MITSUBISHI ELECTRIC

三菱电机微型可编程控制器

MELSEC iQ-F series

MELSEC iQ-F
FX5编程手册(程序设计篇)

安全方面注意事项

(使用之前请务必阅读)

使用FX5可编程控制器之前,应仔细阅读各产品附带的手册及附带手册中所介绍的关联手册,同时在充分注意安全的前提下正确地操作。请妥善保管本手册以备需要时查阅，并应将本手册交给最终用户。

前言

此次承蒙购入MELSEC iQ-F系列可编程控制器产品，诚表谢意。

本手册是帮助用户理解进行FX5编程时所需的指令与函数的手册。

在使用之前，请阅读本书以及相关产品的手册，希望在充分理解其规格的前提下正确使用产品。

此外，希望本手册能够送达至最终用户处。

将本手册中介绍的程序示例应用到实际系统中时，应充分验证对象系统中不存在控制方面的问题。

使用时的请求

- 该产品是以一般的工业为对象制作的通用产品，因此不是以用于关系到人身安全之类的情况下使用的机器或是系统为目的而设计、制造的产品。
- 讨论将该产品用于原子能用、电力用、航空宇宙用、医疗用、搭乘移动物体用的机器或是系统等特殊用途的时候，请与本公司的营业窗口查询。
- 虽然该产品是在严格的质量体系下生产的，但是用于那些因该产品的故障而可能导致的重大故障或是产生损失的设备的时候，请在系统上设置备用机构和安全功能的开关。

预先通知

- 设置产品时如有疑问，请向具有电气知识(电气施工人员或是同等以上的知识)的专业电气技术人员咨询。关于该产品的操作和使用方法有疑问时，请向技术咨询窗口咨询。
- 本书、技术资料、样本等中记载的事例是作为参阅用的，不是保证动作的。选用的时候，请用户自行对机器、装置的功能和安全性进行确认以后使用
- 关于本书的内容，有时候为了改良可能会有不事先预告就更改规格的情况，还望见谅。
- 关于本书的内容期望能做到完美，可是万一有疑问或是发现有错误，烦请联系本公司或办事处。

目录

关联手册

手册名称<手册编号>	内容
MELSEC iQ-F FX5编程手册(程序设计篇) <JY997D58801>(本手册)	记载梯形图、ST、FBD/LD、SFC程序的规格以及标签的内容。
MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇) <JY997D58901>	记载在程序中可使用的指令和函数的规格的内容。
GX Works3操作手册 <SH-081271CHN>	记载GX Works3的系统配置、参数设置、在线功能的操作方法等简单工程及结构化工程通用的功能相关的内容。

术语

除特别注明的情况外，本手册中使用下列术语进行说明。

术语	内容
工程工具	是MELSEC可编程控制器软件包的产品名。
信号流	是程序及FB的各步中运算的上次执行/非执行状态。
软元件	CPU模块内部含有的软元件(X、Y、M、D等)。
缓冲存储器	是用于存储设置值、监视值等数据的智能功能模块的存储器。
程序部件	按功能分别定义的程序单位。通过程序的部件化，可以将程序分级时的低位处理按处理内容及功能分为若干个单位，创建各单位的程序。

总称/简称

除特别注明的情况外，本手册中使用下列总称/简称进行说明。

总称/简称	内容
FX3智能功能模块	FX3U-4AD、FX3U-4DA、FX3U-4LC、FX3U-1PG、FX3U-2HC、FX3U-16CCL-M、FX3U-64CCL、FX3U-128ASL-M、FX3U-32DP的总称。
FX5	FX5S、FX5UJ、FX5U、FX5UC可编程控制器的总称。
FX5 CPU模块	FX5S CPU模块、FX5UJ CPU模块、FX5U CPU模块、FX5UC CPU模块的总称。
FX5智能功能模块	FX5-4AD、FX5-4DA、FX5-8AD、FX5-4LC、FX5-20PG-P、FX5-20PG-D、FX5-40SSC-G、FX5-80SSC-G、FX5-40SSC-S、FX5-80SSC-S、FX5-ENET、FX5-ENET/IP、FX5-CCLGN-MS、FX5-CCLIEF、FX5-CCL-MS、FX5-ASL-M、FX5-DP-M的总称。
GX Works3	SWnDND-GXW3的总称产品名(n表示版本)。
智能功能模块	FX5智能功能模块、FX3智能功能模块的总称。
操作数	是在各个指令及函数的内部配置中使用的源数据(s)、接收数据(d)、软元件数(n)等的软元件部分的总称。

1 概要

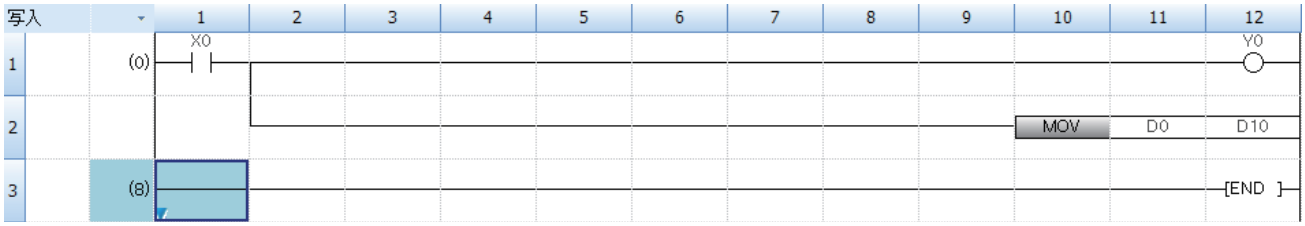
本手册中记载创建程序所必须的程序配置与内容、记述方法等有关内容。
关于在工程工具中的程序创建、编辑、监视方法，请参阅下述手册。
GX Works3操作手册

程序语言的类型

FX5系列中，可以根据用途选择最合适的程序语言使用。
○：支持，—：不支持

程序语言	内容	支持的CPU模块		
		FX5S	FX5UJ	FX5U/FX5UC
梯形图图表语言(梯形图语言)	是用触点及线圈等表示回路的图表语言。 梯形图语言是为了执行简单易懂的顺控程序控制而使用符号化的触点及线圈等记述逻辑回路的语言。	○	○	○
结构化文本语言(ST语言)	是使用IF语句或运算符等记述程序的文本语言。 ST语言与梯形图语言相比，因为可以对难以记述的运算处理简洁而直观地进行记述，因此适用于进行复杂的算术运算及比较运算等的领域。此外，可以与C语言等一样，通过条件语句进行选择分支，通过循环语句进行重复等的语句对控制进行记述，因此可以简洁地编写程序。	○	○	○
FB图表/梯形图语言(FBD/LD语言)	是通过将实施特定处理的块(功能部件、FB部件)、变量部件、常数部件等沿着数据和信号的流动进行连接，记述程序的图表语言。 能够轻松地创建利用梯形图程序创建时较为复杂的程序，改善程序生产效率。	○	○	○
顺控程序功能图(SFC程序)	是将一系列的控制动作分割为多个步，使程序的执行顺序和执行条件清楚的表达的程序记述形式。	○	○	○

■梯形图语言



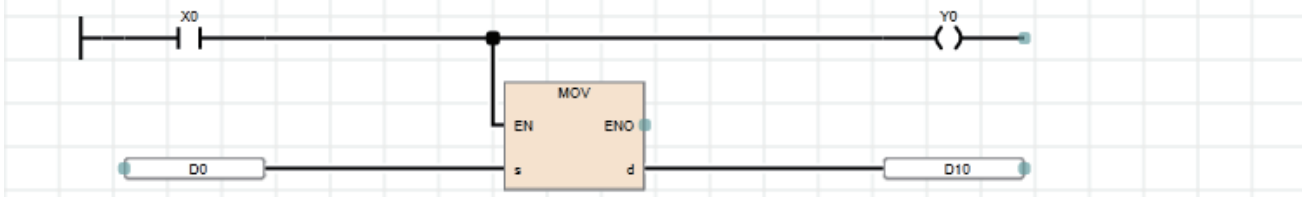
使用梯形图语言的情况下，请参阅下述内容。
☞38页 梯形图语言

■ST语言

```
1 IF X0 THEN
2     Y0 := TRUE ;
3     D0 := D10 ;
4 END_IF ;
5
```

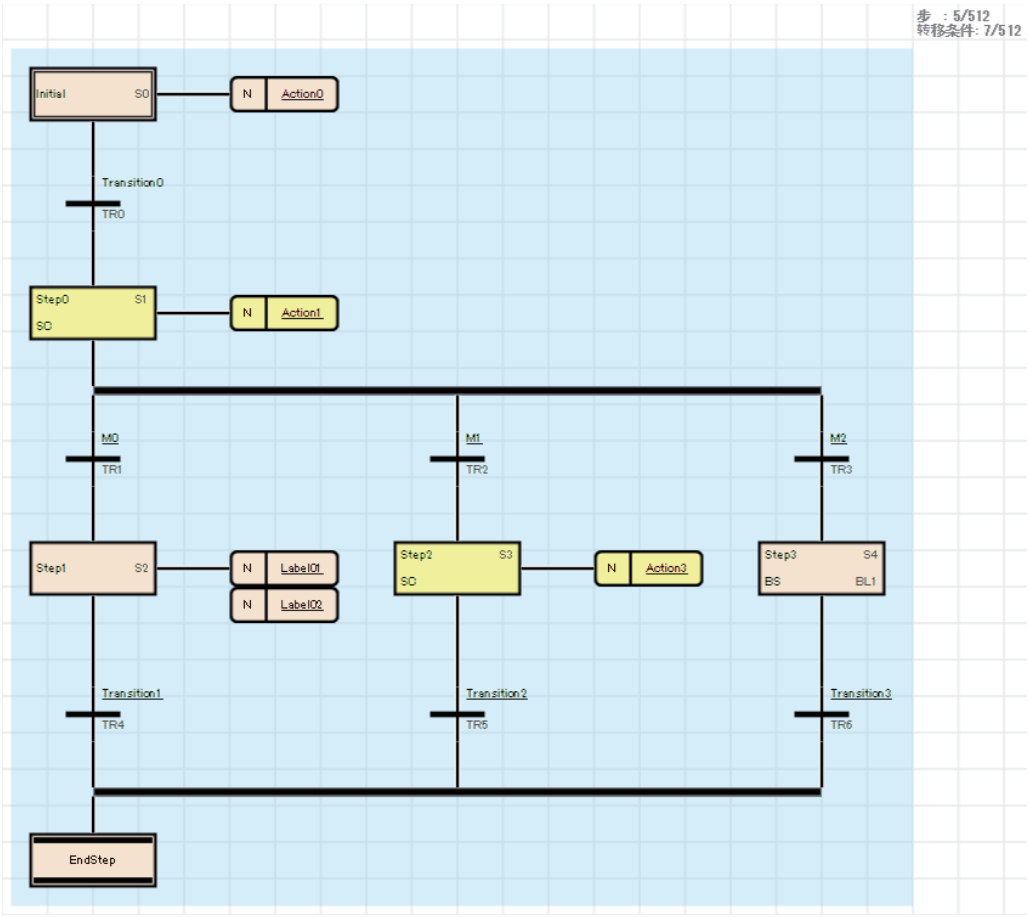
使用ST语言的情况下，请参阅下述内容。
☞43页 ST语言

■FBD/LD语言



使用FBD/LD语言的情况下，请参阅下述内容。
☞55页 FBD/LD语言

■SFC程序



使用SFC程序的情况下，请参阅下述章节。

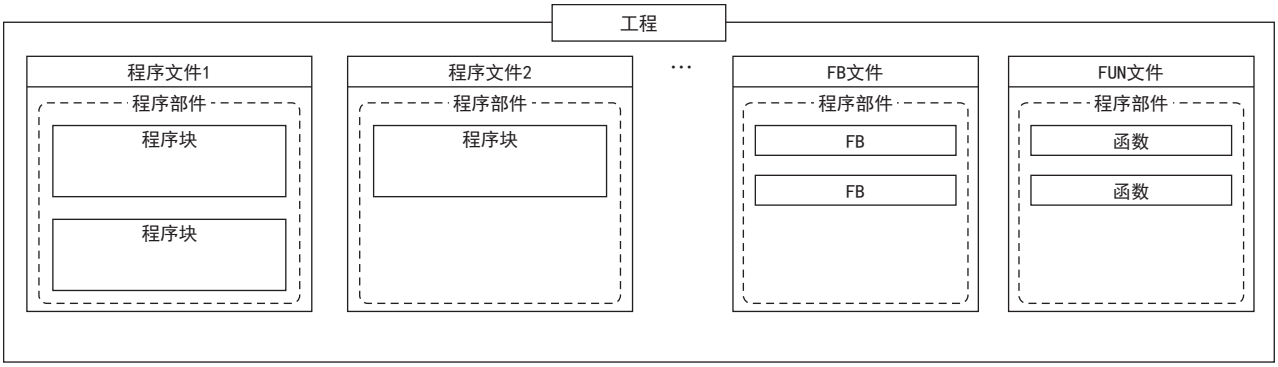
☞63页 SFC程序

要点

- 梯形图语言和FBD/LD语言适合具有顺控程序控制和逻辑回路的相关知识和经验的客户。
- ST语言适合具有C语言等编程知识及经验的用户。
- SFC程序适用于按机械的实际控制划分程序，对作业的切换进行管理的情况。
- 通过在程序中使用标签，可以提高程序的可读性，将程序简单地转变至模块配置不同的系统中。

2 程序配置

工程工具中可以创建多个程序及多个程序部件。
因此，可以根据处理划分程序及程序部件。
本章介绍程序配置有关内容。



关于程序部件，请参阅下述章节。
☞8页 程序部件

工程

工程是在CPU模块中执行的数据(程序、参数等)的集合。
每一个CPU模块中只可写入一个工程。
工程中需要创建一个或其以上的程序文件。

程序文件

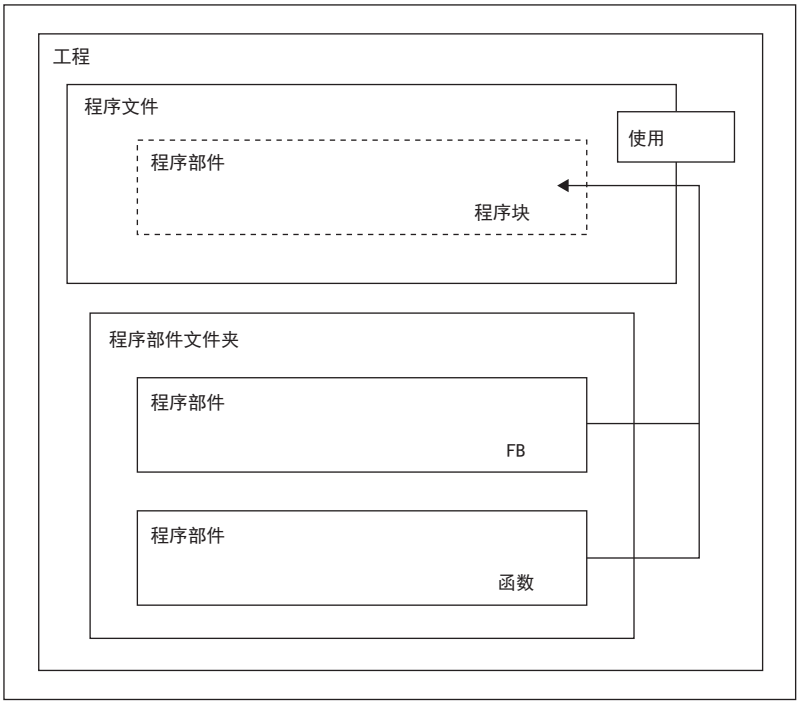
程序文件是程序及程序部件的集合。
程序文件由一个或其以上的程序块构成。(☞9页 程序块)
通过操作程序文件单位，可以将程序的执行类型自恒定周期执行类型切换为待机类型，且可对是否将数据写入至CPU模块中进行更改。

3 程序部件

程序部件有下述几种类型。

- 程序块
- 函数
- 功能块

各程序部件可以通过符合控制的程序语言记述处理。在函数及FB中，可以通过梯形图语言、ST语言或FBD/LD语言记述处理。
从程序块调用函数及FB后执行。



要点

程序的部件化是指按照各处理内容及功能，将程序阶层化时的低位处理分为若干单位，创建各单位的程序。
通过将程序部件化提高独立性，可以设计出更容易添加及更换的程序。
对以下处理进行部件化较好。

- 在程序中重复被记述的处理
- 作为一个功能可被分开的处理

本项使用标签对各程序部件进行说明。
各程序部件的程序本体(工作表)中也可以使用软元件。关于软元件的详细内容，请参阅下述手册。
MELSEC iQ-F FX5用户手册(应用篇)

要点

ST语言与FBD/LD语言中，一个程序部件内最多可以创建32个工作表。
多个工作表的执行顺序，从工程工具的“工作表执行顺序设置”画面设置。(GX Works3操作手册)

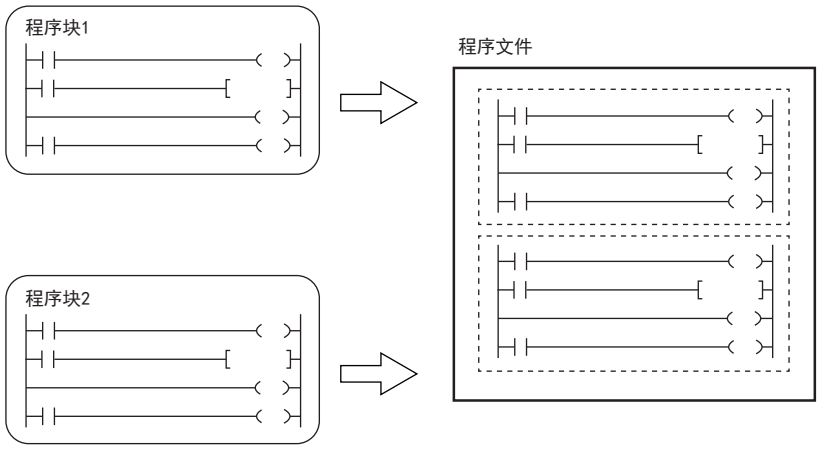
3.1 程序块

程序块为构成程序的单位。

在程序文件内可以创建多个程序块，并按照程序文件设置中指定的顺序执行。程序文件设置中未指定顺序的情况下，按照程序块名的顺序(升序)执行。

如果对各功能及处理划分程序块，可以让程序的顺序更改及更换变得更容易。

将程序块的程序本体存储到各登录目标程序中的程序文件中。



程序块的分割

可分别对各程序块创建主程序、子程序、中断程序。

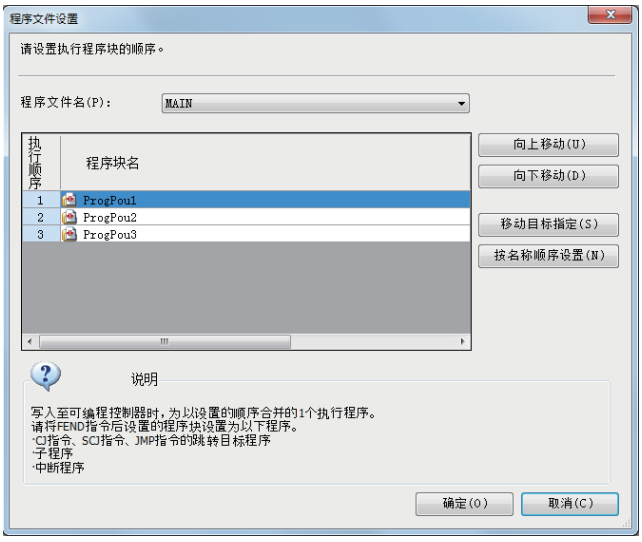
类型	内容
主程序	是从程序的步0开始到FEND指令为止的程序。
子程序	是从指针(P)开始到RET指令为止的程序。 只在通过子程序调用指令(CALL指令、XCALL指令)调用的情况下执行。
中断程序	是从中断指针(I)开始到IRET指令为止的程序。 如果发生中断原因，执行与该中断指针编号相对应的中断程序。

■程序文件设置

在程序文件设置，可以设置程序文件内的程序块执行顺序。

[转换]⇨[程序文件设置]

[导航窗口]⇨选择程序文件，右键单击⇨[程序文件设置]



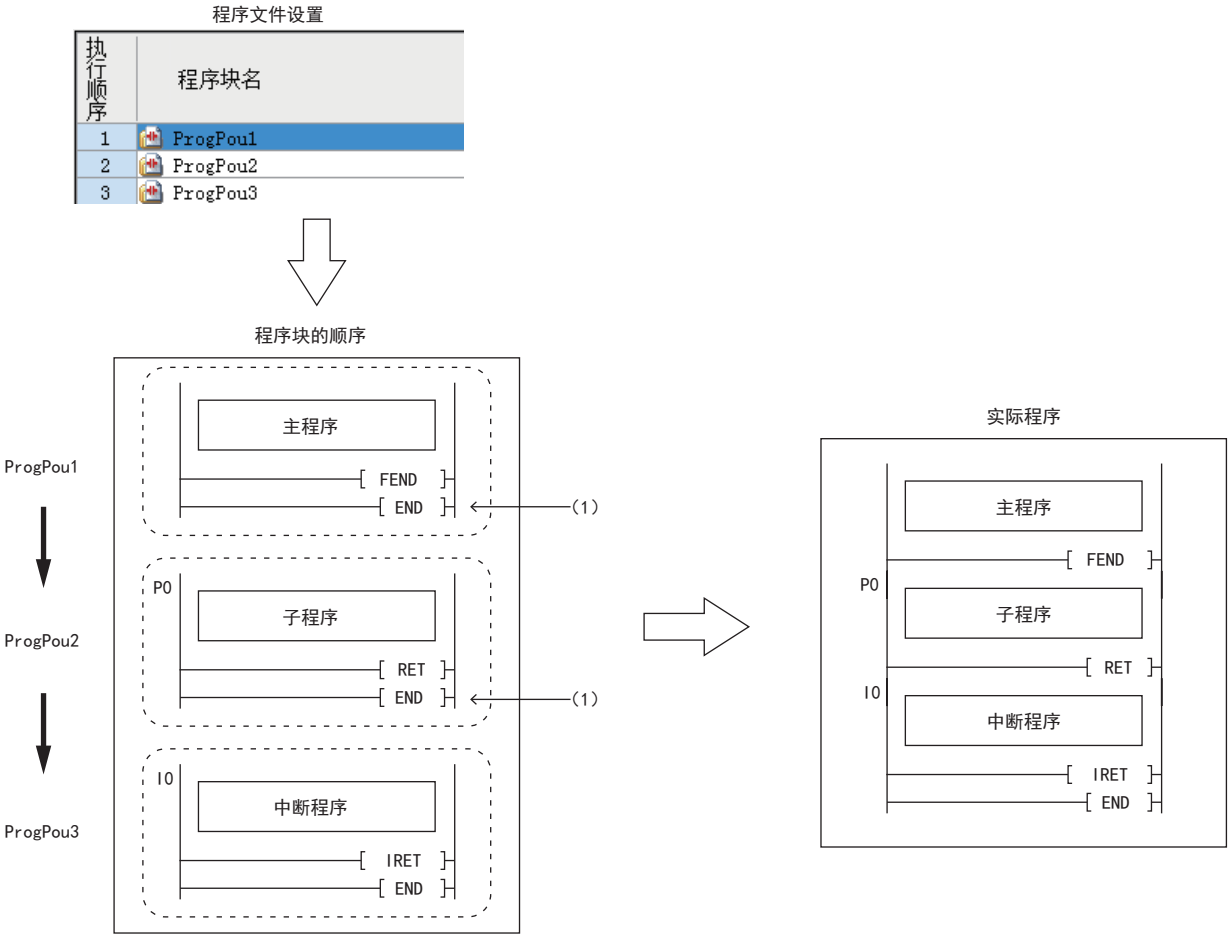
关于程序文件设置的详细内容，请参阅下述手册。

GX Works3操作手册

例

创建程序块方式如下所示。

根据程序文件设置的执行顺序执行程序。



(1) 程序文件中间的END指令被忽略。

要点

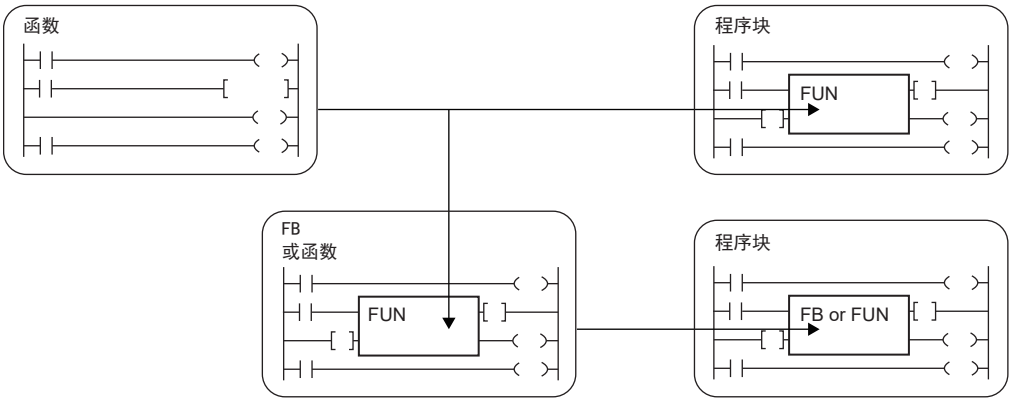
- 子程序以及中断程序是在FEND指令以后进行创建。FEND指令及其以后的程序不作为主程序执行。例如，在第二个程序块的最后使用了FEND指令的情况下，第三个程序块以后将变为子程序或中断程序。(☞ 25页 使用子程序/中断程序的情况)
- 为了创建易懂的程序，应在一个程序块中使用成对的FOR指令与NEXT指令、MC指令与MCR指令。
- 简单程序的情况下，在一个程序块内仅记述主程序便可在CPU模块中执行。

关于子程序、中断程序的详细内容，请参阅下述手册。

MELSEC iQ-F FX5用户手册(应用篇)

3.2 函数(FUN)

函数是在程序块、FB以及其它的功能中使用的程序部件。
函数执行完成后将值交接至调用源。该值称为返回值。
对于同样的输入，函数将作为处理结果始终输出相同的返回值。
如果预先定义经常使用的单纯独立的程序算法，可以有效地再利用。

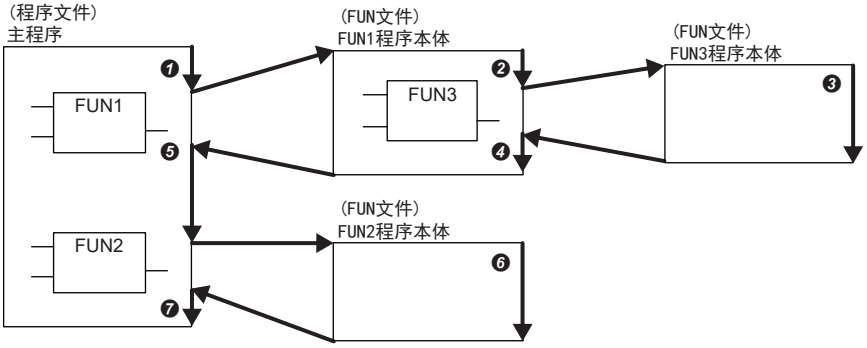


动作概要

将程序本体存储在FUN文件内，通过从调用源程序调用FUN文件内的程序本体来执行函数。

例

从主程序调用FUN1及FUN2，且FUN1又调用FUN3的情况(嵌套数：2次)
❶～❼表示执行的流程(顺序)。

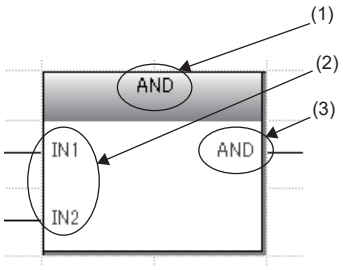


子程序类型FB、宏类型FB、函数全部合计可进行32次嵌套。

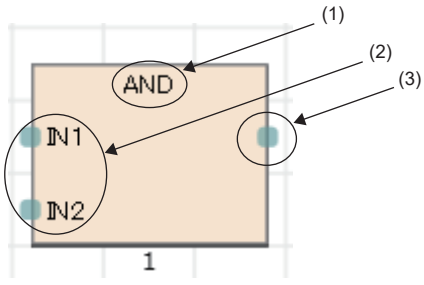
输入变量与输出变量

函数中可以定义输入变量与输出变量。输出变量可以分配与返回值不同的其它输出数据。

梯形图语言的情况



FBD/LD语言的情况



(1) 函数名
(2) 输入变量
(3) 输出变量

不显示函数的返回值名称。

在VAR_INPUT的分类中设置输入变量，在VAR_OUTPUT的分类中设置输出变量。

要点

在函数中定义的变量于每次调用函数时被覆盖。
每次调用时希望保持变量值的情况下，应通过使用FB或将输出变量保存为不同的变量等进行编程。

EN/ENO

通过在函数中附加EN(允许输入)、ENO(允许输出)，可以控制执行处理。

- 在EN中设置作为函数执行条件的布尔型变量。
- 带EN的函数只在EN的执行条件为TRUE的情况下执行。
- 在ENO中设置输出函数执行结果的布尔型变量。

EN状态的ENO与运算结果的内容如下所示。

EN	ENO	运算结果
TRUE(运算执行)	TRUE	运算输出值
FALSE(运算停止)	FALSE	不定值

要点

- 在梯形图语言、FBD/LD语言的程序中，不需要设置至ENO的输出标签。
- 在通用函数中使用EN/ENO的情况下，带EN的函数将变为“函数名_E”。

创建程序

创建函数程序的情况下，执行下述操作。

[导航窗口]⇨[FB/FUN]⇨右击⇨[新建数据]
在“基本设置”的“数据类型”中选择“函数”

创建的程序存储在FUN文件中。

[CPU参数]⇨[程序设置]⇨[FB/FUN文件设置]

1个FUN文件中最多可以存储64个创建的程序。

无法在函数内使用上升沿/下降沿执行指令。

创建程序有关内容，请参阅下述手册。

项目	参照
函数的创建方法	GX Works3操作手册
可写入至CPU模块的FB/FUN文件数	MELSEC iQ-F FX5S/FX5UJ/FX5U/FX5UC用户手册(硬件篇)

■可使用的软元件/标签

函数程序中可使用的软元件及标签一览如下所示。

○：可以使用， △：只可在指令中使用(作为表示程序的步的标签时，禁止使用)， ×：禁止使用

软元件/标签的类型		能否使用
标签(指针型以外)	全局标签	×
	局部标签	○*1
标签(指针型)	指针型全局标签	△
	指针型局部标签	○
软元件	全局软元件	○
指针	全局指针	△

*1 不能使用下述数据类型。
定时器、累计定时器、计数器、长计数器

要点

函数的返回值，可以通过在函数内将函数名作为标签编程进行设置。函数名不需要作为标签进行设置。在函数的属性中，可以使用“返回值的类型”中设置的数据类型。

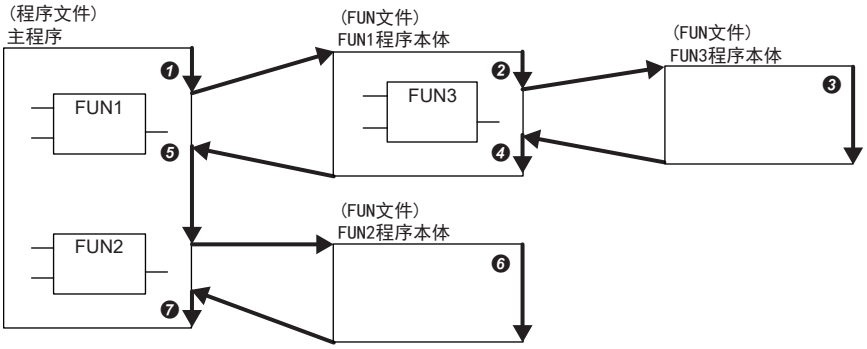
函数中定义的标签

对于在函数中定义的标签分配目标，在执行函数时将确保到存储器内的临时区域(临时工作区)中，并在执行完成时解除。

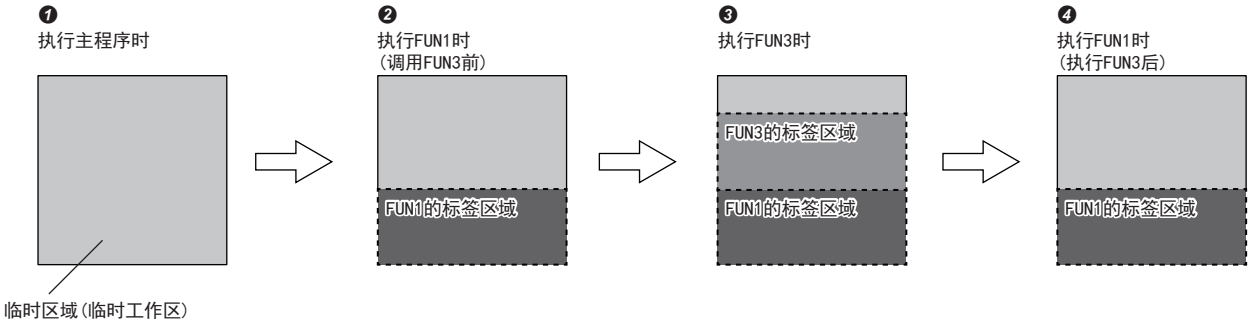
例

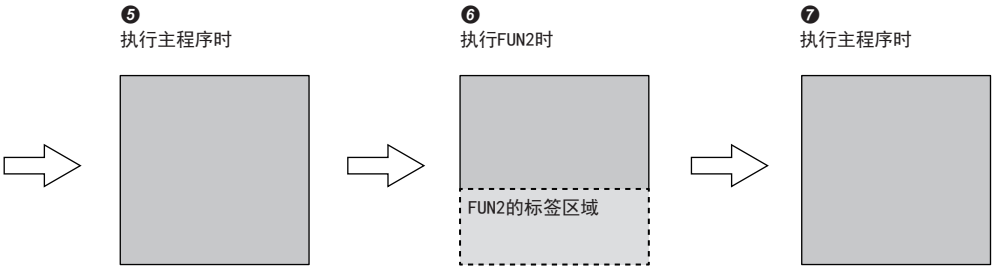
从主程序调用FUN1及FUN2，且FUN1又调用FUN3的情况

(❶～❼表示执行的流程(顺序))



对于上述函数执行动作的标签分配状态如下所示。





函数中可定义的标签的分类为VAR、VAR_CONSTANT、VAR_INPUT、VAR_OUTPUT。

要点

由于在函数中定义的标签变为不定值，最初访问时需要通过程序进行初始化。

步数

调用函数的情况下，除了程序本体的步数，还需要进行交接自变量及返回值的处理、调用程序本体时所需的步数。

■程序本体

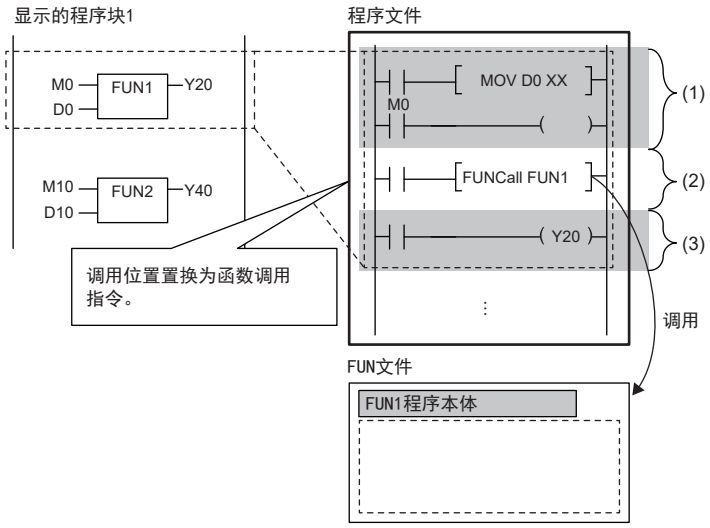
使用功能的程序本体的步数，为所使用系统的指令步数合计加上至少13步的值。关于各指令的步数，请参阅下述手册。

MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)

■调用侧

调用函数的情况下，在函数调用前后生成函数的自变量以及返回值的交接处理。

(1) 自变量交接
(2) FUN1程序本体调用
(3) 返回值交接



• 自变量交接

在自变量交接中使用的指令，根据自变量的分类及自变量的数据类型而不同。在自变量交接中使用的指令如下所示。

自变量的分类	数据类型	使用指令	步数
VAR_INPUT	位	LD+OUT LD+MOVB (根据所使用的程序语言、函数的类型、输入自变量类型的组合，使用其中的一个。)	各指令步数的详细内容，请参阅下述手册。 MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)
	字[无符号]/位列[16位] 双字[无符号]/位列[32位] 字[带符号] 双字[带符号]	LD+MOV LD+DMOV	
	单精度实数	LD+EMOV	
	时间	LD+DMOV	
	字符串(32)	LD+$MOV	
	字符串[Unicode](32)	LD+$MOV_WS	
	数组、结构体	LD+BMOV	

• 程序本体调用

函数的程序本体调用所需步数如下所示。

项目	步数
有EN	10
无EN	12

• 返回值交接

在返回值交接中使用的指令及步数与自变量交接时相同。

自变量的分类	数据类型	使用指令	步数
VAR_OUTPUT	与自变量交接相同	与自变量交接相同	与自变量交接相同

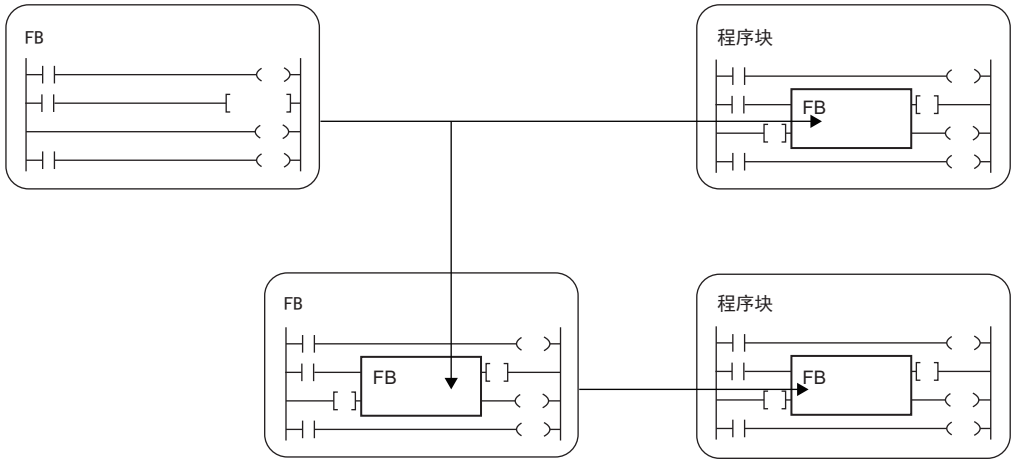
• EN/ENO

EN/ENO所需步数如下所示。

项目	步数
EN	4～7 (根据EN的输入源中指定的软元件种类和编号等，程序的内容有所不同。)
ENO	6～10 (根据ENO的输出目标中指定的软元件种类和编号等，程序的内容有所不同。)

3.3 功能块(FB)

FB是在程序块及其它的FB中使用的程序部件。

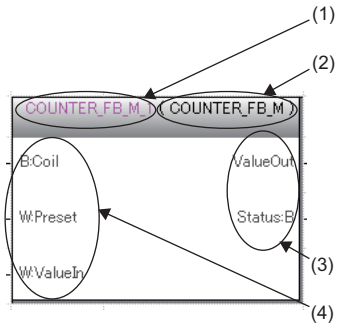


FB与函数不同，不能保持返回值。

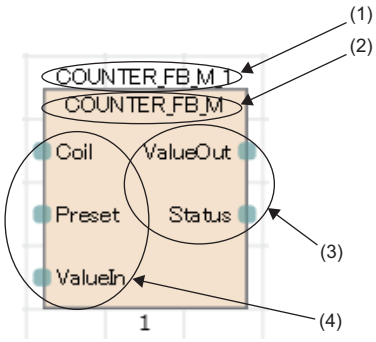
FB能将值保存在变量中，因此也能保持输入状态及处理结果。

在下一次处理中使用保持后的值，因此即使为相同的输入值也不一定每次都输出相同的结果。

梯形图语言的情况



FBD/LD语言的情况



(1) 实例名
(2) FB名
(3) 输出变量
(4) 输入变量

此外，为了在程序上使用FB，需要定义实例。

☞19页 实例

要点

- 关于通用FB的详细内容，请参阅下述手册。

📖MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)

- 关于模块FB的详细内容，请参阅下述手册。

📖所使用的模块的FB的参考

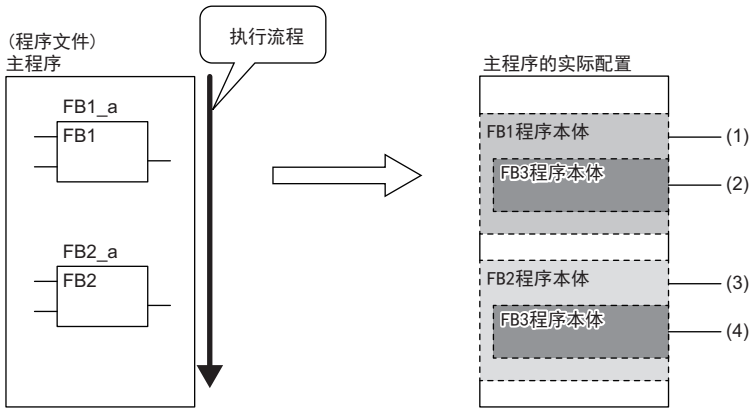
动作概要

■宏类型FB

编程时，宏类型FB对调用源程序展开调用对象的程序本体。执行时，与普通的程序同样执行展开的程序。
希望优先进行程序的处理速度时，应使用宏类型FB。

例

从主程序调用FB1_a及FB2_a，且又从FB1_a调用FB3_a，从FB2_a调用FB3_b的情况



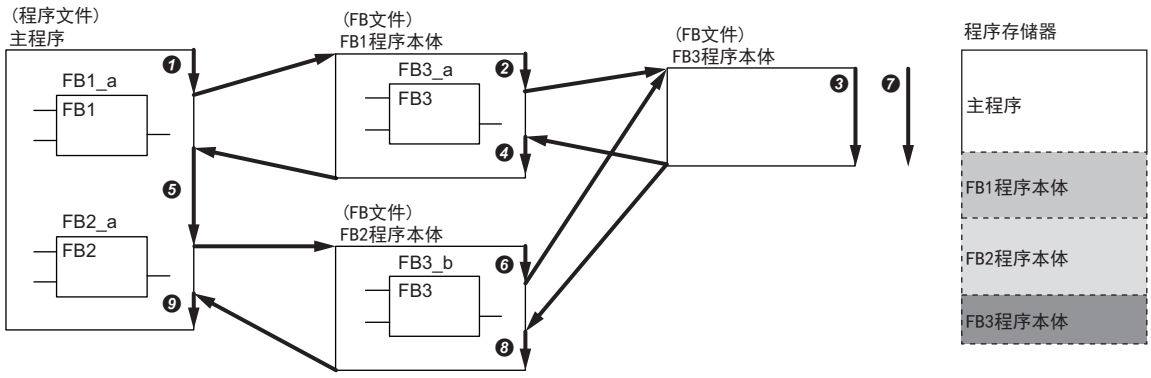
(1) 展开主程序中的FB1程序本体后执行。
(2) 从FB1调用的FB3将在FB1的程序本体内展开。
(3) FB2与FB1同样，FB2程序本体在主程序中展开后执行。
(4) 从FB2调用的FB3将在FB2的程序本体内展开。

■子程序类型FB

将程序本体存储在FB文件中，执行时从调用源程序调用FB文件内的程序本体后再执行子程序类型FB。
希望减少程序容量的情况下，应选择子程序类型FB。

例

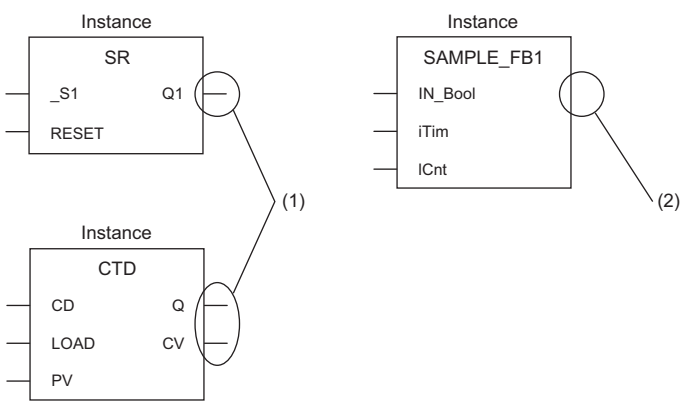
从主程序调用FB1_a及FB2_a，且又从FB1_a调用FB3_a，从FB2_a调用FB3_b的情况(嵌套数：3次)
❶～❾表示执行的流程(顺序)。



子程序类型FB、宏类型FB、函数全部合计可进行32次嵌套。

输入变量、输出变量、输入输出变量

FB中需要定义输入变量、输出变量、输入输出变量。
FB可以输出多个运算结果，也可以不输出。



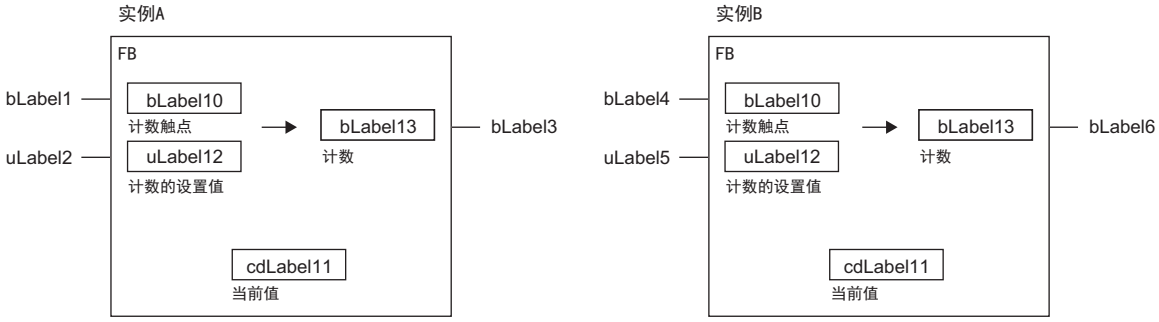
(1) 有输出
(2) 无输出

在VAR_INPUT的分类中设置输入变量，在VAR_OUTPUT及VAR_OUTPUT_RETAIN的分类中设置输出变量，在VAR_IN_OUT的分类中设置输入输出变量。

内部变量

FB使用内部变量。内部变量按FB的实例将标签分配到不同的区域中。即使是同样的标签名，各实例可保持不同的状态。

例



输入变量变为ON时开始计数，当内部变量中保持的当前值达到设置值时，将输出变量置为ON的FB。即使是同一个FB，实例A与实例B保持着各自独自的状态，因此输出的时机有所不同。
在VAR、VAR_CONSTANT、VAR_RETAIN的分类中设置内部变量。

外部变量及公开变量

FB可以使用外部变量(全局标签)及公开变量。
在VAR_PUBLIC、VAR_PUBLIC_RETAIN的分类中设置公开变量。

实例

■实例含义

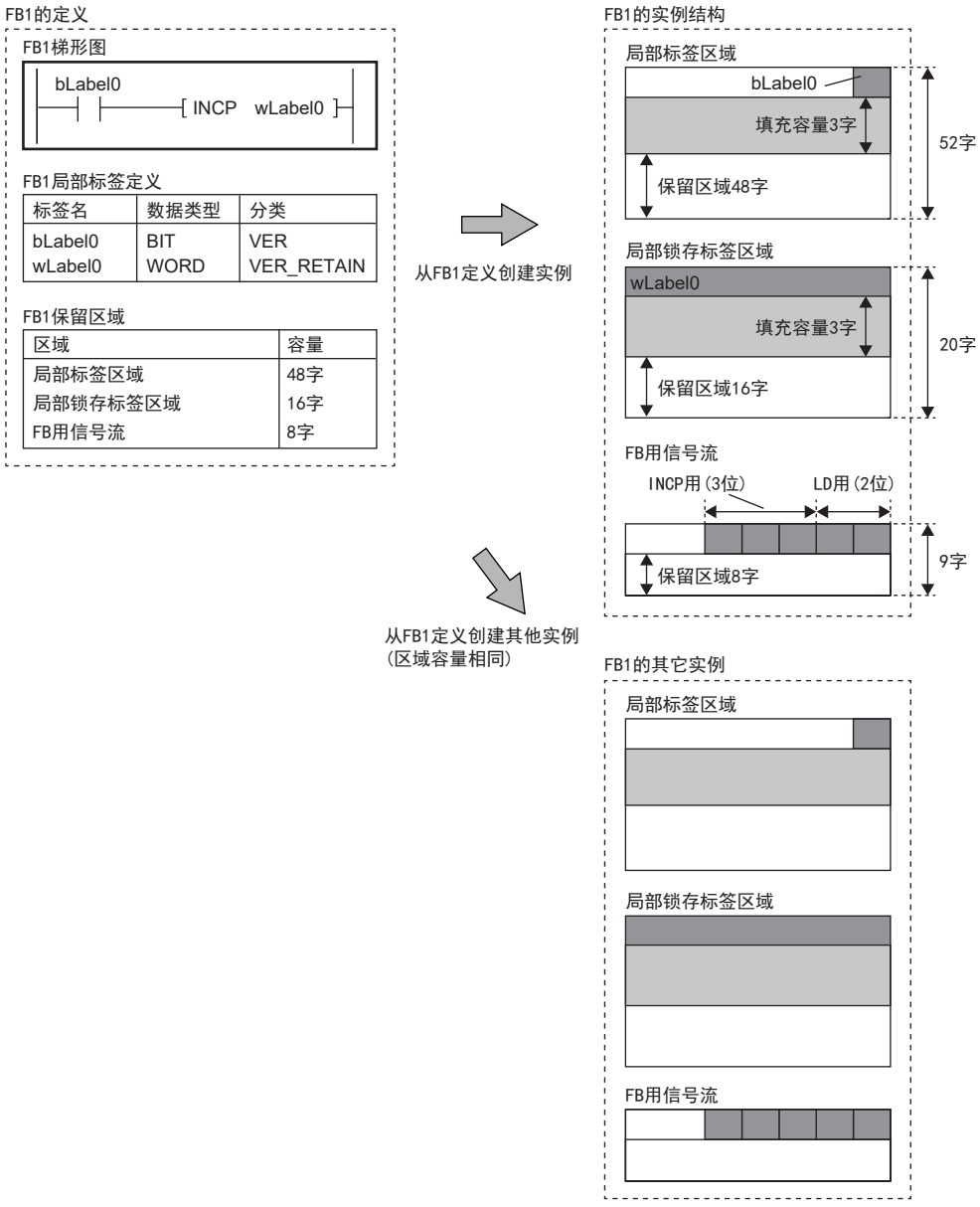
FB的实例是指，在FB定义的基础上分配的标签。可以从一个FB定义创建多个实例。

实例的配置如下。

项目	内容
局部标签区域	分配FB的局部标签的区域。
局部锁存标签区域	对FB的锁存属性的局部标签进行分配的区域。
信号流区域(FB用信号流)	对FB定义内的指令所使用的信号流进行分配的区域。

例

实例的配置(子程序类型FB的示例)



局部标签区域及局部锁存标签区域，确保4字单位的标签使用区域，因此上述的示例已确保3字(填充容量)。

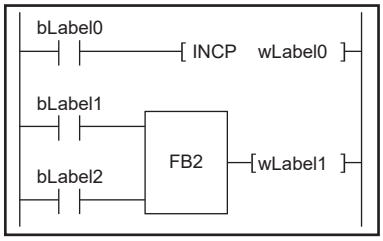
各区域确保保留区域。保留区域是指在转换或RUN中写入等时维持标签的分配不变，局部标签、指令、FB的实例进行添加/更改的区域。无法确保添加/更改对象的数据类型所对应的区域的情况下，需要进行全部转换(重新分配)。

例

对FB进行了嵌套的实例的配置

FB1的定义

FB1梯形图



FB1局部标签定义

标签名	数据类型	分类
bLabel0	BIT	VER
bLabel1	BIT	VER
bLabel2	BIT	VER
wLabel0	WORD	VER_RETAIN
wLabel1	WORD	VER
FB2_a	FB2	VER

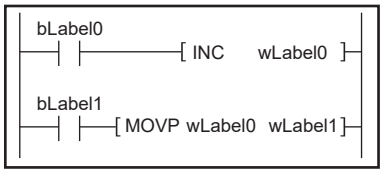
FB1保留区域

区域	容量
局部标签区域	48字
局部锁存标签区域	16字
FB用信号流	8字

从FB1定义创建实例

FB2的定义

FB2梯形图



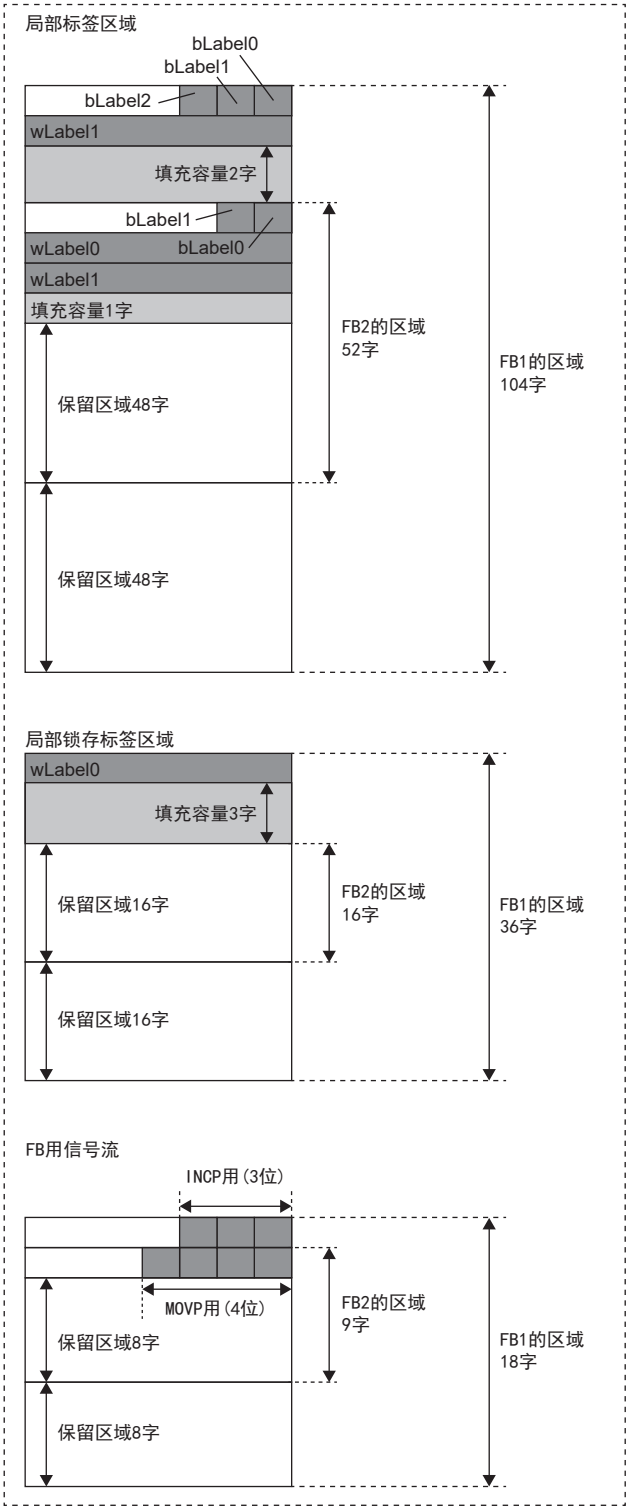
FB2局部标签定义

标签名	数据类型	分类
bLabel0	BIT	VER_INPUT
bLabel1	BIT	VER_INPUT
wLabel0	WORD	VER
wLabel1	WORD	VER_OUTPUT

FB2保留区域

区域	容量
局部标签区域	48字
局部锁存标签区域	16字
FB用信号流	8字

FB1的实例结构



作为局部标签被声明的FB2的实例，将确保到声明源的FB1的局部标签区域、局部锁存标签区域、FB用信号流中。

按上述示例将FB类型的局部标签添加/更改到FB1的情况下，保留区域的容量在局部标签区域时为48字、局部锁存标签区域时为16字、FB用信号流为8字，因此只要添加/更改的FB所附带的区域超过上述限制，即使只有1个，也需要进行全部转换(重新分配)。

■创建实例

为了使用FB，需要创建实例。

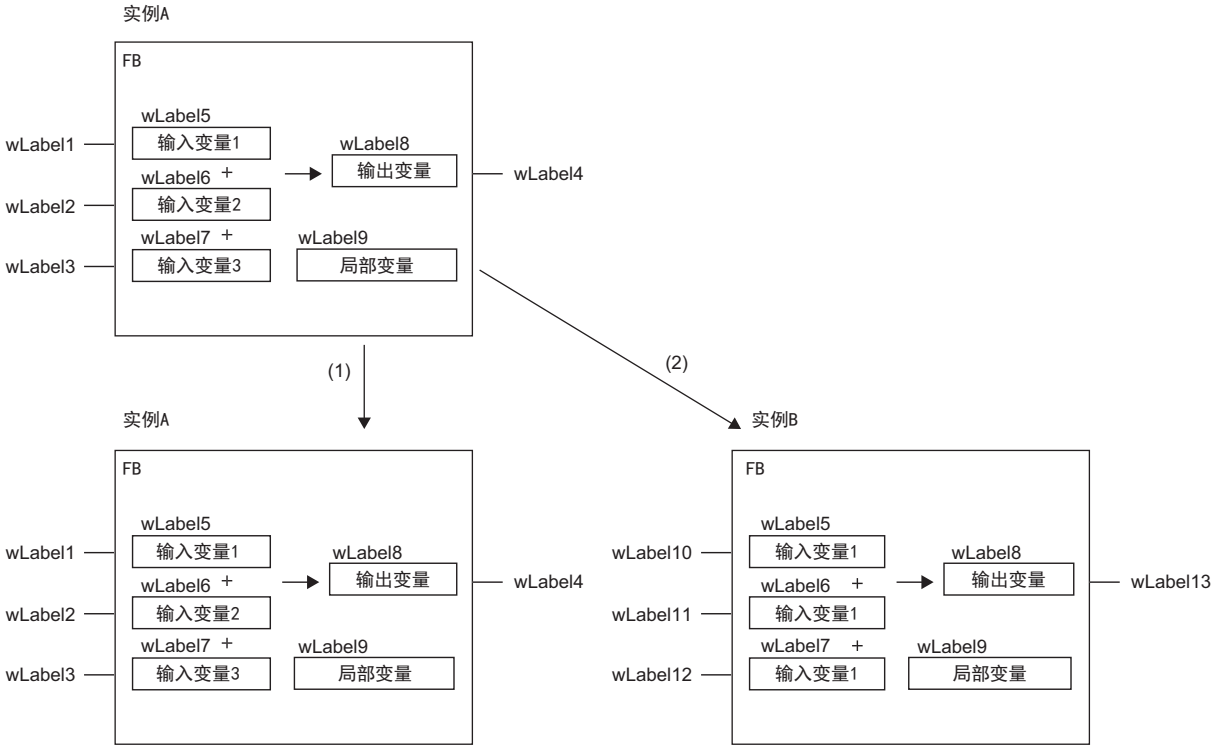
通过创建FB的实例，可以从程序块及其他的FB调用使用。

在全局标签或局部标签中声明实例。

标签的类型	实例的类型	分类
全局标签	全局FB	VAR_GLOBAL
局部标签*1	局部FB	VAR

*1　可以作为程序块或FB的局部标签进行声明。在函数中无法声明。

在一个程序部件中，同一个FB可以在不同的实例中使用。



(1) 相同实例的情况下，使用相同的内部变量。

(2) 不同实例的情况下，使用不同区域的内部变量。

■实例的容量

关于实例的各数据区域的容量，计算方法如下所示。

• 局部标签区域的容量

实例的局部标签区域的容量 = 锁存属性以外的局部标签的数据容量(总和) + 保留区域容量

项目	内容
局部标签容量(锁存属性的局部标签除外)	作为局部标签使用的数据的总和。 实际使用的区域的容量，根据标签的存储器分配而有所不同。标签的存储器分配的详细内容，请参阅下述手册。 📖GX Works3操作手册
保留区域容量	48字。

• 局部锁存标签区域的容量

实例的局部锁存标签区域的容量 = 锁存属性的局部标签的数据容量(总和) + 保留区域容量

项目	内容
锁存属性局部标签容量	作为锁存属性的局部标签使用的数据的总和。 实际使用的区域的容量，根据标签的存储器分配而有所不同。标签的存储器分配的详细内容，请参阅下述手册。 📖GX Works3操作手册
保留区域容量	16字。

• FB用信号流的容量

宏类型FB的情况下，与程序的步数同点数。

子程序类型FB的情况如下所示。

FB用信号流容量(字) = FB的程序步数 / 16 + 保留区域容量

项目	内容
FB用信号流容量	FB定义内的指令所使用的FB用信号流的总和。
保留区域容量	8字。

要点

对于RUN中写入时添加/更改的容量，如果不能确保预留区域容量，将无法进行RUN中写入，需要进行全部转换(重新分配)。

EN/ENO

FB与函数一样通过附带EN(允许输入)、ENO(允许输出)，可以进行执行处理的控制。

☞12页 EN/ENO

调用附带EN/ENO的FB的实例时，必须对EN分配实自变量。

创建程序

创建FB的程序的情况下，实施下述操作。

[导航窗口]⇨[FB/FUN]⇨右击⇨[新建数据]

在“基本设置”的“数据类型”中选择“FB”

创建的程序存储在FB文件中。

[CPU参数]⇨[程序设置]⇨[FB/FUN文件设置]

1个FB文件中最多可以存储64个创建的程序。

创建程序有关内容，请参阅下述手册。

项目	参照
FB的创建方法	📖GX Works3操作手册
可写入至CPU模块的FB/FUN文件数	📖MELSEC iQ-F FX5S/FX5UJ/FX5U/FX5UC用户手册(硬件篇)

■程序的类型

FB有下述几种类型，FB程序本体的存储方式不同。

- 宏类型FB
- 子程序类型FB

详细内容，请参阅下述章节。

☞17页 动作概要

模块FB、通用函数、通用FB无法进行上述选择。

■固有属性设置

创建FB的程序的情况下，可以实施下述设置。(📖GX Works3操作手册)

项目	内容
使用MC/MCR控制EN*1	选择“是”时，使用MC/MCR指令控制EN；选择“否”时，使用CJ指令控制EN。在FB内使用了上升沿/下降沿时，应选择“是”。此外，根据选择，FB内所使用的定时器/计数器及OUT指令的动作将有所不同。详细内容，请参阅下述章节。 ☞111页 使用MC/MCR指令控制EN的动作
使用EN/ENO	选择“是”时，变为附带EN/ENO的FB，即使EN/ENO标签未登录至局部标签，也能在程序中使用。选择“否”时，变为不附带EN/ENO的FB。 关于EN/ENO的详细内容，请参阅下述章节。 ☞22页 EN/ENO

*1 对“使用EN/ENO”选择“是”时进行选择。但是，根据CPU模块及GX Works3的版本，即使对“FB的类型”选择“子程序类型”，也有可能无法进行选择。对应的CPU模块及GX Works3的版本，请参阅下述手册。
📖MELSEC iQ-F FX5用户手册(应用篇)

■可使用的软元件/标签

在FB程序中可使用的软元件及标签一览如下所示。

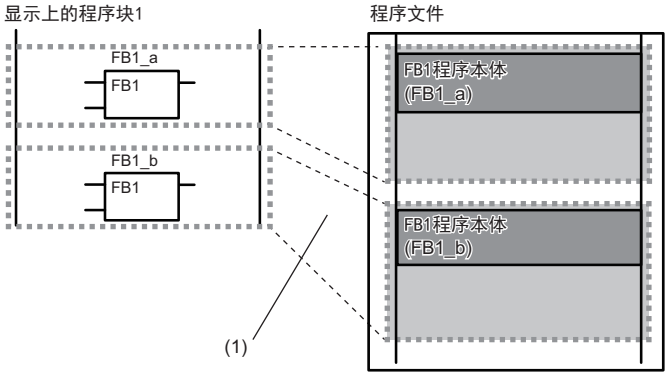
○：可以使用， △：只可在指令中使用(作为表示程序的步的标签时，禁止使用)， ×：禁止使用

软元件/标签的类型		能否使用
标签(指针型以外)	全局标签	○
	局部标签	○
标签(指针型)	指针型全局标签	△
	指针型局部标签	○
软元件	全局软元件	○
指针	全局指针	△

步数(宏类型FB)

■调用侧

调用宏类型FB的情况下，编译时展开调用对象的程序本体。



(1) 程序本体在多个调用位置展开。

■程序本体

FB程序本体的步数与普通的程序一样为指令步数的总计。

关于各指令的步数，请参阅下述手册。

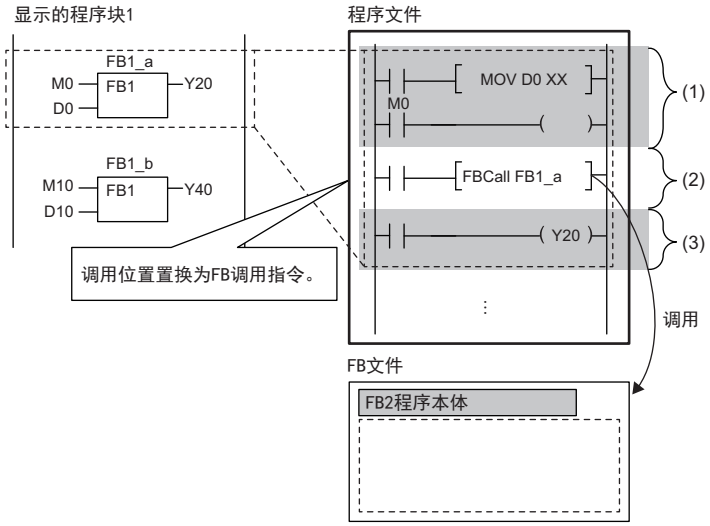
📖MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)

步数(子程序类型FB)

■调用侧

调用子程序类型FB的情况下，在FB调用前后生成FB的自变量的交接处理。

(1) 自变量交接(输入自变量、输入输出自变量)
(2) FB1程序本体调用
(3) 自变量交接(输出自变量、输入输出自变量)



• 自变量交接

在自变量交接中使用的指令，根据自变量的分类及自变量的数据类型而不同。在自变量交接中使用的指令如下所示。

自变量的分类	数据类型	使用指令	步数
VAR_INPUT VAR_IN_OUT VAR_OUTPUT	位	LD+OUT LD+MOVB (根据所使用的程序语言、函数的类型、输入自变量类型的组合，使用其中的一个。)	各指令步数的详细内容，请参阅下述手册。 MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)
	字[无符号]/位列[16位] 双字[无符号]/位列[32位] 字[带符号] 双字[带符号]	LD+MOV LD+DMOV	
	单精度实数	LD+EMOV	
	时间	LD+DMOV	
	字符串(32)	LD+$MOV	
	字符串[Unicode](32)	LD+$MOV_WS	
	数组、结构体	LD+BMOV	

• 程序本体调用

FB的程序本体调用所需步数如下所示。

项目	步数
有EN	10
无EN	12

• EN/ENO

EN/ENO所需步数如下所示。

项目	步数
EN	4～7 (根据EN的输入源中指定的软元件种类和编号等，程序的内容有所不同。)
ENO	6～10 (根据ENO的输出目标中指定的软元件种类和编号等，程序的内容有所不同。)

■程序本体

FB程序本体的步数与普通的程序一样为指令步数的总计。

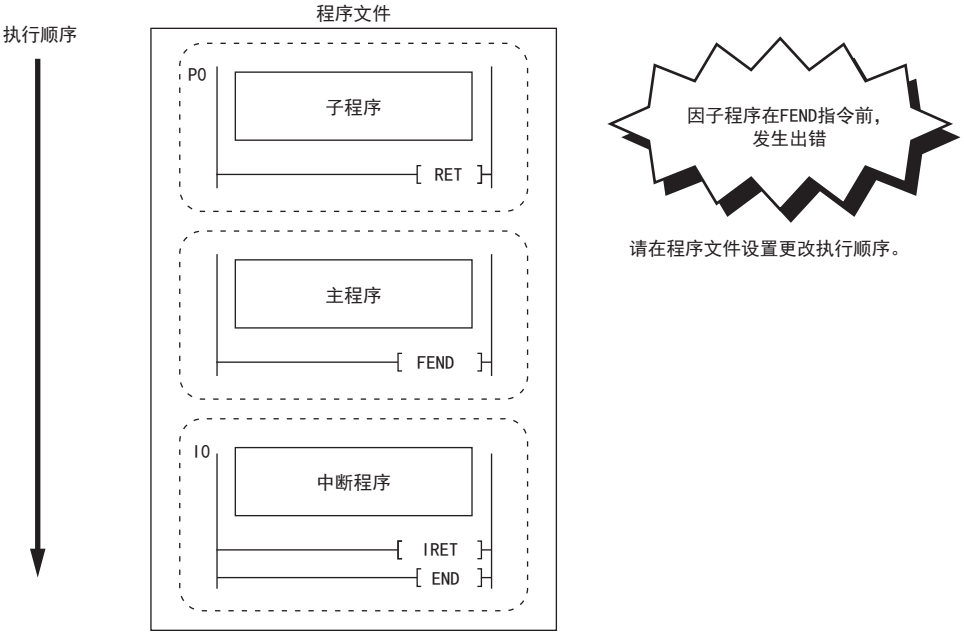
关于各指令的步数，请参阅下述手册。

MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)

3.4 注意事项

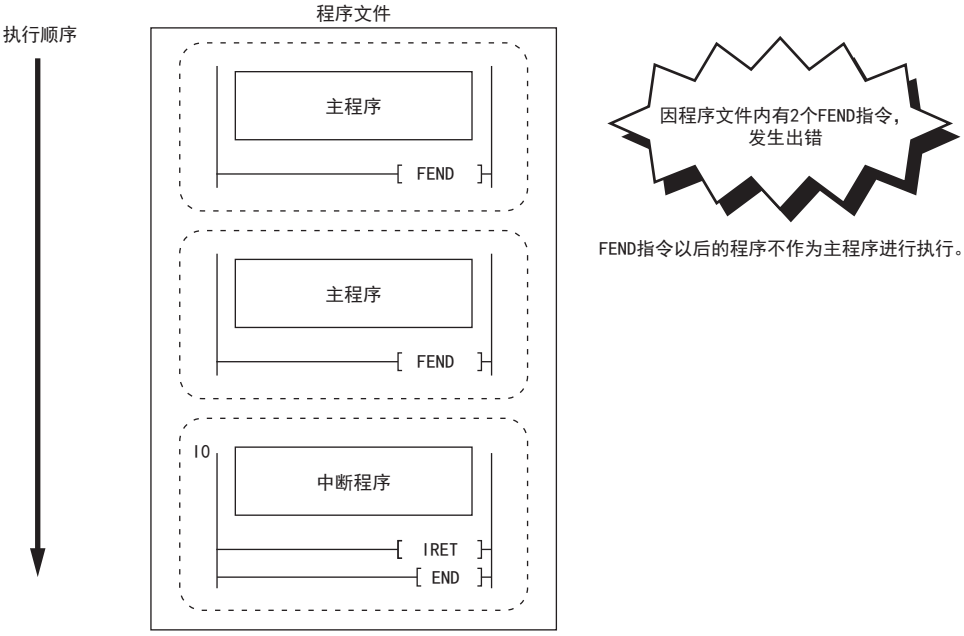
使用子程序/中断程序的情况

• 子程序以及中断程序请在FEND指令以后设置。在FEND指令前设置会发生出错。



请在程序文件设置更改执行顺序。

• 在程序文件内，只可使用1个FEND指令。使用多个会发生出错。



FEND指令以后的程序不作为主程序进行执行。

使用函数的情况

■全局指针/指针型的全局标签

全局指针、指针型的全局标签不能作为表示程序的步数的标签使用。

使用FB的情况

■全局指针/指针型的全局标签

全局指针、指针型的全局标签不能作为表示程序的步数的标签使用。

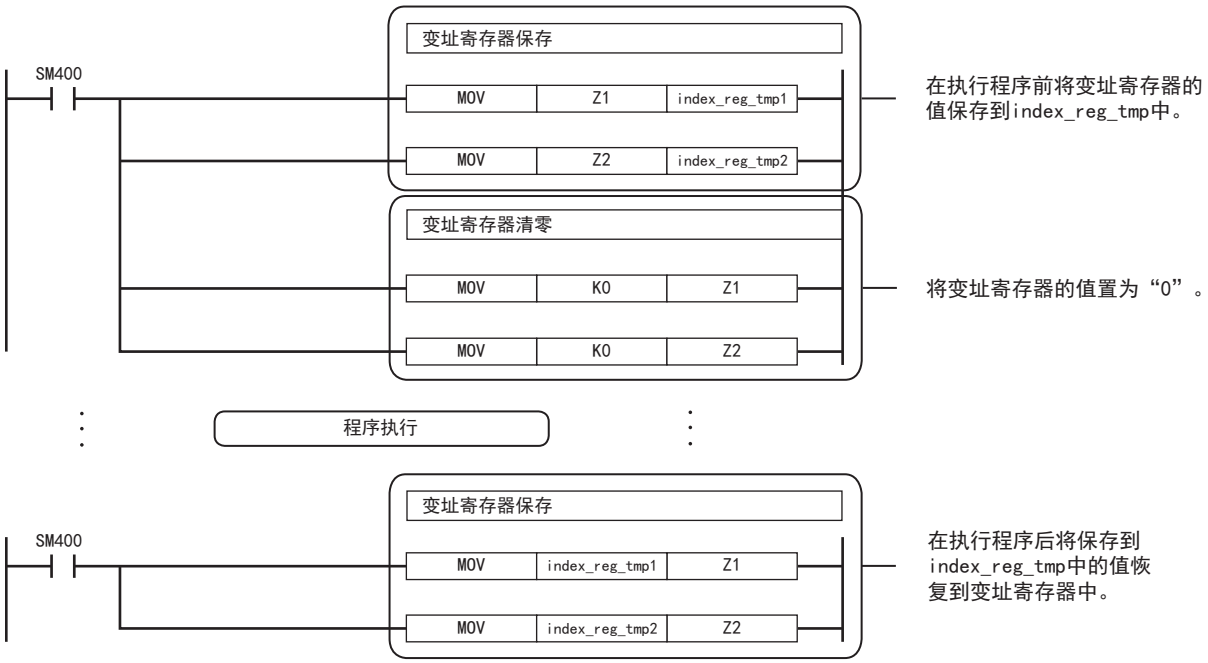
■使用变址寄存器的情况

在FB的程序中使用变址寄存器的情况下，为了保护变址寄存器的值，需要保存梯形图与恢复梯形图。

保存变址寄存器时通过将变址寄存器的值设为0，可以防止变址修饰的整合性检查(软元件编号是否超过软元件范围)的出错。

例

执行程序前保存变址寄存器Z1、Z2，在程序执行后恢复保存的变址寄存器的情况



■关于宏类型FB的自变量

在宏类型FB的程序本体以外使用时，不使用FB的自变量，而应使用用于自变量交接的软元件/标签。

例

用于自变量交接的软元件的情况

```
MacroFbPou_1(EN:= M0 ,ENO=> M1);
M2 := M1;
```

如果在宏类型FB的程序本体以外中使用，则宏类型FB的自变量可能变为意料外的值。

例

变为意料外的值的情况

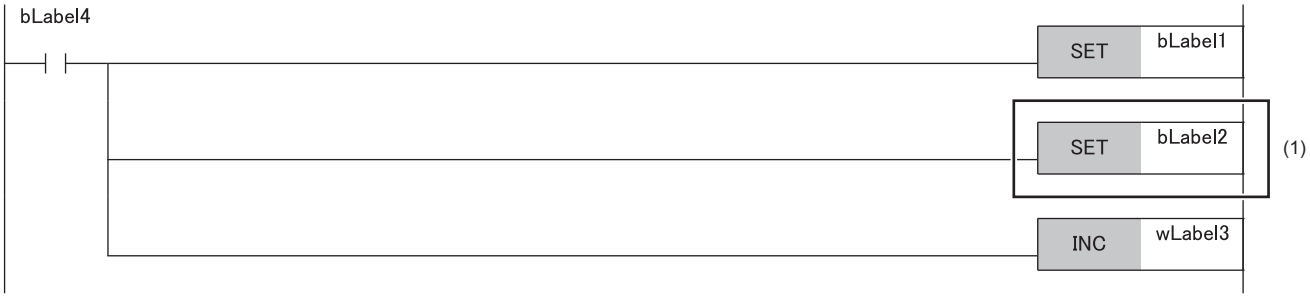
```
MacroFbPou_1(EN:= M0 ,ENO=> M1);
M2 := MacroFbPou_1.ENO;
```

■在宏类型FB的VAR_INPUT、VAR_OUTPUT或VAR_IN_OUT中发生转换出错的情况

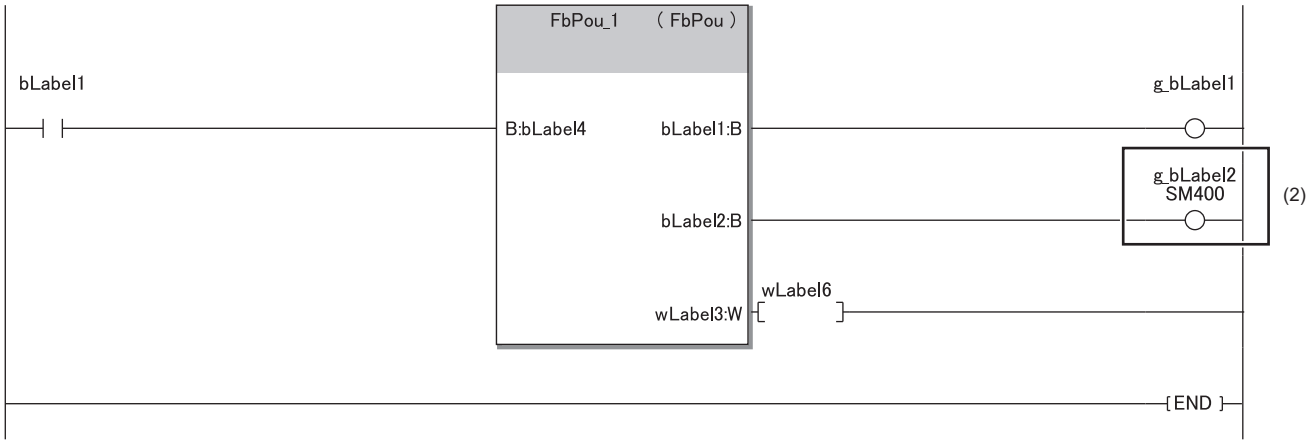
在宏类型FB内的VAR_INPUT、VAR_OUTPUT或VAR_IN_OUT中发生转换出错的情况下，出错的原因可能在于FB的调用源的程序块或FB。此时，应确认FB的调用源的程序块或FB的输入或输出。

例

在宏类型FB(FbPou)内的VAR_OUTPUT中发生了转换出错(1)的情况



(1)中没有异常的情况下，应在调用源程序块确认对应FB的输入或输出(2)。



在上述示例中，由于将FB的输出变量传到了不可写入的标签/软元件，发生了转换出错。

■模块FB的限制事项

使用模块FB的情况下，存在以下限制事项。

- 在MC指令至MCR指令之间调用模块FB的情况下，请勿将MC指令的触点置为OFF。
- 请勿进行会使得在CJ指令中无法调用模块FB的跳转。
- 在子程序内调用模块FB的情况下，每次扫描应执行1次子程序。此外，请勿在XCALL指令中执行子程序的非执行处理。
- 在中断程序或事件执行类型程序内，请勿调用模块FB。
- 在FOR～NEXT指令间、内嵌ST或ST语言的控制语句内(IF语句、FOR语句、CASE语句等)，请勿调用模块FB。

■使用主控指令时

显示主控OFF时的动作。

- 宏类型FB

FB内与触点OFF动作相同(OFF执行或非执行。)

- 子程序类型FB，FUN

FB内为非执行处理。

关于程序容量的变更(仅FX5U/FX5UC CPU模块)

在参数的程序容量设置中选择从64000步到128000步时，动作发生如下变化。

- FB用信号流区域容量从16K字节扩展为32K字节。
- 临时工作区容量从700字扩展为32767字。
- 各指令的执行时间将延长。

对固件版本低于1.100的CPU模块，请勿写入超过64000步的程序。程序无法正常工作。

关于程序容量设置，请参阅下述手册。

MELSEC iQ-F FX5用户手册(应用篇)

4 标签

标签是指在输入输出数据及内部处理中指定了任意字符串的变量。
在编程中使用标签后，在创建程序时便无需考虑软元件和缓冲存储器容量。
因此，使用了标签的程序即使在模块配置不同的系统也可以简单再利用。
在使用标签时，编程及使用的功能中的一部分需要注意。详细内容请参阅下述内容。
☞36页 注意事项

4.1 类型

本手册针对下述的标签进行说明。
- 全局标签
- 局部标签

全局标签

是在一个工程中变为相同数据的标签。可以在工程内的所有程序中使用。
在程序中可以通过程序块与函数块使用。
在全局标签的设置中进行标签名、分类、数据类型及软元件的关联。

■软元件的分配

全局标签可以分配任意的软元件。

项目	内容
不分配软元件的标签	• 编程时无需考虑软元件。 • 定义的标签被配置到软元件/标签存储器中的标签区域或锁存标签区域。
分配软元件的标签	• 对于在输入及输出等中使用的软元件，希望作为标签进行编程的情况下，可以直接分配软元件。 • 定义的标签被配置到软元件/标签存储器内的软元件区域中。

局部标签

只能在各程序部件中使用的标签。不可以使用程序部件外部的局部标签。
在局部标签的设置中进行标签名、分类与数据类型的设置。

要点

作为标签类型，全局标签与局部标签以外有下述几种类型。

■系统标签

iQ Works对应产品中为可共享的标签，通过MELSOFT Navigator进行管理。预先将全局标签作为系统标签进行登录后，能够使用系统标签通过显示器进行监视或数据访问。

关于详细内容，请参阅以下内容。

📖开始吧iQ Works

■模块标签

是各模块固有定义的标签。在工程工具上使用模块时会自动生成，并且能够在程序中用作全局标签。

关于详细内容，请参阅以下内容。

📖MELSEC iQ-F FX5 CPU模块FB参考

关于模块标签的登录，请参阅下述内容。

📖GX Works3操作手册

4.2 分类

标签的分类显示标签在哪个程序部件以及怎样使用。
根据程序部件的类型，可选择的分类也不同。

全局标签				
分类	内容	可使用的程序部件		
		程序块	FB	函数
VAR_GLOBAL	是可以在程序块与FB中使用的通用标签。	○	○	×
VAR_GLOBAL_CONSTANT	是可以在程序块与FB中使用的通用常数。	○	○	×
VAR_GLOBAL_RETAIN	是可以在程序块与FB中使用的锁存类型的标签。	○	○	×

局部标签				
分类	内容	可使用的程序部件		
		程序块	FB	函数
VAR	是在声明的程序部件的范围内使用的标签。 不可以在其他程序部件中使用。	○	○	○
VAR_CONSTANT	是在声明的程序部件的范围内使用的常数。 不可以在其他程序部件中使用。	○	○	○
VAR_RETAIN	是在声明的程序部件的范围内使用的锁存类型的标签。不可以在其他程序部件中使用。	○	○	×
VAR_INPUT	是向函数及FB中输入的标签。 是接受数值的标签，不可以在程序部件内更改。	×	○	○
VAR_OUTPUT	是从函数或FB中输出的标签。	×	○	○
VAR_OUTPUT_RETAIN	是从函数或FB中输出的锁存类型的标签。	×	○	×
VAR_IN_OUT	是接受数值并从程序部件中输出的局部标签。可以在程序部件内更改。	×	○	×
VAR_PUBLIC	是可以从其他程序部件进行访问的标签。	×	○	×
VAR_PUBLIC_RETAIN	是可以从其他程序部件进行访问的锁存类型的标签。	×	○	×

4.3 数据类型

标签的数据类型根据位长、处理方法、值的范围等进行划分。
数据类型有下述几种。

- 基本数据类型
- 总称数据类型(ANY型)

基本数据类型

基本数据类型包括如下所示的数据类型。

数据类型		内容	值的范围	位长
位	BOOL	是表示ON或OFF等二者择一的状态的类型。	0(FALSE)、1(TRUE)	1位
字[无符号]/位列[16位]	WORD	是表示16位的类型。	0～65535	16位
双字[无符号]/位列[32位]	DWORD	是表示32位的类型。	0～4294967295	32位
字[带符号]	INT	是处理正与负的整数值的类型。	-32768～+32767	16位
双字[带符号]	DINT	是处理正与负的倍精度整数值的类型。	-2147483648～+2147483647	32位
单精度实数	REAL	是处理小数点以后的数值(单精度实数值)的类型。 有效位数：7位(小数点以后6位)	-2^{128}～-2^{-126}，0，2^{-126}～2^{128}	32位
时间*1	TIME	是作为d(日)、h(时)、m(分)、s(秒)、ms(毫秒)处理数值的类型。	T#-24d20h31m23s648ms～T#24d20h31m23s647ms*2	32位
字符串(32)	STRING	是处理字符串(ASCII、移位JIS)的数据类型。	最多255个半角字符	可变
字符串[Unicode](32)	WSTRING	是处理Unicode字符串的数据类型。	最多255个字符	可变
定时器	TIMER	是与软元件的定时器(T)相对应的结构体。	☞ 30页 关于定时器与计数器的数据类型	
累计定时器	RETENTIVETIMER	是与软元件的累计定时器(ST)相对应的结构体。		
计数器	COUNTER	是与软元件的计数器(C)相对应的结构体。		
长计数器	LCOUNTER	是与软元件的长计数器(LC)相对应的结构体。		
指针	POINTER	是与软元件的指针(P)相对应的类型。(MELSEC iQ-F FX5用户手册(应用篇))		

*1 时间类型在函数的时间数据类型函数中使用。关于通用函数，请参阅下述手册。
MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)
*2 在时间类型的标签中使用常数的情况下，应在起始添加“T#”。

■关于定时器与计数器的数据类型

定时器、累计定时器、计数器、长计数器的数据类型是具有触点、线圈、当前值的结构体。

数据类型		构件名	构件的数据类型	内容	值的范围
定时器	TIMER	S	位	表示触点。是与定时器软元件的触点(TS)同样的动作。	0(FALSE)、1(TRUE)
		C	位	表示线圈。是与定时器软元件的线圈(TC)同样的动作。	0(FALSE)、1(TRUE)
		N	字[无符号]/位列[16位]	表示当前值。是与定时器软元件的当前值(TN)同样的动作。	0～32767*1
累计定时器	RETENTIVETIMER	S	位	表示触点。是与累计定时器软元件的触点(STS)同样的动作。	0(FALSE)、1(TRUE)
		C	位	表示线圈。是与累计定时器软元件的线圈(STC)同样的动作。	0(FALSE)、1(TRUE)
		N	字[无符号]/位列[16位]	表示当前值。是与累计定时器软元件的当前值(STN)同样的动作。	0～32767*1
计数器	COUNTER	S	位	表示触点。是与计数器软元件的触点(CS)同样的动作。	0(FALSE)、1(TRUE)
		C	位	表示线圈。是与计数器软元件的线圈(CC)同样的动作。	0(FALSE)、1(TRUE)
		N	字[无符号]/位列[16位]	表示当前值。是与计数器软元件的当前值(CN)同样的动作。	0～32767
长计数器	LCOUNTER	S	位	表示触点。是与长计数器软元件的触点(LCS)同样的动作。	0(FALSE)、1(TRUE)
		C	位	表示线圈。是与长计数器软元件的线圈(LCC)同样的动作。	0(FALSE)、1(TRUE)
		N	双字[无符号]/位列[32位]	表示当前值。是与长计数器软元件的当前值(LCN)同样的动作。	*2

*1 当前值的值通过指令名指定单位。
*2 利用OUT LC指令使用时:0～4294967295
利用UDCNTF指令使用时:-2147483648～+2147483647

关于各软元的动作的详细内容，请参阅下述手册。
MELSEC iQ-F FX5用户手册(应用篇)
各构件的指定方法与结构体数据类型的构件指定相同。(☞ 33页 结构体)

总称数据类型(ANY型)

是汇总若干个基本数据类型标签的数据类型。数据类型名以“ANY”开始。
在函数及FB的自变量、返回值等中允许多个数据类型的情况下，使用总称数据类型。
在总称数据类型中定义的标签，可以使用低位的数据类型的任意一种。
关于与总称数据类型的类型相对应的基本数据类型，请参阅下述手册。
MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)

可以定义的数据类型

能否设置在标签的各分类中可以定义的数据类型如下所示。

全局标签	
分类	**可以定义的数据类型**
VAR_GLOBAL	基本数据类型、数组、结构体、FB
VAR_GLOBAL_CONSTANT	基本数据类型*1
VAR_GLOBAL_RETAIN	基本数据类型*1、数组、结构体

局部标签(程序块)	
分类	**可以定义的数据类型**
VAR	基本数据类型、数组、结构体、FB
VAR_CONSTANT	基本数据类型*1
VAR_RETAIN	基本数据类型*1、数组、结构体

局部标签(函数)	
分类	**可以定义的数据类型**
VAR	基本数据类型*2、数组、结构体
VAR_CONSTANT	基本数据类型*1
VAR_INPUT	基本数据类型*1*2、数组、结构体
VAR_OUTPUT	
返回值	

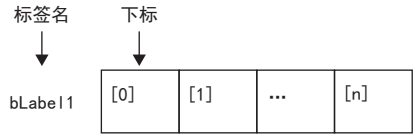
局部标签(FB)	
分类	**可以定义的数据类型**
VAR	基本数据类型、数组、结构体、FB
VAR_CONSTANT	基本数据类型*1
VAR_RETAIN	基本数据类型*1、数组、结构体
VAR_INPUT	
VAR_OUTPUT	
VAR_OUTPUT_RETAIN	
VAR_IN_OUT	
VAR_PUBLIC	
VAR_PUBLIC_RETAIN	

*1　不可以定义指针类型。
*2　不可以定义定时器型、累计定时器型、计数器型、长计数器型。

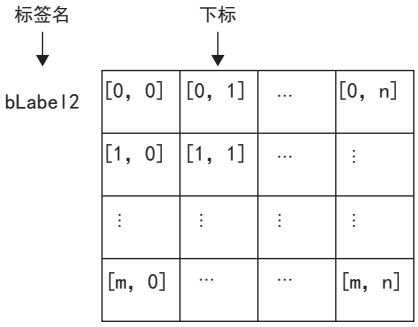
4.4　数组

数组是指将相同数据类型的标签的连续集合体用一个名称表示。
可以将基本数据类型、结构体作为数组进行定义。
根据数据类型，数组的最多个数不同。

■1维数组的图像



■2维数组的图像



数组的定义

■数组的要素

定义数组时，应决定要素数(数组的长度)。要素数的范围请参照下述内容。
☞ 32页 数组要素数的范围

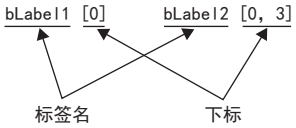
■定义的格式

下面以直到3维数组为例，对定义的格式进行说明。
数组开始值～数组结束值之间的范围即为要素数。

数组的维数	格式	备注
1维数组	基本数据类型/结构体名的数组(数组开始值..数组结束值)	• 关于基本数据类型: ☞ 29页 基本数据类型 • 关于结构体名: ☞ 33页 结构体
	【定义示例】 位(0..2)	
2维数组	基本数据类型/结构体名的数组(数组开始值..数组结束值，数组开始值..数组结束值)	
	【定义示例】 位(0..2，0..1)	
3维数组	基本数据类型/结构体名的数组(数组开始值..数组结束值，数组开始值..数组结束值，数组开始值..数组结束值)	
	【定义示例】 位(0..2，0..1，0..3)	

使用方法

使用数组时，为了识别各个标签，在标签名后用“[]”将下标括起来。
此外，2维以上的数组的情况下，“[]”内的下标要用“逗号(,)”隔开。



数组中的下标可以指定为下述类型。

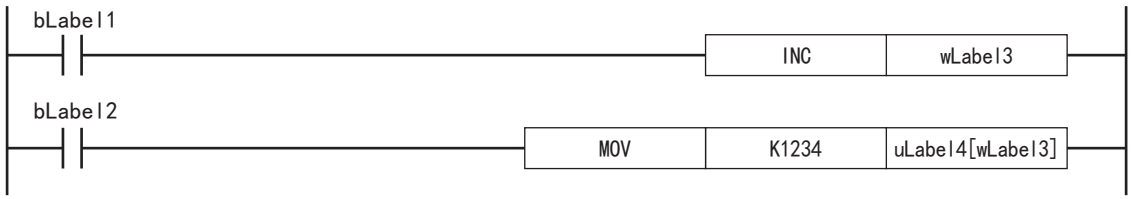
类型	指定示例	备注
常数	bLabel1[0]	可以指定0以上的整数。可以指定10进制数、16进制数。
软元件	bLabel1[D0]	可以指定字软元件、双字软元件。
标签	bLabel1[uLabel2]	可以指定下述的数据类型。 • 字[无符号]/位列[16位] • 双字[无符号]/位列[32位] • 字[带符号] • 双字[带符号]
表达式	bLabel1[5+4]	只能通过ST语言指定。

注意事项

当一个软元件/标签的位被分配到全局标签中的位数组时 （例：D0.0），该软元件/标签不能再用作数组下标。
（例：bLabel1[D0]）

要点

- 通过在数组的下标中指定标签，数据存储目标变为动态，因此可以在执行重复处理的程序中使用。下述为在“uLabel4”的数组中连续存储“1234”的程序。



- 梯形图语言的情况下，使用数组时可省略要素编号。使用时省略了要素编号的情况下，作为数组要素的起始编号进行转换。例如所定义的标签名为“boolAry”，数据类型为“位(0..2,0..2)”的数组时，“boolAry[0,0]”和“boolAry”会进行同样处理。
- 能够在使用数组的指令或函数、FB的设置数据中指定多维的数组。此时，数组要素中最右侧的要素会作为一维数组加以处理。

数组要素数的范围

数组的最多个数根据数据类型而不同。

数据类型	设置范围
位 字[无符号]/位列[16位] 双字[无符号]/位列[32位] 字[带符号] 双字[带符号] 单精度实数 时间 定时器 累计定时器 计数器 长计数器	1～32768
字符串(32)	1～32768÷字符串长度
字符串[Unicode](32)	1～16384÷字符串长度

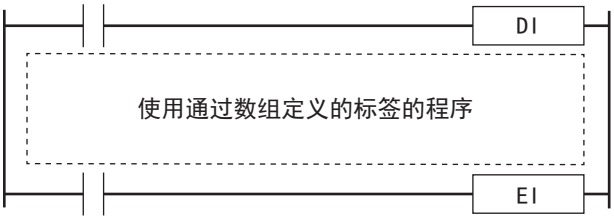
注意事项

■使用中断程序的情况下

在数组的下标中指定了标签或软元件的情况下，组合多个指令进行运算。

因此，如果通过数组定义的标签的运算中发生中断，将发生数据的不完整，变为意料外的运算结果。

应使用下述的中断禁止/允许指令(DI/EI指令)创建程序以确保不发生数据不完整。



关于DI/EI指令的详细内容，请参阅下述内容。

MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)

■关于数组的要素

对于定义的数组的要素数，请勿访问要素编号以外的范围。

用常数指定通过数组定义的范围外的下标的情况下，将变为转换出错。

此外，用常数以外指定数组的下标的情况下不会变为转换出错，而是在执行时访问其他标签区域、锁存标签区域的领域后进行处理。

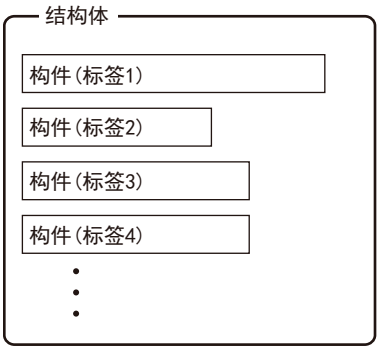
4.5 结构体

结构体是包含一个以上的标签的数据类型，可以在所有的程序部件中使用。

包含在结构体中的各个构件(标签)即使数据类型不同也可以定义。

结构体的创建

创建结构体首先要创建结构体的定义，其次在创建的结构体中定义构件。



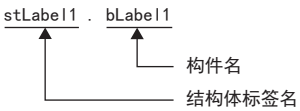
使用方法

使用结构体的情况下，登录预先将定义的结构体置为数据类型的标签。

对于指定配置的各构件，应在结构体标签名后用“句点(.)”断开并附上构件名。

例

使用结构体构件的情况下



要点

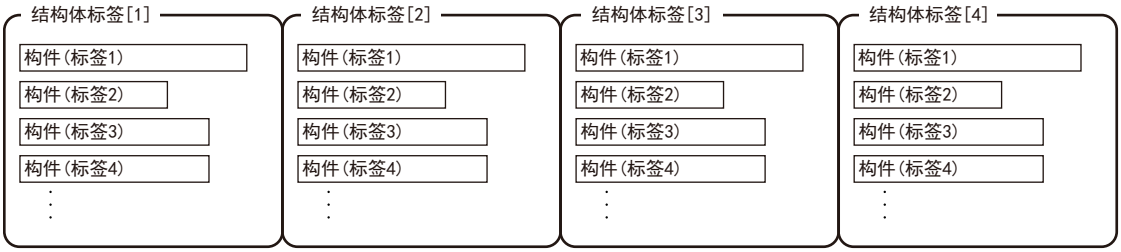
- 在结构体中定义多个数据类型后登录标签，在程序中使用的情况下，转换后的数据存储的顺序不会变为定义了数据类型的顺序。利用工程工具进行转换时，分类为标签类型与数据类型后进行分配。(根据填充块进行存储器分配)

GX Works3操作手册

- 对于使用控制数据(设置指令动作的操作数)的指令，如果指定结构体的标签，根据填充块进行存储器分配，不会变为定义的顺序。

结构体的数组

可以将结构体数组化后使用。

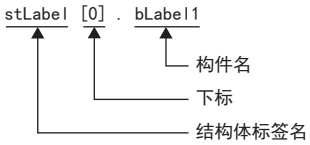


作为数组声明的情况下应在结构体标签名后用“[]”将下标括起来表示。

可以将结构体的数组作为函数及FB的自变量进行指定。

例

使用数组化的结构体的要素的情况下



可以指定的数据类型

下述数据类型可以作为结构体的构件进行指定。

- 基本数据类型
- 指针型
- 数组
- 其他结构体

结构体的类型

下述数据类型预先被定义为结构体。

类型	参照目标
定时器类型	☞ 29页 数据类型
累计定时器类型	
计数器类型	
长计数器类型	

4.6 常数

常数的类型

在标签中设置常数时的标记如下所示。

可以对应的数据类型	类型	标记方法	标记示例
位	布尔	输入“FALSE”或“TRUE”。	TRUE、FALSE
	2进制数	在2进制数的数值前附上“2#”。	2#0、2#1
	8进制数	在8进制数的数值前附上“8#”。	8#0、8#1
	10进制数	直接输入所使用的10进制数。或者在数值前附上“K”。	0、1、K0、K1
	16进制数	在16进制数的数值前附上“16#”。或者在数值前附上“H”。	16#0、16#1、H0、H1
• 字[无符号]/位列[16位] • 双字[无符号]/位列[32位] • 字[带符号] • 双字[带符号]	2进制数*1	在2进制数前附上“2#”。	2#0010、2#01101010、2#1111_1111
	8进制数*1	在8进制数前附上“8#”。	8#0、8#337、8#1_1
	10进制数*1	直接输入10进制数，或者在数值前附上“K”。	123、K123、K-123、12_3
	16进制数*1	在16进制数前附上“16#”。 或者在数值前附上“H”。	16#FF、HFF、16#1_1
单精度实数	实数*1	直接输入实数，或者在数值前附上“E”。	2.34、E2.34、E-2.34、3.14_15
	实数(指数表现)	指数表现，或者在实数值前附上“E”，然后在指数部分前附上“+”。	1.0E6、E1.001+5
字符串(32)	字符串	将字符串(ASCII、移位JIS)用单引号(’)括起来。	‘ABC’
字符串[Unicode](32)	Unicode字符串	将Unicode字符串用双引号(”)括起来。	“ABC”
时间	时间	在开头附上“T#”。	T#1h、T#1d2h3m4s5ms

*1 在2进制数、8进制数、10进制数、16进制数、实数的标记中，用下划线(_)断开数值，可以使程序更易懂。(在程序的处理上可以忽略。)

在字符串类型的常数中使用“$”的情况下

“$”作为转换序列使用。

紧接着“$”的两个16进制数字符作为ASCII代码被识别，与ASCII代码相对应的字符被插入到字符串中。

紧接着“$”的两个16进制数字符与ASCII代码不相对应的情况下，即为转换错误。

但是，紧接着“$”的字符在下述情况下不会变为错误。

标记	在字符串中使用的符号或打印机代码
$$	$
$’	’
$”	”
$L或$I	移行
$N或$n	换行
$P或$p	进页
$R或$r	复位
$T或$t	制表

4.7 注意事项

有限制的功能

在下述功能中使用标签时有限制。

项目	内容
事件执行类型程序的触发	不能使用标签。请考虑下列方法。 • 应使用软元件。 • 应将所使用的标签作为全局标签进行定义，分配软元件以进行对应。
智能功能模块的刷新设置	不能使用标签。请考虑下列方法。 • 应使用软元件。

■定义分配软元件的全局标签并使用的情况下

使用对标签有限制的功能时，应按下述顺序定义全局标签后使用。

另外，由于是在软元件区域消耗软元件/标签存储器，所以应确保软元件区域。(不消耗标签区域)

1. 确保所使用的软元件区域。

CPU参数⇨存储器/软元件设置⇨软元件/标签存储器区域容量设置

2. 在全局标签中定义标签后手动分配软元件。

3. 在可以使用标签的功能中，使用步骤2中定义的标签。在对标签的使用有限制的功能中，应使用分配到标签中的软元件。

■将所使用的标签的值暂时复制到其他软元件中的情况下

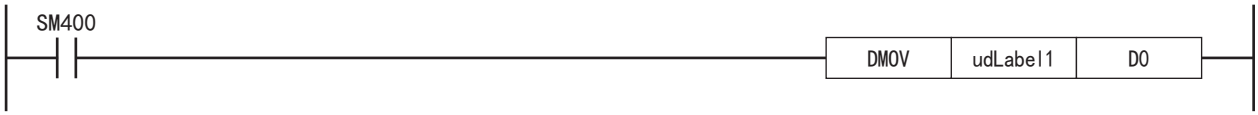
应按照下述步骤暂时将标签值复制到其他软元件中，在对标签有限制的功能中使用该软元件。

另外，由于是在软元件区域消耗软元件/标签存储器，所以应确保软元件区域。

1. 确保所使用的软元件区域。

CPU参数⇨存储器/软元件设置⇨软元件/标签存储器区域容量设置

2. 使用标签进行程序的创建。添加的程序示例如下所示。(通过数据记录功能使用存储在udLabel1中的值的情况下。)



3. 在对标签的使用有限制的功能中，应使用步骤2中传送的软元件。(步骤2的程序示例的情况下，使用D0。)

要点

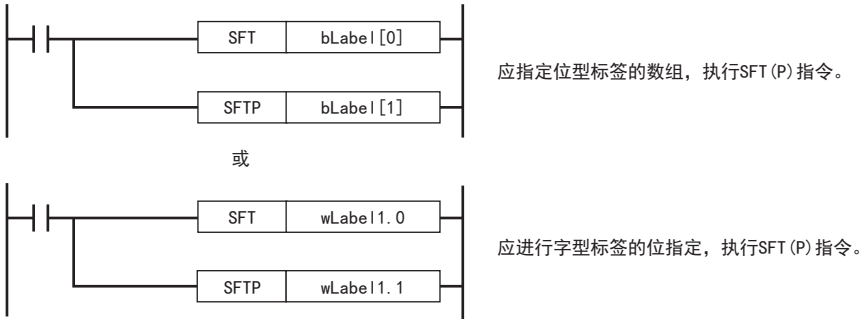
应在考虑将值写入标签的时机及功能的执行时机后决定传送指令的添加位置。

创建程序时的注意事项

在指令等操作数中指定标签的情况下，应确保标签的数据类型与用操作数指定的数据类型相符合。此外，在处理连续数据的指令等操作数中指定标签的情况下，应指定操作指令的数据范围包含在具有标签的数据范围内。

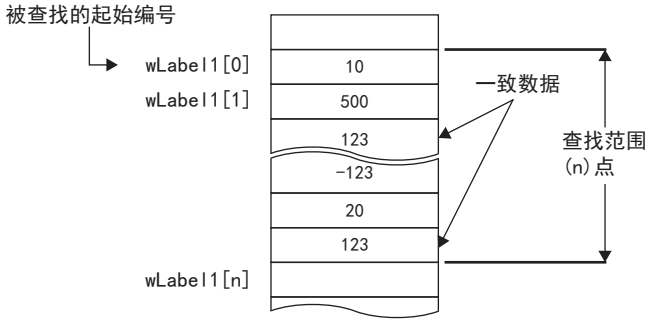
例

SFT(P)指令的情况下



例

SFR(P)指令的情况下



指定的标签的数组应指定具有比查找范围(n)点更大的范围的标签。

标签名的限制

标签名中存在以下限制。

- 标签名应以字符或下划线(_)为开头。不能定义以数字开头的标签名。
- 不能在标签名中定义保留字。

关于保留字的详细内容，请参阅下述手册。

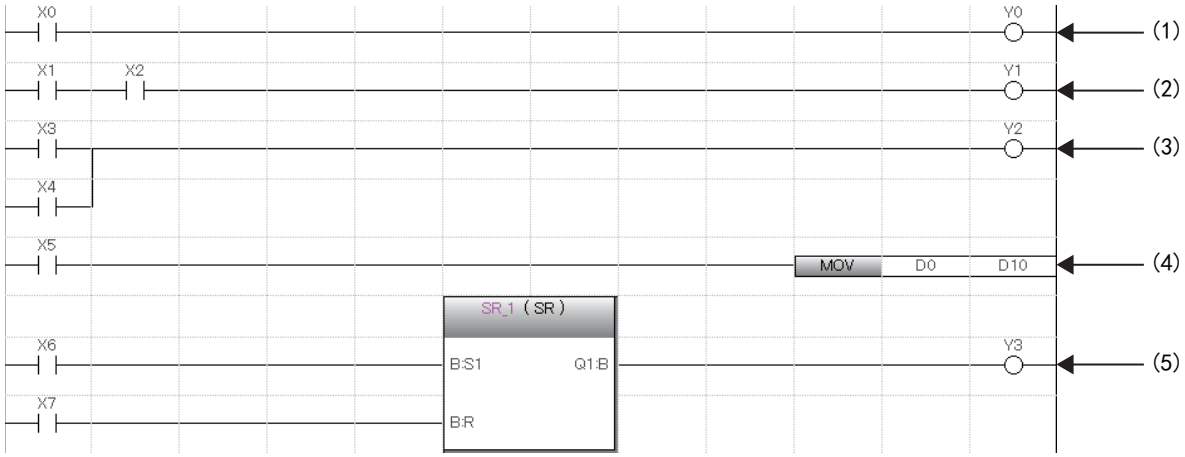
GX Works3操作手册

5 梯形图语言

是在触点与线圈构成的回路中通过串联与并联的组合表示由AND/OR组成的逻辑运算，记述顺控程序的语言。

5.1 配置

使用梯形图语言可以创建下述的回路。

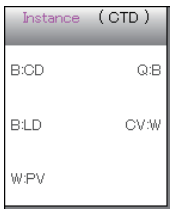


(1) 由触点与线圈组成的回路
(2) 由串联组成的回路
(3) 由并联组成的回路
(4) 使用了指令的回路
(5) 使用了通用函数/功能块的回路

回路符号

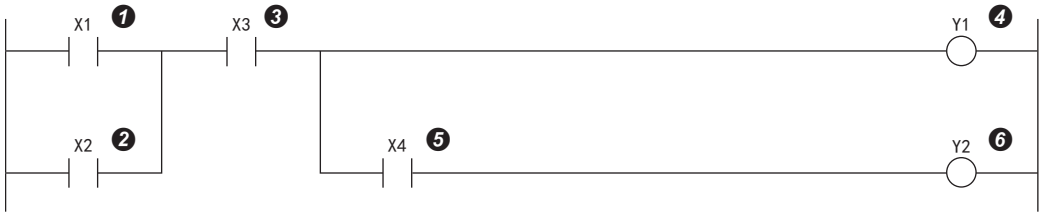
可以在梯形图语言的编程中使用的回路符号如下所示。

要素	符号	说明
常开触点		指定软元件或标签变为ON时导通。
常闭触点		指定软元件或标签变为OFF时导通。
上升沿脉冲		指定软元件或标签上升沿时(OFF→ON)导通。
下降沿脉冲		指定软元件或标签下降沿时(ON→OFF)导通。
非上升沿脉冲		指定软元件或标签OFF时、ON时以及下降沿时(ON→OFF)导通。
非下降沿脉冲		指定软元件或标签OFF时、ON时以及上升沿时(OFF→ON)导通。
运算结果上升沿脉冲化		运算结果上升沿时(OFF→ON)导通。运算结果在上升沿以外的情况下不导通。
运算结果下降沿脉冲化		运算结果下降沿时(ON→OFF)导通。运算结果在下降沿以外的情况下不导通。
运算结果取反		在开始此指令前将运算结果取反。
线圈		将运算结果输出到指定软元件或标签中。
指令	SET M0	在[]内执行指定指令。
换行	K0	超过了一个回路行中可以创建的触点数的情况下，创建换行处的符号及换行目标的符号，执行回路的换行。
函数	ABS IN ABS	执行函数。 • 函数的创建方法(GX Works3操作手册) • 通用函数(MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇))

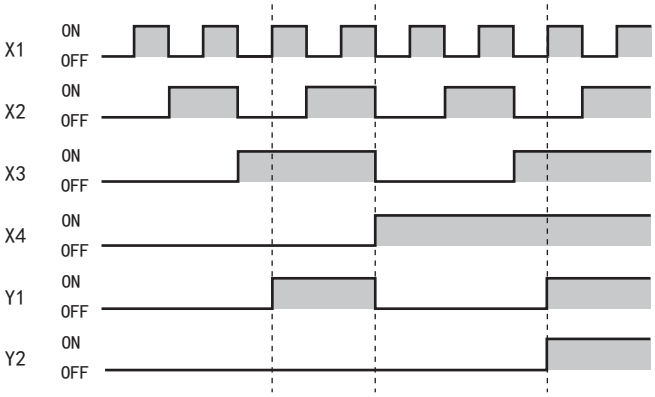
要素	符号	说明
FB		执行FB。 • FB的创建方法(GX Works3操作手册) • 通用FB(MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)) • 模块FB(MELSEC iQ-F FX5 CPU模块FB参考)

程序执行顺序

按照下述的编号顺序执行。



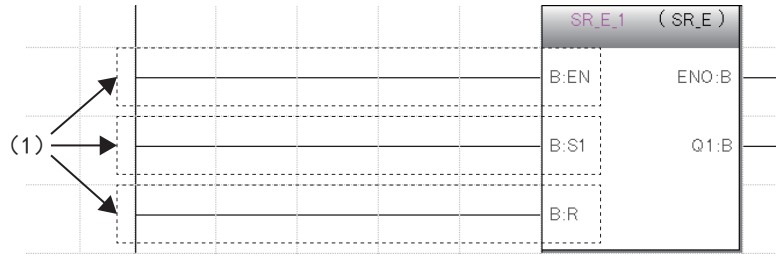
执行了上述程序的情况下，根据X1～X4的ON/OFF，Y1、Y2的ON时机如下所示。



以梯形图语言使用FB时的注意事项

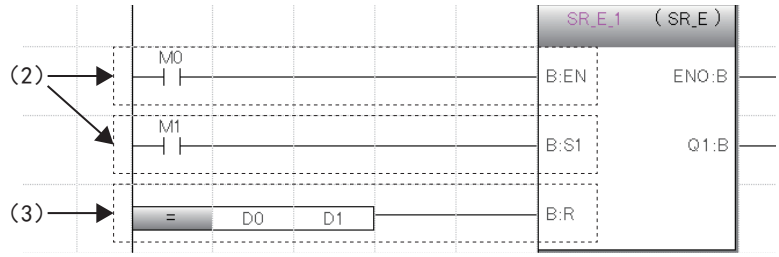
左母线已直接连接FB实例时的注意事项

在FB实例的输入梯形图部，EN及输入变量(位型)与左母线直接连接时，ON/OFF的状态不发生变化。



(1)：ON/OFF的状态不发生变化。

若要使EN及输入变量(位型)的ON/OFF的状态发生变化，应使用触点或相当于触点的指令。

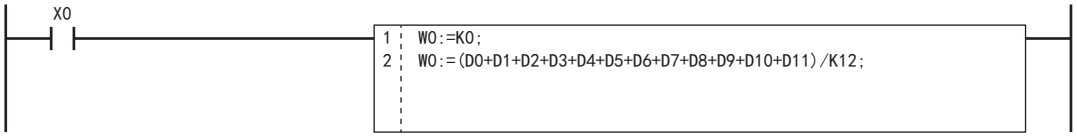


(2)：触点
(3)：相当于触点的指令

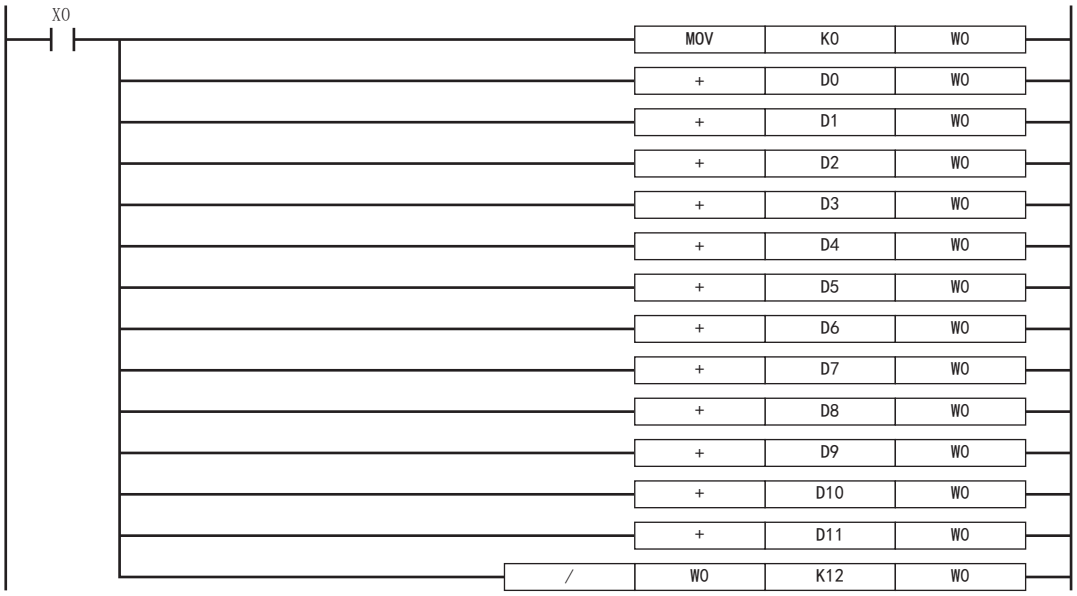
5.2 内嵌ST

内嵌ST是指在梯形图编辑器内创建并编辑/监视在与线圈相当的指令模块中显示ST程序的内嵌ST框的功能。
由此可以轻松地在梯形图程序内创建数据运算或字符串处理。

• 使用了内嵌ST情况下的程序



• 不使用内嵌ST情况下的程序



限制事项

在SFC程序的Zoom编辑器内不能使用内嵌ST。

规格

关于内嵌ST的规格，请参阅ST语言的规格。

☞43页 ST语言

注意事项

- 梯形图程序的1行中只能创建一个内嵌ST。
- 在梯形图程序的1行中无法同时使用FB与内嵌ST框。
- 如果在触点相应的指令位置创建内嵌ST框，在线圈相应的指令位置也会创建内嵌ST框。
- 内嵌ST内最多可输入字符数为2048个。(换行作为2个字符进行计数。)
- 内嵌ST内，有时上升执行指令、下降执行指令、特殊定时器指令、OUT指令、定位指令、通用FB的边缘检测FB以及计数器FB不能正常运行，因此请勿使用。
- 如果在内嵌ST内使用“RETURN语句”，则不会结束程序块的处理，而是结束内嵌ST框内的处理。

5.3 声明/注解

在梯形图回路中可以显示声明与注解。

声明

通过使用声明，可以对回路块添加注释。通过添加注释，处理等流程变得易懂。
声明中有行间声明/P声明/I声明。
行间声明可以在导航窗口的树状结构上显示。

■行间声明

对整个回路块添加注释。

■P声明

对指针编号添加注释。

■I声明

对中断指针编号添加注释。

注解

通过使用注解，可以对程序中的线圈及指令添加注释。
通过添加注释，线圈及指令的内容等变得易懂。

声明/注解的类别

声明与注解的类别分为“PLC”与“外围”。

类别	类型	内容
PLC	• 行间声明 • P声明 • I声明 • 注解	可以将声明/注解存储在CPU模块中。 PLC声明需要消耗下述步数。(小数点以下舍去。) • 无字符：3步 • 有字符：4+(字符数+2+14)/15+字符数
外围	• 行间声明 • P声明 • I声明 • 注解	说明文的字符串不置入到程序内，而是作为程序的附加信息保存。 每一行消耗一步。 在输入的文本前自动添加*印记。

6 ST语言

ST语言是在规定逻辑记述方式的国际标准IEC61131-3中定义的语言。ST语言是具有与C语言等相似的语法结构的文本形式的程序语言。适用于对梯形图语言难以表现的复杂处理进行编程的情况。
ST语言支持控制语法、运算式、功能块(FB)、函数(FUN)，可以进行如下的记述。

例

通过条件语句进行选择分支，通过重复语句进行重复等的控制语法

```
(*在生产线A～C中进行控制*)
  CASE 生产线 OF
       1: 开始开关 := TRUE; (*传送带移动*)
       2: 开始开关 := FALSE;(*传送带停止*)
       3: 开始开关 := TRUE; (*传送带停止 警告*)
       ELSE 警告指示灯 := TRUE;
  END_CASE;

  IF 开始开关 = TRUE THEN   (*传送带运转 处理100次*)
       FOR 处理次数 := 0
            TO 100
            BY 1 DO
            处理数 := 处理数 +1;
       END_FOR;
  END_IF;
```

例

使用运算符(*、/、+、-、<、>、=等)的表达式

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

例

定义的FB的调用

```
//FB数据名      : LINE1_FB
//输入变量      : I_Test
//输出变量      : O_Test
//输入输出变量 : IO_Test
//FB标签名      : FB1
FB1(I_Test :=D0,O_Test => D1,IO_Test := D100);
```

例

通用函数的调用

```
(*将BOOL型数据转换为INT型/DINT型数据*)
wLabel2 := BOOL_TO_INT (bLabel1);
```

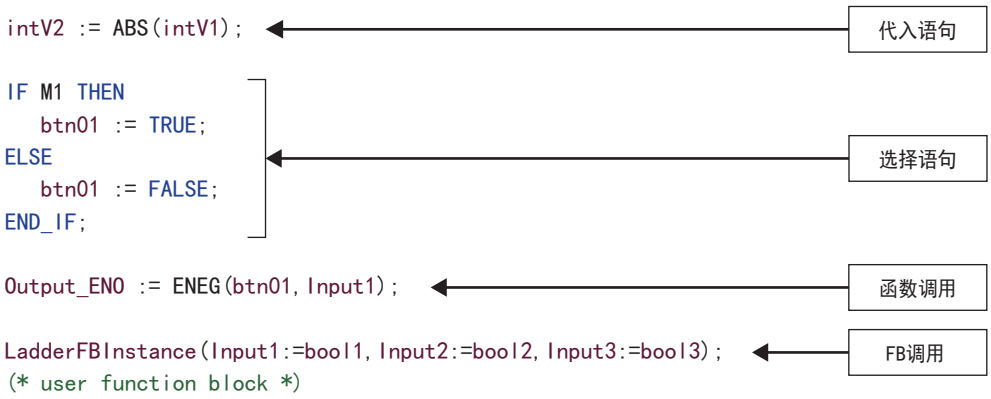
例

汉字等全角字符的使用

```
 //槽罐的限位器变为ON时关闭阀门，变为OFF时打开阀门
 IF 槽罐限位器 = TRUE THEN
    阀门 := FALSE;     (*限位器变为ON，因此关闭阀门*)
    ELSE
    阀门 := TRUE;      /*限位器变为OFF，因此打开阀门*/
 END_IF;
```

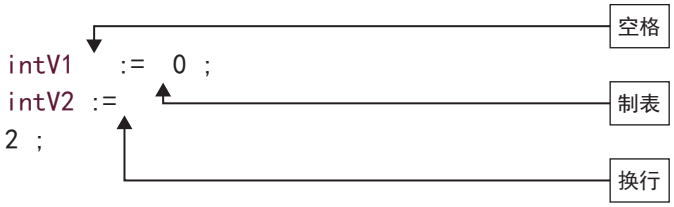
6.1 配置

ST语言中的编程由运算符与语句组成。

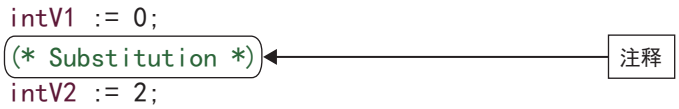


语句的终端必须添加“;”(分号)。

空格、制表、换行可以插入到运算符及数据中。



可以在程序中插入注释。



程序的结构要素

ST程序由以下要素构成。

项目		示例	参照页
段落符号		; , ( )	☞45页 段落符号
运算符		+、-、<、>、=	☞45页 运算符
保留字	语句	IF、CASE、WHILE、RETURN	☞46页 语句
	软元件	X0、Y10、M100	📖MELSEC iQ-F FX5用户手册(应用篇)
	数据类型	BOOL、DWORD	☞29页 数据类型
	函数	ADD、REAL_TO_STRING_E	📖MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)
常数		123、‘abc’	☞53页 常数
标签		Switch_A	☞53页 标签与软元件
注释		(*置为ON*)、//置为ON、/*置为ON*/	☞54页 注释
其他符号		半角空格、换行代码、TAB代码	—

- 段落符号、运算符、保留字应用半角记述。
- 关于保留字的详细内容，请参阅下述手册。

📖GX Works3操作手册

段落符号

在ST语言中，为了明确程序的结构，设有下述的段落符号。

符号	内容
()	括弧
[]	数组要素的指定
.(句点)	结构体、FB构件的指定
,(逗号)	自变量的断开
:(冒号)	软元件型指定符
;(分号)	语句的终端
’(单引号)	字符串(ASCII、移位JIS)的标记
”(双引号)	Unicode字符串的标记
..(两个句点)	整数范围指定

运算符

在ST程序中使用的运算符、对象数据类型与运算结果的数据类型如下所示。

运算符	对象数据类型	运算结果类型
*、/、+、-	ANY_NUM	ANY_NUM
<、>、<=、>=、=、<>	ANY_ELEMENTARY*1	位
MOD	ANY_INT	ANY_INT
AND、&、XOR、OR、NOT	ANY_BIT	ANY_BIT
**	ANY_REAL(底) ANY_NUM(指数)	ANY_REAL

*1　不可以指定WSTRING型的Unicode字符串。

运算符的优先顺序如下所示。

运算符	内容	示例	优先顺序
( )	圆括弧式	(2+3)*(4+5)	1
函数()	函数的自变量	CONCAR(‘AB’、‘CD’)	2
**	幂运算	3.0**4	3
-	符号取反	-10	4
NOT	位型补数	NOT TRUE	
*	乘法运算	10 * 20	5
/	除法运算	20 / 10	
MOD	求余数运算	17 MOD 10	
+	加法运算	1.4 + 2.5	6
-	减法运算	3 - 2	
<、>、<=、=>	比较	10 > 20	7
=	一致	T#26h = T#1d2h	8
<>	不一致	8#15 <> 13	
&、AND	逻辑与	TRUE AND FALSE	9
XOR	逻辑异或	TRUE XOR FALSE	10
OR	逻辑或	TRUE OR FALSE	11

- 在一个公式中有多个优先顺序相同的运算符的情况下，从左侧开始运算。
- 一个公式中可以记述的运算符的使用个数最多为1024个。

语句

可以在ST程序中使用的语句如下所示。

项目	内容	参照页
代入语句	代入语句	46页 代入语句
子程序控制语句	FB调用语句、函数调用语句	47页 子程序控制语句
	RETURN语句	
选择语句	IF语句(IF、IF...ELSE、IF...ELSIF)	48页 选择语句
	CASE语句	
重复语句	FOR语句	48页 重复语句
	WHILE语句	
	REPEAT语句	
	EXIT语句	

应用半角字符记述语句。

代入语句

格式	内容	记述示例
<左边> := <右边>;	具有将右边公式的结果代入到左边的标签及软元件中的功能。 需要使右边公式的结果与左边的数据类型相同。	intV1 := 0; intV2 := 2;

使用数组型标签及结构体标签的情况下，应注意代入语句的左边与右边的数据类型。

数组型标签的情况下，需要使数据类型与要素数的左边与右边相同。此外，请勿指定要素。

例

```
intAry1 := intAry2;
```

结构体标签的情况下，需要使数据类型(结构体的数据类型)的左边与右边相同。

例

```
dutVar1 := dutVar2;
```

■数据类型的自动转换

在ST语言中，在记述不同的数据类型的代入及算数运算公式时，有时会自动转换数据类型。

例

自动转换的示例

```
dintLabel1 := intLabel1;
//代入语句:将INT型(intLabel1)的值自动转换为DINT型，代入到左侧的DINT型(dintLabel1)

dintLabel1 := dintLabel2 + intLabel1;
//算术运算公式:将INT型(intLabel1)的值自动转换为DINT型，执行DINT型的加法运算
```

类型转换通过向代入语句、FB及函数的输入自变量交接(VAR_INPUT部)、算术运算公式进行。

为确保在类型转换时数据不丢失，应仅从容量小的数据类型向容量大的数据类型进行类型转换。类型转换以基本数据类型中的下述数据类型为对象。

数据类型	内容
字[带符号]	转换后为双字[带符号]时，会自动转换为符号扩展后的值。 单精度实数时，会自动转换为与转换前的整数相同的值。*1
字[无符号]/位串[16位]	转换后为双字[无符号]/位串[32位]或者双字[带符号]时，会自动转换为零扩展后的值。*2 单精度实数时，会自动转换为与转换前的整数相同的值。*1

*1 数据类型交接至ANY_REAL的输入自变量时，会将16位的数据(字[带符号]或者字[无符号]/位串[16位])自动转换为单精度实数。

*2 数据类型交接至ANY32的输入自变量时，会将字[无符号]/位串[16位]的数据自动转换为双字[无符号]/位串[32位]。

上述以外的数据类型，应使用类型转换函数。

此外，在下述情况下因为无法进行类型转换，应使用类型转换函数。

- 符号不同的整数型之间的类型转换
- 数据丢失型之间的类型转换

代入算数运算的结果时的注意事项请参阅下述内容。

☞49页 使用算数运算公式时

子程序控制语句

■FB调用语句

格式	内容
实例名(输入变量1:=变量1,...输出变量1=>2,...);	在实例名后，用“()”括住输入变量、输出变量的代入语句。 多个变量的情况下，各代入语句之间用“,”(逗号)隔开。
实例名.输入变量1:=变量1; : 实例名(); 变量2:=实例名.输出变量1;	在FB调用的前后列举输入自变量、输出自变量的代入语句。

在FB调用语句的自变量中使用的符号与可分配公式如下所示。

类型	内容	属性	使用符号	可分配公式
VAR_INPUT	输入变量	无，或RETAIN	:=	所有的公式
VAR_OUTPUT	输出变量	无，或RETAIN	=>	只有变量
VAR_IN_OUT	输入输出变量	无	:=	只有变量

FB的执行结果通过在实例名后添加“.”(句点)指定输出变量，代入变量被存储。

FB	FB定义	记述示例
1个输入变量、1个输出变量的FB的情况下	FB名：FBADD FB实例名：FBADD1 输入变量1：IN1 输出变量1：OUT1	FBADD1(IN1:= Input1); Output1 := FBADD1.OUT1;
3个输入变量、2个输出变量的FB的情况下	FB名：FBADD FB实例名：FBADD1 输入变量1：IN1 输入变量2：IN2 输入变量3：IN3 输出变量1：OUT1 输出变量2：OUT2	FBADD1(IN1:=Input1, IN2:=Input2, IN3:=Input3) Output1 := FBADD1.OUT1; Output2 := FBADD1.OUT2;

■函数调用语句

格式	内容
函数名(变量1, 变量2,...);	用“()”将紧接在函数名后的自变量括起来。 多个自变量的情况下用“,”隔开。

通过向变量代入，存储执行函数的结果。

函数	记述示例
输入变量为1个函数的情况下(例：ABS)	Outout1 := ABS(Input1);
输入变量为3个函数的情况下(例：MAX)	Outout1 := MAX(Input1, Input2, Input3);
具有EN/ENO的函数(通用函数)的情况下(例：MAX_E)	Output1 := MAX_E(boolEN, boolENO, Input1, Input2, Input3);
通用函数以外的情况下(例：MOV)	boolENO := MOV(boolEN, Input1, Output1); (执行函数的结果变为ENO，第一自变量(变量1)变为EN。)

不返回值的用户定义函数和在调用语句的自变量中含有VAR_OUTPUT变量的函数能够通过在其后添加“;”(分号)，作为语句加以执行。

■RETURN语句

语句	格式	内容	记述示例
■RETURN	RETURN;	为了使程序、FB、函数在中途结束而使用。 如果在程序中使用了RETURN语句，将跳转到程序的最后语句的下一步。 如果在FB中使用了RETURN语句，将从FB返回。 如果在函数中使用了RETURN语句，将从函数返回。 相对于1个RETURN语句，在系统使用1点指针型标签。	IF bool1 THEN RETURN; END_IF;

不返回值的用户定义函数和在调用语句的参数中含有VAR_OUTPUT变量的函数能够通过在其后添加“;”(分号)，作为语句加以执行。

选择语句

语句	格式	内容	记述示例

■IF	IF <布尔表达式> THEN
	<语句…>;
	END_IF;
		布尔表达式(条件式)为真(TRUE)时，则执行语句。布尔表达式为假(FALSE)时，则不执行语句。
		在布尔表达式中，作为在单一的位型变量的状态下或包含多个变量的复杂的表达式的布尔运算结果，如果是返回真(TRUE)或假(FALSE)的表达式，则可以使用任意表达式。
			IF bool1 THEN
			intV1:=intV1+1;
			END_IF;

■IF...ELSE	IF <布尔表达式> THEN
	<语句1…>;
	ELSE
	<语句2…>;
	END_IF;
		布尔表达式(条件式)为真(TRUE)时，则执行语句1。
		布尔表达式的值为假(FALSE)时，则执行语句2。
			IF bool1 THEN
			intV3:=intV3+1;
			ELSE
			intV4:=intv4+1;
			END_IF;

■IF...ELSIF	IF <布尔表达式1> THEN
	<语句1…>;
	ELSIF <布尔表达式2> THEN
	<语句2…>;
	ELSIF <布尔表达式3> THEN
	<语句3…>;
	END_IF;
		布尔表达式(条件式)1为真(TRUE)时，则执行语句1。布尔表达式1的值为假(FALSE)而布尔表达式2的值为真(TRUE)时，则执行语句2。
		布尔表达式1、2的值都为假(FALSE)而布尔表达式3的值为真(TRUE)时则执行语句3。
			IF bool1 THEN
			intV1:=intV1+1;
			ELSIF bool2 THEN
			intv2:=intV2+2;
			ELSIF bool3 THEN
			intV3:=intV3+3;
			END_IF;

■CASE	CASE <整数式> OF
	<整数选择值1>:
	<语句1…>;
	<整数选择值2>:
	<语句2…>;
	:
	<整数选择值n>:
	<语句n…>;
	ELSE
	<语句n+1…>;
	END_CASE;
		执行具有与整数式(条件式)的值一致的整数选择值的语句，在无一致的情况下，则执行ELSE语句的下一语句。
		CASE语句可以在例如根据单一的整数值及复杂表达式的结果的整数值执行选择语句的情况下使用。
			CASE intV1 OF
			1:bool1:=TRUE;
			2:bool2:=TRUE;
			ELSE
			intV1:=intV1+1;
			END_CASE;

重复语句

语句	格式	内容	记述示例
■FOR	FOR <重复变量初始化> TO <最终值> BY <增加表达式> DO <语句…>; END_FOR;	首先进行作为重复变量使用的数据的初始化。 根据增加表达式对初始化后的重复变量进行加法或减法运算，在达到最终值前一直重复执行从DO算起END_FOR内的1个以上的语句。 FOR...DO语句结束后的重复变量保持着结束时的值。	FOR intV1:=0 TO 30 BY 1 DO intV3:=intV1+1; END_FOR;

■WHILE	WHILE <布尔表达式> DO
	<语句…>;
	END_WHILE;
		布尔表达式(条件式)为真(TRUE)时，则执行超过1个的语句。
		布尔表达式在语句执行之前判定，布尔表达式为假(FALSE)时则不执行WHILE ...DO内的语句。因为WHILE语句中的<布尔表达式>只要返回结果是真或假即可，因此IF条件语句中的<布尔表达式>中可指定的表达式则全部可以使用。
			WHILE intV1=30 DO
			intV1:=intV1+1;
			END_WHILE;

■REPEAT	REPEAT
	<语句…>;
	UNTIL <布尔表达式>
	END_REPEAT;
		布尔表达式(条件式)为假(FALSE)时，则执行超过1个的语句。
		布尔表达式在语句执行后判定，值为真(TRUE)时则不执行REPEAT...UNTIL内的语句。因为REPEAT语句中的<布尔表达式>只要返回结果是真或假即可，因此IF条件语句中的<布尔表达式>中可指定的表达式则全部可以使用。
			REPEAT
			intV1:=intV1+1;
			UNTIL intV1=30
			END_REPEAT;

语句	格式	内容	记述示例
■EXIT	EXIT;	通过只能在重复语句中使用的语句，使重复语句在中途结束。 如果在执行反复重复语句过程中达到了EXIT语句，则不执行EXIT语句之后的反复重复处理。终止重复语句后从下一行继续程序的执行。	FOR intV1:=0 TO 10 BY 1 DO IF intV1>10 THEN EXIT; END_IF; END_FOR;

注意事项

■使用代入语句时

- 代入字符串的最大字符串长255字符。代入超过最大字符串长的字符串时，即为转换错误。
- 定时器型、计数器型的触点与线圈无法在代入语句的左边使用。
- FB的实例无法在代入语句的左边使用。应在代入语句的左边使用实例的输入变量、输入输出变量、外部变量。

■使用步进继电器(S)及SFC块软元件(BL)的情况

将步进继电器(S)及SFC块软元件(BL)置于代入式的右边、使用函数及FB的输入自变量时，有可能变为转换出错状态。此时，应替换代入式。

例

替换示例如下所示。

替换前	替换后

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

此外，在程序中使用步进继电器(S)或带块指定步进继电器(BL□\S□)的位数指定的情况下，应指定正确的数据容量。由于步进继电器(S)或带块指定步进继电器(BL□\S□)不是数据类型自动转换的对象，如果数据容量不一致，有可能变为转换出错状态。

例

替换示例如下所示。

替换前	替换后
(*K4S0为16位，D0:UD为32位，因此转换出错*) D0:UD := K4S0; (*BL1\K4S10为16位，DMOV的第2自变量为32位，因此转换出错*) DMOV(TRUE,BL1\K4S10,D100);	(*代入至16位数据*) D0 := K4S0; (*DMOV中指定32位数量的数据*) DMOV(TRUE, BL1\K8S10, D100:UD);

■使用算数运算公式时

将算数运算公式的结果代入到数据容量大的数据类型的变量中的情况下，应预先把算数运算公式的变量转换为左边的数据类型之后再进行运算。

例

在把数据容量16位(INT型)的算术运算结果代入到32位的数据类型(DINT型)的情况下

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

算术运算公式的运算结果将变为与输入操作数的数据类型相同的数据类型。因此在上述的程序中，在varInt1 * 10的运算结果超出了INT型的范围(-32768～+32767)的情况下，上溢或下溢的运算结果被代入到varDint1中。

在这种情况下，应预先把运算表达式的操作数转换到左边的数据类型之后再进行运算。

```
varDint2 := INT_TO_DINT(varInt1);            //将INT型变量转换为DINT型变量
varDint2 := varDint2 * 10;                   //用DINT型进行乘法运算，代入运算结果
```

■在算术运算公式中使用符号取反的运算符时

如果相对于数据类型的最小值使用符号取反的运算符(-)，则会为相同值。

例如INT型的最小值时，-(-32768) = -32768。因此，在作为数据类型的自动转换的对象的变量中使用符号取反的运算符时，有时会出现意料外的结果。

例

varInt1(INT型)的值为-32768，varDint1(DINT型)的值为0时

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

这种情况下，(-varInt1)的值原样不变地为-32768，并且-32768会代入varDint2中。

算术运算公式中使用符号取反的运算符时，请预先在算术运算之前自动转换数据类型，或者创建不使用符号取反的运算符的程序。

例

在算术运算之前自动转换数据类型时

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

例

不使用符号取反的运算符时

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

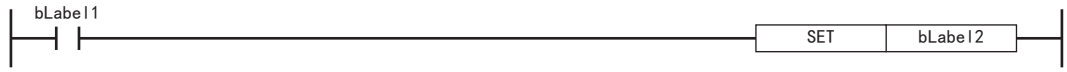
■使用位型标签时

选择语句或重复语句中布尔表达式(条件式)一旦成立，<语句>内的位型标签处在ON状态下时,则这个位型标签将变为始终ON。

例

始终ON程序

ST程序	ST程序同等处理的梯形图程序

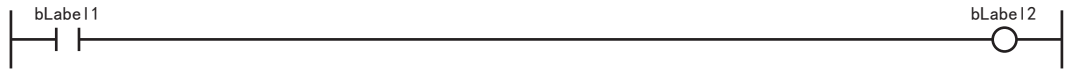
IF bLabel1 THEN
  bLabel2 := TRUE;
END_IF;
	

为避免始终ON，应按下述方式添加将位型标签置为OFF的程序。

例

避免始终ON的程序

ST程序*1	ST程序同等处理的梯形图程序

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

*1 上述程序可以按下述方式记述。
bLabel2 := bLabel1;
或
OUT(bLabel1,bLabel2);
但是，在<语句>内使用了OUT指令的情况下，变为与始终ON程序相同的状态。

■使用定时器FB、计数器FB时

对于选择语句中的布尔表达式(条件式)，计时器FB、计数器FB的执行条件不同。

例

定时器FB的情况下

■ 更改前程序示例

```
IF bLabel1 THEN
    TIMER_100_FB_M_1(Coil:=bLabel2,Preset:=wLabel3,ValueIn:=wLabel4,ValueOut=>wLabel5,Status=>bLabel6);
END_IF;
(*bLabel1=ON并且bLabel2=ON时，开始计数。                    *)
(*bLabel1=ON并且bLabel2=OFF时，计数清零。                   *)
(*bLabel1=OFF并且bLabel2=ON时，停止计数。计数值不清零。     *)
(*bLabel1=OFF并且bLabel2=OFF时，停止计数。计数值不清零。*)
```

■ 更改后程序示例

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

计数器FB的情况下

■ 更改前程序示例

```
IF bLabel1 THEN
    COUNTER_FB_M_1(Coil:=bLabel2,Preset:=wLabel3,ValueIn:=wLabel4,ValueOut=>wLabel5,Status=>bLabel6);
END_IF;
(*bLabel1=ON并且bLabel2=ON/OFF时，计数+1。  *)
(*bLabel1=OFF并且bLabel2=ON/OFF时，不计数。*)
(*bLabel1=ON/OFF不与计数+1联动。             *)
```

■ 更改后程序示例

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

上述更改前程序示例是在选择语句不成立的情况下，为了不执行与定时器、计数器相关联的语句而创建的。
根据bLabel1条件与bLabel2的AND条件使定时器、计数器动作的情况下，不使用控制语法，应仅使用FB。
通过使用更改后的程序，可以使定时器、计数器动作。

■使用FOR...DO语句时

- 无法在重复变量中使用结构体构件及数组要素。
- 应使在重复变量中使用的类型与<最终值的表达式>、<增加表达式>的类型一致。
- <增加表达式>可以省略。省略的情况下<增加表达式>作为1执行。
- 如果向<增加表达式>中代入0，则FOR语法以下可能不被执行或变为无限重复。
- FOR...DO语法中FOR语句中的<语句…>在执行后进行重复变量的计数处理。超过重复变量的数据类型的最大值或低于最小值执行计数处理的情况下，发生无限重复。

■使用上升执行指令、下降执行指令时

以下显示在IF语句和CASE语句中使用上升执行指令、下降执行指令时的动作。

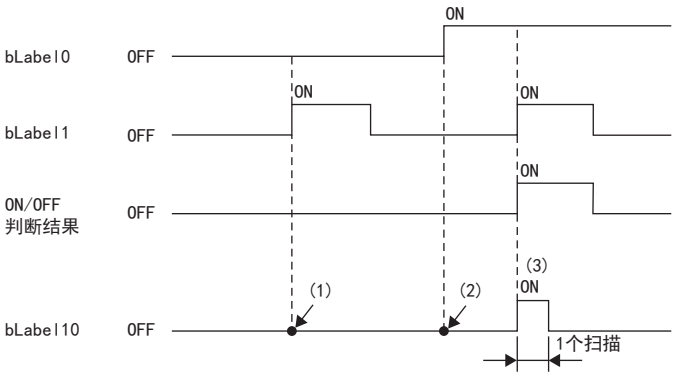
条件			动作结果		
IF语句、CASE语句的条件式	指令执行条件(EN)	前一次扫描时的指令的ON/OFF判定结果	指令的ON/OFF判定结果	上升执行指令	下降执行指令
TRUE或CASE一致	TRUE	ON	ON	不执行	不执行
		OFF	ON	执行	不执行
	FALSE	ON	OFF	不执行	执行
		OFF	OFF	不执行	不执行
FALSE或CASE不一致	TRUE	ON	OFF	不执行	不执行*1
		OFF	OFF	不执行	不执行
	FALSE	ON	OFF	不执行	不执行*1
		OFF	OFF	不执行	不执行

*1 虽然是下降(ON→OFF)，但IF语句或CASE语句的条件不成立，因此不执行指令。

例

IF语句中使用了PLS指令(上升执行指令)时

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



(1) bLabel0 = OFF时(IF语句的条件式为FALSE)，ON/OFF判定结果为OFF，不执行PLS指令。(仍旧为bLabel10 = OFF)

(2) bLabel0 = ON(IF语句的条件式为TRUE)并且bLabel1 = OFF(指令执行条件为OFF)时，ON/OFF判定结果为OFF，不执行PLS指令。(仍旧为bLabel10 = OFF)

(3) bLabel0 = ON(IF语句的条件式为TRUE)并且bLabel1 = ON(指令执行条件为ON)时，ON/OFF判定结果为OFF→ON(上升条件成立)，执行PLS指令。(bLabel10仅1扫描为ON)

■使用主控指令时

显示主控OFF时的动作。

- 选择语句(IF语句或者CASE语句)内或者重复语句(FOR语句、WHILE语句或者REPEAT语句)内的语句为不处理。
- 选择语句或者重复语句以外时，代入语句为不处理，代入语句以外的语句为非执行处理。

例

选择语句(IF语句)内的语句

```
MC(M0,N1,M1);    //主控OFF
IF M2 THEN
  M3:=M4;        //主控OFF时为不处理，因此M3保持前一次扫描时的值
END_IF;
M20:=MCR(M0,N1);
```

例

选择语句或者重复语句以外的语句(位代入语句时)

```
MC(M0,N1,M1);    //主控OFF
M3:=M4;          //主控OFF时为不处理，因此M3保持前一次扫描时的值
M20:=MCR(M0,N1);
```

例

选择语句/重复语句以外的语句 (OUT指令时)

```
MC(M0,N1,M1);    //主控OFF
OUT(M2,M3);      //主控OFF时为非执行处理，因此M3为OFF
M20:=MCR(M0,N1);
```

常数

常数的标记方法

ST程序中字符串的标记方法如下所示。

数据类型		标记方法	标记示例
字符串(32)	STRING	将字符串(ASCII、移位JIS)用单引号(')括起来。	Stest := 'ABC';
字符串[Unicode](32)	WSTRING	将Unicode字符串用双引号(")括起来。	Stest := "ABC";

上述以外的常数的标记方法请参阅下述内容。

☞35页 常数

标签与软元件

指定方法

在ST程序中可以直接记述并使用标签与软元件。标签与软元件可以在表达式的左边、右边、通用函数/功能块的自变量、返回值等中使用。

关于可使用的标签请参阅下述内容。

☞28页 标签

关于可使用的软元件请参阅下述内容。

📖MELSEC iQ-F FX5用户手册(应用篇)

■附带类型指定的软元件标记

字软元件通过向软元件名附加软元件型指定符，可以作为任意的数据类型在ST语言内使用。

软元件型指定符	数据类型	示例	示例的说明
无	总称数据类型ANY16 在算术运算公式等中只使用软元件的情况下，为字[带符号]。 但是在FUN/FB的自变量部分中作为无类型指定的软元件被指定的情况下则为自变量定义的数据类型。	D0	D0中不带类型指定符的情况下
:U	字[无符号]/位串[16位]	D0:U	将D0作为字[无符号]/位串[16位]的值
:D	双字[带符号]	D0:D	将D0、D1作为双字[带符号]的值
:UD	双字[无符号]/位串[32位]	D0:UD	将D0、D1作为双字[无符号]/位串[32位]的值
:E	单精度实数	D0:E	将D0、D1作为单精度实数的值

可以使用软元件类型指定符的软元件如下所示。

- 数据寄存器(D)
- 链接寄存器(W)
- 模块访问软元件(U□\G□)
- 文件寄存器(R)

■软元件的指定方法

关于软元件的指定可以使用下述方法。

- 变址修饰
- 位指定
- 位数指定
- 间接指定

关于详细内容，请参阅以下内容。

📖MELSEC iQ-F FX5用户手册(应用篇)
📖MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)

注意事项

- 在ST程序中无法使用指针型。
- 使用位数指定代入的情况下，应使右边和左边的数据类型相一致。

例

```
D0 := K5X0;
```

在上述情况下，因为K5X0为双字型、D0为字型，程序出错。

- 使用位数指定代入的情况下，右边>左边时，在左边的对象点数范围内进行数据传送。

例

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

在上述情况下，因为K5X0的对象点数20点，向K5X0代入1101_1111_0111_0011_0001(20位)。

- 将计数器(C)、定时器(T)、累计定时器(ST)的当前值(TNn等)在字[无符号]/位串[16位]以外的类型中使用时，或将长计数器(LC)的当前值(LCNn等)在双字[无符号]/位串[32位]以外的类型中使用时，应使用类型转换函数。

例

```
varInt := WORD_TO_INT(TN0); (*使用类型转换函数*)
```

注释

可以在ST程序中使用的注释如下所示。

注释形式	注释符号	内容	记述示例
单一行注释	//	将从开始符号“//”到行尾的内容作为注释。	// 注释内容
多行注释	(* *)	将从开始符号“(*”到结束符号“*)”的内容作为注释。 可以在注释中输入换行。	■无换行 (* 注释内容 *) ■有换行 (* 第1行注释内容 第2行注释内容 *)
	/* */	将从开始符号“/*”到结束符号“*/”的内容作为注释。 可以在注释中输入换行。	■无换行 /* 注释内容 */ ■有换行 /* 第1行注释内容 第2行注释内容 */

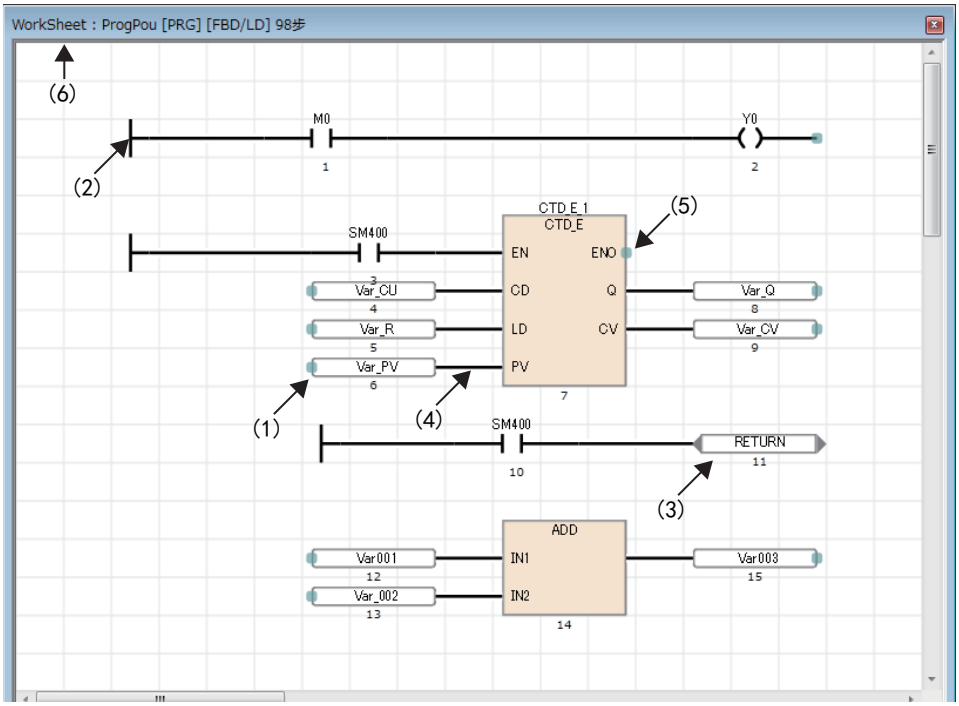
在多行注释中请勿记述含有结束符号的注释。

7 FBD/LD语言

是通过将实施特定处理的块、变量、常数沿着数据和信号的流动进行连接，创建程序的语言。

7.1 配置

使用FBD/LD语言可以创建下述的程序。



(1) FBD部件
(2) LD部件
(3) 通用部件
(4) 连接线
(5) 连接点
(6) 工作表

在FBD/LD语言的程序中，数据从功能块(FB)、函数(FUN)、变量部件(标签和软元件)、常数部件的输出点流至其他、变量部件等的输入点。

程序部件

FBD部件

显示构成FBD/LD程序的部件。

要素	符号	说明
变量	D0 bLabel1	为了存储各个值(数据)，会使用变量。变量中事先规定了数据类型，仅存储该数据类型的值(数据)。 可在变量中指定标签或者软元件。
常数	100	输出所指定的常数值。
函数(FUN)	ADD IN1 IN2	执行函数。 • 函数的创建方法(GX Works3操作手册) • 通用函数(MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇))
功能块(FB)	CTU_1 CTU CU Q R CV PV	执行FB。 • FB的创建方法(GX Works3操作手册) • 通用FB(MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)) • 模块FB(MELSEC iQ-F FX5 CPU模块FB参考)

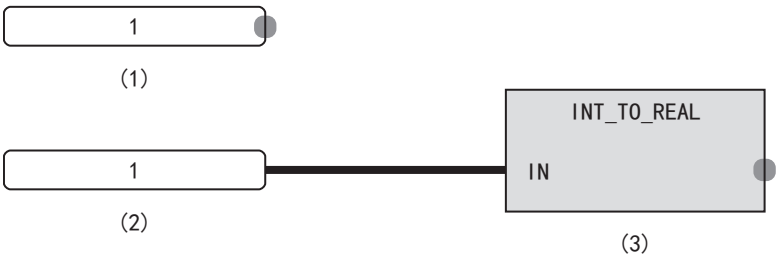
■常数部件的数据类型

常数部件时，在输入常数值时，不决定常数值的数据类型。在利用连接线连接了常数部件与FBD部件时，会决定数据类型。常数值的数据类型与利用连接线连接的目标FBD部件相同。

例

常数值中输入了1时

数据类型候补中存有BOOL型、WORD型、DWORD型、INT型、DINT型、以及REAL型，因此不能决定数据类型。利用连接线连接常数部件和FBD部件后，会成为连接目标的部件的输入点的数据类型。



(1) 数据类型未决定
(2) INT型
(3) INT型

■数据类型的自动转换

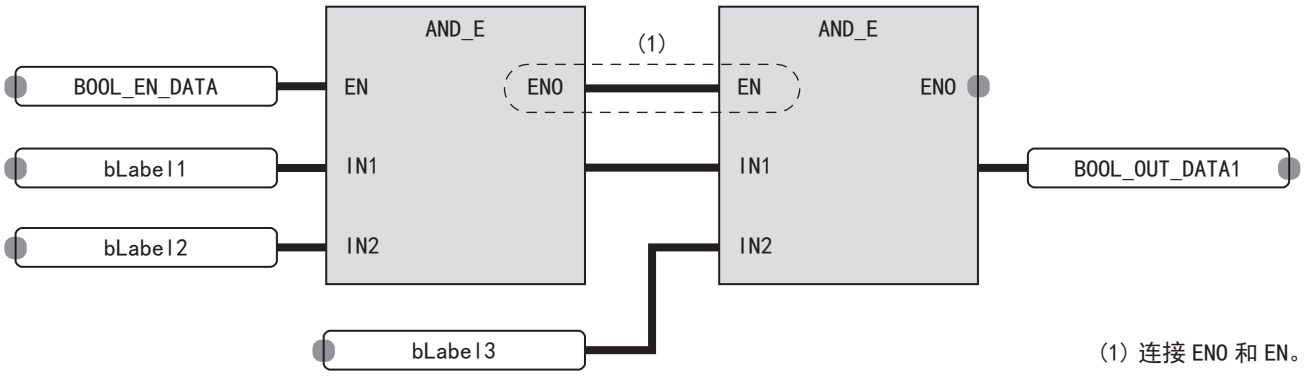
要连接的部件的数据类型不同时，有时会自动转换数据类型。

为确保在类型转换时数据不丢失，应仅从容量小的数据类型向容量大的数据类型进行类型转换。FBD/LD语言时数据类型的自动转换与ST语言的动作相同。关于详细内容，请参阅以下内容。

☞46页 数据类型的自动转换

■函数的输入输出点

- 函数必须事先将输入点与所有其他FBD部件连接。
- 函数的输入变量与输出变量中，必须事先决定数据类型，并使连接至输入点和输出点的FBD部件也与其一致。
- 将CPU模块用指令，模块专用指令的输出变量(不包括ENO)连接至其他函数(或者FB)的输入变量时，请通过变量部件。
- 从附带EN的函数连接至函数的程序中，请将函数置为附带EN函数，使其成为连接ENO和EN的程序，以免函数使用不定值。



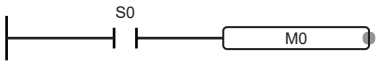
(1) 连接 ENO 和 EN。

■使用步进继电器(S)及SFC块软元件(BL)的情况

在变量部件中使用将步进继电器(S)或SFC块软元件(BL)的情况下，有可能变为转换出错状态。该情况下，应将变量部件替换为触点部件。

例

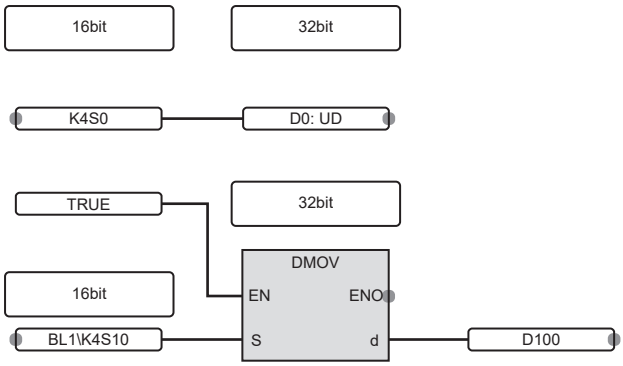
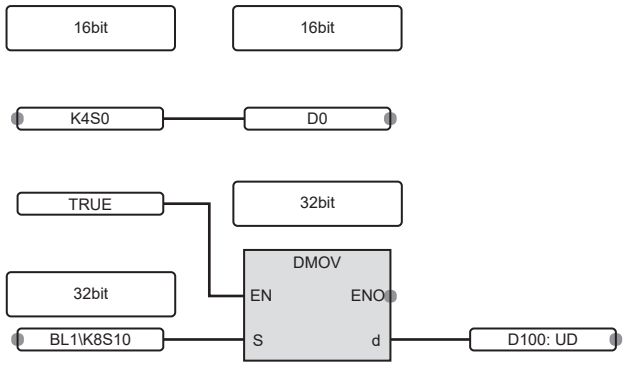
替换示例如下所示。

替换前	替换后
	

此外，在程序中使用步进继电器(S)或带块指定步进继电器(BL□\S□)的位数指定的情况下，应指定正确的数据容量。步进继电器(S)及带块指定步进继电器(BL□\S□)不是数据类型自动转换的对象，因此数据容量不一致的情况下，有可能变为转换出错状态。

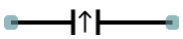
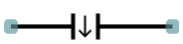
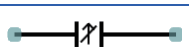
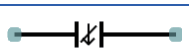
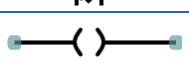
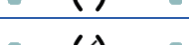
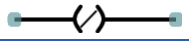
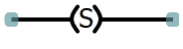
例

替换示例如下所示。

替换前	替换后
	

LD部件

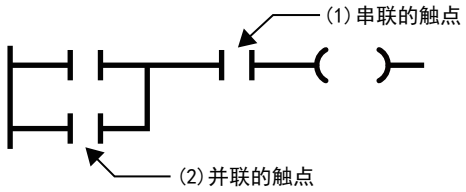
显示能够在FBD/LD语言的程序中使用的梯形图语言的部件。

要素	符号	说明
左母线		是显示母线的部件。创建梯形图回路时的起点。
常开触点		指定软元件或标签变为ON时导通。
常闭触点		指定软元件或标签变为OFF时导通。
上升沿脉冲		指定软元件或标签上升沿时(OFF→ON)导通。
下降沿脉冲		指定软元件或标签下降沿时(ON→OFF)导通。
非上升沿脉冲		指定软元件或标签OFF时、ON时以及下降沿时(OFF→ON)导通。
非下降沿脉冲		指定软元件或标签OFF时、ON时以及上升沿时(ON→OFF)导通。
线圈		将运算结果输出到指定软元件或标签中。
取反型线圈		运算结果为OFF时，指定软元件或者标签为ON。
设置		运算结果为ON时，指定软元件或者标签为ON。 软元件或者标签为ON时，即使运算结果为OFF，也还会原样不变地保持ON。
复位		运算结果为ON时，指定软元件或者标签为OFF。 运算结果为OFF时，软元件或者标签的状态不变化。

■触点符号的AND运算和OR运算

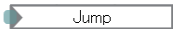
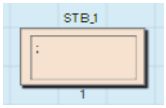
触点符号会相应回路图的连接状态实施AND运算、OR运算，并反映至运算结果中。

- 串联(1)时，与此前的运算结果进行AND运算，并将其作为运算结果。
- 并联(2)时，与此前的运算结果进行OR运算，并将其作为运算结果。



通用部件

显示配置在FBD/LD编辑器上的通用部件。

要素	符号	说明
跳转*1		将执行处理从跳转部件跳转至跳转标签。跳转的部分不执行。 根据对跳转部件的ON/OFF信息，控制是否实施跳转。 ON：将执行处理跳转至跳转标签。 OFF：不进行跳转，而是实施执行处理。
跳转标签*1		成为来自同一程序内的跳转指令的跳转目标。根据跳转标签以后的执行顺序的程序，执行处理。
连接器		将其作为连接线的替代来使用。 处理会移动至成对的连接器部件。 对于一个输出连接器，能够使用一个或多个输入连接器。
返回*1		中断程序上的返回部件以后的处理。用于不执行返回部件以后的程序和函数、FB的处理时。 根据对返回部件的ON/OFF信息，控制是否实施返回处理。 ON：执行返回处理。 OFF：不实施返回处理，而是实施通常的执行处理。
注释		用于记载注释时。
内嵌ST*1		在FBD/LD编辑器内显示ST程序。 通过双击插入的内嵌ST部件，可在显示ST编辑器后进行ST程序的编辑/监视。 详细内容请参阅下述内容。 ☞ 61页 内嵌ST

*1 在SFC程序的Zoom编辑器内不能使用。

■跳转部件的注意事项

- 通过跳转部件使线圈为ON的定时器跳转时，无法实施正常的测量。
- 能够将跳转标签配置在跳转部件的上侧(执行顺序为前)。此时，请包含从重复中抽出的方法在内创建程序，以免超出监视时钟的设置值。
- 跳转部件和跳转标签中，仅能指定指针型的局部标签。而且不可以使用结构体的构件。
- 不能使用指针分支指令(CJ)。跳转时，请使用跳转部件。
- 不能向程序块的外侧跳转或从外侧跳转。以下显示不能执行的有关跳转的动作。

- 向程序块的外侧跳转*1
- 来自程序块的外侧跳转*1
- 调用子程序
- 作为子程序进行调用

*1 包括通过BREAK指令进行的分支。

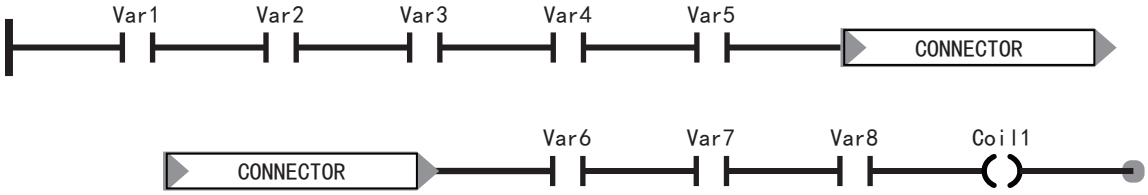
■返回部件的动作

根据使用的程序和函数、FB，返回部件的动作会有所不同。

使用的程序部件	内容
程序	结束程序部件的执行。
函数	结束函数，并且返回调用了函数的指令的下一步。
FB	结束FB，并且返回调用了FB的指令的下一步。

■关于连接器部件

想要在FBD/LD编辑器的显示范围内或者打印范围内配置程序时，会使用连接器部件。



连接线

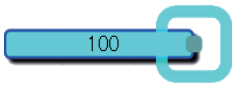
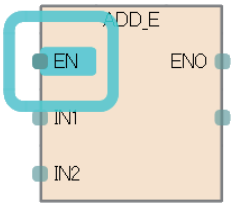
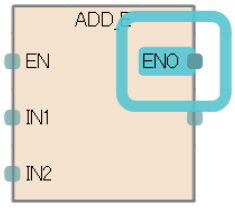
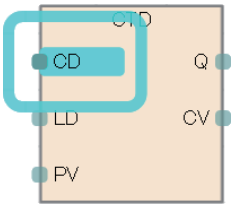
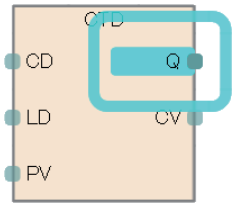
是在FBD部件、LD部件、通用部件的连接点间进行连接的线。

连接部件，将数据从左端交向右端。所连接的部件的数据类型必须一致。

连接点

是通过连接线连接FBD部件、LD部件、通用部件时的端点。

各部件的左侧的点表示输入侧，右侧的点表示输出侧。

部件	输入连接点	输出连接点	部件	输入连接点	输出连接点
触点			线圈		
变量			常数	—	
函数		 函数中不显示返回值。	FB		

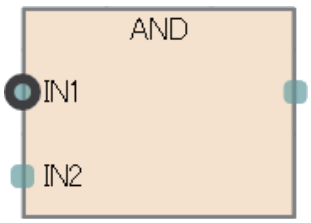
连接点在连接后会隐藏。

■输入输出点的取反

能够在连接点将至部件的输入或者来自部件的输出取反。

取反状态的连接点会被黑圈包围，并将输入或输出的数据取反(FALSE→TRUE或者TRUE→FALSE)。

可取反的数据类型为BOOL、WORD、DWORD、ANY_BIT、ANY_BOOL。



工作表

工作表是用来插入程序部件并连接的作业领域。

常数

常数的标记方法

FBD/LD程序中字符串的标记方法如下所示。

数据类型		标记方法	标记示例
字符串(32)	STRING	将字符串(ASCII、移位JIS)用单引号(')括起来。	'ABC'
字符串[Unicode](32)	WSTRING	将Unicode字符串用双引号(")括起来。	"ABC"

上述以外的常数的标记方法请参阅下述内容。

☞35页 常数

标签与软元件

指定方法

在FBD/LD程序中可以直接记述并使用标签与软元件。标签与软元件可以在部件的输入点、输出点、通用函数/功能块的自变量、返回值等中使用。

关于可使用的标签请参阅下述内容。

☞28页 标签

关于可使用的软元件请参阅下述内容。

MELSEC iQ-F FX5用户手册(应用篇)

■附带类型指定的软元件标记

字软元件通过向软元件名附加软元件型指定符，可以作为任意的数据类型使用。不指定数据类型时，作为字[带符号](INT)进行动作。

关于可与软元件型指定符使用的软元件，请参阅下述内容。

☞53页 附带类型指定的软元件标记

不指定字软元件的数据类型时，根据软元件的种类来决定数据类型。

字软元件	数据类型
定时器软元件的当前值(TN)、累计定时器软元件的当前值(STN)、计数器软元件的当前值(CN)	WORD
长计数器软元件的当前值(LCN)	DWORD
上述以外	INT

注意事项

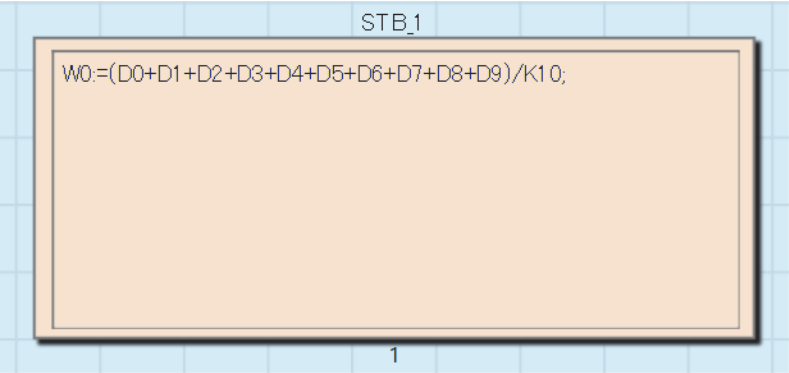
■使用标签时

- 不能使用数组的下标中名称的末尾为"_"的标签。但是，将用于下标中的软元件/标签代入到其他软元件/标签中，并且将代入目标的软元件/标签指定为下标后，即可使用。
- 不能指定名称的末尾为"_"的标签(结构体、FB)的构件。
- 不能在名称的末尾为"_"的标签(数组)中指定下标。

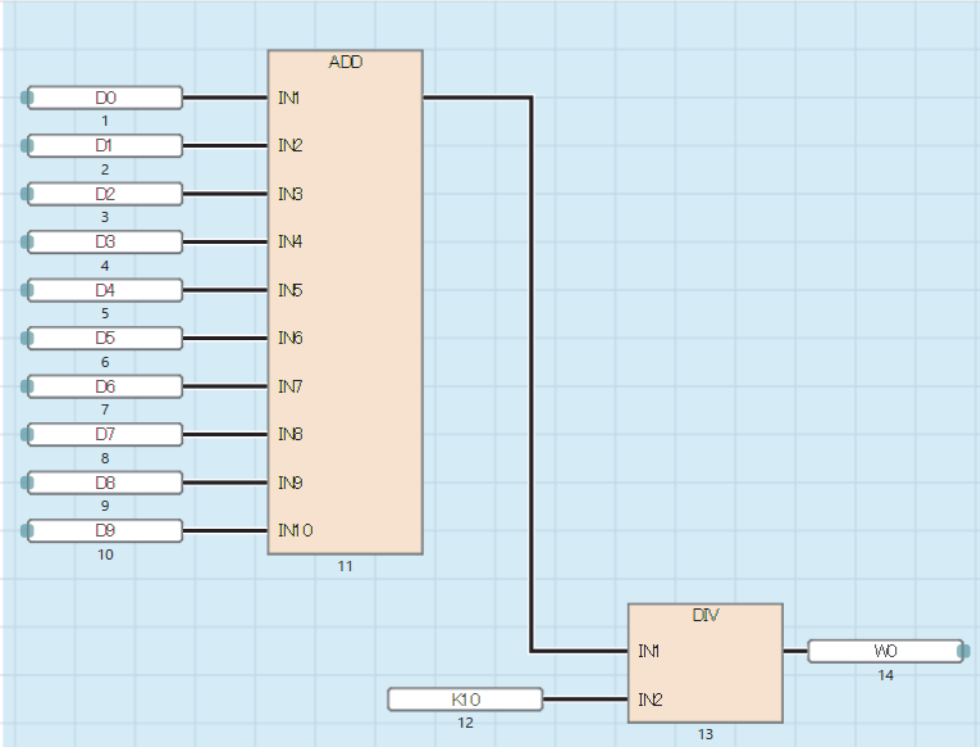
7.2 内嵌ST

内嵌ST是指在FBD/LD编辑器内创建并编辑/监视ST程序显示用内嵌ST部件的功能。
由此可以轻松地在FBD/LD程序内创建数据运算或字符串处理。

• 使用了内嵌ST情况下的程序



• 不使用内嵌ST情况下的程序



限制事项

在SFC程序的Zoom编辑器内不能使用内嵌ST。

规格

关于内嵌ST的规格，请参阅ST语言的规格。

☞ 43页 ST语言

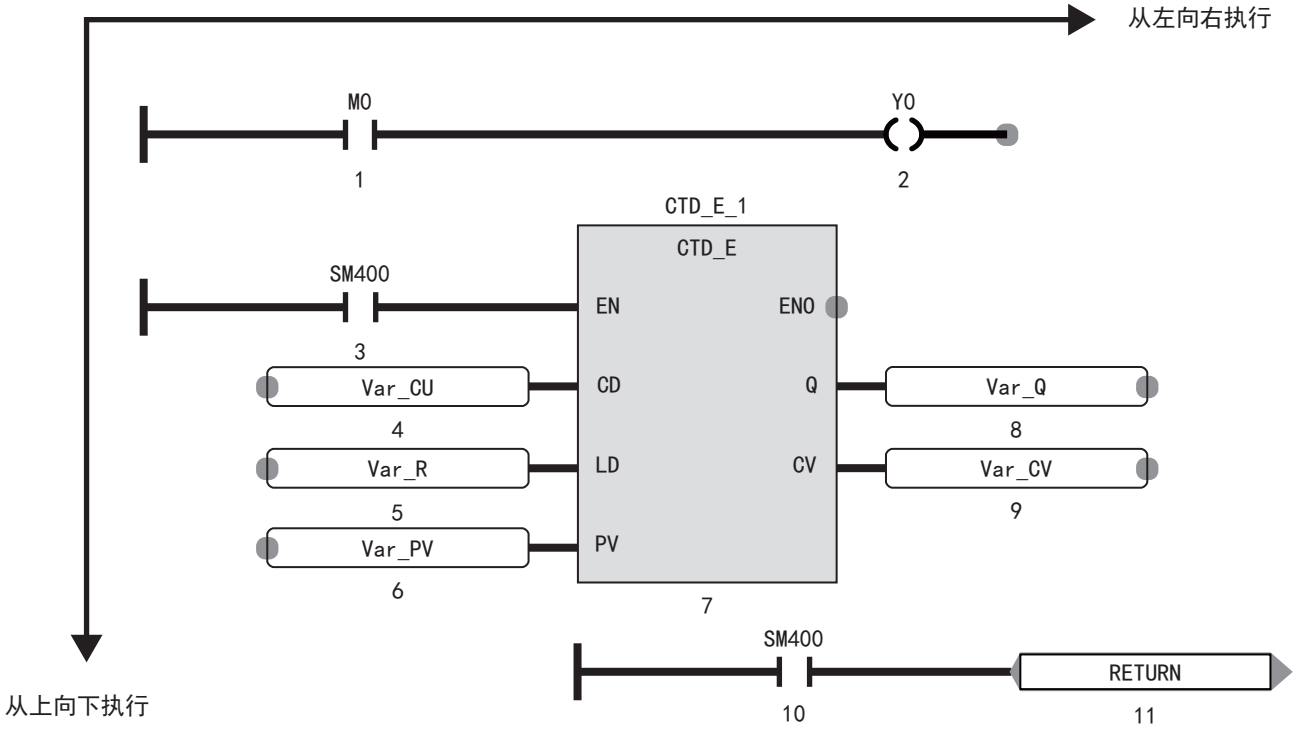
注意事项

• 可插入FBD/LD程序的一个程序部件内的内嵌ST部件的个数，最多为64个。
• 如果在内嵌ST内使用“RETURN语句”，则不会结束程序块的处理，而是结束内嵌ST部件内的处理。
• 插入FBD/LD程序中的内嵌ST部件，因为没有连接点，所以为始终被执行的状态。

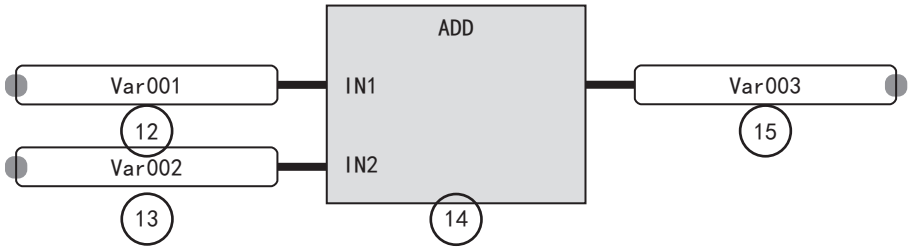
7.3 程序执行顺序

程序部件的执行顺序

根据部件的位置关系和连接状况，决定FBD/LD编辑器上的部件的执行顺序。



配置在FBD/LD编辑器上的各部件中，会显示执行顺序的编号。



8 SFC程序

是将一系列的控制动作分割为多个步，使程序的执行顺序和执行条件清楚的表达的程序记述形式。

要点

- 因为与FX3的SFC程序具有兼容性，所以可以进行FX3→FX5的配置置换。(☞ 94页 FX3兼容转移运行模式设置)
- 在本章中，将对SFC程序的动作及规格有关内容进行说明。关于本章中未记载的内容，请参阅下述手册。

📖GX Works3操作手册

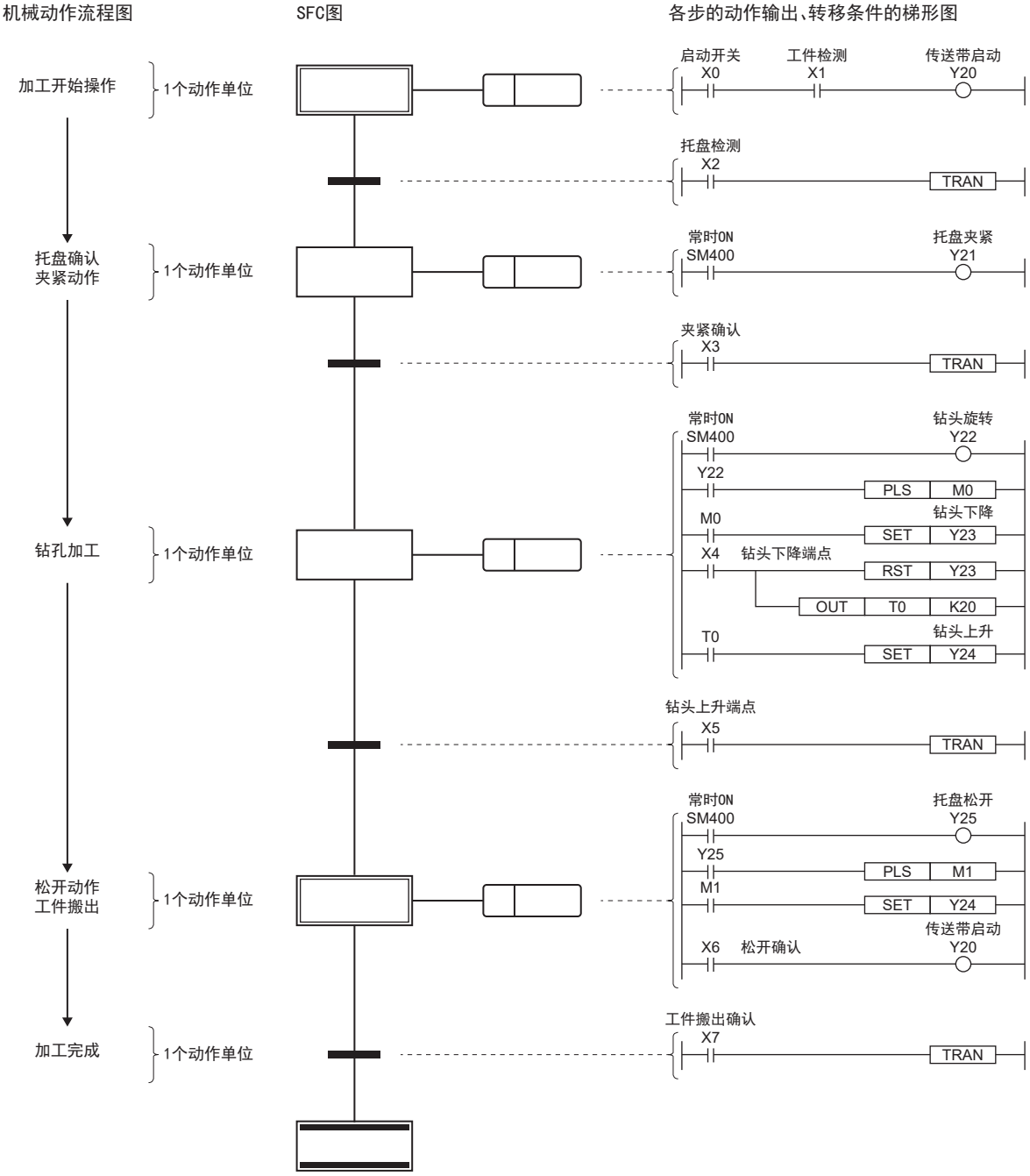
📖MELSEC iQ-F FX5用户手册(应用篇)

📖MELSEC FX3G/FX3U/FX3UC系列替换为MELSEC iQ-F系列的相关说明

限制事项

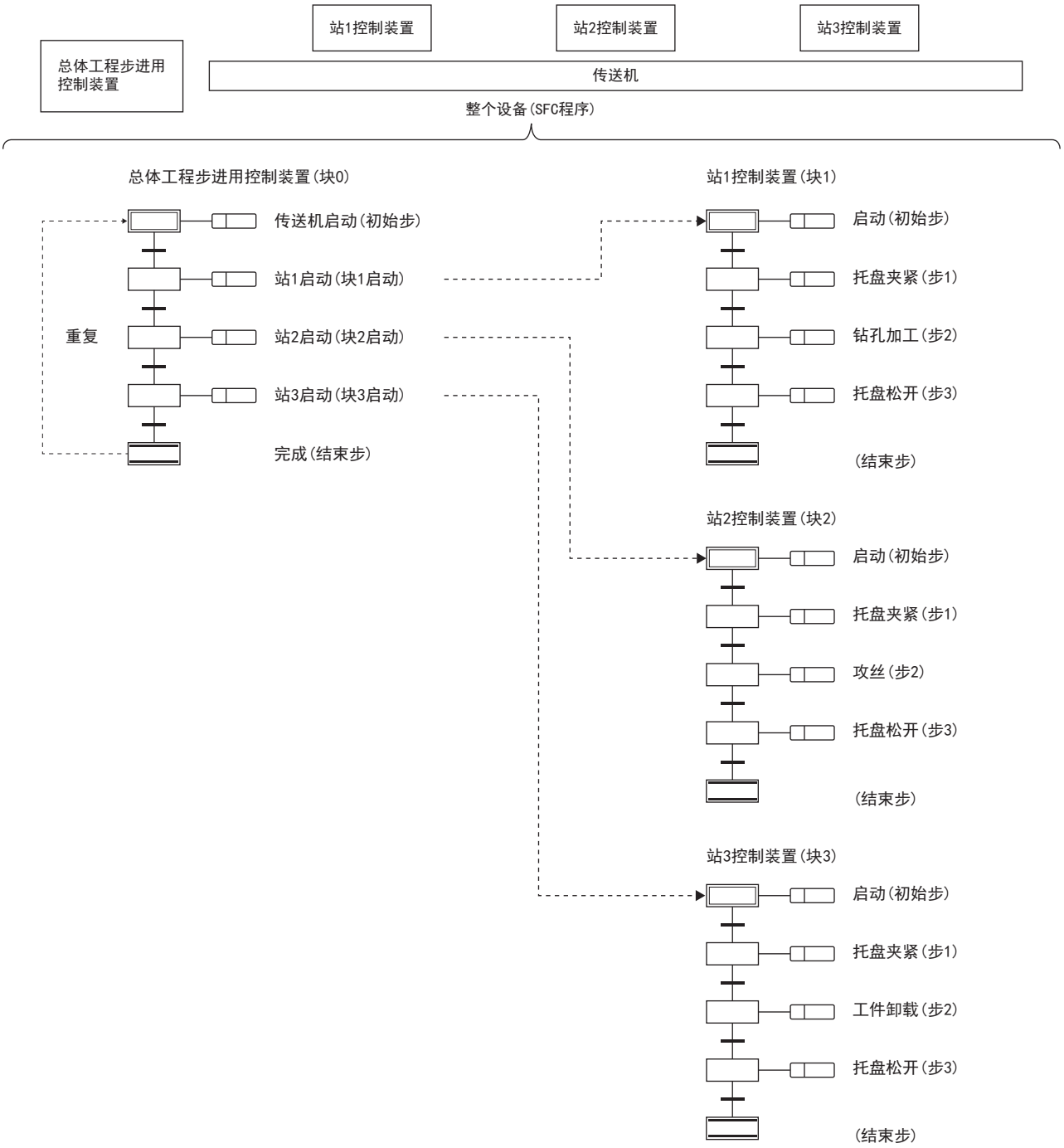
使用SFC程序时，应确认CPU模块及工程工具的版本。关于CPU模块和工程工具的版本，请参照☞ 116页 功能的添加和更改。

SFC程序将机械一系列动作的各动作单位以1个步表示。
在各步中对实际的精细控制的程序进行创建。



SFC程序从初始步开始，每当转移条件成立时按照顺序执行下一个步的动作输出，并通过结束步结束一系列的动作。

可以将整个设备、各站的机械装置、各机械的实际控制与SFC程序的各块、各步相对应。



8.1 规格

SFC程序的相关性能规格如下所示。

项目		规格
软元件点数(SFC关联)	步进继电器(S)	4096点
	SFC块软元件(BL)	32点
	SFC转移软元件(TR)	0点
SFC程序执行个数		1个
块数		32块
SFC步数		所有块最多4096步，1个块最多512步
步No.		每块0～511
分支数		最多32分支
同时激活步数		所有块和1个块最多128步
初始步数		最多1个/块
动作输出数		最多4个/步
顺控程序步数	动作输出	无限制
	转移条件	仅1个电路块
SFC块RUN中写入对象块数		1块*1

*1 关于SFC块RUN中写入的有关内容，请参阅下述内容。
☞107页 SFC块RUN中写入
此外，使用SFC块RUN中写入时，应确认CPU模块及工程工具的版本。
☞116页 功能的添加和更改

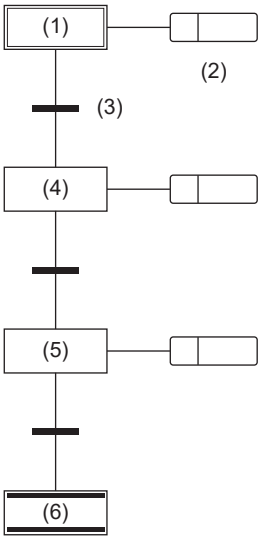
要点

关于SFC程序的处理时间有关内容，请参阅下述手册。
MELSEC iQ-F FX5用户手册(应用篇)

8.2 配置

SFC的基本动作

SFC程序从初始步开始，每当转移条件成立时执行下一个步，并通过结束步结束一系列的动作。



(1) 初始步
(2) 动作输出
(3) 转移条件
(4) 普通步
(5) 普通步
(6) 结束步

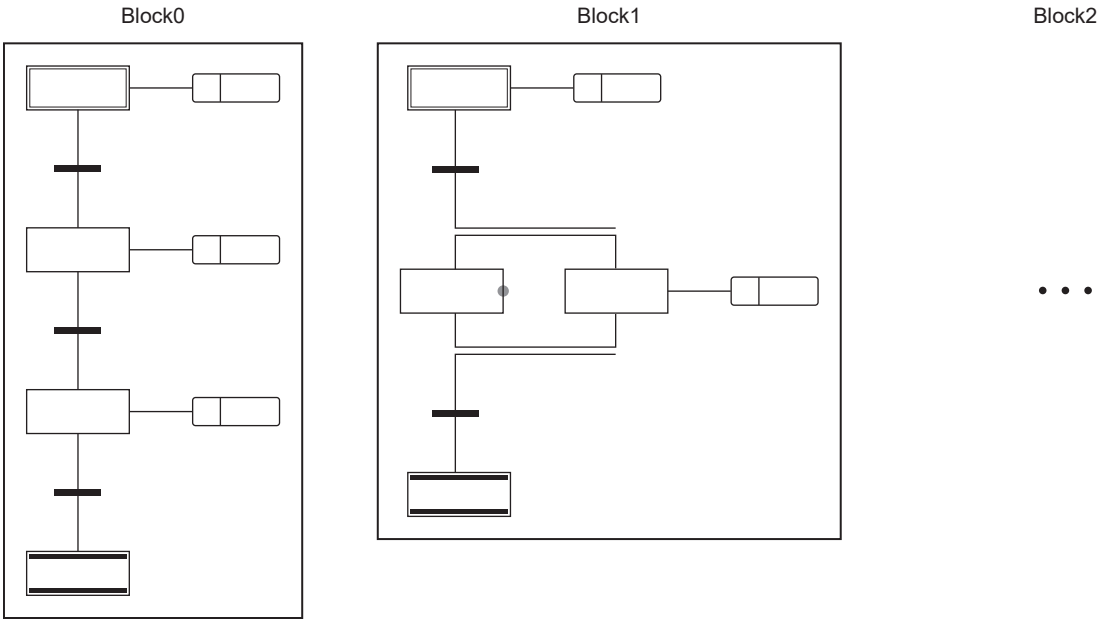
1. 块启动时，首先激活初始步(1)，执行动作输出(2)。动作输出(2)执行后检查下一个转移条件(3)是否成立。
2. 在转移条件(3)成立之前，仅执行动作输出(2)。转移条件(3)成立时将结束动作输出(2)，初始步(1)变为非激活状态，激活下一个普通步(4)。
3. 在执行普通步(4)的动作输出后，检查下一个转移条件是否成立。如果下一个转移条件成立，将重复执行普通步(4)的动作输出。
4. 转移条件成立时将结束动作输出，初始步(4)变为非激活状态，激活下一个步(5)。
5. 每当转移条件成立时都将激活下一个步，最后激活结束步(6)时结束块。

要点

- 1个步最多可创建4个动作输出。创建了多个动作输出的情况下，将由上至下按顺序执行。(☞ 78页 动作输出)
- 初始步与普通步，可以通过赋予属性更改步的类型。(☞ 69页 步的类型)

块

块由步及转移条件构成，是表示一系列动作的单位。



SFC程序内可创建的最多块数，请参阅下述章节。

☞66页 规格

在块内从初始步开始，步与转移条件进行交互，并以结束步或跳转转移结束而构成。

块将处于激活或非激活状态。

- 激活：块内存在激活步的状态
- 非激活：块内的所有步处于非激活的状态

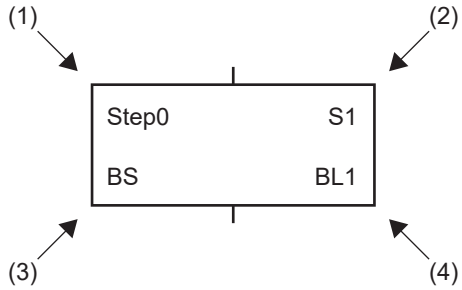
块从非激活变为激活时，初始步将变为激活，依次执行处理。(☞97页 各块的执行顺序)

要点

- 通过CPU参数的设置，仅有块0可以在SFC程序启动时自动启动。在这种情况下激活结束步结束块0时，块0将被自动再启动，再次从初始步开始执行。(☞92页 起动条件设置)
- 通过SFC控制指令(步启动)向非激活块的步发出启动请求的情况下，激活块后将从指定的步开始执行处理。

步

步是构成块的基本单位。



(1) 步名
(2) 步No.
(3) 属性
(4) 属性指定目标

每一个块可创建的最多步数，请参阅下述章节。

☞66页 规格

步有下述特征。

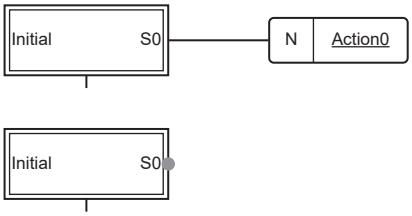
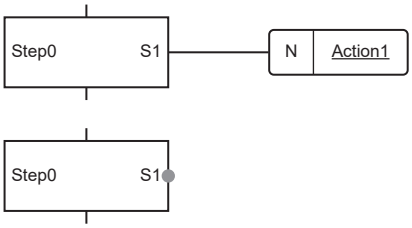
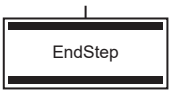
- 步激活时，执行相关的动作输出。
- 各步中，添加了步No.。步No.用于对执行步进行监视或通过SFC控制指令进行强制启动及强制停止的情况。(☞77页 至步进继电器(S)的步的分配)
- 步名及步No.在各块内是固有的。(不能为空。)

要点

可在步的属性画面，变更步名、步号、属性、属性指定目标。
选择步后，选择菜单中的[编辑]⇨[属性]，将会显示步的属性画面。(GX Works3操作手册)

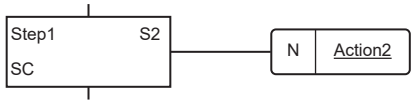
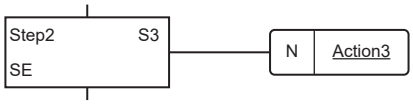
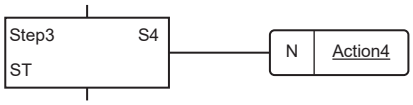
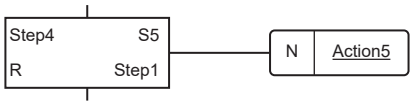
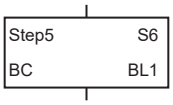
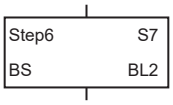
步的类型

步的类型如下所示。

项目		内容
初始步	 	表示块的起始的步。 在步处于激活状态时，始终对该步的下一个转移条件进行检查，并在转移条件成立时将激活转移到下一个步。 可以附加SC、R的属性。 步也可以不创建动作输出。
普通步	 	构成块的基本的步。 在步处于激活状态时，始终对该步的下一个转移条件进行检查，并在转移条件成立时将激活转移到下一个步。 可以附加SC、R、BC、BS的属性。 步也可以不创建动作输出。
结束步		结束块的步。 无法创建动作输出。

步的属性

步的属性如下所示。

属性	项目		内容
SC	线圈保持步[SC]		是激活转移后，仍保持动作输出为ON的线圈输出的步。
SE	动作保持步(无转移检查)[SE]		是激活转移后，仍继续执行动作输出的步。 转移条件成立且下一个步激活后，不进行转移条件的检查。
ST	动作保持步(有转移检查)[ST]		是激活转移后，仍继续执行动作输出的步。 转移条件成立且下一个步激活后，仍重复进行转移条件的检查。
R	复位步[R]		是将指定步置为非激活的步。
BC	块启动步(有结束检查)[BC]		是激活指定块的步。 指定块变为非激活且转移条件成立时，激活将转移到下一个步。 无法创建动作输出。
BS	块启动步(无结束检查)[BS]		是激活指定块的步。 转移条件成立时，激活将转移到下一个步。 无法创建动作输出。

要点

- 通过步的属性画面更改“步属性”的设置可以更改步的类型。
- 针对复位步[R]、块启动步(有结束检查)[BC]、块启动步(无结束检查)[BS]，可对属性画面“步属性指定目标”中的步名或块No.进行指定。

关于设置方法有关内容，请参阅下述手册。

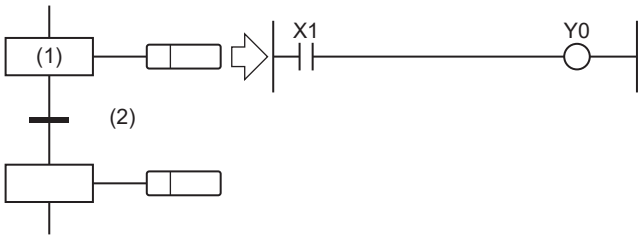
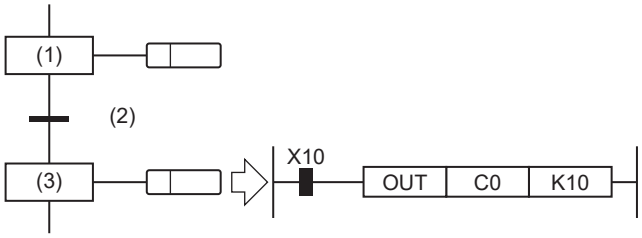
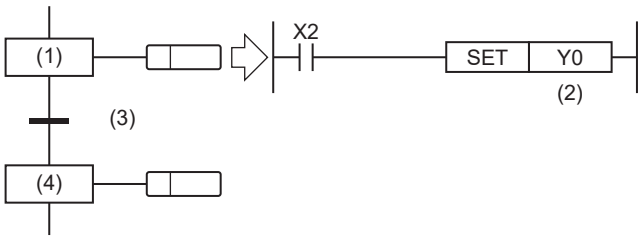
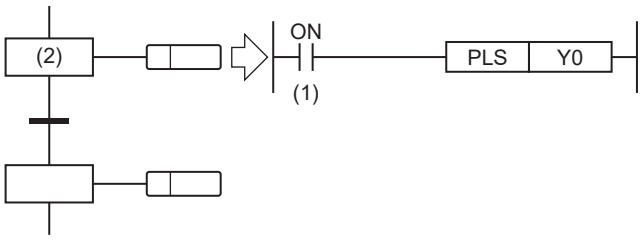
GX Works3操作手册

普通步(无属性)

构成块的基本的步。

在步处于激活状态时，始终对该步的下一个转移条件进行检查，并在转移条件成立时将激活转移到下一个步。

步的动作输出根据使用的指令，转移至下一个步时的输出状态有所不同。

项目	内容	示例
使用OUT指令时(OUT C指令以外)	转移至下一个步后由激活变为非激活时，通过OUT指令进行的输出将自动OFF。 定时器也一样，对当前值进行清除后将触点置为OFF。 但是，在ST语言的选择语句或重复语句内使用的OUT指令的输出，将不自动变为OFF。	 在步(1)的动作输出中通过OUT指令将Y0置为ON的情况下，转移条件(2)成立时Y0将自动变为OFF。
使用OUT C指令时	动作输出的计数器的执行条件已处于ON状态时，转移条件成立且使用计数器的步激活时，将被计数1次。 在执行计数器的复位指令之前激活转移至下一个步的情况下，即使使用计数器的步变为非激活状态也将保持计数器的当前值及触点的ON状态。 对计数器进行复位的情况下，在其他步中将执行RST指令。	 在步(1)激活时X10已处于ON状态的情况下，转移条件(2)成立且转移到步(3)时，计数器C0将进行1次计数。
使用SET指令、基本指令或应用指令时	激活转移至下一个步后，即使使用了指令的步变为非激活状态，也将保持ON状态或保持软元件/标签中存储的数据。 将ON状态的软元件/标签置于OFF或清除软元件/标签中存储的数据时，应在其他步中通过RST指令等进行。	 在步(1)的动作输出中通过SET指令将Y0置为ON的情况下(2)，即使转移条件(3)成立且转移到步(4)，Y0仍将保持ON。
使用PLS指令、上升沿指令时	即使执行条件的触点处于常时ON状态的情况下，每当使用指令的步从非激活状态变为激活状态时也将执行指令。	 即使执行条件触点为常时ON(1)，每当步(2)变为激活状态时也将执行PLS指令。

■无动作输出的步

不对动作输出进行创建的步，可以作为等待用的步使用。

- 在步的激活过程中，始终对转移条件进行检查，在转移条件成立后，下一个步将变为激活。
- 创建动作输出时，将作为普通的步进行动作。

初始步

初始步是各块的起始的步，1个块中只能记述1个。(☞ 66页 规格) 初始步的执行方法与初始步以外的步相同。

■块启动时的激活步

在多个初始步的情况下，块中开始启动时激活步根据启动方法而不同，如下所示。

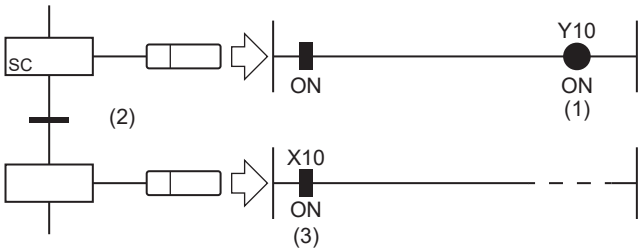
激活步的动作	启动方法
初始步全部被激活	通过块启动步启动时
	通过SFC控制指令的块启动指令启动时
	通过块0的自动启动设置启动了块0时
仅指定步被激活	通过SFC控制指令的步控制指令指定了初始步中的任意一个时

■将步属性附加到初始步

可以将SC(线圈保持)、R(复位)的各属性附加到初始步中。此外，附加了属性的情况下，除了块启动时自动激活以外，与初始步以外的步相同。此外，也可置为无动作输出的步。

线圈保持步[SC]

是激活转移后，动作输出为ON的线圈仍保持输出的步。



通过OUT指令变为ON状态的Y10(1)，即使转移条件(2)成立也不变为OFF，而是保持ON状态不变(3)。

转移条件成立，且转移至下一个步后，不进行动作输出内的运算。因此，即使动作输出内的输入条件发生变化，线圈输出的状态也不变化。

■线圈输出OFF的时机

在转移后的线圈保持步[SC]中，使保持ON的线圈输出变为OFF的时机如下所示。

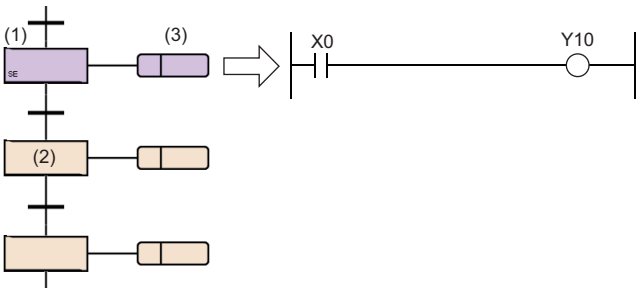
- 执行了块的结束步的情况下(SM327为ON时除外)
- 通过SFC控制指令(块结束)对块进行了强制结束的情况下
- 通过SFC控制指令(步结束)对步进行了复位的情况下
- 设置了用于复位线圈保持步[SC]的复位步[R]被激活的情况下
- 将SM321(启动/停止SFC程序)置为OFF的情况下
- 通过程序对线圈进行了复位的情况下
- 在块内的复位步[R]中指定了S999的情况下

动作保持步(无转移检查)[SE]

是激活转移后，仍继续执行动作输出的步。

转移条件成立，且转移至下一个步后，继续进行动作输出内的运算。因此，输入条件变化时线圈的状态也将变化。

转移条件成立且下一个步激活后，不进行转移条件的检查，即使转移条件再次成立，也不转移至下一个步。



从步(1)转移至步(2)时，步(1)将为保持中。
保持中时，不进行转移检查，但继续进行动作输出(3)。
此时，Y10跟随X0的ON/OFF而ON/OFF。

■变为非激活的时机

表示动作保持步(无转移检查)[SE]变为非激活的时机。

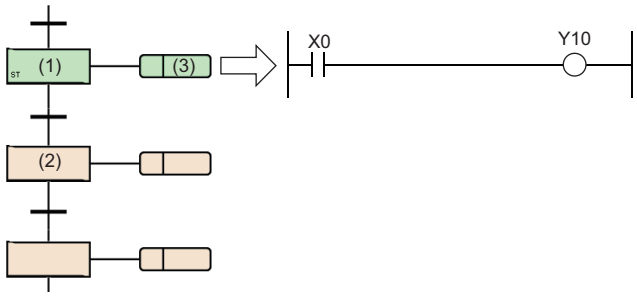
- 执行了块的结束步的情况下
- 通过SFC控制指令的RST指令(块结束)对块进行了强制结束的情况下
- 通过SFC控制指令的RST指令(步结束)对步进行了复位的情况下
- 设置了复位动作保持步(无转移检查)[SE]的复位步[R]为激活的情况下
- 将SM321(启动/停止SFC程序)置为OFF的情况下
- 在块内的复位步[R]中指定了S999的情况下

动作保持步(有转移检查)[ST]

是激活转移后，仍继续执行动作输出的步。

转移条件成立，且转移至下一个步后，继续进行动作输出内的运算。

转移条件成立且下一个步激活后，仍重复进行转移条件的检查。如果转移条件再次成立，激活下一个步的同时，继续进行动作输出内的运算。



从步(1)转移至步(2)时，步(1)将为保持中。
保持中时，也与普通的激活步相同地继续进行动作输出(3)。
此时，Y10跟随X0的ON/OFF而ON/OFF。
此外，也进行转移检查，转移条件成立时激活下一个步。

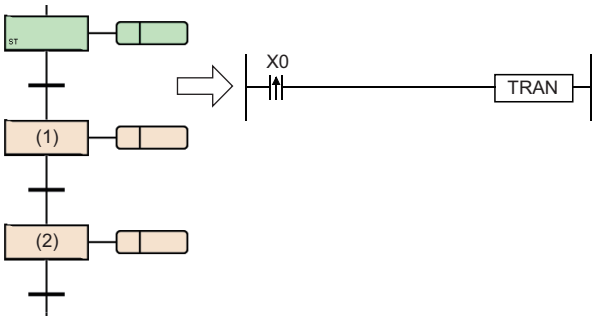
■变为非激活的时机

表示动作保持步(无转移检查)[ST]变为非激活的时机。

- 执行了块的结束步的情况下
- 通过SFC控制指令的RST指令(块结束)对块进行了强制结束的情况下
- 通过SFC控制指令的RST指令(步结束)对步进行了复位的情况下
- 设置了复位动作保持步(有转移检查)[ST]的复位步[R]为激活的情况下
- 将SM321(启动/停止SFC程序)置为OFF的情况下
- 在块内的复位步[R]中指定了S999的情况下

■注意事项

- 动作保持步(有转移检查)[ST]在之后的转移条件成立期间，每次扫描将启动下一个步。为保持每次扫描不转移，应在转移条件中使用PLS指令等上升沿执行的指令。



将转移条件设为上升沿脉冲运算开始的条件，则仅X0变为ON的瞬间的1个扫描启动步(1)。
从步(1)转移至步(2)，且即使步(1)为非激活，只要X0不会再次OFF→ON，则不会启动步(1)。

- SM328(END步到达时清除处理模式)为ON时，应始终保持动作保持步(有转移检查)[ST]之后的转移条件不成立。下一个步始终为非保持的激活状态，因此块无法结束。

复位步[R]

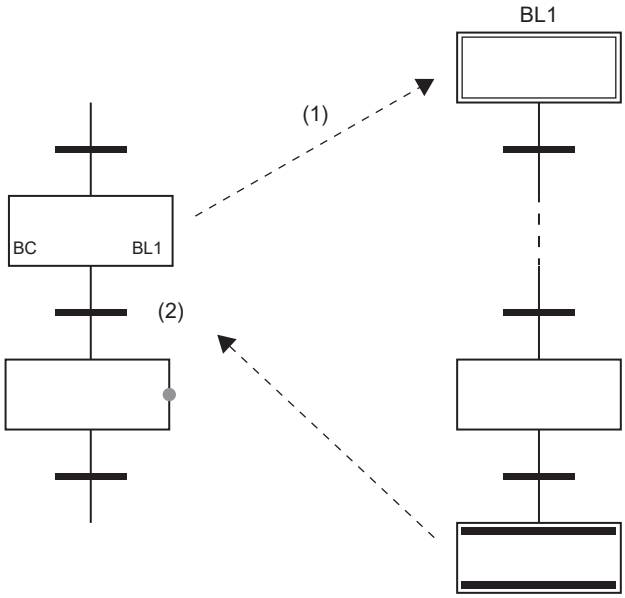
是将指定步置为非激活的步。

- 复位步[R]在执行每个扫描动作输出之前，将本块内的指定步置于非激活状态。除复位指定步以外与普通的步(无属性)相同。
- 对指定步No.,可以指定线圈保持步[SC、SE、ST]]的步No.或指定为S999。
- 指定的步No.为S999的情况下，本块内保持中的保持步[SC、SE、ST]将全部置为非激活。此时，可置为非激活的仅为保持中的保持步[SC、SE、ST]。动作保持步[SE、ST]在非保持状态下动作时不成为非激活对象。
- 本步No.无法指定为指定步No.。

块启动步(有结束检查)[BC]

是激活指定块的步。

指定块变为非激活且转移条件成立时，激活将转移到下一个步。



块启动步(有结束检查)[BC]被激活时，对块(BL1)进行启动(1)。
在启动目标块(BL1)的执行结束后变为非激活状态之前将处于无处理状态，不进行转移条件(2)的检查。
在块(BL1)的执行结束后变为非激活状态时，仅进行转移条件(2)的检查，如果转移条件(2)成立则转移到下一个步。

对1个块同时进行启动或对已启动的块进行启动时，按照块冗余启动时的运行设置进行。(☞95页 块冗余起动时的运行设置)
可指定的块仅1个。同时启动多个块的情况下，使用并联分支后，再使用多个块启动步。

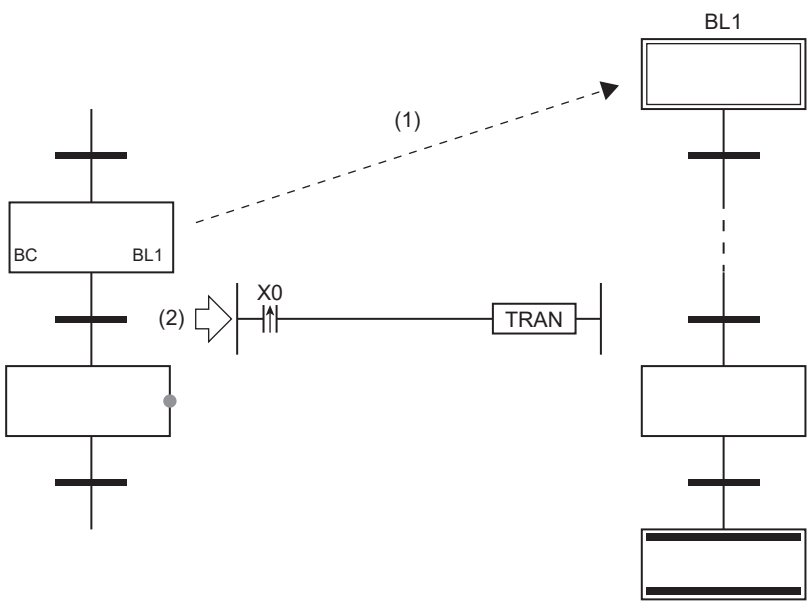
■注意事项

- 不可以将动作输出创建到块启动步(有结束检查)[BC]中。
- 在并联合并的合并之前不可以创建块启动步(有结束检查)[BC]。如需在并联合并的合并之前进行创建，请使用块启动步(无结束检查)[BS]。

块启动步(无结束检查)[BS]

是激活指定块的步。

转移条件成立时，激活将转移到下一个步。



在块启动步(无结束检查)[BS]对块(BL1)进行了启动(1)之后，仅进行转移条件(2)的检查，如果条件成立则不等待启动目标块(BL1)的结束，而转移至下一个步。

对1个块同时进行启动或对已启动的块进行启动时，按照块冗余启动时的运行设置进行。(☞95页 块冗余起动时的运行设置)

可指定的块仅1个。同时启动多个块的情况下，使用并联分支后，再使用多个块启动步。

■注意事项

不可以将动作输出创建到块启动步(无结束检查)[BS]中。

结束步

结束块的步。

- 激活转移到结束步，块内不存在保持中以外的激活步时，将块内的全部保持中步置为非激活后，结束块。
- 块内存在保持中以外的激活步的情况下，根据SM328(END步到达时清除处理模式)的状态进行下述处理。

SM328的状态	内容
OFF(默认)	进行清除处理。 将块内剩余的激活步全部强制结束后，结束块。
ON	不进行清除处理。 以END步到达时的状态继续执行块，块不结束。

- 在执行清除处理时，将通过OUT指令将线圈输出全部置为OFF。但是，关于保持中步的线圈输出，根据SM327(END步执行时的输出)的状态进行下述处理。

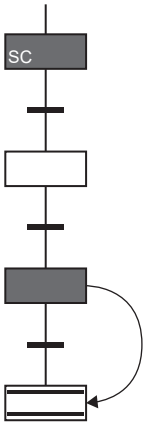
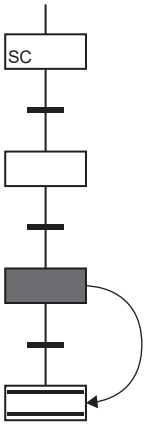
SM327的状态	内容
OFF(默认)	将保持中步的输出全部置为OFF。
ON	将保持中步的输出全部保持。 SM327设置仅对保持中的线圈保持步[SC]的有效。将转移条件未成立，不在保持中的线圈保持步[SC]的输出全部置为OFF。此外即使SM327为ON时，步也将由激活状态变为非激活状态。 但是，通过块结束指令等进行强制结束的情况下，所有步的线圈输出将变为OFF。

- 块结束后再次启动块的方法如下所示。

项目		内容
块0	在参数的SFC设置中将“启动条件设置”设置为“自动启动块0”	自动再次激活初始步，重复执行处理。
	在参数的SFC设置中将“启动条件设置”设置为“不自动启动块0”	通过下述方法，在有对指定块的启动请求时进行再启动。 • 其它块中将块启动步激活。 • 执行SFC控制指令(块启动)。
块0以外的所有块		

注意事项

- 不可以将动作输出创建到结束步中。
- 仅激活转移到结束步中的情况下，SM327(END步执行时的输出)的设置将变为有效。通过SFC控制指令(块结束)等进行强制结束的情况下，所有步的线圈输出将置为OFF。
- 在激活转移到结束步时，仅有保持步剩余的情况下，即使SM328(END步到达时清除处理模式)为ON，该保持中步也将变为非激活。不将保持步的线圈输出置为OFF的情况下，请将SM327置为ON。SM328和线圈保持步[SC]的动作关系如下所示。

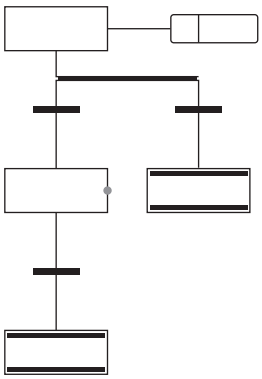
普通的激活步有剩余或转移不成立的线圈保持步[SC]有剩余的情况(非保持)	保持中的激活步剩余的情况
	
• SM328为OFF时，进行清除后结束块。 • SM328为ON时，不进行清除，而是继续进行处理。	与SM328的设置无关，进行清除后结束块。

- SM328为ON的情况下，通过块启动步被启动的块，在非保持的激活步不存在于块内时，返回到原来的块处理。
- 应始终保持动作保持步(有转移检查)[ST]之后的转移条件不成立。动作保持步(有转移检查)[ST]之后的转移条件始终成立的情况下，下一个步始终为激活状态，因此SM328为ON时块无法结束。
- “FX3兼容转移运行模式设置”为有效时，将CPU模块的电源置为OFF→ON，或在复位时将SM328置为ON。

要点

在SFC图内可以创建多个结束步。

对选择分支中的步进行选择后，通过选择菜单的[编辑]⇨[更改]⇨[结束步/跳转]，可以创建多个结束步。



至步进继电器(S)的步的分配

步进继电器是SFC程序中的各步对应的软元件。如果步处于激活中(也包括停止中、保持中)将变为ON，如果处于非激活状态将变为OFF。

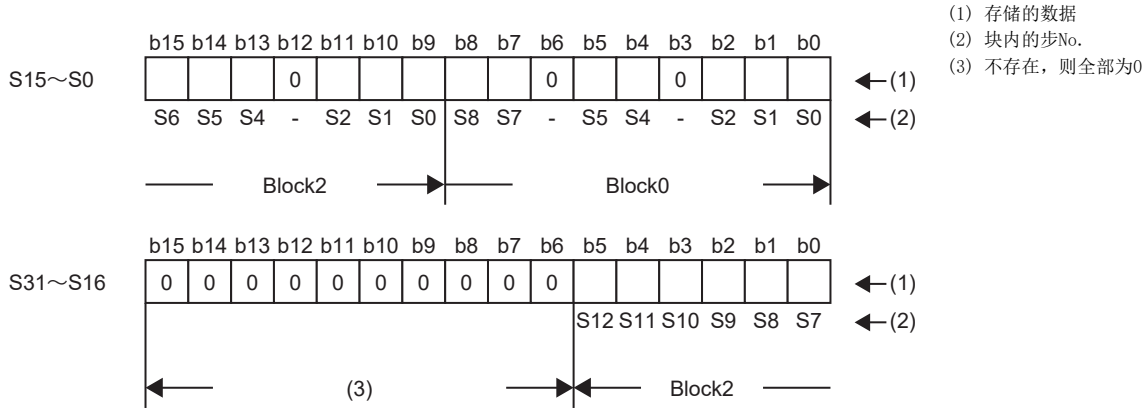
步进继电器按下述方式分配。

- 从SFC程序的块0开始按块No.顺序，在1个块内按步No.顺序从起始开始向末尾分配步进继电器。
- 不分配步进继电器到不存在的块No.中。
- 在1个块内，步进继电器被分配到缺少编号的步No.中。该位始终变为OFF。
- 在最后的块分配完了以后，步进继电器以后所有的位都将变为OFF。

例

下述的块配置时的步进继电器分配如下所示。

- Block0：最大步No.为8，步No.3及步No.6不存在。
- Block1：不存在。
- Block2：最大步No.为12，步No.3不存在。
- Block3及其以后：不存在。



要点

可自由对各步(结束步除外)分配步No.。

- 步No.中缺少编号时，可创建的最多步数会变少，因此应尽可能以小编号顺序创建。
- 在最上行的左端初始步中，只能使用No.0(S0)。

对块的最初初始步分配步No.0。

每一个块可使用的步No.，请参阅下述章节。

☞66页 规格

不可以对进行超出上限的步No.分配。此外在同一块内，步No.不可以重复。但是，在不同的块中可以使用同一步No.。

指定其他块的步进继电器，请按下述形式进行指定。

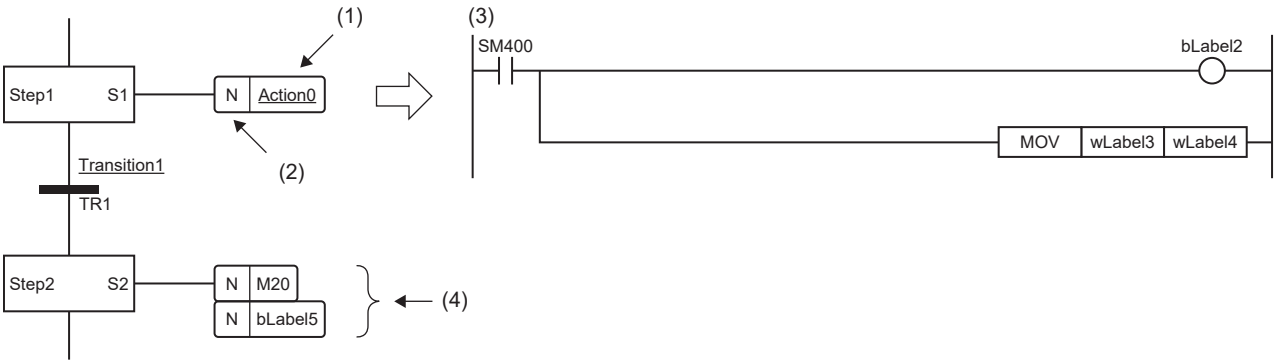
例

指定块No.12的步No.23

程序类型		软元件表记	内容
SFC程序	同一块内	S23	指定同一块内的步的情况下，可以省略块名。
	块12以外	BL12\S23	指定块No.及步No.。
SFC程序以外的顺控程序	指定当前对象块	S23	指定对象块内的步的情况下，可以省略块名。
	指定与当前对象块不同的块	BL12\S23	指定块No.及步No.。

动作输出

动作输出表示在步为激活状态时被执行的程序。



(1) 动作输出名
(2) 修饰语*1
(3) 动作输出的详细表示
(4) 动作输出的标签/软元件

*1 N表示步在激活状态时被执行。不可以设置N以外。

步进行激活时，每个扫描中将执行动作输出。步变为非激活时将结束动作输出，在步变为激活之前保持非执行。

1个步最多可创建4个动作输出。创建了多个动作输出的情况下，将由上至下按顺序执行。

动作输出的详细表示，可以通过梯形图语言、ST语言、FBD/LD语言创建。

要点

关于详细表示及标签/软元件的详细内容，请参阅下述手册。

GX Works3操作手册

无法使用的指令

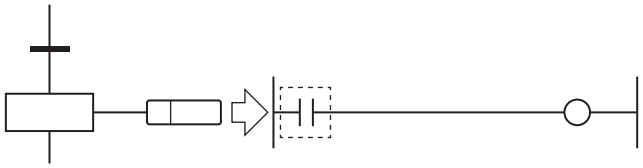
在动作输出中，有一部分的指令无法使用。无法使用的指令如下所示。

分类	指令符号
主控制指令	MC*1
	MCR*1
结束指令	FEND
	END
程序分支指令	CJ*1
	GOEND
程序执行控制指令	IRET
结构化指令	BREAK*1
	RET
	SRET
转移条件虚拟输出	TRAN
步梯形图指令	STL
	RETSTL
初始状态	IST

*1 在动作输出内的函数/功能块内，可以使用。

要点

必须在详细表示内的梯形图中创建各指令输入条件的触点。



■限制事项

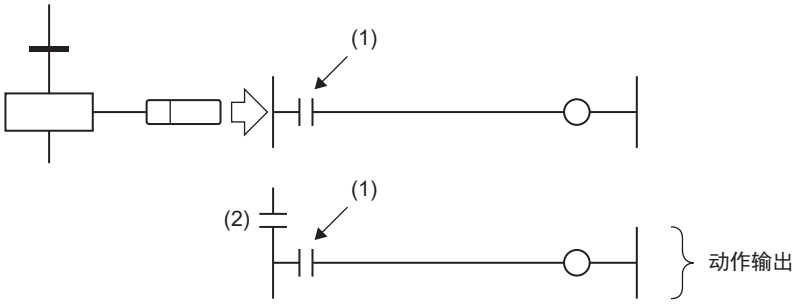
根据创建动作输出的程序语言，有下述限制。

语言		内容
梯形图语言	详细表示	不可以将指针及中断指针输入到指针输入区域中。 ■不可以使用的函数/功能块 • 含有动作输出中无法使用的指令的函数/功能块 • 含有指针的函数/功能块 • “使用MC/MCR控制EN”设置为“是”，且“使用EN/ENO”设置为“否”的宏型功能块
ST语言		☞43页 ST语言
FBD/LD语言		☞55页 FBD/LD语言

注意事项

• 步的动作将变为与下述梯形图大致相同的动作。关于动作输出的执行，请参阅☞98页 动作输出的执行。

(1) 各指令的输入条件
(2) 显示步的状态的触点(激活时：ON，非激活时：OFF)



• 在步的动作输出内进行子程序调用的情况下，如果使用CALL指令，即使转移条件成立，步也变为非激活状态，CALL目标的输出也不变为OFF。在转移条件成立，步变为非激活状态时，如希望将CALL目标的输出置为OFF，请使用XCALL指令。

• 即使动作输出内的输入条件常时为ON，在步为非激活时，输入条件将被视为OFF。因此，步变为激活状态之后，将在OFF→ON的条件下执行指令。例如，PLS指令及INCP指令等的上升沿指令中，将输入条件置为常时ON的情况下，每当步被激活时则执行指令。

• 通过动作输出内的OUT C指令、SET指令、基本指令或应用指令等变为ON的软元件，即使步变为非激活，动作输出结束也不会变为OFF。将软元件置为OFF时需要另外执行RST指令等。

• 通常，PLS指令及PLF指令指定的软元件仅在1个扫描中变为ON，并在其之后变为OFF，但是在线圈保持步[SC]的转移成立的同时，指定软元件变为ON的情况下，将继续保持ON状态。在这种情况下，可通过将线圈保持步[SC]的线圈输出置为OFF，或通过再次激活步，使得软元件变为OFF。关于线圈输出为OFF的条件，请参阅下述内容。

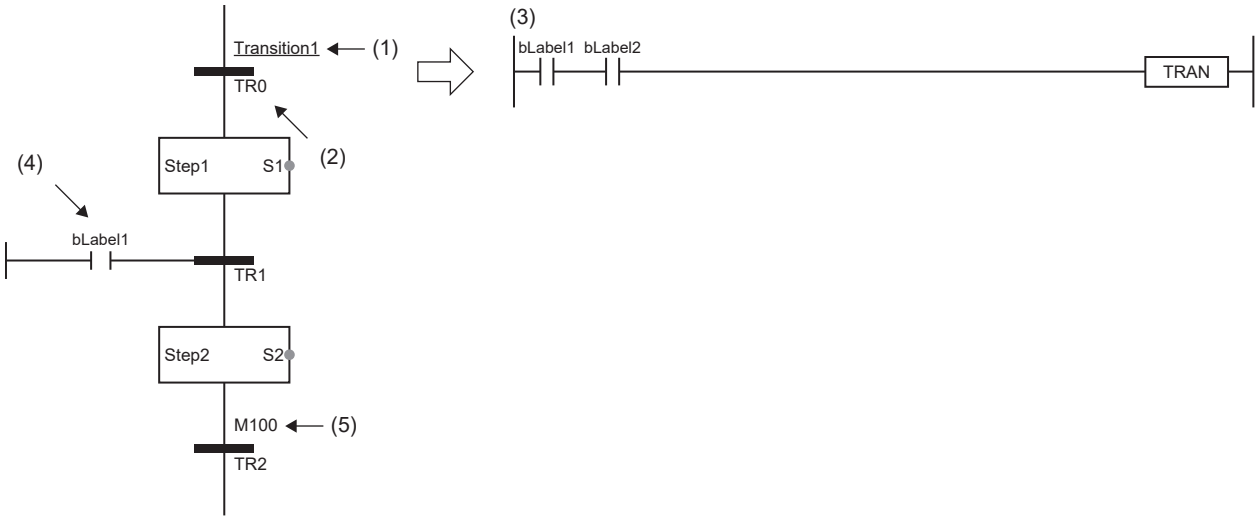
☞72页 线圈输出OFF的时机

• 在PLF指令的输入条件为ON的情况下，步变为非激活状态，动作输出结束时，指定软元件将保持ON状态不变。

• SFC控制指令的动作取决于执行前的执行条件。

转移条件

转移条件是构成块的基本单位，条件成立时将激活步并转移到下一个步。

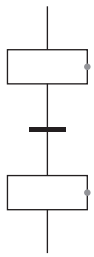
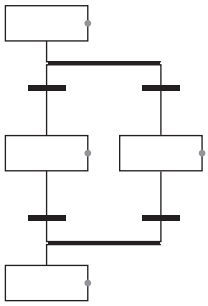
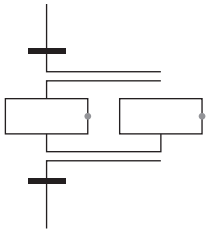
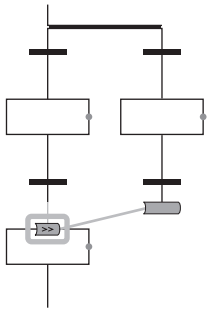


(1) 转移条件名
(2) 转移条件No.
(3) 转移条件的详细表示(☞86页 转移条件的详细表示)
(4) 转移条件的直接表示(☞87页 转移条件的直接表示)
(5) 转移条件的标签/软元件(☞87页 转移条件的标签/软元件)

转移条件的详细表示可以通过梯形图语言、ST语言、FBD/LD语言创建。

转移条件的类型

转移条件的类型如下所示。

项目		内容
串行转移		如果转移条件成立，则激活将从先行的步转移至后续的步。
选择转移(分支/合并)		分支：从1个步分支为多个转移条件，仅转移条件最先成立的列的步进行激活转移。 合并：如果转移条件在转移条件最先成立的列合并前成立，则激活将转移至下一个步。
并联转移(分支/合并)		分支：从1个步进行了分支的多个步全部同时进行激活转移。 合并：如果合并之前的步全部被激活，则在共通的转移条件成立时，激活将转移至下一个步。
跳转转移		转移条件成立时，激活转移至同一块内指定的步。

要点

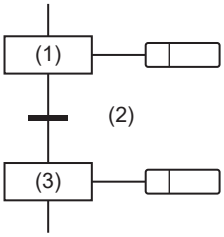
关于转移至已激活步的动作的有关内容，请参阅下述章节。

☞105页 步冗余启动时的注意点

串行转移

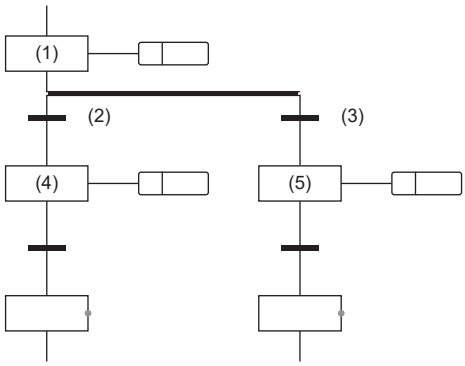
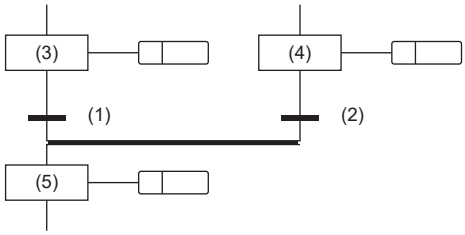
如果转移条件成立，则激活将从先行的步转移至后续的步。

在步(1)处于激活状态时，如果转移条件(2)成立，则将步(1)置于非激活、将步(3)置为激活。



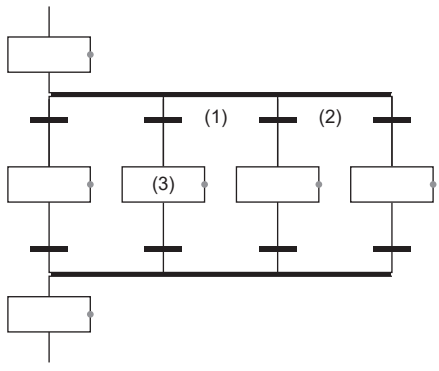
选择转移(分支/合并)

从1个步分支为多个转移条件，仅转移条件最先成立的列的步进行激活转移。如果转移条件在转移条件最先成立的列合并前成立，则激活将转移至下一个步。

项目	内容	
分支		步(1)处于激活时，激活转移条件先成立的步。(转移条件(3)比(2)先成立时，步(5)激活。) 步(1)将变为非激活。但是，在线圈保持步[SC]中，则根据属性保持线圈输出或动作输出。 • 多个转移条件同时成立的情况下，将优先执行左侧的转移条件。 • 选择后，依次执行所选择列的各步，直至各步合并为止。
合并		分支中激活的列的转移条件((1)或(2))成立时，将激活步(5)。 已激活的步(3)或(4)将变为非激活。但是，在线圈保持步[SC]中，则根据属性保持线圈输出或动作输出。

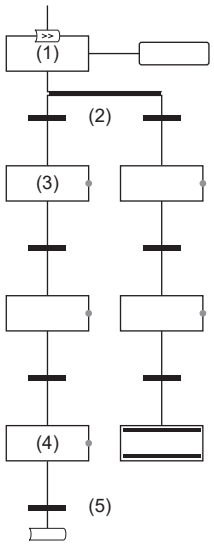
- 在选择转移中，最多可以分支为32个转移条件。
- 多个转移条件同时成立的情况下，将优先执行左侧的转移条件。

转移条件(1)及(2)同时成立的情况下，执行步(3)的动作输出。



- 也可创建选择转移的分支及合并的个数不相同的SFC图。但是，不可以创建选择分支与并联合并，并联分支与选择合并组合而成的SFC图。

• 在选择转移中，可以通过跳转转移及结束步来省略合并。



进行步(1)的动作输出时，如果转移条件(2)成立，则从步(3)开始按顺序执行步(4)。如果转移条件(5)成立，则跳转转移到步(1)。

要点

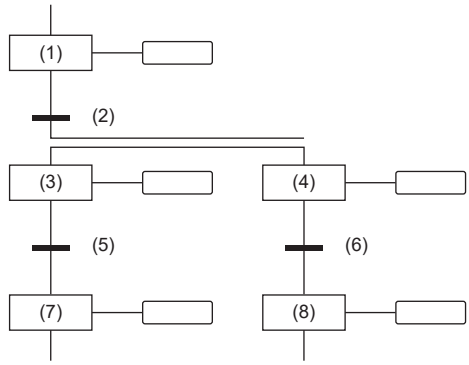
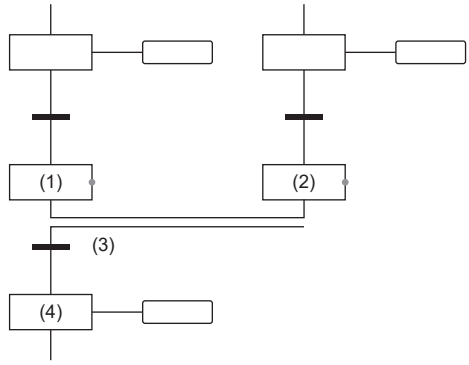
通过将选择分支的左端以外的步更改为结束步，并将位于选择分支的左端的结束步更改为跳转转移，可以创建上述程序。

关于对步进行更改的操作方法的有关内容，请参阅下述手册。

GX Works3操作手册

并联转移(分支/合并)

从1个步进行了分支的多个步全部同时进行激活转移。如果合并之前的步全部被激活，则在共通的转移条件成立时，激活将转移至下一个步。

项目	内容	
分支		步(1)处于激活时，如果转移条件(2)成立，步(3)与步(4)将同时置为激活。 步(1)将变为非激活。但是，在线圈保持步[SC]中，则根据属性保持线圈输出或动作输出。 转移条件(5)成立时转移到步(7)中，转移条件(6)成立时转移到步(8)中。
合并		将合并之前的步(1)和步(2)全部激活后，检查转移条件(3)，如果转移条件(3)成立，则将步(4)置为激活。 步(1)与步(2)将变为非激活。但是，在线圈保持步[SC]中，则根据属性保持线圈输出或动作输出。

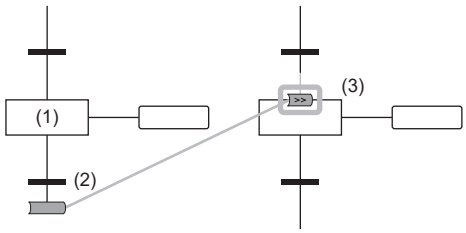
- 在并联转移中，最多可以转移到32个步中。
- 通过并联转移启动了其它块的情况下，将同时执行启动源的块及启动目标的块。
- 并联分支后，一定进行并联合并。

■注意事项

- 在并联合并中，合并的步中存在有保持步的情况下，将作为非激活步处理，不转移到下一步。
- 在并联合并中，不可以在合并之前对块启动步(有结束检查)[BC]进行创建。应使用块启动步(无结束检查)[BS]。

跳转转移

转移条件成立时，激活转移至同一块内指定的步。

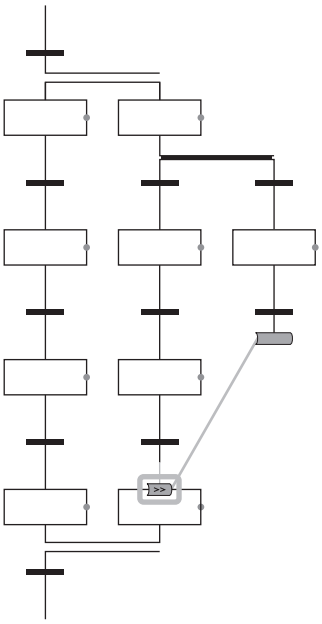


步(1)处于激活时，如果转移条件(2)成立，将步(3)置为激活。
步(1)将变为非激活。但是，在线圈保持步[SC]中，则根据属性保持线圈输出或动作输出。

- 跳转转移的使用个数无限制。
- 并联转移内的跳转转移，仅可在同一分支内进行。不可以创建并联分支内的不同分支的跳转转移，从并联分支脱离的跳转转移，从并联分支外至并联分支的跳转转移。

例

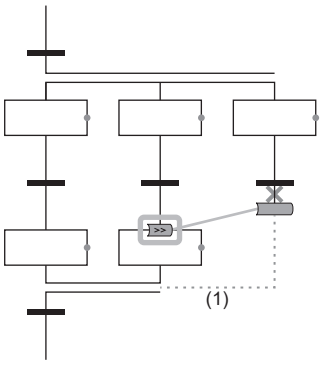
并联分支内可指定的跳转转移示例



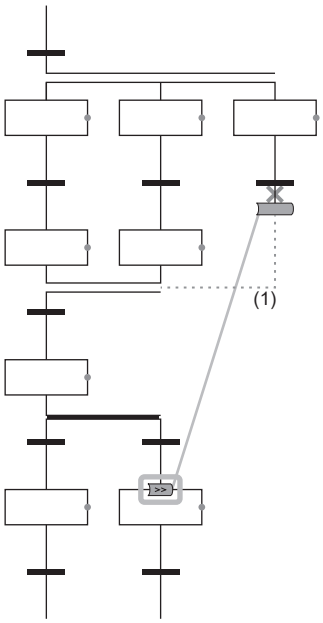
例

并联分支内不可指定的跳转转移示例

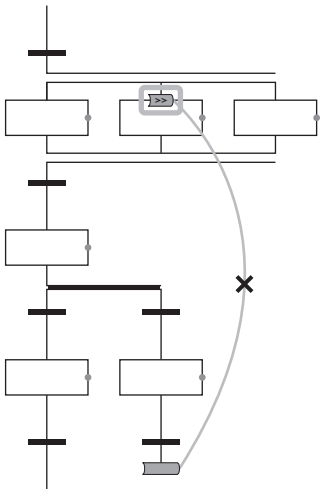
■并联分支内向其他分支跳转转移



■从并联分支脱离跳转转移



■从并联分支外向并联分支内跳转转移

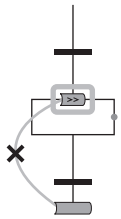


(1) 不能并联合并。

■注意事项

下述情况下，不可以作为跳转转移的目标进行指定。

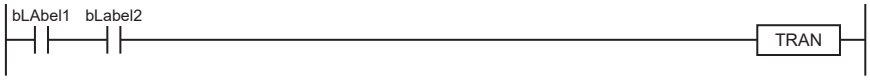
• 在指定为先行转移条件的前一个步的情况下



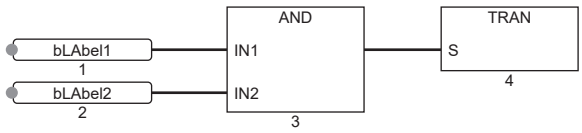
• 在指定为本步的情况下(“FX3兼容转移运行模式设置”为有效时可以指定)

转移条件的详细表示

转移条件的详细表示在Zoom编辑器内创建。由下述程序语言创建条件。

类型		内容
梯形图语言	详细表示	在单一的梯形图块中，对由触点的梯形图和TRAN指令(转移条件虚拟输出)组成的转移条件程序进行创建。如果执行TRAN指令，则转移条件成立。  ■限制事项 • 不可以使用内嵌ST。 • 线圈中仅可以输入TRAN指令。

ST语言		可以创建下述转移条件程序。
		■对TRAN函数(转移条件虚拟输出)的调用语句进行记述的方法
		TRAN(bLabel1 & bLabel2);
		//输入自变量的BOOL式为真(TRUE)的情况下，转移条件成立。
		■对保留字“TRAN”的BOOL式的代入语句进行记述的方法
		TRAN := bLabel1 & bLabel2;
		//右边的BOOL式为真(TRUE)的情况下，转移条件成立。
		■对转移条件名的BOOL式的代入语句进行记述的方法
		Transition1 := bLabel1 & bLabel2;
		//Transition1为显示在SFC编辑器上输入的转移条件名。右边的BOOL式为真(TRUE)的情况下，转移条件成立。

FBD/LD语言		在单一的梯形图块中，可以创建最后由TRAN指令(转移条件虚拟输出)组成的转移条件程序。  ■限制事项 • 不可以使用内嵌ST。 • TRAN指令只可以使用1个。 • 不可以创建代入至软元件/标签的程序。 • 不可以使用线圈部件、功能块部件、函数部件(一部分可以)、跳转部件/跳转标签部件、返回部件。 关于TRAN指令以外可使用的指令，请参阅下述内容。 ☞ 86页 可使用指令

要点

• 可以在多个转移条件中使用相同转移条件的详细表示。

• 创建的详细表示，可通过Zoom列表一览确认。(GX Works3操作手册)

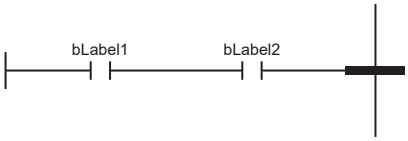
■可使用指令

转移条件的程序中可使用的指令如下所示。

分类	指令符号
触点指令	LD、LDI、AND、ANI、OR、ORI
	LDP、LDF、ANDP、ANDF、ORP、ORF
	LDPI、LDFI、ANDPI、ANDFI、ORPI、ORFI
合并指令	ANB、ORB
	INV
	MEP、MEF
比较运算指令	LD□、LD□_U、AND□、AND□_U、OR□、OR□_U
	LDD□、LDD□_U、ANDD□、ANDD□_U、ORD□、ORD□_U
实数指令	LDE□、ANDE□、ORE□
字符串处理指令	LD$□、AND$□、OR$□
转移条件虚拟输出	TRAN

转移条件的直接表示

可以将下一个步中转移激活的条件直接创建到SFC图上。和转移条件向对应的FBD/LD部件进行连接。



不可以使用线圈部件、功能块部件、函数部件、跳转部件/跳转标签部件、返回部件。

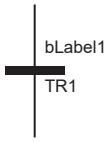
要点

选择转移条件后，对菜单的[编辑]⇨[更改]⇨[直接显示转移条件]进行选择，可以将FBD/LD部件连接到转移条件的左侧。(GX Works3操作手册)

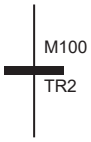
转移条件的标签/软元件

下一个步中转移激活的条件，可以指定位型标签及位软元件或BOOL值。

■位型标签的情况



■位软元件的情况



■BOOL值的情况

要点

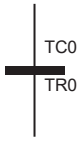
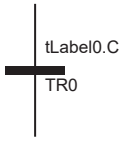
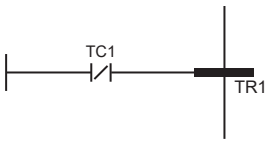
选择转移条件名后，对菜单的[编辑]⇨[更改]⇨[名称]进行选择，输入要指定的位型标签及位软元件或BOOL值。(GX Works3操作手册)

■注意事项

- 转移条件中使用了定时器及计数器的软元件(T、ST、C、LC)的情况下，TS、STS、CS、LCS将作为触点进行动作。在使用了定时器及计数器的软元件的线圈(TC、STC、CC、LCC)的情况下，也同样作为触点进行动作。
- 在转移条件中使用定时器或计数器的线圈的情况下，应使用定时器型或计数器型的标签。

例

使用定时器软元件或定时器型标签时

使用定时器软元件时	使用定时器型标签时
	
触点(TS0)为ON时，转移条件成立。 [直接显示时]	定时器型标签tLabel0的线圈为ON时，转移条件成立。 [直接显示时]
	
触点(TS1)为OFF时，转移条件成立。	定时器型标签tLabel1的线圈为OFF时，转移条件成立。

8

8.3 SFC控制指令

SFC控制指令是指，进行块、步的激活状态的检查以及强制启动、结束等的指令。如果使用SFC控制指令，可以在顺控程序以及SFC程序的动作输出内对SFC程序进行控制。

指令一览

SFC控制指令的一览如下所示。

指令名称	指令符号	处理内容
步激活检查	LD、LDI、AND、ANI、OR、ORI [S□]*1	检查指定步的激活/非激活。
	LD、LDI、AND、ANI、OR、ORI [BL□\S□]	
块激活检查	LD、LDI、AND、ANI、OR、ORI [BL□]	检查指定块的激活/非激活。
激活步批量读取	MOV(P) [KnS□]*1	将指定块的步激活状态作为位信息以BIN16位数据单位读取至指定软元件中。(Kn：K1～K4)
	MOV(P) [BL□\KnS□]	
	DMOV(P) [KnS□]*1	将指定块的步激活状态作为位信息以BIN32位数据单位读取至指定软元件中。(Kn：K1～K8)
	DMOV(P) [BL□\KnS□]	
	BMOV(P) [KnS□]*1	将指定块的步激活状态从指定步以指定字部分批量进行读取。(Kn：K1～K4)
	BMOV(P) [BL□\KnS□]	
块启动	SET [BL□]	单独激活指定块，从初始步开始执行。
块结束	RST [BL□]	将指定块单独置为非激活。
块停止	PAUSE [BL□]	将指定块置为暂停状态。
块重启	RSTART [BL□]	解除指定块的暂停，从停止步开始重新执行。
步启动	SET [S□]*1	激活指定步。
	SET [BL□\S□]	
步结束	RST [S□]*1	将指定步置为非激活。
	RST [BL□\S□]	
步启动/结束指令	OUT [S□]*1	将指定步置为激活/非激活。
	OUT [BL□\S□]	
	ZRST(P) [S□]*1	将指定步批量置为非激活。
	ZRST(P)[BL□\S□]	

*1 在顺控程序内使用时，块0将变为对象。在SFC程序内使用时，本块将变为对象。

SFC控制指令的详细内容，请参阅下述手册。

📖MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)

■注意事项

- 在中断程序内请勿使用SFC控制指令。
- 请仅在SM321(启动/停止SFC程序)为ON时执行SFC控制指令。
- 使用SFC控制指令时，请在“SFC程序设置”中设置为“使用”。
- 使用SFC程序时，请勿使用SFC控制指令以外的指令指定步进继电器。如指定，程序可能无法按预期执行动作。
- “SFC程序设置”设置为“不使用”时，使用未指定块的步进继电器指定了步的指令(LD [S□]或MOV(P) [KnS□]等)将作为普通指令运行。
- 在CPU模块不支持SFC程序的情况下，从工程工具对带块指定的步进继电器(BL□\S□)执行读取/写入当前值时，将对未指定块的步进继电器(S)读取/写入当前值。(☞116页 功能的添加和更改)

  例：对BL5\S12读取当前值时

  - CPU模块支持SFC程序：读取BL5\S12当前值。
  - CPU模块不支持SFC程序：读取S12当前值。

变址修饰

SFC控制指令可指定变址修饰过的步进继电器(S)和SFC块软元件(BL)。但是，单独控制步的指令不能指定变址修饰过的软元件。

软元件	变址修饰对象位置
S□Z□	步进继电器
BL□\S□Z□	带块指定步进继电器的步部分
BL□Z□\S□	带块指定步进继电器的块部分
BL□Z□\S□Z□	带块指定步进继电器的块部分及步部分
BL□Z□	SFC块软元件

在下述范围内指定步进继电器及SFC块软元件，也包括进行变址修饰的情况。

软元件		范围
S□		0～4095
BL□\S□	BL□	0～31
	S□	0～511
BL□		0～31

要点

关于变址修饰的详细内容，请参阅下述手册。

MELSEC iQ-F FX5用户手册(应用篇)

8.4 SFC设置

在CPU参数及SFC块设置中，设置SFC程序的启动条件等。

CPU参数

SFC设置一览如下所示。

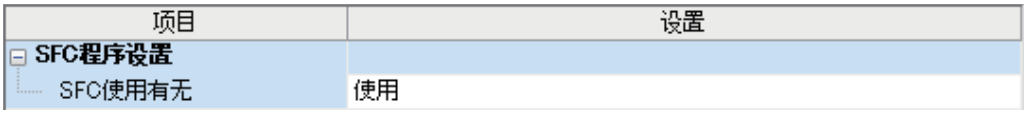
类型	项目	内容
SFC设置	SFC程序设置	设置是否使用SFC程序。
	SFC程序启动模式设置	在SFC程序启动时，对是在初始状态下进行启动(初始启动)，还是在保持之前的执行状态不变的状态下进行启动(继续启动)进行设置。
	起动条件设置	在SFC程序启动时，设置为在自动启动块0后激活，或是设置为保持非激活状态不变直至有启动请求为止。
	块停止时的输出模式设置	块停止时，设置将线圈输出置为OFF或保持线圈输出。
	FX3兼容转移运行模式设置	设置SFC程序是否与FX3兼容运行。

SFC程序设置

设置是否使用SFC程序。不使用SFC程序时，不能进行其他SFC设置。

[CPU参数]⇨[SFC设置]⇨[SFC程序设置]

画面显示



显示内容

设置	内容
不使用(默认)	不使用SFC程序。
使用*1	使用SFC程序。

*1 工程的程序语言为SFC时，默认值为“使用”。

根据SFC程序设置的变化，动作项目和各SFC程序设置的设置项目的动作如下所示。

■根据设置内容而变化的动作项目

动作项目	内容
SFC程序的执行	仅在“SFC程序设置”设置为“使用”时能执行SFC程序。
指令用步进继电器(S)的指定	“SFC程序设置”设置为“使用”时，SFC控制指令以外的指令不能指定步进继电器(S)。

■各设置项目的动作

SFC程序设置	SFC程序的执行	指令用步进继电器(S)的指定
不使用	不能执行*1	无限制
使用	能执行	仅能对SFC控制指令指定*2

*1 创建SFC程序时，工程工具会发生转换出错。使用SD存储卡进行引导运行等情况，对“SFC程序设置”设置为“不使用”的参数和SFC程序分别写入时，会发生CPU模块的自诊断出错。

*2 对SFC控制指令以外的指令指定步进继电器(S)时，会发生CPU模块的自诊断出错。

SFC程序启动模式设置

在SFC程序启动时，对是在初始状态下进行启动(初始启动)，还是在保持之前的执行状态不变的状态下进行启动(继续启动)进行设置。

[CPU参数]⇨[SFC设置]⇨[SFC程序启动模式设置]

画面显示

项目	设置
SFC程序起动模式设置	
SFC程序起动模式	初始启动

显示内容

设置	内容
初始启动(默认)	对上次停止时的激活状态进行清除并启动。 启动后的动作按照SFC设置的“启动条件设置”进行。(☞92页 起动条件设置)
继续启动*1	保持上次停止时的激活状态进行启动。 设置为继续启动时，请在CPU模块中安装电池。(☞92页 注意事项)

*1 仅支持FX5U/FX5UC CPU模块。

通过组合SFC程序启动模式设置和SM322(SFC程序的启动状态)，可决定进行初始启动或继续启动。

动作		SFC程序启动模式设置：初始启动		SFC程序启动模式设置：继续启动	
		SM322：OFF (初始状态)*1	SM322：ON (设置更改时)	SM322：ON (初始状态)*1	SM322：OFF (设置更改时)
(1)	将SM321 OFF→ON	初始启动	初始启动*3	继续启动	初始启动
(2)	将CPU模块的电源OFF→ON		初始启动(M322变为OFF，成为初始状态。)	继续启动/初始启动*4*6	继续启动/初始启动*4*5
(3)	将SM321 ON→OFF或RUN→STOP后将CPU模块电源OFF→ON			继续启动*6	继续启动*5
(4)	复位→RUN			继续启动/初始启动*4*6	继续启动/初始启动*4*5
(5)	将SM321 ON→OFF或RUN→STOP后进行复位→RUN			继续启动*6	继续启动*5
(6)	STOP→RUN	继续启动			
(7)	STOP→程序写入(SFC程序以外)→RUN	初始启动		继续启动	初始启动
(8)	STOP→程序写入(SFC程序)→RUN	初始启动*2			

*1 对于SM322，根据SFC程序启动模式的设置在STOP→RUN时决定初始状态。
*2 将SFC程序启动模式设置设置为“继续启动”，在程序的写入前后无更改的情况下将继续启动。
*3 参数设置为初始启动时，M322的ON状态无效。
*4 某些情况下可能继续启动变为禁止，进行初始启动。
*5 M322变为ON，成为初始状态。
*6 仅FX5U/FX5UC CPU模块支持将CPU模块的电源置为OFF→ON及复位后的继续启动。

■注意事项

- 设置为继续启动时，请在CPU模块中安装电池。发生未安装电池或电池电压过低等电池异常时，CPU模块的电源在OFF→ON后，在SFC程序初次启动时可能会变为初始启动。此外，通过进行选件电池的设定，可对电池异样进行检出。详情请参阅所使用CPU模块的用户手册。
- 继续启动时，SFC程序的停止位置将保持，但是动作输出中使用的标签及软元件的状态不保持。因此，在进行继续启动时，应将需要保持的标签及软原件进行锁存设置。(📖MELSEC iQ-F FX5用户手册(应用篇))
- 线圈保持步[SC]的线圈输出为OFF的条件(表中(1)、(3)、(5))以外的继续启动时，将重启保持中的线圈保持步[SC]，但是输出不变为ON。保持输出时，应将标签及软元件置进行锁存设置。(📖MELSEC iQ-F FX5用户手册(应用篇))
- CPU模块的电源OFF时或复位时，智能功能模块将被初始化。在继续启动的情况下，智能功能模块的初始程序，建议创建在常时激活状态的块或顺控程序上。
- CPU模块的电源OFF时或复位时，标签及软元件会被清除。
- CPU模块的电源OFF后或复位后的继续启动，根据情况有可能无法继续启动。在设置了继续启动而进行了初始启动的情况下，在事件履历中禁止继续启动的事件将被存储。如希望确保进行继续启动，应将SM321置为ON→OFF或RUN→STOP后再将CPU模块电源置为OFF或进行复位。

起动条件设置

在SFC程序启动时，设置为在自动启动块0后激活，或是设置为保持非激活状态不变直至有启动请求为止。

[CPU参数]⇨[SFC设置]⇨[起动条件设置]

画面显示

项目	设置
⊟ 起动条件设置	
起动条件	自动起动块0

显示内容

设置	内容	
	SFC程序启动时	块0结束时
自动起动块0 (默认)	块0将被自动启动，并从初始步开始执行。	块0将被自动再启动，并再次从初始步开始执行。
不自动起动块0	块0也与其他块一样根据SFC控制指令(块启动)或块启动步在有启动请求时变为激活状态。	块0不自动进行再启动，而是保持非激活状态，直至再次有启动请求为止。

启动条件设置，根据产品类型等控制启动块的情况下使用。

“自动启动块0”在按以下方式使用块0的情况下有效。

- 管理块
- 前处理块
- 常时监视块

■注意事项

- 设置为“不自动启动块0”的情况下，执行SFC程序时通过顺控程序执行SFC控制指令(块启动)。
- 设置为“自动启动块0”时，必须创建块0。

块停止时的输出模式设置

块停止时，设置将线圈输出置为OFF或保持线圈输出。

[CPU参数]⇨[SFC设置]⇨[块停止时的输出模式设置]

画面显示

显示内容

设置	内容
OFF (默认)	将线圈输出置为OFF。
保持ON	以停止前的状态保持线圈输出。

- 设置的内容在电源ON时及复位时或由STOP→RUN时，反映为SM325(块停止时的输出模式)的初始值，SFC程序动作时遵循SM325的设置。将忽略CPU参数的设置。

■块停止重启时的动作

块停止/重启时的动作根据SM325(块停止时的输出模式设置)与SFC用信息软元件的块停止模式位的设置、步的保持/非保持的组合而定。

块停止/重启时的动作一览如下所示。

块停止时的输出模式的设置	动作			
	保持中以外的激活中的步*1	保持中的步		
		线圈保持步[SC]	动作保持步(无转移检查)[SE]	动作保持步(有转移检查)[ST]
SM325＝OFF (线圈输出OFF)	接收停止请求后，将动作输出的线圈输出置为OFF后停止。状态保持为激活。	接收停止请求后，将动作输出的线圈输出置为OFF后变为非激活。	接收停止请求后，将动作输出的线圈输出置为OFF后停止。状态保持为激活。	
SM325＝ON (保持线圈输出)	接收停止请求后，在保持动作输出的线圈输出状态下停止。状态保持为激活。	接收停止请求后，在保持动作输出的线圈输出状态下停止。状态保持为激活。		
重启时	恢复为普通的动作。	线圈输出OFF时：为非激活，因此无法重启。 保持线圈输出时：以保持中的状态重启。	以保持状态重启动作输出的执行。	以保持状态重启动作输出，并检查转移条件。

*1　也包含转移条件不成立的SC、SE、ST。

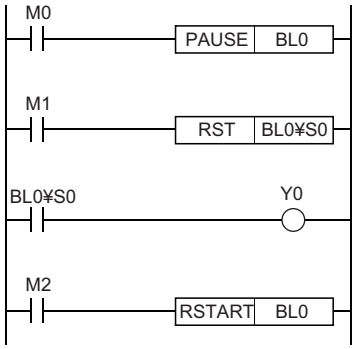
■注意事项

- 通过LD指令(块激活检查)等指定的块为停止中的块时，将变为ON。此外，通过LD指令(步激活检查)等指定的步为停止中的步时，也将变为ON。
- 对停止状态的非激活块执行块启动指令，块以停止状态变为激活状态，但初始步保持非激活状态。重启块时，初始步变为激活状态，执行动作输出。
- SM325=ON(块停止时的线圈输出保持)时，可以以保持线圈输出的状态停止。停止中将SM325置为ON→OFF，线圈输出的状态也不会变化，如果发生块的重启请求，则以保持状态重启。
- SM325=ON时块停止的情况下，保持状态的线圈保持步[SC]在重启后也维持保持状态，但步的动作不会重启。将线圈保持步[SC]置为非激活时，应执行RST指令(步结束)。
- 动作输出内对该块有停止请求，也会执行当前执行中的步直到最后，并执行停止请求。因此，执行中在步内发出停止请求也不会停止。
- 如果在块停止中执行RST指令，指定的步进继电器将OFF。但工程工具的监视画面为激活状态，在重启块时变为非激活。SM325=ON时停止中即使执行也为相同情况，但线圈输出不OFF。

• SET指令(步起动)即使在块停止中也会被立即执行，指定的步进继电器ON，工程工具的监视画面上也为激活状态。但不执行动作输出，直到块重启为止。

例

使用RST指令时的块停止重启时



(1) 将M0置为ON，则块0停止。
(2) 将M1置为ON，则对步No.0发出结束请求，步进继电器的BL0\S0将OFF，但工程工具的监视器上步No.0保持激活中。
(3) BL0\S0变为OFF，因此Y0也将OFF。
(4) 如果在M0、M1为OFF的状态下将M2置为ON，则重启块0，步No.0结束。

FX3兼容转移运行模式设置

本设置可使SFC程序与FX3以相同动作运行。将用户装置中使用的FX3配置置换为FX5的配置时，需要确保与FX3兼容性的情况下使用。

[CPU参数]⇨[SFC设置]⇨[FX3兼容转移运行模式设置]

画面显示

项目	设置
FX3兼容转移运行模式设置	
FX3兼容转移运行模式	禁用

显示内容

设置	内容
禁用(默认)	变为FX5运行，可指定有/无连续转移。
启用	变为FX3运行，以有连续转移(FX3兼容运行)方式运行。

关于连续转移，请参阅☞ 102页 有/无连续转移的运行。

要点

关于SFC程序的FX3→FX5置换，请参阅下述内容。
MELSEC FX3G/FX3U/FX3UC系列替换为MELSEC iQ-F系列的相关说明

SFC块设置

块冗余起动时的运行设置

对已激活的块通过块启动步(有结束检查)[BC]或块启动步(无结束检查)[BS]发出了启动请求时，在停止CPU模块运算的情况下设置。在设置范围中设置停止的块范围。

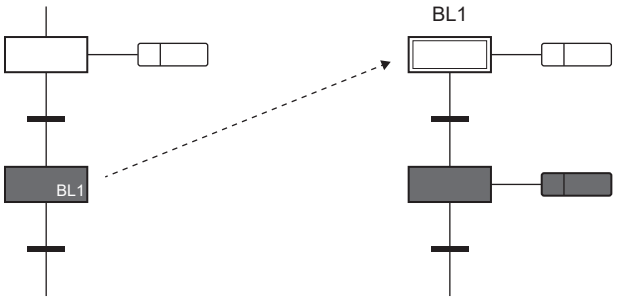
[导航窗口]⇨[程序]⇨要设置的SFC程序文件的属性

画面显示

(1) 设置停止的块范围。

显示内容

设置	内容	
无设置(默认)	待机	继续运行CPU模块的运算，在转移条件成立的状态保持待机，直至启动目标的块变为非激活状态。 启动目标的块变为非激活状态时，将块再次置为激活状态。
有停止的块范围的设置	停止	变为出错状态。



■注意事项

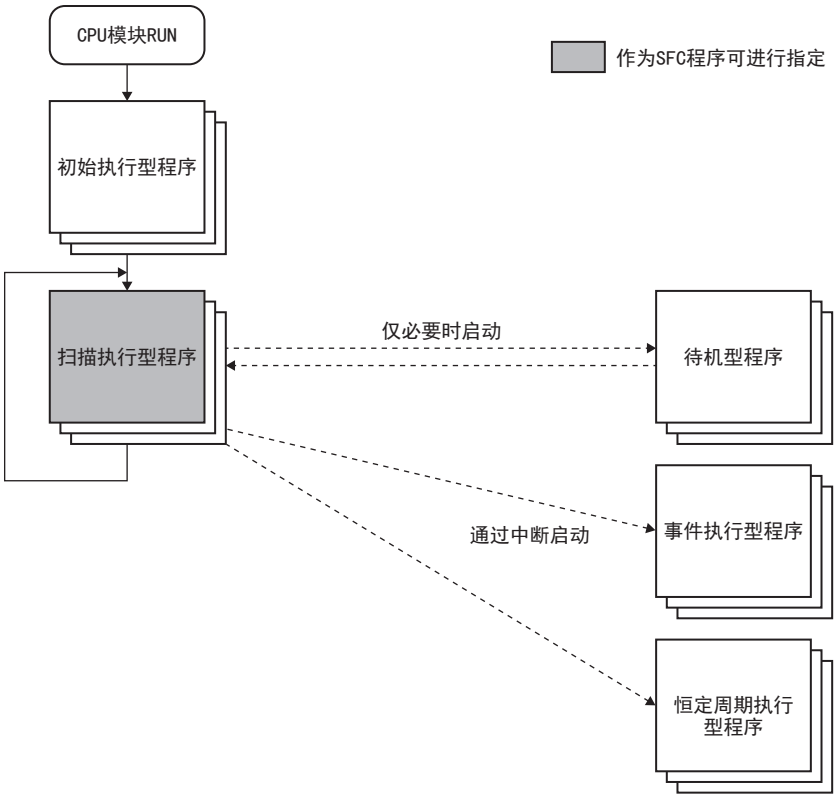
- 对已处于激活中的块，执行了SFC控制指令(块启动)时，将忽略启动请求，继续运行SFC程序的处理。
- 试图转移至激活中的块启动步的情况下，块启动步的启动将被忽略。不会再次从初始步开始执行。

8.5 SFC程序的执行顺序

整个程序的处理

可指定的执行类型

在SFC程序中，程序的执行类型可否指定如下所示。



执行类型	指定可否	备注
初始执行型程序	×	—
扫描执行型程序	○	SFC程序时仅1个可以执行
待机型程序	×	—
事件执行型程序	×	—
恒定周期执行型程序	×	—
无指定	○	—

■注意事项

在存在SFC程序的工程中，不能使用步梯形图(STL/RETSTL指令)。

SFC程序的处理顺序

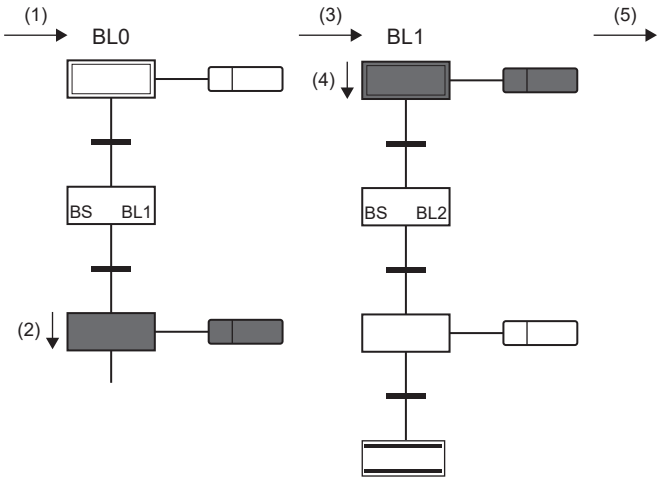
各块的执行顺序

在SFC程序的启动中，从已激活的块的初始步开始按顺序执行各步的动作输出。

在多个块的SFC程序中，将按照块0→块1→块2的顺序从小编号的块开始向大编号的块按顺序进行激活检查。

激活中的块将执行块内的激活步的动作输出。

非激活块将检查启动请求的有无。如果有启动请求，则将块激活，执行块内的激活步。



按下述顺序执行。

(1) 开始块0(BL0)的处理。

(2) 执行块0(BL0)的步。

(3) 执行块1(BL1)的处理。

(4) 执行块1(BL1)的初始步。

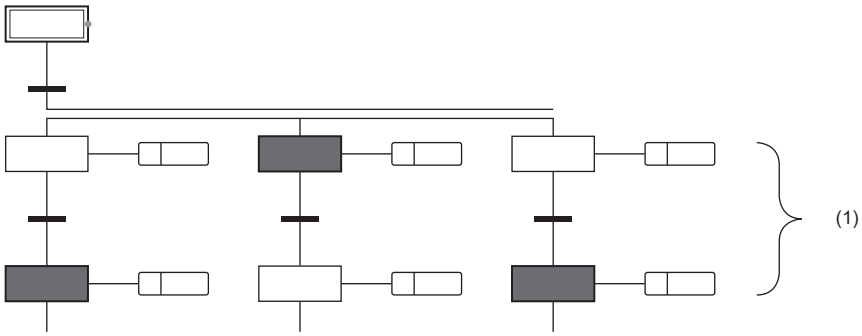
(5) 执行下一个块的处理。

在SFC设置的“启动条件设置”中指定为“自动启动块0”的情况下，可以自动启动块0。在此设置的情况下，即使到达结束步变为非激活状态，块0也将在下一个扫描中再次被启动。(☞92页 起动条件设置)

各步的执行顺序

通过SFC程序，在1个扫描内处理所有激活步的动作输出。

(1) 块内的激活步在1个扫描内被执行。



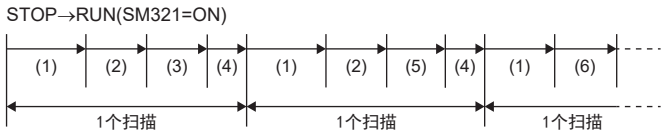
各步的动作输出结束时，会检查下一个步的转移条件的成立状态。

- 转移条件不成立时：执行下一个扫描的同时，再次执行同一步的动作输出。
- 转移条件成立时：将通过执行的动作输出中OUT指令进行的输出全部置为OFF。在执行下一个扫描时，执行下一个步的动作输出。上回执行的步将变为非激活状态，动作输出将变为非执行状态。

即使转移条件成立，将步的属性设为线圈保持步[SC]时，也不变为非激活状态而是按照属性处理。(☞ 72页 线圈保持步[SC])

例

转移动作例(无连续转移)



(1) 顺控程序的执行
(2) 动作输出的执行
(3) 至下一个步的转移条件检查(条件不成立)
(4) END处理
(5) 至下一个步的转移条件检查(条件成立)
(6) 执行下一个动作输出

■动作输出的执行

根据执行步状态，针对在动作输出内记述指令的输入条件，对动作输出执行以下动作。

动作输出的执行	内容
非执行	输入条件不反映至输出。
ON执行	根据输入条件执行动作。
OFF执行	无论实际输入条件如何，都将作为输入条件OFF执行动作。

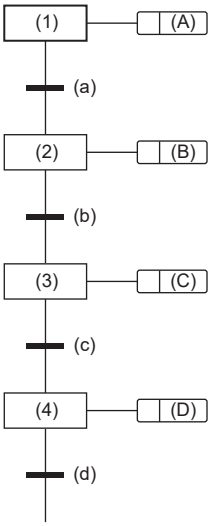
步进行激活时，每个扫描中都将执行动作输出ON。步变为非激活时，执行动作输出OFF后，变为非执行，直至下一个步被激活为止。在以下情况，执行动作输出OFF。OFF的执行仅在激活步中。

- 在普通步后转移条件成立时
- 执行了结束步时(SM327为ON时的保持步及SM328为ON时剩余的非保持的激活中的步除外)
- 通过SFC控制指令(块结束)对块进行了强制结束时
- 通过SFC控制指令(步结束)对步进行了强制结束时
- 将SM321(启动/停止SFC程序)置为OFF时
- 设置了用于复位执行步的复位步[R]被激活时
- 设置了用于复位保持中的步的复位步[R]被激活，且复位步[R]指定为S999时
- 块停止时的输出模式为OFF，通过块停止指令使块停止时

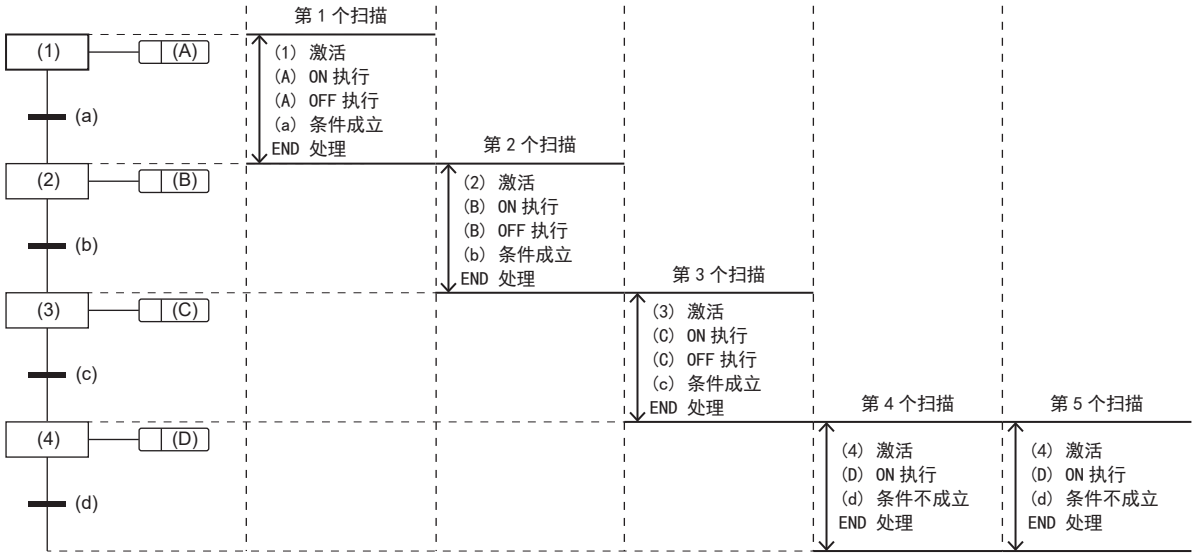
■转移动作

有/无SFC程序连续转移的转移动作情况如下所示。(☞ 102页 有/无连续转移的运行)

激活步(1)～(4)，执行动作输出(A)～(D)。
步(1)～(3)的转移条件(a)～(c)成立，步(4)的转移条件(d)不成立。



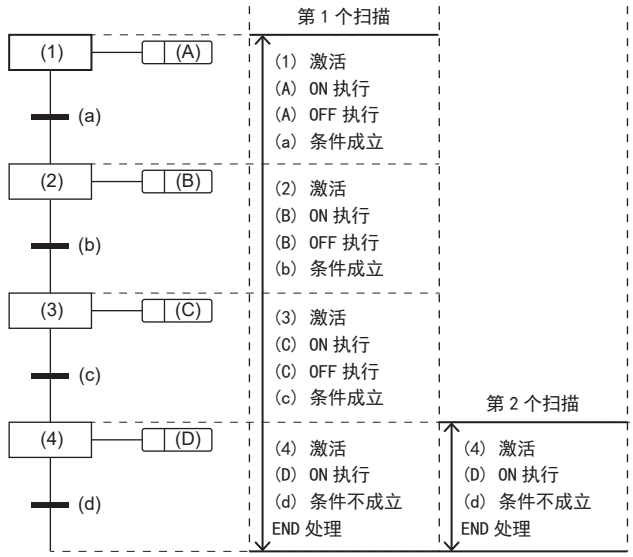
• 无连续转移时



扫描	内容
第1个扫描	激活步(1)，动作输出(A)执行ON后执行OFF。*1
第2个扫描	激活步(2)，动作输出(B)执行ON后执行OFF。*1
第3个扫描	激活步(3)，动作输出(C)执行ON后执行OFF。*1
第4个扫描	激活步(4)，动作输出(D)执行ON。
第5个扫描及其以后	在转移条件(d)成立之前激活步(4)，动作输出(D)执行ON。

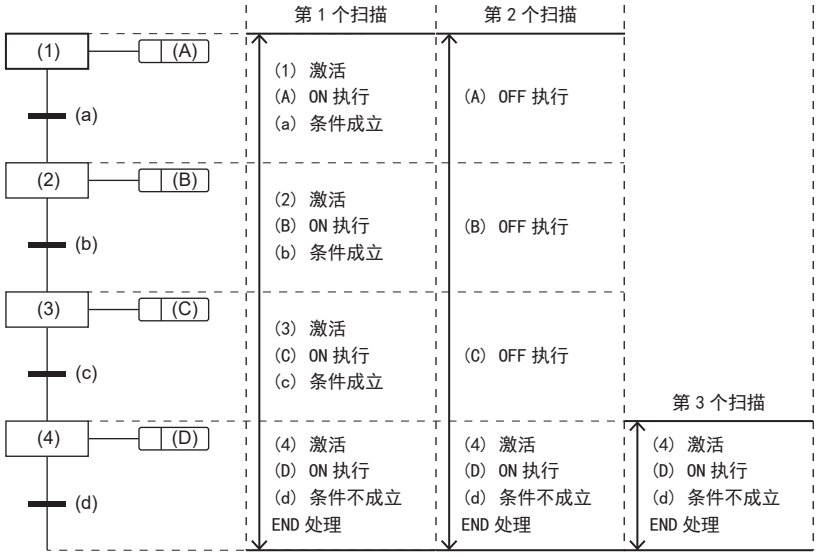
*1 线圈保持步时，不执行OFF。

• 有连续转移时



扫描	内容
第1个扫描	步(1)～(4)连续激活。 各步激活时，对动作输出(A)～(C)执行ON后执行OFF，(D)执行ON。但是，线圈保持步时，不执行OFF。
第2个扫描及其以后	在转移条件(d)成立之前激活步(4)，动作输出(D)执行ON。

• 有连续转移(FX3兼容运行)时



扫描	内容
第1个扫描	步(1)～(4)连续激活。 各步激活时，对动作输出(A)～(D)执行ON。
第2个扫描	对动作输出(A)～(C)执行OFF，对动作输出(D)执行ON。但是，线圈保持步时，不执行OFF。
第3个扫描及其以后	在转移条件(d)成立之前激活步(4)，动作输出(D)执行ON。

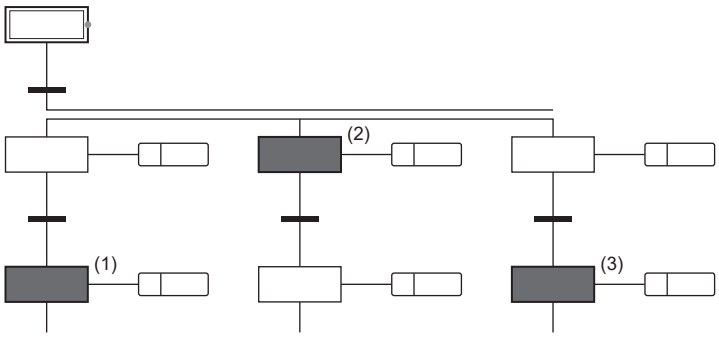
■注意事项

• 在初次执行时步转移条件成立的情况，通过1个扫描启动/结束步。通过1个扫描结束步时，根据各转移条件进行以下动作。

无连续转移	有连续转移	有连续转移(FX3兼容运行)
未反映线圈输出等I/O刷新。为反映I/O刷新，请创建程序，将1个扫描执行多次。但是，其他程序也可在1个扫描前(从前一个扫描到转移之后)检测步进继电器ON状态。	未反映线圈输出等I/O刷新，其他程序不能检测线圈输出的ON状态。为反映I/O刷新，请创建程序，将1个扫描执行多次。	因步的启动和结束由其他扫描执行，所以动作输出内的线圈输出可通过其他程序检测。但是，步进继电器将在转移条件成立后的扫描中OFF。

• 块内的激活步的动作输出同时(同一扫描内)被执行。因此，请勿创建取决于动作输出的执行顺序的SFC程序。

(1)、(2)、(3)的动作输出的执行顺序将变为不定。



• 在有连续转移的情况下，通过跳转转移或选择合并从多个步向1个步转移时，通过1次扫描，1个步的动作输出可执行2次。

• 在有连续转移的情况下，如果使用跳转转移执行重复程序，则会发生出错。

• 有连续转移(FX3兼容运行)，跳转转移的转移目标步所处位置与转移源相比高于SFC图上侧时，即使转移条件成立，转移目标步也会在下一个扫描执行。

有/无连续转移的运行

SFC程序的转移条件中包括“有连续转移”、“有转移条件(FX3兼容运行)”、“无连续转移”动作。
有/无连续转移的设置，由“FX3兼容转移运行模式设置”和SM323(有无全部块连续转移)决定。

FX3兼容转移运行模式设置	SM323	内容	
禁用	OFF	无连续转移	转移条件成立时，转移目标步的动作输出在下一个扫描中执行。
	ON	有连续转移	转移条件成立时，将转移目标步的动作输出在同一扫描内执行。 步的转移条件连续成立的情况下，在转移条件不成立之前或到达结束步之前，在同一扫描内执行。 此外，转移源步的动作输出在转移条件成立后的下一个扫描中执行OFF。
启用	ON/OFF	有连续转移(FX3兼容运行)	转移条件成立时，将转移目标步的动作输出在同一扫描内执行。 步的转移条件连续成立的情况下，在转移条件不成立之前或到达结束步之前，在同一扫描内执行。 此外，与普通的连续转移不同，转移源步的动作输出在转移条件成立后的下一个扫描中执行OFF。

要点

- 通过设置为“有连续转移”或“有连续转移(FX3兼容运行)”，可缩短间歇时间。因此，可消除从转移条件成立到转移目标步动作输出执行之间的等待时间。但是，如果设置为“有连续转移”或“有连续转移(FX3兼容运行)”，其他块或顺控程序的动作可能变慢。
- SM324(连续转移阻止标志)仅在“有连续转移”时变为OFF。(通常，在SFC程序执行时系统自动为ON，因此常时为ON。同时，“有连续转移(FX3兼容运行)”时也常时为ON。)因此，通过在转移条件中使用SM324，可禁用连续转移。
- “FX3兼容转移运行模式设置”为有效时，无论SM323的ON/OFF状态如何，均为“有连续转移(FX3兼容运行)”。

8.6 SFC程序的执行

SFC程序的启动及停止

SFC程序的启动及停止方法如下所示。

- 通过CPU参数进行的自动启动
- 通过特殊继电器(SM321)进行的启动及停止

通过CPU参数进行的自动启动

将CPU参数的“启动条件设置”设置为“自动启动块0”时，CPU模块的电源ON时、复位时或STOP→RUN时启动SFC程序，并启动块0。(☞ 92页 起动条件设置)

通过特殊继电器(SM321)进行的启动及停止

在执行SFC程序时，SM321(启动/停止SFC程序)通过系统自动变为ON。

- 通过将SM321置为OFF，可以结束SFC程序全部的处理。
- 通过将SM321置为ON，可以再次执行已结束的SFC程序。

要点

通过将CPU参数的“SFC程序启动模式设置”设置为“继续启动”，可以继续启动SFC程序。(☞ 91页 SFC程序启动模式设置)

块的启动及结束

块的启动方法

块的启动方法如下所示。

项目	启动方法	备注	参照
通过CPU参数进行的自动启动(仅块0)	通过在CPU参数的SFC设置中将“起动条件设置”设置为“自动起动块0”，在SFC程序的启动时块0将被自动启动，并从初始步开始执行处理。	在将块0作为管理块及前处理块、常时监视块等使用时进行设置。	☞ 92页 起动条件设置
通过块启动步进行的启动	在SFC程序的各块中，通过块启动步[BC或BS]启动其它的块。	在控制的顺序明确时有效。	☞ 74页 块启动步(有结束检查)[BC] ☞ 75页 块启动步(无结束检查)[BS]
通过SFC控制指令进行的启动	从SFC程序的动作输出或其他顺控程序，通过SFC控制指令启动指定块。 • 从指定块的初始步开始执行的情况下，使用SET[BL□]指令(块启动)。 • 从指定块的指定步开始执行的情况下，使用SET[S□/BL□\S□]/OUT [S□/BL□\S□]指令(步启动)。	检测出异常时，出错恢复处理块开始启动、在执行中断处理时有效。	☞ 88页 SFC控制指令
通过工程工具进行的启动	通过将SFC块软元件置为ON启动指定块。	在调试及试运行时有效。	GX Works3操作手册

块的结束方法

块的结束方法如下所示。

项目	结束方法	备注	参照
通过结束步进行的结束	如果执行块内的结束步，将结束块的处理，并变为非激活状态。	通过停止自动运行中的循环，可有效停止运行。	☞ 75页 结束步
通过SFC控制指令进行的结束	从SFC程序的动作输出或其他顺控程序，通过RST [BL□]指令(块结束)结束指定的块，并置为非激活状态。通过(RST [BL□\S□]/OUT [BL□\S□]/ZRST(P)[S□/BL□\S□]指令(步结束)，将指定块内的激活步全部置为非激活状态的同时也结束块。)	与动作状态无关，通过紧急停止等中止处理时有效。	☞ 88页 SFC控制指令
通过工程工具进行的结束	通过将SFC块软元件置为OFF，结束指定块。	在调试及试运行时有效。	GX Works3操作手册

步的启动及结束(激活及非激活)

步的启动(激活)方法

对步进行启动(激活)的方法如下所示。

项目	启动方法	备注	参照
通过转移条件成立进行的启动	之前的转移条件成立时，下一个步将自动启动。	—	☞80页 转移条件
通过SFC控制指令进行的启动	从SFC程序的动作输出或其他顺控程序，通过SET[S□/BL□\S□]/OUT[S□/BL□\S□]指令(步启动)对指定的步进行启动。	—	☞88页 SFC控制指令
通过工程工具进行的启动	• 通过将步进继电器置为ON启动指定步。 • 通过菜单的[调试]⇨[SFC步控制]将选择的步置为激活状态。	在调试及试运行时有效。	📖GX Works3操作手册

步的结束(非激活)方法

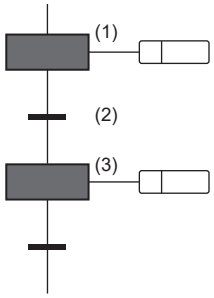
对步进行结束(非激活)的方法如下所示。

项目	结束方法	备注	参照
通过转移条件成立进行的结束	步的下一个转移条件成立时，将自动结束。	—	☞80页 转移条件
通过复位步[R]进行的结束	复位步[R]变为激活时，在属性指定目标中指定的步将结束。	在SFC程序的选择分支中转移至出错处理步的情况等，结束线圈保持步[SC]时有效。	☞74页 复位步[R]
通过SFC控制指令进行的结束	从SFC程序的动作输出或其他顺控程序，通过RST[S□/BL□\S□]/OUT[S□/BL□\S□]/ZRST(P)[S□/BL□\S□]指令(步结束)对指定的步进行结束。	通过SFC控制指令(步结束)指定块的全部步变为非激活状态时，块也将结束。	☞88页 SFC控制指令
通过工程工具进行的结束	• 通过将步进继电器置为OFF结束指定步。 • 通过菜单的[调试]⇨[SFC步控制]将选择的步置为非激活状态。	在调试及试运行时有效。	📖GX Works3操作手册

步冗余启动时的注意点

对步冗余启动时的动作如下所示。

串行转移的情况



转移条件(2)成立时，步(1)将变为非激活状态。
冗余启动的转移目标步(3)激活。

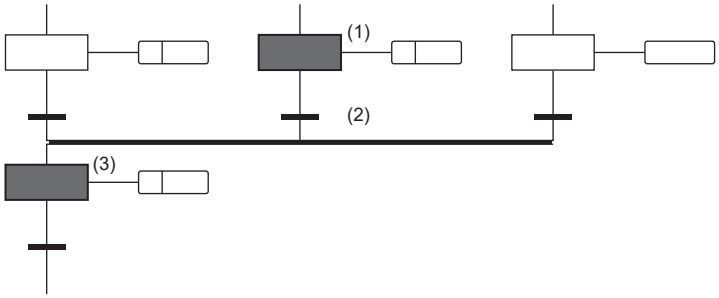
选择转移的情况

■选择分支

从左侧开始优先进行转移条件检查，如果转移条件成立的分支的转移目标为激活步，则进行与串行转移相同动作。与普通的选择转移相同，对冗余启动成立的分支以外的列，不进行转移条件检查。

■选择合并

进行与串行转移相同动作。

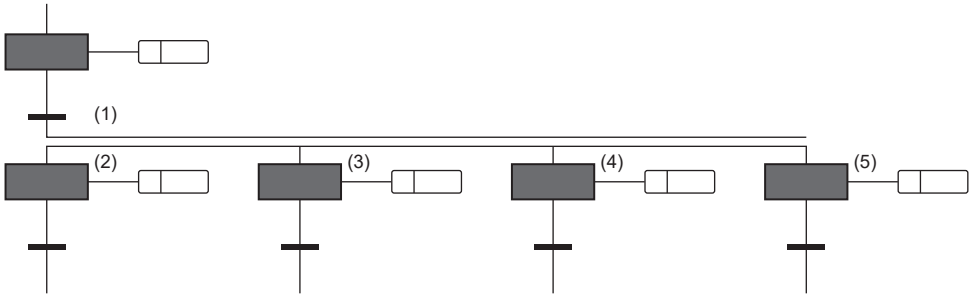


转移条件(2)成立时，步(1)将变为非激活状态。
冗余启动的转移目标步(3)激活。

并联转移的情况

■并联分支

下一个扫描中的转移目标将全部变为激活状态。



转移条件(1)成立时，在下一个扫描
冗余启动的转移目标步(2)～步(5)将
全部变为激活状态。

■并联合并

转移源将变为非激活。线圈保持步[SC]将变为保持状态。

程序更改时的动作

SFC程序的更改方法，如下所示。

- 写入至可编程控制器
- RUN中写入
- SFC块RUN中写入*1

可利用上述方法可更改的SFC程序内容如下所示。

更改的示例			写入至可编程控制器		RUN中写入	SFC块RUN中写入*1
			STOP/PAUSE状态	RUN状态		
SFC程序的添加			○	×	×	×
SFC块的添加/删除			○	×	×	○
SFC块的更改	SFC图的更改	步・转移条件的添加/删除	○	×	×	○
		转移条件(分支/合并/跳转)的更改	○	×	×	○
		步的属性更改	○	×	×	○
	SFC图内的更改	动作输出程序的更改	○	×	○	○
		转移条件程序的更改	○	×	○	○
	块信息的更改		○	×	×	×

*1 关于SFC块RUN中写入的有关内容，请参阅下述内容。
☞ 107页 SFC块RUN中写入
此外，使用SFC块RUN中写入时，应确认CPU模块及工程工具的版本。
☞ 116页 功能的添加和更改

通过写入至可编程控制器进行程序更改

通过写入至可编程控制器进行程序更改后的动作，如下所示。

SM322(SFC程序的启动状态)	写入前后的程序更改有无	
	有更改	无更改
OFF(初始启动)	初始启动	初始启动
ON(继续启动)	初始启动	继续启动

■STOP→RUN操作

在SFC程序的动作中(RUN中)将CPU模块置为STOP的情况下，在STOP→RUN时软元件的状态与SFC程序的激活状态均恢复为STOP前的状态。与CPU参数的“SFC程序启动模式设置”无关，将变为继续启动。

在STOP过程中，将顺控程序文件(包括SFC程序)、FB文件中的任意一个写入到CPU模块中的情况下，在RUN时如果SFC程序存在，将变为初始启动。但是，在程序的写入前后无更改的情况下，有可能进行继续启动。(☞ 91页 SFC程序启动模式设置)

■注意事项

- 通过写入至可编程控制器进行程序更改后，应在进行了一次复位后，再执行SFC程序。
- CPU参数的“SFC程序启动模式设置”为“继续启动”的情况下，应将SM322(SFC程序的启动状态)置为OFF(初始启动)后，通过写入至可编程控制器进行程序更改。程序更改后，应在初始启动SFC程序之后，再次将SM322置为ON(继续启动)。

通过RUN中写入更改程序

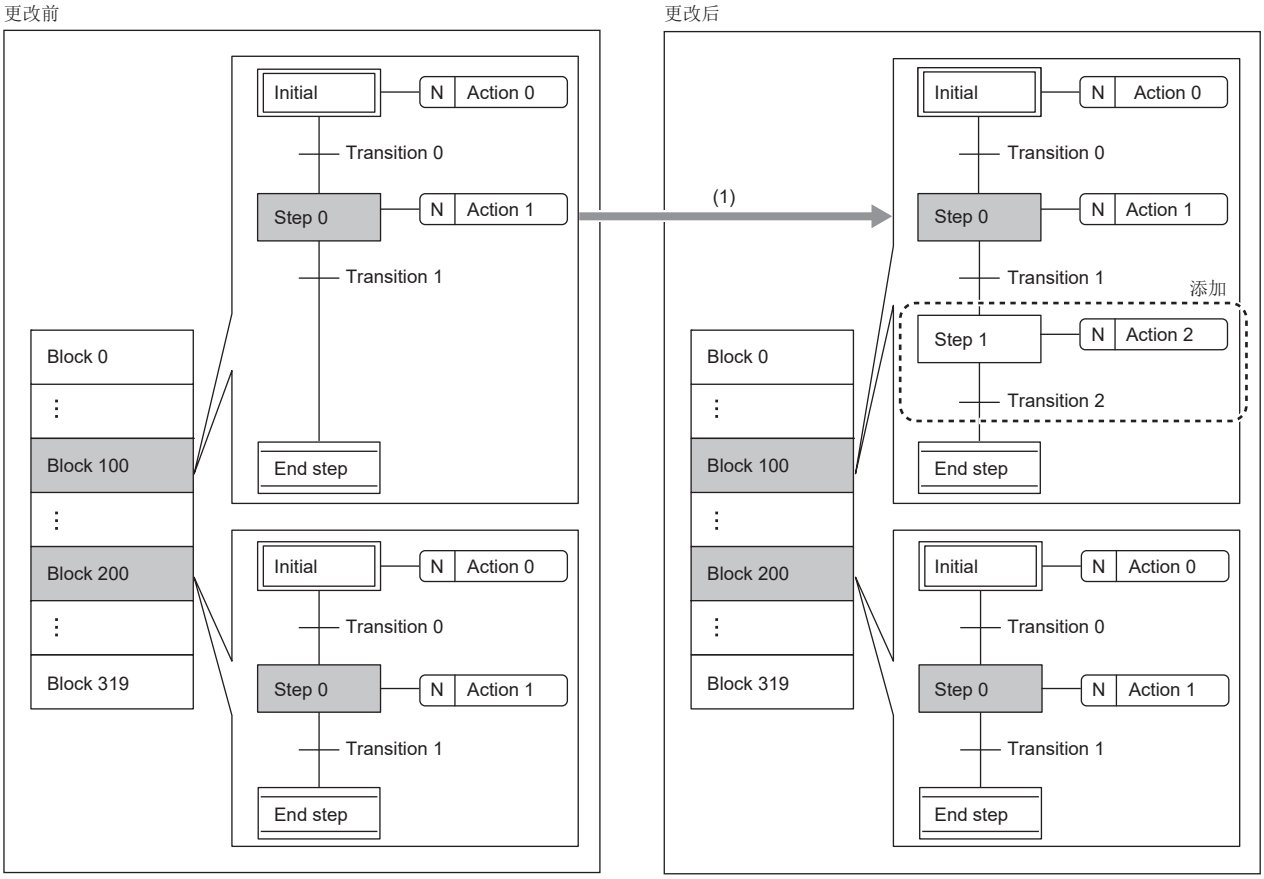
通过RUN中写入更改程序后，与CPU参数的SFC程序启动模式设置无关，必将继续启动。

■注意事项

STOP中写入至可编程控制器后，在CPU模块变为RUN之前，无法对SFC程序执行RUN中写入。应在使CPU模块运行后，对SFC程序执行RUN中写入。

SFC块RUN中写入

可以以块为单位更改SFC程序。即使RUN中也可以在持续激活状态下，以块为单位更改程序，因此可以提高SFC程序的调试及维护的效率。



：非激活块或步

：激活中的块或步

(1) 可以更改激活中SFC块的激活步以外的程序。

限制事项

使用SFC块RUN中写入时，应确认CPU模块及工程工具的版本。
关于CPU模块和工程工具的版本，请参阅下述内容。
☞ 116页 功能的添加和更改

■根据程序的执行类型决定能否执行

仅扫描执行型程序中可执行。(待机型程序中无法执行。)

■SM321(启动/停止SFC程序)为OFF时的SFC块RUN中写入

SM321=OFF时执行SFC块RUN中写入时，与下述设置无关，必将变为初始启动。

- CPU参数的SFC继续启动设置(☞ 91页 SFC程序启动模式设置)
- SM322(SFC程序的启动状态)

■块的更改/添加/删除

下表为SFC块RUN中写入的块的更改/添加/删除的相关内容。

操作	内容	
	对象块为非激活时	对象块为激活中时
块的更改	可更改CPU模块内的SFC程序中既存的SFC块的程序。	可更改CPU模块内的SFC程序中既存的SFC块的程序。但无法进行激活步的删除、属性更改、步No.更改。
块的添加	可向CPU模块内的SFC程序添加新的SFC块。	—
块的删除	• 可从CPU模块中的SFC程序删除指定的SFC块。 • CPU模块的SFC程序中没有对象块时，无法执行。	无法删除激活中的SFC块。

要点

CPU模块为STOP、PAUSE状态时，激活的步保持激活状态。因此，实施步的删除、属性更改、步No.更改时，应在将保持激活状态的步置为非激活化后，再执行SFC块RUN中写入。

■程序替换范围

对象块中，仅替换添加/更改的步及转移条件、添加/更改的动作输出及转移条件内的电路。此外，添加/更改对象的动作输出及转移条件内的电路中的上升沿、下降沿指令的前一次执行信息将被初始化。

■执行状态的确认

SFC块RUN中写入的执行状态可通过SM329(SFC块RUN中写入执行中标志)/SD329(SFC块RUN中写入对象块No.)确认。(MELSEC iQ-F FX5用户手册(应用篇))

■引导运行中的SFC块RUN中写入

通过SD存储卡在引导运行中执行SFC块RUN中写入时，也可以更改引导源SD存储卡内的该文件。

■SFC块RUN中写入中，试图启动对象块或步时的动作

试图在RUN中写入执行过程中启动SFC块RUN中写入的对象块或步时，对象块或步不会启动。块或步的各启动类型的动作如下所示。

对象块或步的启动类型方法(激活方法)	块或步启动时的动作
通过CPU参数进行的自动启动	SFC块RUN中写入完成前不启动。SFC块RUN中写入完成后自动启动。
块启动步(无结束检查)	• SFC块RUN中写入完成前对象块不启动，保持待机。即使步随附的转移条件成立，也不转移到下一个步。 • SFC块RUN中写入完成后，启动对象块。转移条件成立时，转移到下一个步。
块启动步(有结束检查)	• SFC块RUN中写入完成前对象块不启动，保持待机。 • SFC块RUN中写入完成后，启动对象块。块结束后，随附的转移条件成立时转移到下一个步。
SFC控制指令(SET BL□、SET S□、SET BL□\S□命令)	不启动对象块。指令触点持续ON时，在SFC块RUN中写入完成后启动对象块。
通过工程工具进行的块启动*1	不启动对象块。忽略请求。(SFC块RUN中写入完成前系统不启动该块。)
通过转移条件成立进行的步启动(激活化)	• SFC块RUN中写入完成前，因之前的转移条件成立，RUN中写入对象的SFC步不启动。(激活状态不转移，保持待机。) • SFC块RUN中写入完成后，转移条件成立时，激活状态将转移。

*1 表示通过监看窗口的BL□、BL□\S□、软元件/缓冲存储器批量监视的启动以及通过调试→SFC步控制的启动。

■注意事项

- 请勿在SFC块RUN中写入中(包含传送至程序存储器)执行电源OFF或复位。如果执行了电源OFF或复位，应再次进行写入至可编程控制器的操作。
- SFC块RUN中写入无法与下述工程工具上的操作同时执行。
  - RUN中更改电路块、SFC块RUN中写入
  - 写入至可编程控制器(不包括软元件/局部软元件/全局标签/局部标签数据)
  - 存储器的初始化
- 从控制中删除不需要的OUT指令(线圈)时，应在确认OUT指令为OFF后再进行删除。如果不OFF即删除OUT指令，则将保持输出。
- 对象电路中包含FB调用时，调用的FB定义内的信号流的初始化有无根据FB的种类而异。

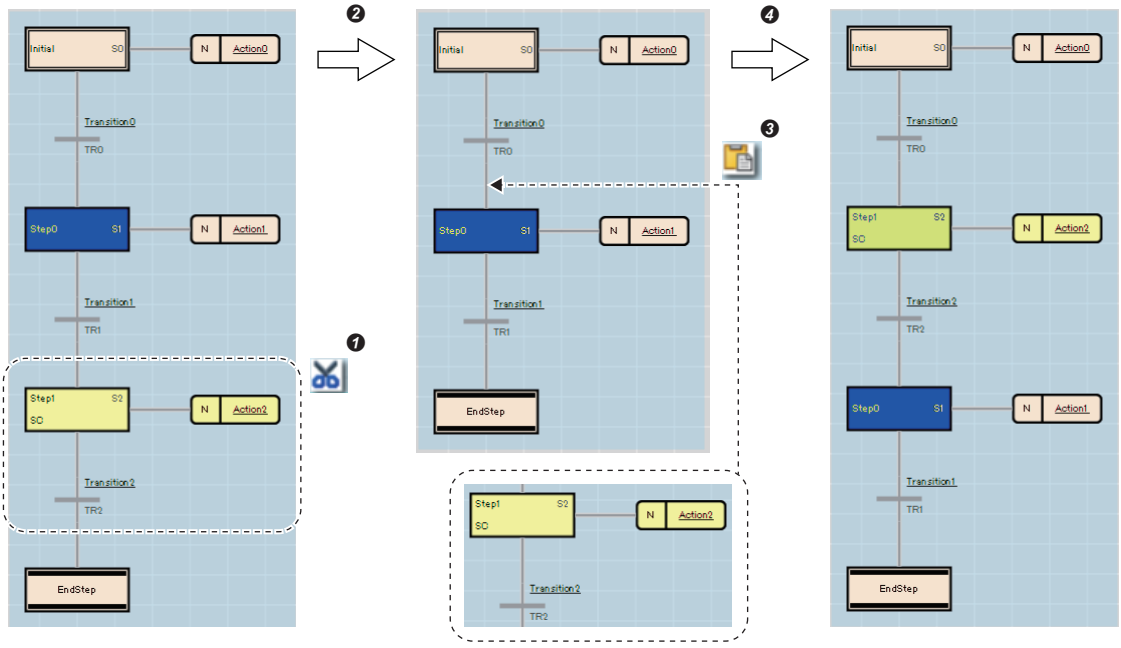
功能块的种类		初始化有无
全局FB	子程序类型	×
	宏类型	○*1
局部FB	子程序类型	×
	宏类型	○*1

*1　更改电路内调用子程序类型FB时，调用的子程序类型FB的FB定义内的上升沿、下降沿指令等的上一次执行信息不会被初始化。

- SFC块RUN中写入时将一部分区间视为扫描监视对象外，因此即使扫描时间超过扫描时间监视时间(WDT)设置中设置的时间，也有可能检测不出WDT时间超过。
- SFC块RUN中写入中执行的指令中检测出错误时，程序位置信息(步No.)变为写入中的程序的步No.，不变为写入完成后的程序的步No.。
- 下述情况时，无法执行SFC块RUN中写入。
  - 参数设置中未登录对象的程序文件时
  - SFC步数超过CPU参数的软元件设置中设置的步进继电器的点数时
  - 将CPU模块置为STOP状态，指定程序或参数，执行写入至可编程控制器时由STOP→RUN之前的期间
  - 发生不可执行程序(错误代码：3204H)时
- 执行包含横跨激活步的步的移动的SFC块RUN中写入时，应按照下述步骤实施。

❶暂时剪切移动的步。
❷执行SFC块RUN中写入。
❸将剪切的步粘贴到目标位置上。
❹再次执行SFC块RUN中写入。

省略上述❷，按照❶→❸→❹的顺序实施时，有可能无法执行SFC块RUN中写入。



SFC程序的动作确认

SFC程序的动作确认中可使用的工程工具的功能如下所示。

- 监视
- 查看
- 软元件/缓冲存储器批量监视
- SFC步控制
- SFC块一览表
- SFC块批量监视
- 激活步监视

要点

- 关于通过工程工具进行的SFC程序的动作确认有关内容，请参阅下述手册。

GX Works3操作手册

- 使用SFC程序时，不能通过工程工具向步进继电器(S)进行写入。可对步进继电器(S)进行读取。

附录

附1　使用MC/MCR指令控制EN的动作

将FB的固有属性设置中的“使用MC/MCR控制EN”置为有效时，FB内所使用的指令、软元件/标签的动作如下所示。

FB内所使用的指令、软元件/标签	FB内所使用的指令、软元件/标签的状态	
	将“使用MC/MCR控制EN”选为“是”时	将“使用MC/MCR控制EN”选为“否”时
上升沿/下降沿指令(PLS指令、脉冲化指令(□P)*1	在下一个EN变为ON时，条件触点若成立，则执行指令。	但在下一个EN变为ON时，即使条件触点成立，也有可能发生不执行指令的情况。
定时器(低速定时器/定时器/高速定时器)	计数值变为0，且线圈、触点也变为OFF。	保持现状。
累计定时器(低速定时器/定时器/高速定时器)、计数器、长计数器	线圈变为OFF，但计数值、触点仍保持现状。	保持现状。
OUT指令的软元件部中指定的软元件	强制变为OFF。	保持现状。

*1　线圈侧中指定的指令为对象。

限制事项

将“使用MC/MCR控制EN”选为“是”的情况下，在该FB处于执行中时，请勿使用MC/MCR指令。使用了MC/MCR指令的情况下，EN的控制可能无法正确动作。

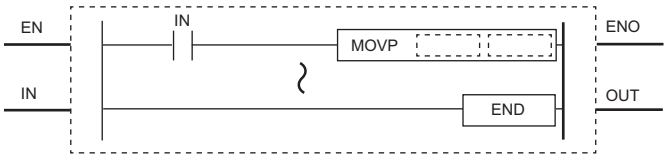
附

上升沿/下降沿指令的动作

上升沿/下降沿指令的动作如下所示。

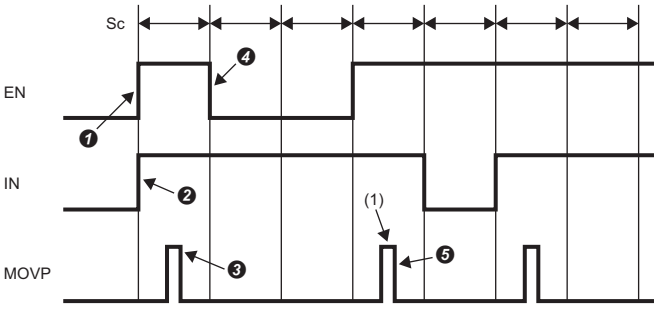
例

使用上升沿指令的子程序类型FB



■将“使用MC/MCR控制EN”选为“是”时

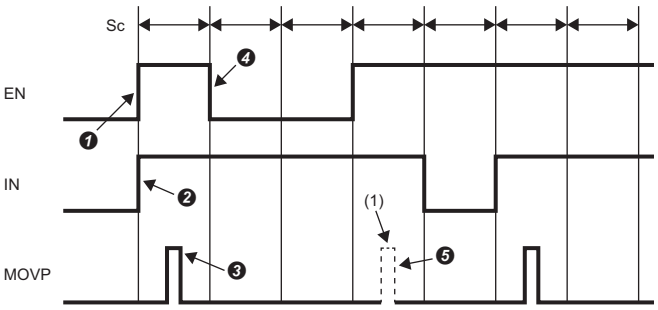
在EN变为ON时，条件触点若成立，则执行指令。(图中(1))



Sc：扫描
❶将EN置为ON。(用户操作)
❷将IN置为ON。(用户操作)
❸执行MOVP指令。(CPU模块动作)
❹将EN置为OFF。(用户操作)
❺执行MOVP指令。(CPU模块动作)

■将“使用MC/MCR控制EN”选为“否”时

EN为OFF时，根据条件触点状态，指令的动作将有所不同。(图中(1))



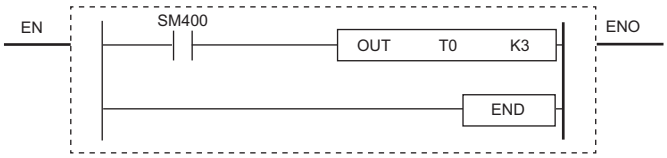
Sc：扫描
❶将EN置为ON。(用户操作)
❷将IN置为ON。(用户操作)
❸执行MOVP指令。(CPU模块动作)
❹将EN置为OFF。(用户操作)
❺❹中，若条件触点在EN变为OFF前变为OFF，则执行MOVP指令。(CPU模块动作)(❹中，若条件触点在EN变为OFF前变为ON，则不执行MOVP指令。)

定时器(低速定时器/定时器/高速定时器)

定时器(低速定时器/定时器/高速定时器)的动作如下所示。

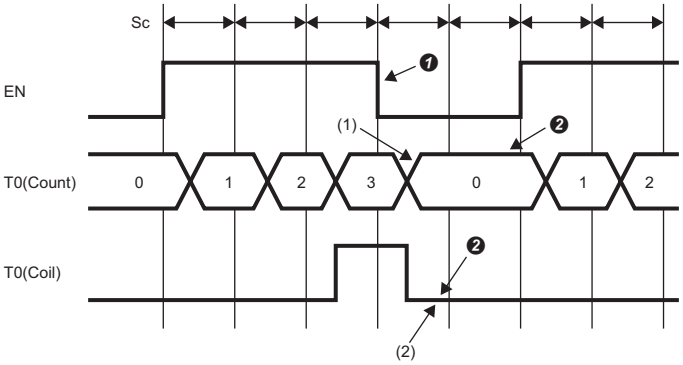
例

使用低速定时器的子程序类型FB



■将“使用MC/MCR控制EN”选为“是”时

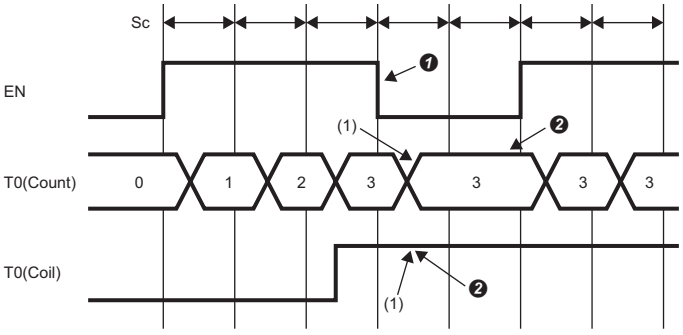
计数值变为0。(图中(1))此外，线圈变为OFF。(图中(2))



Sc：扫描
T0(Count)：T0(计数值)
T0(Coil)：T0(线圈)
❶将EN置为OFF。(用户操作)
❷线圈变为OFF，清除定时器值、计数值。(CPU模块动作)

■将“使用MC/MCR控制EN”选为“否”时

计数值和线圈皆保持现状。(图中(1))



Sc：扫描
T0(Count)：T0(计数值)
T0(Coil)：T0(线圈)
❶将EN置为OFF。(用户操作)
❷保持值。(CPU模块动作)

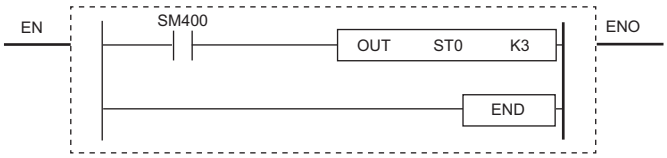
附

累计定时器(低速定时器/定时器/高速定时器)、计数器、长计数器的动作

累计定时器(低速定时器/定时器/高速定时器)、计数器、长计数器的动作如下所示。

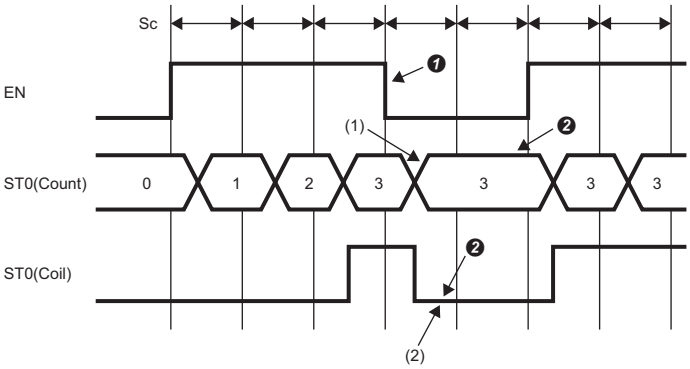
例

使用低速累计定时器的子程序类型FB



■将“使用MC/MCR控制EN”选为“是”时

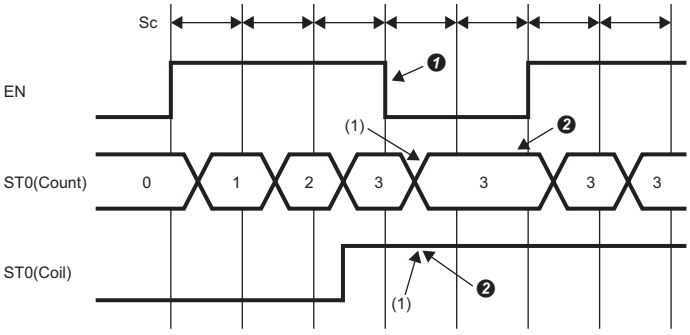
计数值保持现状。(图中(1))此外，线圈变为OFF。(图中(2))



Sc: 扫描
ST0(Count)：T0(计数值)
ST0(Coil)：T0(线圈)
❶将EN置为OFF。(用户操作)
❷线圈变为OFF，但计数值、触点仍保持现状。(CPU模块动作)

■将“使用MC/MCR控制EN”选为“否”时

计数值和线圈皆保持现状。(图中(1))



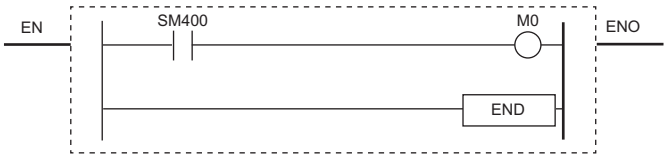
Sc: 扫描
ST0(Count)：T0(计数值)
ST0(Coil)：T0(线圈)
❶将EN置为OFF。(用户操作)
❷保持值。(CPU模块动作)

OUT指令的软元件部中指定的软元件的动作

OUT指令的软元件部中指定的软元件的动作如下所示。

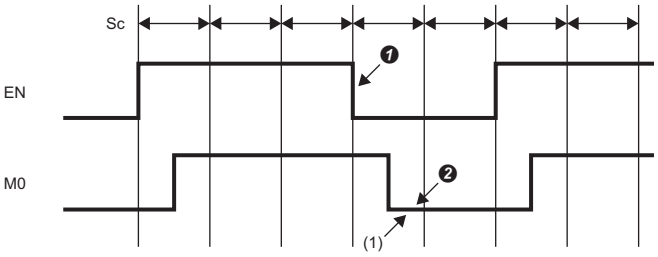
例

OUT指令的软元件部中使用M0的子程序类型FB



■将“使用MC/MCR控制EN”选为“是”时

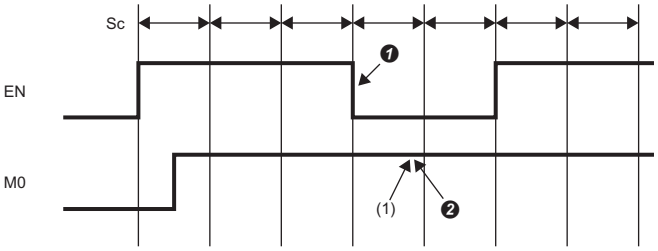
M0强制变为OFF。(图中(1))



Sc：扫描
❶将EN置为OFF。(用户操作)
❷将线圈输出置为OFF。(CPU模块动作)

■将“使用MC/MCR控制EN”选为“否”时

M0保持现状。(图中(1))



Sc：扫描
❶将EN置为OFF。(用户操作)
❷保持线圈。(CPU模块动作)

附

附2　功能的添加和更改

在CPU模块及工程工具中添加或更改的功能和支持的CPU模块固件版本及工程工具的软件版本如下所示。
固件版本可以在模块诊断(CPU诊断)中确认。关于模块诊断(CPU诊断)，请参照以下手册。
📖MELSEC iQ-F FX5S/FX5UJ/FX5U/FX5UC用户手册(硬件篇)
关于软件版本，请参阅📖GX Works3操作手册。

FX5S CPU模块

添加/更改功能	支持CPU模块的固件版本	支持工程工具的软件版本	参照
支持FX5S CPU模块 • 梯形图语言 • ST语言 • FBD/LD语言	从首批产品开始支持	“1.080J”及以后	—
支持Unicode字符串	从首批产品开始支持	“1.080J”及以后	—
支持SFC程序	“1.010”及以后	“1.095Z”及以后	63页 SFC程序

FX5UJ CPU模块

添加/更改功能	支持CPU模块的固件版本	支持工程工具的软件版本	参照
支持FX5UJ CPU模块 • 梯形图语言 • ST语言 • FBD/LD语言	从首批产品开始支持	“1.060N”及以后	—
支持Unicode字符串	“1.030”及以后	“1.085P”及以后	—
支持SFC程序	“1.050”及以后	“1.095Z”及以后	63页 SFC程序

FX5U/FX5UC CPU模块

添加/更改功能	支持CPU模块的固件版本	支持工程工具的软件版本	参照
支持FX5U/FX5UC CPU模块 • 梯形图语言 • ST语言 • FBD/LD语言	从首批产品开始支持	从首批产品开始支持	—
支持SFC程序	“1.220”及以后	“1.070Y”及以后	63页 SFC程序
支持Unicode字符串	“1.240”及以后	“1.075D”及以后	—
支持SFC块RUN中写入	“1.290”及以后	“1.095Z”及以后	107页 SFC块RUN中写入
在步的属性中添加SE、ST	“1.290”及以后	“1.095Z”及以后	70页 步的属性

备忘录

索引

符号

A

B

C

D

E

F

G

H

I

J

K

L

M

索

修订记录

制作日期	版本号	内容
2015年2月	A	制作初版
2015年9月	B	■添加/修改位置 4.4节、第7章
2018年7月	C	■添加/修改位置 关联手册、术语、3.2节、4.4节、5.2节、5.3节、6.1节、7.1节
2018年12月	D	■添加/修改位置 关联手册、术语、第2章、第3章、6.1节、附1、商标
2019年10月	E	■添加机型 FX5UJ CPU模块 ■添加/修改位置 关联手册、术语、3.4节、5.3节、6.1节
2020年5月	F	■添加/修改位置 关联手册、术语、3.2节、6.1节、商标
2020年10月	G	■添加功能 SFC程序 ■添加/修改位置 关联手册、1章、5.2节、6.1节、7.1节、8章、附2
2021年4月	H	■添加/修改位置 关联手册、术语、3.2节、3.3节、4.3节、4.4节、4.6节、6.1节、7.1节、附2、商标
2021年10月	J	■添加机型 FX5S CPU模块 ■添加/修改位置 关联手册、术语、总称/简称、1章、附2、关于保修
2022年4月	K	■添加/修改位置 3.2节、3.3节、5.2节、7.1节、7.2节、8.2节、附2
2023年4月	L	■添加/修改位置 第1章、第8章、附2

日语版手册编号： JY997D54601L

关于保修

在使用时，请务必确认一下以下的有关产品保证方面的内容。

1. 免费保修期和免费保修范围

在产品的免费保修期内，如是由于本公司的原因导致产品发生故障和不良(以下统称为故障)时，用户可以通过当初购买的代理店或是本公司的服务网络，提出要求免费维修。
但是、如果要求去海外出差进行维修时，会收取派遣技术人员所需的实际费用。
此外，由于更换故障模块而产生的现场的重新调试、试运行等情况皆不属于本公司责任范围。

【免费保修期】
产品的免费保修期为用户买入后或是投入到指定的场所后的12个月以内。但是，由于本公司的产品出厂后一般的流通时间最长为6个月，所以从制造日期开始算起的18个月为免费保修期的上限。
此外，维修品的免费保修期不得超过维修前的保证时间而变得更长。

【免费保修范围】
(1) 只限于使用状态、使用方法以及使用环境等都遵照使用说明书、用户手册、产品上的注意事项等中记载的条件、注意事项等，在正常的状态下使用的情况。

(2) 即使是在免费保修期内，但是如果属于下列的情况的话就变成收费的维修。

①由于用户的保管和使用不当、不注意、过失等等引起的故障以及用户的硬件或是软件设计不当引起的故障。

②由于用户擅自改动产品而引起的故障。

③将本公司产品装入用户的设备中使用时，如果根据用户设备所受的法规规定设置了安全装置或是行业公认应该配备的功能构造等情况下，视为应该可以避免的故障。

④通过正常维护・更换使用说明书等中记载的易耗品(电池、背光灯、保险丝等)可以预防的故障。

⑤即使按照正常的使用方法，但是继电器触点或是触点到寿命的情况。

⑥由于火灾、电压不正常等不可抗力导致的外部原因，以及地震、雷电、洪水灾害等天灾引起的故障。

⑦在本公司产品出厂时的科学技术水平下不能预见的原因引起的故障。

⑧其他、认为非公司责任而引起的故障。

2. 停产后的收费保修期

(1) 本公司接受的收费维修品为产品停产后的7年内。有关停产的信息，都公布在本公司的技术新闻等中。

(2) 不提供停产后的产品（包括附属品）。

3. 在海外的服务

对于海外的用户，本公司的各个地域的海外FA中心都接收维修。但是，各地的FA中心所具备的维修条件有所不同，望用户谅解。

4. 机会损失和间接损失不在质保责任范围内

无论是否在免费质保期内，凡以下事由三菱电机将不承担责任。

(1) 任何非三菱电机责任原因而导致的损失。

(2) 因三菱电机产品故障而引起的用户机会损失、利润损失。

(3) 无论三菱电机能否预测，由特殊原因而导致的损失和间接损失、事故赔偿、以及三菱电机产品以外的损伤。

(4) 对于用户更换设备、现场机械设备的再调试、运行测试及其它作业等的补偿。

5. 产品规格的变更

产品样本、手册或技术资料中所记载的规格有时会未经通知就变更，还望用户能够预先询问了解。

6. 关于产品的适用范围

(1) 使用本公司MELSEC iQ-F/FX/F微型可编程控制器时，要考虑到万一可编程控制器出现故障・不良等情况时也不会导致重大事故的使用用途，以及以在出现故障・不良时起到作用。将以上这些作为条件加以考虑。在设备外部系统地做好后备或是安全功能。

(2) 本公司的可编程控制器是针对普通的工业用途而设计和制造的产品。因此，在各电力公司的原子能发电站以及用于其他发电站等对公众有很大影响的用途中，以及用于各铁路公司以及政府部门等要求特别的质量保证体系的用途中时，不适合使用可编程控制器。
此外，对于航空、医疗、燃烧、燃料装置、人工搬运装置、娱乐设备、安全机械等预计会对人身性命和财产产生重大影响的用途，也不适用可编程控制器。
但是，即使是上述的用途，用户只要事先与本公司的营业窗口联系，并认可在其特定的用途下可以不要求特别的质量时，还是可以通过交换必须的资料后，选用可编程控制器的。

(3) 因拒绝服务攻击(DoS攻击)、非法访问、电脑病毒以及其他网络攻击引发的可编程控制器与系统方面的各种问题，三菱电机不承担责任。

商标

Anywire and AnyWireASLINK are either registered trademarks or trademarks of Anywire Corporation.
Unicode is either a registered trademark or a trademark of Unicode, Inc. in the United States and other countries.
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.

手册编号：JY997D58801L

三菱电机自动化(中国)有限公司

地址：上海市虹桥路1386号三菱电机自动化中心
邮编：200336
电话：86-21-2322-3030　传真：86-21-2322-3000
官网：https://www.MitsubishiElectric-FA.cn
技术支持热线　400-821-3030

官方微信

内容如有更改　恕不另行通知