三菱電機 通用 可程式控制器

MELSEC iQ-R series

MELSEC iQ-R
程式手冊(程式設計篇)

安全注意事項

(使用之前務必閱讀)

使用MELSEC iQ-R系列可程式控制器之前，應仔細閱讀各產品手冊及各產品手冊中所介紹的關聯手冊，同時在充分注意安全的前提下正確地操作。

請妥善保管本手冊以備需要時閱讀，並應將本手冊交給最終用戶。

關於產品的應用

（1）使用三菱可程式控制器時，請符合以下條件：

即使可程式控制器出現問題或故障時，也不會導致重大事故。並且在設備外部以系統性規劃，當發生問題或故障時的備份或失效安全防護功能。

（2）三菱可程式控制器是以一般工業等用途為對象，設計和製造的泛用產品。

因此，三菱可程式控制器不適用於以下設備、系統的特殊用途上。如果用於以下特殊用途時，對於三菱可程式控制器的品質、性能、安全等所有相關責任（包括，但不限定於債務未履行責任、瑕疵擔保責任、品質保證責任、違法行為責任、製造物責任），三菱電機將不負責。

- 各電力公司的核能發電廠以及其他發電廠等，對公眾有較大影響的用途。
- 各鐵路公司及公家機關等，對於三菱電機有特別的品質保證體制之架構要求的用途。
- 航空宇宙、醫療、鐵路、焚燒、燃料裝置、乘載移動設備、載人運輸裝置、娛樂設備、安全設備等，預測對性命、人身、財產有較大影響的用途。

但是，即使是上述對象，只要有具體的限定用途，沒有特殊的品質（超出一般規格的品質等）要求之條件下，經過三菱電機的判斷依然可以使用三菱可程式控制器，詳細情形請洽詢當地三菱電機代表窗口。

- 使用SIL2過程CPU時

(1) 儘管安全控制器已經取得了德國TUV Rheinland的國際安全標準IEC61508和IEC61511的產品可靠性認證，但這並不保證本產品不發生任何故障。本產品的用戶應遵守所有現行的安全標準、規則或法律，並應對本產品所安裝或使用的系統採取適當的安全措施，除了本產品之外還應當同時採取其它的安全措施。對於如果遵守了現行的安全標準、規則或法律而可以預防的損害，三菱電機公司(簡稱三菱電機)不負任何責任。

(2) 三菱電機禁止將本產品用於可能涉及人員生命健康安全和重大財產安全的用途，如果違反了三菱電機的指示將其用於以下用途，對於由此引起的一切責任(包括但不僅限於債務未履行責任、瑕疵擔保責任、質量保證責任、違法行為責任、製造物責任)，三菱電機將不負責。

1)火力/水力/核能發電廠

2)火車/鐵路系統、飛機、航空管理、其它交通系統

3)醫院、醫療及與生命維持相關設備的應用

4)娛樂設備

5)焚燒和燃料裝置

6)核物質、有害物質及化學物質的處理設備

7)採礦、挖掘

8)其它上述1)～7)中未包含的涉及人員生命、健康或重大財產安全的用途

• 使用安全CPU時

(1) 儘管安全控制器已經取得了德國TUV Rheinland的國際安全標準IEC61508和EN954-1/ISO13849-1的產品可靠性認證，但這並不保證本產品不發生任何故障。本產品的用戶應遵守所有現行的安全標準、規則或法律，並應對本產品所安裝或使用的系統採取適當的安全措施，除了本產品之外還應當同時採取其它的安全措施。對於如果遵守了現行的安全標準、規則或法律而可以預防的損害，三菱電機公司(簡稱三菱電機)不負任何責任。

(2) 三菱電機禁止將本產品用於可能涉及人員生命健康安全和重大財產安全的用途，如果違反了三菱電機的指示將其用於以下用途，對於由此引起的一切責任(包括但不僅限於債務未履行責任、瑕疵擔保責任、質量保證責任、違法行為責任、製造物責任)，三菱電機將不負責。

1)火力/水力/核能發電廠

2)火車/鐵路系統、飛機、航空管理、其它交通系統

3)醫院、醫療及與生命維持相關設備的應用

4)娛樂設備

5)焚燒和燃料裝置

6)核物質、有害物質及化學物質的處理設備

7)採礦、挖掘

8)其它上述1)～7)中未包含的涉及人員生命、健康或重大財產安全的用途

前言

在此非常感謝貴方購買了三菱電機可程式控制器MELSEC iQ-R系列產品。
本手冊是用於讓用戶了解進行程式設計時必要的程式配置及使用資料相關內容的手冊。
在使用之前應熟讀本手冊及關聯手冊，在充分了解MELSEC iQ-R系列可程式控制器的功能・性能的基礎上正確地使用本產品。
此外，將本手冊中介紹的程式示例應用於實際系統的情況下，應充分驗證對象系統中不存在控制方面的問題。
應將本手冊交給最終用戶。

要點

在本手冊中，主要使用標籤進行說明。而元件也可以像標籤一樣使用。

目錄

關聯手冊

要取得最新的e-Manual以及手冊PDF，請向當地三菱電機代理店諮詢。

手冊名稱[手冊編號]	內容	提供形式
MELSEC iQ-R 程式手冊(程式設計篇) [SH-081320CHT](本手冊)	記載梯形圖、ST、FBD/LD、SFC的程式規格有關內容。	e-Manual PDF
MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇) [SH-081323CHT]	記載CPU模組的指令、通用函數/通用FB有關內容。	e-Manual PDF
MELSEC iQ-R 程式手冊(過程控制FB/指令篇) [SH-081751CHT]	記載過程控制中特有的通用過程FB、標籤訪問FB、標籤FB及過程控制指令有關內容。	e-Manual PDF
GX Works3 操作手冊 [SH-081272CHT]	說明GX Works3的系統配置、參數設置、在線功能的操作方法等有關內容。	e-Manual PDF

要點

e-Manual是指可透過專用工具瀏覽的三菱電機FA電子書籍手冊。

e-Manual有如下所示的特點。

- 可以從多本手冊同時搜尋需要的資訊(手冊交叉搜尋)
- 可以從手冊內的連結參閱其他手冊
- 可以從產品插圖的各部分瀏覽想要了解的硬體規格
- 可以將頻繁瀏覽的資訊登錄到收藏夾
- 可以將樣本程式複製到工程工具中

術語

在本手冊中，除非特別標明，將使用下述術語進行說明。

術語	說明
CPU模組	是MELSEC iQ-R系列CPU模組的總稱。
GX Works3	是MELSEC可程式控制器套裝軟體SWnDNC-GXW3的產品名總稱。(n表示版本。)
常規/安全共享標籤	是常規程式及安全程式中使用的標籤。在安全程式及常規程式之間交接資料時使用。
智能功能模組	是具有A/D、D/A轉換模組等輸入輸出以外功能的模組。
工程工具	是MELSEC可程式控制器套裝軟體的產品名。
操作數	是在各個指令及函數的內部配置中使用的源資料(s)、目標資料(d)、元件數(n)等元件部分的總稱。
通信協定支援功能	是可以在GX Works3(通信協定支援功能)使用的功能。 功能概要如下所示。 ・與對象設備相符合的協定的設置 ・協定設置資料的讀取/寫入
元件	是CPU模組內部具有的元件(X、Y、M、D等)。
輸入輸出模組	是輸入模組、輸出模組、輸入輸出混合模組、中斷模組的總稱。
網路模組	是下述模組的總稱。 ・乙太網路接口模組 ・CC-Link IE控制網路模組 ・CC-Link IE現場網路模組 ・MELSECNET/H網路模組 ・MELSECNET/10網路模組 ・RnENCPU(網路部)
緩衝記憶體	是用於儲存設置值、監視值等資料的智能功能模組的記憶體。 CPU模組的情況下，是指用於儲存乙太網路功能的設置值、監視值等資料及多CPU功能的資料通信中所使用的資料等的記憶體。
程式部件	是按功能分開定義的程式單位。透過程式的部件化，可以將程式分級時的低位處理按處理內容及功能分為若干個單位，創建各單位的程式。
多CPU系統	是在多個(2～4個)CPU模組中控制各自管理的輸入輸出模組及智能功能模組的系統。
模組標籤	是將各模組特有定義的記憶體(輸入輸出信號及緩衝記憶體)以任意字元串表示的標籤。可以從使用的模組由工程工具自動生成，作為全局標籤使用。

此外，使用SIL2過程CPU及安全CPU的情況下，也將使用下述術語進行說明。

術語	說明
安全控制	是執行安全程式及安全通信實施機械的控制。發生異常時，使機械安全停止。
安全通信	是進行安全通信協定所定義的安全層的發送接收處理的通信服務。
安全元件	是安全程式中可使用的元件。
安全程式	是用於執行安全控制的程式。
常規CPU	是執行常規控制的MELSEC iQ-R系列的各CPU模組的總稱。(與執行安全控制的CPU模組進行區別時使用。)
常規控制	是執行常規程式及常規通信實施機械的控制。安全可程式控制器以外僅保有常規控制。(與安全控制進行區別時使用。)
常規通信	是安全通信以外的通信(CC-Link IE現場網路的循環傳送與瞬時傳送等)。
常規元件	是CPU模組內部具有的安全元件以外的元件(X、Y、M、D等)。僅在常規程式中可以使用。(與安全元件進行區別時使用。)
常規程式	是用於執行順控程式控制的安全程式以外的程式。(與安全程式進行區別時使用。)

1 概要

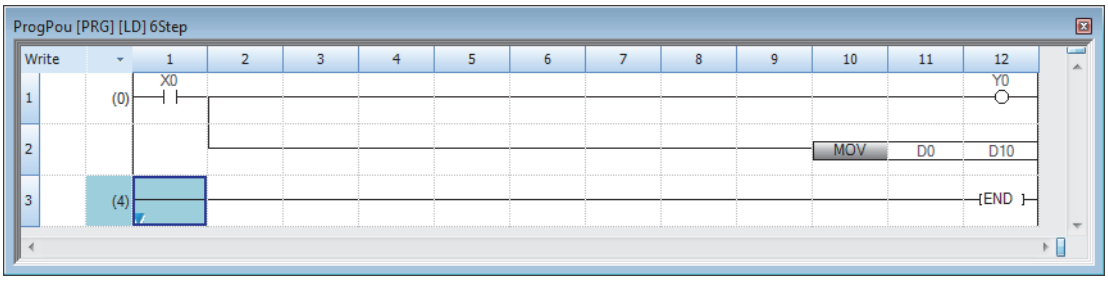
本手冊中記載創建程式所需要的程式配置、內容與記述方法等相關內容。
關於在工程工具中的程式創建、編輯、監視方法，請參閱下述手冊。
GX Works3 操作手冊

程式語言的類型

MELSEC iQ-R系列中，可以根據用途選擇最合適的程式語言使用。

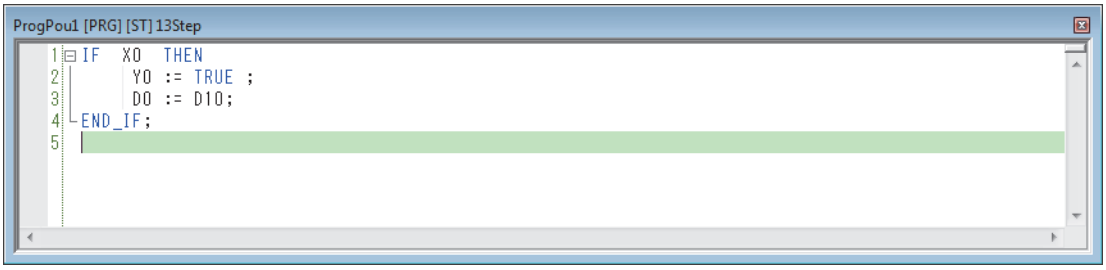
程式語言	內容
梯形圖圖表語言(梯形圖語言)	是用觸點及線圈等表示梯形圖的圖表語言。 梯形圖語言是為了進行簡單易懂的順控程式控制，使用符號化的觸點及線圈等記述邏輯梯形圖的語言。
結構化文本語言(ST語言)	是使用IF語句及運算符等記述程式的文本語言。 與梯形圖語言相比，由於ST語言能對難以記述的運算處理以簡潔易懂的方式記述，因此適用於進行複雜的算術運算及比較運算等的領域。此外，可以和C語言等一樣，記述透過條件語句進行的選擇分支及重複語句等語法進行的控制，因此可以簡潔地編寫出易懂的程式。
FB圖表/梯形圖語言(FBD/LD語言)	是透過按照資料及信號的流向對進行特定處理的塊、變數部件、常數部件進行連接，對程式進行記述的圖表語言。 透過梯形圖程式創建時，可以輕鬆地創建非常複雜的DDC處理程式，提高程式生產能力。
順序功能圖(SFC程式)	是將一系列的控制動作劃分為多個步，使得能夠清楚表示各程式的執行順序及執行條件的程式的記述形式。

■梯形圖語言



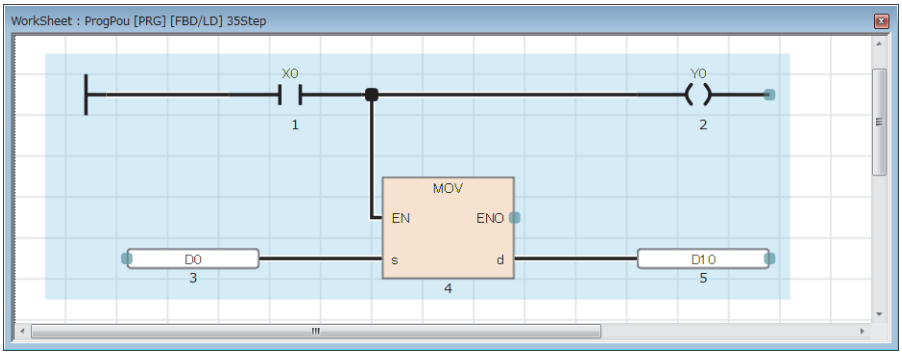
使用梯形圖語言的情況下，請參閱下述章節。
41頁 梯形圖語言

■ST語言



使用ST語言的情況下，請參閱下述章節。
47頁 ST語言

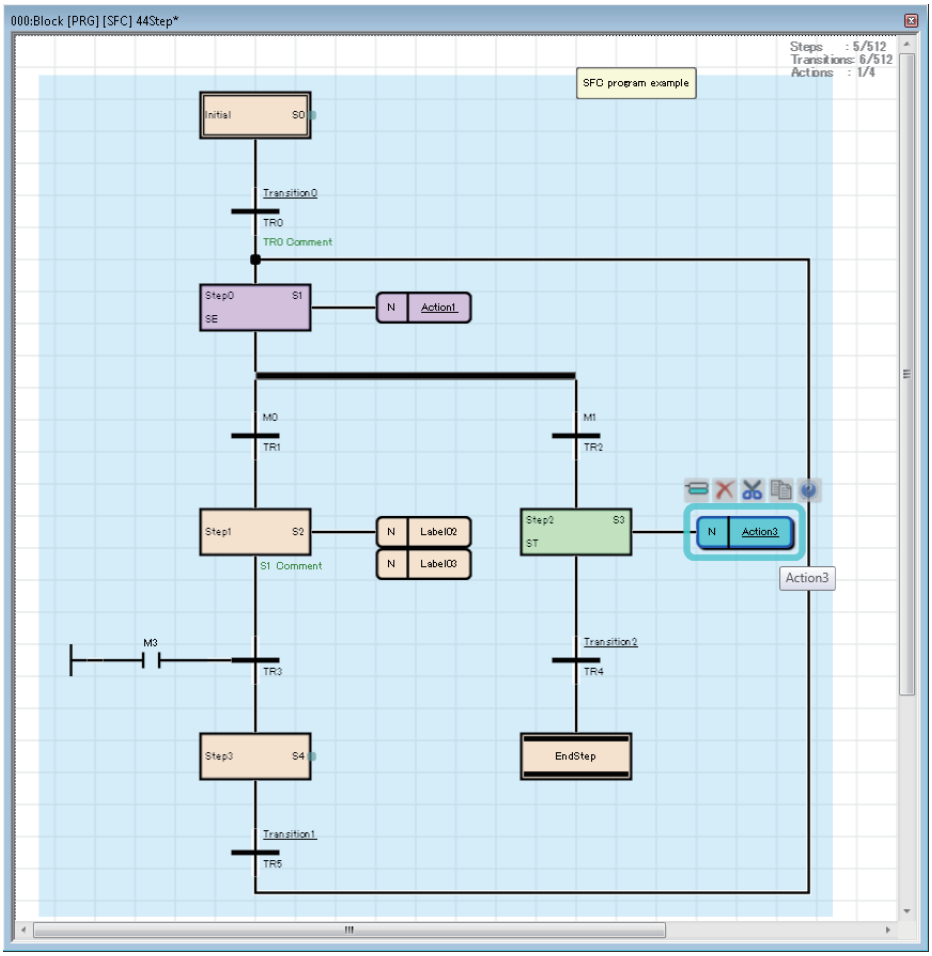
■FBD/LD語言



使用FBD/LD語言的情況下，請參閱下述章節。

☞63頁 FBD/LD語言

■SFC程式



使用SFC程式的情況下，請參閱下述章節。

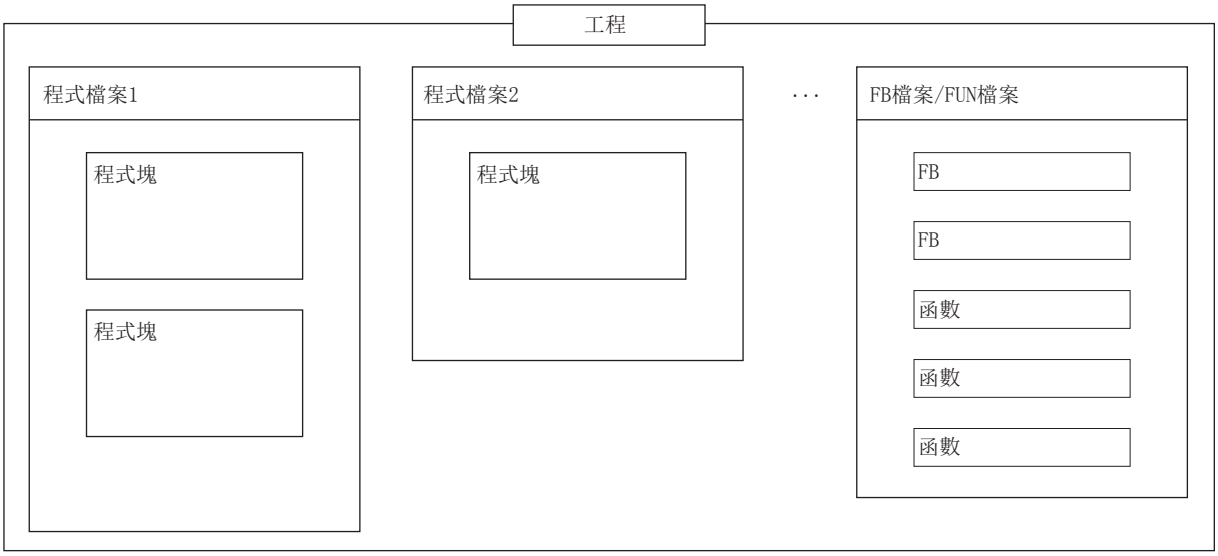
☞76頁 SFC程式

要點

- 梯形圖語言適用於具有順控程式控制、邏輯梯形圖知識及經驗的用戶。ST語言適用於具有C語言等程式的知識及經驗的用戶。FBD/LD語言適用於與過程控制相關的用戶。SFC程式適用於對各機械的實際控制進行分割程式、對作業的切換進行管理的情況。
- 透過在程式中使用標籤，可以提高程式的可讀性，將程式簡單地移轉至模組配置不同的系統中。

2 程式配置

工程工具中可以創建多個程式及多個程式部件。
因此，可以根據處理來劃分程式及程式部件。
本章介紹程式配置有關內容。



關於程式部件，請參閱下述章節。
☞ 12頁 程式部件

工程

工程是在CPU模組中執行的資料(程式、參數等)的集合。
一個CPU模組中只可寫入一個工程。
工程中至少需要創建一個程式檔案。

程式檔案

程式檔案是程式及程式部件的集合。
程式檔案由一個或其以上的程式塊所配置。(☞ 13頁 程式塊)
透過程式檔案單位的操作，可以將程式的執行類型由恆定週期執行類型替換為待機類型，更改是否將資料寫入至CPU模組中。

備忘録

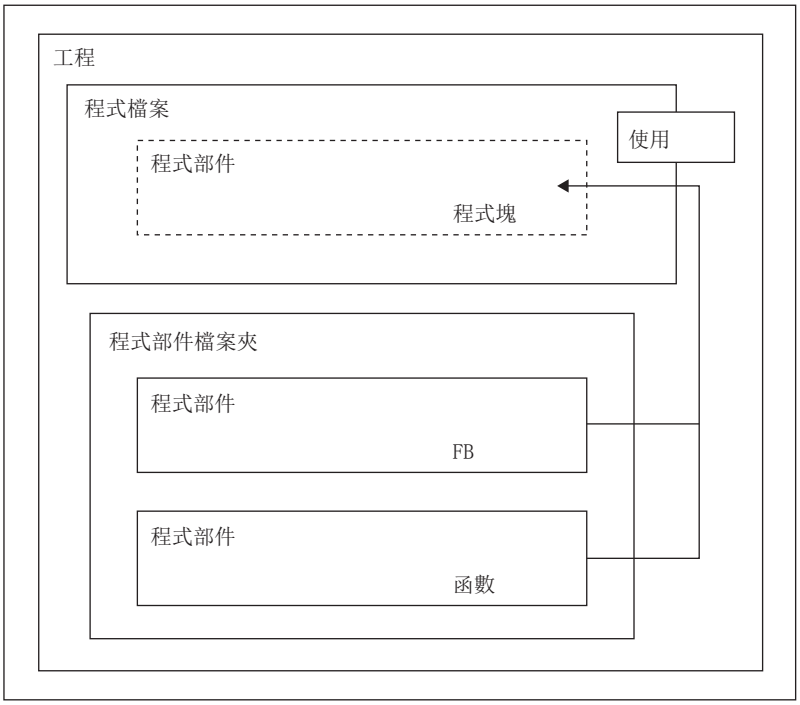
3 程式部件

程式部件有下述幾種類型。

- 程式塊
- 函數
- 功能塊

各程式部件可以透過符合控制的程式語言對處理進行記述。在函數及FB中，可以透過梯形圖語言、ST語言、FBD/LD語言對處理進行記述。

函數與FB是從程式塊調用後執行。



要點

程式的部件化是指將程式階層化時的低位處理按照各處理內容及功能分為若干單位，創建各單位的程式。

透過程式的部件化，可提高獨立性，並設計更容易進行添加及更換的程式。

部件化後變得更好的處理方法，有以下幾種。

- 在程式中被循環記述處理
- 作為一個功能被分開處理

在本項中使用標籤對各程式部件進行說明。

各程式部件的程式本體(工作表)也可以使用元件。關於元件的詳細內容，請參閱下述手冊。

MELSEC iQ-R CPU模組用戶手冊(應用篇)

要點

在ST語言與FBD/LD語言中，在一個程式部件內最多可以創建32個工作表。

多個工作表的執行順序，從工程工具的“工作表執行順序設定”畫面進行設置。(GX Works3 操作手冊)

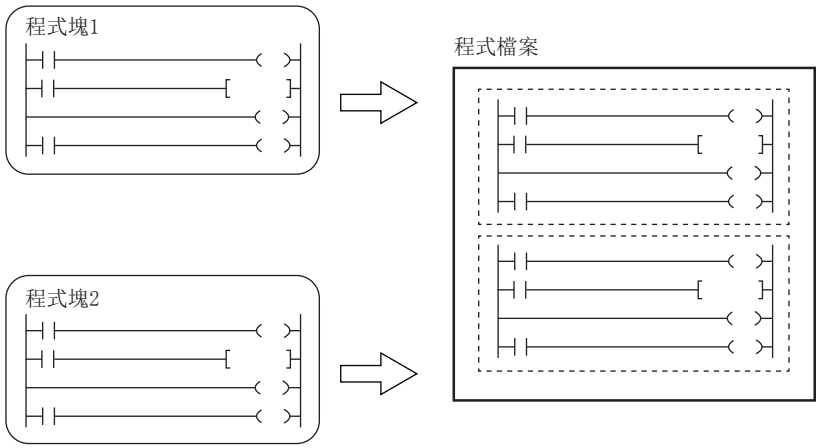
3.1 程式塊

程式塊為配置程式的單位。

在程式檔案內可以創建多個程式塊，並按照程式檔案設置中指定的順序被執行。程式檔案設置中未指定順序的情況下，按照程式塊名的順序(昇序)被執行。

如果對各功能及處理劃分程式塊，則設計時能方便地進行程式的順序更改及更換。

將程式塊的程式本體儲存到各登錄目標程式中的程式檔案中。



程式塊的分割

可分別對程式塊創建主程式、子程式及中斷程式。*1

類型	內容
主程式	是指從程式的步0到FEND指令為止的程式。
子程式	是指從指針(P)到RET指令為止的程式。 只在透過子程式調用指令(CALL指令、ECALL指令等)調用的情況下執行。
中斷程式	是從中斷指針(I)到IRET指令為止的程式。 如果發生中斷原因，將執行與該中斷指針編號相對應的中斷程式。

*1 在安全程式內不能創建子程式及中斷程式。此外，不能透過安全程式執行子程式。

要點

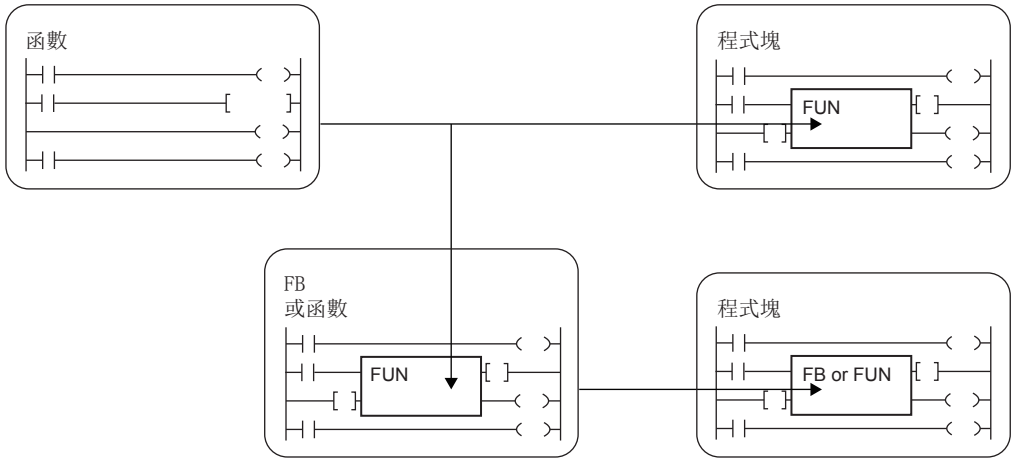
- 子程式以及中斷程式是在FEND指令以後進行創建。FEND指令以後的程式將不作為主程式執行。例如，在第二個程式塊的最後使用了FEND指令的情況下，第三個程式塊及以後將變為子程式或中斷程式。
- 為了創建易懂的程式，應在一個程式塊中使用成對的FOR指令與NEXT指令，或MC指令與MCR指令。
- 簡單程式的情況下，僅在一個程式塊內記述主程式即可使其在CPU模組中執行。

關於子程式、中斷程式的詳細內容，請參閱下述手冊。

MELSEC iQ-R CPU模組用戶手冊(應用篇)

3.2 函數(FUN)

函數是指在程式塊、FB以及其他的函數中所使用的程式部件。
函數執行完成後將值交接至調用源。該值稱為返回值。
對於同樣的輸入，函數將作為處理結果始終輸出相同的返回值。
如果預先定義經常使用的單純獨立的程式算法，可以有效地再利用。

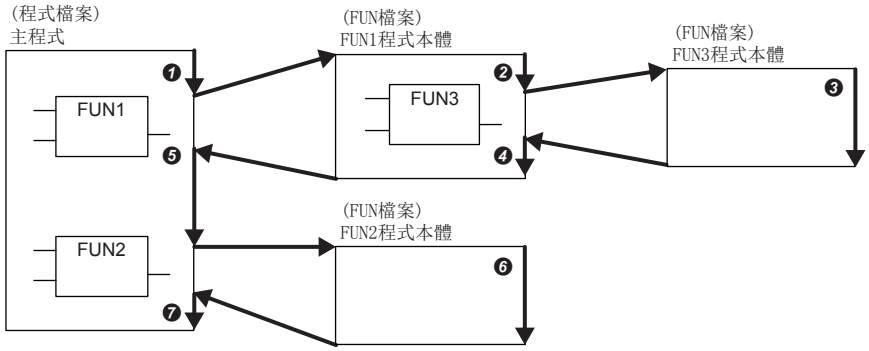


動作概要

將程式本體儲存在FUN檔案內，透過從調用源程式調用FUN檔案內的程式本體來執行函數。

例

從主程式調用FUN1及FUN2，且FUN1又調用FUN3的情況下(嵌套數： 3次)
❶～❼表示執行的流程(順序)。

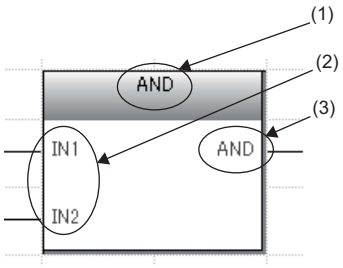


對於子程式型FB、宏型FB、函數，合計最多可進行32次嵌套。

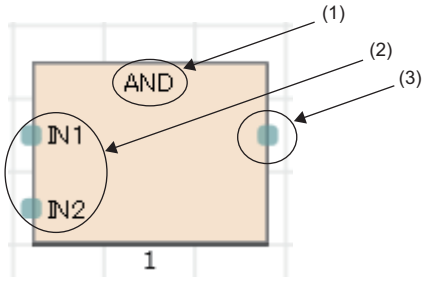
關於輸入變數與輸出變數

函數可以定義輸入變數與輸出變數。輸出變數可以分配與返回值不同的其他輸出資料。

梯形圖語言的情況下



FBD/LD語言的情況下



(1) 函數名
(2) 輸入變數
(3) 輸出變數

不顯示函數的返回值名稱。

輸入變數以VAR_INPUT的分類、輸出變數以VAR_OUTPUT的分類進行設置。

要點

函數中定義的變數在每次函數調用時被覆蓋。
希望在每次調用時保持變數值的情況下，應透過使用FB或將輸出變數儲存為不同的變數等進行程式設計。

關於EN/ENO

透過在函數中附加EN(允許輸入)、ENO(允許輸出)，可以控制執行處理。

- 設置作為函數的執行條件的布爾型變數至EN。
- 帶EN的函數只在EN的執行條件為TRUE的情況下執行。
- 設置輸出函數的執行結果的布爾型變數至ENO。

根據EN狀態的ENO與運算結果的內容如下所示。

EN	ENO	運算結果
TRUE(運算執行)	TRUE	運算輸出值
FALSE(運算停止)	FALSE	不定值

要點

- 在梯形圖語言、FBD/LD語言的程式中，不需要進行至ENO的輸出標籤的設置。
- 在通用函數中使用EN/ENO的情況下，帶EN的函數將變為“函數名_E”。

程式的創建

創建函數程式的情況下，執行下述操作。

[導航視窗]⇨[FB/FUN]⇨右擊⇨[新增資料]
在“基本設定”的“資料類型”中選擇“函數”

已創建的程式儲存在FUN檔案中。

[CPU參數]⇨[程式設定]⇨[FB/FUN檔案設定]

一個FUN檔案中最多可以儲存64個創建的程式。

在函數內無法使用上升沿執行/下降沿執行指令。

關於與程式創建相關的內容，請參閱下述手冊。

項目	參照目標
函數的創建方法	GX Works3 操作手冊
可寫入CPU模組的FB/FUN檔案數	MELSEC iQ-R CPU模組用戶手冊(入門篇)

■可使用的元件/標籤

函數程式中可使用的元件及標籤一覽如下所示。

○：可以使用、 △：只可以在指令中使用(禁止作為表示程式的步的標籤使用)、 ×：禁止使用

元件/標籤的類型		能否使用
標籤(指針型以外)	全局標籤	×
	局部標籤	○*1
標籤(指針型)	指針型全局標籤	×
	指針型局部標籤	○
元件	全局元件	○
	局部元件	×
指針	全局指針	△
	局部指針	×

*1 不能使用下述資料類型。
定時器、累計定時器、計數器、長定時器、長累計定時器、長計數器

要點

函數的返回值，可以透過在函數內將函數名作為標籤進行編程來進行設置。函數名不需要作為標籤進行設置。在函數的屬性中，可以在“返回值類型”中設置的資料類型中使用。

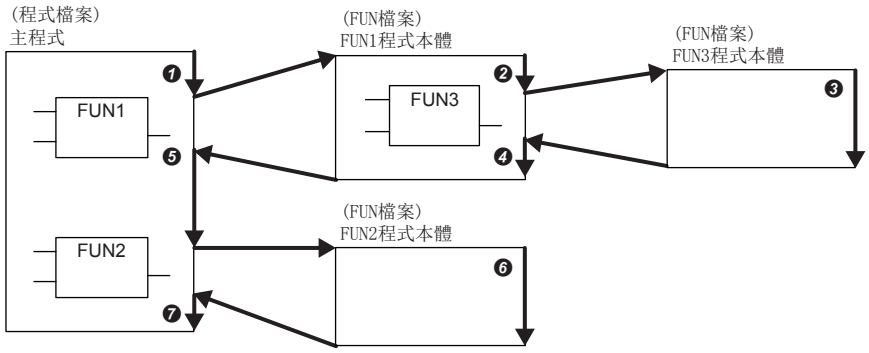
在函數中定義的標籤

函數中定義的標籤的分配目標在執行函數時確保於記憶體內的暫時區域，在執行完成時解除。

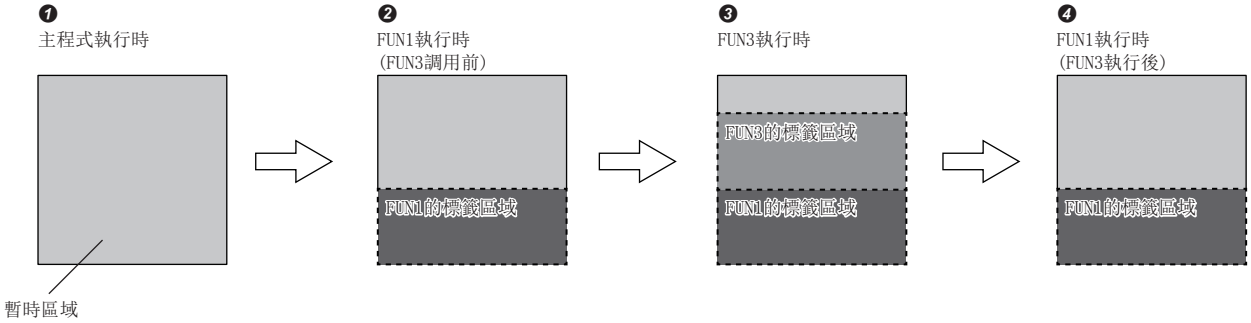
例

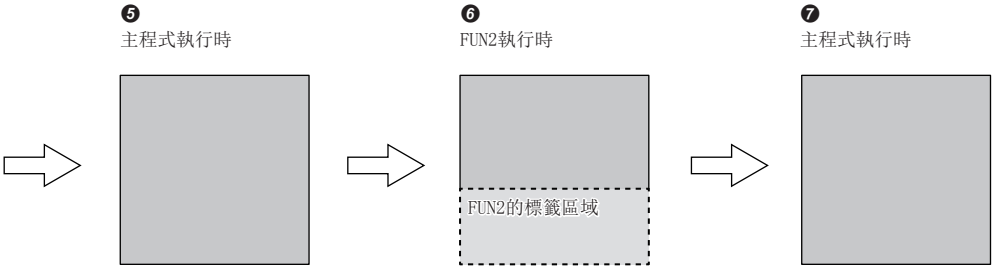
從主程式調用FUN1及FUN2，且FUN1又調用FUN3的情況下

(❶～❼表示執行的流程(順序))



上述函數執行動作的標籤分配狀態如下所示。





在函數內可定義的標籤的分類為VAR、VAR_CONSTANT、VAR_INPUT、VAR_OUTPUT。

要點

由於在函數中定義的標籤變為不定值，因此在最初訪問時需要透過程式進行初始化。

步數

調用函數的情況下，除了程式本體的步數，還需要進行引數及返回值的交接處理及調用程式本體的步數。

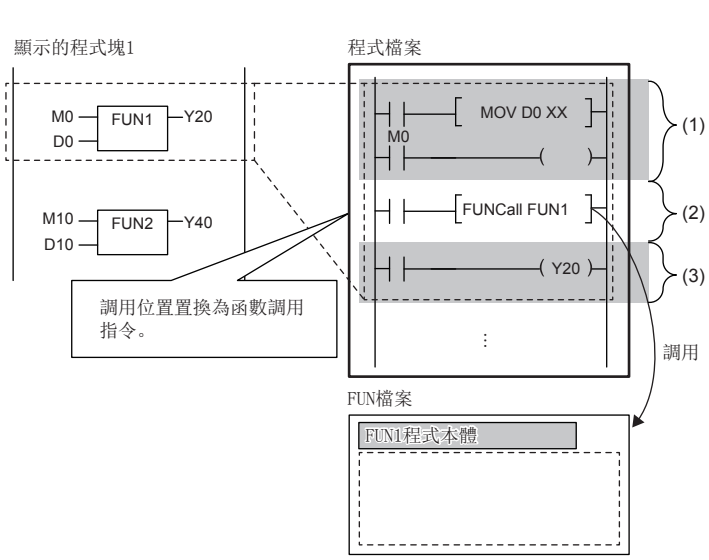
■程式本體

所使用函數的程式本體的步數為指令步數的總計加上22步的值。關於各指令的步數，請參閱下述手冊。

MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)

■調用側

調動函數的情況下，在函數調用前後生成函數的引數以及返回值的交接處理。



• 引數交接

引數交接中使用的指令根據引數的分類及引數的資料類型而不同。在引數交接中使用的指令如下所示。

引數的分類	資料類型	使用指令	步數
VAR_INPUT	位元	LD+OUT LD+MOVB (根據所使用的程式語言、函數的類型、輸入引數類型的組合，使用其中的一個。)	各指令步數的詳細內容，請參閱下述手冊。 MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)
	字[無符號]/位元串[16位元] 雙字[無符號]/位元串[32位元] 字[帶符號] 雙字[帶符號]	LD+MOV LD+DMOV	
	單精度實數	LD+EMOV	
	雙精度實數	LD+EDMOV	
	時間	LD+DMOV	
	字元串	LD+$MOV	
	字元串[Unicode]	LD+$MOV_WS	
	陣列、結構體	LD+BMOV	

• 程式本體調用

函數的程式本體的調用需要26步。

• 返回值交接

在返回值交接中使用的指令及步數與引數交接相同。

引數的分類	資料類型	使用指令	步數
VAR_OUTPUT	與引數交接相同	與引數交接相同	與引數交接相同

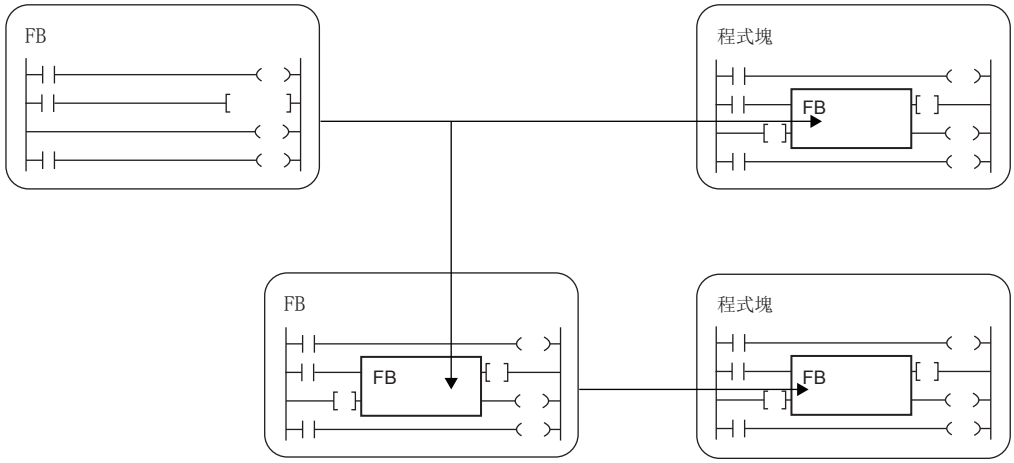
• EN/ENO

EN/ENO所需步數如下所示。

項目	步數
EN	3
ENO	2

3.3 功能塊(FB)

FB是程式塊及其他FB中所使用的程式部件。

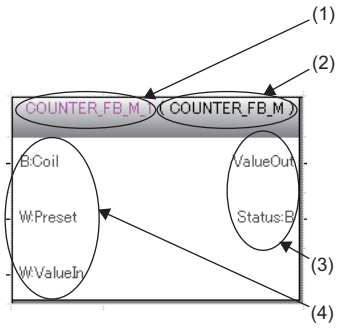


FB與函數不同，不能保持返回值。

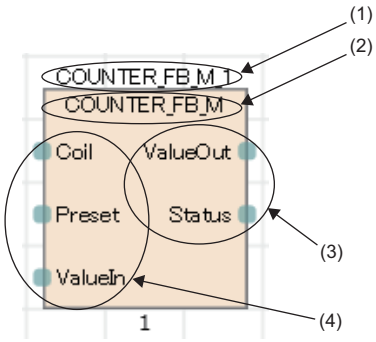
因為FB能將值儲存在變數中，因此也能保持輸入狀態及處理結果。

為了在下一次處理中使用保持後的值，因此即使為相同的輸入值也不一定每次都輸出相同的結果。

梯形圖語言的情況下



FBD/LD語言的情況下



(1) 實例名
(2) FB名
(3) 輸出變數
(4) 輸入變數

此外，為了在程式上使用FB，需要定義實例。

☞22頁 實例

要點

- 通用FB的詳細內容，請參閱下述手冊。

📖MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)

- 過程控制FB的詳細內容，請參閱下述手冊。

📖MELSEC iQ-R 程式手冊(過程控制FB/指令篇)

- 模組FB的詳細內容，請參閱下述手冊。

📖所使用模組的FB參考

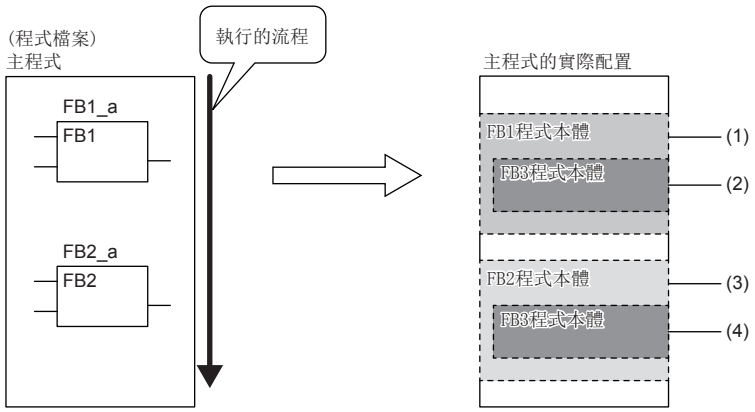
動作概要

■宏型FB

宏型FB在程式設計時，對調用源程式展開調用對象的程式本體。執行時與普通的程式同樣執行展開的程式。
希望優先程式的處理速度的情況下，應使用宏型FB。

例

從主程式調用FB1_a及FB2_a，且FB1_a又調用FB3_a，接著FB2_a又調用FB3_b的情況下



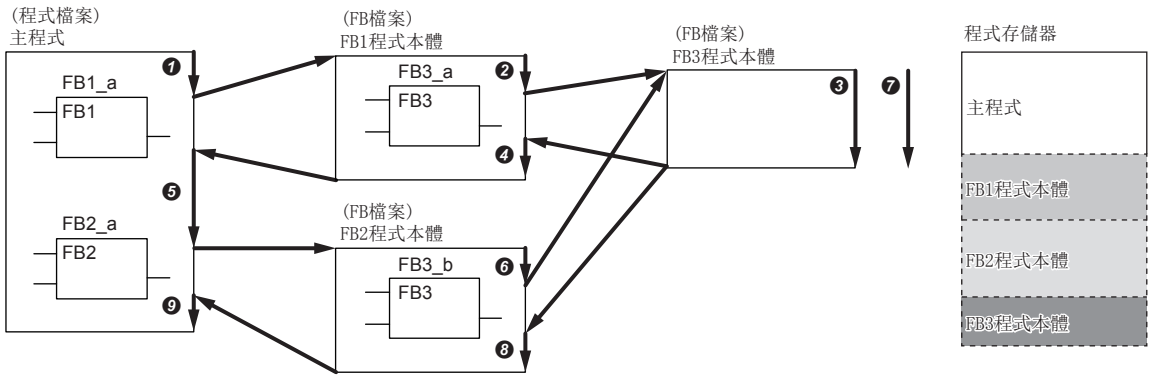
(1) 展開主程式中的FB1程式本體後執行。
(2) 從FB1調用的FB3在FB1程式本體中展開。
(3) FB2與FB1同樣，FB2程式本體在主程式中展開後執行。
(4) 從FB2調用的FB3在FB2程式本體中展開。

■子程式型FB

子程式型FB將程式本體儲存在FB檔案內，要執行時再從調用源程式調用FB檔案內的程式本體執行。
希望使程式的容量變小的情況下，應選擇子程式型FB。

例

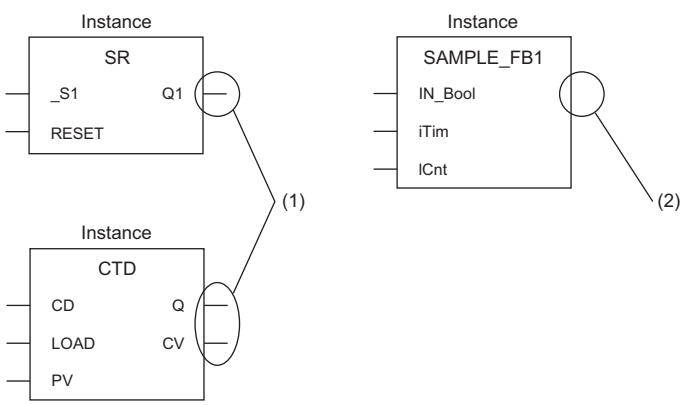
從主程式調用FB1_a及FB2_a，且FB1_a又調用FB3_a，接著FB2_a又調用FB3_b的情況下(嵌套數： 3次)
❶～❾表示執行的流程(順序)。



對於子程式型FB、宏型FB、函數，合計最多可進行32次嵌套。

關於輸入變數、輸出變數、輸入輸出變數

FB中需要定義輸入變數、輸出變數、輸入輸出變數。
FB可以輸出多個運算結果。此外，也可以不輸出。



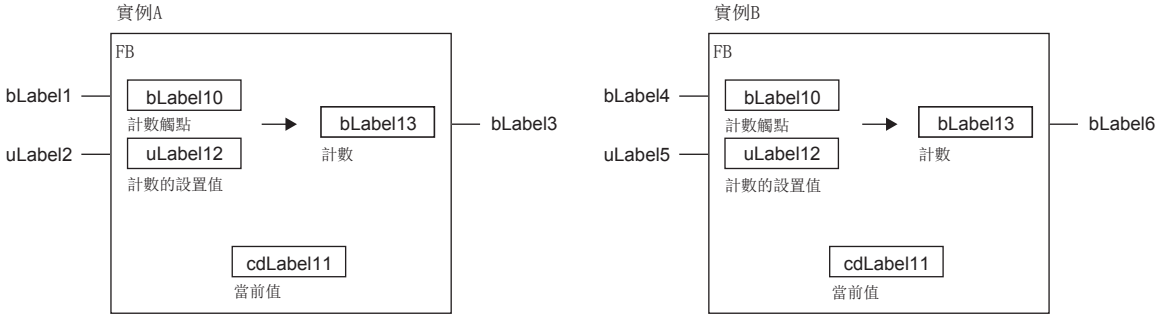
(1) 有輸出
(2) 無輸出

輸入變數以VAR_INPUT的分類、輸出變數以VAR_OUTPUT及VAR_OUTPUT_RETAIN的分類、輸入輸出變數以VAR_IN_OUT的分類進行設置。

關於內部變數

FB使用內部變數。在內部變數中，在不同的區域對FB的各實例分配標籤。即使是同樣的標籤名，各實例都可保持不同的狀態。

例



是指輸入變數變為ON時開始計數，當內部變數中保持的當前值達到設置值時，將輸出變數置為ON的FB。即使是同一個FB，因為實例A與實例B各自保持著獨自的狀態，所以輸出的時機有所不同。
內部變數以VAR、VAR_CONSTANT、VVAR_RETAIN的分類進行設置。

關於外部變數及公開變數

FB可以使用外部變數(全局標籤)及公開變數。
公開變數以VAR_PUBLIC、VAR_PUBLIC_RETAIN的分類進行設置。

實例

■實例含義

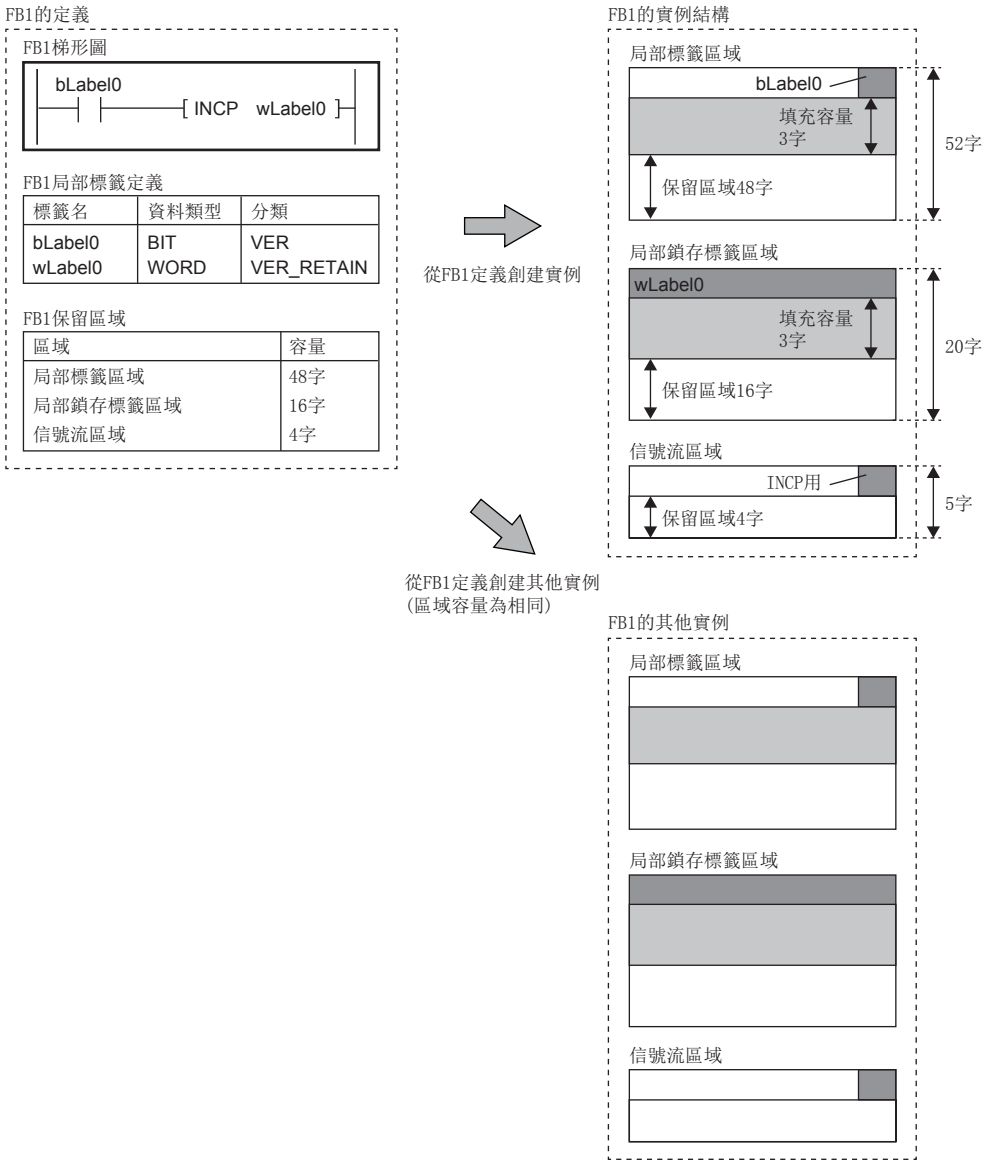
FB的實例是指在FB的定義的基礎上分配的標籤。從一個FB定義可以創建多個實例。

實例是由下述配置。

項目	內容
局部標籤區域	分配FB的局部標籤的區域。
局部鎖存標籤區域	分配FB的鎖存屬性的局部標籤的區域。
信號流區域	分配FB定義內的指令所使用的信號流的區域。

例

實例的配置(子程式型FB的示例)



局部標籤區域及局部鎖存標籤區域是以4字為單位確保標籤的使用區域，上述的示例為以3字(填充容量)進行確保。

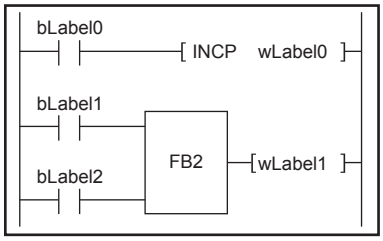
各區域的保留區域被確保。保留區域為透過轉換或RUN中寫入維持標籤的分配不變，添加/更改參閱局部標籤或信號流記憶體的指令、FB實例時所需的區域。無法確保相應於添加/更改對象的資料類型的區域的情況下，必須進行全轉換(再分配)。

例

將FB進行嵌套後的實例的配置(有更改FB2的保留區域的情況下)

FB1的定義

FB1梯形圖



FB1局部標籤定義

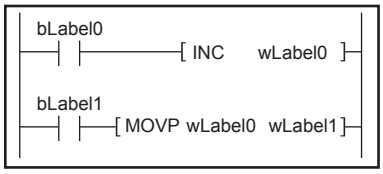
標籤名	資料類型	分類
bLabel0	BIT	VER
bLabel1	BIT	VER
bLabel2	BIT	VER
wLabel0	WORD	VER_RETAIN
wLabel1	WORD	VER
FB2_a	FB2	VER

FB1保留區域

區域	容量
局部標籤區域	48字
局部鎖存標籤區域	16字
信號流區域	4字

FB2的定義

FB2梯形圖



FB2局部標籤定義

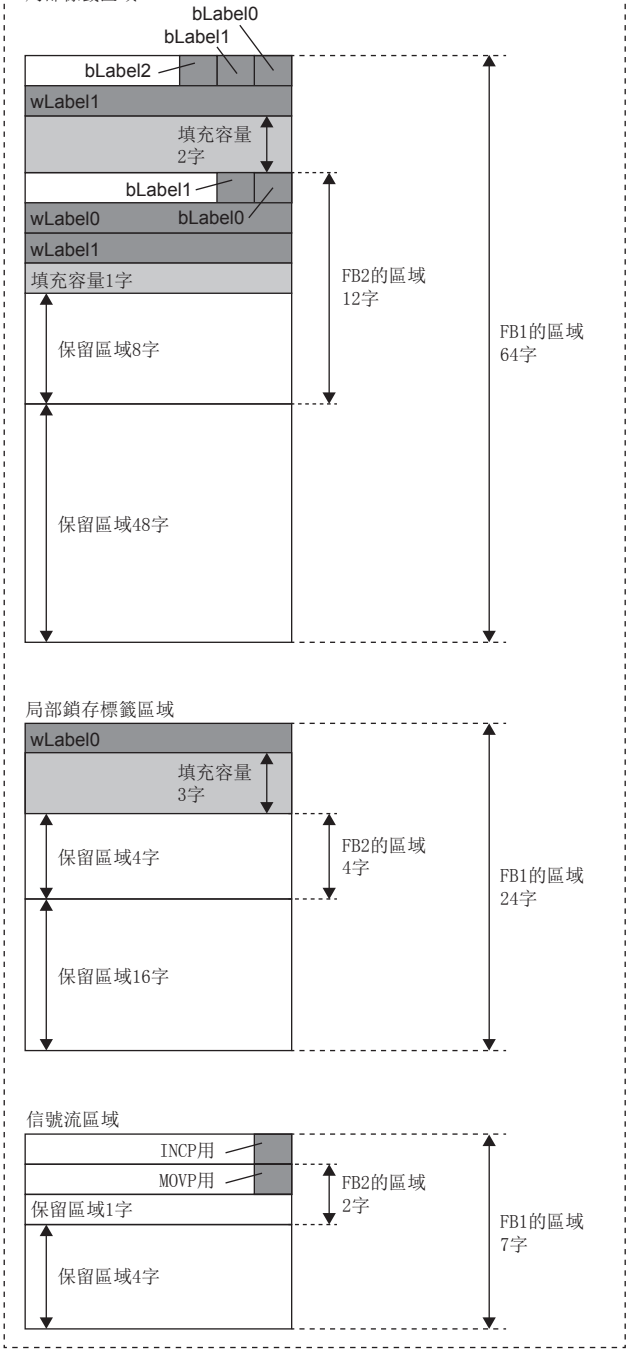
標籤名	資料類型	分類
bLabel0	BIT	VER_INPUT
bLabel1	BIT	VER_INPUT
wLabel0	WORD	VER
wLabel1	WORD	VER_OUTPUT

FB2保留區域

區域	容量
局部標籤區域	8字
局部鎖存標籤區域	4字
信號流區域	1字

從FB1定義創建實例

FB1的實例結構



作為局部標籤進行聲明的FB2的實例，將被確保於聲明源FB1的局部標籤區域、局部鎖存標籤區域、信號流區域。

在上述示例中對FB1添加/更改FB類型的局部標籤的情況下，由於保留區域的容量在局部標籤區域為48字、在局部鎖存標籤區域為16字、在信號流區域為4字，因此如欲添加/更改具有超出上述任一區域的FB的情況下，必須進行全轉換(再分配)。

欲在不進行全轉換(再分配)且維持分配不變的情況下添加/更改局部標籤或FB實例時，應預先在保留區域中確保欲添加/更改的容量。關於保留區域的設置方法，請參閱下述手冊。

GX Works3 操作手冊

■創建實例

為了使用FB，需要創建實例。

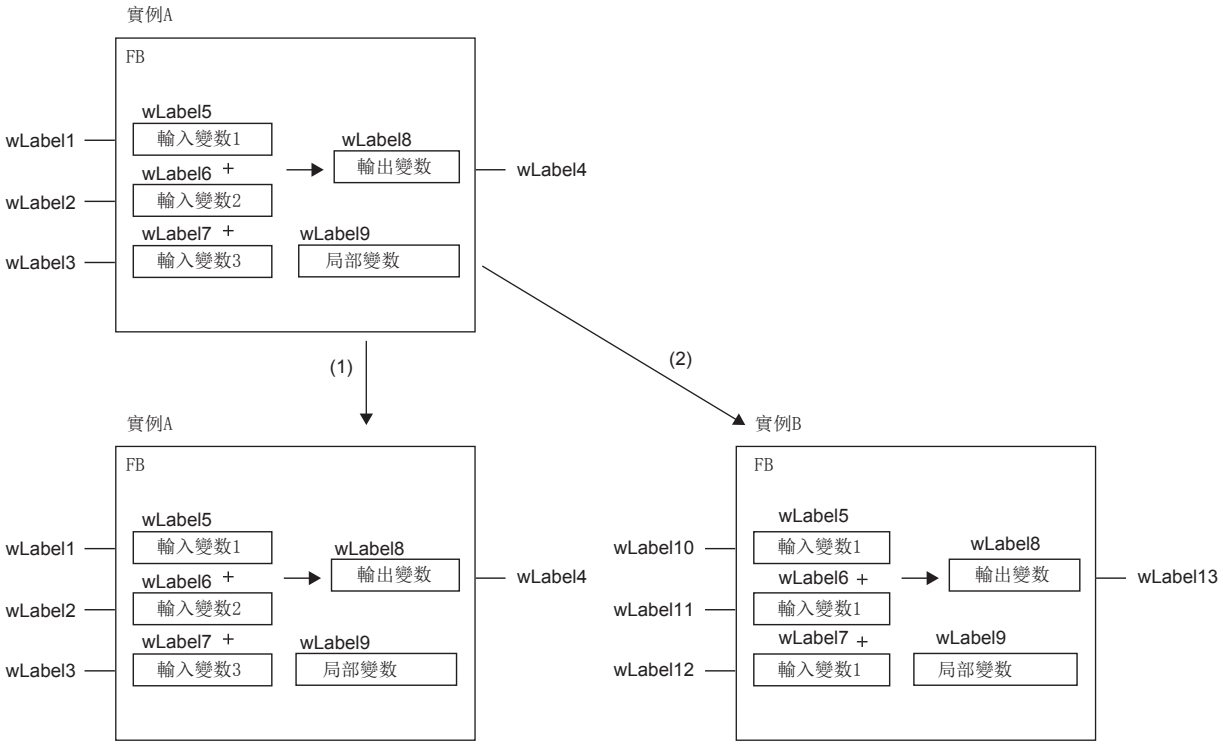
透過創建FB的實例，可以從程式塊及其他的FB調用並使用。

實例透過全局標籤或局部標籤進行聲明。

標籤的類型	實例的類型	分類
全局標籤	全局FB	VAR_GLOBAL
局部標籤*1	局部FB	VAR

*1　可以作為程式塊或FB的局部標籤進行聲明。在函數中無法聲明。

在一個程式部件中，同一個FB可以在不同的實例中使用。



(1) 相同實例的情況下，使用相同的內部變數。

(2) 不同實例的情況下，使用不同區域的內部變數。

■實例的容量

關於實例的各資料區域的容量，計算方法如下所示。

• 局部標籤區域的容量

實例的局部標籤區域的容量＝鎖存屬性以外的局部標籤的資料容量(總和)+保留區域容量

項目	內容
局部標籤容量(鎖存屬性的局部標籤除外)	作為局部標籤使用的資料的總和。 實際使用的區域的容量，將根據標籤的記憶體分配而不同。標籤的記憶體分配的詳細內容，請參閱下述手冊。 📖GX Works3 操作手冊
保留區域容量	預設為48字。可以以4字為單位進行設置。

• 局部鎖存標籤區域的容量

實例的局部鎖存標籤區域的容量＝鎖存屬性的局部標籤的資料容量(總和)+保留區域容量

項目	內容
鎖存屬性局部標籤容量	作為鎖存屬性的局部標籤使用的資料的總和。 實際使用的區域的容量，將根據標籤的記憶體分配而不同。標籤的記憶體分配的詳細內容，請參閱下述手冊。 📖GX Works3 操作手冊
保留區域容量	預設為16字。可以以4字為單位進行設置。

• 信號流區域的容量

宏型FB的情況下為與程式的步數同點數。

子程式型FB的情況下，如下所示。

實例的局部鎖存標籤區域的容量＝鎖存屬性的局部標籤的資料容量(總和)+保留區域容量

項目	內容
信號流區域容量	FB定義內的指令所使用的信號流的區域的總和。
保留區域容量	預設為4字。可以以1字為單位進行設置。

要點

應在保留區域容量中，設置對透過RUN中寫入預期會添加/更改的局部標籤或信號流記憶體進行參照的指令、FB實例的容量。關於設置方法，請參閱下述手冊。

📖GX Works3 操作手冊

對於透過RUN中寫入添加/更改的容量，在無法確保保留區域容量的情況下，將無法RUN中寫入而必須進行全轉換(再分配)。

關於如下的FB，透過將保留區域設置為小於預設的值，將能更有效率的使用CPU模組的記憶體。

• 不添加/更改局部標籤或更改程式的已調試完成的子程式型FB

• 大量進行實例聲明的子程式型FB

初始值的設置

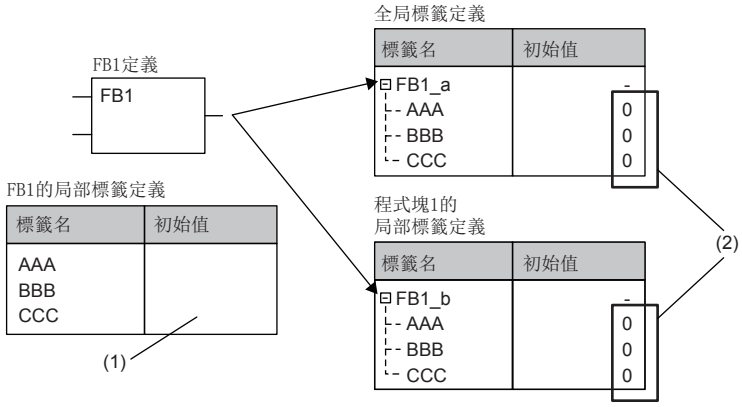
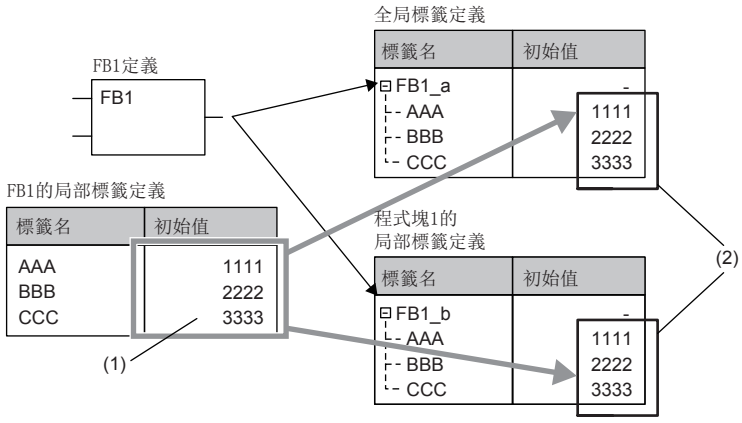
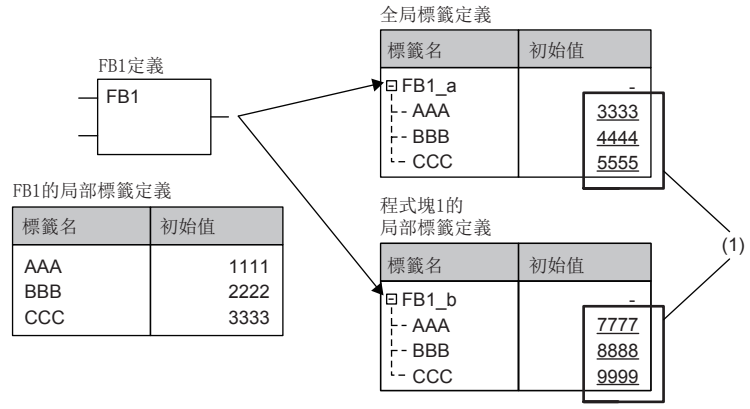
■FB的局部標籤的初始值

FB的局部標籤可以設置FB定義及各實例的初始值。

初始值可設置的局部標籤的分類為VAR、VAR_RETAIN、VAR_INPUT、VAR_OUTPUT、VAR_OUTPUT_RETAIN、VAR_PUBLIC、VAR_PUBLIC_RETAIN。

■實例的初始值

實例的初始值的類型如下所示。

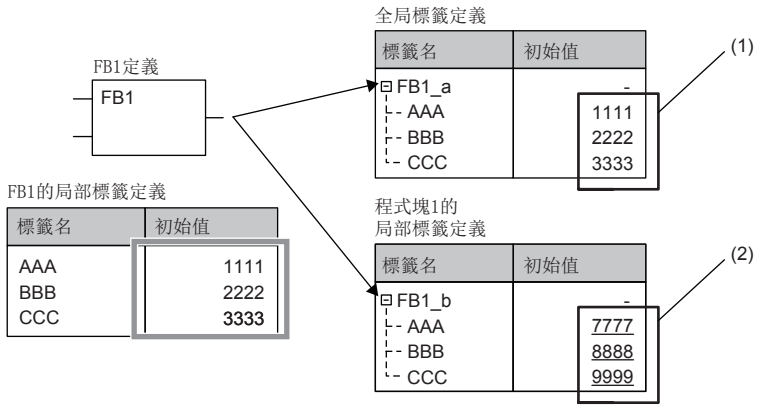
類型	內容
預設初始值	為各資料類型中預先決定的初始值。FB的局部標籤中未設置初始值的情況下，預設初始值也適用。  (1) FB中定義的局部標籤中未設置初始值。 (2) 適用預設初始值。
FB定義初始值	是在定義FB的局部標籤時設置的初始值。設置了FB定義初始值的情況下，對於所有的實例，相同的定義初始值也適用。  (1) FB1定義中的局部標籤中有設置初始值。 (2) FB1的所有實例以相同的定義初始值進行初始化。
實例初始值	是在包含全局標籤及程式塊的局部標籤定義的實例中所設置的初始值。  (1) 可以在FB1定義的各實例中設置初始值。

FB的初始值可以設置FB定義初始值與實例初始值兩者。
設置了這兩個初始值的情況下，適用的初始值的優先順序如下所述。

優先順序	類型
高 ↑ ↓ 低	實例初始值
	FB定義初始值
	預設初始值

要點

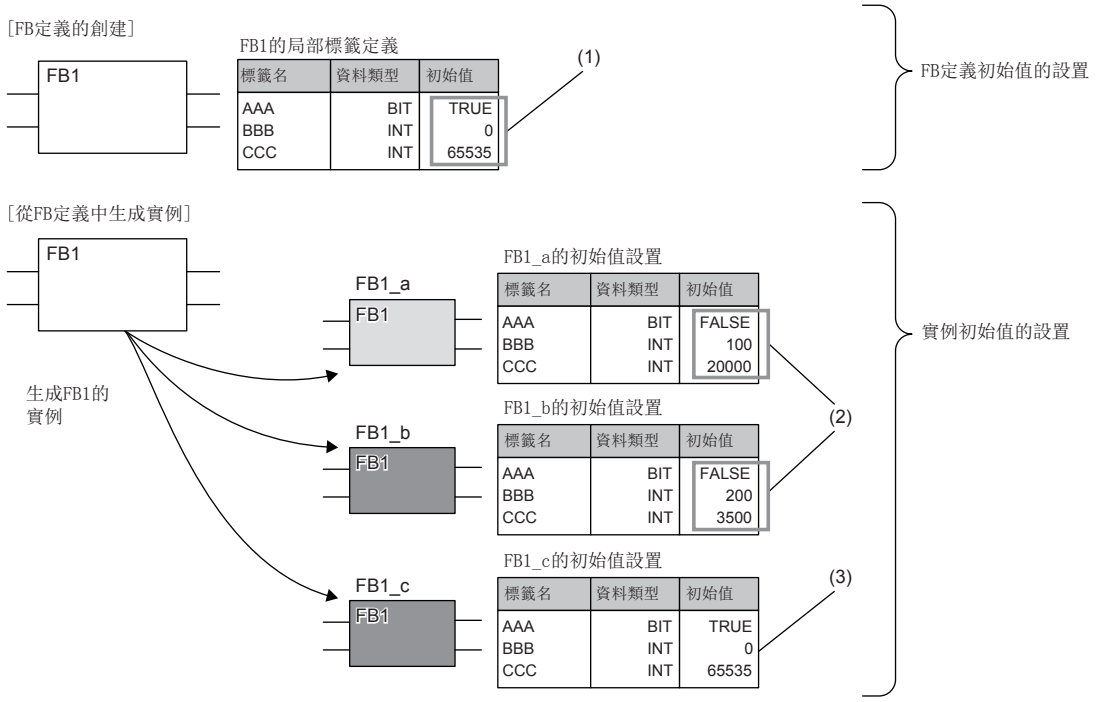
創建兩個設置了FB定義初始值的FB的實例，其中只有一個設置了實例初始值的情況下，FB定義初始值適用於未設置實例初始值的實例，實例初始值適用於設置了實例初始值的實例。



(1)未設置實例初始值的情況下，按照定義初始值進行初始化。
(2)設置了實例初始值的情況下，按照實例初始值進行初始化。

■使用示例

FB的初始值的使用示例如下所示。



(1) 在全部實例中設置通用的初始值。
(2) 可以在各實例中設置個別的初始值。
(3) 未設置個別的初始值的情況下，通用的初始值也適用。

關於EN/ENO

FB與函數一樣，可以透過帶EN(允許輸入)、ENO(允許輸出)進行執行處理的控制。

☞15頁 關於EN/ENO

調用帶EN/ENO的FB的實例時，必須對EN分配實引數。

程式的創建

創建FB的程式的情況下，實施下述操作。

[導航視窗]⇨[FB/FUN]⇨右擊⇨[新增資料]
在“基本設定”的“資料類型”中選擇“FB”

已創建的程式儲存在FB檔案中。

[CPU參數]⇨[程式設定]⇨[FB/FUN檔案設定]

一個FB檔案中最多可以儲存64個創建的程式。

關於與程式創建相關的內容，請參閱下述手冊。

項目	參照目標
FB的創建方法	📖GX Works3 操作手冊
可寫入CPU模組的FB/FUN檔案數	📖MELSEC iQ-R CPU模組用戶手冊(入門篇)

■程式的類型

FB有下述幾種類型，FB的程式本體的儲存方式不同。

- 宏型FB
- 子程式型FB

關於詳細內容，請參閱下述章節。

☞20頁 動作概要

模組FB、通用函數、通用FB無法進行上述選擇。

■固有屬性設置

創建FB的程式的情況下，可以進行下述設置。(📖GX Works3 操作手冊)

項目	內容
使用MC/MCR控制EN*1	選擇“是”的情況下，在EN的控制中使用MC/MCR指令。選擇“否”的情況下，在EN的控制中使用CJ指令。FB中正在使用上升沿/下降沿指令的情況下，應選擇“是”。另外，根據選擇的不同，FB中正在使用的定時器/計數器和OUT指令的動作也會有所不同。關於詳細內容，請參閱下述章節。 ☞ 142頁 使用MC/MCR指令控制EN時的動作
使用EN/ENO	選擇“是”的情況下，為具有EN/ENO的FB，即使不將EN/ENO標籤登錄至局部標籤也能夠在程式中使用。選擇“否”的情況下，為不具有EN/ENO的FB。 關於EN/ENO的詳細內容，請參閱下述章節。 ☞ 28頁 關於EN/ENO

*1 可以在“使用EN/ENO”中選擇了“是”的情況下進行選擇。但是，根據CPU模組及GX Works3的版本，在“FB的類型”中選擇了“子程式類型”的情況下無法進行選擇。關於支援的CPU模組及GX Works3的版本，請參閱下述手冊。
📖MELSEC iQ-R CPU模組用戶手冊(應用篇)

■可使用的元件/標籤

在FB程式中可使用的元件及標籤一覽如下所示。

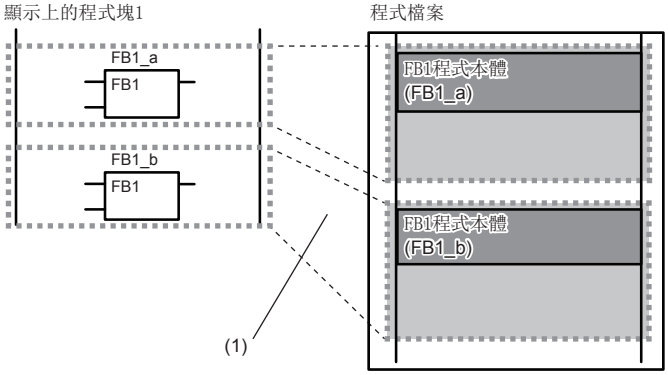
○：可以使用、 △：只可以在指令中使用(禁止作為表示程式的步的標籤使用)、 ×：禁止使用

元件/標籤的類型		能否使用
標籤(指針型以外)	全局標籤	○
	局部標籤	○
標籤(指針型)	指針型全局標籤	△
	指針型局部標籤	○
元件	全局元件	○
	局部元件	×
指針	全局指針	△
	局部指針	×

步數(宏型FB)

■調用側

調用宏型FB的情況下，編譯時展開調用對象的程式本體。



(1) 程式本體在多個調用位置展開。

■程式本體

FB程式本體的步數與普通的程式一樣為指令步數的總計。

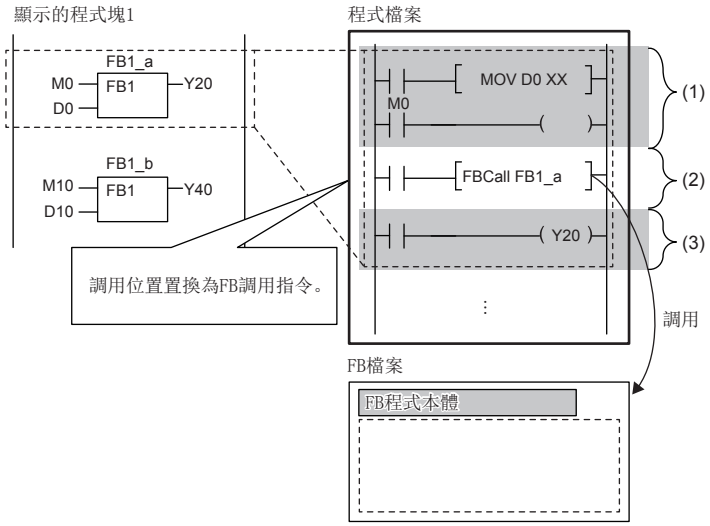
關於各指令的步數，請參閱下述手冊。

📖MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)

步數(子程式型FB)

■調用側

調用子程式型FB的情況下，在調用FB前後生成FB的引數的交接處理。



(1) 引數交接(輸入引數，輸入輸出引數)
(2) FB1程式本體調用
(3) 引數交接(輸出引數，輸入輸出引數)

• 引數交接

引數交接中使用的指令根據引數的分類及引數的資料類型而不同。在引數交接中使用的指令如下所示。

引數的分類	資料類型	使用指令	步數
VAR_INPUT VAR_IN_OUT VAR_OUTPUT	位元	LD+OUT LD+MOVB (根據所使用的程式語言、函數的類型、輸入引數類型的組合，使用其中的一個。)	各指令步數的詳細內容，請參閱下述手冊。 📖MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)
	字[無符號]/位元串[16位元] 雙字[無符號]/位元串[32位元] 字[帶符號] 雙字[帶符號]	LD+MOV LD+DMOV	
	單精度實數	LD+EMOV	
	雙精度實數	LD+EDMOV	
	時間	LD+DMOV	
	字元串	LD+$MOV	
	字元串[Unicode]	LD+$MOV_WS	
	陣列、結構體	LD+BMOV	

• 程式本體調用

FB程式本體的調用需要10步。

• EN/ENO

EN/ENO所需步數如下所示。

項目	步數
EN	3
ENO	2

要點

步數根據下述條件增減。

- FB調用的實引數被變址修飾的情況下
- 指定元件地址超過16位元的情況下
- 位數指定的情況下

■程式本體

FB程式本體的步數與普通的程式一樣為指令步數的總計。

關於各指令的步數，請參閱下述手冊。

📖MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)

3.4 注意事項

使用函數的情況下

■全局指針/局部指針/指針型的全局標籤

不能使用局部指針、指針型的全局標籤。

全局指針不能作為表示程式的步數的標籤使用。

使用FB的情況下

■全局指針/局部指針/指針型的全局標籤

不能使用局部指針。

全局指針、指針型的全局標籤不能作為表示程式的步數的標籤使用。

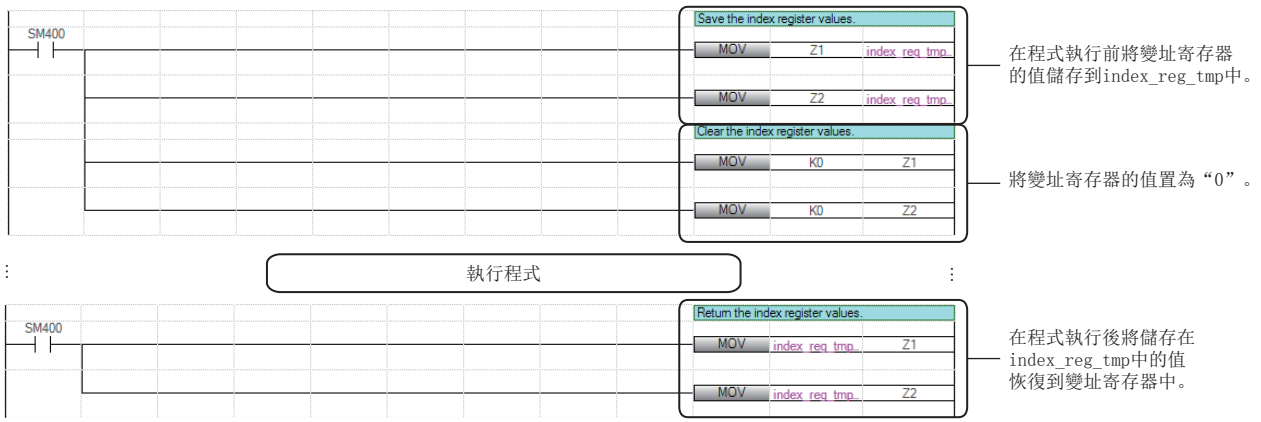
■使用變址寄存器的情況下

在FB的程式中使用變址寄存器的情況下，為了保護變址寄存器的值，必須要有梯形圖與恢復梯形圖。

變址寄存器儲存時透過將變址寄存器的值設為0，可以防止變址修飾的整合性檢查(元件編號是否超過了元件範圍)的出錯。

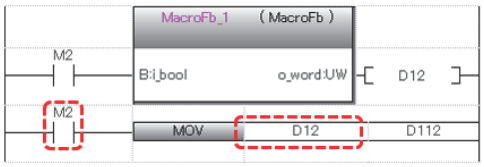
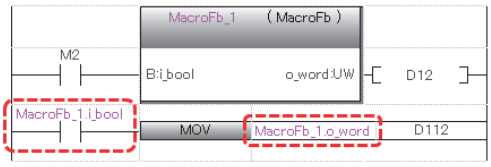
例

執行程式前儲存變址寄存器Z1、Z2，在執行程式後恢復已儲存的變址寄存器的情況下

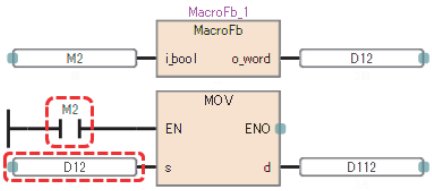
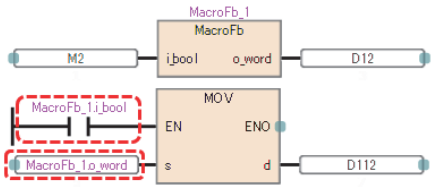


■關於宏型FB的引數

應在宏型FB的程式本體以外使用已用於引數交接的元件/標籤，而不是FB的引數。如果在宏型FB的程式本體以外使用宏型FB的引數，有可能會變為預料外的值。

程式語言	已用於引數交接的元件/標籤的情況下	變為預料外的值的情況下
梯形圖語言		

ST語言	`MacroFb_1(i_bool:= M2 ,o_word=> D12);`
	`IF TRUE = M2 THEN`
	`  D112 := D12 ;`
	`END_IF;`
		`MacroFb_1(i_bool:= M2 ,o_word=> D12);`
		`IF TRUE = MacroFb_1.i_bool THEN`
		`  D112 := MacroFb_1.o_word ;`
		`END_IF;`

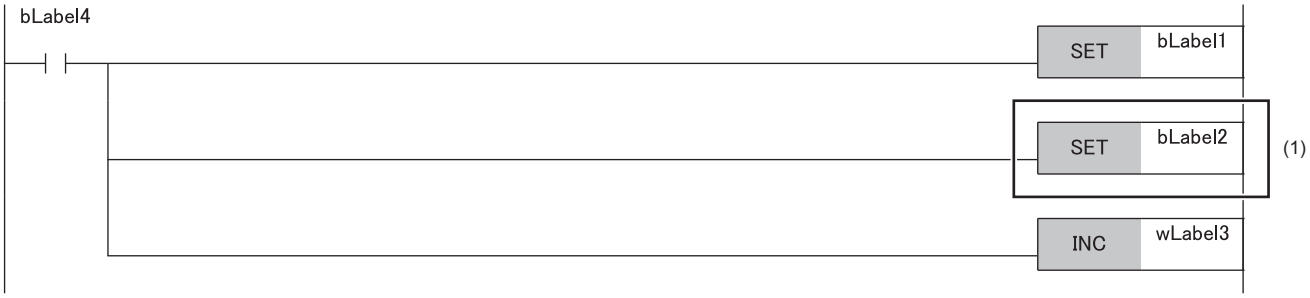
FBD/LD語言		

■在宏型FB的VAR_INPUT、VAR_OUTPUT或VAR_IN_OUT中發生轉換出錯的情況下

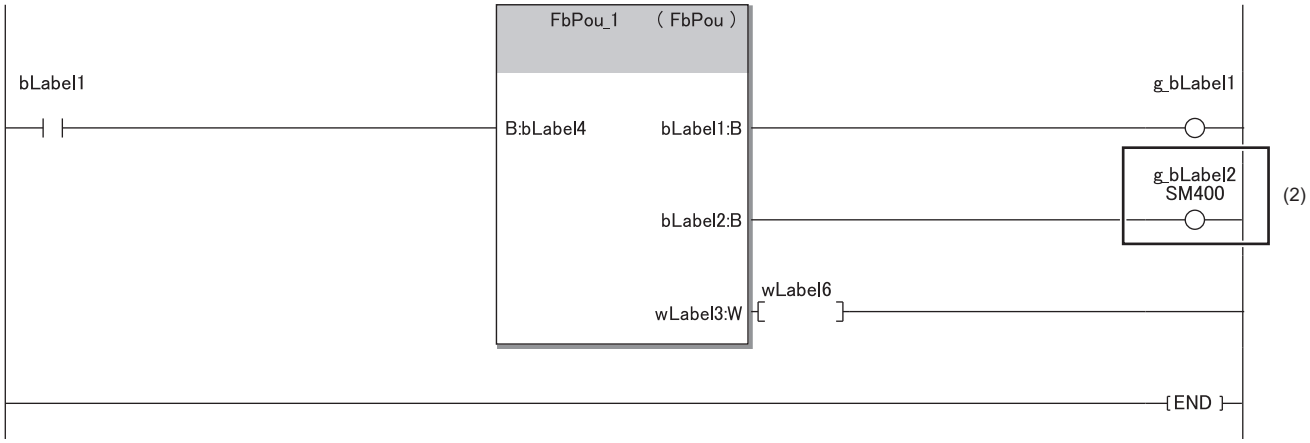
在宏型FB內的VAR_INPUT、VAR_OUTPUT或VAR_IN_OUT中發生轉換出錯的情況下，出錯的原因可能在於FB的調用源的程式塊或FB。此時，應確認FB的調用源程式塊或FB的輸入或輸出。

例

在宏型FB(FbPou)內的VAR_OUTPUT中發生了轉換出錯(1)的情況下



(1)沒有異常的情況下，應在調用源程式塊中確認對應FB的輸入或輸出(2)。



在上述示例中，由於將FB的輸出變數傳遞至了不可寫入的標籤/元件，因此發生了轉換出錯。

■智能功能模組的起始I/O No.的指定

訪問智能功能模組的緩衝記憶體及輸入輸出信號的情況下，應使用變址寄存器指定起始輸入輸出編號。

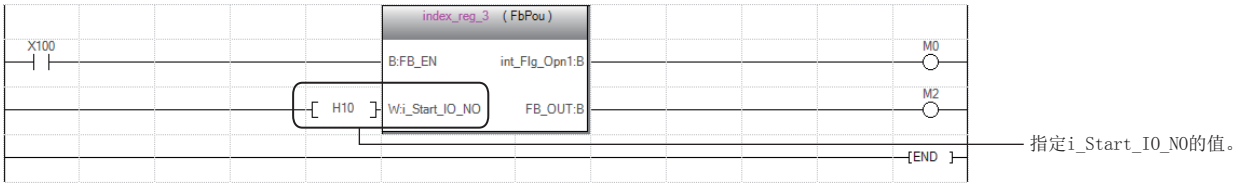
透過將起始輸入輸出編號作為輸入變數進行接收，在安裝位置不同的多個智能功能模組中，可以不用更改起始輸入輸出編號即可使用通用FB。

例

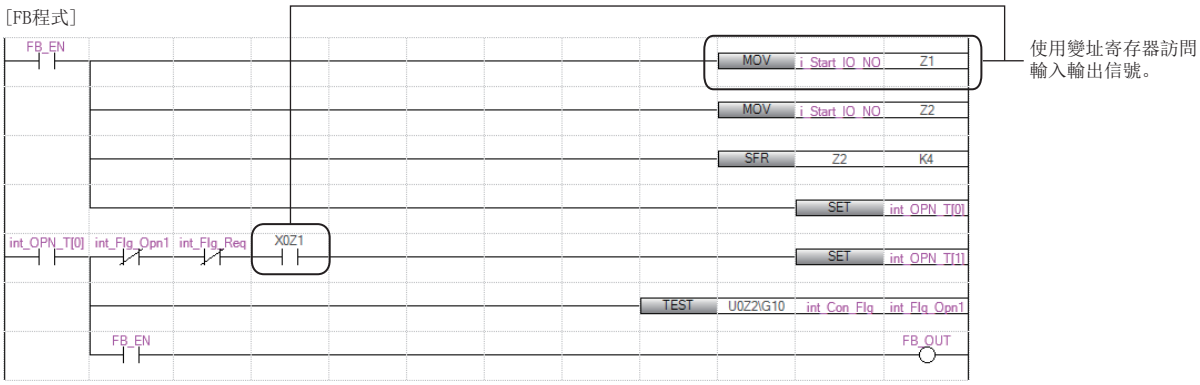
訪問智能功能模組的輸入輸出信號的情況下

透過使用變址寄存器，可以訪問對象智能功能模組的輸入輸出信號。

［順控程式］



［FB程式］

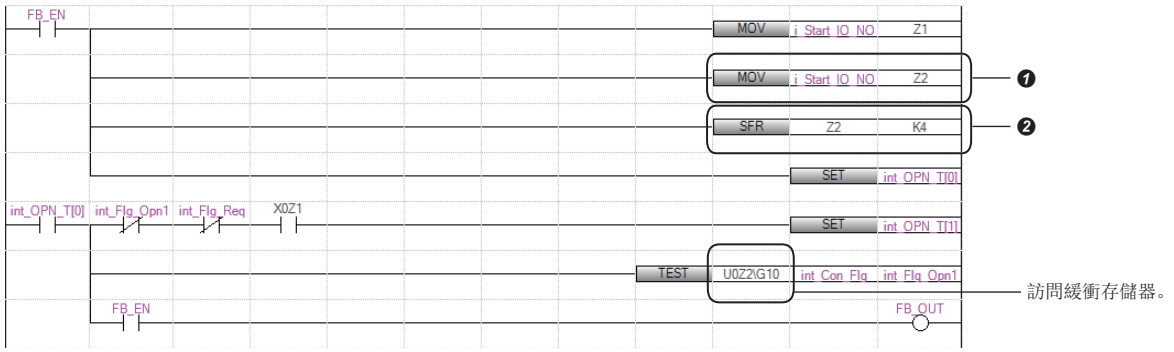


例

訪問智能功能模組的緩衝記憶體的情況下

❶ 將對象智能功能模組的起始輸入輸出編號輸入到變址寄存器中。

❷ 使用SFR指令把值向右移四位元，或使用將值除以16後所得的商。



■模組FB的限制事項

使用模組FB的情況下，存在以下限制事項。

- 在MC指令至MCR指令之間調用模組FB的情況下，請勿將MC指令的觸點OFF。
- 請勿進行會使得在CJ指令、SCJ指令、JMP指令中無法調用模組FB的跳轉。
- 在子程式內調用模組FB的情況下，每次掃描應執行1次子程式。此外，請勿使用FCALL(P)指令、EFCALL(P)指令、XCALL指令執行子程式的非執行處理。
- 在中斷程式或事件執行類型程式內，請勿調用模組FB。
- 在FOR～NEXT指令間、直接ST或ST語言的控制語句內(IF語句、FOR語句、CASE語句等)，請勿調用模組FB。
- 對於調用模組FB的程式，請勿使用程式控制用指令(PSTOP(P)指令、POFF(P)指令、PSCAN(P)指令)。

■更改模組FB的對象模組的管理CPU的情況下

在工程工具的“I/O分配設定”中，如果將“管理CPU設定”更改為其他機號CPU，將會刪除程式上的模組FB。

在更改“管理CPU設定”前，應將程式複製到其他機號CPU工程中。

■關於模組FB的動作參數的更改

模組FB的輸入標籤及輸出標籤以外的動作參數(外部變數)的更改可以在標籤設置中更改。

- 模組FB的實例為局部標籤的情況下，可以在“區域標籤設定”中更改。

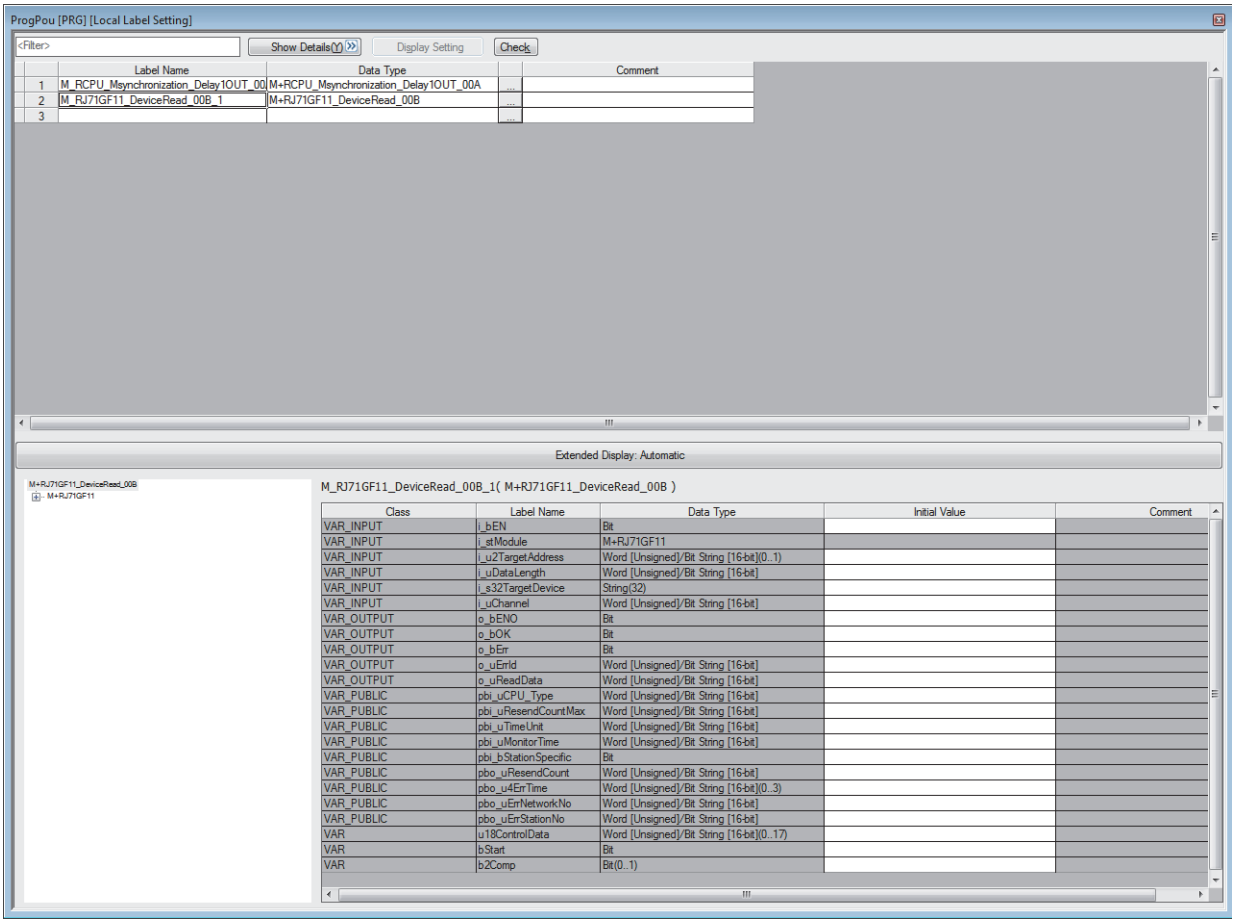
[導航視窗]⇨[程式]⇨執行類型⇨程式檔案⇨程式塊⇨[區域標籤]

- 模組FB的實例為全局標籤的情況下，可以在“全域標籤設定”中更改。

[導航視窗]⇨[標籤]⇨[全域標籤]

例

設置局部標籤的情況下



在“初始值”欄中設置動作參數。

3.5 使用安全程式的情況下

將在安全程式中使用的函數稱為安全函數、FB稱為安全FB。本節中未記載的內容與普通的函數、FB相同。(☞ 14頁 函數(FUN)、 ☞ 19頁 功能塊(FB))

安全函數(安全FUN)

以下對安全函數的有關內容進行說明。

程式的創建

■可使用的元件及標籤

在安全函數中可使用的元件及標籤如下所示。

○：可以使用、 ×：禁止使用

元件/標籤的類型		能否使用
標籤(指針型以外)	全局標籤	×
	局部標籤	×
	常規/安全共享標籤	×
	安全全局標籤	×
	安全局部標籤	○*1
標籤(指針型)	指針型全局標籤	×
	指針型局部標籤	×
元件	全局元件	×
	局部元件	×
	安全全局元件	○
	安全局部元件	×
指針	全局指針	×
	局部指針	×

*1 不能使用下述資料類型。
定時器、累計定時器、計數器、長定時器、長累計定時器、長計數器

步數

■引數交接

調用安全函數的情況下，在調用安全函數前後生成安全函數的引數的交接處理。引數交接中使用的指令根據引數的分類及引數的資料類型而不同。在引數交接中使用的指令如下所示。

○：可以使用、 ×：禁止使用

引數的分類	資料類型	使用指令	能否使用
VAR_INPUT	位元	LD+OUT	○
	字[無符號]/位元串[16位元] 雙字[無符號]/位元串[32位元] 字[帶符號] 雙字[帶符號]	LD+MOV LD+DMOV	○
	單精度實數	LD+EMOV	×
	雙精度實數	LD+EDMOV	×
	時間	LD+DMOV	×
	字元串	LD+$MOV	×
	字元串[Unicode]	LD+$MOV_WS	×
	陣列、結構體	LD+BMOV	○

關於各指令的步數，請參閱下述手冊。

📖MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)

安全功能塊(安全FB)

以下對安全FB有關內容進行說明。

實例

■實例的配置

安全FB的實例由下述所配置。

○：可以使用、 ×：禁止使用

項目	內容	能否使用
局部標籤區域	分配FB的局部標籤的區域。	○
局部鎖存標籤區域	分配FB的鎖存屬性的局部標籤的區域。	×
信號流區域	分配FB定義內的指令所使用的信號流的區域。	○

程式的創建

■可使用的元件/標籤

在安全FB中可使用的元件及標籤如下所示。

○：可以使用、 ×：禁止使用

元件/標籤的類型		能否使用
標籤(指針型以外)	全局標籤	×
	局部標籤	×
	常規/安全共享標籤	○
	安全全局標籤	○
	安全局部標籤	○
標籤(指針型)	指針型全局標籤	×
	指針型局部標籤	×
元件	全局元件	×
	局部元件	×
	安全全局元件	○
	安全局部元件	×
指針	全局指針	×
	局部指針	×

步數(子程式型FB)

■引數交接

調用安全FB的情況下，在調用安全FB前後生成安全FB的引數的交接處理。引數交接中使用的指令根據引數的分類及引數的資料類型而不同。在引數交接中使用的指令如下所示。

○：可以使用、 ×：禁止使用

引數的分類	資料類型	使用指令	能否使用
VAR_INPUT VAR_IN_OUT	位元	LD+OUT	○
	字[無符號]/位元串[16位元] 雙字[無符號]/位元串[32位元] 字[帶符號] 雙字[帶符號]	LD+MOV LD+DMOV	○
	單精度實數	LD+EMOV	×
	雙精度實數	LD+EDMOV	×
	時間	LD+DMOV	×
	字元串	LD+$MOV	×
	字元串[Unicode]	LD+$MOV_WS	×
	陣列、結構體	LD+BMOV	○

關於各指令的步數，請參閱下述手冊。

📖MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)

4 標籤

標籤是指在輸入輸出資料及內部處理中指定任意字元串的變數。
在程式中使用標籤時，可無需在意元件及緩衝記憶體容量創建程式。
因此，如果使用了標籤的程式，則也可以在模組配置不同的系統中簡單再利用。
詳細內容，請參閱下述手冊。
📖MELSEC iQ-R CPU模組用戶手冊(應用篇)

備忘録

5 梯形圖語言

RnCPU RnENCPU RnPCPU (過程) RnPCPU (二重化) RnPSFCPU (常規) RnPSFCPU (安全) RnSFCPU (常規) RnSFCPU (安全)

是在由觸點與線圈配置的梯形圖中，表示在串聯與並聯的組合中由AND/OR組成的邏輯運算，並記述順控程式控制的語言。

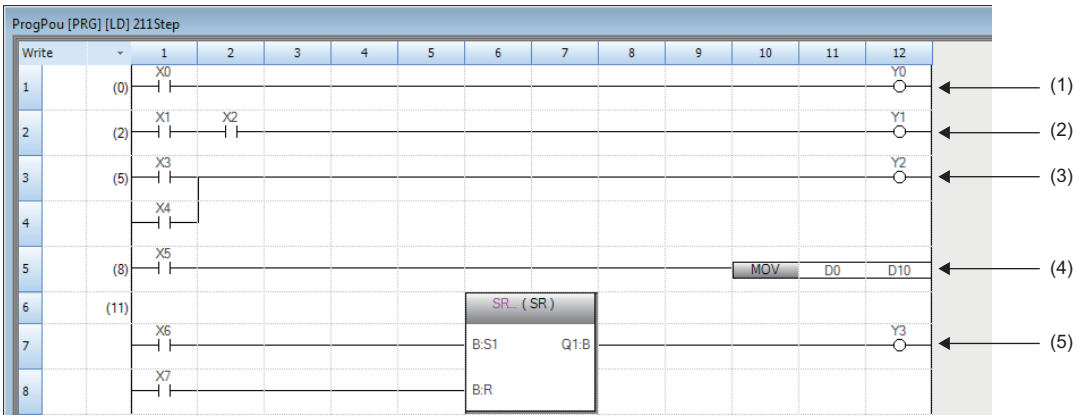
要點

在本章中，對梯形圖語言的動作及規格有關內容進行說明。關於創建梯形圖程式時的操作方法，請參閱下述手冊。

GX Works3 操作手冊

5.1 配置

梯形圖語言中，可以創建下述梯形圖。

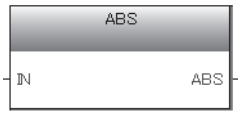
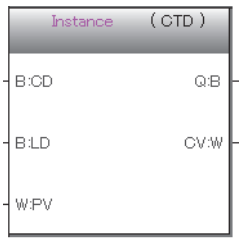


(1) 由觸點與線圈組成的梯形圖
(2) 以串聯配置的梯形圖
(3) 以並聯配置的梯形圖
(4) 使用了指令的梯形圖
(5) 使用了通用函數/FB的梯形圖

梯形圖符號

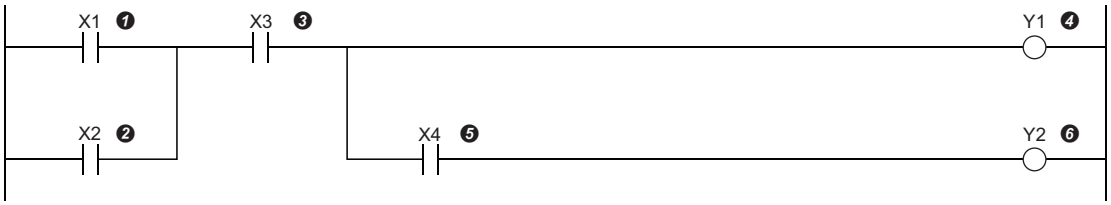
可以在梯形圖語言的程式中使用的梯形圖符號如下所示。

項目		內容
常開觸點		指定元件或標籤為ON時導通。
常閉觸點		指定元件或標籤為OFF時導通。
上升沿脈衝		指定元件或標籤上升沿時(OFF→ON)導通。
下降沿脈衝		指定元件或標籤下降沿時(ON→OFF)導通。
上升沿脈衝否定		指定元件或標籤為OFF、ON以及下降沿時(ON→OFF)導通。
下降沿脈衝否定		指定元件或標籤為OFF、ON以及上升沿時(OFF→ON)導通。

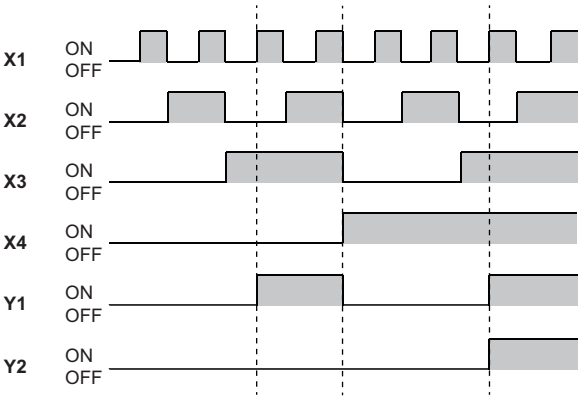
項目		內容
運算結果上升沿脈衝化		運算結果上升沿時(OFF→ON)導通。運算結果在上升沿以外的情況不導通。
運算結果下降沿脈衝化		運算結果下降沿時(ON→OFF)導通。運算結果在下降沿以外的情況不導通。
反轉運算結果		對之前的運算結果反轉。
線圈		將運算結果輸出至指定元件或標籤。
指令		執行[]內指定的指令。
返回		如果超過了一個梯形圖列可創建的觸點數，則創建返回源的符號及返回目標的符號，執行梯形圖的返回。
函數		執行函數。 • 函數的創建方法(GX Works3 操作手冊) • 通用函數(MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇))
FB		執行FB。 • FB的創建方法(GX Works3 操作手冊) • 通用FB(MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)) • 模組FB(所使用模組的FB參考)

程式執行順序

按照下述的編號順序執行。



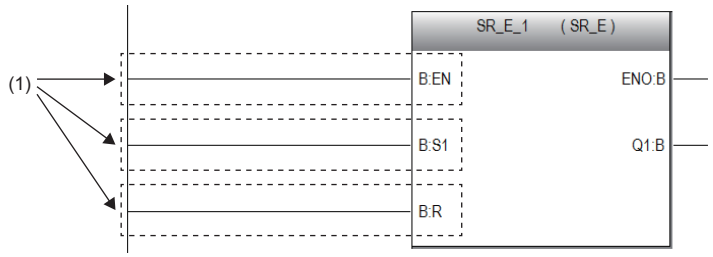
執行了上述程式的情況下，根據X1～X4的ON/OFF，Y1、Y2為ON的時機如下所示。



以梯形圖語言使用FB時的注意事項

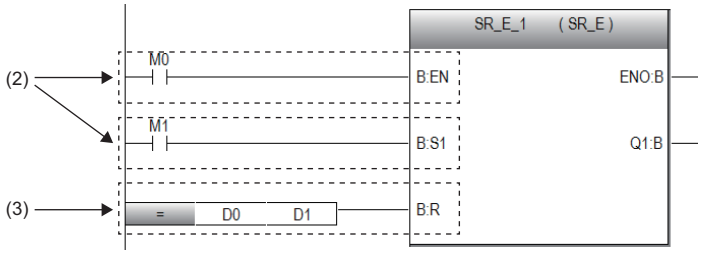
左母線與FB實例處於直接連接狀態時的注意事項

FB實例的輸入梯形圖中，EN及輸入變數(位元型)與左母線處於直接連接狀態時，ON/OFF狀態不變化。



(1)：ON/OFF狀態不變化。

使EN及輸入變數(位元型)的ON/OFF狀態變化時，應使用觸點或與觸點相當的指令。

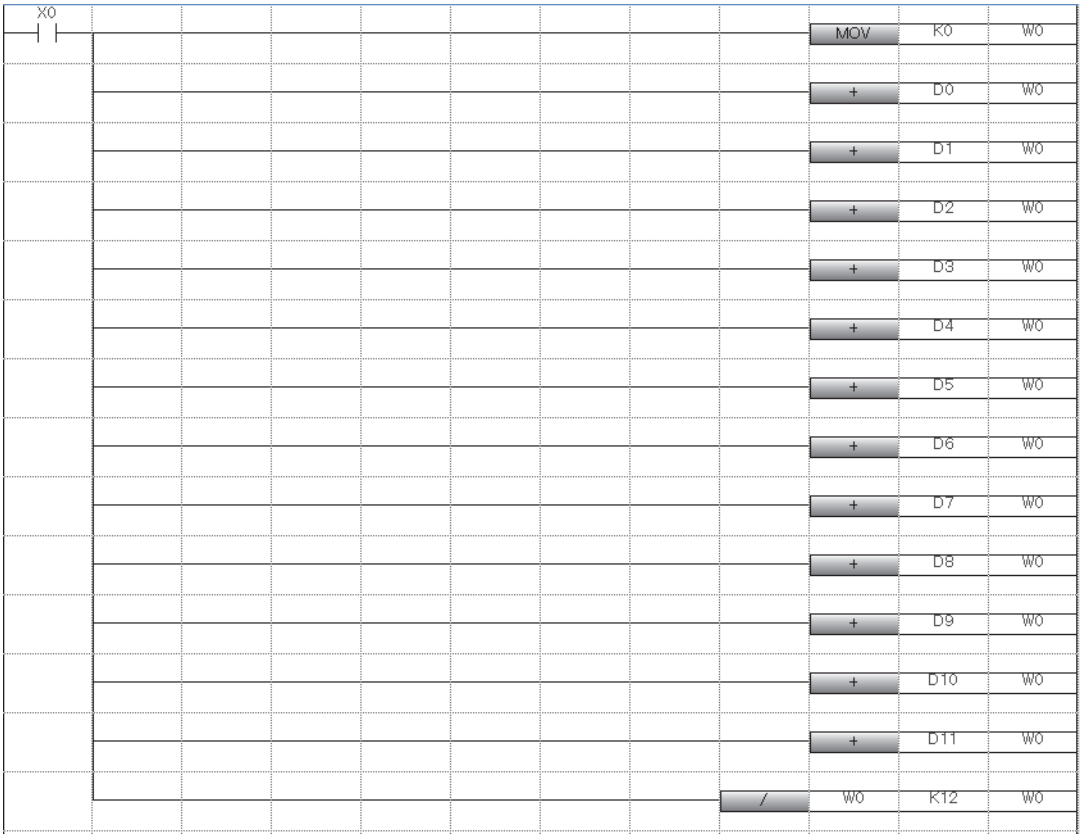


(2)：觸點
(3)：與觸點相當的指令

5.2 內嵌ST

內嵌ST是指在梯形圖編輯器內對在與線圈相當的指令單元格中顯示ST程式的內嵌ST盒子進行創建，執行編輯/監視的功能。
由此，可以輕鬆地在梯形圖程式內創建數值運算及字元串處理。

• 不使用內嵌ST情況下的程式



• 使用了內嵌ST情況下的程式

限制事項

- 在安全程式中，不能使用內嵌ST。
- 在SFC程式的Zoom編輯器內不能使用內嵌ST。

規格

關於內嵌ST中記述的ST程式的規格，請參閱ST語言的規格。

☞ 47頁 ST語言

注意事項

- 梯形圖程式的1列中只能創建1個內嵌ST。
- 在梯形圖程式的1列中無法使用函數/FB與內嵌ST盒兩者。
- 如果在觸點相應的指令位置創建內嵌ST盒，在線圈相應的指令位置也將創建內嵌ST盒。
- 內嵌ST內最多可輸入的字元數為2048個字元。(換列作為2個字元計數。)
- 如果在內嵌ST中使用“RETURN語句”，將不是結束程式塊的處理，而是結束內嵌ST盒內的處理。
- 如果在函數的程式中使用內嵌ST，將無法從內嵌ST中調用FB。
- 內嵌ST在轉換時使用CJ指令控制程式的動作。內嵌ST的觸點為OFF的情況下，內嵌ST內的處理不透過CJ指令執行。因此，即使透過內嵌ST內的代入語句變為ON的元件不執行內嵌ST，也保持輸出狀態。關於CJ指令的詳細內容，請參閱下述手冊。

📖MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)

- 在內嵌ST中，除了無法執行OUT、OUTH以外，參閱信號流存儲器上一次執行狀態的指令也無法使用。

📖MELSEC iQ-R CPU模組用戶手冊(應用篇)

5.3 聲明/注解

在梯形圖中，可以顯示聲明及注解。

聲明

透過使用聲明，可以對梯形圖塊添加注釋。透過進行添加，使處理等流程變得易懂。
聲明有列間聲明/P聲明/I聲明。
列間聲明可以在導航視窗的樹狀圖上顯示。

■列間聲明

對整個梯形圖塊添加注釋。

■P聲明

對指針元件添加注釋。

■I聲明

對中斷指針元件添加注釋。

注解

透過使用注解，可以對程式中的線圈及指令添加注釋。
透過添加注解，使線圈及指令的內容等變得易懂。

聲明/注解的類型

作為聲明與注解的類型，有“全體”與“外圍”。

類型	種類	內容
全體	• 列間聲明 • P聲明 • I聲明 • 注解	說明文的字元串在轉換時將被編入程式內。 在1列中使用(2+字元數)步。
外圍	• 列間聲明 • P聲明 • I聲明 • 注解	說明文的字元串將不被編入程式內，而是作為程式的附加資訊保存。 在1列中使用1步。 在輸入的文本前自動添加*印記。

6 ST語言

RnCPU RnENCPU RnPCPU(過程) RnPCPU(二重化) RnPSFCPU(常規) RnPSFCPU(安全) RnSFCPU(常規) RnSFCPU(安全)

ST語言是關於邏輯記述方式所規定的國際標準IEC61131-3中所定義的語言。ST語言是具有與C語言等相似的語法結構的文本形式的程式語言。適用於對難以在梯形圖語言中表現的複雜處理進行程式設計的情況。

要點

在本章中，對ST語言的動作及規格有關內容進行說明。關於創建ST程式時的操作方法，請參閱下述手冊。
GX Works3 操作手冊

ST語言支援控制語法、運算式、功能塊(FB)、函數(FUN)，可以按如下方式記述。

例
透過條件語句進行的選擇分支、重複語句等控制語法

```
(*以線A～C進行控制*)
CASE 線 OF
    1:
    開始開關:= TRUE;(*傳送帶運行*)
    2:
    開始開關:= FALSE;(*傳送帶停止*)
    3:
    開始開關:= TRUE;(*傳送帶停止 警告*)
    ELSE
    警告指示燈:= TRUE;
END_CASE;
IF 開始開關= TRUE THEN (*傳送帶運行 處理100次*)
    FOR 處理次數 := 0 TO 100 BY 1 DO
      處理數:= 處理數 +1;
    END_FOR;
END_IF;
```

例
使用運算符(*、/、+、-、<、>、＝等)的表達式

```
D0 := D1 * D2 + D3 / D4 - D5 ;
IF D0 > D10 THEN
     D0 := D10 ;
END_IF;
```

例
定義的FB的調用

```
//FB資料名:LINE1_FB
//輸入變數:I_Test
//輸出變數:O_Test
//輸入輸出變數:IO_Test
//FB標籤名:FB1
FB1(I_Test:= D0 , O_Test => D1 ,IO_Test:= D100);
```

例
通用函數的調用

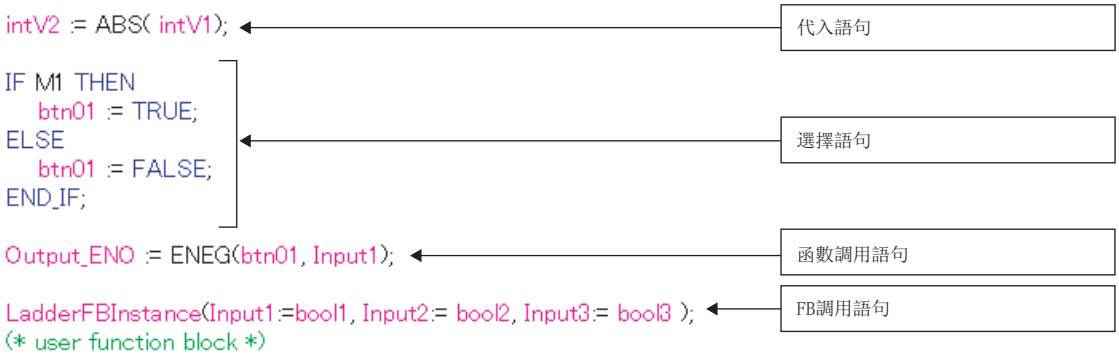
```
(* 將BOOL型資料轉換為INT型/DINT型資料 *)
wLabel2 := BOOL_TO_INT(bLabel1);
```

例
漢字等全形字元的使用

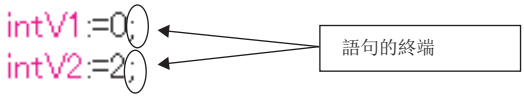
```
//油箱限制器ON時關閉閥，OFF時打開閥
IF 油箱限制器 = TRUE THEN
     閥:= FALSE ; (* 由於限制器變為ON，因此關閉閥 *)
    ELSE
     閥:= TRUE ; /* 由於限制器變為OFF，因此打開閥 */
END_IF;
```

6.1 配置

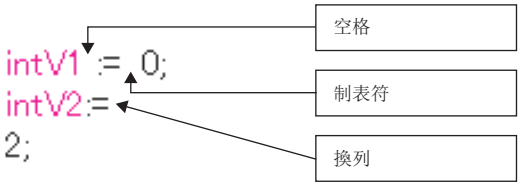
ST語言中的程式由運算符與語法所配置。



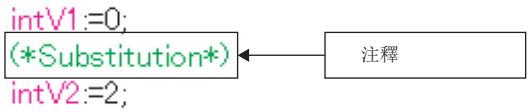
句子的終端必須添加“;”(分號)。



空格、制表符、換列可以插入到運算符及資料之間。



可以在程式中插入注釋。在注釋語句的前後記述“(*注釋語句*)”。



程式的配置要素

ST程式由以下要素所配置。

項目		示例	參照目標
分隔符		; 、(,)	☞49頁 分隔符
運算符		+、-、<、>、=	☞49頁 運算符
保留字	語法	IF、CASE、WHILE、RETURN	☞50頁 語法
	元件	X0、Y10、M100、ZR0	📖MELSEC iQ-R CPU模組用戶手冊(應用篇)
	資料類型	BOOL、DWORD	📖MELSEC iQ-R CPU模組用戶手冊(應用篇)
	通用函數	ADD、REAL_TO_STRING_E	📖MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)
常數		123、“abc”	☞59頁 常數
標籤		Switch_A	☞60頁 標籤與元件
注釋		(*置為ON*)	☞62頁 注釋
其他符號		半形空格、換列代碼、TAB代碼	—

- 分隔符、運算符號、保留字應用半形記述。
- 關於保留字的詳細內容，請參閱下述手冊。

📖GX Works3 操作手冊

分隔符

在ST語言中為了明確程式的結構，有下述的分隔符。

符號	內容
()	圓括弧式
[]	陣列要素編號的指定
.(句號)	結構體、FB構件的指定
,(逗號)	引數的分隔
:(冒號)	元件型指定符、CASE語句的分隔
;(分號)	語句的終端
″(雙引號)	Unicode字元串的記載
’(單引號)	字元串(ASCII、移位JIS)的記載
..(兩個句號)	整數範圍指定

運算符

在ST程式中使用的運算符，對象資料類型與運算結果的資料類型如下所示。

運算符	對象資料類型	運算結果類型
*、/、+、-	ANY_NUM*1	ANY_NUM
< 、>、<=、>=、 =、<>	ANY_ELEMENTARY*2	位元
MOD	ANY_INT	ANY_INT
AND、&、XOROR、NOT	ANY_BIT	ANY_BIT
**	ANY_REAL(底) ANY_NUM(指數)*1	ANY_REAL

*1　無法指定記載為“WORD#”或“DWORD#”的常數。(☞ 59頁 常數的記載方法)

*2　無法指定WSTRING型的Unicode字元串。

運算符的優先順序如下所示。

運算符	內容	示例	優先順序
()	圓括弧式	(2+3)*(4+5)	1
函數()	函數調用表達式	CONCAT(’AB’,’CD’)	2
-	符號的反轉	-10	3
NOT	位元型補數	NOT TRUE	
**	次方	3.0**4	4
*	乘法	10*20	5
/	除法	20/10	
MOD	取餘數運算	17 MOD 10	
+	加法	1.4+2.5	6
-	減法	3-2	
<、>、<=、>=	比較	10>20	7
=	一致	T#26h=T#1d2h	8
<>	不一致	8#15<>13	
&、AND	邏輯且	TRUE AND FALSE	9
XOR	排他邏輯或	TRUE XOR FALSE	10
OR	邏輯或	TRUE OR FALSE	11

- 在一個公式中有多個優先順序一樣的運算符的情況下，從左側的運算符開始運算。
- 一個公式中可以記述的運算符的使用個數最多為1024個。

語法

可在ST程式中使用的語法如下所示。

項目	內容	參照頁
代入語句	代入語句	50頁 代入語句
子程式控制語句	FB調用語句、函數調用語句	52頁 子程式控制語句
	RETURN語句	
選擇語句	IF語句(IF THEN、IF ELSE、IF ELSIF)	53頁 選擇語句
	CASE語句	
重複語句	FOR語句	53頁 重複語句
	WHILE語句	
	REPEAT語句	
	EXIT語句	

應用半形字元記述語法。

代入語句

書寫格式	內容	記述示例
<左邊>:=<右邊>;	具有將右邊公式的結果代入到左邊的標籤及元件中的功能。 必須使右邊公式的結果與左邊的資料類型相同。	intV1:=0; intV2:=2;

使用陣列型標籤及結構體標籤的情況下，應注意代入語句的左邊與右邊的資料類型。

陣列型標籤的情況下，資料類型與要素數必須在左邊與右邊相同。且請勿指定要素。

例

```
intAry1:=intAry2;
```

結構體標籤的情況下，必須使左邊與右邊的資料類型(結構體的資料類型)相同。

例

```
dutVar1:=dutVar2;
```

右邊為函數調用表達式的情況下，函數的返回值將被代入左邊。返回值被代入標籤的示例如下所示。

函數	記述示例
輸入變數為1個函數的情況下(例： ABS)	Output1 := ABS(Input1);
輸入變數為3個函數的情況下(例： MAX)	Output1 := MAX(Input1, Input2, Input3);
具有EN/ENO的函數(通用函數以外)的情況下(例： MAX_E)	Output1 := MAX_E(boolEN, boolENO, Input1, Input2, Input3);
通用函數的情況下(例： MOV)	boolENO := MOV(boolEN, Input1, Output1); (執行函數的結果變為ENO，第一引數(變數1)變為EN。)

■資料類型的自動轉換

在ST語言中，在記述不同的資料類型的代入及算術運算公式時，有可能會自動轉換資料類型。

例

自動轉換示例

```
dintLabel1 := intLabel1 ;
//代入語句： 將INT型(intLabel1)的值自動轉換為DINT型，代入左邊的DINT型(dintLabel1)
dintLabel1 := dintLabel2 + intLabel1 ;
//算術運算公式： 將INT型(intLabel1)的值自動轉換為DINT型，執行DINT型的加法運算
DMOV(TRUE, wordLabel1, dwordLabel1);
//指令、函數、FB調用語句： 將WORD型輸入引數(wordLabel1)的值自動轉換為DWORD型，執行傳送
```

類型轉換透過向代入語句、FB及函數(包括指令、通用函數、通用FB)的輸入引數交接(VAR_INPUT部)、算術運算公式進行。

僅從容量小的資料類型轉換至容量大的資料類型，以確保在類型轉換時資料不被丟失。類型轉換以資本類型中的下述資料類型為對象。

資料類型	內容
字[帶符號]	轉換後變為雙字[帶符號]的情況下，自動轉換為符號擴展值。 單精度實數或雙精度實數的情況下，自動轉換為與轉換前的整數相同的值。*1
字[無符號]/位元串[16位元]	轉換後變為雙字[無符號]/位元串[32位元]或雙字[帶符號]的情況下，自動轉換為零擴展值。*2 單精度實數或雙精度實數的情況下，自動轉換為與轉換前的整數相同的值。*1
雙字[帶符號]	轉換後為雙精度實數的情況下，自動轉換為與轉換前的整數相同的值。
雙字[無符號]/位元串[32位元]	
單精度實數	轉換後為雙精度實數的情況下，自動轉換為相同的值。

*1　將16位資料(字[帶符號]或字[無符號]/位元串[16位元])交接至資料類型為ANY_REAL的輸入引數的情況下，自動轉換為單精度實數。
*2　將字[無符號]/位元串[16位元]資料交接至資料類型為ANY32的輸入引數的情況下，自動轉換為雙字[無符號]/位元串[32位元]。

上述以外的資料類型，應使用類型轉換函數。

此外在下述情況下無法進行類型轉換，因此應使用類型轉換函數。

- 符號不同的整數型之間的類型轉換
- 資料丟失型之間的類型轉換

代入算術運算的結果時的注意事項，請參閱下述章節。

☞54頁 代入算術運算式結果的情況下

使用元件時的注意事項，請參閱下述章節。

☞61頁 使用元件時的數據類型自動轉換

子程式控制語句

■FB調用語句

書寫格式	內容
實例名(輸入變數1:=變數1,...輸出變數1=>變數2,...);	在實例名後，用“()”括住輸入變數、輸出變數的代入語句。 多個變數的情況下，各代入語句之間用“,”(逗號)隔開。
實例名.輸入變數1:=變數1; ⋮ 實例名(); 變數2:=實例名.輸出變數1;	在FB調用的前後列舉輸入引數、輸出引數的代入語句。

在FB調用語句的引數中所使用的符號與可分配表達式如下所示。

類型	內容	使用符號	可分配表達式
EN、VAR_INPUT	輸入變數	:=	所有的表達式
ENO、VAR_OUTPUT、 VAR_OUTPUT_RETAIN	輸出變數	=>	只有變數
VAR_IN_OUT	輸入輸出變數	:=	只有變數

FB的執行結果，透過在實例名後添加“.”(句號)指定輸出變數且代入至變數後儲存。

FB	FB定義	記述示例
1個輸入變數、1個輸出變數的FB的情況下	FB名： FBADD FB實例名： FBADD1 輸入變數1： IN1 輸出變數1： OUT1	FBADD1(IN1:= Input1); Output1 := FBADD1.OUT1;
3個輸入變數、2個輸出變數的FB的情況下	FB名： FBADD FB實例名： FBADD1 輸入變數1： IN1 輸入變數2： IN2 輸入變數3： IN3 輸出變數1： OUT1 輸出變數2： OUT2	FBADD1(IN1:= Input1,IN2:= Input2,IN3:= Input3); Output1 := FBADD1.OUT1; Output2 := FBADD1.OUT2;

■函數調用語句

無返回值的函數及參數中包含VAR_OUTPUT變數的函數，透過將“;”(分號)連接在函數調用表達式後面，可作為語句執行。

書寫格式	內容
函數名(變數1,變數2,...);	用“()”將緊接在函數名後的引數括起來。 多個引數的情況下用“,”(逗號)隔開。

■RETURN語句

語法	書寫格式	內容	記述示例
■RETURN	RETURN;	為了使程式、FB、函數在中途結束而使用。 如果在程式中使用RETURN語句，將跳轉到程式的最後語句的下一步。 如果在FB中使用RETURN語句，將從FB返回。 如果在函數中使用RETURN語句，將從函數返回。 對1個RETURN語句，在系統中使用1點指針型標籤。	IF bool1 THEN RETURN; END_IF;

選擇語句

語法	書寫格式	內容	記述示例

■IF THEN	IF<布爾表達式>THEN
	　<語句…>;
	END_IF;
		布爾表達式(條件表達式)為真(TRUE)時，則執行語句。如果布爾表達式為假(FALSE)，則不執行語句。
		在布爾表達式中，作為在單一的位元型變數的狀態下或包含多個變數的複雜的表達式的布爾運算結果，如果為返回真(TRUE)或假(FALSE)的表達式，可以使用任意表達式。
			IF bool1 THEN
			　intV1 := intV1 + 1;
			END_IF;

■IF...ELSE	IF<布爾表達式>THEN
	　<語句1…>;
	　ELSE
	　<語句2…>;
	END_IF;
		布爾表達式(條件表達式)為真(TRUE)時，則執行語句1。
		布爾表達式的值為假(FALSE)時，則執行語句2。
			IF bool1 THEN
			　intV3 := intV3 + 1;
			　ELSE
			　intV4 := intV4 + 1;
			END_IF;

■IF...ELSEIF	IF<布爾表達式1>THEN
	　<語句1…>;
	　ELSEIF<布爾表達式2>THEN
	　<語句2…>;
	　ELSEIF<布爾表達式3>THEN
	　<語句3…>;
	END_IF;
		布爾表達式(條件表達式)1為真(TRUE)時，則執行語句1。布爾表達式1的值為假(FALSE)而布爾表達式2的值為真(TRUE)時，則執行語句2。
		布爾表達式1、2的值都為假(FALSE)而布爾表達式3的值為真(TRUE)時，則執行語句3。
			IF bool1 THEN
			　intV1 := intV1 + 1;
			　ELSIF bool2 THEN
			　intV2 := intV2 + 2;
			　ELSIF bool3 THEN
			　intV3 := intV3 + 3;
			END_IF;

■CASE	CASE<整數表達式>OF
	　<整數選擇值1>:
	　<語句1…>;
	　<整數選擇值2>:
	　<語句2…>;
	　⋮
	　<整數選擇值n>:
	　<語句n…>;
	　ELSE
	　<語句n+1…>;
	END_CASE;
		執行具有與整數表達式(條件表達式)的值一致的整數的選擇值的語句，在無一致的情況下，則執行ELSE語句的下一語句。
		對於CASE語句，例如可以根據單一的整數值及複雜的表達式的結果的整數值，在執行選擇語句的情況下使用。
			CASE intV1 OF
			　1:
			　bool1 := TRUE;
			　2:
			　bool2 := TRUE;
			　ELSE
			　intV1 := intV1 + 1;
			END_CASE;

重複語句

語法	書寫格式	內容	記述示例
■FOR...DO	FOR<重複變數初始化> 　TO<最終值> 　BY<增加表達式>DO 　<語句…>; END_FOR;	進行作為重複變數使用資料的初始化。 根據增加表達式對初始化後的重複變數進行加法運算或減法運算，重複執行DO與END_FOR之間的1個或其以上的語句直至超出最終值為止。 FOR...DO語句結束後的重複變數，保持在結束時點的值。	FOR intV1 := 0 　TO 30 　BY 1 DO 　intV3 := intV1 + 1; END_FOR;

■WHILE...DO	WHILE<布爾表達式>DO
	　<語句…>;
	END_WHILE;
		布爾表達式(條件表達式)為真(TRUE)時，則執行1個或其以上的語句。
		在執行語句之前判定布爾表達式，布爾表達式為假(FALSE)的情況下，不執行DO...END_WHILE中的語句。對於WHILE語句中的<布爾表達式>，由於只要是結果返回真或假即可，因此IF語句中的<布爾表達式>中的可指定的表達式全部可以使用。
			WHILE intV1 = 30 DO
			　intV1 := intV1 + 1;
			END_WHILE;

■REPEAT...UNTIL	REPEAT
	　<語句…>;
	　UNTIL<布爾表達式>
	END_REPEAT;
		布爾表達式(條件表達式)為假(FALSE)時，則執行1個或其以上的語句。
		布爾表達式在語句執行後判定，值為真(TRUE)時則不執行REPEAT...UNTIL內的語句。REPEAT語句中的<布爾表達式>，由於只要是結果返回真或假即可，因此IF語句中的<布爾表達式>中的可指定的表達式全部可以使用。
			REPEAT
			　intV1 := intV1 + 1;
			　UNTIL intV1 = 30
			END_REPEAT;

■EXIT	EXIT;	在只能於重複語句中使用的語法中，中途結束重複語句。 如果在執行重複環路過程中達到EXIT語句，則不執行EXIT語句之後的重複環路處理。終止重複語句後，從該語句的下一列開始繼續執行程式。	FOR intV1 := 0 　TO 10 　BY 1 DO 　IF intV1 > 10 THEN 　　EXIT; 　END_IF; END_FOR;

注意事項

■使用代入語句時

- 字元串的代入為最大字元串長255字元。代入超過最大字元串長的字元串時，將變為轉換出錯。
- 定時器型、計數器型的觸點與線圈無法在代入語句的左邊使用。
- FB的實例無法在代入語句的左邊使用。應在代入式的左邊使用實例的輸入變數、輸出變數、外部變數。

■使用步繼電器(S)及SFC塊元件(BL)的情況下

將步繼電器(S)及SFC塊元件(BL)在代入式的右邊、或於函數及FB的輸入引數中使用的情況下，有可能變為轉換出錯狀態。該情況下，應替換代入式。

例

替換示例如下所示。

替換前	替換後

M0 := S0;	IF S0 THEN
	M0 := TRUE;
	ELSE
	M0 := FALSE;
	END_IF;

此外，將步繼電器(S)及帶塊指定步繼電器(BL□\S□)的位指定在程式中使用的情況下，應指定正確的資料容量。由於步繼電器(S)及帶塊指定步繼電器(BL□\S□)不是資料類型自動轉換的對象，因此資料容量不一致的情況下，有可能變為轉換出錯狀態。

例

替換示例如下所示。

替換前	替換後
(*K4S0為16位元、D0:UD為32位元，因此轉換出錯*) D0:UD := K4S0; (*BL1\K4S10為16位元、DMOV的第2引數為32位元，因此轉換出錯*) DMOV(TRUE,BL1\K4S10,D100);	(*代入至16位元資料*) D0 := K4S0; (*DMOV中指定32位元份的資料*) DMOV(TRUE, BL1\K8S10, D100:UD);

■代入算術運算式結果的情況下

將算術運算表達式的結果代入到資料容量較大的資料類型的變數中的情況下，應預先把算術運算表達式的變數轉換為左邊的資料類型之後再進行運算。

例

在把資料容量16位元(INT型)的算術運算結果代入到32位元的資料類型(DINT型)的情況下

```
varDint1 := varInt1 * 10; //varInt1為INT型，varDint1為DINT型
```

算術運算表達式的運算結果，其資料類型將與輸入操作數的資料類型相同。因此在上述的程式中，varInt1*10的運算結果超出了INT型的範圍(-32768～32767)的情況下，進行了上溢或下溢的運算結果被代入到varDint1中。

在這種情況下，應預先將運算表達式的操作數轉換到左邊的資料類型之後再進行運算。

```
varDint2 := INT_TO_DINT(varInt1); //將INT型變數轉換為DINT型變數
varDint1 := varDint2 * 10; //以DINT型進行乘法運算，代入運算結果
```

■在算術運算式中使用符號反轉運算符的情況下

對資料類型的最小值，使用符號反轉運算符(-)時，將變為相同的值。

例如INT型最小值的情況下，變為-(-32768)=-32768。

因此，在資料類型的自動轉換的對象變數中使用符號反轉運算符時，可能不會變為希望的結果。

例

varInt1(INT型)的值為-32768、varDint1(DINT型)的值為0的情況下

```
varDint2 := -varInt1 + varDint1;
```

該情況下，(-varInt1)的值將保持為-32768不變，在varDint2中代入-32768。

在算術運算式中使用符號反轉運算符的情況下，應預先在算術運算前進行資料類型的自動轉換或創建不使用符號反轉運算符的程式。

例

算術運算之前進行資料類型自動轉換的情況下

```
varDint3 := varInt;
varDint2 := -varDint3 + varDint1;
```

例

不使用符號反轉運算符的情況下

```
varDint2 := varDint1 - varInt1;
```

■資料類型從單精度實數轉換至雙精度實數的情況下

從單精度實數至雙精度實數的類型轉換(REAL_TO_LREAL)中，轉換結果可能會發生誤差。

因此，進行資料類型自動轉換的情況下，或返回值在代入語句的右邊或於算術運算式的操作數中使用了實數型函數(SIN等)的情況下，可能無法變為希望的結果。

例

發生誤差的情況下

```
varReal1 := -1234.567;
varLReal1 := ABS(varReal1);
```

上述情況下，ABS(varReal1)的返回值為單精度實數，由於將該值型轉換為雙精度實數並代入varLReal1中，因此將發生誤差。

這種情況下，應透過與代入目標相同的資料類型(雙精度實數)創建執行函數的程式。

例

不發生誤差的情況下

```
varLReal2 := -1234.567;
varLReal1 := ABS(varLReal2);
```

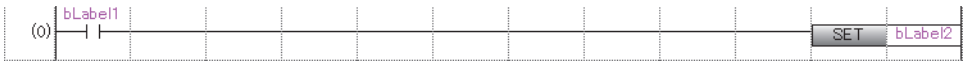
■使用位元型標籤時

在選擇語句或重複語句中一旦布爾表達式(條件表達式)成立，在<語句>內的位元型標籤置為ON狀態時，該位元型標籤將變為常時ON。

例

常時ON的程式

ST程式	與ST程式同等處理的梯形圖程式

IF bLabel1 THEN
  bLabel2 := TRUE;
END_IF;
	

為避免常時ON，應按下述方式添加將位元型標籤置為OFF的程式。

例

避免常時ON的程式

ST程式*1	與ST程式同等處理的梯形圖程式

IF bLabel1 THEN
  bLabel2 := TRUE;
  ELSE
  bLabel2 := FALSE;
END_IF;
	

*1　上述程式可以按下述方式記述。
bLabel2:=bLabel1;
或
OUT(bLabel1,bLabel2);
但是，在<語句>內使用了OUT指令的情況下，變為與常時ON的程式同樣的狀態。

■使用定時器FB、計數器FB時

選擇語句中的布爾表達式(條件表達式)，與定時器FB、計數器FB的執行條件不同。

例

定時器FB的情況下

更改前程式示例

```
IF bLabel1 THEN
    TIMER_100_FB_M_1(Coil:= bLabel2, Preset:= wLabel3, ValueIn:= wLabel4, ValueOut=> wLabel5, Status=> bLabel6);
END_IF;
(* bLabel1=ON且bLabel2=ON時，開始計數。*)
(* bLabel1=ON且bLabel2=OFF時，清除計數。*)
(* bLabel1=OFF且bLabel2=ON時，停止計數。不清除計數值。*)
(* bLabel1=OFF且bLabel2=OFF時，停止計數。不清除計數值。*)
```

更改後的程式示例

```
TIMER_100_FB_M_1(Coil:= (bLabel1 & bLabel2), Preset:= wLabel3, ValueIn:= wLabel4, ValueOut=> wLabel5 ,Status=> bLabel6);
```

例

計數器FB的情況下

更改前程式示例

```
IF bLabel1 THEN
    COUNTER_FB_M_1(Coil:= bLabel2, Preset:= wLabel3, ValueIn:= wLabel4, ValueOut=> wLabel5, Status=> bLabel6 );
END_IF;
(* bLabel1=ON且bLabel2=ON/OFF時，將計數+1。*)
(* bLabel1=OFF且bLabel2=ON/OFF時，不進行計數。*)
(* bLabel1=ON/OFF與計數+1不聯動。*)
```

更改後的程式示例

```
COUNTER_FB_M_1(Coil:= (bLabel1 & bLabel2), Preset:= wLabel3, ValueIn:= wLabel4, ValueOut=> wLabel5, Status=> bLabel6);
```

上述更改前的程式示例是在選擇語句不成立的情況下，為了不執行與定時器、計數器相關的語句而創建的。
根據bLabel1條件與bLabel1的AND條件使定時器、計數器動作的情況下，應不使用控制語句，而是僅使用FB。
透過使用更改後的程式，可以使定時器、計數器動作。

■使用FOR...DO語句時

- 無法在重複變數中使用結構體構件及陣列要素。
- 應使在重複變數中使用的類型與<最終值的表達式>、<增加表達式>的類型一致。
- <增加表達式>可以省略。省略的情況下<增加表達式>作為1執行。
- 如果向<增加表達式>中代入0，則FOR語法以下可能不被執行或變為無限循環。
- 在FOR...DO構文中執行FOR構文中的<語句…>後進行重複變數的計數處理。大於重複變數的資料類型的最大值或小於最小值的計數處理被執行的情況下，將發生無限循環。

■使用上升沿指令、下降沿指令時

- 在IF語句以及CASE語句中使用上升沿指令、下降沿指令時的動作如下所示。

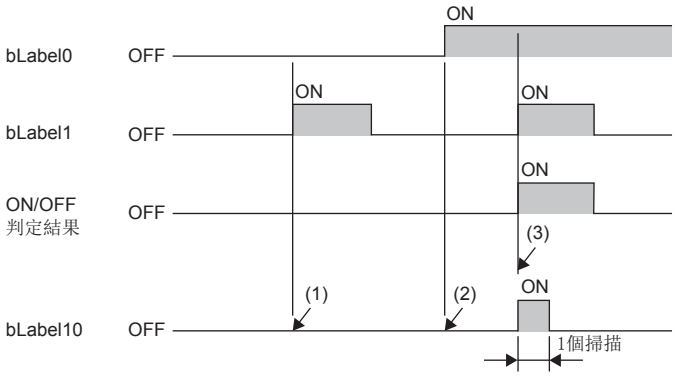
條件			動作結果		
IF語句、CASE語句的條件表達式	指令執行條件(EN)	上一次掃描時的指令ON/OFF判定結果	指令ON/OFF判定結果	上升沿指令	下降沿指令
TRUE或CASE一致	TRUE	ON	ON	不執行	不執行
		OFF	ON	執行	不執行
	FALSE	ON	OFF	不執行	執行
		OFF	OFF	不執行	不執行
FALSE或CASE不一致	TRUE	ON	OFF	不執行	不執行*1
		OFF	OFF	不執行	不執行
	FALSE	ON	OFF	不執行	不執行*1
		OFF	OFF	不執行	不執行

*1　雖然為下降沿(ON→OFF)，但是由於IF語句或CASE語句的條件不成立，因此不執行指令。

例

在IF語句中使用了PLS指令(上升沿執行)的情況下

```
IF bLabel0 THEN
    PLS(bLabel1, bLabel10);
END_IF;
```



(1) bLabel0 = OFF的情況下(IF語句的條件表達式為FALSE)，ON/OFF判定結果為OFF，不執行PLS指令。(保持bLabel10 = OFF不變)

(2) bLabel0 = ON(IF語句的條件表達式為TRUE)，且bLabel1 = OFF(指令執行條件為OFF)的情況下，ON/OFF判定結果為OFF，不執行PLS指令。(保持bLabel10 = OFF不變)

(3) bLabel0 = ON(IF語句的條件表達式為TRUE)，且bLabel1 = ON(指令執行條件為ON)的情況下，ON/OFF判定結果為OFF→ON(上升沿條件成立)，執行PLS指令。(bLabel10僅1掃描ON)

• 在重複語句(FOR語句、WHILE語句或REPEAT語句)執行上升沿指令或下降沿指令的情況下，使用變址繼電器(V)以及變址修飾。該情況下，在系統中對每個正在使用變址繼電器(V)的指令使用1點變址繼電器(V)。因此，應在重複語句中加入使用的點數，確保正在使用的指令數的變址繼電器(V)。

例

在FOR語句中使用上升沿指令及下降沿指令的情況下

```
在1個地方使用變址繼電器(V)的示例
(可最多使用變址繼電器(V)合計11點(INC指令中V0～V10)。)
FOR Z0 := 0 TO 9 BY 1 DO
    INC(EGP(M100Z0 , V0Z0) , D100Z0);
END_FOR;
在2個地方使用變址繼電器(V)的示例
(可最多使用變址繼電器(V)合計22點(INC指令中V0～V10、DEC指令中V11～V21)。)
FOR Z0 := 0 TO 9 BY 1 DO
    INC(EGP(M100Z0 , V0Z0) , D100Z0);
    DEC(EGF(M200Z0 , V11Z0) , D200Z0);
END_FOR;
```

■使用主控制指令時

主控制OFF時的動作如下所示。

• 選擇語句(IF語句或CASE語句)中或重複語句(FOR語句、WHILE語句或REPEAT語句)中的語句變為無處理。

• 選擇語句或重複語句之外，代入語句的情況下變為無處理，代入語句以外的語句變為非執行處理。

例

選擇語句(IF語句)中的語句

```
MC(M0 , N1 , M1);  //主控制OFF
IF M2 THEN
    M3 := M4;  //主控制OFF時為無處理，因此M3保持之前掃描時的值
END_IF;
M20 := MCR(M0, N1);
```

例

選擇語句或重複語句之外的語句(位元代入語句的情況下)

```
MC(M0 , N1 , M1);  //主控制OFF
M3 := M4;  //主控制OFF時為無處理，因此M3保持之前掃描時的值
M20 := MCR(M0, N1);
```

例

選擇語句或重複語句之外的語句(OUT指令的情況下)

```
MC(M0 , N1 , M1);  //主控制OFF
OUT(M2, M3);  //主控制OFF時為非執行處理，因此將M3置為OFF
M20 := MCR(M0, N1);
```

常數

常數的記載方法

在ST程式中除了普通的常數記載以外，也可以進行下述的常數記載。

可對應的資料類型	類型	記載方法*1	記載示例
位元	引導	在使用的引導值前附上“BOOL#”。	BOOL#1、BOOL#0 BOOL#TRUE、BOOL#FALSE
字[無符號]/位元串[16位元]	2進制數	在2進制數前附上“UINT#2#”或“WORD#2#”。*2	UINT#2#10101010 WORD#2#10101010
	8進制數	在8進制數前附上“UINT#8#”或“WORD#8#”。*2	UINT#8#3370 WORD#8#3370
	10進制數	在10進制數前附上“UINT#”或“WORD#”。*2	UINT#123 WORD#123
	16進制數	在16進制數前附上“UINT#16#”或“WORD#16#”。*2	UINT#16#FF WORD#16#FF
雙字[無符號]/位元串[32位元]	2進制數	在2進制數前附上“UDINT#2#”或“DWORD#2#”。*2	UDINT#2#1100110011001100 DWORD#2#1100110011001100
	8進制數	在8進制數前附上“UDINT#8#”或“DWORD#8#”。*2	UDINT#8#33703370 DWORD#8#33703370
	10進制數	在10進制數前附上“UDINT#”或“DWORD#”。*2	UDINT#456789 DWORD#456789
	16進制數	在16進制數前附上“UDINT#16#”或“DWORD#16#”。*2	UDINT#16#FFFF DWORD#16#FFFF
字[帶符號]	2進制數	在2進制數前附上“INT#2#”。	INT#2#01010101
	8進制數	在8進制數前附上“INT#8#”。	INT#8#3370
	10進制數	在10進制數前附上“INT#”。	INT#-123
	16進制數	在16進制數前附上“INT#16#”。	INT#16#1F
雙字[帶符號]	2進制數	在2進制數前附上“DINT#2#”。	DINT#2#00110011001100110011
	8進制數	在8進制數前附上“DINT#8#”。	DINT#8#33703370
	10進制數	在10進制數前附上“DINT#”。	DINT#-456789
	16進制數	在16進制數前附上“DINT#16#”。	DINT#16#1F1F
單精度實數	實數	在實數前附上“REAL#”。	REAL#2.34
	實數(指數表現)		REAL#1.0E6
雙精度實數	實數	在實數前附上“LREAL#”。	LREAL#-2.34
	實數(指數表現)		LREAL#1.001E16
字元串	STRING	將字元串(ASCII、移位JIS)用單引號(')括住。	'ABC'
字元串[Unicode]	WSTRING	將Unicode字元串用雙引號(")括住。	"ABC"

*1 不區分大小寫。另外，無法與使用了K、H、E的常數記載同時使用。

*2 為ANY_NUM的算數運算子的操作數、函數調用語句、FB調用語句、函數調用表達式的引數的情況下，應透過“UINT#”或“UDINT#”作為常數記載。“WORD#”或“DWORD#”的常數記載的情況下，因被解釋為位元串，在轉換時將變為出錯。

上述以外常數的記載方法，請參閱下述手冊。

MELSEC iQ-R CPU模組用戶手冊(應用篇)

要點

在2進制數、8進制數、10進制數、16進制數、實數的記載中，使用下劃線(_)隔開數值可使程式更易懂。例如雙字[無符號]的2進制數記載的情況下，可記載為如下。

UDINT#2#1100_1100_1100_1100

在程式的處理上，將會忽略用下劃線(_)的數值隔開處。

標籤與元件

指定方法

在ST程式中可以直接記述並使用標籤與元件。標籤與元件可以在表達式的左邊、右邊、通用函數/FB的引數、返回值等中使用。

標籤及元件的詳細內容，請參閱下述手冊。

MELSEC iQ-R CPU模組用戶手冊(應用篇)

■帶類型指定元件記載

透過在元件名中添加元件型指定符，可將字元件作為任意資料類型使用。

元件型指定符	資料類型	示例	示例的說明
無	總稱資料類型的ANY16 在算術運算式等中只使用了元件的情況下，變為字[帶符號]。 但是在FUN/FB的引數部分中作為無類型指定的元件被指定的情況下，則變為引數定義的資料類型。	D0	D0中不附加類型指定符的情況下
:U	字[無符號]/位元串[16位元]	D0:U	將D0作為字[無符號]/位元串[16位元]的值
:D	雙字[帶符號]	D0:D	將D0、D1作為雙字[帶符號]的值
:UD	雙字[無符號]/位元串[32位元]	D0:UD	將D0、D1作為雙字[無符號]/位元串[32位元]的值
:E	單精度實數	D0:E	將D0、D1作為單精度實數的值
:ED	雙精度實數	D0:ED	將D0、D1、D2、D3作為雙精度實數的值

元件類型指定符可使用的元件如下所示。

- 資料寄存器(D)
- 鏈接寄存器(W)
- 鏈接直接元件(J□\W□)
- 模組訪問元件(U□\G□)
- CPU緩衝記憶體訪問元件(U3E□\G□/U3E□\HG□)
- 檔案寄存器(R/ZR)
- 更新資料寄存器(RD)

無法對進行了位指定或間接指定的元件賦予元件型指定符。

■元件的指定方法

關於元件的指定可以使用下述方法。

- 變址修飾
- 位元指定
- 位指定
- 間接指定

關於詳細內容，請參閱以下手冊。

MELSEC iQ-R CPU模組用戶手冊(應用篇)
MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)

注意事項

- 在ST程式中無法使用指針型。
- 用當前值使用定時器、計數器、累計定時器的元件時的資料類型為字[無符號]/位元串[16位元]。用當前值使用長定時器、長計數器、長累計定時器的元件時的資料類型為雙字[無符號]/位元串[32位元]。
- 使用位指定代入的情況下，應使右邊和左邊的資料類型一致。

例

```
D0:=K5X0;
```

在上述情況下，因為K5X0為雙字型、D0為字型，因此程式出錯。

- 使用位指定代入的情況下，右邊>左邊時，在左邊的對象點數範圍內進行資料傳送。

例

```
K5X0:＝2#1011_1101_1111_0111_0011_0001;
```

在上述情況下，由於K5X0的對象點數20點，因此向K5X0代入1101_1111_0111_0011_0001(20位)。

- 以字[無符號]/位元串[16位元]以外的類型使用計數器(C)、定時器(T)、累計定時器(ST)的當前值(TNn等)時，或以雙字[無符號]/位元串[32位元]以外的類型使用長計數器(LC)、長定時器(LT)、長累計定時器(LST)的當前值(LTNn等)的情況下，應使用類型轉換函數。

例

```
varInt:＝WORD_TO_INT(T0);(*使用類型轉換函數*)
```

- 定時器及計數器的元件的線圈(TC、STC、LTC、LSTC、CC、LCC)用於代入式的右邊及函數、FB的輸入引數的情況下，將作為觸點(TS、STS、LTS、LSTS、CS、LCS)進行動作。
- 希望將定時器及計數器的線圈用於輸入引數的情況下，應使用定時器型及計數器型的標籤。

例

定時器元件及定時器型標籤的情況下

```
M1 := TC0; (* 將觸點(TS0)的值代入到M1中。 *)
M2 := INV(TC1); (* 將觸點(TS1)的反轉結果代入到M2中。 *)
M1 := tLabel0.C; (* 將定時器型標籤tLabel0的線圈的值代入到M1中。 *)
M2 := INV(tLabel1.C); (* 將定時器型標籤tLabel1的線圈的反轉結果代入到M2中。 *)
```

■使用元件時的數據類型自動轉換

以字[帶符號]以外的資料類型使用字元件的情況下，應賦予元件類型指定符。(☞ 60頁 帶類型指定元件記載)

例

將D2、D3的值傳送至雙字[無符號]的標籤dwordLabel1的情況下

```
//賦予元件類型指定符，以正確的資料類型傳送的示例
dwordLabel1:= D2:UD;
//賦予的元件類型指定符的D2:UD為雙字[無符號]，因此D2、D3的值被傳送至dwordLabel1。

//出現預料外的傳送結果的示例
dwordLabel1:= D2;
//沒有元件類型指定符的D2為字[帶符號]，所以將資料類型自動轉換為雙字[無符號]後，傳送至dwordLabel1。
//因此，僅傳送D2的值，不傳送D3的值。
```

注釋

可以在ST程式中使用的注釋如下所示。

注釋形式	注釋符號	內容	記述示例
單一列注釋	//	將從開始符號“//”到列尾的內容作為注釋處理。	//注釋內容
多列注釋	(**)	將從開始符號“(*”到結束符號“*)”的內容作為注釋處理。 可以在注釋中換列輸入。	■無換列 (*注釋內容*) ■有換列 (*第1列注釋內容 第2列注釋內容*)
	/**/	將從開始符號“/*”到結束符號“*/”的內容作為注釋處理。 可以在注釋中換列輸入。	■無換列 /*注釋內容*/ ■有換列 /*第1列注釋內容 第2列注釋內容*/

在多列注釋中請勿記述含有結束符號的注釋。

7 FBD/LD語言

RnCPU RnENCPU RnPCPU(過程) RnPCPU(二重化) RnPSFCPU(常規) RnPSFCPU(安全) RnSFCPU(常規) RnSFCPU(安全)

是透過按照資料及信號的流向對進行特定處理的塊、變數部件、常數部件進行連接，對程式進行記述的圖表語言。

要點

- 在本章中，對FBD/LD語言的動作及規格相關內容進行說明。關於創建FBD/LD程式時的操作方法，請參閱下述手冊。

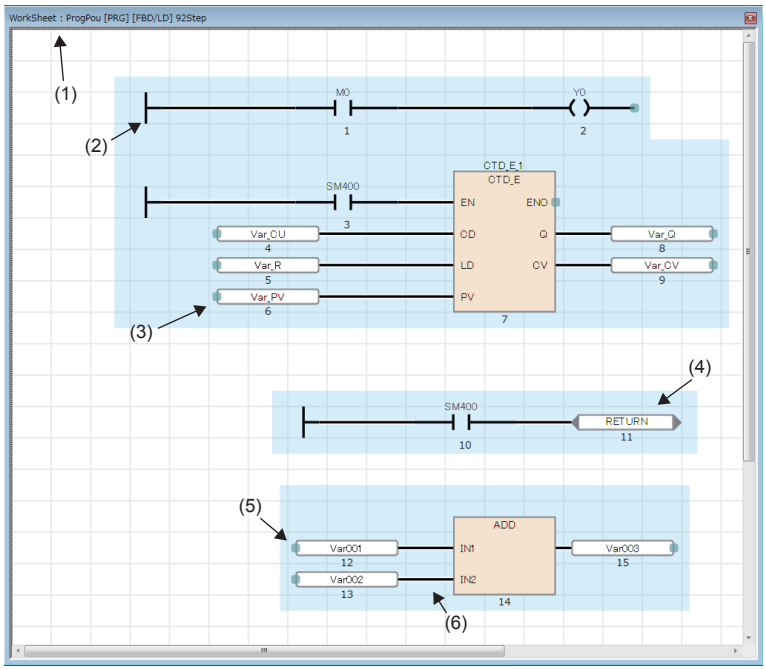
GX Works3 操作手冊

- 在過程CPU的過程控制FB中，透過FB部件的配置及連接將可以簡單地創建程式，由於具備了執行過程控制所需的種類豐富的FB，使得程式設計變得簡單。在過程CPU中使用過程控制FB時，請參閱下述手冊。

MELSEC iQ-R 程式手冊(過程控制FB/指令篇)

7.1 配置

FBD/LD語言中，可創建下述所示程式。



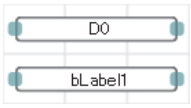
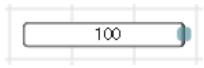
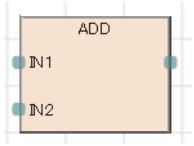
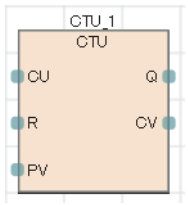
(1) 工作表
(2) LD部件
(3) FBD部件
(4) 通用部件
(5) 連接點
(6) 連接線

在FBD/LD語言的程式中，資料從功能塊(FB)、函數(FUN)、變數部件(標籤與元件)、常數部件的輸出點流向其他FB及變數部件等的輸入點。

部件

FBD部件

配置FBD/LD程式的FBD部件如下所示。

項目		內容
變數		在儲存各個值(資料)時使用變數。變數中規定資料類型，僅儲存該資料類型的值(資料)。 在變數中，可以指定標籤或元件。
常數		輸出所指定的常數值。
函數(FUN)		執行函數。 • 函數的創建方法(GX Works3 操作手冊) • 通用函數(MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇))
功能塊(FB)		執行FB。 • FB的創建方法(GX Works3 操作手冊) • 通用FB(MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)) • 模組FB(所使用模組的FB參考)

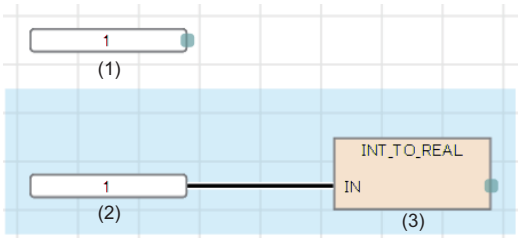
■常數部件的資料類型

常數部件的情況下，常數值的資料類型在輸入常數值時尚未決定。資料類型在以連接線將常數部件與FBD部件連接時決定。常數值的資料類型將變為與以連接線連接的目標的FBD部件相同的資料類型。

例

在常數值中輸入1的情況下

由於資料類型後補中存在BOOL型、WORD型、DWORD型、INT型、DINT型、REAL型、LREAL型，因此無法確定資料類型。以連接線將常數部件與FBD部件連接時，將變為連接目標中的部件輸入點的資料類型。



(1) 資料類型未決定
(2) INT型
(3) INT型

■資料類型的自動轉換

被連接的部件的資料類型不同時，可能會有自動轉換資料類型的情況。

類型轉換時，為了確保不丟失資料，僅從容量小的資料類型轉換至容量大的資料類型。FBD/LD語言中資料類型的自動轉換的動作與ST語言相同。詳細內容，請參閱下述章節。

☞51頁 資料類型的自動轉換

■函數的輸入輸出點

- 函數需要預先將全部輸入點與其他部件連接。
- 函數的輸入變數與輸出變數在確定資料類型後，連接至輸入點與輸出點的部件也需要配合該資料類型。

■將帶EN的函數、帶EN的FB的輸出變數與其他部件連接時的注意事項

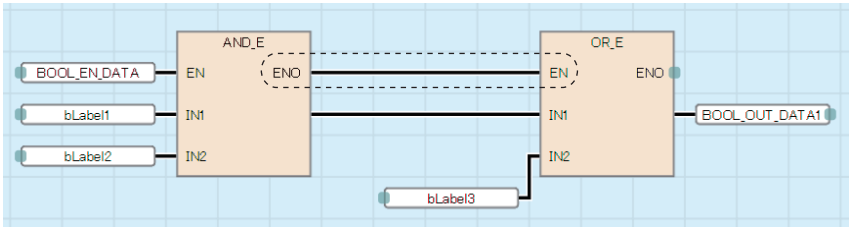
在帶EN的函數及帶EN的FB的ENO為FALSE(運算停止)時，根據連接的部件，程式的動作將有所不同。

連接的部件	在帶EN的函數及帶EN的FB的ENO為FALSE(運算停止)時的程式的動作
• 函數 • FB	與帶EN的函數及帶EN的FB的輸出變數連接的輸入變數值變為不定值。
上述以外	與帶EN的函數及帶EN的FB的輸出變數連接的部件值不會被更改。(變為上次的值。)

將帶EN的FB的輸出變數與其他函數或FB的輸入變數直接連接時，可能會發生預料外的動作。
為了防止使用不定值，應以下述示例的任一方法連接。

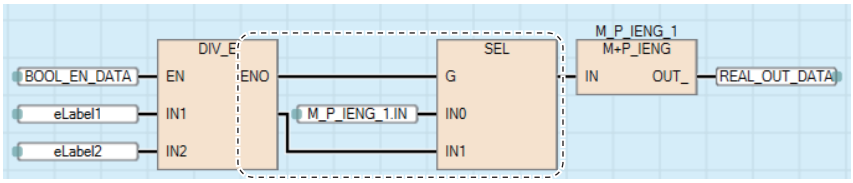
例

將帶EN的函數或帶EN的FB作為連接目標，連接源的ENO也與連接目標的EN連接。



例

使用選擇函數SEL，使ENO為FALSE時能輸入上次的值。



■使用步繼電器(S)及SFC塊元件(BL)的情況下

將步繼電器(S)及SFC塊元件(BL)在變數部件中使用的情況下，有可能變為轉換出錯狀態。該情況下，應將變數部件替換為觸點部件。

例

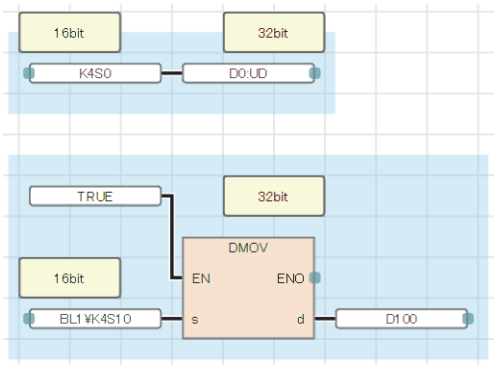
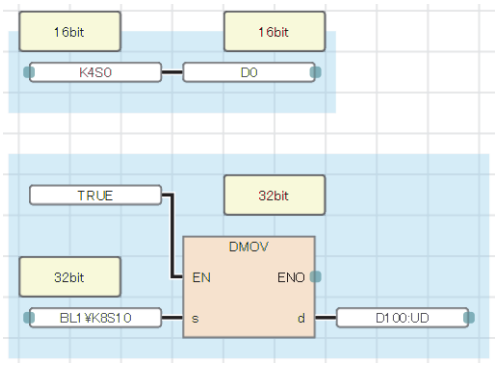
替換示例如下所示。

替換前	替換後
	

此外，在程式中使用步繼電器(S)及帶塊指定的步繼電器(BL□\S□)的位指定的情況下，應指定正確的資料容量。步繼電器(S)及帶塊指定的步繼電器(BL□\S□)不是資料類型自動轉換的對象，因此資料容量不一致的情況下，有可能變為轉換出錯狀態。

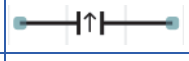
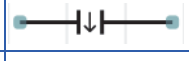
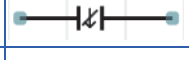
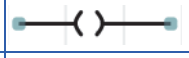
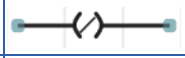
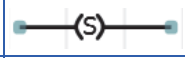
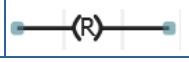
例

替換示例如下所示。

替換前	替換後
	

LD部件

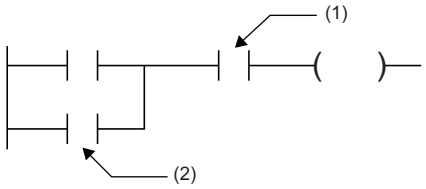
在FBD/LD程式中可使用的梯形圖部件如下所示。

項目		內容
左母線		是表示母線的部件。為創建梯形圖時的起點。 左母線的輸出為常時ON。
常開觸點		指定元件或標籤變為ON時導通。
常閉觸點		指定元件或標籤變為OFF時導通。
上升沿脈衝		指定元件或標籤上升沿時(OFF→ON)導通。
下降沿脈衝		指定元件或標籤下降沿時(ON→OFF)導通。
上升沿脈衝否定		指定元件或標籤OFF時、ON時以及下降沿時(ON→OFF)導通。
下降沿脈衝否定		指定元件或標籤OFF時、ON時以及上升沿時(OFF→ON)導通。
線圈		將運算結果輸出至指定元件或標籤。
反轉型線圈		運算結果變為OFF時，指定元件或標籤將變為ON。
設置線圈		運算結果變為ON時，指定元件或標籤將變為ON。 即使運算結果變為OFF，已ON的元件或標籤也將保持ON狀態不變。
復位線圈		運算結果變為ON時，指定元件或標籤將變為OFF。 運算結果變為OFF的情況下，元件或標籤的狀態將不變化。

■觸點符號的AND運算與OR運算

觸點符號根據梯形圖的連接狀態，進行AND運算、OR運算，並反映至運算結果。

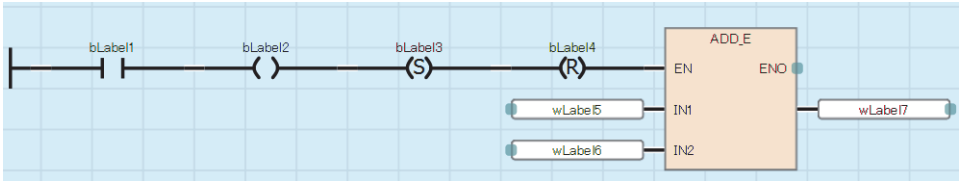
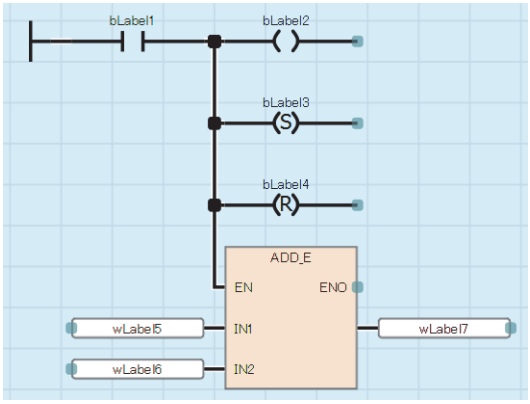
- 串聯連接(1)時，進行之前的運算結果與AND運算，並作為運算結果。
- 並聯連接(2)時，進行之前的運算結果與OR運算，並作為運算結果。



(1) 串聯連接的觸點
(2) 並聯連接的觸點

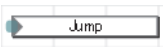
■將其他部件連接到線圈的輸出連接點上的情況下

將其他部件串聯連接到線圈的輸出連接點上時，動作將變為與並聯連接時相同。

項目	內容
程式示例	
上述示例的動作	

通用部件

配置在FBD/LD程式上的通用部件如下所示。

項目		內容
跳轉*1		從跳轉部件開始到跳轉標籤為止跳轉執行處理。不執行已跳轉的部分。 根據至跳轉部件的ON/OFF資訊，控制是否進行跳轉。 ON：在跳轉標籤中跳轉執行處理。 OFF：不跳轉，進行普通的執行處理。
跳轉標籤*1		變為來自於同一程式內的跳轉部件的跳轉目標。進行了跳轉的情況下，將從跳轉標籤及其以後的執行順序的程式執行處理。
連接器		作為連接線的替代品使用。 處理將移轉至成對的連接器部件。 對一個輸出連接器，可以使用一個或多個輸入連接器。
返回*1		程式上的返回部件及其以後的處理將中斷。在不執行返回部件及其以後的程式以及函數、FB的處理的情況下使用。 根據至返回部件的ON/OFF資訊，控制是否進行返回處理。 ON：執行返回處理。 OFF：不進行返回處理，進行普通的執行處理。
注釋		記載注釋的情況下使用。

*1　在SFC程式的Zoom編輯器內不能使用。

■關於跳轉部件

- 在跳轉部件中使線圈處於ON狀態的定時器跳轉時，將導致無法進行正常的測量。
- 在跳轉部件的上側(執行順序在前)可配置跳轉標籤。該情況下，應創建包含脫離循環的方法的程式，以確保不超出看門狗定時器的設置值。
- 在跳轉部件與跳轉標籤中僅可指定指針型的局部標籤。另外，無法使用結構體的構件。
- 不可以使用指針分支指令(CJ、SCJ、JMP)。進行跳轉的情況下，應使用跳轉部件。
- 不可以進行至程式塊外側的跳轉及自外側的跳轉。

跳轉關聯的動作	執行可否
至程式塊外側的跳轉*1	不可以執行
自程式塊外側的跳轉*1	不可以執行
子程式的調用	可以執行
作為子程式被調用	不可以執行

*1　包含透過BREAK指令進行的分支。

■關於返回部件

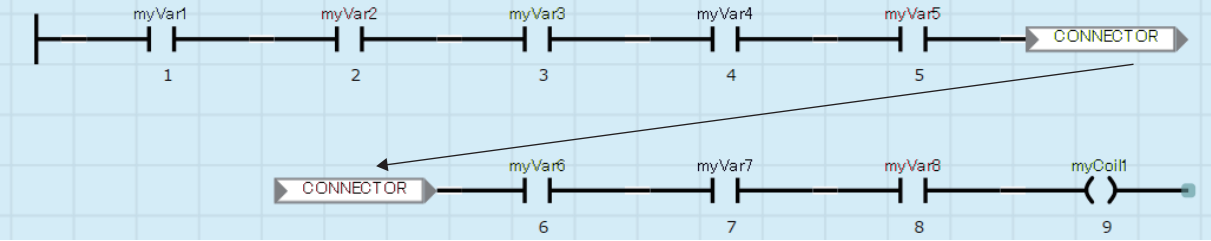
• 返回部件的動作根據使用的程式及函數、FB而有所不同。

所使用的程式部件	內容
程式	結束程式部件的執行。
函數	結束函數，返回至調用了函數的指令的下一步。
FB	結束FB，返回至調用了FB的指令的下一步。

• 宏型FB中使用返回部件的情況下，請勿對相同FB實例名的FB部件進行多個配置。

■關於連接器部件

連接器部件在FBD/LD編輯器的顯示範圍內或印刷範圍內，在希望配置程式的情況下使用。



連接線

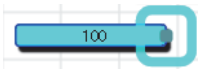
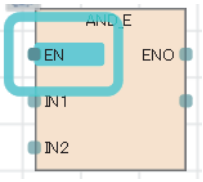
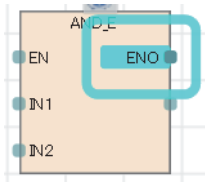
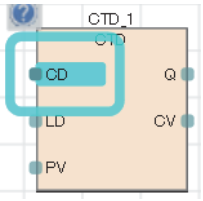
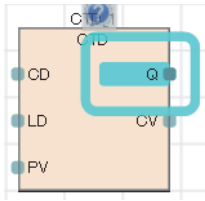
是連接FBD部件、LD部件、通用部件的連接點間的線。

連接部件並從左端向右端交接資料。需與連接的部件的資料類型需為一，或資料類型需為可進行自動轉換的類型。

連接點

是以連接線連接FBD部件、LD部件、通用部件時的端點。

各部件左側點表示輸入側、右側點表示輸出側。

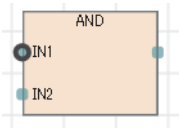
項目	輸入連接點	輸出連接點
觸點		
線圈		
變數		
常數	—	
函數		 不顯示函數的返回值名稱。
FB		

連接點被連接時將變為隱藏。

■輸入輸出點的反轉

透過連接點可對至部件的輸入或自部件的輸出進行反轉。

用黑圓圈括住反轉狀態的連接點，反轉輸入或輸出的資料(FALSE→TRUE或TRUE→FALSE)。

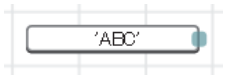
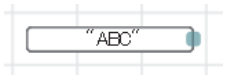


可反轉資料類型為BOOL、WORD、DWORD、ANY_BIT、ANY_BOOL。

常數

常數的記載方法

FBD/LD程式中的字元串的記載方法如下所示。

資料類型		記載方法	記載示例
字元串	STRING	將字元串(ASCII、移位JIS)用單引號(')括住。	
字元串[Unicode]	WSTRING	用雙引號(")括住Unicode字元串。	

上述以外常數的記載方法，請參閱下述手冊。

MELSEC iQ-R CPU模組用戶手冊(應用篇)

標籤與元件

指定方法

在FBD/LD程式上，可以直接記述標籤與元件進行使用。對部件的輸入點、輸出點、通用函數/FB的引數、返回值等可以使用標籤與元件。

標籤及元件的詳細內容，請參閱下述手冊。

MELSEC iQ-R CPU模組用戶手冊(應用篇)

■帶類型指定的元件記載

透過對元件名附加元件型指定符，可以以任意資料類型使用字元件。

FBD/LD語言中的元件型指定符與可使用的元件與ST語言相同。詳細內容，請參閱下述章節。

☞ 60頁 帶類型指定元件記載

不指定字元件的資料類型時，資料類型將根據元件的類型來決定。

字元件	資料類型
定時器元件的當前值(TN)、累計定時器元件的當前值(STN)、計數器元件的當前值(CN)	WORD
長定時器元件的當前值(LTN)、長累計定時器元件的當前值(LSTN)、長計數器元件的當前值(LCN)	DWORD
長變址寄存器(LZ)	DINT
上述以外	ANY16

注意事項

■使用標籤的情況下

- 在陣列的下標中，不能使用局部元件。但是，將下標中使用的元件代入其他元件中，並將代入目標的元件指定為下標時將變為可以使用。

■使用元件時的資料類型自動轉換

以字[帶符號]以外的資料類型使用字元件的情況下，應賦予元件類型指定符。(☞72頁 帶類型指定的元件記載)

例

將D2、D3的值傳送至雙字[無符號]的標籤dwordLabel1的情況下

賦予元件類型指定符，以正確的資料類型傳送的示例	出現預料外的傳送結果的示例
	
賦予元件類型指定符的D2:UD為雙字[無符號]，所以D2、D3的值被傳送至dwordLabel1。	沒有元件類型指定符的D2為字[帶符號]，所以將資料類型自動轉換為雙字[無符號]後，傳送至dwordLabel1。 因此，僅傳送D2的值，不傳送D3的值。

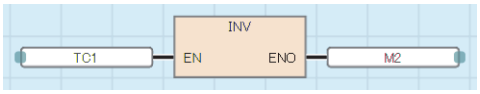
■使用定時器、計數器的情況下

- 定時器及計數器的元件的線圈(TC、STC、LTC、LSTC、CC、LCC)作為變數及函數、FB的輸入使用的情況下，將作為觸點(TS、STS、LTS、LSTS、CS、LCS)進行動作。
- 希望將定時器及計數器的線圈作為輸入使用的情況下，應使用定時器型及計數器型的標籤。

例

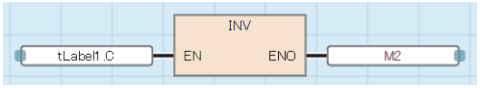
定時器元件及定時器型標籤的情況下

將觸點(TS0)的值代入到M1中。



將觸點(TS1)的反轉結果代入到M2中。

將定時器型標籤tLabel0的線圈值代入到M1中。

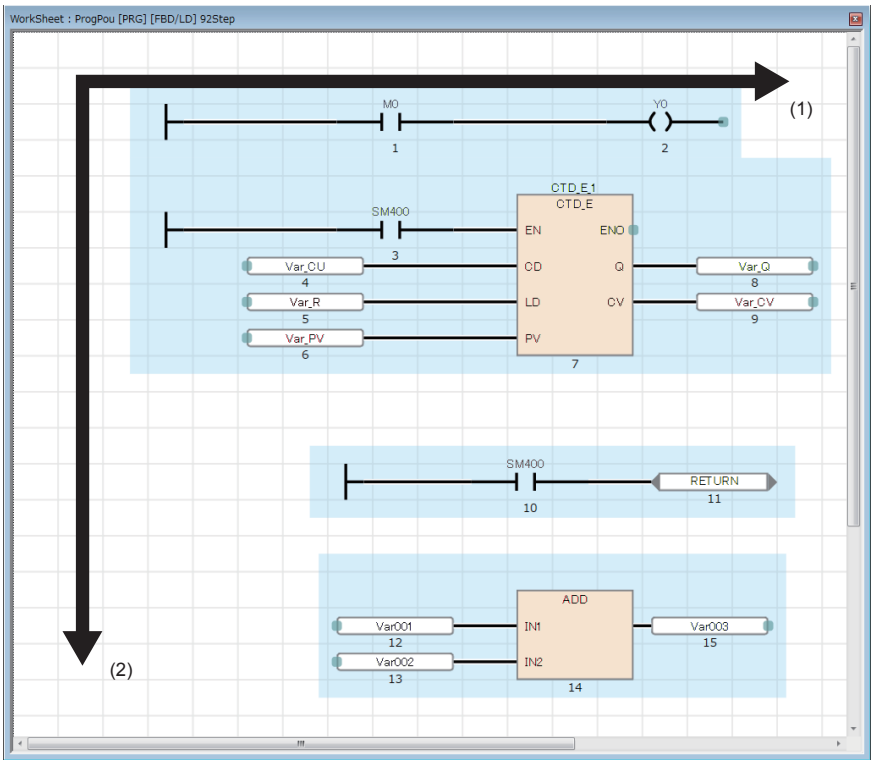


將定時器型標籤tLabel1的線圈反轉結果代入到M2中。

7.2 程式執行順序

部件的執行順序

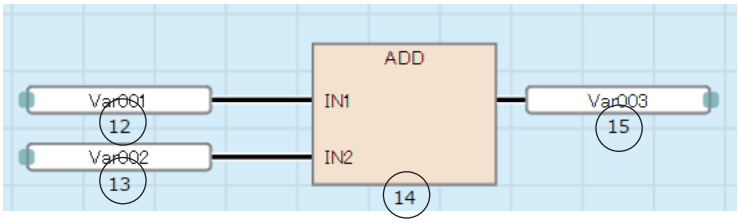
FBD/LD編輯器上部件的執行順序將根據部件位置關係與連接狀況而定。



(1) 從左至右執行。

(2) 從上至下執行。

配置在FBD/LD編輯器上的各部件，將顯示執行順序的編號。如果轉換程式，則顯示確定的執行順序。

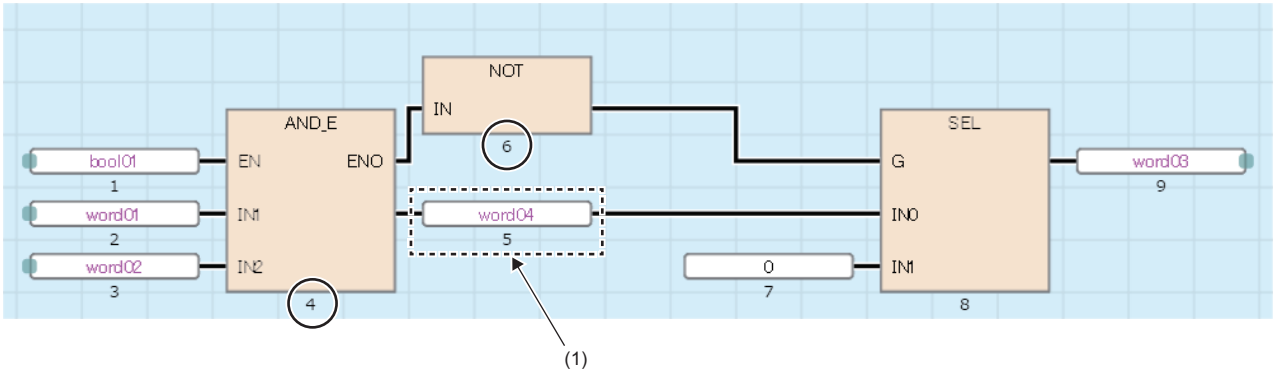


注意事項

如果是使用函數的程式，應不直接連接函數的返回值與其他函數的輸入變數，而是在中間連接變數部件。

例

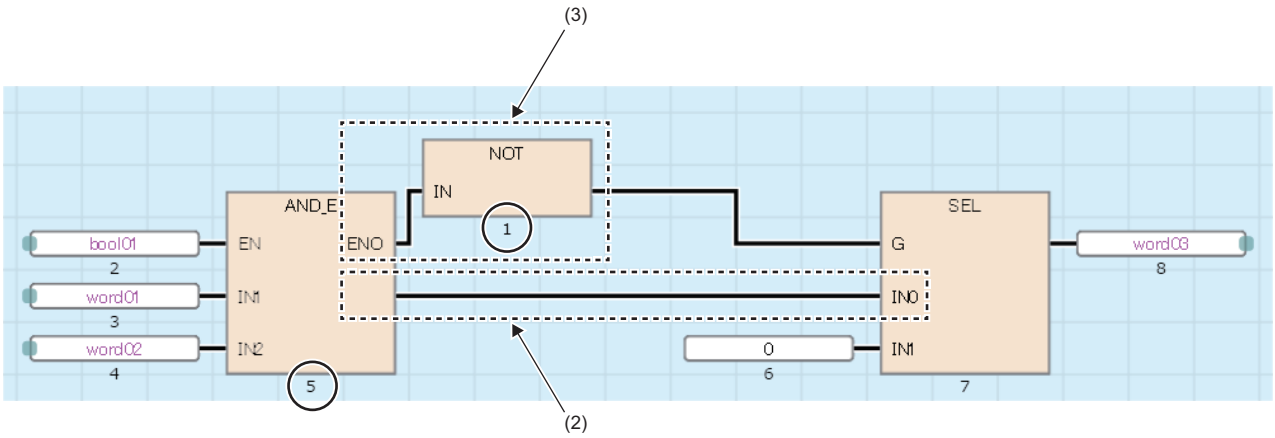
在返回值與輸入變數之間連接了變數部件(1)的情況下



如果直接連接函數的返回值與輸入變數，有可能變為預想外的執行順序。

例

不為預想的執行順序的情況下



配置在後方的部件的輸入連接了配置在前方的部件的返回值(2)與經由其他部件的輸出變數(3)，所以執行順序不同。

8 SFC程式

RnCPU RnENCPU RnPCPU(通程) RnPCPU(二重化) RnPSFCPU(常規) RnPSFCPU(安全) RnSFCPU(常規) RnSFCPU(安全)

是將一系列的控制動作分割為多個步，以便可清楚地表示各程式的執行順序及執行條件的程式的記述形式。

要點

在本章中，對SFC程式的動作及規格有關內容進行說明。關於本章中未記載的內容，請參閱下述手冊。

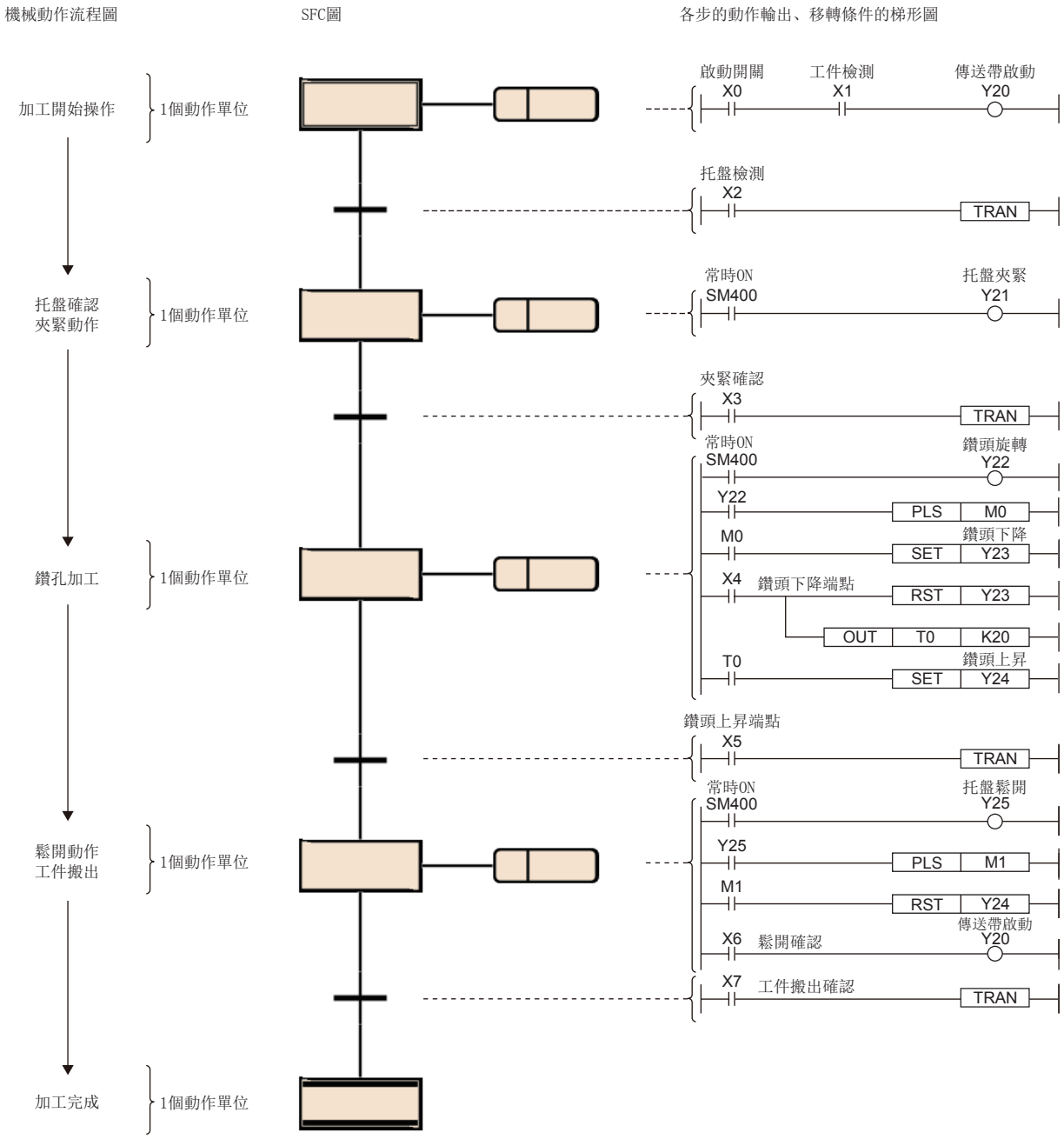
GX Works3 操作手冊
MELSEC iQ-R CPU模組用戶手冊(應用篇)

限制事項

欲使用SFC程式時，應確認CPU模組及工程工具的版本。關於CPU模組及工程工具的版本，請參閱下述手冊。

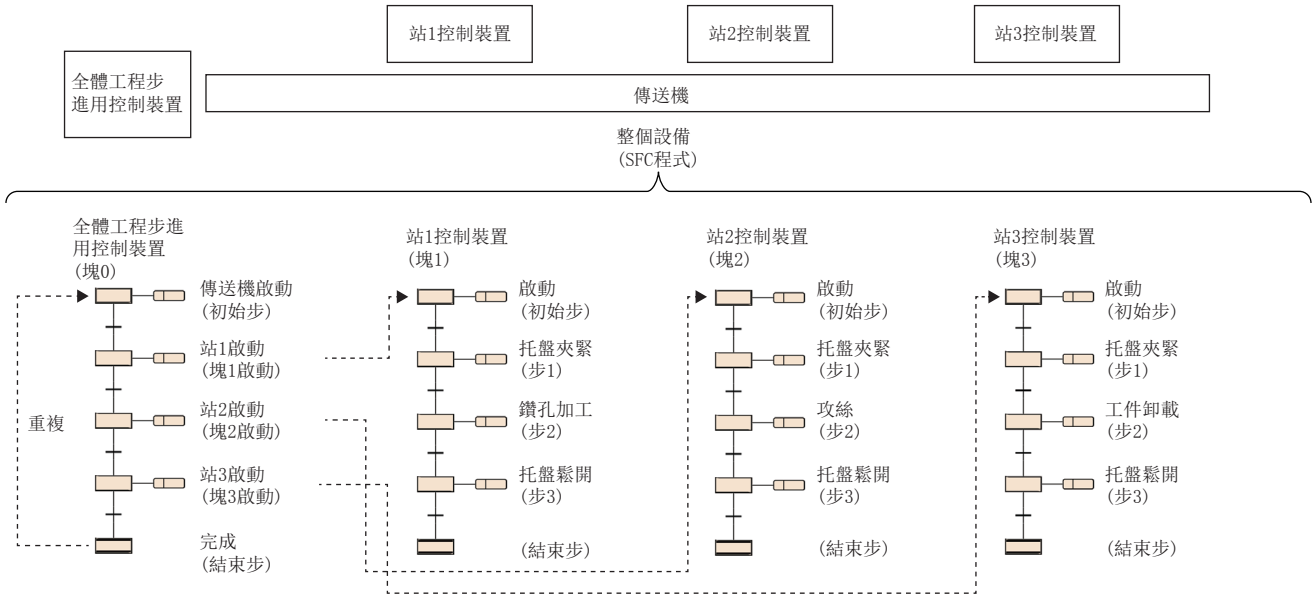
MELSEC iQ-R CPU模組用戶手冊(應用篇)

SFC程式將機械一系列動作中的各動作單位表示為1個步。
在各步中對實際的精細控制的程式進行創建。



SFC程式從初始步開始，每當移轉條件成立時按照順序執行下一個步的動作輸出，並透過結束步結束一系列的動作。

可以將整個設備、各站的機械裝置、各機械的實際控制與SFC程式的各塊、各步以1對1對應。



8.1 規格

SFC程式相關的性能規格如下所示。

項目		規格
元件點數(SFC相關)	步繼電器(S)	R00CPU、R01CPU、R02CPU： 最多8192點 上述以外的CPU模組： 最多16384點
	SFC塊元件(BL)	R00CPU、R01CPU、R02CPU： 128點 上述以外的CPU模組： 320點
	SFC移轉元件(TR)	0點
SFC程式執行個數		1個
塊數		R00CPU、R01CPU、R02CPU： 最多128塊 上述以外的CPU模組： 最多320塊
SFC步數		R00CPU、R01CPU、R02CPU： 全塊中最多1024步、1塊中最多128步 上述以外的CPU模組： 全塊中最多16384步、1塊中最多512步
步No.		R00CPU、R01CPU、R02CPU： 每1塊0～127 上述以外的CPU模組： 每1塊0～511
分支數		最多32分支
同時激活步數		R00CPU、R01CPU、R02CPU： 全塊中最多1024步、1塊中最多128步 上述以外的CPU模組： 全塊中最多1280步、1塊中最多256步
初始步數		最多32個/塊
動作輸出數		最多4個/步
順控程式步數	動作輸出	1個塊約32K順控程式步 (每1步無限制)
	移轉條件	僅1個梯形圖塊
SFC塊RUN中寫入的對象塊數		1塊*1

*1 SFC塊RUN中寫入的有關內容，請參閱下述章節。
☞137頁 SFC塊RUN中寫入
此外，欲使用SFC塊RUN中寫入的情況下，應確認CPU模組及工程工具的版本。(MELSEC iQ-R CPU模組用戶手冊(應用篇))

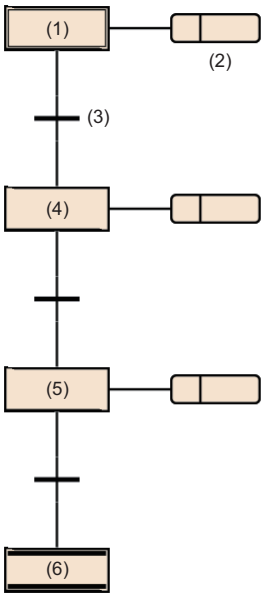
要點

SFC程式的處理時間的有關內容，請參閱下述手冊。
MELSEC iQ-R CPU模組用戶手冊(應用篇)

8.2 配置

SFC的基本動作

SFC程式從初始步開始，每當移轉條件成立時執行下一個步，並透過結束步結束一系列的動作。



(1) 初始步
(2) 動作輸出
(3) 移轉條件
(4) 普通步
(5) 普通步
(6) 結束步

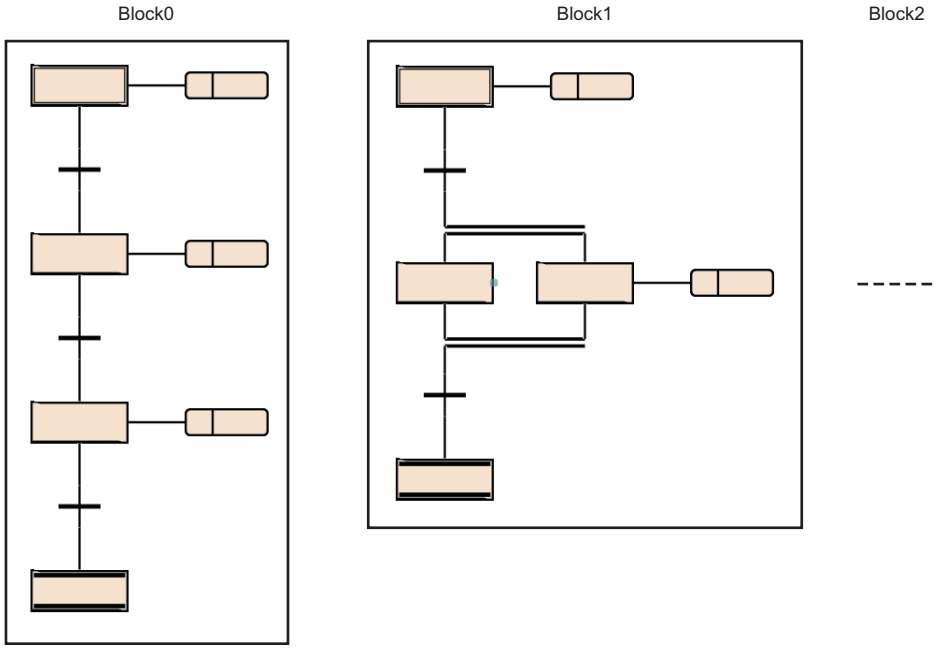
1. 啟動塊時，首先激活初始步(1)，再執行動作輸出(2)。動作輸出(2)執行後將檢查下一個移轉條件(3)是否成立。
2. 在移轉條件(3)成立之前，僅執行動作輸出(2)。移轉條件(3)成立時將結束動作輸出(2)，在初始步(1)變為非激活狀態後激活下一個普通步(4)。
3. 在執行普通步(4)的動作輸出後，檢查下一個移轉條件是否成立。如果下一個移轉條件未成立，將重複執行普通步(4)的動作輸出。
4. 移轉條件成立時將結束動作輸出，在步(4)變為非激活狀態後激活下一個步(5)。
5. 每當移轉條件成立時將激活下一個步，在最後激活結束步(6)後結束塊。

要點

- 1個步最多可創建4個動作輸出。創建了多個動作輸出的情況下，將從上面開始按順序執行。(☞94頁 動作輸出)
- 初始步與普通步，透過賦予屬性，可以更改步的類型。(☞83頁 步的類型)

塊

塊由步及移轉條件配置，是表示一系列動作的單位。



在SFC程式中最多可以創建的塊數，請參閱下述章節。

☞79頁 規格

塊內為從初始步開始交互連接步及移轉條件，並以結束步或跳轉移轉結束的配置。

塊具有激活/非激活的狀態。

- 激活： 塊內存在激活步的狀態
- 非激活： 塊內的所有步處於非激活的狀態

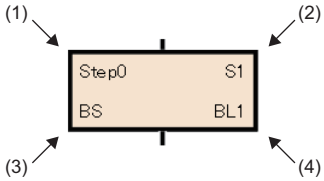
塊從非激活變為激活時，初始步將變為激活，依次執行處理。(☞127頁 各塊的執行順序)

要點

- 透過CPU參數的設置，僅塊0在SFC程式啟動時可以自動啟動。在這種情況下激活結束步並結束塊0時，塊0將自動被再啟動並再次從初始步開始執行。(☞121頁 啟動條件設置)
- 透過SET指令(步啟動)在非激活塊的步中存在啟動請求的情況下，激活塊後，從指定的步開始執行處理。

步

步是指用於配置塊的基本單位。



(1) 步名
(2) 步No.
(3) 屬性
(4) 屬性指定目標

在1塊中最多可以創建的步數，請參閱下述章節。

☞79頁 規格

步有下述特徵。

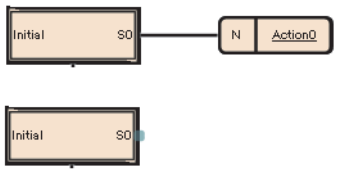
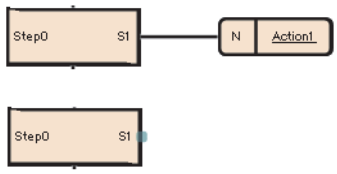
- 步激活時，執行相關的動作輸出。
- 步No.被添加在各步中。步No.用於監視執行步的情況，或透過SFC控制指令進行強制啟動或強制停止的情況。(☞92頁 分配步繼電器(S)至步中)
- 在各塊內，步名及步No.是特有的。(不可以空欄。)

要點

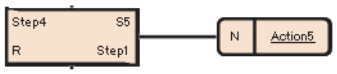
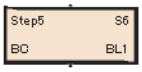
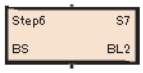
步名、步No.、屬性、屬性指定目標可以透過步的屬性畫面進行更改。
選擇步並選擇選單的[編輯]⇨[內容]時，將顯示步的屬性畫面。(GX Works3 操作手冊)

步的類型

步的類型如下所示。

項目		內容
初始步		表示塊的起始的步。 當步為激活中時，始終對該步的下一個移轉條件進行檢查，並在移轉條件成立時移轉激活到下一個步。 可以附加SC、SE、ST、R的屬性。 也可以置為不創建動作輸出的步。
普通步		配置塊的基本的步。 當步為激活中時，始終對該步的下一個移轉條件進行檢查，並在移轉條件成立時移轉激活到下一個步。 可以附加SC、SE、ST、R、BC、BS的屬性。 也可以置為不創建動作輸出的步。
結束步		使塊結束的步。 無法創建動作輸出。

步的屬性如下所示。

屬性	項目		內容
SC	線圈保持步[SC]		激活移轉後也保持透過動作輸出變為ON的線圈的輸出的步。
SE	動作保持步(無移轉檢查)[SE]		激活移轉後繼續執行動作輸出的步。 移轉條件成立，下一個步激活後將不進行移轉條件的檢查。
ST	動作保持步(有移轉檢查)[ST]		激活移轉後繼續執行動作輸出的步。 移轉條件成立且下一個步被激活後，也將重複進行移轉條件的檢查。
R	復位步[R]		是將指定步置為非激活的步。
BC	塊啟動步(有結束檢查)[BC]		使指定塊激活的步。 指定塊變為非激活且移轉條件成立時，激活將移轉到下一個步。 無法創建動作輸出。
BS	塊啟動步(無結束檢查)[BS]		使指定塊激活的步。 移轉條件成立時，激活將移轉到下一個步。 無法創建動作輸出。

要點

- 透過在步的屬性畫面中，對“屬性”的設置進行更改可以更改步的類型。
- 對於復位步[R]、塊啟動步(有結束檢查)[BC]、塊啟動步(無結束檢查)[BS]，在屬性畫面的“屬性指定目標”中指定步名或塊No.。

設置方法的有關內容，請參閱下述手冊。

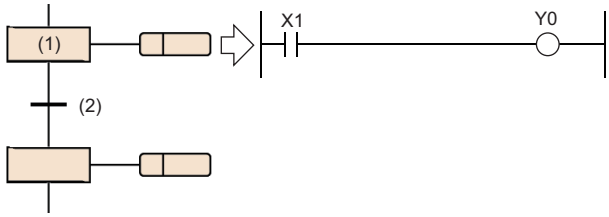
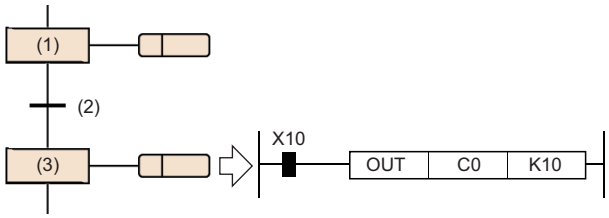
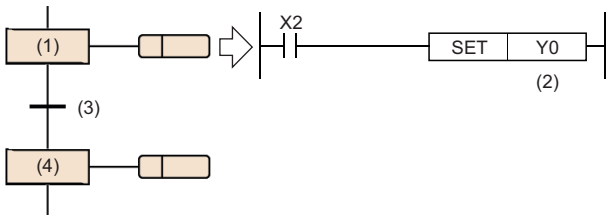
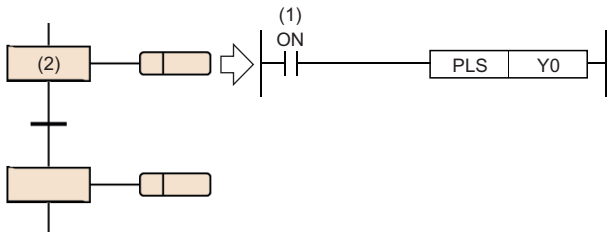
GX Works3 操作手冊

普通步(無屬性)

配置塊的基本的步。

當步為激活中時，始終對該步的下一個移轉條件進行檢查，並在移轉條件成立時移轉激活到下一個步。

步的動作輸出根據使用的指令，移轉至下一個步時的輸出狀態將有所不同。

項目	內容	示例
使用OUT指令時 (OUT C指令以外)	激活移轉至下一個步後變為非激活時，透過OUT指令的輸出將自動OFF。 定時器也一樣，對當前值進行清除後將觸點置為OFF。 但是，透過在ST語言的選擇語句或重複語句內使用的OUT指令的輸出，將不自動變為OFF。	 在步(1)的動作輸出中透過OUT指令將Y0置為ON的情況下，移轉條件(2)成立時Y0將自動變為OFF。
使用OUT C指令時	位於動作輸出的計數器的執行條件已處於ON狀態時，移轉條件成立且該步激活時，將被計數1次。 在執行計數器的復位指令之前激活移轉至了下一個步的情況下，即使該步變為非激活狀態也將保持計數器的當前值及觸點的ON狀態。 對計數器進行復位的情況下，在其他步中將執行RST指令。	 在步(1)激活時X10已處於ON狀態的情況下，移轉條件(2)成立且移轉到步(3)時，計數器C0將進行1次計數。
使用SET指令、基本指令或應用指令時	激活移轉至下一個步後，即使該步變為非激活狀態，也將保持ON狀態或元件/標籤中儲存的資料。 ON狀態的元件/標籤的OFF或元件/標籤中儲存的資料的清除，應在其他步中透過RST指令等進行。	 在步(1)的動作輸出中透過SET指令將Y0置為ON的情況下(2)，即使移轉條件(3)成立且移轉到步(4)，Y0的ON仍將被保持。
使用PLS指令、上升沿指令時	即使執行條件的觸點處於常時ON狀態的情況下，每當該步從非激活狀態變為激活狀態時也將執行指令。	 即使執行條件觸點為常時ON(1)，每當步(2)變為激活狀態時也將執行PLS指令。

■無動作輸出的步

不創建動作輸出的步，可以作為等待用的步使用。

- 步的激活過程中，始終對移轉條件進行檢查，移轉條件成立時將激活移轉至下一個步。
- 創建動作輸出時，將作為普通的步進行動作。

初始步

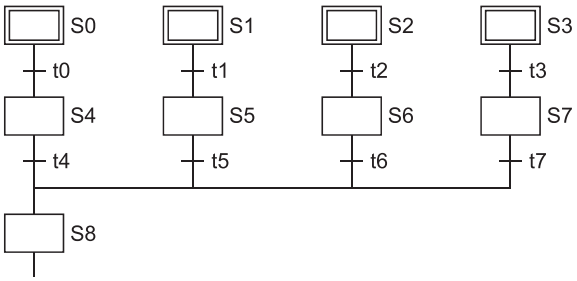
是表示各塊的起始的步，各塊中最多可記述32個。(☞ 79頁 規格) 對多個初始步進行合併時只能進行選擇合併。初始步的執行方法與初始步以外的步相同。

■塊啟動時的激活步

在初始步為多個步的情況下，在塊中執行了啟動時的激活步根據啟動方法，其動作如下所示。

激活步的動作	啟動方法
初始步全部被激活	透過塊啟動步執行了啟動時
	透過SFC控制指令的塊啟動指令執行了啟動時
	透過SFC用資訊元件的塊啟動結束位元強制執行了啟動時
	透過塊0的自動啟動設置啟動了塊0時
僅指定步被激活	透過SFC控制指令的步控制指令指定了其中一個初始步時

■初始步為多個激活步時的移轉處理



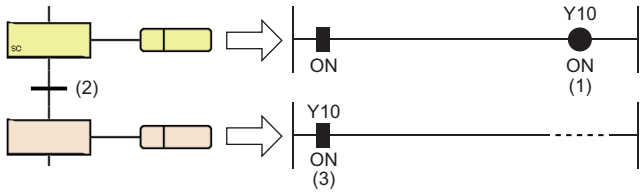
選擇合併初始步為多個激活步的塊的情況下，如果合併之前其中一個移轉條件成立，則合併之後的步將被激活。在上述程式示例中，如果移轉條件t4～t7中的其中一個條件成立，則步8(S8)將被激活。此外，合併之後的步(在上述程式示例中為S8)被激活後，合併之前的其他移轉條件(在上述程式示例中為t4～t7)成立時，合併之後的步將再次被激活。

■初始步中附加步屬性時的動作

對於初始步，可以附加SC(線圈保持)、SE(動作保持(無移轉檢查))、ST(動作保持(有移轉檢查))、R(復位)的各屬性。此外，附加了屬性的情況下，塊啟動時自動被激活的動作以外皆與初始步以外的步相同。此外，也可置為無動作輸出的步。

線圈保持步[SC]

激活移轉後也保持透過動作輸出變為ON的線圈的輸出的步。



透過OUT指令變為ON狀態的Y10(1)，即使移轉條件(2)成立也不變為OFF，而是保持ON狀態不變(3)。

移轉條件成立，且移轉至下一個步後，不進行動作輸出內的運算。因此，即使動作輸出內的輸入條件變化，線圈輸出的狀態也不變化。

■線圈輸出OFF的時機

在移轉後的線圈保持步[SC]中，保持為ON狀態的線圈輸出變為OFF的時機如下所示。

- 執行了塊的結束步的情況下(SM327為ON時除外)
- 透過SFC控制指令的RST指令(塊結束)對塊進行了強制結束的情況下
- 透過SFC控制指令的RST指令(步結束)對步進行了復位的情況下
- 對SFC用資訊元件的塊啟動結束位元中指定的元件進行了復位的情況下
- 所設置的用於復位線圈保持步[SC]的復位步[R]被激活的情況下
- 將SM321(SFC程式的啟動/停止)置為了OFF的情況下
- 透過程式對線圈進行了復位的情況下
- 停止時輸出模式為OFF的狀態下開始了停止指令的情況下
- 在塊內的復位步[R]中指定了S999的情況下

■塊停止/重啟時的動作

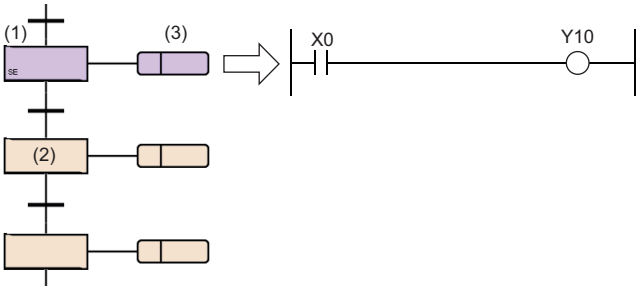
塊停止/重啟時的動作，根據SM325(塊停止時的輸出模式設置)與SFC用資訊元件的停止時模式位元的設置、步的保持/非保持的組合來決定。(☞ 122頁 塊停止重啟時的動作)

動作保持步(無移轉檢查)[SE]

激活移轉後繼續執行動作輸出的步。

透過移轉條件的成立移轉到下一個步後，仍將繼續進行動作輸出內的運算。因此，輸入條件變化時線圈的狀態也將發生變化。

移轉條件成立且下一個步被激活後不進行移轉條件的檢查，即使移轉條件再次成立，也不進行至下一個步的移轉。



從步(1)移轉至步(2)時，步(1)將變為保持中。
保持中時，不進行移轉檢查，但是動作輸出(3)將繼續執行。
在這種情況下，根據X0的ON/OFF，Y10將變為ON/OFF。

■變為非激活的時機

動作保持步(無移轉檢查)[SE]變為非激活狀態的時機如下所示。

- 執行了塊的結束步的情況下
- 透過SFC控制指令的RST指令(塊結束)對塊進行了強制結束的情況下
- 透過SFC控制指令的RST指令(步結束)對步進行了復位的情況下
- 對SFC用資訊元件的塊啟動結束位元中指定的元件進行了復位的情況下
- 所設置的用於復位動作保持步(無移轉檢查)[SE]的復位步[R]被激活的情況下
- 將SM321(SFC程式的啟動/停止)置為了OFF的情況下
- 在塊內的復位步[R]中指定了S999的情況下

■塊停止/重啟時的動作

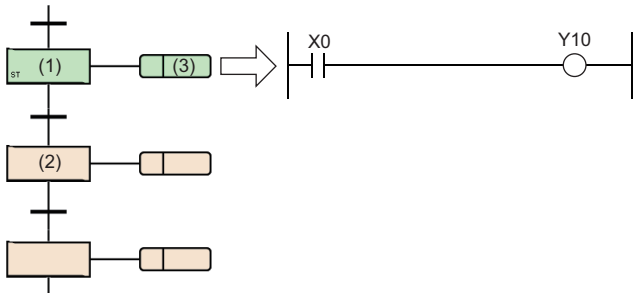
塊停止/重啟時的動作，根據SM325(塊停止時的輸出模式設置)與SFC用資訊元件的停止時模式位元的設置、步的保持/非保持的組合來決定。(☞ 122頁 塊停止重啟時的動作)

動作保持步(有移轉檢查)[ST]

激活移轉後繼續執行動作輸出的步。

透過移轉條件的成立移轉到下一個步後，仍將繼續進行動作輸出內的運算。因此，輸入條件變化時線圈的狀態也將發生變化。

移轉條件成立且下一個步被激活後，也將重複進行移轉條件的檢查。移轉條件再次成立時，則再次使下一個步激活的同時，繼續進行動作輸出內的運算的繼續運行。



從步(1)移轉至步(2)時，步(1)將變為保持中。
保持中時與普通的激活步一樣，動作輸出(3)也將繼續執行。
在這種情況下，根據X0的ON/OFF，Y10將變為ON/OFF。
也另外進行移轉檢查，並在移轉條件成立時激活下一個步。

■變為非激活的時機

動作保持步(有移轉檢查)[ST]變為非激活狀態的時機如下所示。

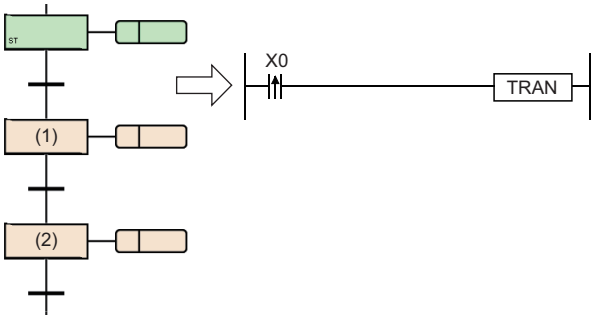
- 執行了塊的結束步的情況下
- 透過SFC控制指令的RST指令(塊結束)對塊進行了強制結束的情況下
- 透過SFC控制指令的RST指令(步結束)對步進行了復位的情況下
- 對SFC用資訊元件的塊啟動結束位元中指定的元件進行了復位的情況下
- 所設置的用於復位動作保持步(有移轉檢查)[ST]的復位步[R]被激活的情況下
- 將SM321(SFC程式的啟動/停止)置為了OFF的情況下
- 在塊內的復位步[R]中指定了S999的情況下

■塊停止/重啟時的動作

塊停止/重啟時的動作，根據SM325(塊停止時的輸出模式設置)與SFC用資訊元件的停止時模式位元的設置、步的保持/非保持的組合來決定。(☞ 122頁 塊停止重啟時的動作)

■注意事項

- 對於動作保持步(有移轉檢查)[ST]，在之後的移轉條件成立期間，每個掃描下一個步將被啟動。欲使每個掃描不進行移轉時，應在移轉條件中使用PLS指令等執行上升沿的指令。



透過將移轉條件置為上升沿脈衝運算開始的條件，僅X0變為ON的瞬間的1個掃描可啟動步(1)。
即使從步(1)移轉至步(2)、步(1)變為非激活狀態，如果X0不再次變為OFF→ON，則步(1)將不啟動。

- SM328(END步到達時清除處理模式)為ON的情況下，應將動作保持步(有移轉檢查)[ST]之後的移轉條件置為不常時成立。下一個步始終為非保持的激活狀態，因此無法結束塊。

復位步[R]

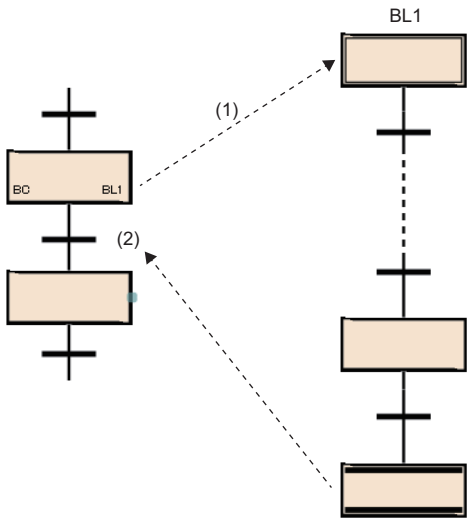
是將指定步置為非激活的步。

- 復位步[R]在執行每個掃描動作輸出之前，將本塊內的指定步置於非激活狀態。復位指定步之外將與普通的步(無屬性)相同。
- 指定的步No.為S999的情況下，將本塊內保持中的保持步[SC、SE、ST]全部置為非激活狀態。在這種情況下，僅保持中的保持步[SC、SE、ST]可進行非激活。動作保持步[SE、ST]以非保持進行動作時，將不變為非激活的對象。
- 本步No.不可以指定到指定步No.中。

塊啟動步(有結束檢查)[BC]

使指定塊激活的步。

指定塊變為非激活且移轉條件成立時，激活將移轉到下一個步。



塊啟動步(有結束檢查)[BC]被激活時，對塊(BL1)進行啟動(1)。
在啟動目標塊(BL1)的執行結束後變為非激活狀態之前將處於無處理狀態，不進行移轉條件(2)的檢查。
在塊(BL1)的執行結束後變為非激活狀態時，僅進行移轉條件(2)的檢查，如果移轉條件(2)成立則移轉到下一個步。

對1個塊同時進行了啟動或對已啟動的塊進行了啟動時的動作，將按照塊雙重啟動時的運行設置執行。(☞124頁 塊雙重啟動時的運行設置)

可指定的塊僅1個。同時啟動多個塊的情況下，在使用並聯分支後再使用多個塊啟動步。

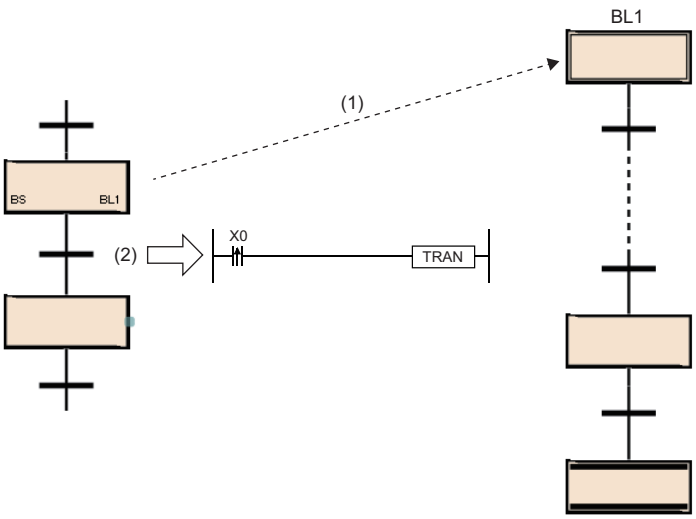
■注意事項

- 無法將動作輸出創建到塊啟動步(有結束檢查)[BC]中。
- 在並聯合併的合併之前無法創建塊啟動步(有結束檢查)[BC]。在並聯合併的合併之前進行創建的情況下，使用塊啟動步(無結束檢查)[BS]。

塊啟動步(無結束檢查)[BS]

使指定塊激活的步。

移轉條件成立時，激活將移轉到下一個步。



在塊啟動步(無結束檢查)[BS]對塊(BL1)進行了啟動(1)之後，僅進行移轉條件(2)的檢查，如果條件成立則不等待啟動目標塊(BL1)結束而移轉至下一個步。

對1個塊同時進行了啟動或對已啟動的塊進行了啟動時的動作，將按照塊雙重啟動時的運行設置執行。(☞ 124頁 塊雙重啟動時的運行設置)

可指定的塊僅1個。同時啟動多個塊的情況下，在使用並聯分支後再使用多個塊啟動步。

■注意事項

- 無法將動作輸出創建到塊啟動步(無結束檢查)[BS]中。

結束步

使塊結束的步。

- 激活移轉到結束步且塊內不存在保持中以外的激活步時，將塊內的全部保持中步[SC、SE、ST]置為非激活狀態後結束塊。
- 塊內存在保持中以外的激活步的情況下，根據SM328(到達END步時清除處理模式)的狀態進行下述處理。

SM328的狀態	內容
OFF(預設)	進行清除處理。 將塊內剩餘的激活步全部強制結束後，結束塊。
ON	不進行清除處理。 在保持狀態不變的情況下繼續塊的執行，不結束塊。

- 在執行清除處理時，將透過OUT指令進行的線圈輸出全部置為OFF。但是，對於保持中步[SC、SE、ST]的線圈輸出，將根據SM327(END步執行時的輸出)的狀態進行下述處理。

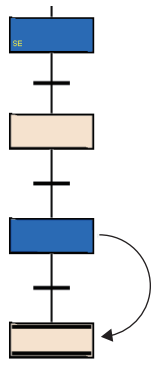
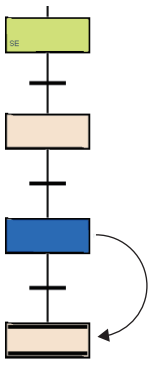
SM327的狀態	內容
OFF(預設)	將保持中步[SC、SE、ST]的輸出全部置為OFF。
ON	將保持中步[SC、SE、ST]的輸出全部保持。 SM327的設置僅對保持中的保持步[SC、SE、ST]有效。移轉條件未成立且不處於保持中的保持步[SC、SE、ST]的輸出將全部OFF。此外即使SM327為ON時，步也將由激活狀態變為非激活狀態。 但是，透過塊結束指令等進行強制結束的情況下，所有步的線圈輸出將OFF。

- 塊結束後使塊再次啟動的方法如下所示。

項目		內容
塊0	在參數的SFC設置中將塊0的啟動條件設置為“自動啟動”	自動地再次激活初始步，重複執行處理。
	在參數的SFC設置中將塊0的啟動條件設置為“不自動啟動”	透過下述方法，有對指定塊的啟動請求時進行再啟動。 • 在其他塊中將塊啟動步激活。 • 執行SFC控制指令的SET指令(塊啟動)。 • 將SFC用資訊元件的塊啟動結束位元置為ON。
塊0以外的所有塊		

■注意事項

- 無法將動作輸出創建到結束步中。
- 僅激活移轉到結束步中的情況下，SM327(END步執行時的輸出)的設置為有效。透過RST指令(塊結束)等進行強制結束的情況下，會將所有步的線圈輸出置為OFF。
- 在激活移轉到結束步中時僅剩下保持中步[SC、SE、ST]的情況下，即使SM328(END步到達時清除處理模式)為ON，該保持中步[SC、SE、ST]將變為非激活狀態。不希望將保持中步[SC、SE、ST]的線圈輸出置為OFF的情況下，應將SM327置為ON。SM328與保持步[SC、SE、ST]的動作的關係如下所示。

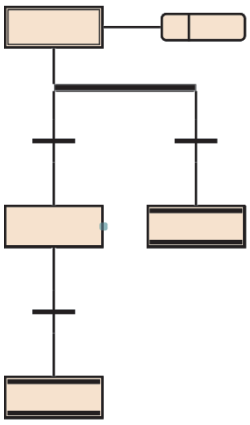
剩下普通的激活步、或剩下移轉不成立的保持步[SC、SE、ST]的情況下(非保持)	剩下保持中的激活步的情況下
	
• SM328為OFF時，進行清除後結束塊。 • SM328為ON時，不進行清除而繼續進行處理。	• 與SM328的設置無關，進行清除後結束塊。

- 如果塊在SM328為ON時透過塊啟動步被啟動，則將在非保持的激活中步不存在於塊內時返回到原來的塊處理。
- 應將動作保持步(有移轉檢查)[ST]之後的移轉條件置為不常時成立。動作保持步(有移轉檢查)[ST]之後的移轉條件為常時成立的情況下，由於下一個步變為始終激活狀態，因此SM328為ON時無法結束塊。

要點

在SFC圖內可以創建多個結束步。

對選擇分支中的步進行選擇且選擇選單的[編輯]⇨[變更]⇨[結束步序/定位]時，可以創建多個結束步。



分配步繼電器(S)至步中

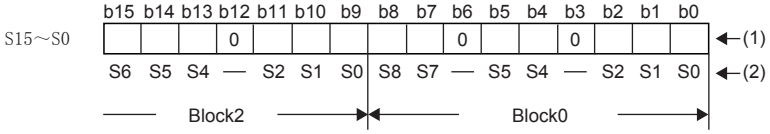
步繼電器是對應SFC程式中的各步的元件。在步處於激活中(也包括停止中、保持中)為ON、處於非激活狀態時為OFF。
步繼電器按下述方式被分配。

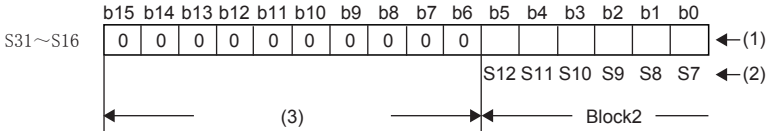
- 步繼電器，從SFC程式的塊0開始按塊No.順序，在1個塊內按步No.順序從起始開始向末尾進行分配。
- 對於不存在的塊No.，將不分配步繼電器。
- 在1個塊內，缺少編號的步No.中也將分配步繼電器。該位元始終為OFF。
- 最後的塊中分配的步繼電器以後的位元全為OFF。

例

下述配置塊時的步繼電器分配如下所示。

Block0： 最大步No.為8，且步No.3及步No.6不存在。
Block1： 不存在。
Block2： 最大步No.為12，且步No.3不存在。
Block3及其以後： 不存在。





(1) 儲存的資料
(2) 塊內的步No.
(3) 由於不存在，因此全部為0

要點

可自由對各步(結束步除外)分配步No.。

- 步No.有遺漏編號時，可創建的最大步數將變少，因此應盡可能從小編號開始往大編號按順序進行創建。
- 在最上行的左端初始步中，不可以使用步No.0(S0)以外。

對塊的最初的初始步中分配塊No.0。
在每1塊中最多可以使用的步No.，請參閱下述章節。
☞79頁 規格
不可以分配超出上限的步No.。此外在同一塊內，步No.不可以重複。但是，在不同的塊中可以使用同一步No.。
指定其他塊的步繼電器的情況下，按下述形式進行指定。

例
指定塊No.12的步No.23的情況下

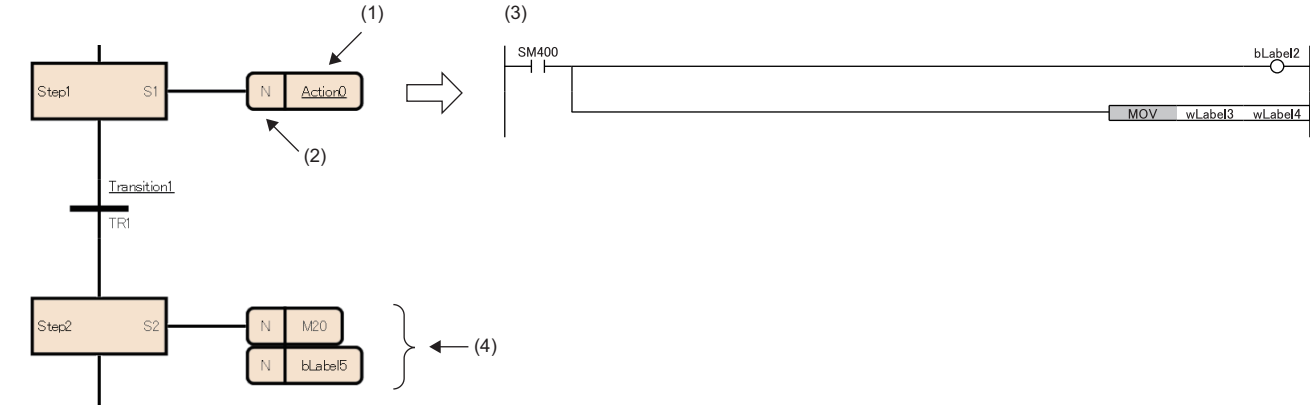
程式類型		元件記載	內容
SFC程式	同一塊內	S23	指定同一塊內的步的情況下，可以省略塊名。
	塊12以外	BL12\S23	指定塊No.及步No.。
SFC程式以外的順控程式	指定當前對象塊的情況下	S23	指定對象塊內的步的情況下，可以省略塊名。
	指定與當前對象塊不同的塊的情況下	BL12\S23	指定塊No.及步No.。

■注意事項

• 即使將SFC設置的“塊停止時的輸出模式”設置為OFF，停止中時步繼電器也為ON。

動作輸出

對於動作輸出，在步為激活中時執行的程式如下所示。



(1) 動作輸出名
(2) 修飾語*1
(3) 動作輸出的詳細表示
(4) 動作輸出的標籤/元件

*1 N表示在步為激活中時執行。無法設置N以外。

步進行激活時，每個掃描中將執行動作輸出。步變為非激活時，將結束動作輸出，變為非執行直至下一個步被激活為止。

1個步最多可創建4個動作輸出。創建了多個動作輸出的情況下，將從上面開始按順序執行。

動作輸出的詳細表示，可以使用梯形圖語言、ST語言、FBD/LD語言創建。如果是梯形圖語言，則可以切換成詳細表示與MELSAP-L(指令形式)。(☞95頁 MELSAP-L(指令形式)的動作輸出)

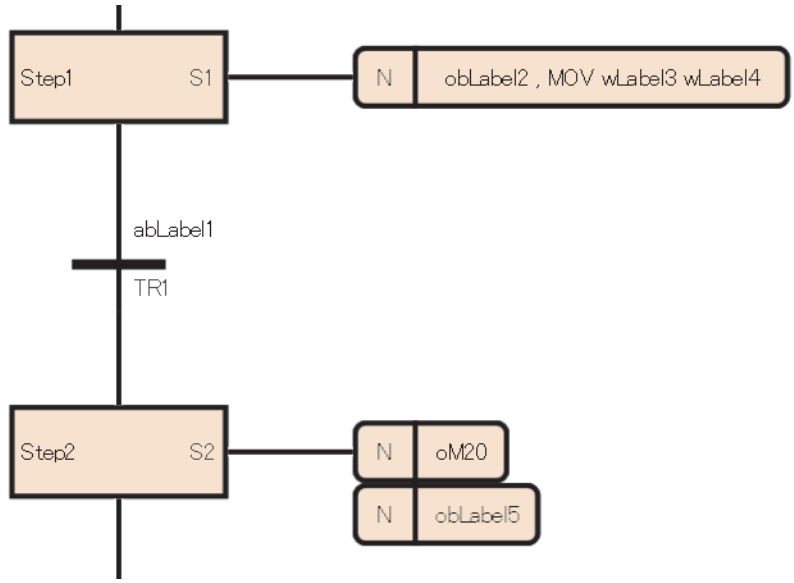
要點

關於詳細表示及標籤/元件的詳細內容，請參閱下述手冊。

GX Works3 操作手冊

MELSAP-L(指令形式)的動作輸出

MELSAP-L(指令形式)是在SFC圖內使用文本記述動作輸出指令的形式。



要點

從梯形圖的詳細表示切換為MELSAP-L(指令形式)的情況下，選擇選單的[顯示]⇨[梯形圖顯示切換]⇨[MELSAP-L(指令格式)]。(GX Works3 操作手冊)

基於MELSAP-L(指令形式)的動作輸出不記述為各指令輸入條件的觸點，而是記述指令或希望輸出的線圈。

MELSAP-L(指令形式)採用下述形式記述程式。

□：可使用的標籤/元件、Kn： 定時器/計數器的設置值

項目	MELSAP-L(指令形式)	記述示例
線圈輸出(OUT指令)	o□	oY0
元件的設置(SET指令)	s□	sM0
元件的復位(RST指令)	r□	rM0
低速定時器(OUT T指令)*1	o□ Kn	oT0 K100
高速定時器(OUTH T指令)	h□ Kn	hT1 K10
計數器(OUT C指令)*1	o□ Kn	oC0 K10
上述以外的指令*2	與梯形圖語言中的指令輸入同樣記述	MOV D10 D120

*1 長定時器、長計數器也同樣指定。

*2 部分指令無法使用。(☞ 96頁 無法使用的指令)

記述多個指令時，使用“,”(逗號)間隔。IMASK、NOPLF在動作輸出的最後記述。

無法使用的指令

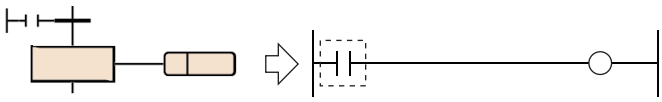
在動作輸出內一部分的指令無法使用。無法使用的指令如下所示。

分類	指令符號
主控制指令	MC*1
	MCR*1
結束指令	FEND
	END
程式分支指令	CJ*1
	SCJ*1
	JMP*1
	GOEND
程式執行控制指令	IRET
結構化指令	BREAK*1
	RET
移轉條件虛擬輸出	TRAN

*1　在動作輸出內的函數/FB內可以使用。

要點

對於詳細表示內的梯形圖，必須創建各指令的輸入條件的觸點。



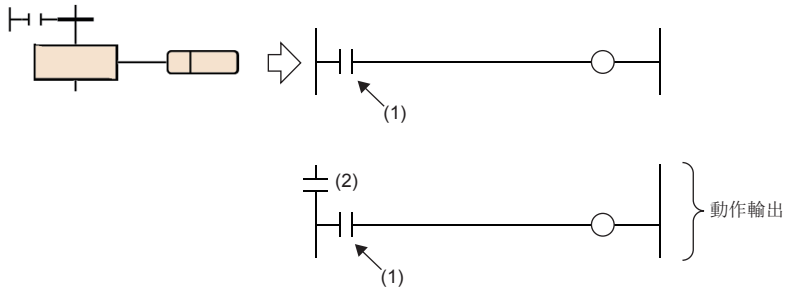
■限制事項

根據創建動作輸出的程式語言，有下述限制。

語言		內容
梯形圖語言	詳細表示	指針及中斷指針，無法輸入到指針輸入區中。 ■無法使用的函數/FB ・包括無法在動作輸出中使用的指令的函數/FB ・包括指針的函數/FB ・“使用MC/MCR控制EN”設置為“是”，且“使用EN/ENO”設置為“否”的宏型FB
	MELSAP-L(指令形式)	無法記述相當於觸點的指令(包含LD<等比較運算指令)、NOP、MPS、MRD、MPP、指針、中斷指針、函數、FB。
ST語言		☞47頁 ST語言
FBD/LD語言		☞63頁 FBD/LD語言

注意事項

• 步的動作將變為與下述梯形圖大致相同的動作。



(1)各指令的輸入條件
(2)顯示步的狀態的觸點(激活時：ON、非激活時：OFF)

• 要在步的動作輸出內進行子程式調用的情況下，如果使用CALL指令，即使移轉條件成立且步變為非激活狀態，CALL目標的輸出也不OFF。在移轉條件成立且步變為非激活狀態時，希望使CALL目標的輸出為OFF的情況下，應在CALL指令後記述FCALL指令或使用XCALL指令。要在步的動作輸出內進行子程式調用的情況下，如果使用XCALL指令可以減少步數。

• 即使動作輸出內的輸入條件為常時ON，在步為非激活狀態時，將被視為輸入條件為OFF。因此，步變為激活狀態之後，將在OFF→ON的條件下執行指令。例如，如果在PLS指令及INCP指令等的上升沿指令中將輸入條件置為了常時ON，則每當步被激活時指令將被執行。

• 透過動作輸出內的OUT C指令、SET指令、基本指令或應用指令等變為ON的元件，即使步變為非激活狀態、動作輸出結束也不變為OFF。將元件置為OFF時需要另外執行RST指令等。

• 通常，透過PLS指令及PLF指令，指定元件僅1個掃描變為ON，且會在之後變為OFF，但是當線圈保持步[SC]的移轉成立的同時，指定元件變為了ON的情況下，將繼續保持ON狀態。在這種情況下，進行將線圈保持步[SC]的線圈輸出變為OFF的條件，或透過再次激活步使元件變為OFF。關於線圈輸出OFF的條件，請參閱下述章節。

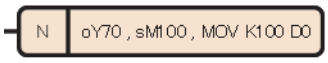
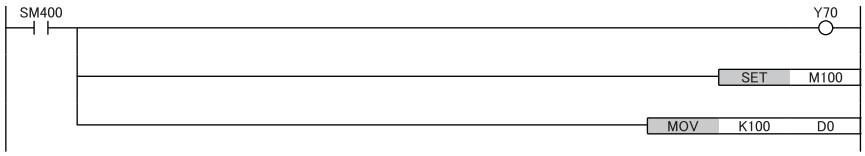
☞86頁 線圈輸出OFF的時機

• 在PLF指令的輸入條件為ON的情況下，當步變為非激活狀態且動作輸出結束時，指定元件將保持為ON狀態不變。

• 在線圈保持步[SC]中移轉條件成立的情況或SM325(塊停止時的輸出模式)設置為保持且停止了步的情況下，如果僅在保持輸出線圈時不進行動作，由於處於非執行的狀態，動作重啟時的各指令的動作將被變為非執行前的執行條件所左右。

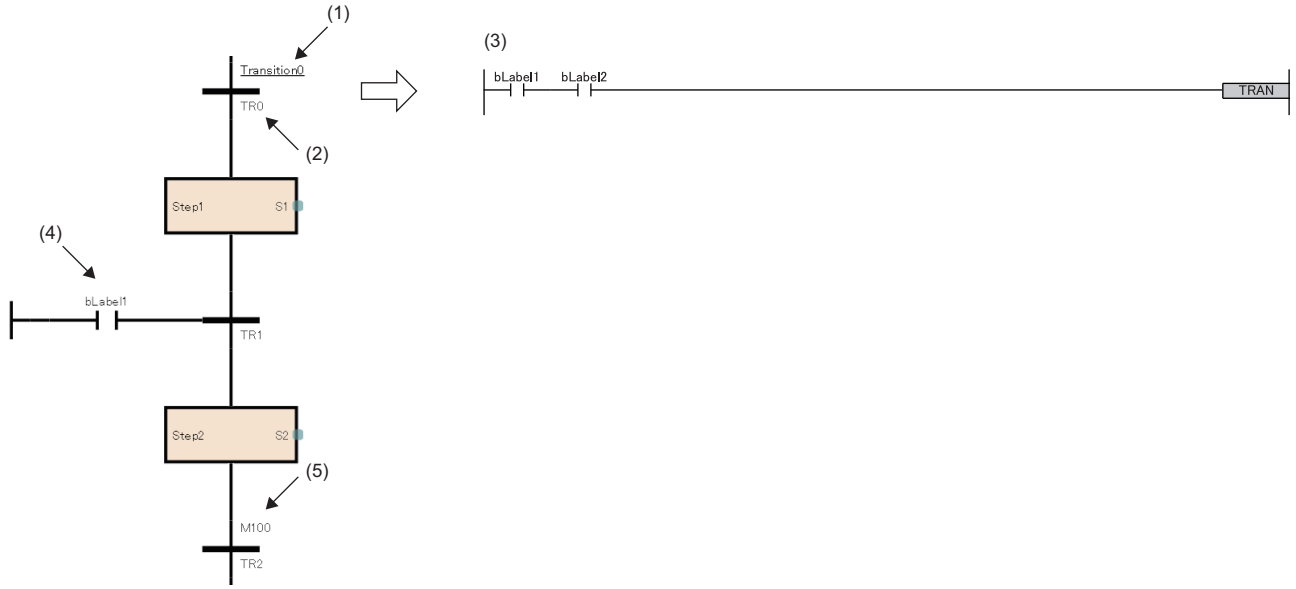
• 從梯形圖的詳細表示切換為MELSAP-L(指令形式)時，在詳細表示中創建了MELSAP-L(指令形式)無法記述的程式的情況下，在MELSAP-L(指令形式)中顯示為“????????”。此外，在程式內使用的標籤的定義被刪除的情況下也有相同顯示。希望進行程式的確認或修正的情況下，應切換為詳細表示。

• 以MELSAP-L(指令形式)創建的程式切換為梯形圖的詳細表示的情況下，作為指令的執行條件將添加SM400(常時ON)的觸點。

MELSAP-L(指令形式)	梯形圖的詳細表示
	

移轉條件

移轉條件是配置塊的基本單位，透過條件成立將激活移轉到下一個步。

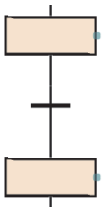
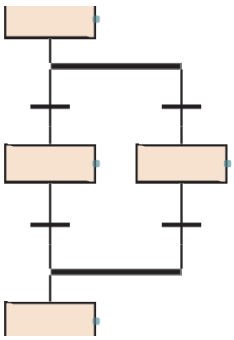
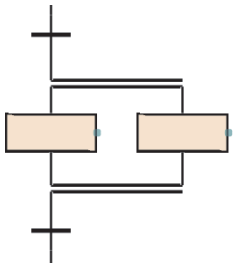
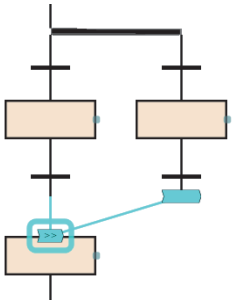


(1) 移轉條件名
(2) 移轉條件No.
(3) 移轉條件的詳細表示(☞104頁 移轉條件的詳細表示)
(4) 移轉條件的直接表示(☞107頁 移轉條件的直接表示)
(5) 移轉條件的標籤/元件(☞107頁 移轉條件的標籤/元件)

移轉條件的詳細表示，可以使用梯形圖語言、ST語言、FBD/LD語言創建。如果是梯形圖語言，則可以切換成詳細表示與MELSAP-L(指令形式)。(☞106頁 MELSAP-L(指令形式)的移轉條件)

移轉條件的類型

移轉條件的類型如下所示。

項目		內容
串聯移轉		如果移轉條件成立，則激活從先行的步移轉至後續的步。
選擇移轉(分支/合併)		分支： 從1個步分支為多個移轉條件，激活僅移轉至最先成立移轉條件的行的步。 合併： 在最先成立移轉條件的行執行合併前移轉條件成立時，激活將移轉至下一個步。
並聯移轉(分支/合併)		分支： 從1個步進行了分支的多個步將全部同時進行激活移轉。 合併： 如果合併之前的步全部激活，則透過通用的移轉條件成立，將激活移轉至下一個步。
跳轉移轉		透過移轉條件成立，激活移轉至同一塊內的指定的步。

要點

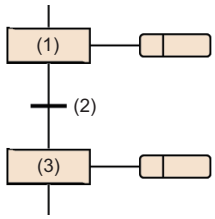
移轉至已激活的步時的動作的有關內容，請參閱下述章節。

☞ 134頁 步雙重啟動時的注意點

串聯移轉

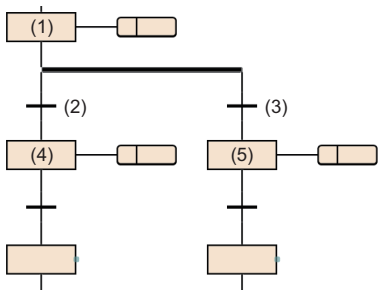
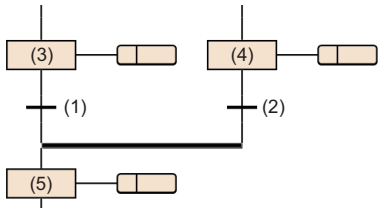
如果移轉條件成立，則激活從先行的步移轉至後續的步。

在步(1)處於激活狀態時移轉條件(2)成立時，將步(1)置於非激活狀態、將步(3)置於激活狀態。



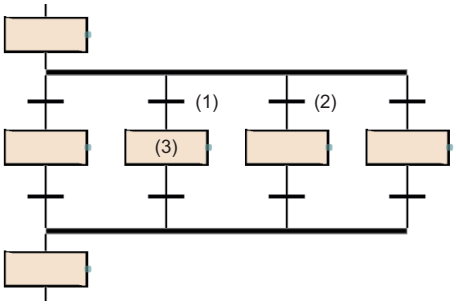
選擇移轉(分支/合併)

從1個步分支為多個移轉條件，激活僅移轉至最先成立移轉條件的行的步。在最先成立移轉條件的行執行合併前移轉條件成立時，激活將移轉至下一個步。

項目	內容	
分支		步(1)處於激活狀態時，將移轉條件(2)或移轉條件(3)之中條件先成立的步((4)或(5))置於激活狀態。 步(1)將變為非激活狀態。但是，保持步[SC、SE、ST]的情況下，將按照屬性保持線圈輸出或動作輸出。 • 多個移轉條件同時成立的情況下，將優先執行左側的移轉條件。 • 選擇後，依次執行所選擇行的各步直到進行合併為止。
合併		分支中激活的行的移轉條件((1)或(2))成立時，將步(5)置於激活狀態。 已激活的步(3)或步(4)將變為非激活狀態。但是，保持步[SC、SE、ST]的情況下，將按照屬性保持線圈輸出或動作輸出。

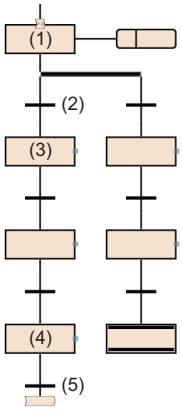
- 在選擇移轉中，最多可以分支為32個移轉條件。
- 多個移轉條件同時成立的情況下，將優先執行左側的移轉條件。

移轉條件(1)及(2)同時成立的情況下，執行步(3)的動作輸出。



- 也可創建選擇移轉的分支及合併的個數不相同的SFC圖。但是，無法創建選擇分支與並聯合併及並聯分支與選擇合併組合的SFC圖。
- 在選擇移轉中，可以透過跳轉移轉及結束步對合併進行省略。

進行步(1)的動作輸出時，如果移轉條件(2)成立，則從步(3)開始按順序執行步(4)。如果移轉條件(5)成立，則跳轉移轉到步(1)。



要點

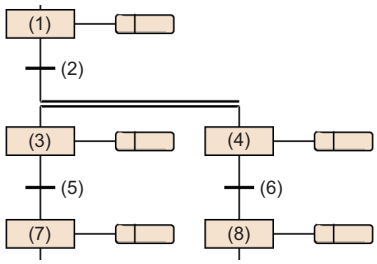
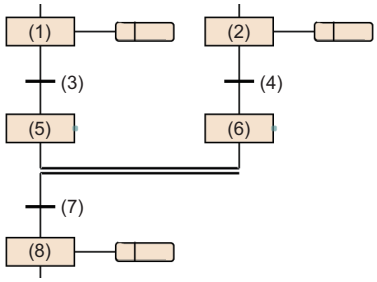
透過將選擇分支的左端以外的步更改為結束步，並將位於選擇分支的左端的結束步更改為跳轉後，可以創建上述程式。

關於對步進行更改的操作方法，請參閱下述手冊。

GX Works3 操作手冊

並聯移轉(分支/合併)

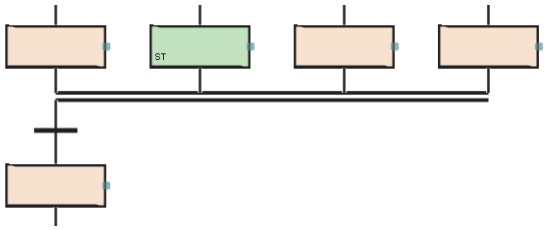
從1個步進行了分支的多個步將全部同時進行激活移轉。如果合併之前的步全部激活，則透過通用的移轉條件成立，將激活移轉至下一個步。

項目	內容	
分支		在步(1)處於激活狀態時移轉條件(2)成立時，將步(3)與步(4)同時置於激活狀態。 步(1)將變為非激活狀態。但是，保持步[SC、SE、ST]的情況下，將按照屬性保持線圈輸出或動作輸出。 移轉條件(5)成立時移轉到步(7)中，移轉條件(6)成立時移轉到步(8)中。
合併		步(1)與步(2)處於激活狀態時，在移轉條件(3)、移轉條件(4)成立後，將步(5)、步(6)置於激活狀態。 將合併之前的步(5)及步(6)全部激活後，檢查移轉條件(7)，並在移轉條件(7)成立後將步(8)置於激活狀態。 步(5)與步(6)將變為非激活狀態。但是，保持步[SC、SE、ST]的情況下，將按照屬性保持線圈輸出或動作輸出。

- 在並聯移轉中，最多可以移轉到32個步中。
- 透過並聯移轉啟動了其他塊的情況下，將同時執行啟動源的塊及啟動目標的塊。
- 一定會在並聯分支之後進行並聯合併。

■注意事項

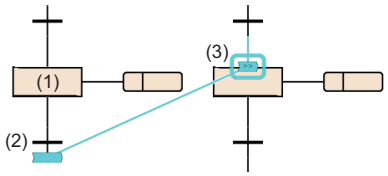
- 在並聯合併中，合併的步中存在有保持中步[SC、SE、ST]的情況下，將進行下述動作。

項目	內容
線圈保持步[SC]	與非激活步一樣，不移轉到下一個步中。
動作保持步(無移轉檢查)[SE]	
動作保持步(有移轉檢查)[ST]	其他合併的步處於激活狀態時，移轉到下一個步中。 

- 在並聯合併中，無法在合併之前對塊啟動步(有結束檢查)[BC]進行創建。應使用塊啟動步(無結束檢查)[BS]。

跳轉移轉

透過移轉條件成立，激活移轉至同一塊內的指定的步。

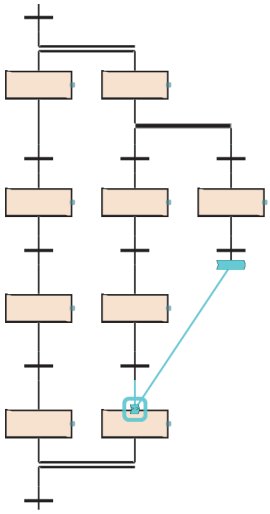


步(1)處於激活狀態時，在移轉條件(2)成立後，將步(3)置於激活狀態。
步(1)將變為非激活狀態。但是，保持步[SC、SE、ST]的情況下，將按照屬性保持線圈輸出或動作輸出。

- 跳轉移轉的使用個數無限制。
- 並聯移轉內的跳轉移轉，僅在同一分支內進行。無法創建至並聯分支內的不同分支的跳轉移轉、從並聯分支脫離的跳轉移轉、從並聯分支外至並聯分支的跳轉移轉。

例

並聯分支內可指定的跳轉移轉示例



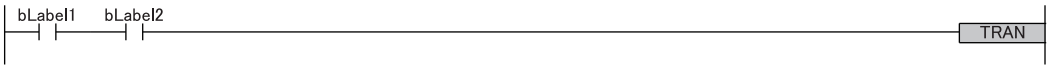
■注意事項

下述情況下，無法作為跳轉移轉的指定目標進行指定。

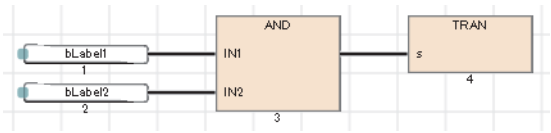
- 指定了從並聯移轉脫離的位置中的步的情況下
- 指定了進入並聯移轉的位置中的步的情況下
- 指定了先行的移轉條件之前的步的情況下
- 指定了本步的情況下

移轉條件的詳細表示

移轉條件的詳細表示在Zoom編輯器內創建。可以透過下述程式語言創建條件。

類型		內容
梯形圖語言	詳細表示	在單一的梯形圖塊中，可以創建由觸點的梯形圖及TRAN指令(移轉條件虛擬輸出)組成的移轉條件程式。如果執行TRAN指令，則移轉條件成立。  ■限制事項 • 不可以使用直接ST。 • 線圈中僅TRAN指令可以輸入。
	MELSAP-L(指令形式)	☞ 106頁 MELSAP-L(指令形式)的移轉條件

ST語言		可以創建下述移轉條件程式。
		■對TRAN函數(移轉條件虛擬輸出)的調用語句進行記述的方法
		TRAN(bLabel1 & bLabel2);
		//輸入引數的BOOL式為真(TRUE)的情況下，移轉條件成立。
		■對保留字“TRAN”的BOOL式的代入語句進行記述的方法
		TRAN := bLabel1 & bLabel2;
		//右邊的BOOL式為真(TRUE)的情況下，移轉條件成立。
		■對移轉條件名的BOOL式的代入語句進行記述的方法
		Transition1 := bLabel1 & bLabel2;
		//Transition1顯示在SFC編輯器上輸入的移轉條件名。右邊的BOOL式為真(TRUE)的情況下，移轉條件成立。

FBD/LD語言		在單一的梯形圖塊中，可以創建最後由TRAN指令(移轉條件虛擬輸出)組成的移轉條件程式。  ■限制事項 • TRAN指令只可以使用1個。 • 不可以創建代入至元件/標籤的程式。 • 不可以使用線圈部件、FB部件、函數部件(一部分可以)、跳轉部件/跳轉標籤部件、返回部件。 關於TRAN指令以外其他可使用的指令，請參閱下述章節。 ☞ 105頁 可使用指令

要點

- 相同移轉條件的詳細表示可以在多個移轉條件中使用。
- 創建後的詳細表示可以從Zoom一覽表中確認。(GX Works3 操作手冊)

■可使用指令

移轉條件的程式中可使用的指令如下所示。

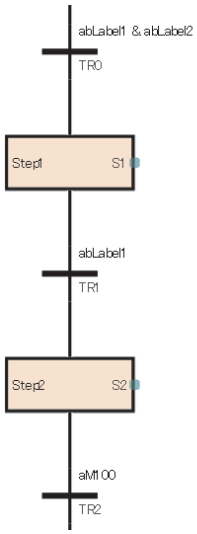
分類	指令符號
觸點指令	LD、LDI、AND、ANI、OR、ORI
	LDP、LDF、ANDP、ANDF、ORP、ORF
	LDPI、LDFI、ANDPI、ANDFI、ORPI、ORFI*2
合併指令	ANB、ORB
	INV
	MEP、MEF
	EGP、EGF*1
比較運算指令	LD□、LD□_U、AND□、AND□_U、OR□、OR□_U
	LDD□、LDD□_U、ANDD□、ANDD□_U、ORD□、ORD□_U
實數指令	LDE□、ANDE□、ORE□
	LDED□、ANDED□、ORED□
字元串處理指令	LD$□、AND$□、OR$□
移轉條件虛擬輸出	TRAN*2

*1　在ST語言、FBD/LD語言的移轉條件程式中，不可以使用EGP指令及EGF指令。

*2　在MELSAP-L(指令形式)的移轉條件程式中，無法使用LDPI、LDFI、ANDPI、ANDFI、ORPI、ORFI、TRAN。

MELSAP-L(指令形式)的移轉條件

MELSAP-L(指令形式)是在SFC圖內使用文本記述移轉條件的形式。



要點

從梯形圖的詳細表示切換為MELSAP-L(指令形式)的情況下，選擇選單的[顯示]⇨[梯形圖顯示切換]⇨[MELSAP-L(指令格式)]。(GX Works3 操作手冊)

在MELSAP-L(指令形式)中，使用與觸點相當的指令記述移轉條件。移轉條件的BOOL式為真(TRUE)的情況下，移轉條件成立。

MELSAP-L(指令形式)採用下述形式記述程式。

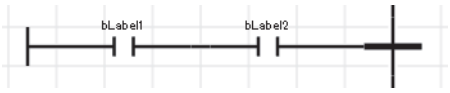
□：可使用的標籤/元件

項目	MELSAP-L(指令形式)	記述示例
常開觸點(LD指令)	a□	aX0
常閉觸點(LDI指令)	b□	bX1
上升沿常開觸點(LDP指令)	p□	pM2
下降沿常開觸點(LDF指令)	f□	fM3
反轉運算結果(INV指令)	&INV	aM0 & INV
運算結果脈衝化指令(上升沿)(MEP指令)	&MEP	aM1 & MEP
運算結果脈衝化指令(下降沿)(MEF指令)	&MEF	aM2 & MEF
變址繼電器運算結果脈衝化指令(上升沿)(EGP指令)	&EGP □	aM3 & EGP V0
變址繼電器運算結果脈衝化指令(下降沿)(EGF指令)	&EGF □	aM4 & EGF V1
與觸點相當的比較運算指令	與梯形圖語言中的指令輸入進行同樣記述。 可以使用下述比較運算指令。 BIN16位元資料比較(帶符號)：<，<=，<>，=，>，>= BIN32位元資料比較(帶符號)：D<，D<=，D<>，D=，D>，D>= 單精度實數比較：E<，E<=，E<>，E=，E>，E>= 雙精度實數比較：ED<，ED<=，ED<>，ED=，ED>，ED>= BIN16位元資料比較(無符號)：<_U，<=_U，<>_U，=_U，>_U，>=_U BIN32位元資料比較(無符號)：D<_U，D<=_U，D<>_U，D=_U，D>_U，D>=_U 字元串比較：$<，$<=，$<>，$=，$>，$>=	< D10 D20
並聯連接(OR)	|	aX0 | aM0
串聯連接(AND)	&	aX0 & aM0
括弧	()	(aX0 | aM0) & aX1

同時使用&與|的情況下，優先運算&。希望更改優先順序的情況下，使用( )。

移轉條件的直接表示

可以將激活移轉至下一個步的條件直接在SFC圖上創建。對移轉條件連接FBD/LD部件的觸點。



不可以使用線圈部件、FB部件、函數部件、跳轉部件/跳轉標籤部件、返回部件。

要點

透過選擇移轉條件選擇選單的[編輯]⇨[變更]⇨[移轉條件的直接表現]，可以將FBD/LD部件連接到移轉條件的左側中。(GX Works3 操作手冊)

移轉條件的標籤/元件

可以在使激活移轉至下一個步的條件中指定位元型標籤及位元元件或BOOL值。

位元型標籤的情況下

位元元件的情況下

BOOL值的情況下

要點

選擇移轉條件名、選擇選單的[編輯]⇨[變更]⇨[名稱]，輸入希望指定的位元型標籤及位元元件或BOOL值。(GX Works3 操作手冊)

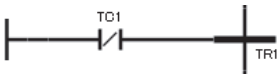
■注意事項

- 移轉條件中使用了定時器及計數器的元件(T、ST、LT、LST、C、LC)的情況下，將作為觸點(TS、STS、LTS、LSTS、CS、LCS)進行動作。使用了定時器及計數器的元件的線圈(TC、STC、LTC、LSTC、CC、LCC)的情況下，也同樣作為觸點進行動作。
- 希望在移轉條件中使用定時器及計數器的線圈的情況下，應使用定時器型及計數器型的標籤。

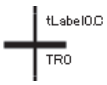
例

定時器元件及定時器型標籤的情況下

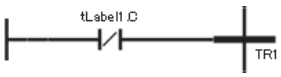
觸點(TS0)為ON時，移轉條件成立。



觸點(TS1)為OFF時，移轉條件成立。



定時器型標籤tLabel0的線圈為ON時，移轉條件成立。



定時器型標籤tLabel1的線圈為OFF時，移轉條件成立。

8

8.3 SFC控制指令

SFC控制指令是指進行塊或步的激活狀態的檢查以及強制啟動、結束等的指令。如果使用SFC控制指令，可以從順控程式以及SFC程式的動作輸出內對SFC程式進行控制。

指令一覽

SFC控制指令的一覽如下所示。

指令名稱	指令符號	處理內容
步激活檢查	LD、LDI、AND、ANI、OR、ORI［S□］*1	檢查指定步的激活/非激活。
	LD、LDI、AND、ANI、OR、ORI［BL□\S□］	
塊激活檢查	LD、LDI、AND、ANI、OR、ORI［BL□］	檢查指定塊的激活/非激活。
激活步批量讀取	MOV(P)［K4S□］*1	將指定塊的步激活狀態作為位元資訊以BIN16位元資料單位讀取至指定元件中。
	MOV(P)［BL□\K4S□］	
	DMOV(P)［K8S□］*1	將指定塊的步激活狀態作為位元資訊以BIN32位元資料單位讀取至指定元件中。
	DMOV(P)［BL□\K8S□］	
	BMOV(P)［K4S□］*1	按照指定字從指定步批量讀取指定塊的步激活狀態。
	BMOV(P)［BL□\K4S□］	
塊啟動	SET［BL□］	對指定塊進行單獨激活，並從初始步開始執行。
塊結束	RST［BL□］	將指定塊單獨置為非激活狀態。
塊停止	PAUSE［BL□］	將指定塊置為暫時停止狀態。
塊重啟	RSTART［BL□］	對指定塊的暫時停止進行解除，並從停止步開始重啟執行。
步啟動	SET［S□］*1	激活指定步。
	SET［BL□\S□］	
步結束	RST［S□］*1	將指定步置為非激活。
	RST［BL□\S□］	
塊切換	BRSET	指定SFC控制指令的對象塊。

*1 在順控程式內使用時，塊0為對象。在SFC程式內使用時，本塊為對象。

SFC控制指令的詳細內容，請參閱下述手冊。

MELSEC iQ-R 程式手冊(CPU模組用指令/通用FUN/通用FB篇)

■注意事項

• 在中斷程式內請勿使用SFC控制指令。

• SFC控制指令應只有在SM321(SFC程式啟動/停止)為ON時才執行。

變址修飾

透過SFC控制指令指定的步繼電器及SFC塊元件可以進行變址修飾並指定。

元件	變址修飾對象位置
S□Z□	步繼電器
BL□\S□Z□	帶塊指定步繼電器的步部分
BL□Z□\S□	帶塊指定步繼電器的塊部分
BL□Z□\S□Z□	帶塊指定步繼電器的塊部分及步部分
BL□Z□	SFC塊元件

對於步繼電器及SFC塊元件，也包括進行變址修飾的情況，在下述範圍內進行指定。

元件		範圍
S□		R00CPU、R01CPU、R02CPU：0～8191 上述以外的CPU模組：0～16383 (最大值將根據CPU參數的設置有所不同)
BL□\S□	BL□	R00CPU、R01CPU、R02CPU：0～127 上述以外的CPU模組：0～319
	S□	R00CPU、R01CPU、R02CPU：0～127 上述以外的CPU模組：0～511
BL□		R00CPU、R01CPU、R02CPU：0～127 上述以外的CPU模組：0～319

要點

關於變址修飾的詳細內容，請參閱下述手冊。

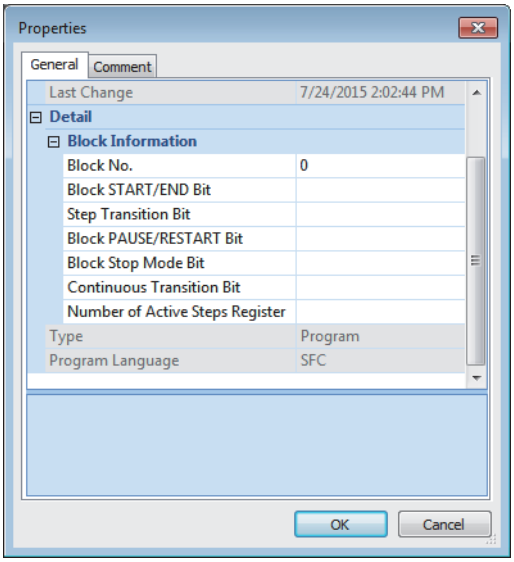
MELSEC iQ-R CPU模組用戶手冊(應用篇)

8.4 SFC用資訊元件

SFC用資訊元件是對塊指示強制啟動/結束與暫時停止/重啟的指示，以及步的移轉條件的成立與激活步數的確認，或指示移轉條件的連續移轉動作的元件或標籤。
SFC用資訊元件在各塊進行設置。

[導航視窗]⇨[程式]⇨SFC程式檔案⇨希望設置的塊屬性

畫面顯示



顯示內容

項目	內容	可使用的資料	
		元件	資料類型(標籤)
塊啟動結束位元	設置對塊的激活狀態進行確認的元件或標籤。 另外可以在設置的位元ON時啟動塊、OFF時結束塊。	位元：Y、M、L、F、V、B 字：D、W、RD的位元指定	BOOL、BOOL的陣列、INT的位元指定、WORD的位元指定
步序移轉位元	設置對執行中的步的移轉條件的成立進行確認的元件或標籤。 各步的動作輸出執行後，至下一個步的移轉條件成立時將ON。		
塊停止重啟位元	設置使激活中的塊暫時停止或重啟的元件或標籤。 設置的位元ON時塊在執行中步進行停止，並在OFF時從停止了塊的步重啟執行。		
塊停止模式位元	對決定使塊停止的時機的元件或標籤進行設置。 設置的位元ON時在各步的移轉後使其停止，並在OFF時使所有步立即停止。		
連續移轉位元	設置移轉條件成立時決定連續移轉動作的元件或標籤。 設置的位元ON時將變為有連續移轉，在同一掃描內執行下一個步的動作輸出。OFF時將變為無連續移轉，且在1個掃描中逐步執行。		
激活步數寄存器	設置對塊的當前激活中的步數進行儲存的元件或標籤。	D、W、R、ZR、RD	INT、WORD

對於SFC用資訊元件，除全局元件及局部元件以外，也可以指定全局標籤或局部標籤。不可以間接指定、位指定、變址修飾(Z、LZ)。

要點

SFC用資訊元件的設置僅在使用SFC用資訊元件的情況下需要。不使用的情況下，無需設置SFC用資訊元件。

塊啟動結束位元

塊啟動結束位元是對塊的激活狀態進行確認的元件或標籤。

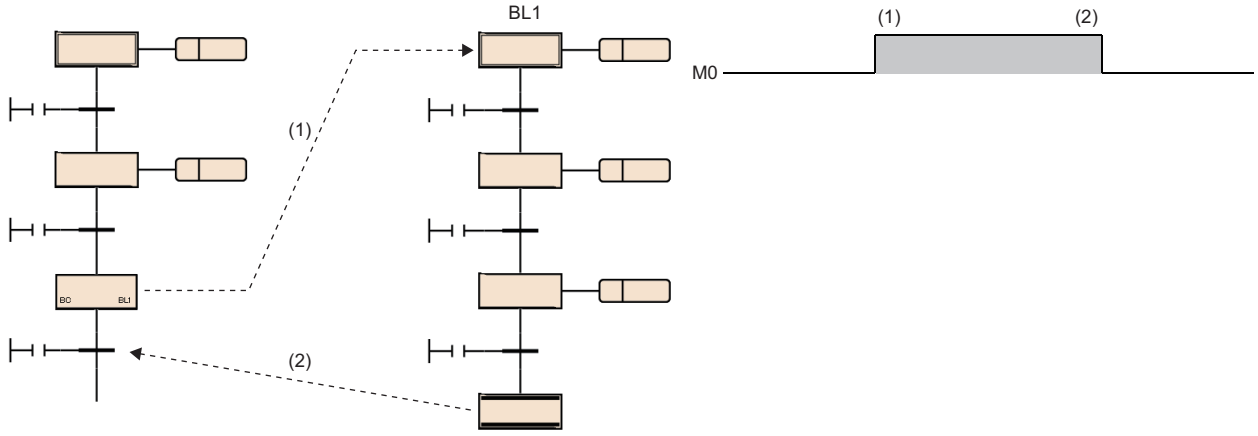
另外可以在設置的位元ON時啟動塊、OFF時結束塊。

無塊啟動程式的情況下，透過工程工具也可對塊的啟動/結束進行控制，因此在塊單位中的偵錯及試運行中可以使用。

- 設置的塊啟動時，塊啟動結束位元將自動ON。設置的塊處於激活中時，塊啟動結束位元將保持為ON狀態不變。
- 設置的塊變為非激活狀態時，塊啟動結束位元將自動OFF。設置的塊為非激活狀態時，塊啟動結束位元將保持為OFF狀態不變。

例

在塊1(BL1)的塊啟動結束位元中指定了M0的情況下



(1) 啟動塊1(BL1)，M0變為ON。

(2) 塊1(BL1)變為非激活狀態，M1變為OFF。

- 在設置的塊為非激活時，如果將塊啟動結束位元置為ON，則單獨啟動設置的塊。
- 在設置的塊為激活中時，如果將塊啟動結束位元置為OFF，則結束設置的塊。

塊啟動結束位元的ON/OFF也可透過工程工具的測試操作進行。(GX Works3 操作手冊)

將塊啟動結束位元置為OFF，並將設置的塊置為非激活狀態的情況下，進行下述處理。

- 停止設置的塊的執行，執行的步的輸出也全部OFF。但是，透過SET指令變為ON的元件將不OFF。
- 在設置的塊內透過塊啟動步啟動了其他塊的情況下，設置的塊將結束，但是啟動目標的塊保持激活狀態不變，繼續運行處理。

要點

透過從工程工具的檢視對BL□或BL□\S□的當前值進行更改，也可以對塊進行啟動/結束或對步進行激活/非激活。

透過選單的[偵錯]⇨[SFC步序控制]，也可以對選擇的步進行激活/非激活。(GX Works3 操作手冊)

■注意事項

• 將設置的塊置為非激活之後的重啟動作如下所示。

設置的塊		內容
塊0	在CPU參數的SFC設置中，啟動條件設置為“自動啟動塊0”的情況下	結束步處理後，從初始步開始重啟。
	在CPU參數的SFC設置中，啟動條件設置為“不自動啟動塊0”的情況下	結束步處理後，將設置的塊置為非激活狀態，如果再次對設置的塊發出了啟動請求則從初始步開始重啟。
塊0以外		

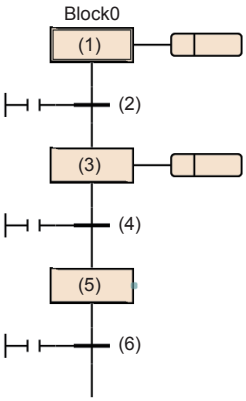
• 在SFC程式結束時，SFC用資訊元件中設置的所有塊啟動結束位元將OFF。但是，在設置繼續運行啟動時僅允許繼續運行啟動的情況下SFC程式啟動時，所有塊啟動結束位元將恢復。

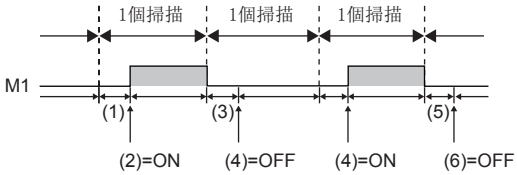
步序移轉位元

是對執行中的步的移轉條件的成立進行確認的元件或標籤。
各步的動作輸出執行後，至下一個步的移轉條件成立時將ON。
變為ON的步移轉位元再次執行指定的塊的處理時，將自動OFF。

例

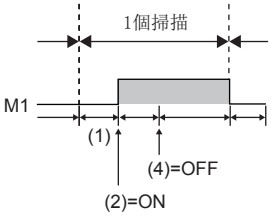
在Block0的步移轉位元中指定了M1的情況下





在步(1)的執行後移轉條件(2)成立時，執行其他塊期間M1將變為ON。
下一個掃描的Block0的處理時M1將OFF。
在步(3)的執行後，移轉條件(4)不成立的情況下，M1將保持為OFF狀態不變。
移轉條件(4)成立時，執行其他塊期間M1將變為ON。
在步(5)的執行後，移轉條件(6)不成立的情況下，M1將保持為OFF狀態不變。

將連續移轉位元置為ON設置了“有連續移轉”的情況下，移轉條件成立後下一個步的動作輸出執行中及執行多個步後移轉條件未成立時，步移轉位元也將保持為ON狀態不變，在執行下一個掃描的指定塊時將變為OFF。



在步(1)的執行後移轉條件(2)成立時，M1將變為ON。
移轉條件(4)不成立的情況下，M1也將保持為ON狀態不變。
下一個掃描的Block0的處理時M1將OFF。

在塊內存在多個激活步的情況下，只要其中一個移轉條件成立，在該時點步移轉位元將ON。

■注意事項

- 如果執行結束步，塊的步移轉位元將ON。然後步移轉位元將保持為ON狀態不變直至該塊再次激活為止。
- 在SFC程式啟動時及SFC程式結束時，步移轉位元將不OFF。

塊停止重啟位元

使激活中的塊暫時停止或重啟的元件或標籤。

設置的位元ON時塊在執行中步進行停止，並在OFF時從停止了塊的步重啟執行。

設置	內容
OFF→ON	如果進行OFF→ON，指定塊在執行中的步中停止。
ON→OFF	如果進行ON→OFF，指定塊將從停止的步的動作輸出開始重啟執行。 • 在動作保持狀態下變為停止的動作保持步(無移轉檢查)[SE]或動作保持步(有移轉檢查)[ST]，在動作保持狀態下重啟執行。 • 對於線圈保持步[SC]，透過線圈輸出OFF的設置(SM325 = OFF)停止的情況下，將變為非激活狀態，因此無法重啟保持狀態。透過線圈輸出保持的設置(SM325 = ON)停止的情況下，將維持保持狀態，因此重啟後也將保持狀態不變。

- 透過塊啟動步啟動了其他塊的情況下，如果將塊停止重啟位元置為ON則指定的塊將停止，但是啟動目標的其他塊將保持為激活狀態不變，繼續運行處理。也要同時停止啟動目標的塊的情況下，啟動目標的塊停止重啟位元也將ON。
- 如果將非激活的塊中設置的塊停止重啟位元置為ON，則在非激活狀態中不進行動作，在塊變為了激活狀態時立即變為停止狀態。
- 如果強制結束了指定塊，則塊停止重啟位元的狀態也將被保持為不變。在停止中強制結束且不對塊停止重啟位元的狀態進行更改的情況下，再次啟動時將立即變為停止狀態。

塊停止/重啟時的動作，根據SM325(塊停止時的輸出模式設置)與SFC用資訊元件的塊停止模式位元的設置、步的保持/非保持的組合來決定。(☞ 122頁 塊停止重啟時的動作)

■注意事項

- 在SFC程式啟動時及SFC程式結束時，塊停止重啟位元將不OFF。

塊停止模式位元

決定使塊停止的時機的元件或標籤。

設置的位元ON時在各步的移轉後使其停止，並在OFF時使所有步立即停止。

設置	內容
OFF時(立即停止)	發出停止請求時，立即變為停止狀態。
ON時(移轉後停止)	發出停止請求後，執行中的步的移轉條件成立，移轉時將停止。 無法執行移轉後的步的動作輸出。 在塊內存在多個激活步時，將從移轉成立的步開始依次停止。 與塊停止模式位元的設置無關，保持中的步在停止請求後將立即停止。

塊停止/重啟時的動作，根據SM325(塊停止時的輸出模式設置)與塊停止模式位元的設置、步的保持/非保持的組合來決定。(☞ 122頁 塊停止重啟時的動作)

■注意事項

- 在SFC程式啟動時及SFC程式結束時，塊停止模式位元將不OFF。

連續移轉位元

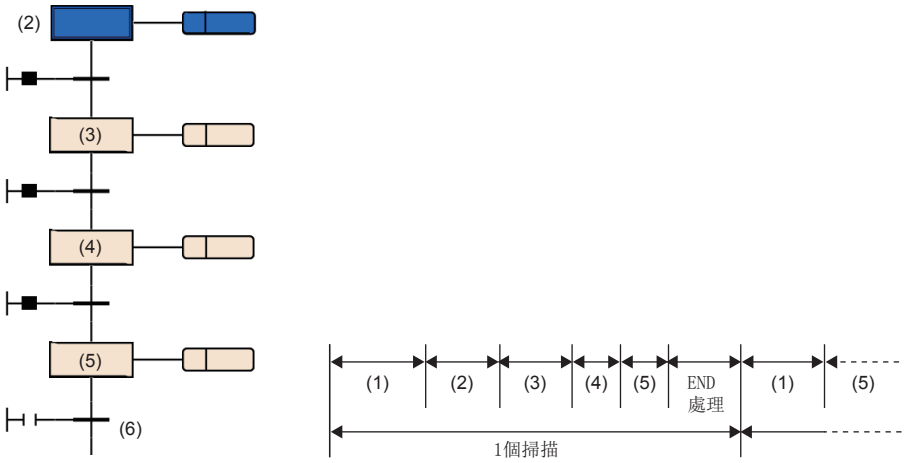
是當移轉條件成立時，決定連續移轉的動作的元件或標籤。

設置的位元ON時將變為有連續移轉，在同一掃描內執行下一個步的動作輸出。OFF時將變為無連續移轉，且在1個掃描中逐步執行。

設置	內容
OFF時(無連續移轉)	移轉條件成立時，將在下一個掃描內執行移轉目標步的動作輸出。
ON時(有連續移轉)	移轉條件成立時，將在同一掃描內執行移轉目標步的動作輸出。 步的移轉條件連續成立的情況下，在移轉條件不成立之前或達到結束步之前，將在同一掃描內執行。

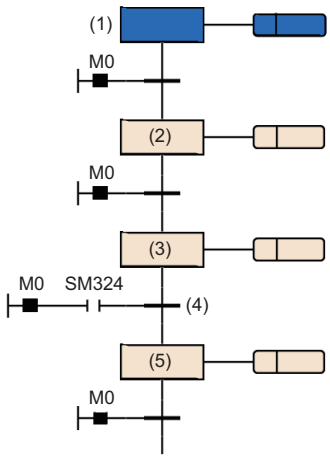
例

有指定SFC用資訊元件的連續移轉位元的情況下



掃描	內容
第1個掃描	順控程式(1)的執行後，連續執行SFC程式的步(2)～步(5)。
第2個掃描移轉	順控程式(1)的執行後，在移轉條件(6)成立之前執行步(5)的動作輸出。

- 對連續移轉位元進行了設置的情況下，與SM323(所有塊連續移轉的有無)的ON/OFF無關，設置的位元元件為OFF時變為無連續移轉的動作、ON時變為有連續移轉的動作。未對連續移轉位元進行設置的情況下，SM323為OFF時變為無連續移轉的動作、ON時變為有連續移轉的動作。(☞ 129頁 有/無連續移轉的動作)
- SM324(連續移轉阻止標誌)在執行SFC程式時系統將自動ON，但是連續移轉中時將變為OFF。透過在移轉條件中將SM324以AND條件使用，可以禁止連續移轉。

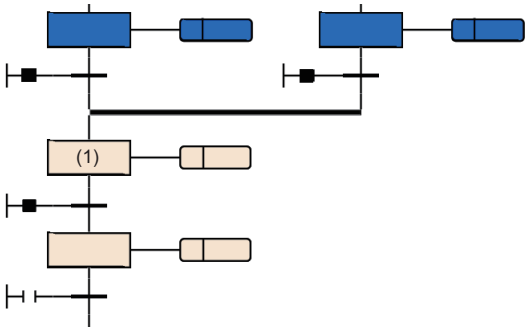


M0為ON時，1個掃描中從步(1)到步(3)變為連續移轉。
透過將SM324作為AND條件附加到移轉條件(4)中，步(3)的執行後的移轉條件(4)將變為不成立。
在下一個掃描中，執行步(3)後SM324變為ON，因此在該掃描內移轉到步(5)中。

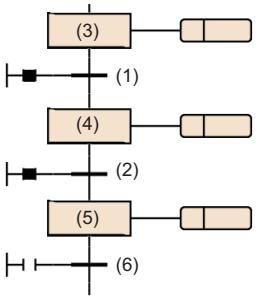
■注意事項

- 設置為有連續移轉時，由於從移轉條件成立開始，移轉目標步的動作輸出執行比其他處理優先執行，因此可以縮短節拍時間。但是，在該情況下，其他塊及順控程式的動作有可能會變慢。
- 在SFC程式啟動時及SFC程式結束時，連續移轉位元將不OFF。
- 透過跳轉移轉及選擇合併，在激活從多個步移轉至1個步的情況下，1個步的動作輸出有可能在1個掃描中執行2次。

有連續移轉的情況下，將步(1)在1個掃描中執行2次。



- 有連續移轉的設置中，步後的移轉條件成立的情況下，在1個掃描內進行步的啟動及結束。在這種情況下，由於不執行END處理，因此透過動作輸出內的OUT指令進行的線圈輸出的輸入輸出更新將不被反映，無法在其他程式中檢測到線圈的ON。例如，輸出(Y)的情況下，在未執行END處理時輸出(Y)將不被輸出，且無法自其他程式中檢測到輸出(Y)的ON。因此，也無法檢測到步繼電器的ON。為了反映OUT指令的輸入輸出更新，應創建將1個步在多個掃描中執行的程式。



移轉條件(1)與移轉條件(2)成立的情況下，在1個掃描內執行下述操作。
- 執行步(3)的動作輸出。
- 由於移轉條件(1)成立，因此將步(3)的動作輸出置為OFF。
- 步(3)變為非激活、步(4)變為激活。
- 由於有連續移轉，因此執行步(4)的動作輸出。
- 由於移轉條件(2)成立，因此將步(4)的動作輸出置為OFF。
- 步(4)變為非激活、步(5)變為激活。
- 由於有連續移轉，因此執行步(5)的動作輸出。
- 由於移轉條件(6)不成立，因此步(5)的動作輸出將不OFF。

- 在使用跳轉移轉進行循環的程式時，應置為無連續移轉，或將執行中環路內的移轉條件置為全部不成立。有連續移轉且在執行中環路內的移轉條件全部成立時，則將在1個掃描內變為無限循環。

激活步數寄存器

對塊的當前激活中的步數進行儲存的元件或標籤。

激活步數寄存器中儲存的激活步數包括下述步。

- 普通的激活步
- 保持中的線圈保持步[SC]
- 保持中的動作保持步(有移轉檢查)[ST]
- 保持中的動作保持步(無移轉檢查)[SE]
- 停止中的步

■注意事項

- 在塊結束時，激活步數寄存器將變為0。
- 在SFC程式結束時激活步數寄存器不變為0、在SFC程式啟動時變為0。

8.5 SFC設置

在CPU參數及SFC塊設置內，對SFC程式的啟動條件等進行設置。

CPU參數

SFC設置一覽如下所示。

類型	項目	內容
SFC設置	SFC程式啟動模式設置	設置在SFC程式啟動時是在初始狀態下進行啟動(初始啟動)，或是在保持之前的執行狀態不變的情況下進行啟動(繼續運行啟動)。
	啟動條件設置	設置在SFC程式啟動時是讓塊0自動啟動後再激活，或是保持非激活狀態直到有啟動請求為止。
	塊停止時的輸出模式設置	在塊停止時，可將線圈輸出置為OFF，或是保持線圈輸出。

要點

使用SFC程式的情況下，應預先確保步繼電器(S)的點數。(步繼電器(S)的預設點數為0點。)
在[CPU參數]⇨[記憶體/元件設定]⇨[元件/標籤記憶體區域進階設定]⇨[元件設定]中，以1024點為單位設置步繼電器(S)的點數。(☞ 79頁 規格)

SFC程式啟動模式設置

設置在SFC程式啟動時是在初始狀態下進行啟動(初始啟動)，或是在保持之前的執行狀態不變的情況下進行啟動(繼續運行啟動)。

[CPU參數]⇨[SFC設定]⇨[SFC程式啟動模式設定]

畫面顯示

顯示內容

設置	內容
初始啟動 (預設)	對上次停止時的激活狀態進行清除並啟動。 啟動後的動作按照SFC設置的啟動條件設置進行。(☞ 121頁 啟動條件設置)
繼續運行啟動	保持上次停止時的激活狀態不變的狀況下進行啟動。

根據SFC程式啟動模式設置及SM322(SFC程式的啟動狀態)的狀態組合，決定是進行初始啟動還是進行繼續運行啟動。

動作		SFC程式啟動模式設置： 初始啟動		SFC程式啟動模式設置： 繼續運行啟動	
		SM322： OFF (初始狀態)*1	SM322： ON (設置更改時)	SM322： ON (初始狀態)*1	SM322： OFF (設置更改時)
(1)	將SM321置為OFF→ON	初始啟動		繼續運行啟動	初始啟動
(2)	將電源置為OFF→ON			繼續運行啟動/初始啟動*4	
(3)	將SM321置為ON→OFF或RUN→STOP後將電源置為OFF→ON			繼續運行啟動	
(4)	復位→RUN			繼續運行啟動/初始啟動*4	
(5)	將SM321置為ON→OFF或RUN→STOP後進行復位→RUN			繼續運行啟動	
(6)	STOP→RUN	繼續運行啟動*3			
(7)	STOP→程式寫入→RUN	初始啟動*2			

*1 對於SM322，根據SFC程式啟動模式的設置，在STOP→RUN時決定初始狀態。
*2 將SFC程式啟動模式設置設置為繼續運行啟動，且在程式的寫入前後無更改的情況下，將繼續運行啟動。
*3 動作輸出的ON/OFF，按照參數設置的“STOP→RUN時的輸出模式”的設置進行。
*4 根據時機將變為禁止繼續運行啟動狀態，且有可能進行初始啟動。

■注意事項

- 繼續運行啟動時，SFC程式的停止位置將保持，但是動作輸出中使用的標籤及元件的狀態將不保持。因此，在進行完繼續運行啟動後，將需要預先保持的標籤及元件置為鎖存設置。
- 線圈保持步[SC]的線圈輸出為OFF的條件(表中(1)、(3)、(5))以外的繼續運行啟動時，將重啟保持中的線圈保持步[SC]，但是輸出將不變為ON。希望繼續輸出的情況下，應將希望保持繼續的標籤及元件置為鎖存設置。此外，STOP→RUN時的輸出的ON/OFF動作按照CPU參數設置的“STOP→RUN時的輸出模式設定”的設置進行。(MELSEC iQ-R CPU模組用戶手冊(應用篇))
- 電源OFF時或復位時，智能功能模組將被初始化。繼續運行啟動的情況下，建議將至智能功能模組的初始化程式創建到常時激活狀態的塊或順控程式上。
- 電源OFF時或復位時，標籤及元件也將被清除。設置SFC用資訊元件時，僅在進行了鎖存設置的情況下保持值。
- 電源OFF後或復位後的繼續運行啟動，根據時機有可能無法繼續運行啟動。在繼續運行啟動的設置時進行了初始啟動的情況下，事件履歷中禁止繼續運行啟動的事件將被儲存。確實希望進行繼續運行啟動的情況下，應將SM321置為ON→OFF或在RUN→STOP後將電源置為OFF或進行復位。

啟動條件設置

設置在SFC程式啟動時是讓塊0自動啟動後再激活，或是保持非激活狀態直到有啟動請求為止。

[CPU參數]⇨[SFC設定]⇨[啟動條件設定]

畫面顯示

顯示內容

設置	內容	
	SFC程式啟動時	塊0結束時
自動啟動塊0 (預設)	塊0將被自動啟動，並從初始步開始執行。	塊0將被自動再啟動，並再次從初始步開始執行。
不自動啟動塊0	塊0也與其他塊一樣透過SFC控制指令的SET指令(塊啟動)及塊啟動步在有啟動請求時變為激活狀態。	塊0將無法自動進行再啟動，會一直保持非激活狀態直至再次有啟動請求為止。

對於啟動條件設置，在希望根據產品類型等對SFC程式啟動時的啟動塊進行指定時使用。

“自動啟動塊0”在按以下方式使用塊0的情況下有效。

- 管理塊
- 前處理塊
- 常時監視塊

■注意事項

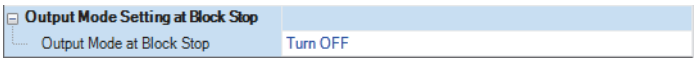
- 設置為“不自動啟動塊0”的情況下，執行SFC程式時透過順控程式執行SET指令(塊啟動)，或將SFC用資訊元件中設置的塊啟動結束位元置為ON。
- 設置為“自動啟動塊0”的情況下，必須創建塊0。

塊停止時的輸出模式設置

在塊停止時，可將線圈輸出置為OFF，或是保持線圈輸出。

[CPU參數]⇨[SFC設定]⇨[塊停止時的輸出模式設定]

畫面顯示



顯示內容

設置	內容
變為OFF (預設)	將線圈輸出置為OFF。
保持ON	將線圈輸出保持為停止之前的狀態。

- 在電源ON時、復位時或STOP→RUN時，已設置的內容將被反映到SM325(塊停止時的輸出模式)的初始值中，並在SFC程式動作時按照SM325的設置執行。CPU參數的設置將被忽略。

■塊停止重啟時的動作

塊停止/重啟時的動作，根據SM325(塊停止時的輸出模式設置)與SFC用資訊元件的塊停止模式位元的設置、步的保持/非保持的組合來決定。

塊停止/重啟時的動作一覽如下所示。

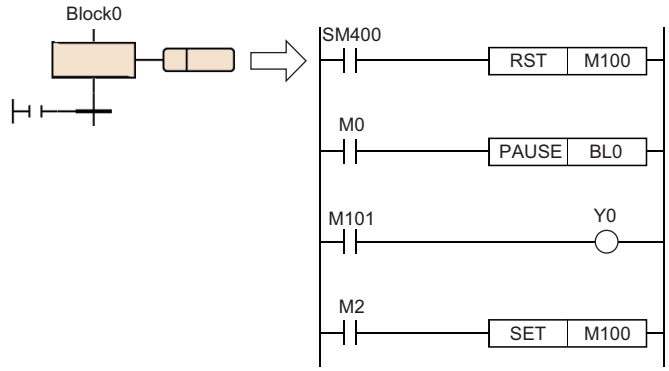
塊停止時的輸出模式的設置	塊停止模式位元的設置	動作			
		保持中以外的激活中步(也包括移轉條件不成立的SC、SE、ST)	保持中步		
			線圈保持步[SC]	動作保持步(無移轉檢查)[SE]	動作保持步(有移轉檢查)[ST]
SM325＝OFF(線圈輸出OFF)	OFF或無設置(立即停止)	有停止請求之後，將動作輸出的線圈輸出置為OFF後進行停止。狀態保持為激活不變。	有停止請求之後，將動作輸出的線圈輸出置為OFF後變為非激活狀態。	有停止請求之後將動作輸出的線圈輸出置為OFF後進行停止。狀態保持為激活不變。	
	ON(移轉後停止)	移轉成立後，進行步的結束處理，同時移轉目標步將變為激活狀態，在執行動作輸出之前將停止。			
SM325＝ON(線圈輸出保持)	OFF或無設置(立即停止)	有停止請求之後，在保持動作輸出的線圈輸出的狀態下進行停止。狀態保持為激活不變。	有停止請求之後，在保持動作輸出的線圈輸出的狀態下進行停止。狀態保持為激活不變。		
	ON(移轉後停止)	在移轉成立之前與普通的動作相同。移轉成立後，進行步的結束處理，同時移轉目標步將變為激活狀態，在執行動作輸出之前將停止。			
重啟時		返回普通的動作。	線圈輸出OFF時：變為非激活狀態，因此無法重啟。 線圈輸出保持時：保持中的狀態下進行重啟。	在保持狀態下重啟動作輸出的執行。	在保持狀態下，重啟動作輸出後，檢查移轉條件。

■注意事項

- 使用LD指令(塊激活檢查)等指定的塊為停止中的塊的情況下，變為ON。此外，使用LD指令(步激活檢查)等指定的步即使為停止中的步，也變為ON。
- 在將SFC用資訊元件的停止重啟位元為ON的狀態下進行塊啟動時，在初始步變為激活狀態之前將停止。此外，對於非激活塊執行了SET指令(步啟動)的情況下，指定步變為激活狀態之前將停止。
- SM325(塊停止時的輸出模式)為ON時(線圈輸出保持)，可以在保持線圈輸出不變的狀態下進行停止。在停止中即使將SM325置為ON→OFF線圈輸出的狀態也不變化，發生塊的重啟請求時，在保持狀態下進行重啟。
- 在SM325為ON時停止了塊的情況下，保持狀態的線圈保持步[SC]在重啟後也維持保持狀態，但是步的動作不重啟。將線圈保持步[SC]置為非激活時，應執行RST指令(步結束)。
- 在動作輸出內即使對該塊有停止請求，也必須等到當前執行中的步執行至最後為止後才能開始執行停止請求。因此，在執行中步內，即使在塊停止模式位元為OFF時(立即停止)執行了停止請求也不停止。此外，之後在相同步內，塊停止模式位元為ON時(移轉後停止)進行了切換的情況下，將在移轉後於停止模式中執行停止請求。

例

M100為停止時模式位元，且M101為塊停止重啟位元的情況下



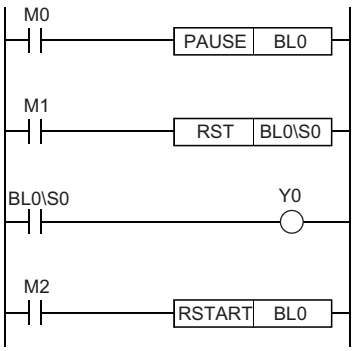
執行上述動作輸出時，如果將M0置為ON則執行PAUSE指令，Block0的塊停止重啟位元(M101)將變為ON，但是由於將會執行到動作輸出的最後為止，因此Y0將變為ON。

此外，M2為ON時，即使執行了PAUSE指令後，在停止時模式位元變為ON且執行了全部動作輸出後，也將在移轉後於停止模式中執行停止請求。

- 在塊停止中執行RST指令(步結束)時，指定的步繼電器將OFF。但是，工程工具的監視畫面將保持為激活狀態不變，重啟了塊時將變為非激活狀態。即使在SM325=ON(塊停止時的線圈輸出保持)在停止中執行也一樣，但是線圈輸出不變為OFF。
- 對於SET指令(步啟動)，即使在塊停止中也立即被執行且指定的步繼電器將變為ON，工程工具的監視畫面中也將變為激活狀態。但是，動作輸出將不執行直至塊被重啟為止。

例

使用RST指令(步結束)時的塊的停止重啟的情況下



(1) 如果將M0置為ON，塊0將停止。
(2) 如果將M1置為ON，步No.0中將執行結束請求，步繼電器的BL0\S0將變為OFF，但是在工程工具的監視上步No.0將保持激活中不變。
(3) BL0\S0變為OFF，因此Y0也變為OFF。
(4) 在M0、M1處於OFF的狀態下將M2置為ON時，重啟塊0之後結束步No.0。

- 停止時模式位元為ON(移轉後停止模式)時，即使在存在移轉後停止等待狀態的步的狀態下將停止時模式位元為OFF，也將保持移轉後停止等待狀態不變。從此狀態對移轉後停止等待狀態進行解除後立即停止時，需要重啟塊，在停止模式位元為OFF的狀態下再次執行停止請求。
- 在移轉後停止模式中步的移轉目標為結束步的情況下，將執行結束步的處理，因此不變為停止狀態。
- 對有停止請求進行確認時，將在工程工具的塊一覽表顯示中進行監視，或對塊停止重啟位元中設置的位元進行監視。但是，無法透過工程工具的監視確認步是否處於停止中或是否以停止等待進行動作中。
- 在移轉成立之前，透過將塊停止重啟位元置為OFF或執行RSTART指令，可以對移轉後停止狀態進行解除。在已停止的步與停止等待的步同時存在的狀態下開始了重啟請求時，已停止的步將開始動作，停止等待步將繼續進行動作。停止請求被解除。

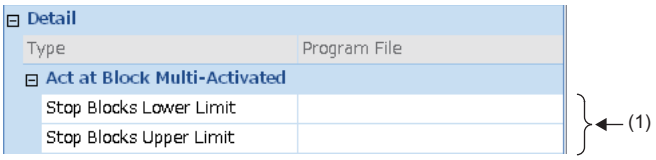
SFC塊設置

塊雙重啟動時的運行設置

對於已激活的塊透過塊啟動步(有結束檢查)[BC]或塊啟動步(無結束檢查)[BS]發出了啟動請求時，在希望停止CPU模組的運算的情況下進行設置。在設置範圍設置希望停止的塊範圍。

[導航視窗]⇨[程式]⇨希望設置的SFC程式檔案的屬性

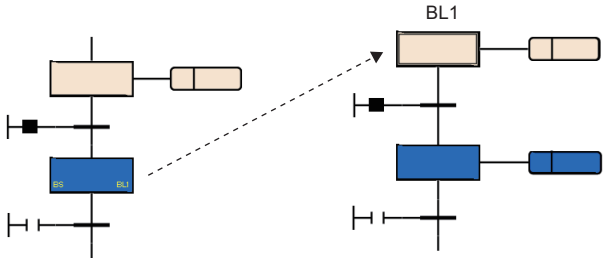
畫面顯示



(1) 對希望停止的塊範圍進行設置。

顯示內容

設置	內容	
無設定 (預設)	待機	繼續運行CPU模組的運算，保持移轉條件成立的狀態不變，進行待機直至啟動目標的塊變為非激活狀態。 啟動目標的塊變為非激活狀態時，將塊再次置為激活狀態。 變為移轉等待狀態時，之前的步將非激活且輸出變為OFF，將不執行動作輸出的運算。
有停止的塊範圍的設定	停止	變為出錯狀態。



■注意事項

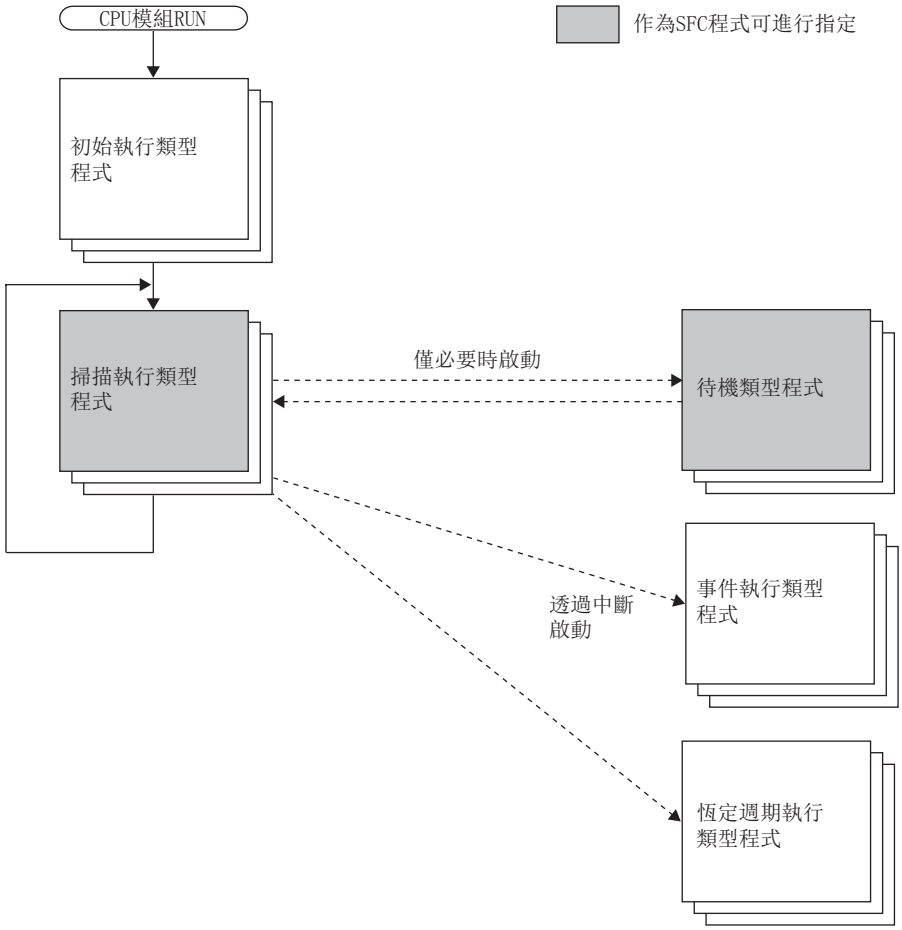
- 對於已處於激活中的塊，執行了SFC控制指令的SET指令(塊啟動)時，將忽略啟動請求直接繼續運行SFC程式的處理。
- 試圖移轉至激活中的塊啟動步的情況下，塊啟動步的啟動將被忽略。不會再次從初始步開始執行。

8.6 SFC程式的執行順序

整個程式的處理

可指定的執行類型

對於SFC程式，程式的執行類型的指定可否如下所示。



執行類型	指定可否	備注
初始執行類型程式	×	—
掃描執行類型程式	○	SFC程式時僅1個可以執行
待機類型程式	○	透過PSCAN(P)指令來指定SFC程式，可以更改掃描執行類型
事件執行類型程式	×	—
恆定週期執行類型程式	×	—

■注意事項

不存在掃描執行類型的SFC程式(僅待機類型程式)的情況下，請勿對SFC程式執行SFC控制指令及監視。

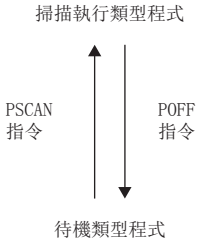
透過指令進行執行類型的更改

透過使用程式控制用指令，可以更改程式的執行類型。

對於程式控制用指令，SFC程式的指定可否如下所示。

指令符號	指定可否	備注
PSCAN(P)	○*1	將指定的SFC程式的執行類型更改為掃描執行類型。 在已存在有掃描執行類型的SFC程式時，指定了其他SFC程式後執行的情況下，將變為出錯狀態。
PSTOP(P)	×	對於SFC程式執行的情況下，將變為出錯狀態。
POFF(P)	○*1	指定的SFC程式在下一個掃描中執行所有塊的結束處理，在其後的下一個掃描中將執行類型更改為待機類型。

*1　在過程CPU(二重化模式)中無法指定。



■注意事項

- 自CPU模組進行檔案讀取/檔案寫入或使用資料記錄功能時，請勿執行PSCAN(P)指令。如果執行PSCAN(P)指令，掃描時間有可能延長數100ms。
- 在指定繼續運行啟動時，如果透過PSCAN(P)指令使與上次動作的SFC程式不同的SFC程式執行了動作，則指定的SFC程式將初始啟動。在這種情況下，事件履歷中“不能進行SFC程式的繼續啟動”(事件代碼： 0430)將被儲存。

SFC程式的處理順序

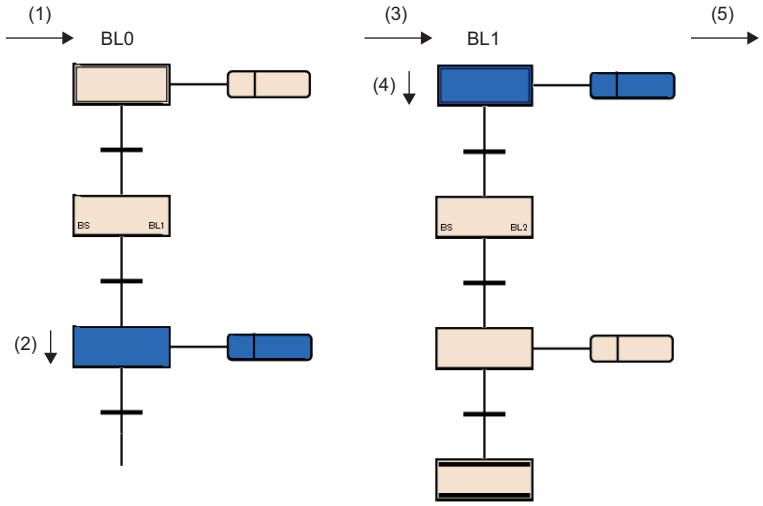
各塊的執行順序

在SFC程式的啟動中，從激活塊的初始步開始按順序執行各步的動作輸出。

存在多個塊的SFC程式的情況下，將按照塊0→塊1→塊2的順序從小編號的塊開始向大編號的塊按順序進行激活檢查。

激活中的塊將執行塊內的激活步的動作輸出。

非激活塊將檢查啟動請求的有無。如果有啟動請求將會把塊激活後，執行塊內的激活步。



按下述順序執行。

(1) 開始塊0(BL0)的處理。

(2) 執行塊0(BL0)的步。

(3) 執行塊1(BL1)的處理。

(4) 執行塊1(BL1)的初始步。

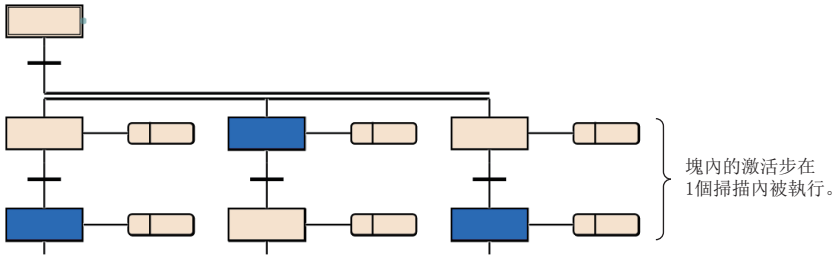
(5) 執行下一個塊的處理。

SFC設置的啟動條件設置中指定了塊0的自動啟動的情況下，只可以使塊0自動啟動。在此設置的情況下，即使到達結束步變為非激活狀態，塊0也將在下一個掃描中再次被啟動。(☞ 121頁 啟動條件設置)

此外，對於塊的結束、停止、重啟的請求，將在塊內的執行處理之前被處理。

各步的執行順序

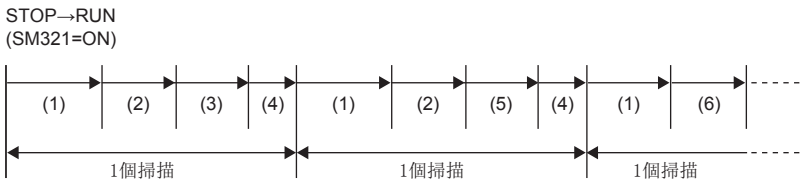
在SFC程式中，將所有激活步的動作輸出在1個掃描內進行處理。



各步的動作輸出結束時，檢查至下一個步的移轉條件的成立狀態。

- 移轉條件不成立時： 執行下一個掃描時，再次執行同一步的動作輸出。
- 移轉條件成立時： 將透過執行的動作輸出的OUT指令進行的輸出全部置為OFF。在執行下一個掃描時，執行下一個步的動作輸出。上次執行的步將變為非激活狀態，動作輸出將變為非執行狀態。

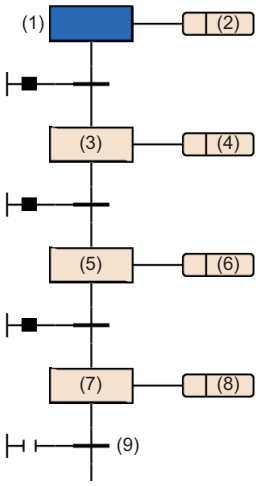
即使移轉條件成立，將步的屬性設置為線圈保持步[SC]時，將不變為非激活狀態而按照屬性進行處理。(☞86頁 線圈保持步[SC])



(1) 順控程式的執行
(2) 動作輸出的執行
(3) 至下一個步的移轉條件檢查(條件不成立)
(4) END處理
(5) 至下一個步的移轉條件檢查(條件成立)
(6) 執行下一個動作輸出

例

無SFC用資訊元件的連續移轉位元的指定的情況下

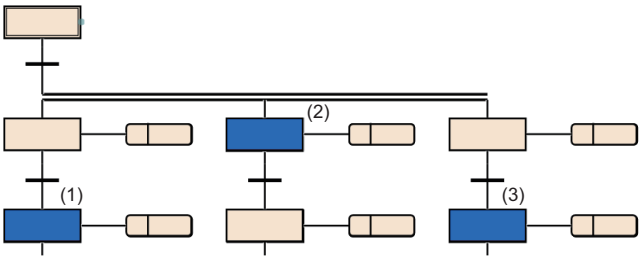


掃描	內容
第1個掃描	激活步(1)，執行動作輸出(2)。
第2個掃描	激活步(3)，執行動作輸出(4)。
第3個掃描	激活步(5)，執行動作輸出(6)。
第4個掃描	激活步(7)，執行動作輸出(8)。
第5個掃描及其以後	在移轉條件(9)成立之前激活步(7)，執行動作輸出(8)。

■注意事項

- 在首次執行時移轉條件已成立的步的情況下，由於在1個掃描中結束步，因此線圈輸出等的I/O更新將無法被反映，無法在其他程式中檢測到線圈輸出的ON。為了反映I/O更新，應創建將1個步在多個掃描中執行的程式。
- 塊內的激活步的動作輸出將同時(同一掃描內)被執行。因此，請勿創建依存於動作輸出的執行順序的SFC程式。

(1)、(2)、(3)的動作輸出的執行順序不固定。



有/無連續移轉的動作

在SFC程式的移轉條件中，存在“有連續移轉”及“無連續移轉”的動作。

有/無連續移轉的設置，根據SFC用資訊元件的連續移轉位元的設置及SM323(所有塊連續移轉的有無)來決定。

連續移轉位元	SM323	內容	
無設置	OFF	無連續移轉	移轉條件成立時，將在下一個掃描內執行移轉目標步的動作輸出。
	ON	有連續移轉	移轉條件成立時，將在同一掃描內執行移轉目標步的動作輸出。 步的移轉條件連續成立的情況下，在移轉條件變為不成立之前或到達結束步之前，將在同一掃描內執行。
OFF	ON/OFF	無連續移轉	移轉條件成立時，將在下一個掃描內執行移轉目標步的動作輸出。
ON	ON/OFF	有連續移轉	移轉條件成立時，將在同一掃描內執行移轉目標步的動作輸出。 步的移轉條件連續成立的情況下，在移轉條件變為不成立之前或到達結束步之前，將在同一掃描內執行。

要點

透過設置為“有連續移轉”，可以縮短節拍時間。因此，可以消除從移轉條件成立開始到移轉目標步的動作輸出執行為止的等待時間。

但是，設置為“有連續移轉”時，其他塊及順控程式的動作有可能會變慢。

8.7 SFC程式的執行

SFC程式的啟動及停止

SFC程式的啟動及停止方法如下所示。

- 透過CPU參數進行的自動啟動
- 透過特殊繼電器(SM321)進行的啟動及停止
- 透過指令進行的啟動及停止

透過CPU參數進行的自動啟動

將CPU參數的“啟動條件設定”設置為“自動啟動”時，CPU模組的電源ON時、復位時或STOP→RUN時SFC程式將啟動，並啟動塊0。(☞121頁 啟動條件設置)

透過特殊繼電器(SM321)進行的啟動及停止

在執行SFC程式時，SM321(SFC程式的啟動/停止)透過系統自動被ON。

- 透過將SM321置為OFF，可以結束全部SFC程式的處理。
- 透過將SM321置為ON，可以再次執行結束的SFC程式。

要點

透過對CPU參數的“SFC程式啟動模式設定”進行設置，可以繼續啟動SFC程式。(☞119頁 SFC程式啟動模式設置)

透過指令進行的啟動及停止

透過程式控制用指令，可以對SFC程式進行啟動或停止。(☞126頁 透過指令進行執行類型的更改)

- 如果執行PSCAN指令，可以啟動待機類型的SFC程式。執行類型將變為掃描執行類型。
- 如果執行POFF指令，則可在將執行中的SFC程式的輸出置為OFF後停止。執行類型將變為待機類型。

塊的啟動及結束

塊的啟動方法

塊的啟動方法如下所示。

項目	啟動方法	備注	參照目標
透過CPU參數進行的自動啟動(僅塊0)	透過在CPU參數的SFC設置中將“啟動條件設定”設置為“自動啟動塊0”，則在SFC程式的啟動時塊0將被自動啟動，並從初始步開始執行處理。	在將塊0作為管理塊及前處理塊、常時監視塊等使用時進行設置。	☞121頁 啟動條件設置
透過塊啟動步進行的啟動	在SFC程式的各塊中，透過塊啟動步[BC或BS]啟動其他塊。	控制的順序明確時有效。	☞88頁 塊啟動步(有結束檢查)[BC] ☞89頁 塊啟動步(無結束檢查)[BS]
透過SFC控制指令進行的啟動	從SFC程式的動作輸出或其他順控程式，透過SFC控制指令對指定的塊進行啟動。 • 從指定的塊的初始步開始執行的情況下，使用SET [BL□]指令(塊啟動)。 • 從指定的塊的指定步開始執行的情況下，使用SET [S□/BL□\S□]指令(步啟動)。	在異常檢測時等對出錯恢復處理塊執行啟動、執行中斷處理時等有效。	☞108頁 SFC控制指令
透過SFC用資訊元件進行的啟動	透過將各塊中設置的“塊啟動結束位元”作為SFC用資訊元件置為ON來啟動指定塊。	也可透過外部設備進行啟動，因此在塊單位的偵錯、試運行時有效。	☞111頁 塊啟動結束位元
透過工程工具進行的啟動	透過將SFC塊元件置為ON來啟動指定塊。	在偵錯及試運行時有效。	📖GX Works3 操作手冊

塊的結束方法

塊的結束方法如下所示。

項目	結束方法	備注	參照目標
透過結束步進行的結束	如果執行塊內的結束步，則將結束塊的處理，並變為非激活狀態。	在自動運行時循環停止中停止動作等時有效。	☞90頁 結束步
透過SFC控制指令進行的結束	從SFC程式的動作輸出或其他順控程式，透過RST [BL□]指令(塊啟動)使指定的塊結束，並置為非激活狀態。(透過RST [BL□\S□]指令(步結束)，將指定塊內的激活步全部置為了非激活狀態時也結束塊。)	與動作狀態無關，透過緊急停止等中止處理時有效。	☞108頁 SFC控制指令
透過SFC用資訊元件進行的結束	透過將各塊中設置的“塊啟動結束位元”作為SFC用資訊元件置為OFF來結束指定塊。	也可透過外部設備進行結束，因此在塊單位的偵錯、試運行時有效。	☞111頁 塊啟動結束位元
透過工程工具進行的結束	透過將SFC塊元件置為OFF來結束指定塊。	在偵錯及試運行時有效。	📖GX Works3 操作手冊

塊的暫時停止及重啟

塊的停止方法

在SFC程式執行中使指定的塊停止的方法如下所示。

項目	停止方法	備注	參照目標
透過SFC控制指令進行的停止	從SFC程式的動作輸出或其他順控程式，透過PAUSE[BL□]指令(塊停止)暫時停止指定的塊的執行。	在檢測出異常等情況下，暫時停止機械並透過手動運行對異常位置進行修復時有效。	☞108頁 SFC控制指令
透過SFC用資訊元件進行的停止	作為SFC用資訊元件，透過將各塊中設置的“塊停止再次開始位元”置為ON來停止指定塊。	也可透過外部設備進行停止，因此在偵錯、試運行時進行確認的同時進行控制等情況下有效。	☞114頁 塊停止重啟位元

塊停止/重啟時的動作，根據SM325(塊停止時的輸出模式設置)與SFC用資訊元件的塊停止模式位元的設置、步的保持/非保持的組合來決定。(☞122頁 塊停止重啟時的動作)

塊的重啟方法

在SFC程式執行中使暫時停止的塊的處理重啟的方法如下所示。

項目	重啟方法	備注	參照目標
透過SFC控制指令進行的重啟	從SFC程式的停止塊以外的動作輸出或其他順控程式，透過RSTART[BL□]指令(塊重啟)對指定的塊進行重啟。	暫時停止中的手動控制完成，返回至自動運行等時有效。	☞108頁 SFC控制指令
透過SFC用資訊元件進行的重啟	透過將各塊中設置的“塊停止再次開始位元”作為SFC用資訊元件置為OFF，重啟指定塊。	也可透過外部設備進行啟動，因此在塊單位的偵錯、試運行時有效。	☞114頁 塊停止重啟位元

塊停止/重啟時的動作，根據SM325(塊停止時的輸出模式設置)與SFC用資訊元件的塊停止模式位元的設置、步的保持/非保持的組合來決定。(☞122頁 塊停止重啟時的動作)

步的啟動及結束(激活及非激活)

步的啟動(激活)方法

對步進行啟動(激活)的方法如下所示。

項目	啟動方法	備注	參照目標
透過移轉條件成立進行的啟動	如果之前的移轉條件成立，則下一個步將自動啟動。	—	☞98頁 移轉條件
透過SFC控制指令進行的啟動	從SFC程式的動作輸出或其他順控程式，透過SET [S□/BL□\S□]指令(步啟動)對指定的步進行啟動。	—	☞108頁 SFC控制指令
透過工程工具進行的啟動	• 透過將步繼電器置為ON來啟動指定步。 • 透過選單的[偵錯]⇨[SFC步序控制]將選擇的步置為激活狀態。	在偵錯及試運行時有效。	📖GX Works3 操作手冊

步的結束(非激活)方法

對步進行結束(非激活)的方法如下所示。

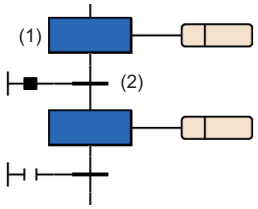
項目	結束方法	備注	參照目標
透過移轉條件成立進行的結束	步的下一個移轉條件成立時，將自動結束。	—	☞98頁 移轉條件
透過復位步[R]進行的結束	復位步[R]變為激活時，在屬性指定目標中指定的步將結束。	在SFC程式的選擇分支中移轉至出錯處理的步時等，結束保持步[SC、SE、ST]的情況下有效。	☞88頁 復位步[R]
透過SFC控制指令進行的結束	從SFC程式的動作輸出或其他順控程式，透過RST [S□/BL□\S□]指令(步結束)對指定的步進行結束。	透過RST指令指定塊的全部步變為非激活狀態時，塊也將結束。	☞108頁 SFC控制指令
透過工程工具進行的結束	• 透過將步繼電器置為OFF來結束指定步。 • 透過選單的[偵錯]⇨[SFC步序控制]將選擇的步置為非激活狀態。	在偵錯及試運行時有效。	📖GX Works3 操作手冊

步雙重啟動時的注意點

對步進行了雙重啟動時的動作如下所示。

串聯移轉的情況下

移轉條件(2)成立時，步(1)將變為非激活狀態



選擇移轉的情況下

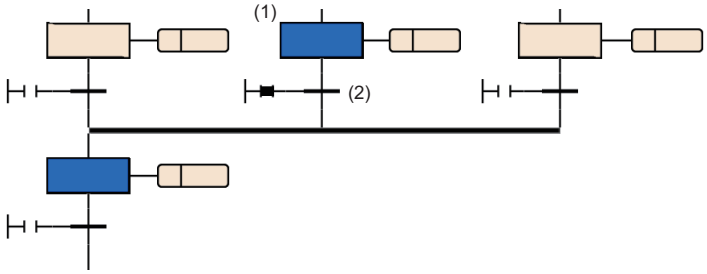
■選擇分支

選擇分支時，從左開始按順序對移轉條件進行檢查，如果移轉條件成立的分支的移轉目標為激活步，將與串聯移轉的情況相同。此時，即使在右分支中條件也成立的情況下，也不檢查該條件。

■選擇合併

選擇合併時的雙重啟動的動作與串聯移轉的情況相同。

移轉條件(2)成立時，步(1)將變為非激活狀態。

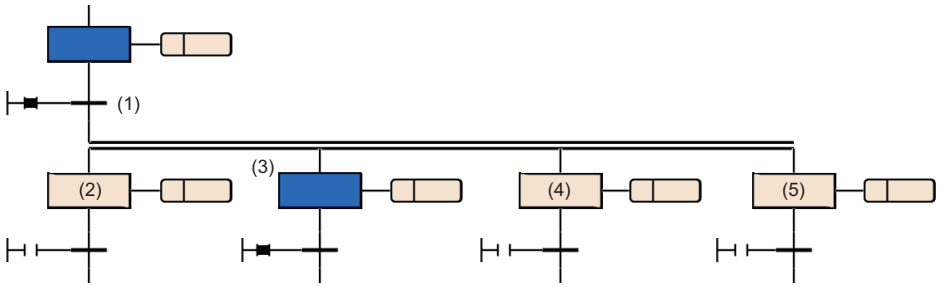


並聯移轉的情況下

■並聯分支

並聯分支的情況下，在多個移轉目標中只要有一個為激活狀態，在下一個掃描中移轉目標將全部變為激活狀態。

移轉條件(1)成立時，在下一個掃描中步(2)～步(5)將全部變為激活狀態。



■並聯合併

移轉源將變為非激活狀態。保持步[SC、SE、ST]將變為保持狀態。

程式更改時的動作

SFC程式的更改方法如下。

- 寫入至可程式控制器
- RUN中寫入
- SFC塊RUN中寫入*1

可用上述方法更改的SFC程式內容如下所示。

更改的情況			寫入至可程式控制器		RUN中寫入	SFC塊RUN中寫入*1
			STOP/PAUSE狀態	RUN狀態		
SFC程式的添加			○	×	×	×
SFC塊的添加/刪除			○	×	×	○
SFC塊的更改	SFC圖的更改	步・移轉條件的添加/刪除	○	×	×	○
		移轉條件(分支/合併/跳轉)的更改	○	×	×	○
		步的屬性更改	○	×	×	○
	SFC圖內的更改	動作輸出程式的更改	○	×	○	○
		移轉條件程式的更改	○	×	○	○
	塊資訊的更改		○	×	×	○

*1 SFC塊RUN中寫入的有關內容，請參閱下述章節。
☞ 137頁 SFC塊RUN中寫入
此外，欲使用SFC塊RUN中寫入的情況下，應確認CPU模組及工程工具的版本。(MELSEC iQ-R CPU模組用戶手冊(應用篇))

透過寫入至可程式控制器進行的程式更改

透過寫入至可程式控制器進行的程式更改後的動作如下所示。

SM322(SFC程式的啟動狀態)	寫入前後的程式更改有無	
	有更改	無更改
OFF(初始啟動)	初始啟動	初始啟動
ON(繼續運行啟動)	初始啟動	繼續運行啟動

關於全部程式內使用的各元件或標籤，根據SM326(SFC的元件·標籤清除模式)的設置進行的動作如下所示。

SM326(SFC的元件·標籤清除模式)*1	內容
OFF	對下述以外的全部元件及標籤進行清除之後，執行SFC程式。 ・步繼電器(S) ・檔案寄存器(R/ZR)*2 ・鎖存指定的標籤
ON	對步繼電器(S)以外的全部元件及標籤的值進行保持的狀態下，執行SFC程式。

*1 SM326的設置僅在寫入至可程式控制器後存在SFC程式時有效。此外，不僅SFC程式的寫入，在有程式檔案及參數檔案的寫入的情況下也有效。但是，僅在通用元件注釋、元件記憶體、元件初始值的寫入時無效。
*2 即使未對檔案寄存器(R/ZR)進行鎖存設置，也不會變為SM326的清除對象。

■STOP→RUN操作

在SFC程式的動作中(RUN中)使CPU模組STOP的情況下，在STOP→RUN時元件的狀態與SFC程式的激活狀態均恢復為STOP前的狀態。與CPU參數的SFC程式啟動模式設置無關，將變為繼續運行啟動。

在STOP過程中，將順控程式檔案(包括SFC程式)、FB檔案、參數檔案(CPU參數、系統參數等)的任意一個寫入到CPU模組中的情況下，在RUN時如果SFC程式存在，將變為初始啟動。但是在程式的寫入前後無更改的情況下有可能繼續運行啟動。(☞119頁 SFC程式啟動模式設置)

■注意事項

- 透過寫入至可程式控制器進行程式更改後，應在進行了一次復位後，執行SFC程式。
- CPU參數的SFC程式啟動模式設置為繼續運行啟動的情況下，應將SM322(SFC程式的啟動狀態)置為OFF(初始啟動)後，進行透過寫入至可程式控制器的程式更改。之後，應在使SFC程式初始啟動之後，再次將SM322置為ON(繼續運行啟動)。

透過RUN中寫入進行的程式更改

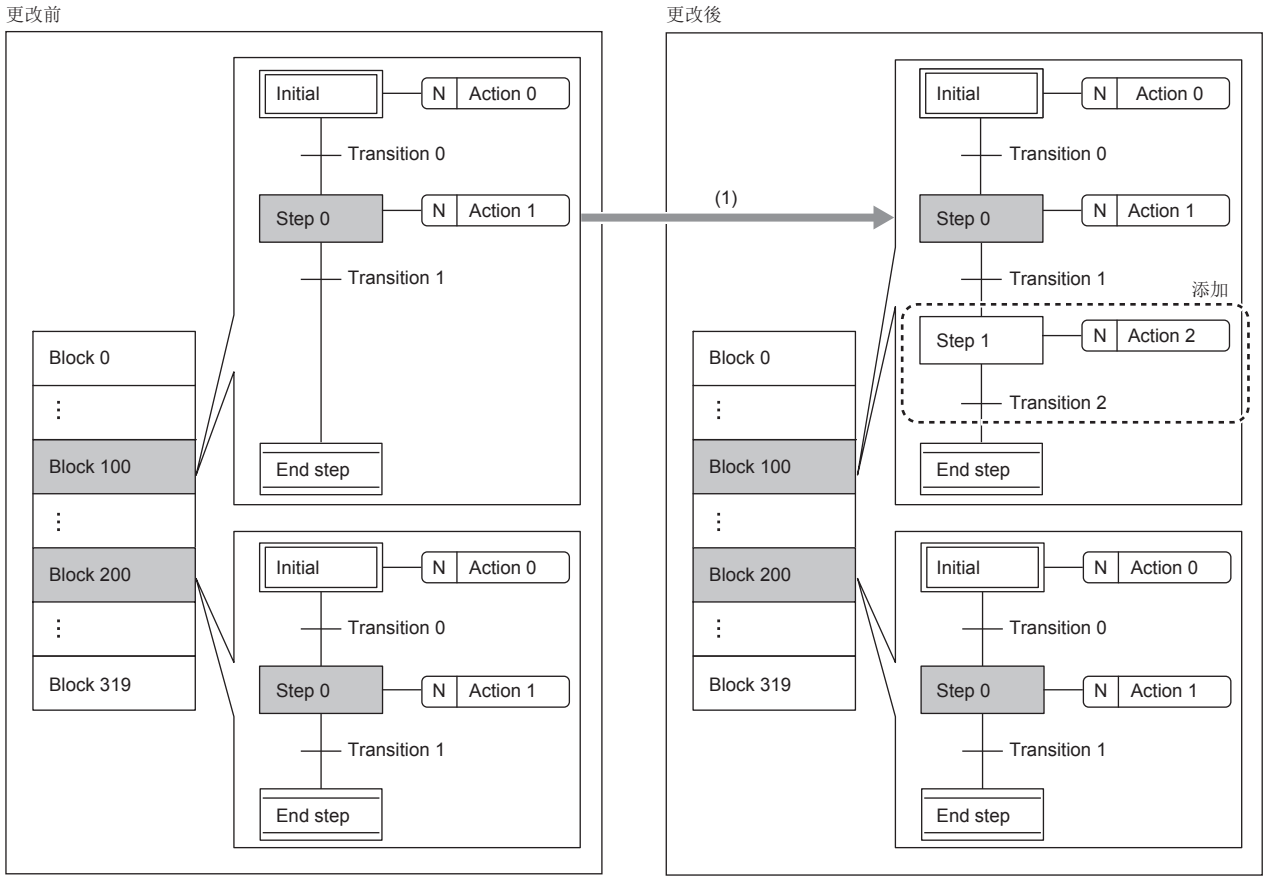
透過RUN中寫入進行程式更改後，與CPU參數的SFC程式啟動模式設置無關，必須進行繼續運行啟動。

SFC塊RUN中寫入

RnCPU RnENCPU RnPCPU(過程) RnPCPU(二重化) RnPSFCPU(常規) RnPSFCPU(安全) RnSFCPU(常規) RnSFCPU(安全)

• 應確認CPU模組及工程工具的版本。(MELSEC iQ-R CPU模組用戶手冊(應用篇))

可以以塊為單位對SFC程式進行更改。即使在RUN中也繼續保持激活狀態且能以塊單位進行程式更改，因此能夠提高SFC程式的偵錯及維護的效率。



：非激活的塊或步

：激活中的塊或步

(1)可以對激活中的SFC塊(激活步除外)的程式進行更改。

限制事項

使用SFC塊RUN中寫入的情況下，應確認CPU模組及工程工具的版本。
關於CPU模組及工程工具的版本，請參閱下述手冊。
MELSEC iQ-R CPU模組用戶手冊(應用篇)

■程式中的執行類型的執行可否

僅可在掃描執行類型程式中執行。(在待機類型程式中無法執行。)

■SM321(SFC程式的啟動/停止)OFF時的SFC塊RUN中寫入

SM321(SFC程式的啟動/停止)為OFF時，無論變為OFF前的對象塊的活性狀態為何，皆可以執行RUN中寫入。此外，在SM321=OFF時執行了SFC塊RUN中寫入的情況下，無論下述的設置為何，皆會變為初始啟動。

- CPU參數的SFC繼續啟動設置(☞ 119頁 SFC程式啟動模式設置)
- SM322(SFC程式的啟動狀態)

■更改的對象塊

應在逐塊更改後執行SFC塊RUN中寫入。更改了多個SFC塊的狀態下，無法執行SFC塊RUN中寫入。

■塊的更改/添加/刪除

SFC塊RUN中寫入的塊的更改/添加/刪除如下所示。

操作	內容	
	對象塊為非激活的情況下	對象塊為激活的情況下
塊的更改	• 可以更改已經存在於CPU模組內的SFC程式中的SFC塊的程式。 • 可以更改對象SFC塊的SFC用資訊元件。	• 可以更改已經存在於CPU模組內的SFC程式中的SFC塊的程式。但是，無法對激活步進行刪除、屬性變更、步No.變更。 • 可以更改對象SFC塊的SFC用資訊元件。
塊的添加	• 可以將新的SFC塊添加到CPU模組內的SFC程式中。 • 可以添加對象SFC塊的SFC用資訊元件。	—
塊的刪除	• 可以將指定的SFC塊從CPU模組中的SFC程式刪除。 • 可以刪除對象SFC塊的SFC用資訊元件。 • 對象塊不存在於CPU模組的SFC程式中時，無法執行。	激活中的SFC塊無法刪除。

要點

CPU模組為STOP、PAUSE狀態時，已激活的步將會維持激活狀態。因此，實施步的刪除、屬性變更、步No.變更的情況下，應在將維持著激活狀態的步置為非激活化後，再執行SFC塊RUN中寫入。

■程式取代範圍

對象塊中，僅取代添加/更改的步及移轉條件、添加/更改的動作輸出及移轉條件內的梯形圖。此外，添加/更改對象的動作輸出、移轉條件內的梯形圖中的上升沿、下降沿指令的上次的執行資訊會被初始化。

■SFC塊RUN中寫入中的程式的執行類型更改可否

對於RUN中寫入執行中的程式檔案，無法透過程式控制指令(POFF/PSCAN指令)進行執行類型的更改。已執行的情況下，指令將變為無處理。

■執行狀態的確認

對於SFC塊RUN中寫入的執行狀態，可以透過SM329(SFC塊RUN中寫入執行中標誌)/SD329(SFC塊RUN中寫入對象塊No.)進行確認。
(📖MELSEC iQ-R CPU模組用戶手冊(應用篇))

■RUN中寫入用確保步

為了執行SFC塊RUN中寫入，需要在CPU模組中添加•更改容量份的RUN中寫入用確保步容量。執行SFC塊RUN中寫入時，添加•更改容量超過RUN中寫入用確保步容量的情況下，如果程式記憶體有空間，則可以重新設置RUN中寫入用確保步。
關於RUN寫入用確保步，請參閱下述手冊。
📖MELSEC iQ-R CPU模組用戶手冊(應用篇)

■引導運行中的SFC塊RUN中寫入

在從SD記憶卡進行引導運行的過程中執行了SFC塊RUN中寫入的情況下，也可以更改引導源的SD記憶卡中相應的檔案。

■在SFC塊RUN中寫入中試圖啟動對象塊或步時的動作

欲在RUN中寫入執行中時將SFC塊RUN中寫入的對象塊或步啟動的情況下，對象塊或步將不啟動。塊或步的啟動類型的動作分別如下所示。

對象塊或步的啟動類型(激活方法)	塊或步啟動時的動作
透過CPU參數進行的自動啟動	在SFC塊RUN中寫入完成之前將不會啟動。在SFC塊RUN中寫入完成後將會自動啟動。
塊啟動步(無結束檢查)	• 在SFC塊RUN中寫入完成之前，對象塊將會待機不啟動。即使步中附隨的移轉條件成立，也不會移轉到下一個步。 • 在SFC塊RUN中寫入完成後啟動對象塊。移轉條件成立時，將移轉到下一個步。
塊啟動步(有結束檢查)	• 在SFC塊RUN中寫入完成之前，對象塊將會待機不啟動。 • 在SFC塊RUN中寫入完成後啟動對象塊。塊結束後，只要附隨的移轉條件成立，即會移轉到下一個步。
SFC控制指令(SET BL□、SET S□、SET BL□\S□指令)	不啟動對象塊。指令觸點持續為ON的情況下，將會在SFC塊RUN中寫入完成後啟動對象塊。
SFC用資訊元件(透過塊啟動結束位元進行的啟動)	即使塊啟動結束位元為ON，對象塊也不會啟動。塊啟動結束位元為ON的情況下，將會在SFC塊RUN中寫入完成後啟動對象塊。(在SFC塊RUN中寫入完成之前，系統將不會啟動相應的塊。)
透過工程工具進行的塊啟動*1	不啟動對象塊。忽略請求。(在SFC塊RUN中寫入完成之前，系統將不會啟動相應的塊。)
透過移轉條件成立進行的步的啟動(激活)	• 在SFC塊RUN中寫入完成之前，透過之前的移轉條件的成立，RUN中寫入對象的SFC步將不會啟動。(激活狀態將會待機不移轉。) • 移轉條件在SFC塊RUN中寫入完成後成立的情況下，激活狀態將會移轉。

*1　為透過監看視窗中的BL□、BL□\S□、元件/緩衝記憶體批量監視、元件→SFC控制的啟動。

■注意事項

- SFC塊RUN中寫入中(包含傳送至程式記憶體)時，請勿進行電源OFF或復位。如果已進行，應再次進行寫入至可程式控制器的操作。
- 無法同時執行SFC塊RUN中寫入及從工程工具進行下述操作。
  - RUN中的梯形圖塊更改/寫入至可程式控制器、SFC塊RUN中寫入
  - 寫入至可程式控制器(元件/局部元件/全局元件/局部標籤資料除外)
  - 記憶體的初始化
- 欲刪除控制中不需要的OUT指令(線圈)的情況下，應確認OUT指令為OFF後再進行刪除。未進行OFF而刪除OUT指令時，輸出將維持被保持的狀態。
- 在更改梯形圖內調用子程式類型FB的情況下，正在調用的子程式類型FB的FB定義內的上升沿，下降沿指令等前一次的執行資訊將不會被初始化。
- 使用了SFC塊RUN中寫入的情況下，中斷程式的啟動可能需要等待。因此，在使用了模組間同步中斷(I44)、多CPU間同步中斷(I45)的程式中，監視中斷程式執行時間的情況下(CPU參數的異常檢測設置)，CPU模組中有可能會檢測出出錯。(MELSEC iQ-R CPU模組用戶手冊(應用篇))
- SFC塊RUN中寫入時，由於部分區間非為掃描監視對象，因此即使掃描時間超出了掃描時間監視時間(WDT)設置中設置的時間，也可能會檢測不出WDT時間超出的情況。
- SFC塊RUN中寫入過程中，在執行的指令中檢測出出錯的情況下，由於程式位置資訊(步No.)將變為寫入中的程式的步No.，因此不會變為寫入完後的程式的步No.。
- 下述的情況下，無法執行SFC塊RUN中寫入。
  - 對象的程式檔案在參數設置中未登錄的情況下
  - SFC步數超過在CPU參數的元件設置中設置的步繼電器的點數的情況下
  - 將CPU模組置為STOP狀態，指定程式或參數並執行寫入至可程式控制器時的STOP→RUN的期間
  - 發生程式執行不可(出錯代碼：3204H)的情況下
- 激活塊的SFC塊RUN中寫入中時，應使用SM329(SFC塊RUN中寫入執行中標誌)或SD329(SFC塊RUN中寫入的對象塊No.)採取互鎖等，以防止激活步的移轉條件成立。至對象步的移轉條件在SFC塊RUN中寫入中成立的情況下，激活狀態將待機不移轉。移轉條件在SFC塊RUN中寫入完成後成立的情況下，移轉至對象步。此外，已設置步移轉位元的情況下，待機中的位元不變為ON。

• 欲實施SFC塊RUN中寫入(包含跨越激活步的步移動)的情況下，應按下述步驟實施。

❶剪下欲移轉的步。

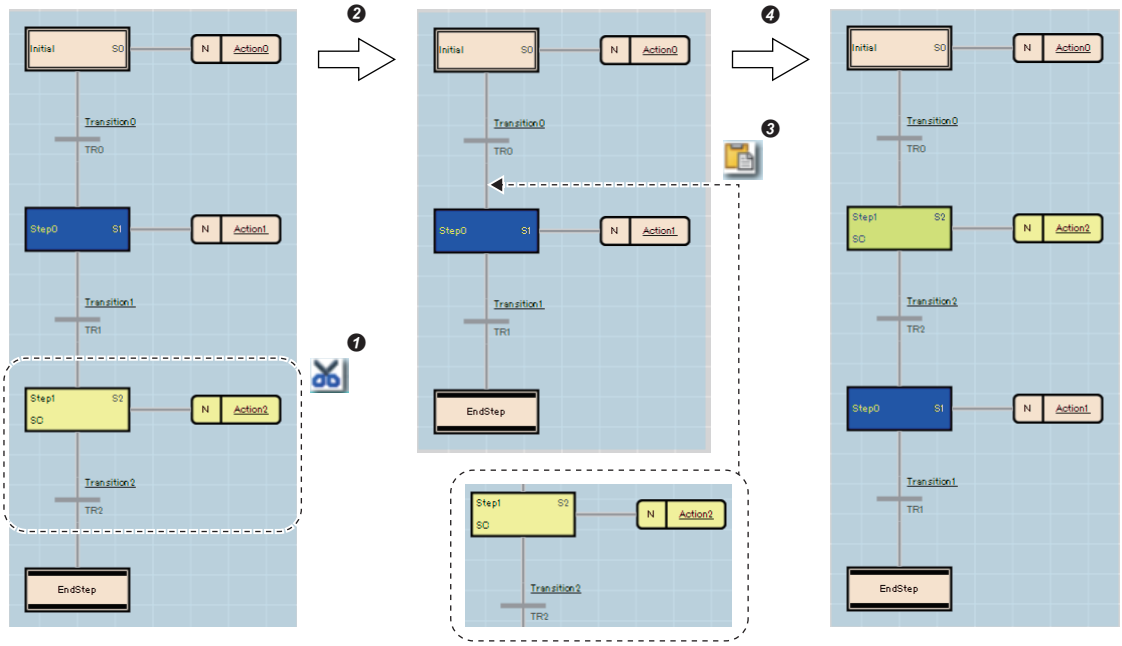
❷執行SFC塊RUN中寫入。

❸將剪下的步貼到目的位置。

❹再次執行SFC塊RUN中寫入。

如果省略上述❷，按❶→❸→❹的順序執行，則可能發生無法執行SFC塊RUN中寫入的情況。

編輯前　　編輯後



SFC程式的動作確認

SFC程式的動作確認中可使用的工程工具的功能如下所示。

- 監視
- 檢視
- 元件/緩衝記憶體批量監視
- SFC步控制
- SFC塊一覽表
- SFC塊批量監視
- 激活步監視
- SFC已激活步監視

要點

透過工程工具進行的SFC程式的動作確認的有關內容，請參閱下述手冊。

GX Works3 操作手冊

附錄

附1　使用MC/MCR指令控制EN時的動作

在FB的固有屬性設置中“使用MC/MCR控制EN”置為有效的情況下，FB中使用的指令、元件/標籤的動作如下所示。

FB中使用的指令、元件/標籤	FB中使用的指令、元件/標籤的狀態	
	在“使用MC/MCR控制EN”中選擇“是”時	在“使用MC/MCR控制EN”中選擇“否”時
上升沿/下降沿指令(PLS指令、脈衝化指令(□P)*1	下次EN為ON時，如果條件觸點成立，則執行指令。	下次EN為ON時，即使條件觸點成立也有可能不執行指令。
定時器(低速/高速)、長定時器	計數值變為0，線圈、觸點變為OFF。	保持現在的狀態。
累計定時器(低速/高速)、長累計定時器、計數器、長計數器	線圈變為OFF，但計數值、觸點都將保持現在的狀態。	保持現在的狀態。
OUT指令的元件部份中指定的元件	強制變為OFF。	保持現在的狀態。

*1　在線圈側指定的指令為對象。

限制事項

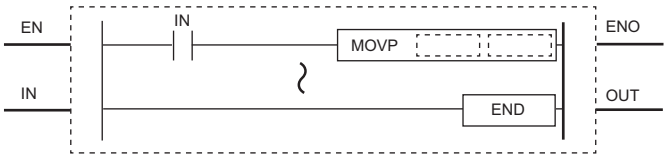
將“使用MC/MCR控制EN”置為“是”的情況下，請勿在該FB的執行過程中使用MC/MCR指令。如果使用了MC/MCR指令，則EN的控制有可能不會正確地動作。

上升沿/下降沿指令的動作

上升沿/下降沿指令的動作如下所示。

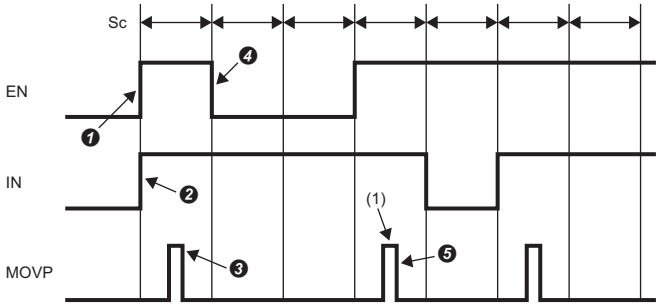
例

使用上升沿指令的子程式類型FB



■在“使用MC/MCR控制EN”中選擇“是”時

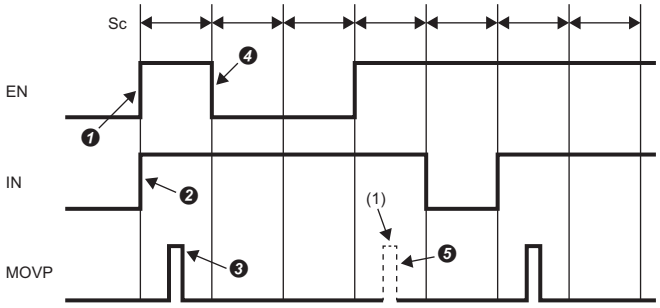
EN為ON時，如果條件觸點成立，則執行指令。(圖中(1))



Sc： 掃描
❶將EN置為ON。(用戶操作)
❷將IN置為ON。(用戶操作)
❸執行MOVP指令。(CPU模組動作)
❹將EN置為OFF。(用戶操作)
❺執行MOVP指令。(CPU模組動作)

■在“使用MC/MCR控制EN”中選擇“否”時

EN為OFF時，指令的動作將根據條件觸點狀態而有所不同。(圖中(1))



Sc： 掃描
❶將EN置為ON。(用戶操作)
❷將IN置為ON。(用戶操作)
❸執行MOVP指令。(CPU模組動作)
❹將EN置為OFF。(用戶操作)
❺在❹中EN變為OFF前的條件觸點為OFF的情況下，執行MOVP指令。(CPU模組動作)(在❹中EN變為OFF前的條件觸點為ON的情況下，不執行MOVP指令。)

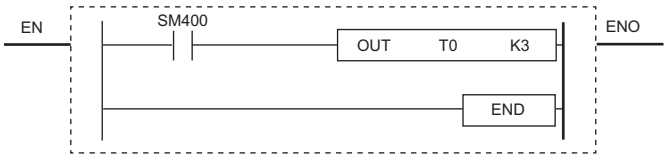
附

定時器(低速/高速)、長定時器的動作

定時器(低速/高速)、長定時器的動作如下所示。

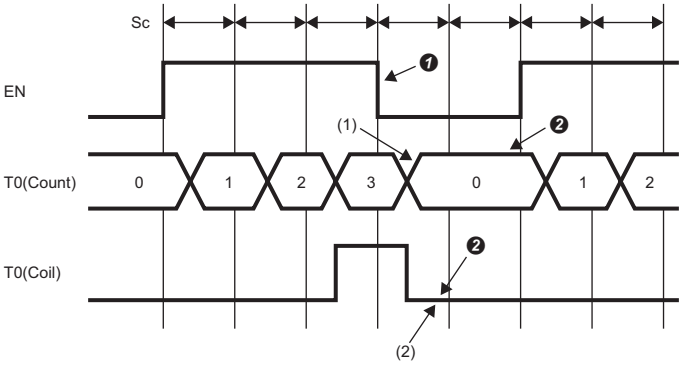
例

使用低速定時器的子程式類型FB



■在“使用MC/MCR控制EN”中選擇“是”時

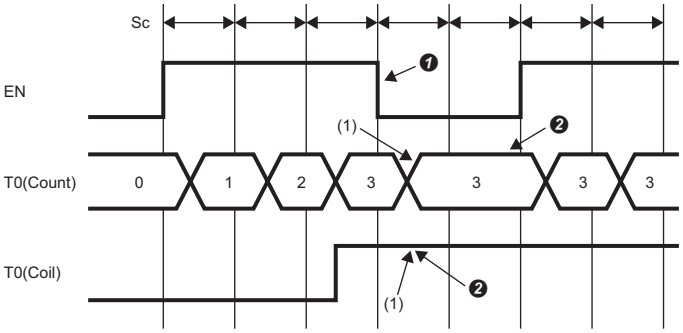
計數值變為0。(圖中(1))或線圈變為OFF。(圖中(2))



Sc: 掃描
T0(Count): T0(計數值)
T0(Coil): T0(線圈)
❶將EN置為OFF。(用戶操作)
❷線圈變為OFF，並清除定時器值、計數值。(CPU模組動作)

■在“使用MC/MCR控制EN”中選擇“否”時

計數值、線圈都將保持現在的狀態。(圖中(1))



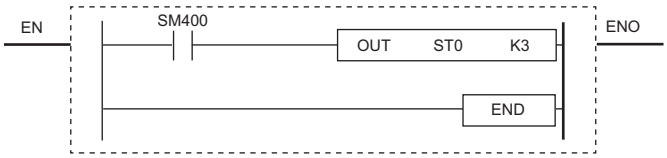
Sc: 掃描
T0(Count): T0(計數值)
T0(Coil): T0(線圈)
❶將EN置為OFF。(用戶操作)
❷保持值。(CPU模組動作)

累計定時器(低速/高速)、長累計定時器、計數器、長計數器的動作

累計定時器(低速/高速)、長累計定時器、計數器、長計數器的動作如下所示。

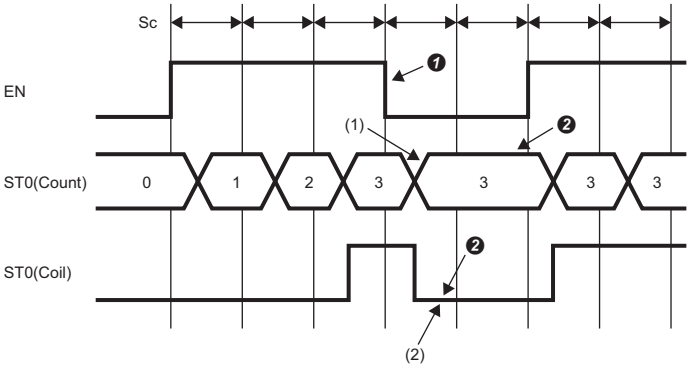
例

使用低速累計定時器的子程式類型FB



■在“使用MC/MCR控制EN”中選擇“是”時

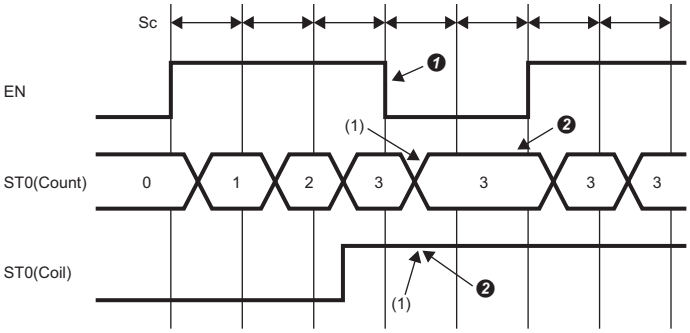
計數值將保持現在的狀態。(圖中(1))或線圈變為OFF。(圖中(2))



Sc: 掃描
ST0(Count): T0(計數值)
ST0(Coil): T0(線圈)
❶將EN置為OFF。(用戶操作)
❷線圈變為OFF，但計數值、觸點都將保持現在的狀態。(CPU模組動作)

■在“使用MC/MCR控制EN”中選擇“否”時

計數值、線圈都將保持現在的狀態。(圖中(1))



Sc: 掃描
ST0(Count): T0(計數值)
ST0(Coil): T0(線圈)
❶將EN置為OFF。(用戶操作)
❷保持值。(CPU模組動作)

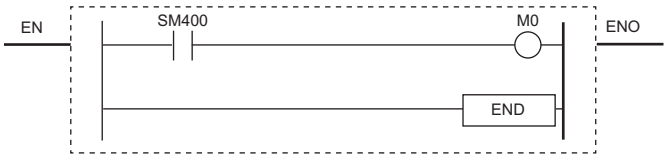
附

OUT指令的元件部份中指定的元件的動作

OUT指令的元件部份中指定的元件的動作如下所示。

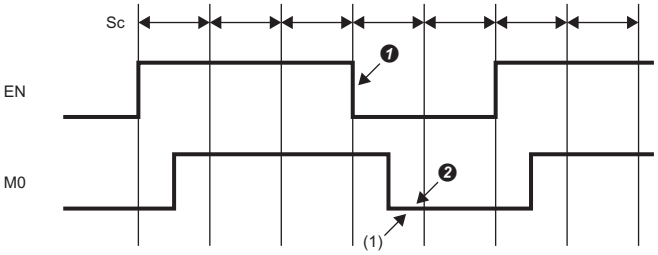
例

OUT指令的元件部份中使用M0的子程式類型FB



■在“使用MC/MCR控制EN”中選擇“是”時

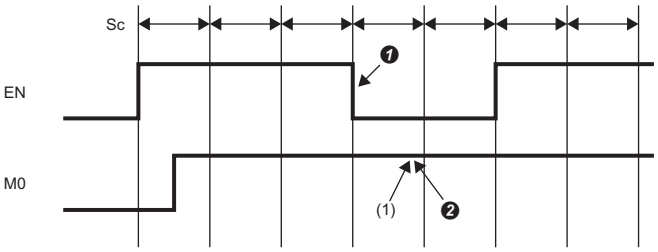
M0將強制變為OFF。(圖中(1))



Sc： 掃描
❶將EN置為OFF。(用戶操作)
❷將線圈置為OFF。(CPU模組動作)

■在“使用MC/MCR控制EN”中選擇“否”時

M0將保持現在的狀態。(圖中(1))



Sc： 掃描
❶將EN置為OFF。(用戶操作)
❷保持線圈。(CPU模組動作)

備忘錄

索引

七畫

八畫

九畫

十一畫

十二畫

十三畫

十四畫

十五畫

十六畫

十七畫

十九畫

索

修訂記錄

*本手冊號在封底的左下角。

修訂日期	*手冊編號	修改內容
2014年8月	SH(NA)-081320CHT-A	第一版
2014年12月	SH(NA)-081320CHT-B	■第二版 部分修改
2015年3月	SH(NA)-081320CHT-C	■第三版 部分修改
2015年10月	SH(NA)-081320CHT-D	■第四版 部分修改
2016年6月	SH(NA)-081320CHT-E	■第五版 部分修改
2017年1月	SH(NA)-081320CHT-F	■第六版 部分修改
2017年7月	SH(NA)-081320CHT-G	■第七版 部分修改
2017年11月	SH(NA)-081320CHT-H	■第八版 部分修改
2018年6月	SH(NA)-081320CHT-I	■第九版 部分修改
2018年11月	SH(NA)-081320CHT-J	■第十版 部分修改

日語版手冊編號：SH-081225-L

保固

使用之前請確認以下產品保固的詳細說明。

1. 免費保固期限和免費保固範圍

在免費保固期內使用本產品時如果出現任何屬於三菱電機責任的故障或缺陷（以下稱“ 故障 ”），則經銷商或三菱電機服務公司將負責免費維修。

但是如果需要在國內現場或海外維修時，則要收取派遣工程師的費用。對於涉及到更換故障模組後的任何再試運轉、維護或現場測試，三菱電機將不負任何責任。

【免費保固期限】

免費保固期限為自購買日或交貨的 36 個月內。

注意產品從三菱電機生產並出貨之後，最長分銷時間為 6 個月，生產後最長的免費保固期為 42 個月。維修零組件的免費保固期不得超過修理前的免責保固期。

【免費保固範圍】

(1) 範圍局限於按照使用說明書、用戶手冊及產品上的警示標語規定的使用狀態，使用方法和使用環境正常使用的情況下。

(2) 以下情況下，即使在免費保固期內，也要收取維修費用。

① 因不適當存放或搬運、用戶過失或疏忽而引起的故障。因使用者的硬體或軟體設計而導致的故障。

② 因用戶未經批准對產品進行改造而導致的故障等。

③ 對於裝有三菱電機產品的用戶設備，如果根據現有的法定安全措施或工業標準要求配備必需的功能或結構後，本可以避免的故障。

④ 如果正確維護或更換了使用手冊中指定的耗材（電池、背光燈、保險絲等）後，本可以避免的故障。

⑤ 因火災或異常電壓等外部因素以及因地震、雷電、風災和水災等不可抗力而導致的故障。

⑥ 根據從三菱出貨時的科技標準還無法預知的原因而導致的故障。

⑦ 任何非三菱電機或用戶責任而導致的故障。

2. 產品停產後的有償維修期限

(1) 三菱電機在本產品停產後的 7 年內受理該產品的有償維修。
停產的消息將以三菱電機技術公告等方式予以通告。

(2) 產品停產後，將不再提供產品（包括備品）。

3. 海外服務

在海外，維修由三菱電機在當地的海外 FA 中心受理。注意各個 FA 中心的維修條件可能會不同。

4. 機會損失、間接損失不在品質保證責任範圍

無論在保修期內的內和外，對於以下三菱將不承擔責任。

(1) 非三菱責任原因所導致的損害。

(2) 因三菱產品故障原因而引起客戶的機會損失，利潤的損失。

(3) 無論三菱是否預測由特殊原因而導致的損失和間接損失、事故賠償、以及三菱產品以外的損失。

(4) 對於用戶更換設備，重新調整了現場的機械設備，測試及其它作業等的補償。

5. 產品規格的改變

目錄、手冊或技術文檔中的規格如有改變，恕不另行通知。

商標

The company names, system names and product names mentioned in this manual are either registered trademarks or trademarks of their respective companies.

In some cases, trademark symbols such as ‘™’ or ‘®’ are not specified in this manual.

SH(NA)-081320CHT-J(1811)STC
MODEL: R-P-PS-CHT

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : TOKYO BUILDING, 2-7-3 MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS : 1-14 , YADA-MINAMI 5-CHOME , HIGASHI-KU, NAGOYA , JAPAN