชั่วคราว Parameter 4 ออฟเซ็ต: ค่าตัวเลข, ตำแหน่งอุปกรณ์, ตำแหน่งชั่วคราว (จำนวนสูงสุดที่สามารถระบุได้: 65,535 สำหรับความยาว 16 บิต, 4,294,967,295 สำหรับความยาว 32 บิต) Parameter 5 จำนวนไบต์: ค่าตัวเลข, ตำแหน่งอุปกรณ์, ตำแหน่งชั่วคราว (ความยาวสูงสุด: 1,280)

ตัวอย่างนิพจน์

การอ่านข้อมูลจำนวน 16 ไบต์ในไฟล์ที่ระบุ เมื่อออฟเซ็ตเท่ากับ 16

_CF_read ("\DATA", "DATA0001.BIN", [w:[#INTERNAL]LS0100], 16, 16)

จากตัวอย่างข้างบน ระบบจะเขียนข้อมูลจำนวน 16 ไบต์จากไบต์ที่ 17 เป็นต้นไปในไฟล์ “\DATA\DATA0001.BIN” ลงในพื้นที่ LS0100 และพื้นที่ถัดไป

ข้อสำคัญ

- ชื่อไฟล์สามารถใช้ได้เฉพาะ “รูปแบบ 8.3” เท่านั้น (อักขระสูงสุด 12 อักขระ โดยเป็นชื่อไฟล์, เครื่องหมายมหัพภาค 8 อักขระ และนามสกุลอีก 3 อักขระ) ไม่สามารถใช้ชื่อไฟล์ยาวกว่านี้ได้
- พารามิเตอร์แรก “ชื่อโฟลเดอร์” และพารามิเตอร์ที่สอง “ชื่อไฟล์” สามารถใช้อักขระแบบไบต์เดี่ยวได้สูงสุด 32 ตัว

ข้อสำคัญ

- สามารถระบุอุปกรณ์ภายในเป็น **"ชื่อไฟล์"** ของพารามิเตอร์ที่สองได้ การระบุอุปกรณ์ภายในทำให้สามารถอ้างตำแหน่งของชื่อไฟล์โดยอ้อมได้ โดยสามารถระบุชื่อไฟล์ด้วยอักขระแบบไบต์เดี่ยวได้สูงสุด 32 ตัว

  ตัวอย่าง:

  การอ่านข้อมูลจำนวน 10 ไบต์ที่จัดเก็บไว้ในไฟล์ เมื่อระบุไฟล์ในพื้นที่ LS0100 และพื้นที่ถัดไป และค่าออฟเซ็ตเท่ากับ 0

  _CF_read (**"**\DATA**"**, [w:LS0100], [w:LS0200], 0, 10)

  การจัดเก็บชื่อไฟล์ในพื้นที่ LS0100 ช่วยอ้างตำแหน่งชื่อไฟล์โดยอ้อมได้ ในตัวอย่างนี้ ชื่อไฟล์จะถูกจัดเก็บไว้ในพื้นที่ LS0100 จนถึง LS0106 ดังต่อไปนี้

  16 บิต

LS0100	'D'	'A'
LS0101	'T'	'A'
LS0102	'0'	'0'
LS0103	'0'	'1'
LS0104	'.'	'B'
LS0105	'I'	'N'
LS0106	'\0'	'\0'
	:	

  ← ตอนท้ายของชื่อไฟล์ต้องเป็นอักขระค่าศูนย์ อุปกรณ์แสดงผลรับรู้ข้อมูลที่อยู่ก่อนหน้าอักขระค่าศูนย์ว่าเป็นชื่อไฟล์

  จากตัวอย่างข้างบน ระบบจะอ่านข้อมูลจำนวน 10 ไบต์จากจุดเริ่มต้นของไฟล์ **"**\DATA\DATA0001.BIN**"** แล้วเขียนลงใน LS0200 และพื้นที่ถัดไป

- จำนวนไบต์ที่อ่านได้สำเร็จจะถูกเขียนลงในจำนวนไบต์ที่อ่านของการ์ด CF
- [s:CF_READ_NUM] โปรดดูรายละเอียดเพิ่มเติมได้ที่ **"**21.5.1 Label Settings ■ สถานะข้อผิดพลาดของการ์ด CF**"** (หน้า 21-37)
- อุปกรณ์ภายในที่ระบุใน **"ชื่อไฟล์"** และ **"ตำแหน่งปลายทางการเขียน"** จะไม่นับเป็นตำแหน่งของ D-Script
- เมื่อระบุตำแหน่งปลายทางการเขียนเป็นอุปกรณ์ PLC จำเป็นต้องใช้เวลาเพิ่มขึ้นในการเขียนข้อมูลลงใน PLC เพราะจำนวนเวิร์ด (ไบต์) เพิ่มขึ้น โดยอาจต้องใช้เวลาหลายวินาทีขึ้นอยู่กับจำนวนเวิร์ด
- หากข้อมูลที่อ่านจากไฟล์เกินกว่าช่วงอุปกรณ์ที่กำหนดไว้ของ PLC จะเกิดข้อผิดพลาดในการสื่อสารขึ้น ในกรณีนี้ คุณต้องปิดเครื่อง PLC แล้วเปิดใหม่เพื่อให้ PLC หายจากข้อผิดพลาด
- เมื่อระบุปลายทางเป็นอุปกรณ์ PLC ค่าจะไม่ถูกเขียนในทันทีเนื่องจากต้องใช้เวลาในการส่งข้อมูลจาก GP ไปยัง PLC

  ตัวอย่าง:

  ในสคริปต์ด้านล่าง ข้อความคำสั่ง (1) จะอ่านข้อมูลจำนวน 10 ไบต์จากไฟล์และเขียนข้อมูลลงใน [w:D0100] เมื่อเรียกใช้ข้อความคำสั่ง (2) ระบบยังไม่ได้เขียนข้อมูลลงใน [w:[PLC1]D0100] แต่อย่างใด เพราะต้องใช้เวลาในการส่งข้อมูล

  _CF_read ("\DATA", "DATA0001.BIN", [w:[PLC1]D0100], 0, 10) ..... (1)

  [w:[PLC1]D0200] = [w:[PLC1]D0100] + 1 ..... (2)

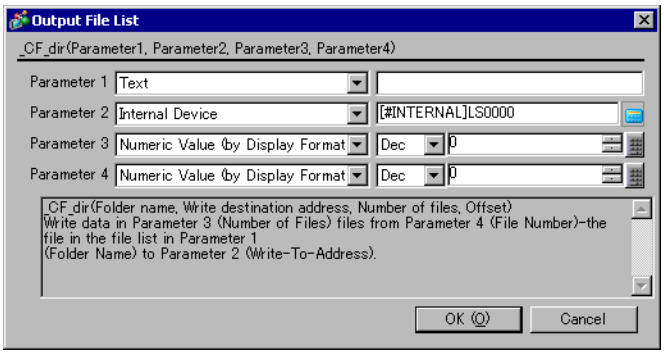
  ในกรณีเช่นนี้ ให้จัดเก็บข้อมูลหนึ่งครั้งในพื้นที่ LS แล้วเรียกใช้ข้อความคำสั่งที่สองดังต่อไปนี้

  _CF_read ("\DATA", "DATA0001.BIN", [w:[PLC1]D0100], 0, 10)

  memcpy ([w:[#INTERNAL]LS0100], [w:[PLC1]D0100], 10)

  [w:[PLC1]D0200] = [w:[#INTERNAL]LS0100] + 1

21.5.6 Output File List

รายการ	คำอธิบาย
ข้อมูลสรุป	เขียนรายการไฟล์ที่มีอยู่ในโฟลเดอร์ที่ระบุลงในอุปกรณ์ภายใน Parameter 1 จะบ่งชี้โฟลเดอร์ ข้อมูลการ์ด CF Parameter 4 จะระบุค่าออฟเซ็ตที่ใช้เลือกไฟล์หนึ่งหรือหลายไฟล์ในโฟลเดอร์นั้น Parameter 3 จะระบุจำนวนไฟล์ที่เลือกภายในโฟลเดอร์นั้น Parameter 2 จะระบุพื้นที่ LS ลงในไฟล์ที่จะถูกเขียน เมื่อระบุออฟเซ็ตเป็น “0” รายการจะเริ่มจากไฟล์แรกสุด
รูปแบบ	_CF_dir (ชื่อโฟลเดอร์, ตำแหน่งปลายทางการเขียน, จำนวนไฟล์, ออฟเซ็ต)  Parameter 1 ชื่อโฟลเดอร์: ข้อความที่กำหนดไว้ตายตัว (ความยาวสูงสุด: อักขระแบบไบต์เดี่ยว 32 อักขระ) Parameter 2 ตำแหน่งปลายทางการเขียน: อุปกรณ์ภายใน, อุปกรณ์ภายในที่กำหนดด้วยออฟเซ็ต Parameter 3 จำนวนไฟล์: ค่าตัวเลข, ตำแหน่งอุปกรณ์, ตำแหน่งชั่วคราว (ความยาวสูงสุด: 32) Parameter 4 ออฟเซ็ต: ค่าตัวเลข, ตำแหน่งอุปกรณ์, ตำแหน่งชั่วคราว

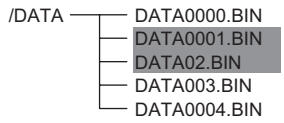
ตัวอย่างนิพจน์

การส่งข้อมูลรายการไฟล์ที่มีสองไฟล์เมื่อค่าออฟเซ็ตเท่ากับ 1 (ไฟล์ที่สอง)

_CF_dir ("\DATA*.*", [w:[#INTERNAL]LS0100], 2, 1)

เมื่อเรียกใช้ข้อความคำสั่งข้างบนขณะมีไฟล์ต่อไปนี้อยู่ในโฟลเดอร์ DATA ชื่อไฟล์ “DATA0001.BIN” และ “DATA02.BIN” จะถูกเขียนลงในพื้นที่ LS0100 และพื้นที่ถัดไป

รายละเอียดของโฟลเดอร์



รายละเอียดของพื้นที่ LS

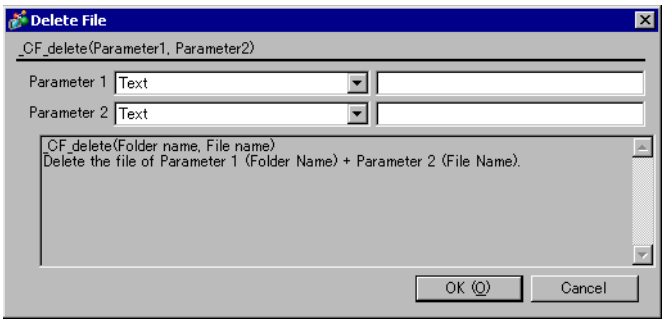
16 บิต

LS0100	‘D’	‘A’	ใช้เวิร์ด 7 เวิร์ด
LS0101	‘T’	‘A’	
LS0102	‘0’	‘0’	
LS0103	‘0’	‘1’	
LS0104	‘.’	‘B’	
LS0105	‘I’	‘N’	
LS0106	‘\0’	‘\0’	
LS0107	‘D’	‘A’	ใช้เวิร์ด 7 เวิร์ด
LS0108	‘T’	‘A’	
LS0109	‘0’	‘2’	
LS0110	‘.’	‘B’	
LS0111	‘I’	‘N’	
LS0112	‘\0’	‘\0’	
LS0113	‘\0’	‘\0’	

ข้อสำคัญ

- เมื่อระบุออฟเซ็ตเป็น “0” รายการจะเริ่มจากไฟล์ (เริ่มต้น) แรกสุด
- ชื่อไฟล์สามารถใช้ได้เฉพาะ “รูปแบบ 8.3” เท่านั้น (อักขระสูงสุด 12 อักขระ โดยเป็นชื่อไฟล์, เครื่องหมายมหัพภาค 8 อักขระ และนามสกุลอีก 3 อักขระ) ไม่สามารถใช้ชื่อไฟล์ยาวกว่านี้ได้
- หากโฟลเดอร์ที่ระบุมีไฟล์ไม่ครบตามที่ระบุไว้ ระบบจะเติมอักขระที่มีค่าศูนย์ (‘\0’) ลงในพื้นที่ LS ที่เหลืออยู่
- หากชื่อไฟล์มีน้อยกว่า 12 อักขระ ระบบจะเติมอักขระที่มีค่าศูนย์ (‘\0’) ลงในตำแหน่งที่ว่างอยู่
- หากต้องการระบุชื่อโฟลเดอร์ ต้องพิมพ์ "*.*" ลงไปด้วย (เช่น "\DATA*.*") "*.*" หมายความว่าให้แสดงรายการไฟล์ทั้งหมด เช่น "*" โปรดตรวจสอบว่าได้อธิบาย "*.*" "*.*" หมายความว่าแสดงรายการไฟล์ทั้งหมด
- จำนวนไฟล์ตามที่แสดงจริงจะถูกเขียนลงในจำนวนไฟล์ที่แสดงในการ์ด CF [s:CF_FILELIST_NUM] โปรดดูรายละเอียดเพิ่มเติมได้ที่ “ ■ สถานะข้อผิดพลาดของการ์ด CF” (หน้า 21-37)
- ตำแหน่ง LS ที่เป็นปลายทางการเขียนจะไม่นับเป็นตำแหน่งของ D-Script
- ระบบจะเขียนชื่อไฟล์ลงในพื้นที่ LS โดยไม่เรียงลำดับตัวอักษร แต่จะเขียนตามลำดับการสร้าง (ลำดับการป้อนข้อมูล FAT)
- คุณสามารถสร้างรายการได้โดยระบุนามสกุลไฟล์ หากต้องการแสดงไฟล์เฉพาะที่มีนามสกุลไฟล์ที่ต้องการ ให้ใช้รูปแบบต่อไปนี้ เช่น “\DATA*.BIN” แต่ทั้งนี้ ในชื่อไฟล์จะมีเครื่องหมาย "*" อยู่ไม่ได้

21.5.7 Delete File

รายการ	คำอธิบาย
ข้อมูลสรุป	**ลบไฟล์ที่ระบุออกจากการ์ด CF Parameter 1 จะบ่งชี้โฟลเดอร์ข้อมูลการ์ด CF Parameter 2 จะบ่งชี้ชื่อของไฟล์ที่จะถูกลบ**
รูปแบบ	**_CF_delete (ชื่อโฟลเดอร์, ชื่อไฟล์)** **คุณสามารถกำหนดชื่อไฟล์โดยอ้อมด้วยตำแหน่ง LS ได้ด้วย**  Parameter 1 **ชื่อโฟลเดอร์: ข้อความที่กำหนดไว้ตายตัว** Parameter 2 **ชื่อไฟล์: ข้อความที่กำหนดไว้ตายตัว, อุปกรณ์ภายใน, อุปกรณ์ภายใน + ตำแหน่งชั่วคราว**

ตัวอย่างนิพจน์

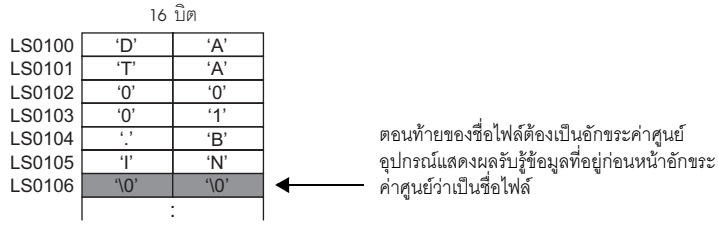
_CF_delete ("\DATA", "DATA0001.BIN")

ตัวอย่างข้างบนเป็นการลบไฟล์ ″\DATA \DATA0001.BIN″

ข้อสำคัญ

- ชื่อไฟล์สามารถใช้ได้เฉพาะ **"รูปแบบ 8.3"** เท่านั้น (อักขระสูงสุด 12 อักขระ โดยเป็นชื่อไฟล์, เครื่องหมายมหัพภาค 8 อักขระ และนามสกุลอีก 3 อักขระ) ไม่สามารถใช้ชื่อไฟล์ยาวกว่านี้ได้
- พารามิเตอร์แรก **"ชื่อโฟลเดอร์"** และพารามิเตอร์ที่สอง **"ชื่อไฟล์"** สามารถใช้อักขระแบบไบต์เดี่ยวได้สูงสุด 32 ตัว
- สามารถระบุอุปกรณ์ภายในเป็น **"ชื่อไฟล์"** ของพารามิเตอร์ที่สองได้ การระบุอุปกรณ์ภายในทำให้สามารถอ้างตำแหน่งของชื่อไฟล์โดยอ้อมได้ โดยสามารถระบุชื่อไฟล์ด้วยอักขระแบบไบต์เดี่ยวได้สูงสุด 32 ตัว

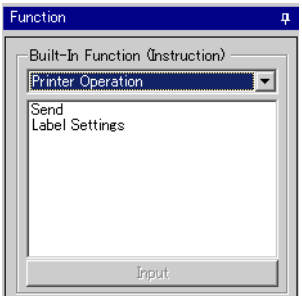
ในตัวอย่างนี้ ชื่อไฟล์จะถูกจัดเก็บไว้ในพื้นที่ LS0100 จนถึง LS0106 ดังต่อไปนี้



จากตัวอย่างข้างบน ไฟล์ "\DATA\DATA0001.BIN" ถูกลบทิ้ง

- หากต้องการระบุโฟลเดอร์ราก (ไดเร็กทอรี) ให้ระบุชื่อโฟลเดอร์เป็น **" "** (สตริงเปล่า)
- เมื่อระบุพื้นที่ LS เป็น **"ชื่อไฟล์"** จะไม่นับ **"ตำแหน่งปลายทางการเขียน"** เป็นตำแหน่งของ D-Script
- หากต้องการระบุพาธแบบครบถ้วนเป็นชื่อไฟล์ ให้ระบุ **"*"** (เครื่องหมายดอกจัน) เป็นชื่อโฟลเดอร์

21.6 การทำงานของเครื่องพิมพ์

Printer Operation	ข้อมูลสรุปของฟังก์ชัน
	Label Settings ☞ "21.6.1 Label Settings" (หน้า 21-59) **กำหนดจากตัวแปรควบคุมและตัวแปรสถานะ**
	Send ☞ "21.6.2 Send" (หน้า 21-61) **ส่งข้อมูลเท่ากับจำนวนไบต์ที่กำหนดไปยังพอร์ต COM**

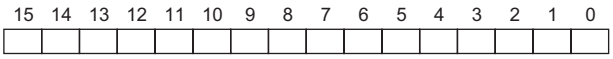
ข้อสำคัญ
- พอร์ตที่สามารถใช้ฟังก์ชันการทำงานของเครื่องพิมพ์ได้ ได้แก่ COM1 หรือ USB/PIO (USB-PIO)

21.6.1 Label Settings

■ ตัวแปรควบคุม

(PRN_CTRL) คือตัวแปรควบคุม ใช้สำหรับล้างข้อมูลในบัฟเฟอร์การส่งข้อมูลและสถานะข้อผิดพลาด ตัวแปรนี้เป็นแบบเขียนอย่างเดียว

- สรุปตัวแปรควบคุม (PRN_CTRL)



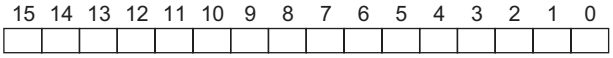
บิต	รายละเอียด
15	สำรอง
14	
13	
12	
11	
10	
9	
8	
7	
6	
5	
4	
3	
2	1: ล้างข้อผิดพลาด
1	สำรอง
0	1: ล้างบัฟเฟอร์การส่งข้อมูล

ข้อสำคัญ
- เมื่อเลือกเวิร์ด และตั้งค่าบิตพร้อมกันสองบิตขึ้นไป จะทำการประมวลผลตามลำดับดังนี้:
  ล้างข้อผิดพลาด
  ถึง
  ล้างข้อมูลในบัฟเฟอร์การส่งข้อมูล
- ห้ามใช้บิตที่สำรองไว้ ตั้งค่าเฉพาะบิตที่จำเป็นเท่านั้น

■ สถานะ

ตัวแปรสถานะนี้ (PRN_STAT) ใช้สำหรับตรวจสอบการมีอยู่/การขาดหายไปของข้อมูลในบัฟเฟอร์การส่งข้อมูล และใช้สำหรับรับสถานะข้อผิดพลาด ตัวแปรสถานะนี้เป็นแบบเขียนอย่างเดียว

- **รายละเอียดของตัวแปรสถานะ (PRN_STAT)**

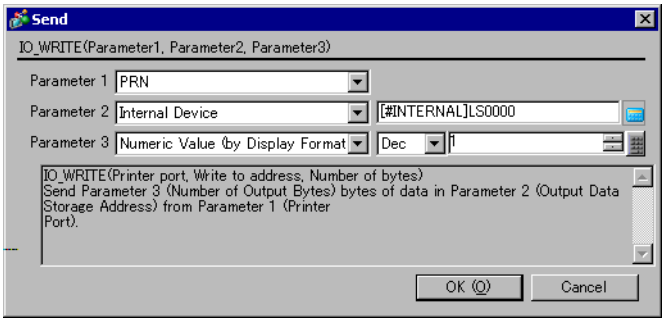


บิต	รายละเอียด
15	สำรอง
14	สถานะของสัญญาณ I/F ERROR ของเครื่องพิมพ์ ข้อผิดพลาดของเครื่องพิมพ์ (อินพุต): 0: ข้อผิดพลาด 1: ปกติ
13	สถานะของสัญญาณ I/F SLCT ของเครื่องพิมพ์ เลือก (อินพุต): 0: ออฟไลน์ 1: ออนไลน์
12	สถานะของสัญญาณ I/F PE ของเครื่องพิมพ์ กระดาษหมด (อินพุต): 0: ปกติ 1: กระดาษหมด
11	สำรอง
10	
9	
8	
7	
6	
5	
4	
3	
2	
1	0: ปกติ 1: การส่งเกิดข้อผิดพลาด
0	0: มีข้อมูลอยู่ในบัฟเฟอร์การส่งข้อมูล 1: ไม่มีข้อมูลอยู่ในบัฟเฟอร์การส่งข้อมูล

ข้อสำคัญ

- หากบัฟเฟอร์การส่งข้อมูลเกิดโอเวอร์โฟลว์ จะเกิดข้อผิดพลาด เมื่อเกิดข้อผิดพลาดนี้ขึ้น บิตข้อผิดพลาดในการส่งจะเปิดขึ้น
- บัฟเฟอร์การส่งข้อมูลมีขนาดเท่ากับ 8,192 ไบต์
- บิตที่สำรองไว้อาจถูกกำหนดในภายหลัง ดังนั้น ให้กำหนดเฉพาะบิตที่จำเป็นเท่านั้น

21.6.2 Send

รายการ	คำอธิบาย
ข้อมูลสรุป	ส่งข้อมูลเท่ากับจำนวนไบต์ที่กำหนดไปยังพอร์ต COM ระบบจะส่งข้อมูลโดยไม่คำนึงถึงชนิดของเครื่องพิมพ์ที่ระบุไว้
รูปแบบ	IO_WRITE ([p:PRN], ตำแหน่งจัดเก็บข้อมูลส่งออก, จำนวนไบต์ที่ส่งออก)  Parameter 1: [p:PRN] Parameter 2: อุปกรณ์ภายใน Parameter 3: ค่าจำนวนเต็ม, ตำแหน่งอุปกรณ์, ตำแหน่งชั่วคราว

ข้อสำคัญ

- ค่าตัวเลขสูงสุดที่สามารถระบุให้ Parameter 3 คือ 1024 ถึงแม้จะระบุค่ามากกว่า 1024 แต่จะส่งออกข้อมูลจากพอร์ต COM ได้เพียง 1024 ไบต์เท่านั้น

ตัวอย่างนิพจน์ 1

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 10)

จากตัวอย่างข้างบน เป็นการส่งข้อมูลจำนวน 10 ไบต์ที่จัดเก็บอยู่ในพื้นที่ LS1000 และพื้นที่ถัดไปจากพอร์ต COM

ตัวอย่างนิพจน์ 2

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], [w:[#INTERNAL]LS0800])

จากตัวอย่างข้างบน เป็นการส่งข้อมูลที่จัดเก็บอยู่ในพื้นที่ LS1000 และพื้นที่ถัดไปจากพอร์ต COM จำนวนไบต์จะเหมือนกับที่เขียนใน LS0800

ตัวอย่างนิพจน์ 3

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS 1000], [t:0010])

จากตัวอย่างข้างบน เป็นการส่งข้อมูลที่จัดเก็บอยู่ในพื้นที่ LS1000 และพื้นที่ถัดไปจากพอร์ต COM จำนวนไบต์จะเหมือนกับที่เขียนในตำแหน่งชั่วคราว [t:0010]

โหมดการจัดเก็บข้อมูล

เมื่ออ่านข้อมูลจากตำแหน่งอุปกรณ์ขณะเรียกใช้ฟังก์ชันการทำงานของพอร์ต COM คุณสามารถระบุลำดับการจัดเก็บของข้อมูลที่อ่านได้

การตั้งค่าโหมดการจัดเก็บข้อมูลในพื้นที่ LS9130 สามารถเปลี่ยนลำดับการจัดเก็บได้ โดยสามารถเลือกโหมดได้จากตัวเลือกทั้งสี่ ได้แก่ โหมด 0, 1, 2 หรือ 3

◆ **โหมด** 0

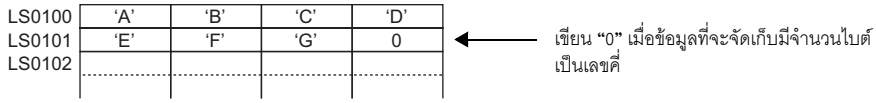
ตัวอย่างเช่น การใช้ฟังก์ชันการทำงานของพอร์ต COM อ่านสตริง “ABCDEFG” จากตำแหน่งอุปกรณ์

[w:[#INTERNAL]LS9130] = 0

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)

- เมื่อตำแหน่งอุปกรณ์ยาว 16 บิต

- เมื่อตำแหน่งอุปกรณ์ยาว 32 บิต



◆ **โหมด** 1

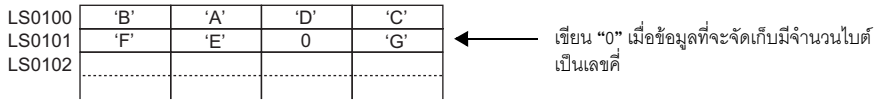
ตัวอย่างเช่น การใช้ฟังก์ชันการทำงานของพอร์ต COM อ่านสตริง “ABCDEFG” จากตำแหน่งอุปกรณ์

[w:[#INTERNAL]LS9130] = 1

IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)

- เมื่อตำแหน่งอุปกรณ์ยาว 16 บิต

- เมื่อตำแหน่งอุปกรณ์ยาว 32 บิต

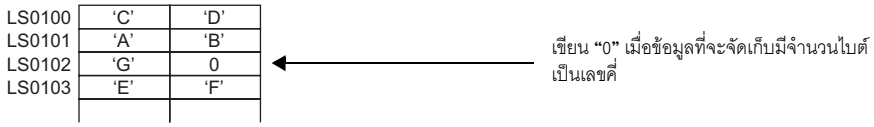


◆ **โหมด 2**

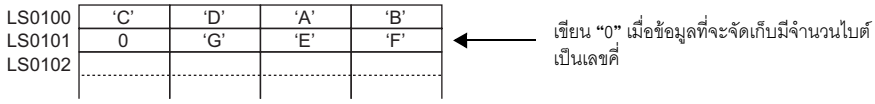
ตัวอย่างเช่น การใช้ฟังก์ชันการทำงานของพอร์ต COM อ่านสตริง “ABCDEFG” จากตำแหน่งอุปกรณ์

```
[w:[#INTERNAL]LS9130] = 2
IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)
```

- **เมื่อตำแหน่งอุปกรณ์ยาว 16 บิต**



- **เมื่อตำแหน่งอุปกรณ์ยาว 32 บิต**

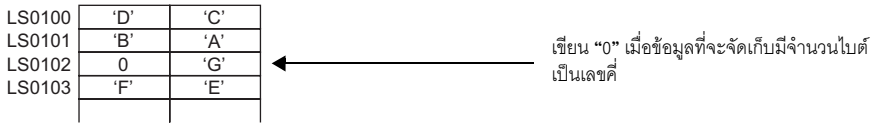


◆ **โหมด 3**

ตัวอย่างเช่น การใช้ฟังก์ชันการทำงานของพอร์ต COM อ่านสตริง “ABCDEFG” จากตำแหน่งอุปกรณ์

```
[w:[#INTERNAL]LS9130] = 3
IO_WRITE ([p:PRN], [w:[#INTERNAL]LS1000], 7)
```

- **เมื่อตำแหน่งอุปกรณ์ยาว 16 บิต**



- **เมื่อตำแหน่งอุปกรณ์ยาว 32 บิต**

LS0100	'D'	'C'	'B'	'A'
LS0101	0	'G'	'F'	'E'
LS0102				

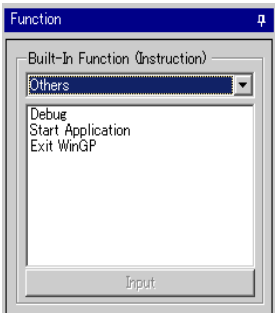
← เขียน "0" เมื่อข้อมูลที่จะจัดเก็บมีจำนวนไบต์เป็นเลขคี่

ข้อสำคัญ

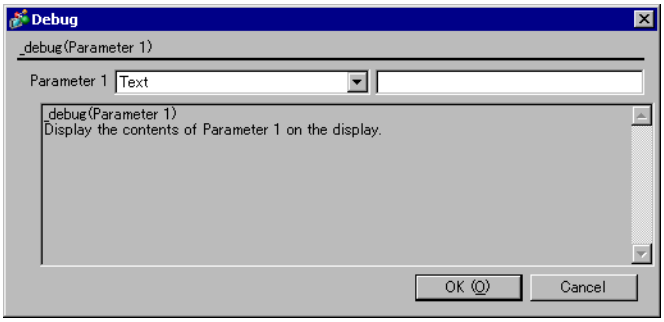
- โหมดการจัดเก็บข้อมูลไม่เหมือนกับโหมดข้อมูลสตริงในการตั้งค่าระบบ ความสัมพันธ์ระหว่างโหมดการจัดเก็บข้อมูลกับโหมดข้อมูลสตริงจะแสดงอยู่ในตารางด้านล่าง

ลำดับการจัดเก็บของอุปกรณ์ข้อมูล	ลำดับการจัดเก็บไบต์ในเวิร์ดแบบ LH หรือ HL	ลำดับการจัดเก็บในดับเบิลเวิร์ดแบบ LH หรือ HL	โหมดการจัดเก็บข้อมูล D-Script	โหมดข้อมูลตัวอักษร
จัดเก็บจากข้อมูลเริ่มต้น	ลำดับ HL	ลำดับ HL	0	1
	ลำดับ LH		1	2
	ลำดับ HL	ลำดับ LH	2	5
	ลำดับ LH		3	4
จัดเก็บจากข้อมูลท้ายสุด	ลำดับ HL	ลำดับ HL	-	3
	ลำดับ LH		-	7
	ลำดับ HL	ลำดับ LH	-	8
	ลำดับ LH		-	6

21.7 อื่นๆ

อื่นๆ	ข้อมูลสรุปของฟังก์ชัน
	ฟังก์ชัน Debug ☞ "21.7.1 ฟังก์ชัน Debug" (หน้า 21-65) **แสดงตำแหน่งหรือข้อความที่กำหนดบนหน้าจอเพื่อทำการตรวจแก้ข้อบกพร่อง**
	Start Application ☞ "21.7.2 Triggering Application" (หน้า 21-67) **สั่งใช้งานช่วงที่ระบุแล้วเริ่มเปิดการทำงานของแอ็พพลิเคชัน**
	Exit WinGP ☞ "21.7.3 Exit WinGP" (หน้า 21-69) Exit WinGP

21.7.1 ฟังก์ชัน Debug

รายการ	คำอธิบาย
ข้อมูลสรุป	**แสดงตำแหน่งหรือข้อความที่กำหนดบนหน้าจอเพื่อทำการตรวจแก้ข้อบกพร่อง** **หลังจากตรวจแก้ข้อบกพร่องเสร็จแล้ว ให้ยกเลิกการเลือกช่องทำเครื่องหมาย [Enable Debug Function] ของตัวแก้ไขสคริปต์ และฟังก์ชันดีบักจะปิดลง**
รูปแบบ	_debug (Parameter 1)  Parameter 1: **ข้อความ (อักขระแบบไบต์เดี่ยวสูงสุด 32 อักขระ**

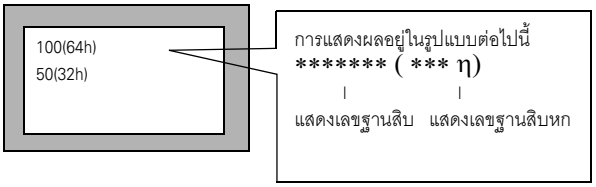
■ **รายละเอียดของ** Parameter 1

Parameter 1	**รูปแบบ**	**คำอธิบาย**
ข้อความ	_debug (″ABC″)	แสดงข้อความในเครื่องหมาย ″ ″ ข้อความสามารถใช้อักขระแบบไบต์เดี่ยวได้สูงสุด 32 อักขระ
ตำแหน่งเวิร์ด หรือ ตำแหน่งชั่วคราว	_debug (w:D1000)	แสดงค่าของตำแหน่งเวิร์ดหรือตำแหน่งชั่วคราวที่กำหนดไว้
การขึ้นบรรทัดใหม่	_debug (_CRLF)	เลื่อนเคอร์เซอร์ไปยังจุดเริ่มต้นของบรรทัดถัดไป
การปัดแคร่	_debug (_CR)	เลื่อนเคอร์เซอร์ไปยังจุดเริ่มต้นของบรรทัดเดียวกัน

ตัวอย่างนิพจน์ 1

สคริปต์ต่อไปนี้จะแสดงค่าของตำแหน่งเวิร์ด

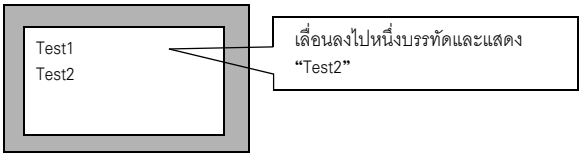
```
[w:[#INTERNAL]LS0100]=100
_debug
([w:[#INTERNAL]LS0100])
_debug (_CRLF)
[w:[#INTERNAL]LS0100]=50
_debug
([w:[#INTERNAL]LS0100])
```



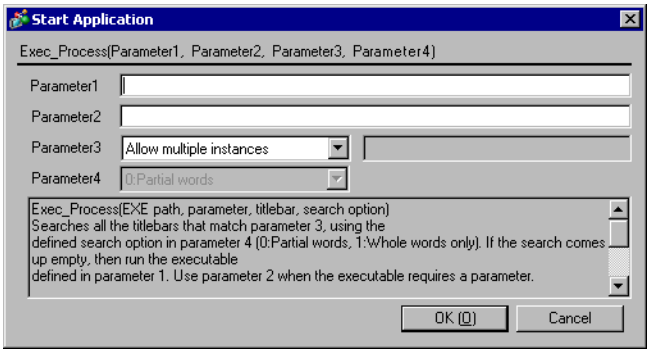
ตัวอย่างนิพจน์ 2

สคริปต์ต่อไปนี้จะแสดงการขึ้นบรรทัดใหม่และข้อความ

```
_debug ("Test1")
_debug (_CRLF)
_debug ("Test2")
```



21.7.2 Triggering Application

รายการ	คำอธิบาย
ข้อมูลสรุป	สั่งใช้งานไฟล์ EXE ที่ระบุเพื่อเริ่มการทำงานของแอ็พพลิเคชัน คุณสามารถระบุการตั้งค่าต่าง ๆ เช่น พารามิเตอร์เริ่มต้น และการตรวจสอบ (Watch) สำหรับการเริ่มทำงานหลายแอ็พพลิเคชัน
รูปแบบ	Exec Process (Parameter1, Parameter2, Parameter3, Parameter4)  Parameter 1 EXE path: ป้อนพาธที่สมบูรณ์ของไฟล์ปฏิบัติการ (.exe) สำหรับแอ็พพลิเคชันที่ต้องการให้เริ่มทำงาน คุณสามารถป้อนอักขระได้ไม่เกิน 255 ตัว Parameter 2 Parameter: ป้อนอาร์กิวเมนต์เริ่มต้นของไฟล์ปฏิบัติการ คุณสามารถป้อนอักขระได้ไม่เกิน 255 ตัว Parameter 3 Window Title: ถ้าไม่ต้องการอนุญาตให้ใช้งานอินสแตนซ์หลายค่า ให้เลือก "Do not allow multiple instances" แล้วป้อนข้อมูล [Window Title] คุณสามารถป้ออักขระได้ไม่เกิน 63 ตัว แอ็พพลิเคชันจะไม่สามารถเริ่มทำงานได้หากพบหน้าต่างเดียวกับ [Window Title] อนุญาตให้ใช้อินสแตนซ์หลายค่าได้ หากคุณเลือก "Allow multiple instances" หรือไม่ระบุ [Window Title] Parameter 4 Find whole window titles only: เปิดใช้งานเฉพาะเมื่อเลือก Parameter3 - "Do not allow multiple instances" เมื่อเลือก "0: Partial Words" แอ็พพลิเคชันที่ระบุจะไม่ทำงานถ้าพบหน้าต่างที่มีชื่อแม้บางส่วนตรงกับใน [Window Title] เมื่อเลือก "1: Whole Words Only" แอ็พพลิเคชันที่ระบุจะไม่ทำงาน ถ้าพบหน้าต่างที่มีส่วนตรงทั้งหมดกับใน [Window Title]

หมายเหตุ

- Parameter1 จำเป็นต้องมีข้อความ (EXE path) ข้อผิดพลาดจะเกิดขึ้นเมื่อคุณไม่ป้อนข้อความ
- คุณสมบัตินี้จะไม่ทำงานในผลิตภัณฑ์รุ่นอื่น ๆ นอกเหนือจาก IPC Series

■ วิธีป้อนข้อมูล Parameter 1 (EXE path)

มี 3 วิธีที่จะป้อนข้อมูลที่ EXE path:
คำอธิบายต่อไปนี้ คือตัวอย่างการเรียกใช้งานไฟล์ sample.exe ใน C:\Documents and Settingsuser\Local Settings\Temp

1. การระบุพาธแบบครบถ้วน
   ตัวอย่าง C:\Documents and Settings\user\Local Settings\Temp\sample.exe
2. ชื่อไฟล์ EXE เท่านั้น
   ถ้าไฟล์ปฏิบัติการอยู่ในโฟลเดอร์ ซึ่งระบุว่าเป็นพาธใน Environment Settings ที่ IPC Series
   ตัวอย่าง sample.exe
   (เริ่มทำงานถ้าการตั้งค่าคือ Path=C:\Documents and Settings\user\Local Settings\Temp)

   ถ้าไฟล์ปฏิบัติการอยู่ในโฟลเดอร์ ซึ่งระบุโดยพารามิเตอร์สภาพแวดล้อมใน Environment Settings ที่ IPC Series
   ตัวอย่าง %TEMP%\sample.exe
   (เริ่มทำงานถ้าระบุพารามิเตอร์สภาพแวดล้อมเป็น TEMP=C:\Documents and Settings\user\Local Settings\Temp)

ตัวอย่างนิพจน์ 1

อนุญาตให้มีอินสแตนซ์หลายค่า (เปิดใช้งาน notepad และแสดงไฟล์ Readme.txt)

```
Exec_Process ("C:\WINDOWS\SYSTEM32\notepad.exe","D:\TEMP\Readme.txt","",0)
Exec_Process ("%SystemFolder%\notepad.exe","D:\TEMP\Readme.txt","",1)
```

ตัวอย่างนิพจน์ 2

ไม่อนุญาตให้มีอินสแตนซ์หลายค่า: Partial Words (บางคำ) (เปิดใช้งาน notepad และแสดงไฟล์ Readme.txt)

```
Exec_Process
("C:\WINDOWS\SYSTEM32\notepad.exe","D:\TEMP\Readme.txt","Readme",0)
```

ตัวอย่างนิพจน์ 3

ไม่อนุญาตให้มีอินสแตนซ์หลายค่า: Whole Words Only (ทุกคำเท่านั้น) (เปิดใช้งาน notepad และแสดงไฟล์ Readme.txt)

```
Exec_Process ("C:\WINDOWS\SYSTEM32\notepad.exe","D:\TEMP\Readme.txt","
Readme.txt",1)
```

ตัวอย่างนิพจน์ 4

ไม่อนุญาตให้มีอินสแตนซ์หลายค่า: Partial Words (บางคำ) (เปิดใช้งาน notepad)

```
Exec_Process ("C:\WINDOWS\SYSTEM32\notepad.exe","","notepad",0Åj
```

ตัวอย่างนิพจน์ 5

ไม่มีพารามิเตอร์ (เปิดใช้งาน notepad)

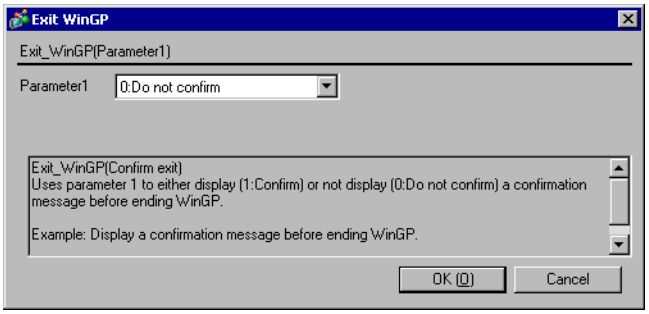
```
Exec_Process ("C:\WINDOWS\SYSTEM32\notepad.exe","","",0)
```

ตัวอย่างนิพจน์ 6

พารามิเตอร์หลายค่า (เปิดใช้งาน sample.exe)

```
Exec_Process ("C:\WINDOWS\SYSTEM32\sample.exe","/v /a/s","",1)
```

21.7.3 Exit WinGP

รายการ	คำอธิบาย
ข้อมูลสรุป	Exit WinGP คุณสามารถกำหนดให้แสดงข้อความรับทราบเมื่อออกจากระบบได้
รูปแบบ	_Exit WinGP (ยืนยันการสิ้นสุด)  Parameter 1 ชื่อโฟลเดอร์: เลือก "0: Do not Confirm" หรือ "1: Confirm".

หมายเหตุ

- Parameter1 จำเป็นต้องมีข้อความ (EXE path) ข้อผิดพลาดจะเกิดขึ้นเมื่อคุณไม่ป้อนข้อความ
- คุณสมบัตินี้จะไม่ทำงานถ้าคุณถ่ายโอนสคริปต์ "Exit WinGP" ไปยังผลิตภัณฑ์รุ่นอื่นนอกเหนือจาก IPC Series

ตัวอย่างนิพจน์

แสดงข้อความรับทราบเมื่อออกจาก WinG

Exit WinGP(1)

21.8 นิพจน์เงื่อนไข

นิพจน์เงื่อนไข	ข้อมูลสรุปของฟังก์ชัน
Description Expression if - endif if - else - endif loop - endloop break	if - endif ☞ "21.8.1 if - endif" (หน้า 21-70) เมื่อเงื่อนไขของ "if" ที่อยู่ในวงเล็บ "( )" เป็นจริง นิพจน์ที่อยู่ตามหลังข้อความคำสั่ง "if ( )" จะเริ่มทำงาน
	if - else - endif ☞ "21.8.2 if - else - endif" (หน้า 21-70) เมื่อเงื่อนไขของ "if" ที่อยู่ในวงเล็บ "( )" เป็นจริง นิพจน์ที่อยู่ตามหลังข้อความคำสั่ง "if ( )" จะเริ่มทำงาน เมื่อเงื่อนไขเป็นเท็จ นิพจน์ "else" จะทำงาน
	loop - endloop ☞ "21.8.3 loop - endloop" (หน้า 21-71) ประมวลผลแบบวนลูปเป็นจำนวนครั้งเท่ากับตัวเลขที่จัดเก็บอยู่ในตำแหน่งชั่วคราว ซึ่งกำหนดอยู่ในวงเล็บ "( )" ที่อยู่หลัง "loop"
	break ☞ "21.8.4 break" (หน้า 21-73) หยุดการวนลูปขณะกำลังเรียกใช้สมการ loop ( )
	return ☞ "21.8.5 return" (หน้า 21-74) ดำเนินการอีกครั้งตั้งแต่เริ่มต้น ตัวแปรนี้ใช้ได้เฉพาะใน Extended Script เท่านั้น

21.8.1 if - endif

เมื่อเงื่อนไขภายในวงเล็บ "( )" ที่อยู่ตามหลัง "if" เป็นจริง ระบบจะดำเนินกระบวนการที่อยู่หลังข้อความคำสั่ง "if ( )"

หมายเหตุ
- อักขระ "=" ในคำสั่ง Assign ใช้ไม่ได้ในนิพจน์เงื่อนไข

21.8.2 if - else - endif

เมื่อเงื่อนไขภายในวงเล็บ "( )" ที่อยู่ตามหลัง "if" เป็นจริง ระบบจะดำเนินกระบวนการที่อยู่หลังข้อความคำสั่ง "if ( )" เมื่อเงื่อนไขเป็นเท็จ ระบบจะดำเนินการตามข้อความคำสั่งที่อยู่ถัดจาก "else"

หมายเหตุ
- อักขระ "=" ในคำสั่ง Assign ใช้ไม่ได้ในนิพจน์เงื่อนไข

21.8.3 loop - endloop

ประมวลผลแบบวนลูปเป็นจำนวนครั้งเท่ากับตัวเลขที่จัดเก็บอยู่ในตำแหน่งชั่วคราวซึ่งกำหนดอยู่ในวงเล็บ "( )" ที่อยู่หลัง "loop"

การวนลูปแบบไม่รู้จบ

เมื่อไม่ได้ป้อนข้อความคำสั่งในวงเล็บของ loop ( ) การวนลูปจะเป็นแบบไม่รู้จบ
คุณใช้การวนลูปแบบไม่รู้จบได้เฉพาะใน Extended Script เท่านั้น

ตัวอย่างนิพจน์

```
loop ( )
{
    [w:[#INTERNAL]LS0100]=[w:[#INTERNAL]LS0100]+1
    if ( [w:[#INTERNAL]LS0100] >10)
    {
        break
    }
    endif
}
endloop
```

หมายเหตุ

- รูปแบบของ loop ( ) จะเป็นดังต่อไปนี้
- ตัวอย่าง:

```
loop (จำนวนลูป)   <= กำหนดตำแหน่งชั่วคราวที่จัดเก็บจำนวนครั้งในการวนลูป
  {
    สมการการดำเนินการ
    break      <= ใช้เพื่อออกจากพาร์ทการวนลูป (เลือกหรือไม่ก็ได้)
  } endloop    <= กำหนดจุดสิ้นสุดการวนลูป
```

- สามารถป้อนได้เฉพาะตำแหน่งเวิร์ดชั่วคราวในวงเล็บเท่านั้น (ตัวอย่าง: loop ([t:000]))
- ใช้ "loop ( )" กับสมการทริกเกอร์ไม่ได้
- ค่าตำแหน่งเวิร์ดชั่วคราวที่ใช้กำหนดจำนวนครั้งของการวนลูป จะลดลงทุกครั้งที่ทำการวนลูป เมื่อค่าเปลี่ยนเป็น 0 การวนลูปจะสิ้นสุดลง หากมีการแก้ไขค่าตำแหน่งเวิร์ดชั่วคราว ที่ใช้กำหนดจำนวนครั้งของการวนลูป การวนลูปจะดำเนินไปไม่รู้จบ ตำแหน่งเวิร์ดชั่วคราว ที่ใช้จะถูกกำหนดเป็นค่าที่ใช้เหมือนกันในทุกที่ ดังนั้น การใช้ตำแหน่งเวิร์ดชั่วคราวนี้พร้อมกันกับวัตถุประสงค์อื่นอาจทำให้เกิดการวนลูปไม่รู้จบได้
- การแสดงพาร์ทบนหน้าจอ ฯลฯ จะไม่ถูกอัพเดต/รีเฟรชจนกว่าจะวนลูปเสร็จ
- loop ( ) ยังสามารถเขียนซ้อนกันได้ด้วย เมื่อเขียนการวนลูปซ้อนกัน loop ( ) ที่อยู่ด้านในสุด จะถูกข้ามไปโดยผ่านทางคำสั่ง "break"

```
loop ([t:0000])  // ลูป 1
{
   loop  ([t:0001]) // ลูป 2
   {
      break      // ออกจากลูป 2
   }endloop

   break         // ออกจากลูป 1
}endloop
```

- หากการวนลูปเสร็จสิ้นลงโดยไม่ได้ใช้คำสั่ง escape ค่าตำแหน่งเวิร์ดชั่วคราวจะกลายเป็น 0

หมายเหตุ

- ช่วงค่าตำแหน่งเวิร์ดชั่วคราวที่สามารถใช้ได้จะแตกต่างไป โดยขึ้นอยู่กับรูปแบบของข้อมูล (Bin, BCD), ความยาวบิต และรหัส +/- ที่ใช้ หากตั้งค่ารหัส +/- และตำแหน่งเวิร์ดชั่วคราวเปลี่ยนเป็นค่าลบ ระบบจะพิจารณาเงื่อนไขที่ตอนเริ่มต้นของการวนลูป และการวนลูปจะหยุดลง
- ห้ามใช้อุปกรณ์ PLC ในสูตรการวนลูป ให้ใช้ตำแหน่งพื้นที่สำหรับผู้ใช้ของพื้นที่ LS ภายในของจอแสดงผล หรือตำแหน่งเวิร์ดชั่วคราวแทน ตัวอย่างเช่น คำอธิบายต่อไปนี้เขียนข้อมูลลงใน PLC หลายครั้งในช่วงเวลาสั้น ๆ (100 ครั้งในตัวอย่างต่อไปนี้) ซึ่งอาจทำให้ระบบเกิดข้อผิดพลาดได้ เนื่องจากไม่สามารถประมวลผลการสื่อสาร (เวลาที่ต้องใช้ในการเขียนข้อมูลลงใน PLC) ด้วยความเร็วระดับนี้ได้
  ตัวอย่าง)

```
[t:0000] = 100                                          // จำนวนลูป:
loop ([t:0000])
{
  [w:[PLC1]D0200] = [w:[#INTERNAL]LS0100]               // เขียนลงใน D0200
  [w:[#INTERNAL]LS0100] =
  [w:[#INTERNAL]LS0100] + 1                             // เพิ่มขึ้น LS0100
}endloop
```

โปรดเปลี่ยนให้เป็นดังนี้

```
[t:0000] = 100                                          // จำนวนลูป:
loop ([t:0000])
{
  [w:[#INTERNAL]LS0200] =
  [w:[#INTERNAL]LS0100]                                 // เขียนลงใน D0200
  [w:[#INTERNAL]LS0100] =
  [w:[#INTERNAL]LS0100] + 1                             // เพิ่มขึ้น LS0100
}endloop
[w:[PLC1]D0200]=[w:[#INTERNAL]LS0200]                   //รายละเอียด LS0200,
                                                        เขียนลงใน D0200
```

- การใช้ “loop” หรือ “break” เป็นชื่อฟังก์ชันสำหรับฟังก์ชัน D-Script จะทำให้เกิดข้อผิดพลาด

21.8.4 break

หยุดการวนลูปขณะกำลังเรียกใช้สมการ loop ( )

หมายเหตุ

- สามารถใช้คำสั่ง “break” ได้เฉพาะในส่วน { } ของ loop ( ) เท่านั้น

21.8.5 return

เมื่อ “ฟังก์ชันที่กำหนดโดยผู้ใช้” มีคำสั่ง “return” อยู่ด้วย
การประมวลผลฟังก์ชันสิ้นสุดลงและการควบคุมกลับไปที่ตัวเรียกของฟังก์ชัน

เมื่อการดำเนินงาน (ฟังก์ชันหลัก) มีคำสั่ง “return” อยู่ด้วย
การประมวลผลฟังก์ชันหลักถูกยกเลิกครู่หนึ่ง และเริ่มต้นใหม่จากตอนเริ่มต้นของฟังก์ชันหลัก

หมายเหตุ

- อักขระ “=” ในคำสั่ง Assign ใช้ไม่ได้ในนิพจน์เงื่อนไข

ตัวอย่างนิพจน์

```
[w:[#INTERNAL]LS0100]=([w:[#INTERNAL]LS0200]>> 8) & 0xFF
if ([w:[#INTERNAL]LS0100]==0)        // เมื่อ LS0100 มีค่าเป็น “0” จะไม่มีการประมวลผลอีกต่อไป
{
    set([b:[#INTERNAL]LS005000])     // กำหนดตำแหน่งบิตสำหรับแสดงข้อผิดพลาด
    return                           //จบ
}
endif
```

21.9 การเปรียบเทียบ

การเปรียบเทียบ	ข้อมูลสรุปของฟังก์ชัน
Comparison Logical AND (AND) Logical OR (OR) Negation (not) less than (<) less than or equal to (<=) not equal to (<>) more than (>) more than or equal to (>=) Equivalent (==)	Logical AND (AND) ☞ "21.9.1 Logical AND (AND)" (หน้า 21-75) N1 and N2: เป็นจริงหาก N1 และ N2 เปิดทั้งคู่
	Logical OR (OR) ☞ "21.9.2 Logical OR (OR)" (หน้า 21-75) N1 or N2: เป็นจริงหาก N1 หรือ N2 ตัวใดตัวหนึ่งเปิด
	Negation (not) ☞ "21.9.3 Negation (not)" (หน้า 21-75) not N1: กลายเป็น 0 หาก N1 เป็น 1 และกลายเป็น 1 หาก N1 เป็น 0
	Less than (<) ☞ "21.9.4 Less than (<)" (หน้า 21-75) เป็นจริงหาก N1 น้อยกว่า N2 (N1 < N2)
	Less than or equal to (=) ☞ "21.9.5 Less than or equal to (<=)" (หน้า 21-76) เป็นจริงหาก N1 น้อยกว่าหรือเท่ากับ N2 (N1 <= N2)
	Not equal to (<>) ☞ "21.9.6 Not equal to (<>)" (หน้า 21-76) เป็นจริงหาก N1 ไม่เท่ากับ N2 (N1 <> N2)
	Greater than (>) ☞ "21.9.7 Greater than (>)" (หน้า 21-76) เป็นจริงหาก N1 มากกว่า N2 (N1 > N2)
	Greater than or equal to (>=) ☞ "21.9.8 Greater than or equal to (>=)" (หน้า 21-76) เป็นจริงหาก N1 น้อยกว่าหรือเท่ากับ N2 (N1 >= N2)
	Equivalent (==) ☞ "21.9.9 Equal to (==)" (หน้า 21-76) เป็นจริงหาก N1 เท่ากับ N2 (N1 == N2)

21.9.1 Logical AND (AND)

เชื่อมค่าทางขวากับทางซ้ายด้วยเงื่อนไข AND จะถือว่าค่า 0 (ศูนย์) คือปิด และค่าอื่นคือเปิด
N1 and N2: เป็นจริงหาก N1 และ N2 เปิดทั้งคู่ หากไม่ใช่ จะเป็นเท็จ

21.9.2 Logical OR (OR)

เชื่อมค่าทางขวาและทางซ้ายด้วยเงื่อนไข OR จะถือว่าค่า 0 (ศูนย์) คือปิด และค่าอื่นคือเปิด
N1 or N2: เป็นจริงหาก N1 หรือ N2 ตัวใดตัวหนึ่งเปิด หากไม่ใช่ จะเป็นเท็จ

21.9.3 Negation (not)

สลับค่า ค่า 0 (ศูนย์) คือ 1 และค่าอื่นคือ 0
not N1: กลายเป็น 0 หาก N1 เป็น 1 และกลายเป็น 1 หาก N1 เป็น 0

21.9.4 Less than (<)

เปรียบเทียบข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือข้อมูลในตำแหน่งเวิร์ดกับค่าคงที่หนึ่งค่า
เป็นจริงหาก N1 น้อยกว่า N2 (N1 < N2)

21.9.5 Less than or equal to (<=)

เปรียบเทียบข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือข้อมูลในตำแหน่งเวิร์ดกับค่าคงที่หนึ่งค่า เป็นจริงหาก N1 น้อยกว่าหรือเท่ากับ N2 (N1 <= N2)

21.9.6 Not equal to (<>)

เปรียบเทียบข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือข้อมูลในตำแหน่งเวิร์ดกับค่าคงที่หนึ่งค่า เป็นจริงหาก N1 ไม่เท่ากับ N2 (N1 <> N2)

21.9.7 Greater than (>)

เปรียบเทียบข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือข้อมูลในตำแหน่งเวิร์ดกับค่าคงที่หนึ่งค่า เป็นจริงหาก N1 มากกว่า N2 (N1 > N2)

21.9.8 Greater than or equal to (>=)

เปรียบเทียบข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือข้อมูลในตำแหน่งเวิร์ดกับค่าคงที่หนึ่งค่า เป็นจริงหาก N1 น้อยกว่าหรือเท่ากับ N2 (N1 >= N2)

21.9.9 Equal to (==)

เปรียบเทียบข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือข้อมูลในตำแหน่งเวิร์ดกับค่าคงที่หนึ่งค่า เป็นจริงหาก N1 เท่ากับ N2 (N1 == N2)

คำสั่ง		ตัวอย่าง
การเชื่อม	and	if ((การทำงาน) and (การทำงาน))
การเลือก	or	if ((การทำงาน) or (การทำงาน))
นิเสธ	not	if (not (การทำงาน))
น้อยกว่า	<	(ค่า 1) < (ค่า 2)
น้อยกว่าหรือเท่ากับ	<=	(ค่า 1) <= (ค่า 2)
ไม่เท่ากับ	<>	(ค่า 1) <> (ค่า 2)
มากกว่า	>	(ค่า 1) > (ค่า 2)
มากกว่าหรือเท่ากับ	>=	(ค่า 1) >= (ค่า 2)
เท่ากับ	==	(ค่า 1) == (ค่า 2)

21.10 ตัวดำเนินการ

ตัวดำเนินการ	ข้อมูลสรุปของฟังก์ชัน
Operator Addition (+) Subtraction (-) Margin (%) Multiplication (*) Division (/) Assignment (=) Left Shift (<<) Right Shift (>>) Bit Operator Logical AND (&) Bit Operator Logical OR (|) Bit Operator Exclusive OR (^) Bit Operator 1's Complement (~)	Addition (+) ☞ **"**21.10.1 Addition (+)**"** (หน้า 21-78) **เพิ่มข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือเพิ่มข้อมูลในตำแหน่งเวิร์ดหนึ่งตำแหน่งและค่าคงที่หนึ่งค่า**
	Subtraction (-) ☞ **"**21.10.2 Subtraction (-)**"** (หน้า 21-78) **ลบข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือลบข้อมูลในตำแหน่งเวิร์ดหนึ่งตำแหน่งและค่าคงที่หนึ่งค่า**
	Modulus (%) ☞ **"**21.10.3 Modulus (%)**"** (หน้า 21-78) **ตรวจหาเศษที่เหลือจากการหารข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือเศษที่เหลือจากการหารข้อมูลในตำแหน่งเวิร์ดหนึ่งตำแหน่งและค่าคงที่หนึ่งค่า**
	Multiplication (*) ☞ **"**21.10.4 Multiplication (*)**"** (หน้า 21-78) **คูณข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือคูณข้อมูลในตำแหน่งเวิร์ดหนึ่งตำแหน่งและค่าคงที่หนึ่งค่า**
	Division (/) ☞ **"**21.10.5 Division (/)**"** (หน้า 21-78) **หารข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือหารข้อมูลในตำแหน่งเวิร์ดหนึ่งตำแหน่งและค่าคงที่หนึ่งค่า**
	Assignment (=) ☞ **"**21.10.6 Assignment (=)**"** (หน้า 21-78) **กำหนดค่าฝั่งขวาให้แก่ค่าฝั่งซ้าย**
	Left Shift (<<) ☞ **"**21.10.7 Shift Left (<<)**"** (หน้า 21-78) **เลื่อนข้อมูลฝั่งซ้ายไปทางซ้ายตามจำนวนที่อยู่ในฝั่งขวา**
	Right Shift (>>) ☞ **"**21.10.8 Right Shift (>>)**"** (หน้า 21-79) **เลื่อนข้อมูลฝั่งซ้ายไปทางขวาตามจำนวนที่อยู่ในฝั่งขวา**
	Bit Operator Logical AND (&) ☞ **"**21.10.9 Bitwise AND (&)**"** (หน้า 21-79) **ใช้ตัวดำเนินการ logical AND กับข้อมูลระหว่างอุปกรณ์ชนิดเวิร์ดสองอุปกรณ์หรือระหว่างข้อมูลอุปกรณ์ชนิดเวิร์ดและค่าคงที่**
	Bit Operator Logical OR (|) ☞ **"**21.10.10 Bitwise OR (|)**"** (หน้า 21-79) **ใช้ตัวดำเนินการ logical OR กับข้อมูลระหว่างอุปกรณ์ชนิดเวิร์ดสองอุปกรณ์หรือระหว่างข้อมูลอุปกรณ์ชนิดเวิร์ดและค่าคงที่**
	Bit Operator Exclusive OR (^) ☞ **"**21.10.11 Bitwise Exclusive OR (^)**"** (หน้า 21-79) **ใช้ตัวดำเนินการ exclusive OR กับข้อมูลระหว่างอุปกรณ์ชนิดเวิร์ดสองอุปกรณ์หรือระหว่างข้อมูลอุปกรณ์ชนิดเวิร์ดและค่าคงที่**
	Bit Operator 1's Complement (~) ☞ **"**21.10.12 Bitwise 1's Complement (~)**"** (หน้า 21-79) **สลับบิตเป็นตรงกันข้าม**

21.10.1 Addition (+)

เพิ่มข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือเพิ่มข้อมูลในตำแหน่งเวิร์ดหนึ่งตำแหน่งและค่าคงที่หนึ่งค่า Overflowing Digit ใด ๆ ที่เป็นผลจากการคำนวณจะถูกปัดเศษ

21.10.2 Subtraction (-)

ลบข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือลบข้อมูลในตำแหน่งเวิร์ดหนึ่งตำแหน่งและค่าคงที่หนึ่งค่า Overflowing Digit ใด ๆ ที่เป็นผลจากการคำนวณจะถูกปัดเศษ

21.10.3 Modulus (%)

ตรวจหาเศษที่เหลือจากการหารข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือเศษที่เหลือจากการหารข้อมูลในตำแหน่งเวิร์ดหนึ่งตำแหน่งและค่าคงที่หนึ่งค่า ผลการคำนวณอาจขึ้นอยู่กับสัญลักษณ์ทางฝั่งซ้ายและฝั่งขวา

21.10.4 Multiplication (*)

คูณข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือคูณข้อมูลในตำแหน่งเวิร์ดหนึ่งตำแหน่งและค่าคงที่หนึ่งค่า Overflowing Digit ใด ๆ ที่เป็นผลจากการคำนวณจะถูกปัดเศษ

21.10.5 Division (/)

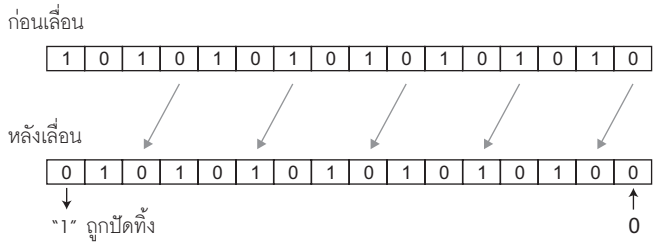
หารข้อมูลในตำแหน่งเวิร์ดสองตำแหน่ง หรือหารข้อมูลในตำแหน่งเวิร์ดหนึ่งตำแหน่งและค่าคงที่หนึ่งค่า ตำแหน่งทศนิยมที่เป็นผลจากการคำนวณจะถูกปัดเศษ Overflowing Digit ใด ๆ ที่เป็นผลจากการคำนวณจะถูกปัดเศษ

21.10.6 Assignment (=)

กำหนดค่าฝั่งขวาให้แก่ค่าฝั่งซ้าย สามารถเขียนค่าฝั่งซ้ายลงในอุปกรณ์เท่านั้น สามารถเขียนค่าฝั่งขวาลงในอุปกรณ์และค่าคงที่ ถ้าผลลัพธ์การดำเนินงานเกิด overflow จะมีการปัดเศษทิ้ง

21.10.7 Shift Left (<<)

เลื่อนข้อมูลฝั่งซ้ายไปทางซ้ายตามจำนวนที่อยู่ในฝั่งขวา คุณสมบัตินี้สนับสนุนการเลื่อนเชิงตรรกะเท่านั้น เช่น การเลื่อนไปทางซ้าย (เลื่อนไปทางซ้ายทีละบิต)



21.10.8 Right Shift (>>)

เลื่อนข้อมูลฝั่งซ้ายไปทางขวาตามจำนวนที่อยู่ในฝั่งขวา คุณสมบัตินี้สนับสนุนการเลื่อนเชิงตรรกะเท่านั้น

21.10.9 Bitwise AND (&)

ใช้ตัวดำเนินการ logical AND กับข้อมูลระหว่างอุปกรณ์ชนิดเวิร์ดสองอุปกรณ์ หรือระหว่างข้อมูลอุปกรณ์ชนิดเวิร์ดและค่าคงที่ ใช้สำหรับดึงบิตที่ระบุหรือพรางสตริงที่ระบุของบิต

21.10.10 Bitwise OR (|)

ใช้ตัวดำเนินการ logical OR กับข้อมูลระหว่างอุปกรณ์ชนิดเวิร์ดสองอุปกรณ์ หรือระหว่างข้อมูลอุปกรณ์ชนิดเวิร์ดและค่าคงที่ ใช้เปิดบิตที่ระบุ

21.10.11 Bitwise Exclusive OR (^)

ใช้ตัวดำเนินการ exclusive OR กับข้อมูลระหว่างอุปกรณ์ชนิดเวิร์ดสองอุปกรณ์ หรือระหว่างข้อมูลอุปกรณ์ชนิดเวิร์ดและค่าคงที่

21.10.12 Bitwise 1's Complement (~)

สลับบิตเป็นตรงกันข้าม

หมายเหตุ

- สำหรับข้อมูลเกี่ยวกับการปัดเศษเลขทศนิยมหรือ overflowing digit ที่เกิดจากผลการคำนวณ โปรดดูที่

☞ "20.9.4 ข้อควรทราบเกี่ยวกับผลการทำงาน" (หน้า 20-66)

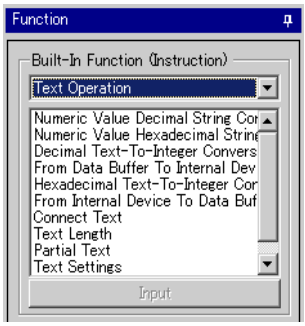
ลำดับความสำคัญและการเชื่อมโยง

ตารางต่อไปนี้แสดงลำดับความสำคัญของเงื่อนไขการทริกเกอร์ หากตัวดำเนินการสองตัวขึ้นไปมีลำดับความสำคัญเท่ากัน ให้ไปตามทิศทางที่แสดงไว้ในการเชื่อมโยง

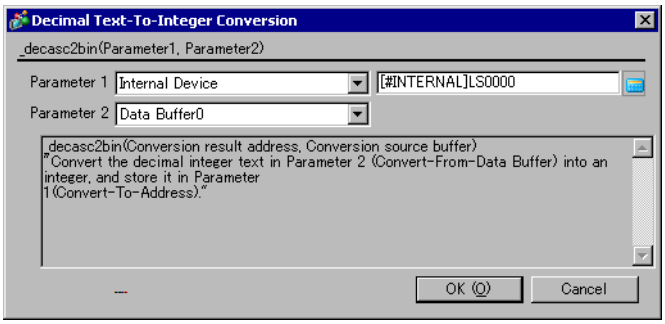
ระดับความสำคัญ	ตัวดำเนินการ	การเชื่อมโยง
สูง	( )	→
	not ~	←
	* / %	→
	+ –	→
	<< >>	→
	< <= > >=	→
	== <>	→
	& ^ |	→
	and or	→
ต่ำ	=	←

21.11 Text Operation

ฟังก์ชันการทำงานของข้อความสามารถใช้ได้เฉพาะใน Extended Script เท่านั้น

Text Operation	ข้อมูลสรุปของฟังก์ชัน
	Decimal Text-To-Integer Conversion ☞ "21.11.1 Decimal Text-To-Integer Conversion" (หน้า 21-81) **ฟังก์ชันนี้ใช้แปลงข้อความเลขฐานสิบให้เป็นจำนวนเต็ม**
	Hexadecimal Text-To-Integer Conversion ☞ "21.11.2 Hexadecimal Text-To-Integer Conversion" (หน้า 21-83) **ฟังก์ชันนี้ใช้แปลงข้อความเลขฐานสิบหกให้เป็นจำนวนเต็ม**
	From Internal Device To Data Buffer ☞ "21.11.3 From Internal Device To Data Buffer" (หน้า 21-85) **คัดลอกข้อมูลของสตริงที่จัดเก็บในอุปกรณ์ภายในไปยังบัฟเฟอร์ข้อมูล**
	From Data Buffer to Internal Device ☞ "21.11.4 From Data Buffer To Internal Device" (หน้า 21-87) **คัดลอกข้อมูลของสตริงที่จัดเก็บในบัฟเฟอร์ข้อมูลไปยังอุปกรณ์ภายใน**
	สถานะ ☞ "21.11.5 สถานะข้อผิดพลาดของฟังก์ชันการทำงานของข้อความ" (หน้า 21-89) **จัดเก็บข้อผิดพลาดต่าง ๆ ที่เกิดขึ้น**
	Numeric Value Decimal String Conversion ☞ "21.11.6 Numeric Value Decimal String Conversion" (หน้า 21-90) **ใช้ฟังก์ชันนี้เพื่อแปลงจำนวนเต็มให้เป็นสตริงเลขฐานสิบ**
	Numeric Value Decimal String Conversion ☞ "21.11.7 Numeric Value Decimal String Conversion" (หน้า 21-91) **ใช้ฟังก์ชันนี้เพื่อแปลงข้อมูลเลขฐานสองให้เป็นสตริงเลขฐานสิบหก**
	Partial Text ☞ "21.11.8 Partial Text" (หน้า 21-92) **ค้นข้อมูลตามความยาวของสตริงจากออฟเซ็ตที่ระบุไว้ของสตริง แล้วจัดเก็บข้อมูลไว้ในบัฟเฟอร์ข้อมูลอื่น**
	Text Settings ☞ "21.11.9 Text Settings" (หน้า 21-93) **จัดเก็บสตริงแบบตายตัวในบัฟเฟอร์ข้อมูล**
	Text Length ☞ "21.11.10 Text Length" (หน้า 21-94) **รับความยาวของสตริงที่จัดเก็บไว้**
	Connect Text ☞ "21.11.11 Connect Text" (หน้า 21-95) **เชื่อมสตริงอักขระหรือรหัสอักขระด้วยบัฟเฟอร์ข้อความ**

21.11.1 Decimal Text-To-Integer Conversion

รายการ	คำอธิบาย
ข้อมูลสรุป	ฟังก์ชันนี้ใช้แปลงสตริงเลขฐานสิบให้เป็นจำนวนเต็ม โดยแปลงข้อความจำนวนเต็มเลขฐานสิบใน Parameter 2 (บัฟเฟอร์ข้อมูลที่ถูกแปลง) ให้เป็นจำนวนเต็ม และจัดเก็บไว้ใน Parameter 1(ตำแหน่งที่แปลงแล้ว)
รูปแบบ	_decasc2bin ([ตำแหน่งที่แปลงแล้ว], [บัฟเฟอร์ข้อมูลที่ถูกแปลง])  Parameter 1: อุปกรณ์ภายใน, ตำแหน่งชั่วคราว Parameter 2: บัฟเฟอร์ข้อมูล

ตัวอย่างนิพจน์ 1 (เมื่อข้อมูลยาว 16 บิต)

_decasc2bin ([w:[#INTERNAL]LS0100], databuf0)

รายละเอียดของ “databuf0” มีดังนี้

	8 บิต	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	ค่าศูนย์

ข้อมูลข้างบนจะถูกแปลงดังต่อไปนี้

	16 บิต
LS0100	1234

ตัวอย่างนิพจน์ 2 (เมื่อข้อมูลยาว 32 บิต)

_decasc2bin ([w:[#INTERNAL]LS0100], databuf0)

รายละเอียดของ “databuf0” มีดังนี้

	8 บิต	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	35h	‘5’
databuf0[5]	36h	‘6’
databuf0[6]	37h	‘7’
databuf0[7]	38h	‘8’
databuf0[8]	00h	ค่าศูนย์

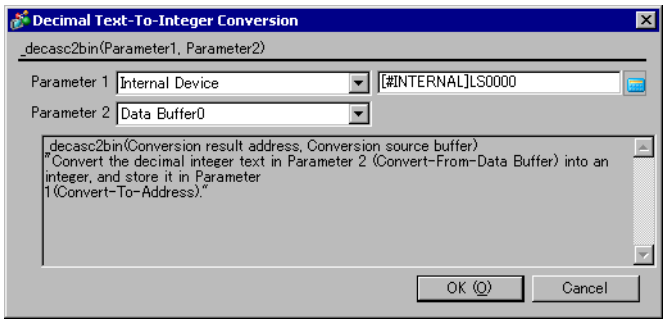
ข้อมูลข้างบนจะถูกแปลงดังต่อไปนี้

	32 บิต
LS0100	12345678
LS0102	

ข้อสำคัญ

- เมื่อความยาวบิตที่แปลงแล้วมากกว่าความยาวบิตของ D-Script Editor จะเกิดข้อผิดพลาดขึ้น
  ตัวอย่างเช่น เมื่อสคริปต์มีความยาวบิตเท่ากับ 16 บิต
  _strset (databuf 0, " 123456") // เมื่อป้อนสตริงเลขฐานสิบ 6 ตัวโดยไม่ตั้งใจ
  _decasc2bin ([w:[#INTERNAL]LS0100], databuf0)
  เมื่อเรียกใช้นิพจน์ข้างบน ข้อผิดพลาดหมายเลข 2 (ข้อผิดพลาดในการแปลงสตริง) ของสถานะข้อผิดพลาดของสตริง [e: STR_ERR_STAT] จะถูกทริกเกอร์ อย่างไรก็ตาม บิตจะกลับไปที่ตอนเริ่มต้นของฟังก์ชันหลักเมื่อเกิดข้อผิดพลาด คุณจึงไม่สามารถอ้างอิงถึงฟังก์ชันอื่นโดยตรงได้หลังจากเรียกใช้ _decasc2bin แล้ว (หากมีคำสั่งเข้ามาระหว่างฟังก์ชันกำลังทำงาน ระบบจะกลับไปที่บรรทัดที่เรียกฟังก์ชันนั้นขึ้นมา)
- การแปลงสตริงข้อมูลที่มีอักขระอื่นที่ไม่ใช่ “0” ถึง “9” อยู่จะเกิดข้อผิดพลาดขึ้น
  ตัวอย่างเช่น เมื่อสคริปต์มีความยาวบิตเท่ากับ 16 บิต:
  _strset (databuf0, "12AB") // เมื่อป้อนสตริงที่ไม่ใช่เลขฐานสิบโดยไม่ตั้งใจ
  _decasc2bin ([w:[#INTERNAL]LS0100], databuf0)
  เมื่อเรียกใช้นิพจน์ข้างบน ข้อผิดพลาดหมายเลข 2 (ข้อผิดพลาดในการแปลงสตริง) ของสถานะข้อผิดพลาดของสตริง [e: STR_ERR_STAT] จะถูกทริกเกอร์ อย่างไรก็ตาม บิตจะกลับไปที่ตอนเริ่มต้นของฟังก์ชันหลักเมื่อเกิดข้อผิดพลาด คุณจึงไม่สามารถอ้างอิงถึงฟังก์ชันอื่นโดยตรงได้หลังจากเรียกใช้ _decasc2bin แล้ว (หากมีคำสั่งเข้ามาระหว่างฟังก์ชันกำลังทำงาน ระบบจะกลับไปที่บรรทัดที่เรียกฟังก์ชันนั้นขึ้นมา)
- การประมวลผลจะสิ้นสุดลงเมื่อเกิดข้อผิดพลาดและกลับไปที่ตอนเริ่มต้นของฟังก์ชันหลัก (หากมีคำสั่งเข้ามาระหว่างฟังก์ชันกำลังทำงาน ระบบจะกลับไปที่บรรทัดที่เรียกฟังก์ชันนั้นขึ้นมา)

21.11.2 Hexadecimal Text-To-Integer Conversion

รายการ	คำอธิบาย
ข้อมูลสรุป	ฟังก์ชันนี้จะแปลงสตริงเลขฐานสิบหกให้เป็นข้อมูลเลขฐานสอง โดยแปลงข้อความจำนวนเต็มเลขฐานสิบหกใน Parameter 2 (บัฟเฟอร์ข้อมูลที่ถูกแปลง) ให้เป็นจำนวนเต็ม และจัดเก็บไว้ใน Parameter 1(ตำแหน่งที่แปลงแล้ว)
รูปแบบ	_hexasc2bin ([ตำแหน่งที่แปลงแล้ว], [บัฟเฟอร์ข้อมูลที่ถูกแปลง])  Parameter 1: อุปกรณ์ภายใน, ตำแหน่งชั่วคราว Parameter 2: บัฟเฟอร์ข้อมูล

ตัวอย่างนิพจน์ 1 (เมื่อข้อมูลยาว 16 บิต)

_hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)

รายละเอียดของ “databuf0” มีดังนี้

	8 บิต	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	ค่าศูนย์

ข้อมูลข้างบนจะถูกแปลงดังต่อไปนี

	16 บิต
LS0100	1234h

ตัวอย่างนิพจน์ 2 (เมื่อข้อมูลยาว 32 บิต)

_hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)

รายละเอียดของ “databuf0” มีดังนี้

	8 บิต	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	35h	‘5’
databuf0[5]	36h	‘6’
databuf0[6]	37h	‘7’
databuf0[7]	38h	‘8’
databuf0[8]	00h	ค่าศูนย์

ข้อมูลข้างบนจะถูกแปลงดังต่อไปนี้

	32 บิต
LS0100	12345678h
LS0102	

ข้อสำคัญ

- เมื่อสตริงที่แปลงมีขนาดใหญ่กว่า 16 บิตหรือ 32 บิต จะเกิดข้อผิดพลาดขึ้น
  ตัวอย่างเช่น เมื่อสคริปต์มีความยาวบิตเท่ากับ 16 บิต:

  _strset (databuf0, "123456")

  _hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)

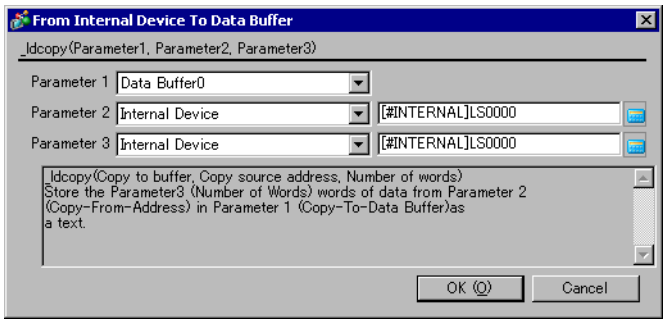
  เมื่อเรียกใช้นิพจน์ข้างบน ข้อผิดพลาดหมายเลข 2 (ข้อผิดพลาดในการแปลงสตริง) ของสถานะข้อผิดพลาดของสตริง [e: STR_ERR_STAT] จะถูกทริกเกอร์
- การแปลงสตริงข้อมูลที่มีอักขระอื่นที่ไม่ใช่ “0” ถึง “9”, “A” ถึง “F” หรือ “a” ถึง “f” จะเกิดข้อผิดพลาดขึ้น
  ตัวอย่างเช่น เมื่อสคริปต์มีความยาวบิตเท่ากับ 16 บิต:

  _strset (databuf 0, "123G")

  _hexasc2bin ([w:[#INTERNAL]LS0100], databuf0)

  เมื่อเรียกใช้นิพจน์ข้างบน ข้อผิดพลาดหมายเลข 2 (ข้อผิดพลาดในการแปลงสตริง) ของสถานะข้อผิดพลาดของสตริง [e: STR_ERR_STAT] จะถูกทริกเกอร์
- การประมวลผลจะสิ้นสุดลงเมื่อเกิดข้อผิดพลาดและกลับไปที่ตอนเริ่มต้นของฟังก์ชันหลัก (หากมีคำสั่งเข้ามาระหว่างฟังก์ชันกำลังทำงาน ระบบจะกลับไปที่บรรทัดที่เรียกฟังก์ชันนั้นขึ้นมา)

21.11.3 From Internal Device To Data Buffer

รายการ	คำอธิบาย
ข้อมูลสรุป	ข้อมูลของสตริงที่จัดเก็บอยู่ในพื้นที่ LS จะถูกคัดลอกไปยังบัฟเฟอร์ข้อมูลตามจำนวนสตริง ในการถ่ายโอนข้อมูลแบบไบต์ต่อไบต์ ฟังก์ชันนี้จะจัดเก็บข้อมูลเวิร์ดของ Parameter 3 (เวิร์ด) จาก Parameter 2 (ตำแหน่งที่ถูกคัดลอก) ใน Parameter 1 (บัฟเฟอร์ข้อมูลที่คัดลอกแล้ว) ให้เป็นข้อความ
รูปแบบ	_ldcopy (บัฟเฟอร์ข้อมูลปลายทางการคัดลอก, [ตำแหน่งที่ถูกคัดลอก], เวิร์ด)  Parameter 1: บัฟเฟอร์ข้อมูล Parameter 2: อุปกรณ์ภายใน Parameter 3: ค่าจำนวนเต็ม, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 3 คือตั้งแต่ 1 ถึง 1,024)

ตัวอย่างนิพจน์ 1

_ldcopy (databuf0, [w:[#INTERNAL]LS0100], 4)

	16 บิต
LS0100	31h
LS0101	32h
LS0102	33h
LS0103	34h

ข้อมูลในพื้นที่ LS0100 ถึง LS0103 จะถูกเขียนลงในบัฟเฟอร์ข้อมูล 4 ไบต์ตามลำดับโดยเริ่มจาก “databuf0” พื้นที่ LS ถูกอ่านทีละไบต์ (บิตต่ำที่สุด)

	8 บิต	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	ค่าศูนย์

ข้อสำคัญ

- ระบบจะอ่านไบต์ล่าง 1 ไบต์ของพื้นที่ LS และเขียนข้อมูลตามจำนวนที่ระบุลงในบัฟเฟอร์ข้อมูล
- ค่าตัวเลขสูงสุดที่สามารถกำหนดให้สำหรับ Parameter 3 คือ 1,024 เมื่อมีการตั้งค่าที่เกินกว่าที่กำหนดไว้ ข้อผิดพลาดหมายเลข 1 (สตริงโอเวอร์โฟลว์) ของสถานะข้อผิดพลาดของสตริง [e: STR_ERR_STAT] จะถูกทริกเกอร์
- แม้จะจัดเก็บข้อมูลไว้ในไบต์สูงสุดในพื้นที่ LS แต่ระบบจะอ่านเฉพาะข้อมูลในไบต์ล่าง 1 ไบต์เท่านั้น
- การประมวลผลจะสิ้นสุดลงเมื่อเกิดข้อผิดพลาดและกลับไปที่ตอนเริ่มต้นของฟังก์ชันหลัก (หากมีคำสั่งเข้ามาระหว่างฟังก์ชันกำลังทำงาน ระบบจะกลับไปที่บรรทัดที่เรียกฟังก์ชันนั้นขึ้นมา)

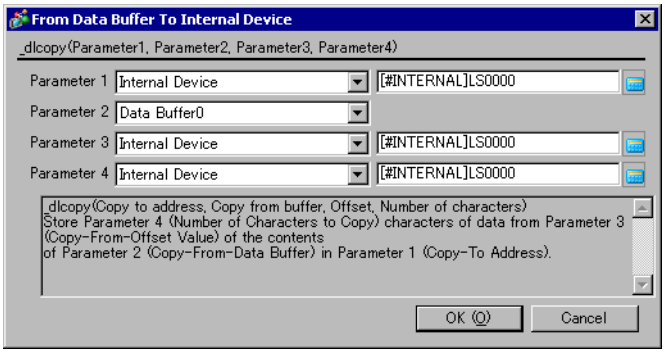
```
_ldcopy (databuf0, [w:[#INTERNAL]LS0100], 4)
```

	16 บิต
LS0100	3132h
LS0101	3334h
LS0102	3536h
LS0103	3738h

เมื่อจัดเก็บข้อมูลไว้ตามที่แสดงข้างบนแล้ว ระบบจะอ่านข้อมูลของไบต์ล่าง 1 ไบต์ และเขียนลงในบัฟเฟอร์ข้อมูล

	8 บิต	
databuf0[0]	32h	'2'
databuf0[1]	34h	'4'
databuf0[2]	36h	'6'
databuf0[3]	38h	'8'
databuf0[4]	00h	ค่าศูนย์

21.11.4 From Data Buffer To Internal Device

รายการ	คำอธิบาย
ข้อมูลสรุป	ฟังก์ชันนี้จะคัดลอกข้อมูลสตริงแต่ละไบต์ที่เก็บไว้ในออฟเซ็ตของบัฟเฟอร์ข้อมูลไปยังพื้นที่ LS ตามจำนวนสตริง ฟังก์ชันจะจัดเก็บข้อมูลตามจำนวนอักขระของ Parameter 4 (จำนวนอักขระที่จะคัดลอก) จาก Parameter 3 (ค่าออฟเซ็ตที่ถูกคัดลอก) ของรายละเอียดของ Parameter 2 (บัฟเฟอร์ข้อมูลที่ถูกคัดลอก) ไว้ใน Parameter 1 (ตำแหน่งปลายทางการคัดลอก)
รูปแบบ	_dlcopy ([ตำแหน่งปลายทางการคัดลอก], บัฟเฟอร์ข้อมูลที่ถูกคัดลอก, ค่าออฟเซ็ตที่ถูกคัดลอก, จำนวนอักขระที่คัดลอก)  Parameter 1: อุปกรณ์ภายใน Parameter 2: บัฟเฟอร์ข้อมูล Parameter 3: ค่าตัวเลข, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 3 คือตั้งแต่ 1 ถึง 1,024) Parameter 4: ค่าตัวเลข, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 4 คือตั้งแต่ 1 ถึง 1,024)

ตัวอย่างนิพจน์ 1

_dlcopy ([w:[#INTERNAL]LS0100], databuf0, 2, 4)

	8 บิต	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	35h	'5'
databuf0[5]	36h	'6'
databuf0[6]	37h	'7'
databuf0[7]	38h	'8'

ระบบจะเขียนข้อมูลจำนวน 4 ไบต์ที่ค้นจาก "offset 2" ของ "databuf0" ลงในพื้นที่ LS0100 ถึง LS0103 โดยเขียนข้อมูลลงในพื้นที่ LS ทีละ 1 ไบต์

	16 บิต
LS0100	33h
LS0101	34h
LS0102	35h
LS0103	36h

ข้อสำคัญ

- ระบบจะอ่านข้อมูล 1 ไบต์จากบัฟเฟอร์ข้อมูลและเขียนลงในพื้นที่ LS ซึ่งหมายความว่า จะใช้เฉพาะ 8 บิตต่ำสุด (1 ไบต์) ของพื้นที่ LS เท่านั้น ข้อมูล 8 บิตสูงสุด (1 ไบต์) จะถูกล้างด้วยค่า “0”
- เมื่อค่าที่ระบุ [ค่าออฟเซ็ตต้นทาง + จำนวนอักขระที่จะคัดลอก] มากกว่าขนาดบัฟเฟอร์ข้อมูล ระบบจะแสดงข้อผิดพลาดหมายเลข 3 (ข้อผิดพลาดในการดึงสตริง) ของสถานะข้อผิดพลาดของสตริง [e: STR_ERR_STAT]
- การประมวลผลจะสิ้นสุดลงเมื่อเกิดข้อผิดพลาดและกลับไปที่ตอนเริ่มต้นของฟังก์ชันหลัก (หากมีคำสั่งเข้ามาระหว่างฟังก์ชันกำลังทำงาน ระบบจะกลับไปที่บรรทัดที่เรียกฟังก์ชันนั้นขึ้นมา)

21.11.5 สถานะข้อผิดพลาดของฟังก์ชันการทำงานของข้อความ

เมื่อเกิดข้อผิดพลาดระหว่างเรียกใช้ฟังก์ชันการทำงานของข้อความ ระบบจะกำหนดข้อผิดพลาดให้กับสถานะข้อผิดพลาดของฟังก์ชันการทำงานของข้อความ [e: STR_ERR_STAT] หาก [e: STR_ERR_STAT] มีค่า “0” แสดงว่าฟังก์ชันทำงานได้เป็นปกติ แต่หากมีค่าอื่นที่ไม่ใช่ “0” แสดงว่ามีข้อผิดพลาดเกิดขึ้น ข้อผิดพลาดล่าสุดจะเก็บไว้ในสถานะข้อผิดพลาดของฟังก์ชันการทำงานของข้อความ [e: STR_ERR_STAT] คุณสามารถตั้งค่าสถานะข้อผิดพลาดของฟังก์ชันการทำงานของข้อความโดยใช้ฟังก์ชัน [SIO Port Operation/ Label Settings] ที่อยู่ใต้เมนูกล่องเครื่องมือ D-Script ตารางต่อไปนี้แสดงรายการข้อผิดพลาดของฟังก์ชันการทำงานของข้อความ

หมายเลขข้อผิดพลาด	ชื่อข้อผิดพลาด	คำอธิบาย
0	ปกติ	ไม่มีข้อผิดพลาด
1	ข้อความโอเวอร์โฟลว์	มีการป้อนสตริงขนาดตั้งแต่ 256 ไบต์ขึ้นไปไว้ในอาร์กิวเมนต์ของฟังก์ชันต่อไปนี้โดยตรง ได้แก่ _strset ( ), _strlen ( ), _strcat ( ), _strmid ( ) และ IO_READ_WAIT ( ) หรือสร้างสตริงที่เกินกว่าขนาดบัฟเฟอร์ข้อมูลในระหว่างเรียกใช้ฟังก์ชัน _strcat ( ) หรือ _ldcopy ( ) ตัวอย่าง: _strcat (databuf0, databuf1) ระบบจะเรียกใช้ฟังก์ชันข้างบนเมื่อจัดเก็บสตริงขนาด 1,020 ไบต์ใน databuf0 และจัดเก็บสตริงขนาด 60 ไบต์ใน databuf1 (สตริงที่เกินกว่าบัฟเฟอร์ข้อมูลซึ่งมีขนาดเท่ากับ 1,024 ไบต์จะทำให้เกิดสถานะข้อผิดพลาด)
2	ข้อผิดพลาดในการแปลงสตริง	มีการป้อนรหัสอักขระที่ใช้ไม่ได้ไว้ในฟังก์ชัน _hexasc2bin ( ) หรือ _decasc2bin ( ) ตัวอย่าง: มีการใส่รหัสอักขระอื่นที่ไม่ใช่ “0” ถึง “9”, “A” ถึง “F” หรือ “a” ถึง “f” ไว้ในอาร์กิวเมนต์ที่สองของ _hexasc2bin ( )
3	ข้อผิดพลาดในการค้นสตริง	มีการค้นหาสตริงอักขระที่ยาวกว่าที่ระบุไว้ด้วยฟังก์ชัน “_strmid ( )” หรือกำหนดค่าออฟเซ็ตมากกว่าสตริงที่ระบุไว้ ตัวอย่าง: _strmid (databuf0, ″12345678″, 2, 8) ค้นหาสตริง 8 อักขระจากออฟเซ็ต 2

สถานะข้อผิดพลาดในการควบคุมสตริงใช้กับ D-Scripts และ Global D-Scripts ไม่ได้ หากมีการอ่านสถานะนั้นโดยไม่ตั้งใจ ระบบจะโหลดค่า “0”
และจัดเก็บไว้ในสถานะข้อผิดพลาดระหว่างเรียกใช้ฟังก์ชันแต่ละฟังก์ชัน
หากต้องการตรวจสอบข้อผิดพลาด [e: STR_ERR_STAT] ให้เขียนข้อความคำสั่งต่อไปนี้ คุณสามารถยืนยันข้อผิดพลาดด้วยนิพจน์ต่อไปนี้ได้

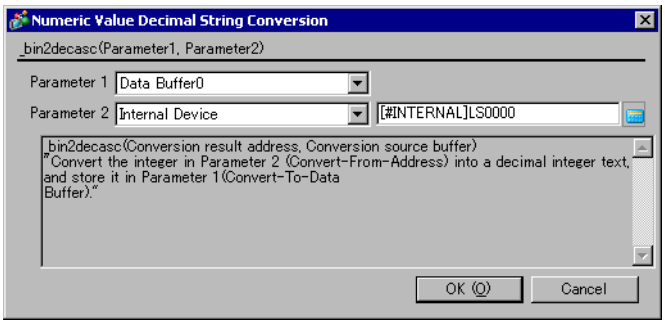
ตัวอย่างนิพจน์

```
if ([e:STR_ERR_STAT] <> 0)              // ตรวจสอบสถานะข้อผิดพลาด
{
    set([b:[#INTERNAL]LS005000]) // กำหนดบิตของไฟสัญญาณแสดงข้อผิดพลาด
}
endif
```

ข้อสำคัญ

- การประมวลผลจะสิ้นสุดลงเมื่อเกิดข้อผิดพลาดและกลับไปที่ตอนเริ่มต้นของฟังก์ชันหลัก (หากมีคำสั่งเข้ามาระหว่างฟังก์ชันกำลังทำงาน ระบบจะกลับไปที่บรรทัดที่เรียกฟังก์ชันนั้นขึ้นมา)

21.11.6 Numeric Value Decimal String Conversion

รายการ	คำอธิบาย
ข้อมูลสรุป	ใช้ฟังก์ชันนี้เพื่อแปลงจำนวนเต็มให้เป็นสตริงเลขฐานสิบ โดยฟังก์ชันจะแปลงจำนวนเต็มใน Parameter 2 (ตำแหน่งที่ถูกแปลง) ให้เป็นข้อความจำนวนเต็มเลขฐานสิบ และเก็บไว้ใน Parameter 1(บัฟเฟอร์ข้อมูลที่แปลงแล้ว)
รูปแบบ	_bin2decasc (บัฟเฟอร์ข้อมูลที่แปลงแล้ว, [ตำแหน่งที่ถูกแปลง])  Parameter 1: บัฟเฟอร์ข้อมูล Parameter 2: อุปกรณ์ภายใน, ตำแหน่งชั่วคราว

ตัวอย่างนิพจน์ 1 (เมื่อข้อมูลยาว 16 บิต)

```
_bin2decasc (databuf0, [w:[#INTERNAL]LS0100])
```

	16 บิต
LS0100	1234

ข้อมูลข้างบนจะถูกแปลงดังต่อไปนี้ โปรดสังเกตว่าระบบจะเพิ่ม “ค่าศูนย์ (0x00)” เข้าไป

	8 บิต	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	ค่าศูนย์

ตัวอย่างนิพจน์ 2 (เมื่อข้อมูลยาว 32 บิต)

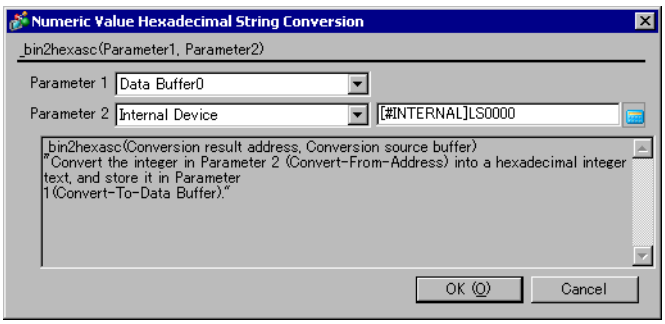
_bin2decasc (databuf0, [w:[#INTERNAL]LS0100])

	32 บิต
LS0100	12345678
LS0102	

ข้อมูลข้างบนจะถูกแปลงดังต่อไปนี้

	8 บิต	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	35h	'5'
databuf0[5]	36h	'6'
databuf0[6]	37h	'7'
databuf0[7]	38h	'8'
databuf0[8]	00h	ค่าศูนย์

21.11.7 Numeric Value Decimal String Conversion

รายการ	คำอธิบาย
ข้อมูลสรุป	ใช้ฟังก์ชันนี้เพื่อแปลงข้อมูลเลขฐานสองให้เป็นสตริงเลขฐานสิบหก โดยฟังก์ชันจะแปลงจำนวนเต็มใน Parameter 2 (ตำแหน่งที่ถูกแปลง) ให้เป็นข้อความจำนวนเต็มเลขฐานสิบหก และเก็บไว้ใน Parameter 1(บัฟเฟอร์ข้อมูลที่แปลงแล้ว)
รูปแบบ	_bin2hexasc (บัฟเฟอร์ข้อมูลที่แปลงแล้ว, [ตำแหน่งที่ถูกแปลง])  Parameter 1: บัฟเฟอร์ข้อมูล Parameter 2: อุปกรณ์ภายใน, ตำแหน่งชั่วคราว

ตัวอย่างนิพจน์ 1 (เมื่อข้อมูลยาว 16 บิต)

_bin2hexasc (databuf0, [w:[#INTERNAL]LS0100])

	16 บิต
LS0100	1234h

ข้อมูลข้างบนจะถูกแปลงดังต่อไปนี้ โปรดสังเกตว่าระบบจะเพิ่ม "ค่าศูนย์ (0x00)" เข้าไป

	8 บิต	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	00h	ค่าศูนย์

ตัวอย่างนิพจน์ 2 (เมื่อข้อมูลยาว 32 บิต)

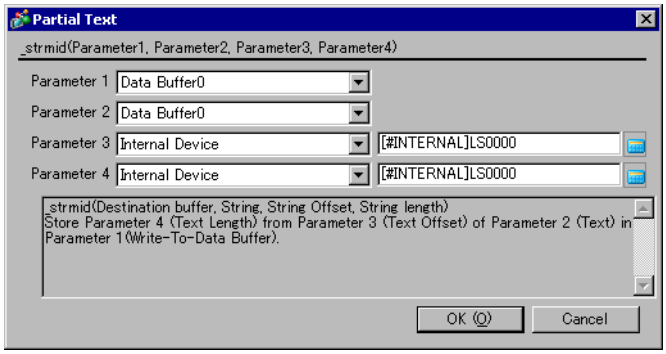
_bin2hexasc (databuf0, [w:[#INTERNAL]LS0100])

	32 บิต
LS0100	12345678h
LS0102	

ข้อมูลข้างบนจะถูกแปลงดังต่อไปนี้

	8 บิต	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	35h	'5'
databuf0[5]	36h	'6'
databuf0[6]	37h	'7'
databuf0[7]	38h	'8'
databuf0[8]	00h	ค่าศูนย์

21.11.8 Partial Text

รายการ	คำอธิบาย
ข้อมูลสรุป	ค้นข้อมูลตามความยาวของสตริงจากออฟเซ็ตที่ระบุไว้ของสตริง แล้วจัดเก็บข้อมูลไว้ในบัฟเฟอร์ข้อมูลอื่น ฟังก์ชันจะจัดเก็บ Parameter 4 (ความยาวข้อความ) จาก Parameter 3 (ออฟเซ็ตข้อความ) ของ Parameter 2 (ข้อความ) ไว้ใน Parameter 1 (บัฟเฟอร์ข้อมูลปลายทางการเขียน)
รูปแบบ	_strmid (บัฟเฟอร์ข้อมูลปลายทางการเขียน, ข้อความ, ออฟเซ็ตข้อความ, ความยาวข้อความ)  Parameter 1: บัฟเฟอร์ข้อมูล Parameter 2: สตริง, บัฟเฟอร์ข้อมูล Parameter 3: ค่าตัวเลข, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 3 คือตั้งแต่ 1 ถึง 1,024) Parameter 4: ค่าตัวเลข, อุปกรณ์ภายใน, ตำแหน่งชั่วคราว (ช่วงที่ใช้ได้สำหรับ Parameter 4 คือตั้งแต่ 1 ถึง 1,024)

ตัวอย่างนิพจน์

_strmid (databuf0, "12345678", 2, 4)

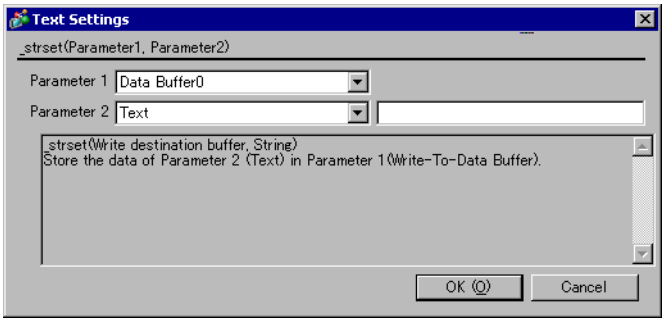
ระบบจะจัดเก็บข้อมูล 4 ไบต์ที่ค้นมาจากออฟเซ็ต 2 ของสตริง “12345678” ไว้ใน “databuf0”

	8 บิต	
databuf0[0]	33h	'3'
databuf0[1]	34h	'4'
databuf0[2]	35h	'5'
databuf0[3]	36h	'6'
databuf0[4]	00h	ค่าศูนย์

ข้อสำคัญ

- เมื่อพยายามจะค้นสตริงที่ยาวกว่าสตริงที่ฟังก์ชัน “strmid ( )” ระบุไว้ หรือเมื่อระบุค่าออฟเซ็ตที่มากกว่าสตริงที่ระบุ ระบบจะแสดงข้อผิดพลาดหมายเลข 3 (ข้อผิดพลาดในการดึงสตริง) ของสถานะข้อผิดพลาดของสตริง [e: STR_ERR_STAT]
- การประมวลผลจะสิ้นสุดลงเมื่อเกิดข้อผิดพลาดและกลับไปที่ตอนเริ่มต้นของฟังก์ชันหลัก (หากมีคำสั่งเข้ามาระหว่างฟังก์ชันกำลังทำงาน ระบบจะกลับไปที่บรรทัดที่เรียกฟังก์ชันนั้นขึ้นมา)

21.11.9 Text Settings

รายการ	คำอธิบาย
ข้อมูลสรุป	**จัดเก็บสตริงแบบตายตัวไว้ในบัฟเฟอร์ข้อมูล ฟังก์ชันนี้จะจัดเก็บข้อมูลของ Parameter 2 (ข้อความ) ไว้ใน Parameter 1(บัฟเฟอร์ข้อมูลปลายทางการเขียน)**
รูปแบบ	_strset (**บัฟเฟอร์ข้อมูลปลายทางการเขียน, ข้อความ**)  Parameter 1: **บัฟเฟอร์ข้อมูล** Parameter 2: **ข้อความ, ค่าตัวเลข (รหัสตัวอักษร) (ช่วงที่ใช้ได้สำหรับ Parameter 2 คือ 0 และตั้งแต่ 1 ถึง 255)**

ตัวอย่างนิพจน์

_strset (databuf0, "ABCD")

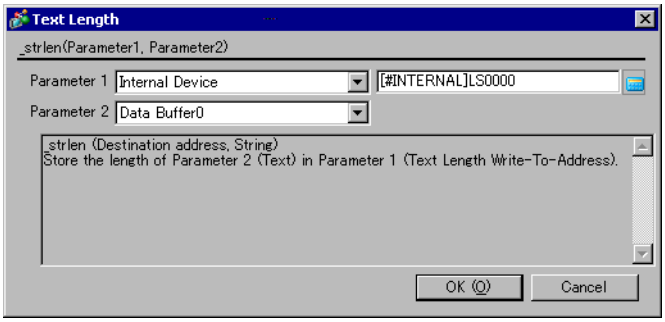
สตริงถูกจัดเก็บไว้ในบัฟเฟอร์ข้อมูลตามที่แสดงไว้ด้านล่าง

	8 บิต	
databuf0[0]	41h	‘A’
databuf0[1]	42h	‘B’
databuf0[2]	43h	‘C’
databuf0[3]	44h	‘D’
databuf0[4]	00h	ค่าศูนย์

ข้อสำคัญ

- สามารถกำหนดสตริงได้สูงสุด 255 อักขระ หากต้องการสร้างสตริงที่ยาวกว่านี้ ให้จัดเก็บสตริงไว้ในบัฟเฟอร์อื่นและเชื่อมสตริงด้วยฟังก์ชันการเชื่อมสตริง (_strcat)
- หากต้องการล้างบัฟเฟอร์ข้อมูล ให้สร้างสตริงเปล่า “” ตัวอย่าง)_strset (databuf0,””)_strset (databuf0,0)

21.11.10 Text Length

รายการ	คำอธิบาย
ข้อมูลสรุป	รับความยาวของสตริงที่จัดเก็บไว้ ฟังก์ชันนี้จะจัดเก็บความยาวของ Parameter 2 (ข้อความ) ไว้ใน Parameter 1 (ตำแหน่งที่จะเขียนความยาวข้อความ) (ไม่รวมอักขระค่าศูนย์)
รูปแบบ	_strlen (ตำแหน่งที่จะเขียนความยาวข้อความ, ข้อความ)  Parameter 1: อุปกรณ์ภายใน, ตำแหน่งชั่วคราว Parameter 2: สตริง, บัฟเฟอร์ข้อมูล

ตัวอย่างนิพจน์ 1

_strlen ([w:[#INTERNAL]LS0100], "ABCD")

เมื่อเรียกใช้ข้อความคำสั่งข้างบน ระบบจะเขียนความยาวของสตริงลงในพื้นที่ LS0100 ตามที่แสดงไว้ด้านล่าง

LS0100	4

ตัวอย่างนิพจน์ 2

_strlen ([t:0000], databuf0)

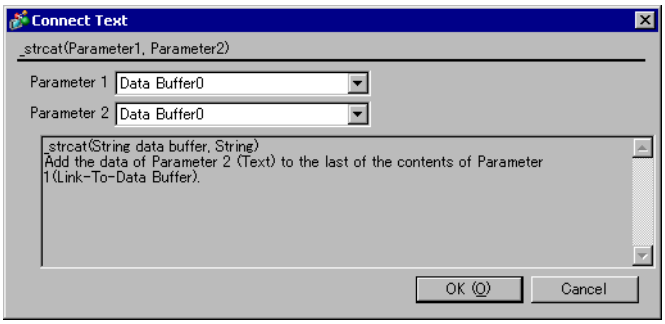
รายละเอียดของ “databuf0” มีดังนี้

	8 บิต	
databuf0[0]	31h	‘1’
databuf0[1]	32h	‘2’
databuf0[2]	33h	‘3’
databuf0[3]	34h	‘4’
databuf0[4]	00h	ค่าศูนย์

เมื่อเรียกใช้ข้อความคำสั่งข้างบน ระบบจะเขียนความยาวของสตริงลงใน [t: 0000] ตามที่แสดงไว้ด้านล่าง

t0000	4

21.11.11 Connect Text

รายการ	คำอธิบาย
ข้อมูลสรุป	เชื่อมสตริงอักขระหรือรหัสอักขระด้วยบัฟเฟอร์ข้อความ ฟังก์ชันนี้จะเพิ่มข้อมูลของ Parameter 2 (ข้อความ) ไว้ที่ตอนท้ายเนื้อหาของ Parameter 1 (บัฟเฟอร์ข้อมูลการติดต่อ)
รูปแบบ	_strcat (บัฟเฟอร์ข้อมูลการติดต่อ, ข้อความ)  Parameter 1: บัฟเฟอร์ข้อมูล Parameter 2: ข้อความ, ค่าตัวเลข (รหัสตัวอักษร), บัฟเฟอร์ข้อมูล (ช่วงที่ใช้ได้สำหรับ Parameter 2 คือ 0 และตั้งแต่ 1 ถึง 255)

ตัวอย่างนิพจน์ 1

_strcat (databuf0, "ABCD")

	8 บิต	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	00h	ค่าศูนย์

เมื่อเชื่อม "ABCD" เข้าด้วยกันตามที่แสดงข้างบน ผลที่ได้จะเป็นดังนี้ โปรดสังเกตว่าระบบจะเพิ่ม "ค่าศูนย์ (0x00)" เข้าไป

	8 บิต	
databuf0[0]	31h	'1'
databuf0[1]	32h	'2'
databuf0[2]	33h	'3'
databuf0[3]	34h	'4'
databuf0[4]	41h	'A'
databuf0[5]	42h	'B'
databuf0[6]	43h	'C'
databuf0[7]	44h	'D'
databuf0[8]	00h	ค่าศูนย์

ข้อสำคัญ

- สามารถกำหนดสตริงได้สูงสุด 255 อักขระ
- หากคุณตั้งค่าสตริงเปล่าที่มีค่าตัวเลข 0 ให้กับ Parameter 2 บัฟเฟอร์ข้อมูลของ Parameter 1 จะไม่เปลี่ยนแปลง ตัวอย่าง:_strcat (databuf0,"")_strcat (databuf0,0)

21.12 ตัวอย่างการทำงาน

21.12.1 ตัวอย่างการทำงานของฟังก์ชันตรรกศาสตร์

■ ต่อไปนี้เป็นตัวอย่างการทำงานของฟังก์ชันตรรกศาสตร์

◆ ((100 > 99) and (200 <> 100))
ผลลัพธ์: เปิด

◆ ((100 > 99) and (200 <> 200))
ผลลัพธ์: ปิด

◆ ((100 > 99) or (200 <> 200))
ผลลัพธ์: เปิด

◆ ((100 < 99) or (200 <> 200))
ผลลัพธ์: ปิด

◆ not (100 > 99)
ผลลัพธ์: ปิด

◆ not (100 < 99)
ผลลัพธ์: เปิด

◆ [w:D200] < 10
ผลลัพธ์: เป็นจริงหาก D200 น้อยกว่า 10

◆ not [w:D200]
ผลลัพธ์: เป็นจริงหาก D200 คือ 0

◆ ([w:D200] == 2) or ([w:D200] == 5)
ผลลัพธ์: เป็นจริงหาก D200 คือ 2 หรือ 5

◆ ([w:D200] < 5) and ([w:D300] < 8)
ผลลัพธ์: เป็นจริงหาก D200 น้อยกว่า 5 และ D300 น้อยกว่า 8

◆ [w:D200] < 10
ผลลัพธ์: เป็นจริงหาก D200 น้อยกว่า 10

◆ not [w:D200]
ผลลัพธ์: เป็นจริงหาก D200 คือ 0

◆ ([w:D200] == 2) or ([w:D200] == 5)
ผลลัพธ์: เป็นจริงหาก D200 คือ 2 หรือ 5

◆ ([w:D200] < 5) and ([w:D300] < 8)
ผลลัพธ์: เป็นจริงหาก D200 น้อยกว่า 5 และ D300 น้อยกว่า 8

21.12.2 ตัวอย่างการทำงานของบิต

■ ต่อไปนี้เป็นตัวอย่างการทำงานของบิต

◆ [w:D200] << 4
ผลลัพธ์: ข้อมูลใน D200 จะถูกเลื่อนไปทางซ้าย 4 บิต

◆ [w:D200] >> 4
ผลลัพธ์: ข้อมูลใน D200 จะถูกเลื่อนไปทางขวา 4 บิต

◆ จัดเก็บ 12(0000Ch) ไว้ใน D301 โดยใช้รูปแบบ BIN
[w:D200] = [w:D300] >> [w:D301]
ผลลัพธ์: ข้อมูลใน D300 ถูกเลื่อนไปทางขวา 12 บิตและกำหนดให้ตำแหน่ง D200

◆ [w:D200] << 4
ผลลัพธ์: ข้อมูลใน D200 จะถูกเลื่อนไปทางซ้าย 4 บิต

◆ [w:D200] >> 4
ผลลัพธ์: ข้อมูลใน D200 จะถูกเลื่อนไปทางขวา 4 บิต

◆ จัดเก็บ 12(0000Ch) ไว้ใน D310 โดยใช้รูปแบบ BIN
[w:D200] = [w:D300] >> [w:D310]
ผลลัพธ์: เลื่อนข้อมูลใน D300 ไปทางขวา 12 บิตและกำหนดให้ตำแหน่ง D200

◆ Bitwise AND

0 & 0	ผลลัพธ์: 0
0 & 1	ผลลัพธ์: 0
1 & 1	ผลลัพธ์: 1
0x1234 & 0xF0F0	ผลลัพธ์: 0x1030

◆ Bitwise OR

0 | 0	ผลลัพธ์: 0
0 | 1	ผลลัพธ์: 1
1 | 1	ผลลัพธ์: 1
0x1234 | 0x9999	ผลลัพธ์: 0x9BBD

◆ Bitwise XOR

0 ^ 0	ผลลัพธ์: 0
0 ^ 1	ผลลัพธ์: 1
1 ^ 1	ผลลัพธ์: 0

◆ Bitwise 1's Complement (เมื่อรูปแบบข้อมูลคือ BIN16+)

~ 0	ผลลัพธ์: 0xFFFF
~ 1	ผลลัพธ์: 0xFFFE

21.12.3 ตัวอย่างการคำนวณเพื่อใช้แบรนช์ตามเงื่อนไข

■ ควบคุมการทำงานของโปรแกรมโดยใช้นิพจน์ "if-endif" และ "if-else-endif"

◆ if-endif

```
if (เงื่อนไข)
{กระบวนการ 1}
endif
```

หากเงื่อนไขเป็นจริง จะดำเนินการกระบวนการ 1 หากเป็นเท็จ ระบบจะข้ามกระบวนการ 1

ตัวอย่าง:

```
if ( [ w:D200 ] < 5 )
{
    [ w:D100 ] = 1
}
endif
```

หากข้อมูลใน D200 น้อยกว่า 5 จะกำหนดค่า 1 ในตำแหน่ง D100

◆ if-else-endif

```
if (เงื่อนไข)
{กระบวนการ 1}
else
{กระบวนการ 2}
endif
```

หากเงื่อนไขเป็นจริง จะดำเนินการกระบวนการ 1 หากเป็นเท็จ จะดำเนินการกระบวนการ 2

ตัวอย่าง:

```
if ( [ w:D200 ] < 5 )
{
    [ w:D100 ] = 1
}
else
{
    [ w:D100 ] = 0
}
endif
```

หากค่าใน D200 น้อยกว่า 5 จะกำหนดค่า 1 ในตำแหน่ง D100 หากมากกว่า 5 จะกำหนดค่า 0

21.12.4 ตัวอย่างการคำนวณเพื่อใช้ตำแหน่งออฟเซ็ต

■ การระบุออฟเซ็ต: ตัวอย่างการคำนวณพิเศษโดยใช้ [w:D00100]#[t:0000]

◆ Script I/O: 16 **บิตที่ไม่ได้ระบุเครื่องหมาย**, [t:0000]= 65526, **ตำแหน่งผลลัพธ์คือ** [w:D00090]

100 + 65526 = 64(Hex) + FFF6(Hex) = 1005A(Hex) --> 005A(Hex) = 90

บิตที่ใช้ได้คือ 16 บิตล่าง

◆ Script I/O: 16 **บิตที่ระบุเครื่องหมาย**, [t:0000]= -10, **ตำแหน่งผลลัพธ์คือ** [w:D00090].

100 + (-10) = 64(Hex) + FFF6(Hex) = 1005A(Hex) --> 005A(Hex) = 90

บิตที่ใช้ได้คือ 16 บิตล่าง

◆ Script I/O: 32 **บิตที่ไม่ได้ระบุเครื่องหมาย**, [t:0000]= 4294901840, **ตำแหน่งผลลัพธ์คือ** [w:D00180]

100 + 4294901840 = 64(Hex) + FFFF0050(Hex) = FFFF00B4(Hex) --> 00B4(Hex) = 180

บิตที่ใช้ได้คือ 16 บิตล่าง

◆ Script I/O: 32 **บิตที่ระบุเครื่องหมาย**, [t:0000]= -65456, **ตำแหน่งผลลัพธ์คือ** [w:D00180]

100 + (-65456)= 64(Hex) + FFFF0050(Hex) = FFFF00B4(Hex) --> 00B4(Hex) = 180

บิตที่ใช้ได้คือ 16 บิตล่าง

ข้อสำคัญ

- ระบบจะถือว่าตำแหน่งออฟเซ็ตเป็นข้อมูลแบบ Bin 16 บิตเสมอ ไม่ว่าจะตั้งค่าความยาวบิตและชนิดข้อมูลของสคริปต์ไว้เช่นไรก็ตาม หากผลลัพธ์เกินกว่า 16 บิต (ค่าสูงสุด: 65535) ระบบจะถือว่าบิต 0 ถึง 15 เป็นบิตที่ถูกต้อง โดยไม่สนใจบิต 16 และที่สูงกว่า

21.13 รายการคำสั่ง

รายการ	คำสั่ง/ฟังก์ชัน	D-Script/ Global D-Script	Extended Script
Data Type	Bin, BCD	O	เฉพาะ Bin เท่านั้น
Bit Length	16 bit, 32 bit	O	O
Signed/	Unsigned	O	O
Trigger	Timer Setting	O	X
	Rising bit	O	X
	Falling bit	O	X
	Toggle bit	O	X
	Expression is true	O	X
	Expression is false	O	X
Draw	Load Screen	O	X
	Dot	O	O
	Line	O	O
	Circle	O	O
	Rectangle	O	O
Operator	Addition (+)	O	O
	Subtraction (−)	O	O
	Modulus (%)	O	O
	Multiplication (*)	O	O
	Division (/)	O	O
	Assignment (=)	O	O
Comparison	Logical AND	O	O
	Logical OR	O	O
	Negation (NOT)	O	O
	Less than (<)	O	O
	Less than or equal to (<=)	O	O
	Not equal to (<>)	O	O
	Greater than (>)	O	O
	Greater than or equal to (>=)	O	O
	Equals (==)	O	O

ต่อ

รายการ	คำสั่ง/ฟังก์ชัน	D-Script/ Global D-Script	Extended Script
Memory Operation	Copy Memory: memcpy ( )	O	O
	Initialize Memory: memset ( )	O	O
	Copy Memory (Specifying Variable): _memcpy_EX ( )	O	O
	Initialize Memory (Specifying Variable): _memset_EX ( )	O	O
	Offset Address	O	O
	Shift Memory	O	O
	Ring Shift Memory	O	O
	Search Memory	O	O
	Compare Memory	O	O
Bit Operation	Shift Left (<<)	O	O
	Shift Right (>>)	O	O
	Bitwise AND (&)	O	O
	Bitwise OR (|)	O	O
	Bitwise XOR (^)	O	O
	1's Complement	O	O
	Set Bit: set ( )	O	O
	Clear Bit: clear ( )	O	O
	Toggle Bit: toggle ( )	O	O
Description Expression	if ( )	O	O
	if ( ) else	O	O
	loop ( ), break	O	O
	loop ( ) infinite loop	X	O
Address	Bit Address	O	Internal Device
	Word Address	O	Internal Device
	Temporary Working Address	O	O*1
Constant	Dec, Hex, Oct	O	O

ต่อ

รายการ	คำสั่ง/ฟังก์ชัน	D-Script/ Global D-Script	Extended Script
SIO Function	Receive: IO_READ ([p:SIO])	O	O
	Send: IO_WRITE ([p:SIO])	O	O
	Extended Receive: _IO_READ_EX ( )	X	O
	Extended Send: _IO_WRITE_EX ( )	X	O
	Standby Receive Function: _IO_READ_WAIT ( )	X	O
	ตัวแปรควบคุม [c:EXT_SIO_CTRL]	O	O
	Status [s:EXT_SIO_STAT]	O	O
	Number of Received Data [r:EXT_SIO_RCV]	O	O
	Pause: _wait ( )	X	O

ต่อ

รายการ	คำสั่ง/ฟังก์ชัน	D-Script/ Global D-Script	Extended Script
Text Operation	Text	X	O
	Data Buffer: databuf0, databuf1, databuf2, databuf3	X	O
	Write String: _strset ( )	X	O
	Cop from Data Buffer to Internal Device: _dlcopy ( )	X	O
	Copy from Internal Device to Data Buffer: _ldcopy ( )	X	O
	Hexadecimal Text-To-Integer Conversion: _hexasc2bin ( )	X	O
	Decimal Text-To-Integer Conversion: _decasc2bin ( )	X	O
	Hexadecimal Number to String Conversion: _bin2hexasc ( )	X	O
	Decimal Number to String Conversion: _bin2decasc ( )	X	O
	String Length: _strlen ( )	X	O
	String Concatenate: _strcat ( )	X	O
	Copy Partial String: _strmid ( )	X	O
	Status: [e: STR_ERR_STAT]	X	O
Standby Reception	Call	O	O
	return	X	O

ต่อ

รายการ	คำสั่ง/ฟังก์ชัน	D-Script/ Global D-Script	Extended Script
CF File Operation	Read CSV File	O	O
	Output File List: _CF_dir ( )	O	O
	Read File: _CF_read ( )	O	O
	Write File: _CF_write ( )	O	O
	Delete File: _CF_delete ( )	O	O
	Edit File Name: _CF_rename ( )	O	O
Printer Operation	Output COM Port: IO_WRITE ([p:PRN])	O	O
Debug:	_debug ( )	O	O

*1 ตำแหน่งชั่วคราวมีอยู่โดยแยกต่างหากจาก D-Script และ Global D-Script