MITSUBISHI ELECTRIC

三菱电机 通用 可编程控制器

MELSEC iQ-R series

MELSEC iQ-R
编程手册(程序设计篇)

安全注意事项

(使用之前请务必阅读)

使用MELSEC iQ-R系列可编程控制器之前，应仔细阅读各产品手册及各产品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阅读，并应将本手册交给最终用户。

关于产品的应用

(1) 在使用三菱可编程控制器时，应该符合以下条件：即使在可编程控制器设备出现问题或故障时也不会导致重大事故，并且应在设备外部系统地配备能应付任何问题或故障的备用设备及失效安全功能。

(2) 三菱可编程控制器是以一般工业用途等为对象设计和制造的通用产品。

因此，三菱可编程控制器不应用于以下设备・系统等特殊用途。如果用于以下特殊用途，对于三菱可编程控制器的质量、性能、安全等所有相关责任（包括但不限于债务未履行责任、瑕疵担保责任、质量保证责任、违法行为责任、制造物责任），三菱电机将不负责。

- 面向各电力公司的核电站以及其它发电厂等对公众有较大影响的用途。
- 用于各铁路公司或公用设施目的等有特殊质量保证体系要求的用途。
- 航空航天、医疗、铁路、焚烧・燃料装置、载人移动设备、载人运输装置、娱乐设备、安全设备等预计对人身财产有较大影响的用途。

然而，对于上述应用，如果在限定于具体用途，无需特殊质量（超出一般规格的质量等）要求的条件下，经过三菱电机的判断也可以使用三菱可编程控制器，详细情况请与当地三菱电机代表机构协商。

・使用SIL2过程CPU时

(1) 尽管安全控制器已经取得了德国TUV Rheinland的国际安全标准IEC61508和IEC61511的产品可靠性认证，但这并不保证本产品不发生任何故障。本产品的用户应遵守所有现行的安全标准、规则或法律，并应对本产品所安装或使用的系统采取适当的安全措施，除了本产品之外还应当同时采取其它的安全措施。对于如果遵守了现行的安全标准、规则或法律则可以预防的损害，三菱电机公司(简称三菱电机)不负任何责任。

(2) 三菱电机禁止将本产品用于可能涉及人员生命健康安全和重大财产安全的用途，如果违反了三菱电机的指示将其用于这些用途，对于由此引起的一切责任(包括但不限于债务未履行责任、瑕疵担保责任、质量保证责任、违法行为责任、制造物责任)，三菱电机将不负责。

1)火力・水力・核能发电厂

2)火车・铁路系统、飞机、航空管理、其它交通系统

3)医院、医疗及生命维持相关设备的应用

4)娱乐设备

5)焚烧和燃料装置

6)核物质、有害物质及化学物质的处理设备

7)采矿、挖掘

8)其它上述1)～7)中未包含的涉及人员生命、健康或重大财产安全的用途

• 使用安全CPU时

(1) 尽管安全控制器已经取得了德国TUV Rheinland的国际安全标准IEC61508和EN954-1/ISO13849-1的产品可靠性认证，但这并不保证本产品不发生任何故障。本产品的用户应遵守所有现行的安全标准、规则或法律，并应对本产品所安装或使用的系统采取适当的安全措施，除了本产品之外还应当同时采取其它的安全措施。对于如果遵守了现行的安全标准、规则或法律则可以预防的损害，三菱电机公司(简称三菱电机)不负任何责任。

(2) 三菱电机禁止将本产品用于可能涉及人员生命健康安全和重大财产安全的用途，如果违反了三菱电机的指示将其用于这些用途，对于由此引起的一切责任(包括但不限于债务未履行责任、瑕疵担保责任、质量保证责任、违法行为责任、制造物责任)，三菱电机将不负责。

1)火力・水力・核能发电厂

2)火车・铁路系统、飞机、航空管理、其它交通系统

3)医院、医疗及生命维持相关设备的应用

4)娱乐设备

5)焚烧和燃料装置

6)核物质、有害物质及化学物质的处理设备

7)采矿、挖掘

8)其它上述1)～7)中未包含的涉及人员生命、健康或重大财产安全的用途

前言

在此非常感谢贵方购买了三菱电机可编程控制器MELSEC iQ-R系列产品。
本手册是用于让用户了解进行编程时必要的程序配置及使用的数据有关内容的手册。
在使用之前应熟读本手册及关联手册，在充分了解MELSEC iQ-R系列可编程控制器的功能・性能的基础上正确地使用本产品。
此外，将本手册中介绍的程序示例应用于实际系统的情况下，应充分验证对象系统中不存在控制方面的问题。
应将本手册交给最终用户。

要点

在本手册中，主要使用标签进行说明。软元件也可以像标签一样使用。

目录

关联手册

要取得最新的e-Manual及手册PDF，请向当地三菱电机代理店咨询。

手册名称[手册编号]	内容	提供形式
MELSEC iQ-R 编程手册(程序设计篇) [SH-081319CHN](本手册)	记载了梯形图、ST、FBD/LD、SFC的程序规格有关内容。	e-Manual PDF
MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇) [SH-081322CHN]	记载了CPU模块的指令、通用函数/通用FB的有关内容。	e-Manual PDF
MELSEC iQ-R 编程手册(过程控制FB/指令篇) [SH-081750CHN]	记载了过程控制中特化的通用过程FB、标签访问FB、标签FB、过程控制指令的有关内容。	e-Manual PDF
GX Works3 操作手册 [SH-081271CHN]	对GX Works3的系统配置、参数设置、在线功能的操作方法等有关内容进行说明。	e-Manual PDF

要点

e-Manual是可以使用专用工具进行浏览的三菱电机FA电子书籍手册。

e-Manual具有以下特点。

- 可以从多本手册同时搜索需要的信息(跨手册搜索)
- 可以通过手册内的链接浏览其它手册
- 可以通过产品插图的各部分浏览想要了解的硬件规格
- 可以将频繁浏览的信息登录到收藏夹
- 可以将样本程序复制到工程工具中

术语

在本手册中，除非特别标明，否则皆将使用下述术语进行说明。

术语	说明
CPU模块	是MELSEC iQ-R系列CPU模块的总称。
GX Works3	是MELSEC可编程控制器软件包SWnDNC-GXW3的产品名总称。(n表示版本。)
常规/安全共享标签	是常规程序及安全程序中使用的标签。在安全程序及常规程序之间交接数据时使用。
智能功能模块	是A/D、D/A转换模块等具有输入输出以外功能的模块。
工程工具	是MELSEC可编程控制器软件包的产品名。
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)、目标数据(d)、软元件数(n)等的软元件部分的总称。
通信协议支持功能	是可以在GX Works3(通信协议支持功能)使用的功能。 功能概要如下所示。 • 与对象设备相符合的协议的设置 • 协议设置数据的读取/写入
软元件	是CPU模块内部具有的软元件(X、Y、M、D等)。
输入输出模块	是输入模块、输出模块、输入输出混合模块、中断模块的总称。
网络模块	是下述模块的总称。 • 以太网接口模块 • CC-Link IE控制网络模块 • CC-Link IE现场网络模块 • MELSECNET/H网络模块 • MELSECNET/10网络模块 • RnENCPU(网络部)
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。 是指在CPU模块的情况下，用于存储以太网功能的设置值、监视值等的数据及多CPU功能的数据通信中使用的数据等的存储器。
程序部件	是按各功能分别定义的程序单位。通过程序的部件化，在程序分级时可按处理内容及功能将低位处理分为若干个单位，创建各单位的程序。
多CPU系统	是在多个(2～4个)CPU模块中控制各自管理的输入输出模块及智能功能模块的系统。
模块标签	是各模块将特有定义的存储器(输入输出信号及缓冲存储器)以任意字符串表示的标签。对于使用的模块，工程工具将自动生成标签，且该标签可作为全局标签使用。

此外，在使用SIL2过程CPU以及安全CPU时，也用以下用语进行说明。

术语	说明
安全控制	执行安全程序或安全通信，实施机械的控制。发生异常时，使机械安全停止。
安全通信	是进行通过安全通信协议定义的安全层的发送接收处理的通信服务。
安全软元件	是安全程序中可使用的软元件。
安全程序	是用于执行安全控制的程序。
常规CPU	是执行常规控制的MELSEC iQ-R系列的各CPU模块的总称。(与执行安全控制的CPU模块区别时使用。)
常规控制	执行常规程序或常规通信，实施机械的控制。安全可编程控制器以外仅保有常规控制。(与安全控制区别时使用。)
常规通信	是安全通信以外的通信(CC-Link IE现场网络的循环传送与瞬时传送等)。
常规软元件	是CPU模块内部中安全软元件以外的软元件(X、Y、M、D等)。仅在常规程序中可以使用。(与安全软元件区别时使用。)
常规程序	是用于执行顺控程序控制的安全程序以外的程序。(与安全程序区别时使用。)

1 概要

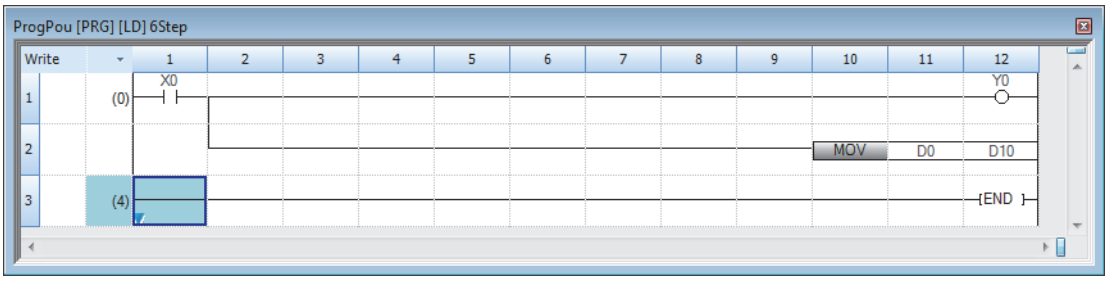
本手册中记载创建程序所需要的程序配置与内容、记述方法等有关内容。
关于在工程工具中的程序创建、编辑、监视方法，请参阅下述手册。
GX Works3 操作手册

程序语言的类型

MELSEC iQ-R系列中，可以根据用途选择最合适的程序语言使用。

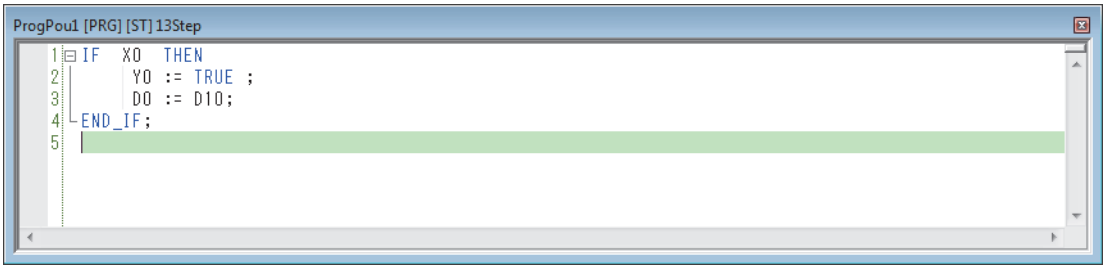
程序语言	内容
梯形图图表语言(梯形图语言)	是用触点及线圈等表示梯形图的图表语言。 梯形图语言是为了进行简单易懂的顺控程序控制，使用符号化的触点及线圈等记述逻辑梯形图的语言。
结构化文本语言(ST语言)	是使用IF语句及运算符等记述程序的文本语言。 与梯形图语言相比，ST语言可简洁而且容易地对难以记述的运算处理进行记述，因此适用于复杂的算术运算及比较运算等的区域。此外，可以与C语言等一样，记述通过条件语句进行的选择分支及重复语句等语法进行的控制，因此可简洁地编写易懂的程序。
FB图表/梯形图语言(FBD/LD语言)	是通过按照数据及信号的流向对特定处理的块、变量部件、常数部件进行连接，记述程序的图表语言。 通过梯形图程序创建时，可以轻松地创建非常复杂的DDC处理程序，提高程序生产能力。
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，使得各程序的执行顺序及执行条件能清楚地表示出来的程序的记述形式。

■梯形图语言



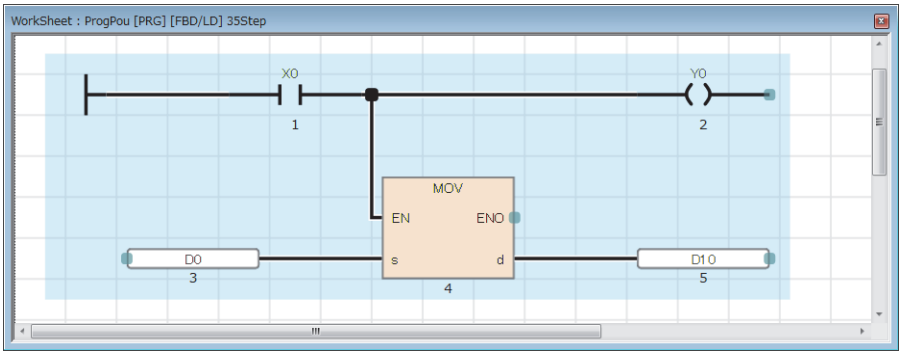
使用梯形图语言的情况下，请参阅下述章节。
☞41页 梯形图语言

■ST语言



使用ST语言的情况下，请参阅下述章节。
☞47页 ST语言

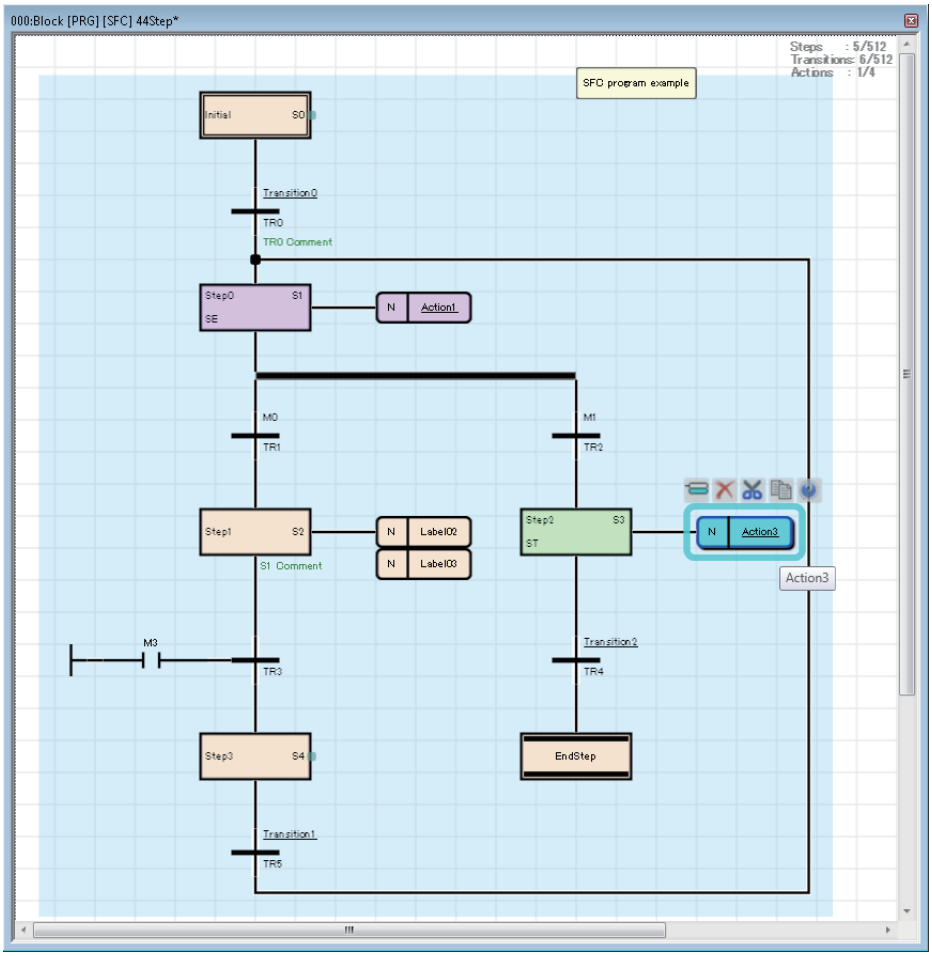
■FBD/LD语言



使用FBD/LD语言的情况下，请参阅下述章节。

☞63页 FBD/LD语言

■SFC程序



使用SFC程序的情况下，请参阅下述章节。

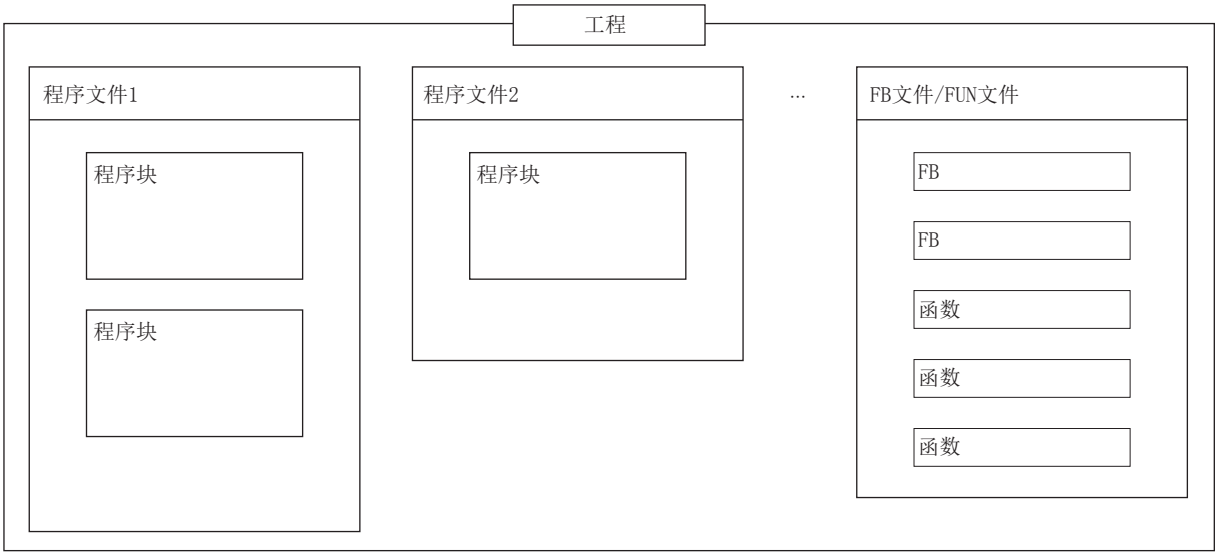
☞76页 SFC程序

要点

- 梯形图语言适合由具有顺控程序控制、逻辑梯形图知识及经验的用户操作。ST语言适合由具有C语言等编程知识及经验的用户操作。FBD/LD语言适合由处理过程控制有关内容的用户操作。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，使程序更简单地转移至不同模块配置的系统中。

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。
☞ 12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。(☞ 13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。

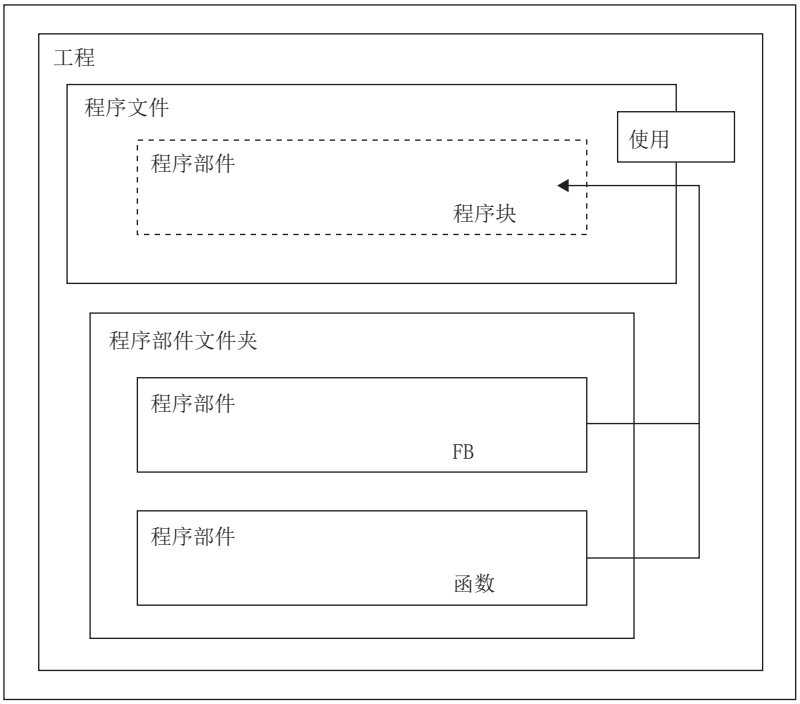
备忘录

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-R CPU模块用户手册(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)

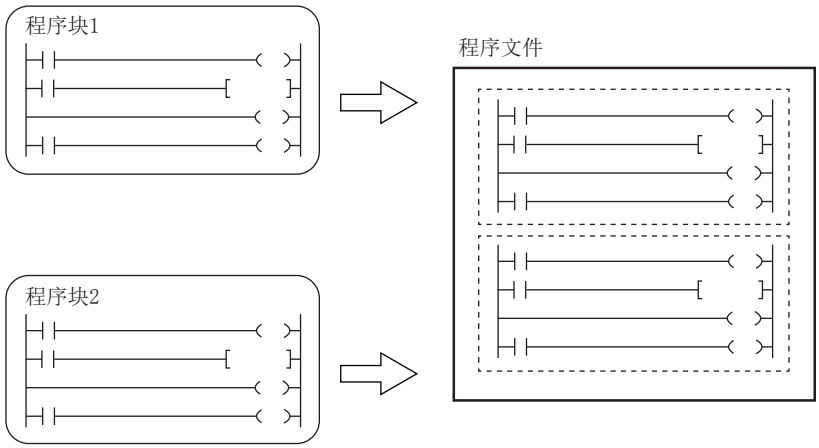
3.1 程序块

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。*1

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、ECALL指令等)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。

*1 在安全程序内不能创建子程序及中断程序。此外，不能通过安全程序执行子程序。

要点

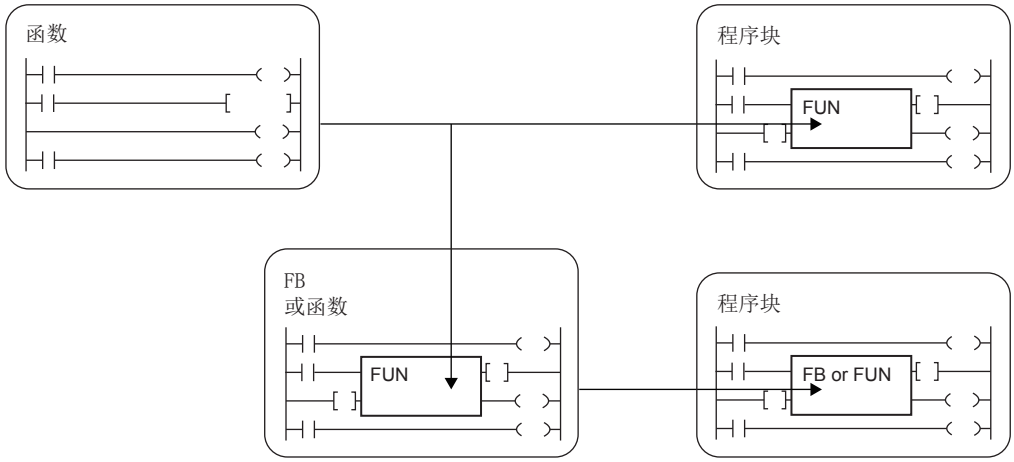
- 子程序以及中断程序是在FEND指令以后进行创建。FEND指令及其以后的程序不作为主程序执行。例如，在第二个程序块的最后使用了FEND指令的情况下，第三个程序块以后将变为子程序或中断程序。
- 为了创建易懂的程序，应在一个程序块中使用成对的FOR指令与NEXT指令、MC指令与MCR指令。
- 简单程序的情况下，在一个程序块内仅记述主程序便可在CPU模块中执行。

关于子程序、中断程序的详细内容，请参阅下述手册。

MELSEC iQ-R CPU模块用户手册(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。

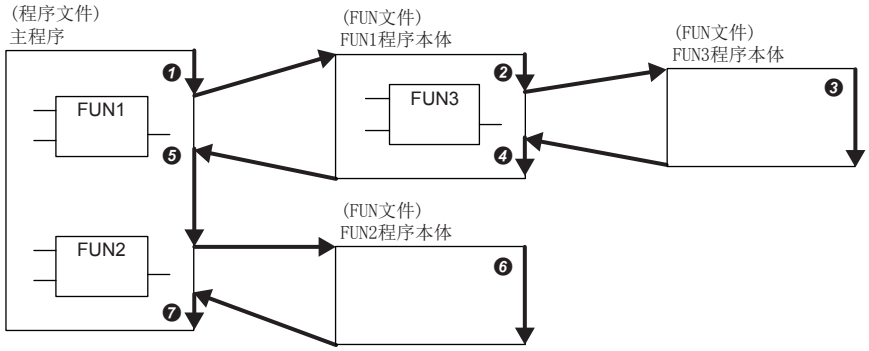


动作概要

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： 3次)
❶～❼表示执行的流程(顺序)。

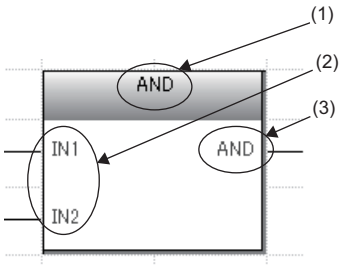


子程序型FB、宏型FB、函数全部合计可进行32次嵌套。

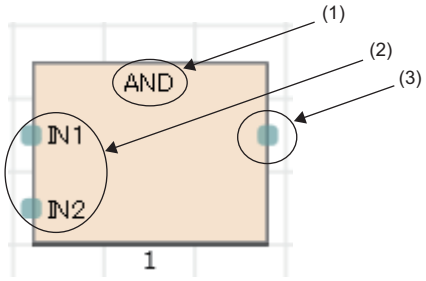
输入变量与输出变量

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-R CPU模块用户手册(入门篇)

■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	○
	局部软元件	×
指针	全局指针	△
	局部指针	×

*1 不能使用下述数据类型。
定时器、累计定时器、计数器、长定时器、长累计定时器、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。

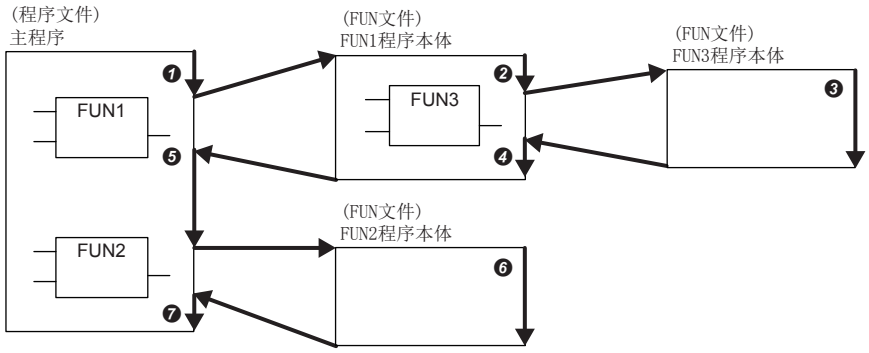
函数中定义的标签

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。

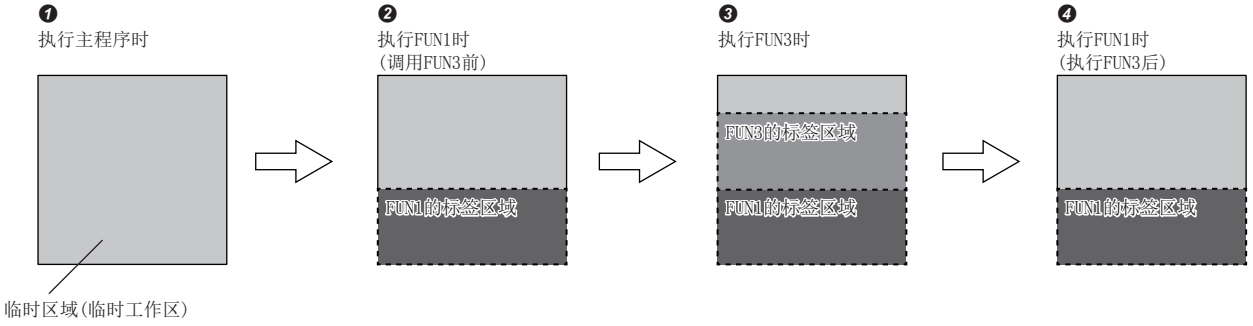
例

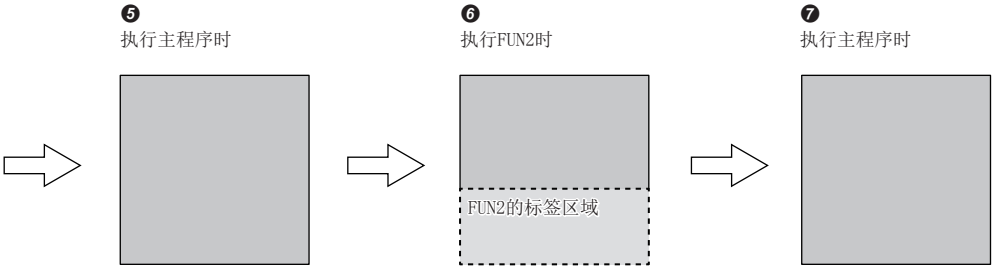
从主程序调用FUN1及FUN2，且FUN1又调用FUN3的情况

(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。

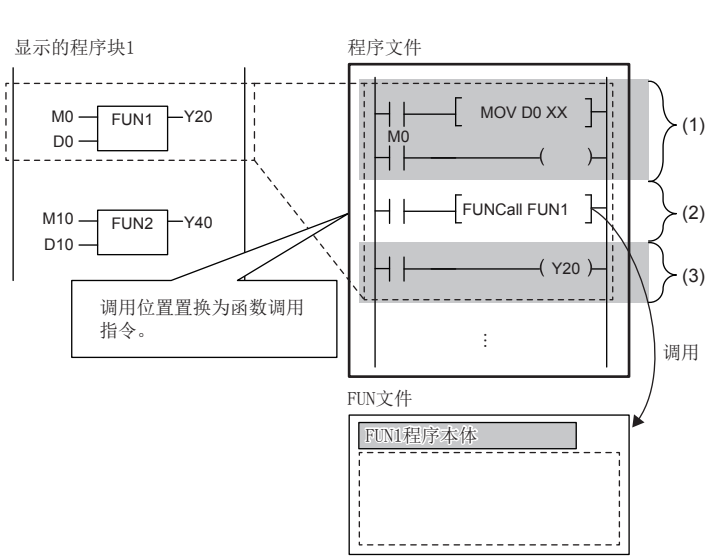
■程序本体

使用了函数的程序本体的步数为指令步数的总计加上22步的值。关于各指令的步数，请参阅下述手册。

MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用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。



• 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-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)
	字[无符号]/位串[16位] 双字[无符号]/位串[32位] 字[带符号] 双字[带符号]	LD+MOV LD+DMOV	
	单精度实数	LD+EMOV	
	双精度实数	LD+EDMOV	
	时间	LD+DMOV	
	字符串	LD+$MOV	
	字符串[Unicode]	LD+$MOV_WS	
	数组、结构体	LD+BMOV	

• 程序本体调用

函数的程序本体的调用需要26步。

• 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

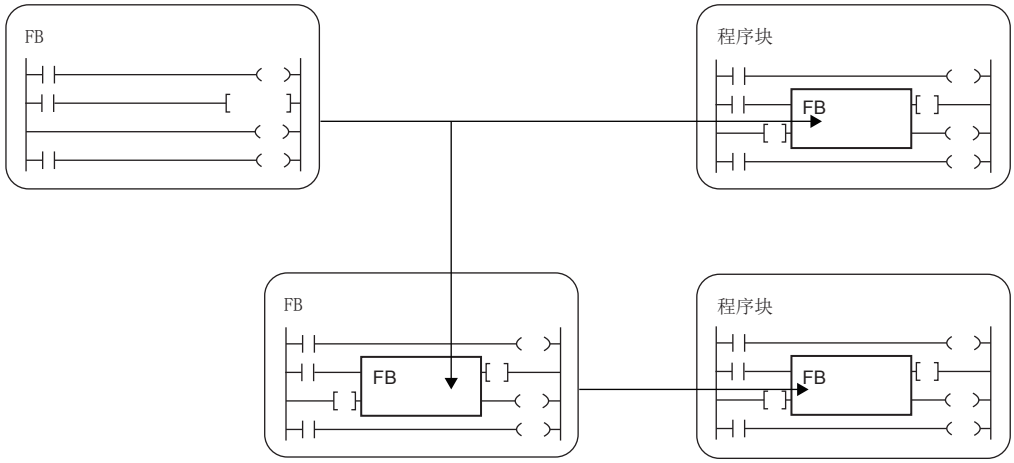
• EN/ENO

EN/ENO所需步数如下所示。

项目	步数
EN	3
ENO	2

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。

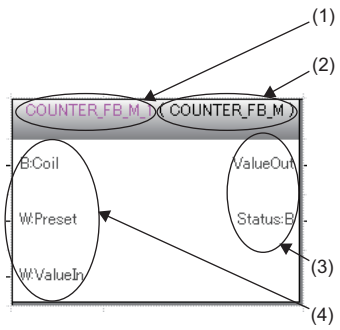


FB与函数不同，不能保持返回值。

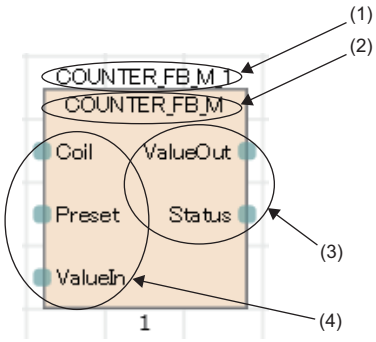
FB能将值保存在变量中，因此也能保持输入状态及处理结果。

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。

☞22页 实例

要点

- 关于通用FB的详细内容，请参阅下述手册。

📖MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)

- 关于过程控制FB的详细内容，请参阅下述手册。

📖MELSEC iQ-R 编程手册(过程控制FB/指令篇)

- 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

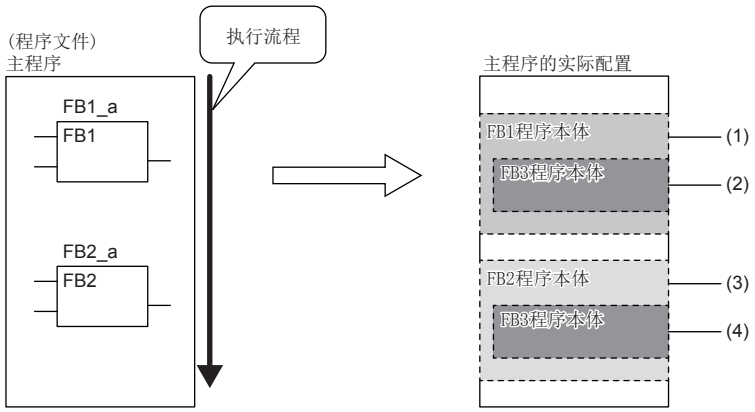
动作概要

■宏型FB

编程时，宏型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



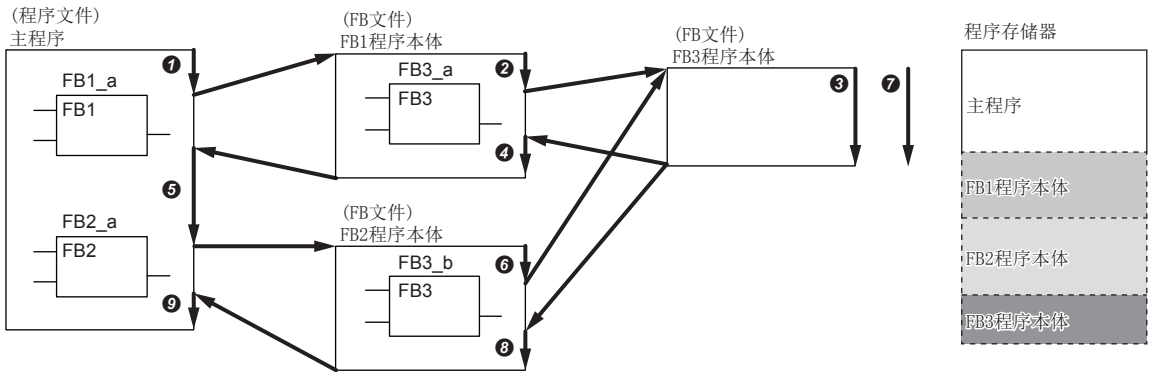
(1) 展开主程序中的FB1程序本体后执行。
(2) 从FB1调用的FB3将在FB1的程序本体内展开。
(3) FB2与FB1同样，FB2程序本体在主程序中展开后执行。
(4) 从FB2调用的FB3将在FB2的程序本体内展开。

■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型FB。
希望减少程序容量的情况下，应选择子程序型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)。

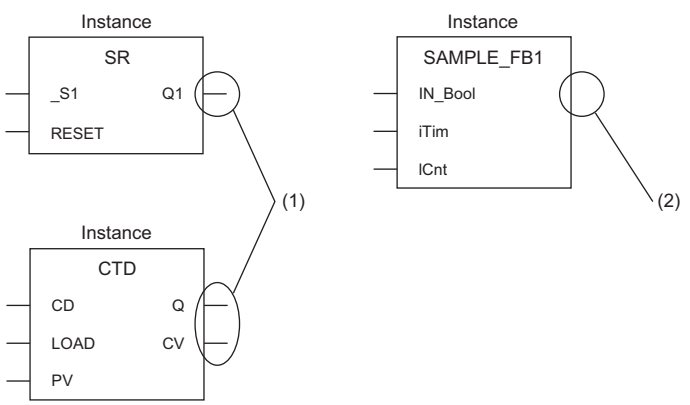


子程序型FB、宏型FB、函数全部合计可进行32次嵌套。

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。



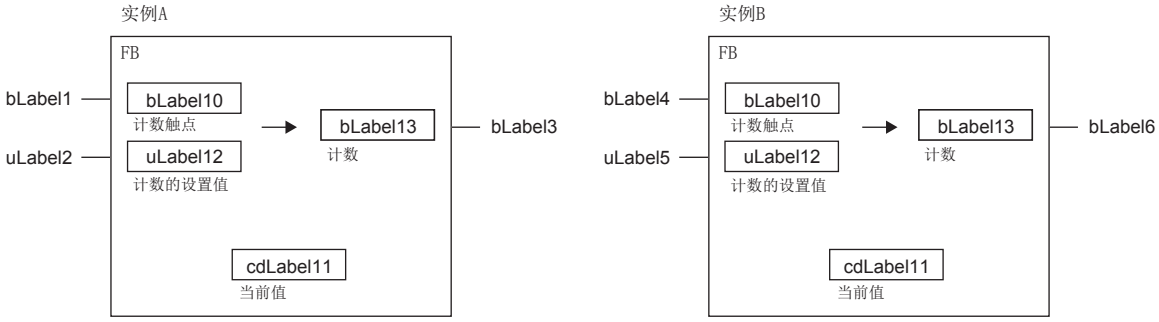
(1) 有输出
(2) 无输出

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

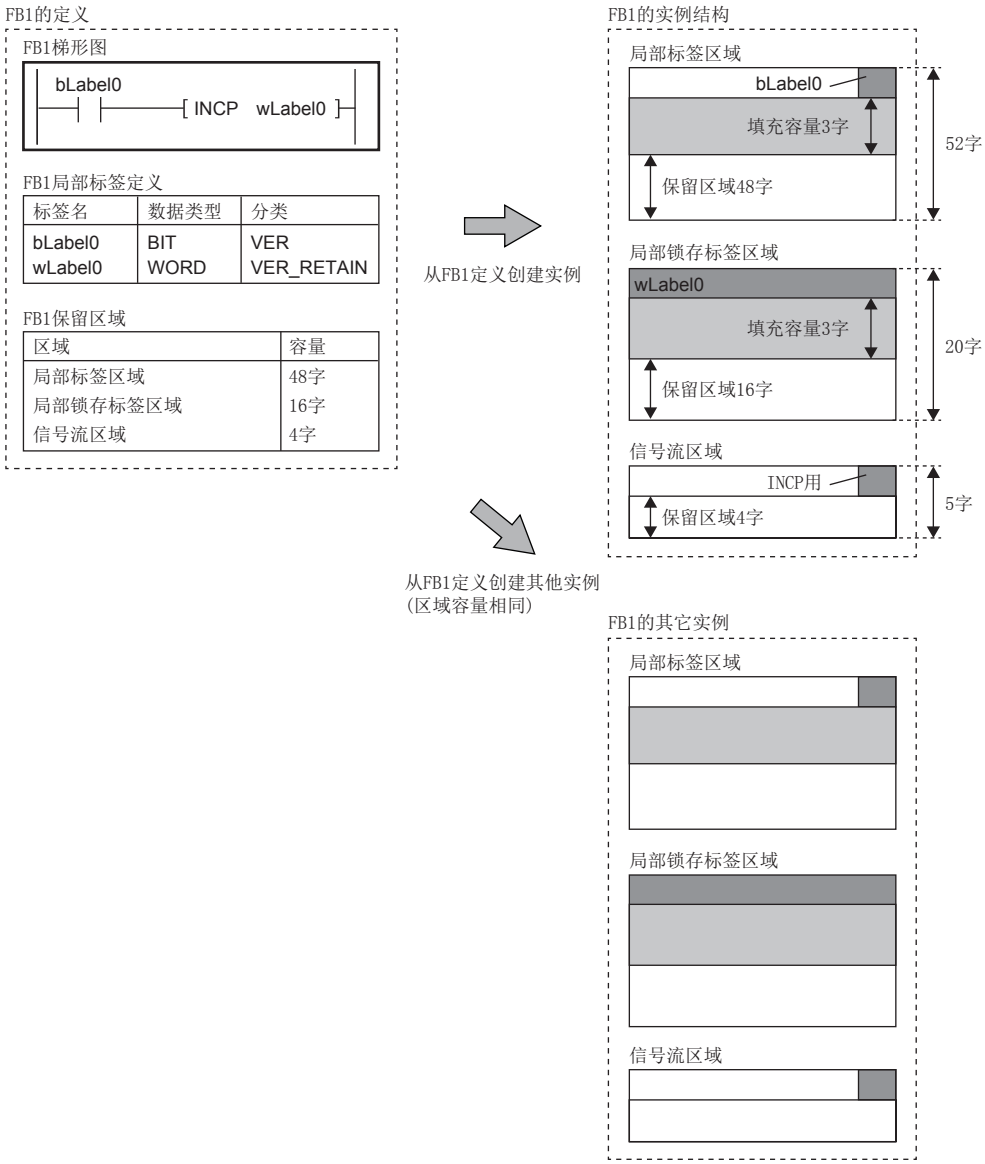
FB的实例是指，在FB定义的基础上分配的标签。可以从一个FB定义创建多个实例。

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	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)。

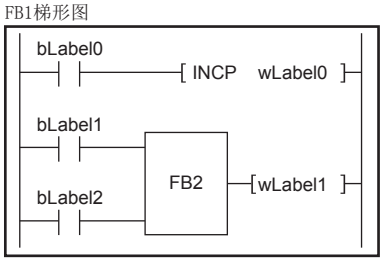
各区域确保保留区域。保留区域是指在转换或RUN中写入等时维持标签的分配不变，用于对参照局部标签或信号流存储器的指令、FB的实例进行添加/更改的区域。无法确保添加/更改对象的数据类型所对应的区域的情况下，需要进行全部转换(重新分配)。

例

对FB进行了嵌套的实例的配置(已变更FB2的保留区域的情况)

FB1的定义

FB1梯形图



FB1局部标签定义

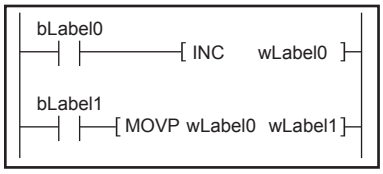
标签名	数据类型	分类
bLabel0	BIT	VER
bLabel1	BIT	VER
bLabel2	BIT	VER
wLabel0	WORD	VER_RETAIN
wLabel1	WORD	VER
FB2_a	FB2	VER

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
信号流区域	4字

FB2的定义

FB2梯形图



FB2局部标签定义

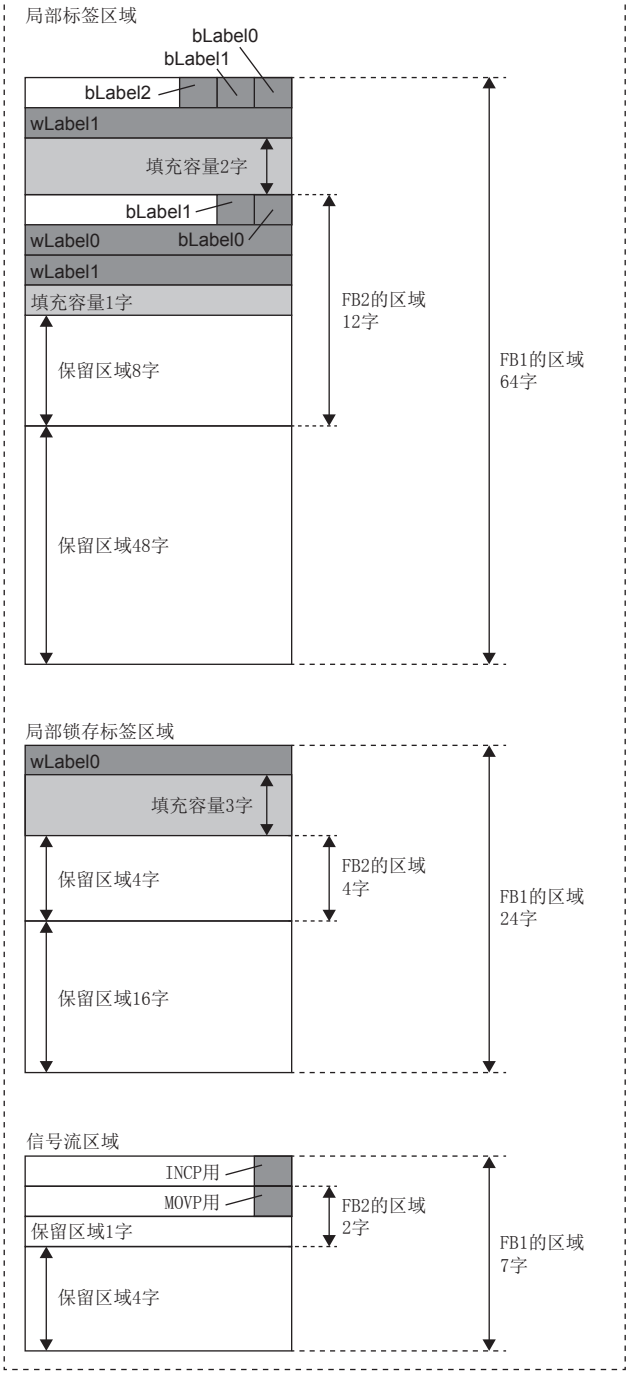
标签名	数据类型	分类
bLabel0	BIT	VER_INPUT
bLabel1	BIT	VER_INPUT
wLabel0	WORD	VER
wLabel1	WORD	VER_OUTPUT

FB2保留区域

区域	容量
局部标签区域	8字
局部锁存标签区域	4字
信号流区域	1字

从FB1定义创建实例

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、信号流区域中。

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、信号流区域为4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)。

希望不全部转换(重新分配)且维持分配不变，而对局部标签或FB实例进行添加/更改的情况下，应预先在保留区域中确保将来用于添加/更改的容量。关于保留区域的设置方法，请参阅下述手册。

GX Works3 操作手册

■创建实例

为了使用FB，需要创建实例。

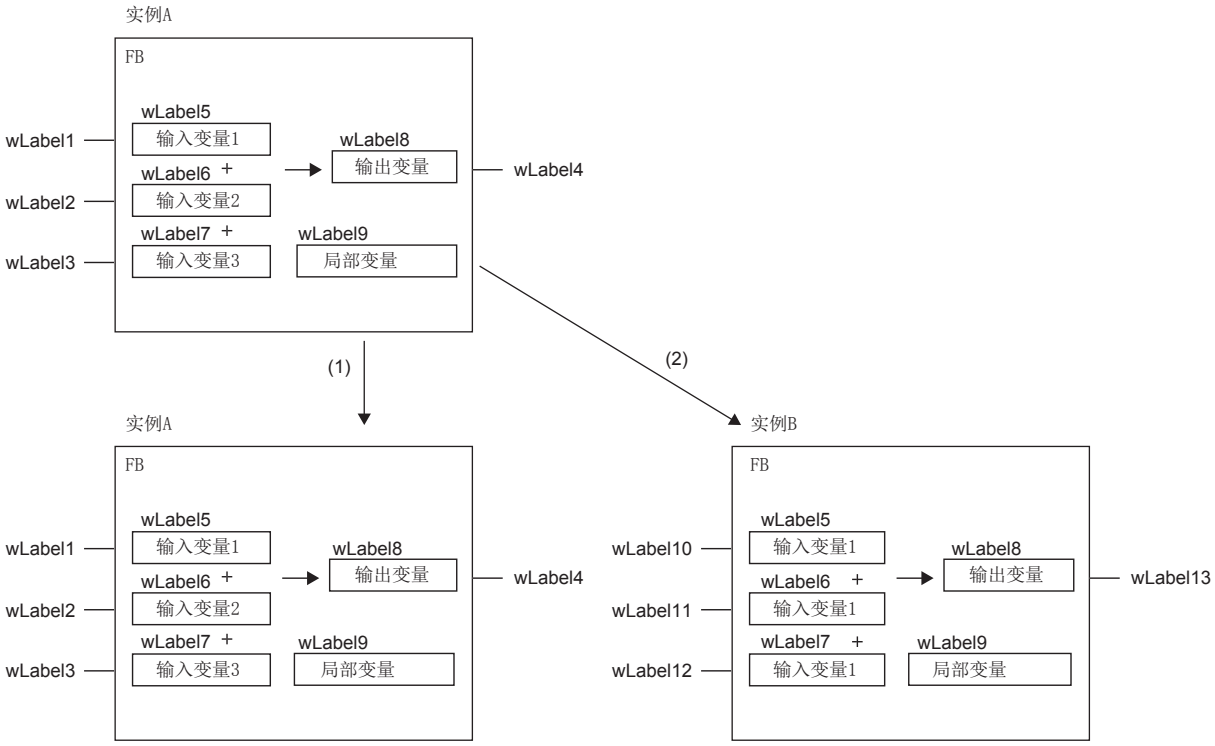
通过创建FB的实例，可以从程序块及其他的FB调用使用。

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	默认为48字。可以以4字单位设置。

• 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	默认为16字。可以以4字单位设置。

• 信号流区域的容量

宏型FB的情况下，与程序的步数同点数。

子程序型FB的情况如下所示。

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
信号流区域容量	FB定义内的指令所使用的信号流区域的总和。
保留区域容量	默认为4字。可以以1字单位设置。

要点

应在保留区域容量中设置RUN中写入时对预计会添加/更改的局部标签或信号流存储器进行参照的指令、FB实例的容量。关于设置方法，请参阅下述手册。

📖GX Works3 操作手册

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)。

对于下述的FB，通过将保留区域设置为比默认值还小的值，可以更有效率地使用CPU模块的存储器。

• 不进行局部标签的添加/更改或程序更改的已调试完成的子程序型FB

• 大量声明实例的子程序型FB

初始值的设置

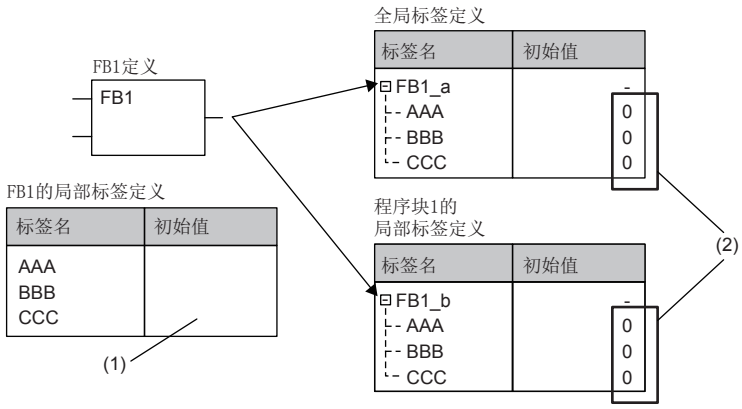
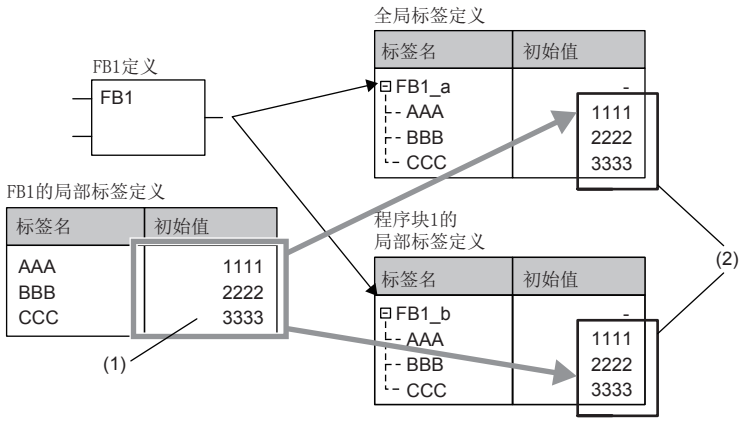
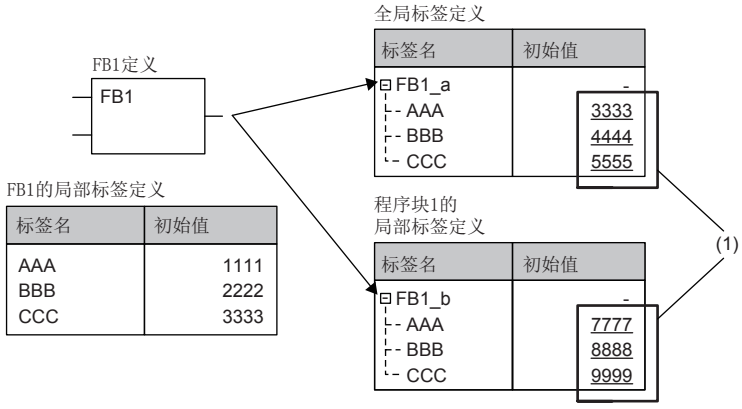
■FB的局部标签的初始值

FB的局部标签可以设置FB定义及各实例的初始值。

可设置初始值的局部标签的分类为VAR、VAR_RETAIN、VAR_INPUT、VAR_OUTPUT、VAR_OUTPUT_RETAIN、VAR_PUBLIC、VAR_PUBLIC_RETAIN。

■实例的初始值

实例的初始值的类型如下所示。

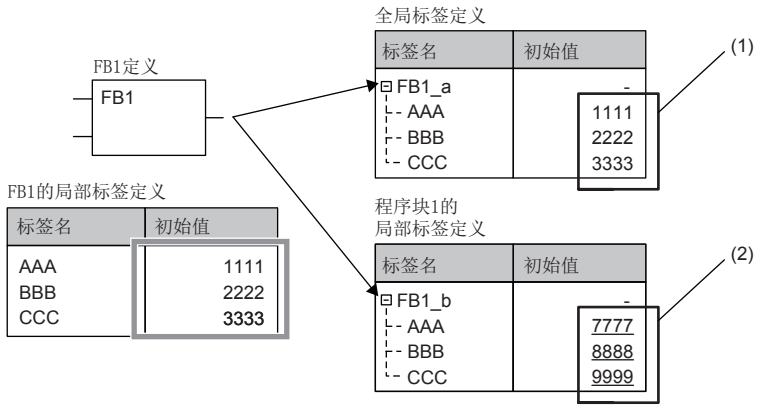
类型	内容
默认初始值	是各数据类型决定的初始值。FB的局部标签中未设置初始值的情况下，默认初始值也适用。  (1) FB1定义的局部标签中未设置初始值。 (2) 默认初始值也适用。
FB定义初始值	是在定义FB的局部标签时设置的初始值。设置了FB定义初始值的情况下，对于所有的实例，相同的定义初始值也适用。  (1) FB1定义的局部标签中已设有初始值。 (2) FB1的所有实例根据相同的定义初始值进行初始化。
实例初始值	是在包含全局标签及程序块的局部标签定义的实例中所设置的初始值。  (1) 可以在FB1定义的各实例中设置初始值。

可以设置FB定义初始值与实例初始值两者的FB的初始值。
设置两个初始值的情况下，适用的初始值的优先顺序如下所述。

优先顺序	类型
高 ↑ ↓ 低	实例初始值
	FB定义初始值
	默认初始值

要点

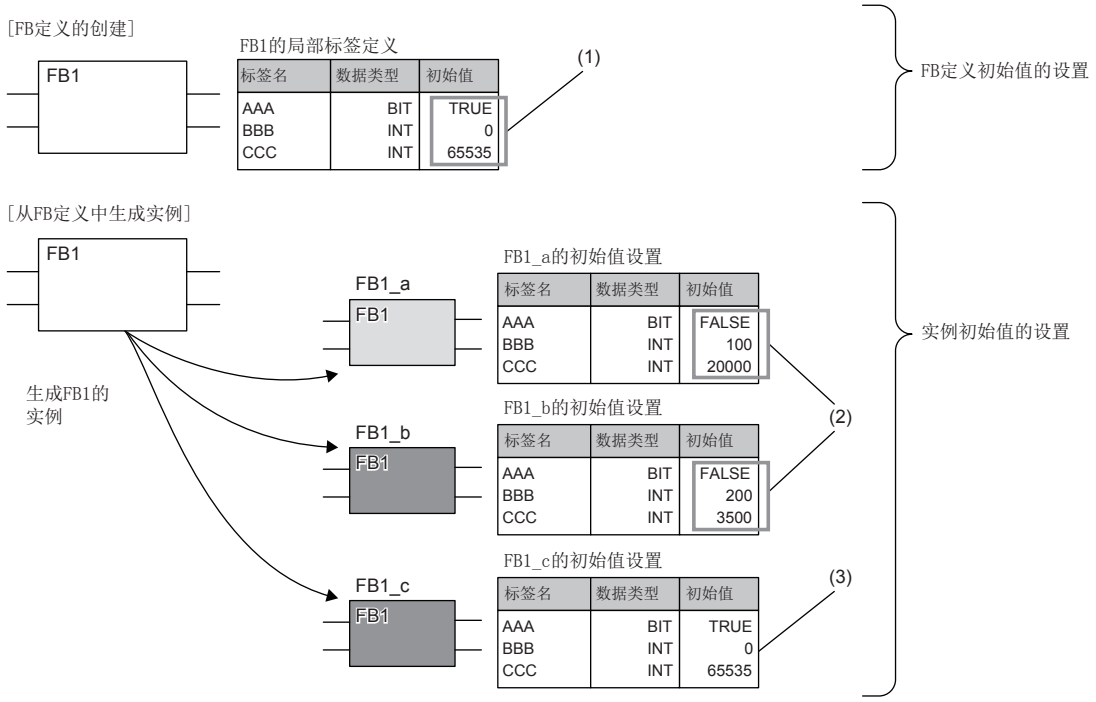
创建两个设置有FB定义初始值的FB的实例，只对其中一个设置了实例初始值的情况下，FB定义初始值适用于未设置实例初始值的实例，实例初始值适用于设置了实例初始值的实例。



(1)未设置实例初始值的情况下，按照定义初始值进行初始化。
(2)设置了实例初始值的情况下，按照实例初始值进行初始化。

■使用示例

FB的初始值的使用示例如下所示。



(1) 在全部实例中设置通用的初始值。
(2) 可以对各实例设置个别的初始值。
(3) 未设置个别初始值的情况下，通用的初始值也适用。

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。

☞15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-R CPU模块用户手册(入门篇)

■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。

- 宏型FB
- 子程序型FB

详细内容，请参阅下述章节。

☞20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。 ☞ 142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。 ☞ 28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-R CPU模块用户手册(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。

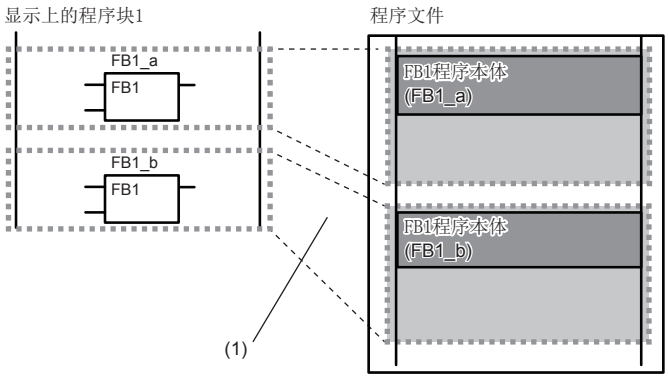
○：可以使用， △：只可在指令中使用(作为表示程序的步的标签时，禁止使用)， ×：禁止使用

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	○
	局部软元件	×
指针	全局指针	△
	局部指针	×

步数(宏型FB)

■调用侧

调用宏型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。

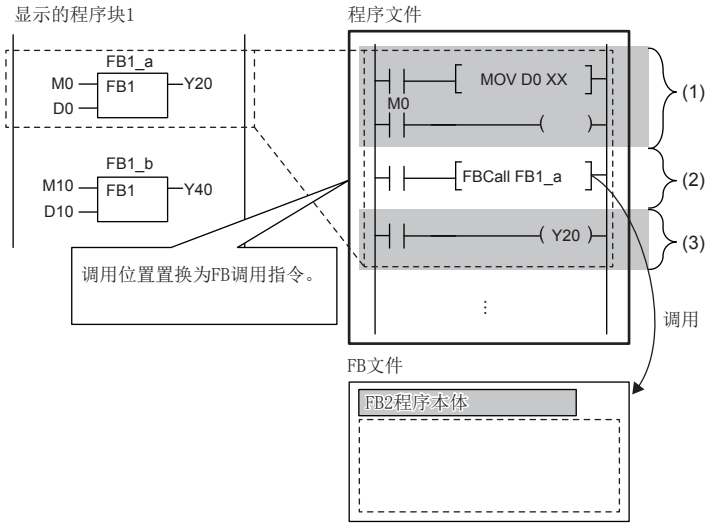
关于各指令的步数，请参阅下述手册。

📖MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)

步数(子程序型FB)

■调用侧

调用子程序型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-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)
	字[无符号]/位串[16位] 双字[无符号]/位串[32位] 字[带符号] 双字[带符号]	LD+MOV LD+DMOV	
	单精度实数	LD+EMOV	
	双精度实数	LD+EDMOV	
	时间	LD+DMOV	
	字符串	LD+$MOV	
	字符串[Unicode]	LD+$MOV_WS	
	数组、结构体	LD+BMOV	

• 程序本体调用

调用FB程序本体需要10步。

• EN/ENO

EN/ENO所需步数如下所示。

项目	步数
EN	3
ENO	2

要点

步数根据下述情况增减。

- 对FB调用的实自变量进行了变址修饰的情况
- 指定软元件地址超过16位的情况
- 位数指定的情况

■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-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)

3.4 注意事项

使用函数的情况

■全局指针/局部指针/指针型的全局标签

局部指针、指针型的全局标签不能使用。

全局指针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/局部指针/指针型的全局标签

局部指针不能使用。

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。

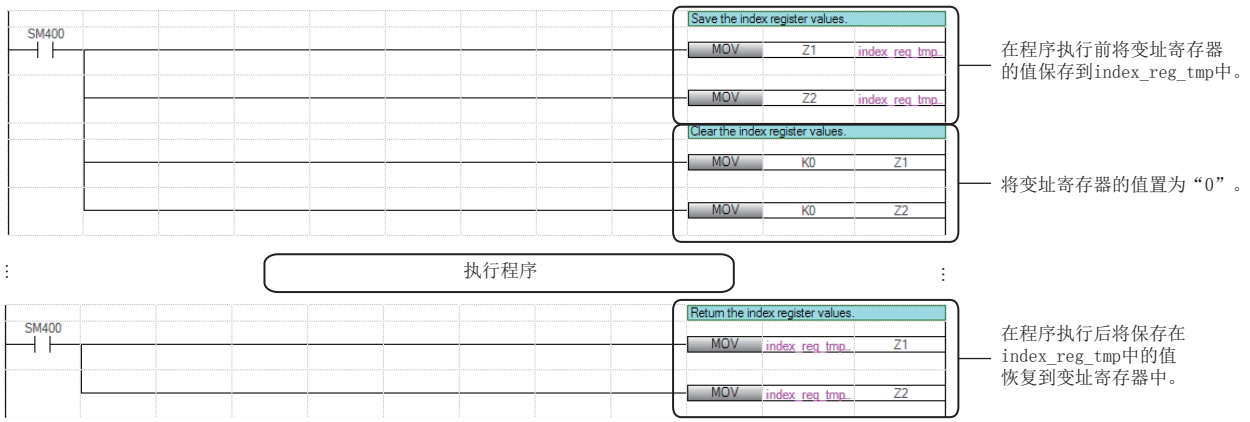
■使用变址寄存器的情况

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。

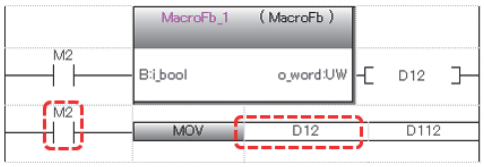
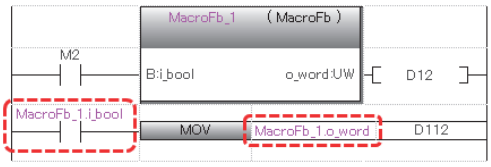
例

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

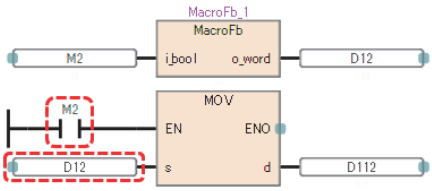
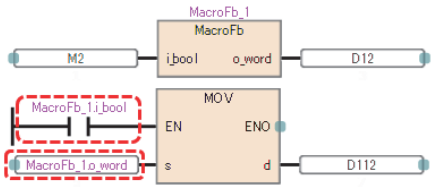


■关于宏型FB的自变量

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

程序语言	用于自变量交接的软元件/标签的情况下	变为意料外的值的情况
梯形图语言		

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

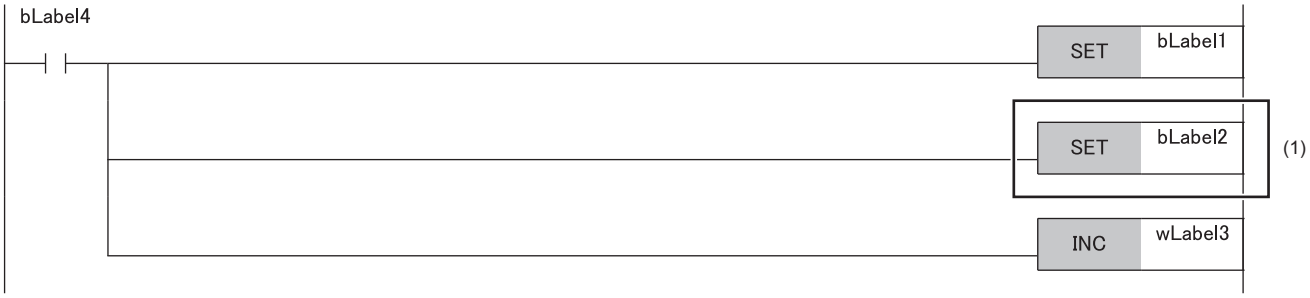
FBD/LD语言		

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

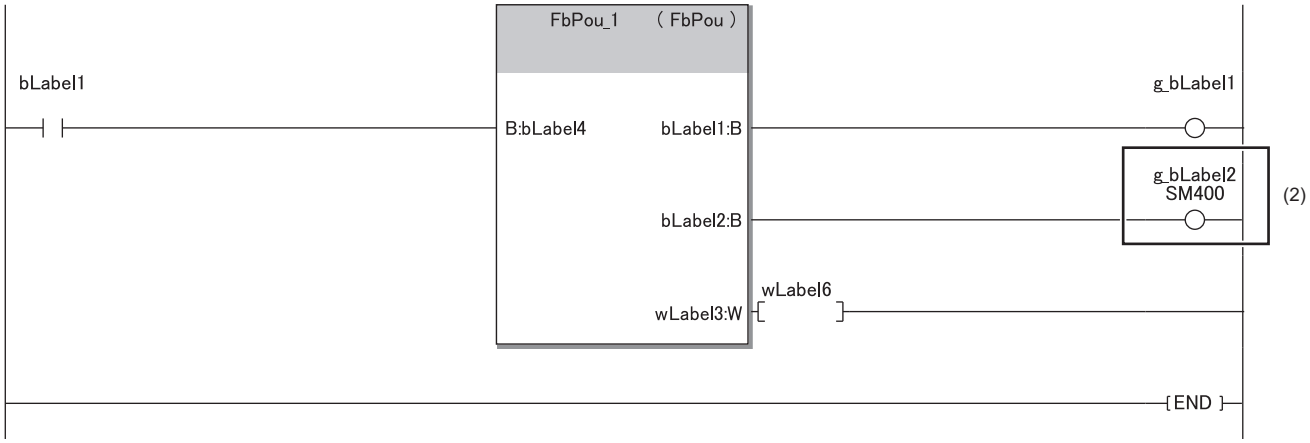
在宏型FB内的VAR_INPUT、VAR_OUTPUT或VAR_IN_OUT中发生转换出错的情况下，出错的原因可能在于FB的调用源的程序块或FB。此时，应确认FB的调用源的程序块或FB的输入或输出。

例

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



(1)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。

■智能功能模块的起始I/O No.的指定

访问智能功能模块的缓冲存储器或输入输出信号的情况下，应使用变址寄存器指定起始输入输出编号。

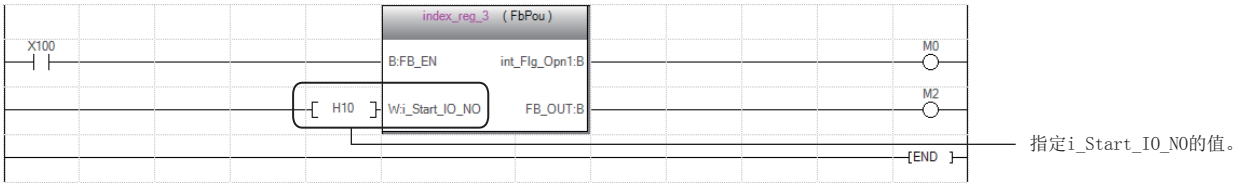
通过将起始输入输出编号作为输入自变量进行接收，则在安装位置不同的多个智能功能模块中，可以不用更改起始输入输出编号而直接使用通用FB。

例

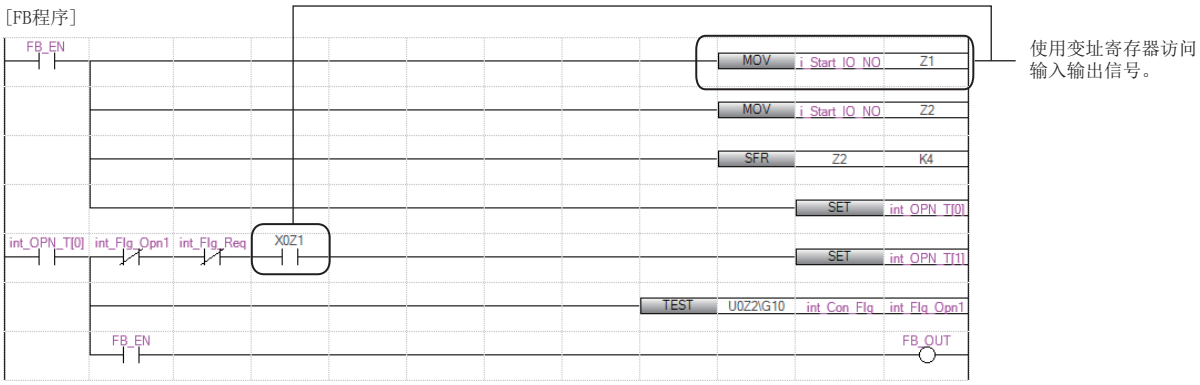
访问智能功能模块的输入输出信号的情况

通过使用变址寄存器，可以访问对象智能功能模块的输入输出信号。

[顺控程序]



[FB程序]

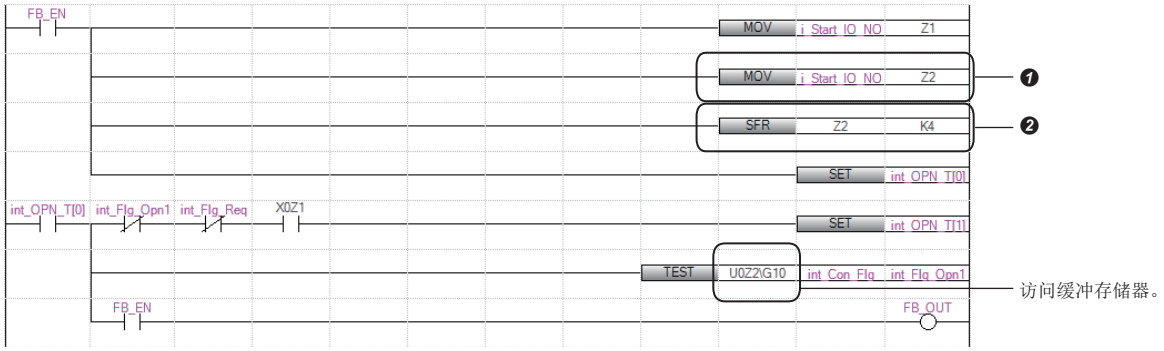


例

访问智能功能模块的缓冲存储器的情况

❶ 将对象智能功能模块的起始输出编号输入到变址寄存器中。

❷ 使用SFR指令，把值向右移四位，或者使用将值除以16所得的商。



■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、SCJ指令、JMP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FCALL(P)指令、EFCALL(P)指令、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。
- 针对执行调用模块FB的程序，请勿使用程序控制用指令(PSTOP(P)指令、POFF(P)指令、PSCAN(P)指令)。

■更改模块FB的对象模块的管理CPU的情况

在工程工具的“I/O分配设置”中，如果将“管理CPU设置”更改为其它机号CPU，将会删除程序上的模块FB。

在更改“管理CPU设置”前，应预先将程序复制到其它机号CPU工程中。

■模块FB的动作参数的更改

模块FB的输入标签及输出标签以外的动作参数(外部变量)的更改可以在标签设置中更改。

- 将模块FB的实例置为局部标签的情况下，可以在“局部标签设置”中更改。

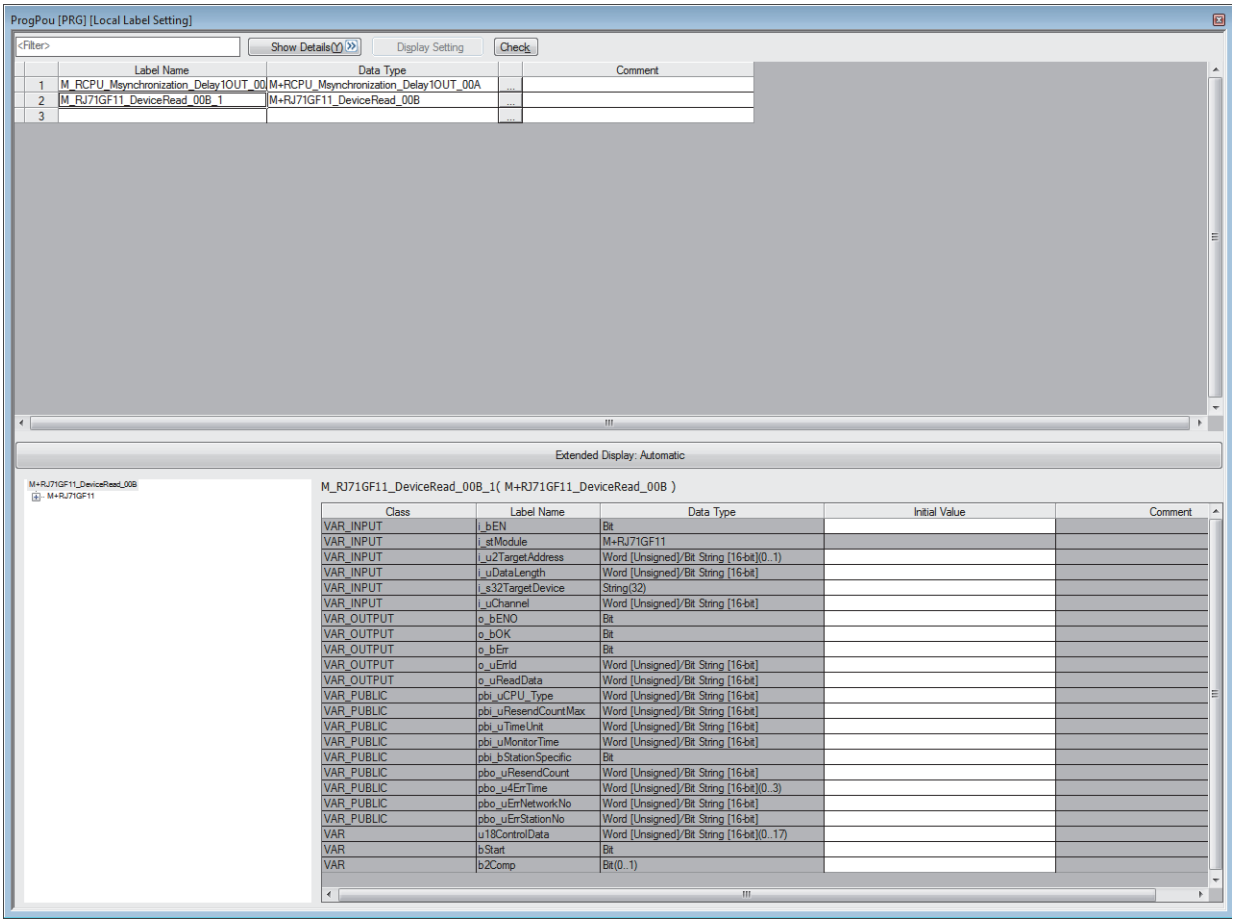
[导航窗口]⇨[程序]⇨执行类型⇨程序文件⇨程序块⇨[局部标签]

- 将模块FB的实例置为全局标签的情况下，可以在“全局标签设置”中更改。

[导航窗口]⇨[标签]⇨[全局标签]

例

设置局部标签的情况



在“初始值”栏中设置动作参数。

3.5 使用安全程序的情况

安全程序中所使用的函数称之为安全函数，FB称之为安全FB。本节中未记载的内容与普通的函数、FB相同。(☞ 14页 函数(FUN)、☞ 19页 功能块(FB))

安全函数(安全FUN)

以下对安全函数有关内容进行说明。

创建程序

■可使用的软元件及标签

在安全函数中可使用的软元件及标签如下所示。

○：可以使用， ×：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	×
	常规/安全共享标签	×
	安全全局标签	×
	安全局部标签	○*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	×
	局部软元件	×
	安全全局软元件	○
	安全局部软元件	×
指针	全局指针	×
	局部指针	×

*1 不能使用下述数据类型。
定时器、累计定时器、计数器、长定时器、长累计定时器、长计数器

步数

■自变量交接

调用安全函数的情况下，在安全函数调用的前后生成安全函数的自变量的交接处理。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。

○：可以使用， ×：禁止使用

自变量的分类	数据类型	使用指令	能否使用
VAR_INPUT	位	LD+OUT	○
	字[无符号]/位串[16位] 双字[无符号]/位串[32位] 字[带符号] 双字[带符号]	LD+MOV LD+DMOV	○
	单精度实数	LD+EMOV	×
	双精度实数	LD+EDMOV	×
	时间	LD+DMOV	×
	字符串	LD+$MOV	×
	字符串[Unicode]	LD+$MOV_WS	×
	数组、结构体	LD+BMOV	○

关于各指令的步数，请参阅下述手册。

MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)

安全功能块(安全FB)

以下对安全FB有关内容进行说明。

实例

■实例的配置

安全FB的实例的构成如下。

○：可以使用， ×：禁止使用

项目	内容	能否使用
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	对FB定义内的指令所使用的信号流进行分配的区域。	○

创建程序

■可使用的软元件/标签

在安全FB中可使用的软元件及标签如下所示。

○：可以使用， ×：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	×
	常规/安全共享标签	○
	安全全局标签	○
	安全局部标签	○
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	×
	局部软元件	×
	安全全局软元件	○
	安全局部软元件	×
指针	全局指针	×
	局部指针	×

步数(子程序型FB)

■自变量交接

调用安全FB的情况下，在安全FB调用的前后生成安全FB的自变量的交接处理。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。

○：可以使用， ×：禁止使用

自变量的分类	数据类型	使用指令	能否使用
VAR_INPUT VAR_IN_OUT	位	LD+OUT	○
	字[无符号]/位串[16位] 双字[无符号]/位串[32位] 字[带符号] 双字[带符号]	LD+MOV LD+DMOV	○
	单精度实数	LD+EMOV	×
	双精度实数	LD+EDMOV	×
	时间	LD+DMOV	×
	字符串	LD+$MOV	×
	字符串[Unicode]	LD+$MOV_WS	×
	数组、结构体	LD+BMOV	○

关于各指令的步数，请参阅下述手册。

MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)

4 标签

标签是指在输入输出数据或内部处理中指定任意字符串的变量。
在编程中使用标签时，无需软元件或缓冲存储器容量即可创建程序。
因此，即使在模块配置不同的系统中，使用了标签的程序也可以简单再利用。
详细内容，请参阅下述手册。
MELSEC iQ-R CPU模块用户手册(应用篇)

备忘录

5 梯形图语言

RnCPU RnENCPU RnPCPU(过程) RnPCPU(冗余) RnPSFCPU(常规) RnPSFCPU(安全) RnSFCPU(常规) RnSFCPU(安全)

是在由触点与线圈构成的梯形图中，表示以串联与并联的组合进行AND/OR逻辑运算，记述顺控程序控制的语言。

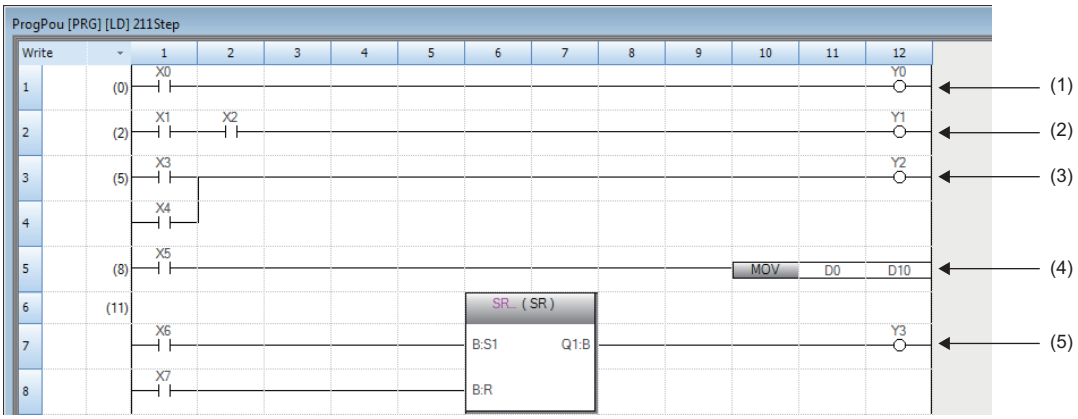
要点

在本章中，对梯形图语言的动作及规格有关内容进行说明。关于创建梯形图程序时的操作方法，请参阅下述手册。

GX Works3 操作手册

5.1 配置

梯形图语言中，可以创建下述梯形图。

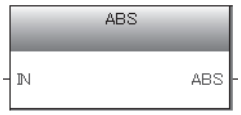
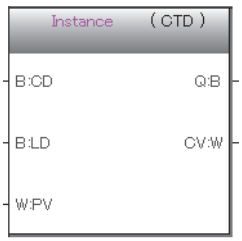


(1) 由触点与线圈组成的梯形图
(2) 由串联组成的梯形图
(3) 由并联组成的梯形图
(4) 使用了指令的梯形图
(5) 使用了通用函数/FB的梯形图

梯形图符号

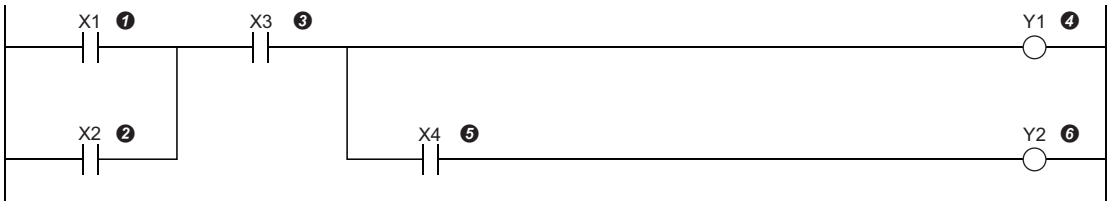
可以在梯形图语言的编程中使用的梯形图符号如下所示。

项目		内容
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。
上升沿脉冲否定		指定软元件或标签OFF时、ON时及下降沿时(ON→OFF)导通。
下降沿脉冲否定		指定软元件或标签OFF、ON及上升沿时(OFF→ON)导通。

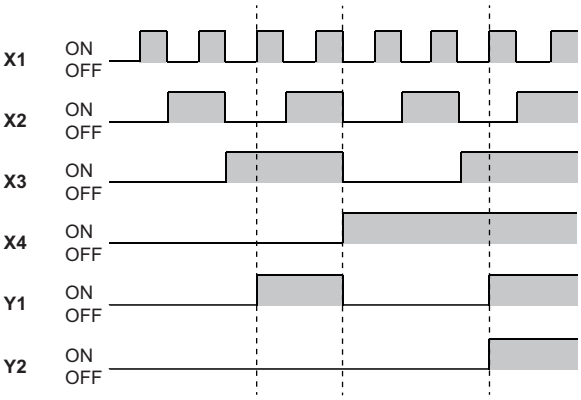
项目		内容
运算结果上升沿脉冲化		运算结果上升沿时(OFF→ON)导通。运算结果在上升沿以外的情况不导通。
运算结果下降沿脉冲化		运算结果下降沿时(ON→OFF)导通。运算结果在下降沿以外的情况不导通。
运算结果取反		将之前的运算结果取反。
线圈		将运算结果输出至指定软元件或标签。
指令		执行[]内指定的指令。
返回		触点数超过了一个梯形图的行中可创建的情况下，创建返回源的符号及返回目标的符号，执行梯形图的返回。
函数		执行函数。 • 函数的创建方法(GX Works3 操作手册) • 通用函数(MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇))
FB		执行FB。 • FB的创建方法(GX Works3 操作手册) • 通用FB(MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)) • 模块FB(所使用的模块的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变为ON的时机如下所示。

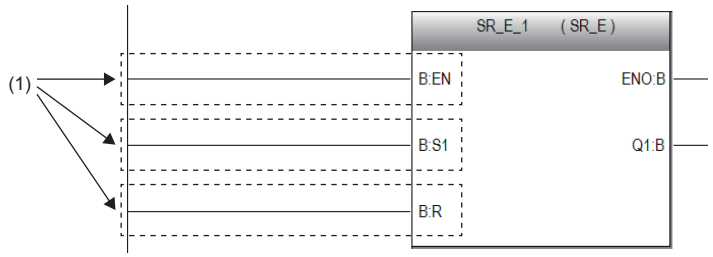


以梯形图语言使用FB时的注意事项

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。

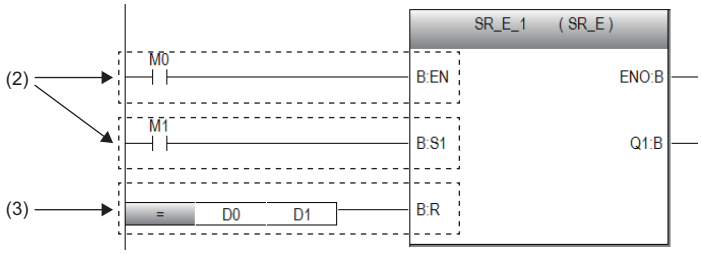
(1)：ON/OFF的状态不发生变化。



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

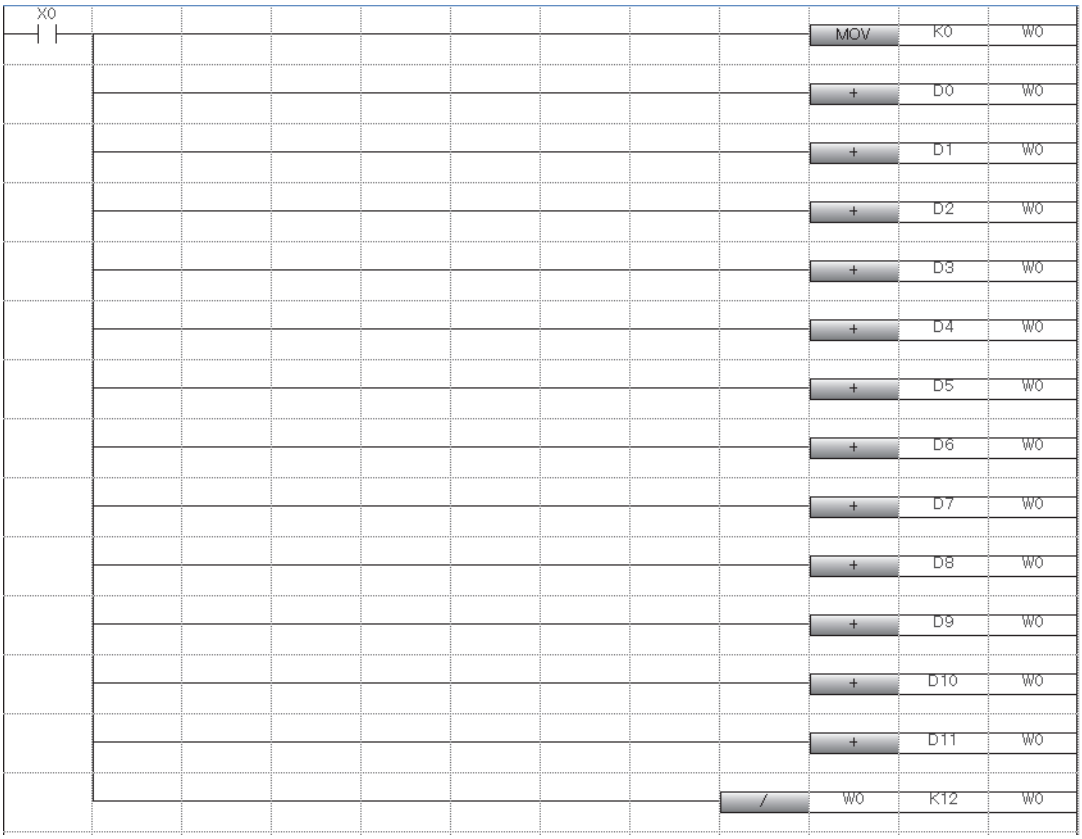


5

5.2 内嵌ST

内嵌ST是指在梯形图编辑器内创建在与线圈相当的指令单元格中显示ST程序的内嵌ST盒子，进行编辑/监视的功能。
由此，可以轻松地在梯形图程序内创建数值运算及字符串处理。

• 不使用内嵌ST情况下的程序



• 使用了内嵌ST情况下的程序

限制事项

• 在安全程序中，不能使用内嵌ST。
• 在SFC程序的Zoom编辑器内不能使用内嵌ST。

规格

关于内嵌ST中记述的ST程序的规格，请参阅ST语言的规格。
☞ 47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行中无法使用函数/FB与内嵌ST盒两者。
- 如果在触点相应的指令位置创建内嵌ST盒，在线圈相应的指令位置也创建内嵌ST盒。
- 内嵌ST内最多可输入的字符数为2048个字符。(换行作为2个字符进行计数。)
- 如果在内嵌ST内使用“RETURN语句”，不是结束程序块的处理，而是结束内嵌ST盒内的处理。
- 在函数的程序中使用内嵌ST的情况下，无法从内嵌ST内调用FB。
- 在内嵌ST中，转换时使用CJ指令控制程序的动作。内嵌ST的触点为OFF的情况下，内嵌ST内的处理将不通过CJ指令执行。因此，即使通过内嵌ST内的代入语句变为ON的软元件不执行内嵌ST，也将保持输出状态。关于CJ指令的详细内容，请参阅下述手册。

MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)

- 无法在内嵌ST内使用OUT、OUTH及参照信号流存储器上一次执行状态的指令。

MELSEC iQ-R CPU模块用户手册(应用篇)

5.3 声明/注解

在梯形图中，可以显示声明及注解。

声明

通过使用声明，可以对梯形图块添加注释。通过进行添加，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图上显示。

■行间声明

对整个梯形图块添加注释。

■P声明

对指针软元件添加注释。

■I声明

对中断指针软元件添加注释。

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/注解的类型

声明与注解的类型有“全体”与“外围”两种。

类型	种类	内容
全体	• 行间声明 • P声明 • I声明 • 注解	说明文的字符串，在转换时置入到程序内。 一行使用(2+字符数)步。
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。 一行使用一步。 在输入的文本前自动添加*印记。

6 ST语言

RnCPU RnENCPU RnPCPU(过程) RnPCPU(冗余) RnPSFCPU(常规) RnPSFCPU(安全) RnSFCPU(常规) RnSFCPU(安全)

ST语言是规定逻辑记述方法的国际标准IEC61131-3所定义的语言。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。适用于对难以梯形图语言表现的复杂处理进行编程的情况下。

要点

在本章中，对ST语言的动作及规格有关内容进行说明。关于创建ST程序时的操作方法，请参阅下述手册。

GX Works3 操作手册

ST语言支持控制语法、运算式、功能块(FB)、函数(FUN)，可以按以下方式记述。

例

条件语句的选择分支、重复语句等的控制语法

```
(*以线A～C进行控制*)
CASE  线  OF
    1:
    开始开关 := TRUE;  (*传送带运行*)
    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  (*传送带运行 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);
```

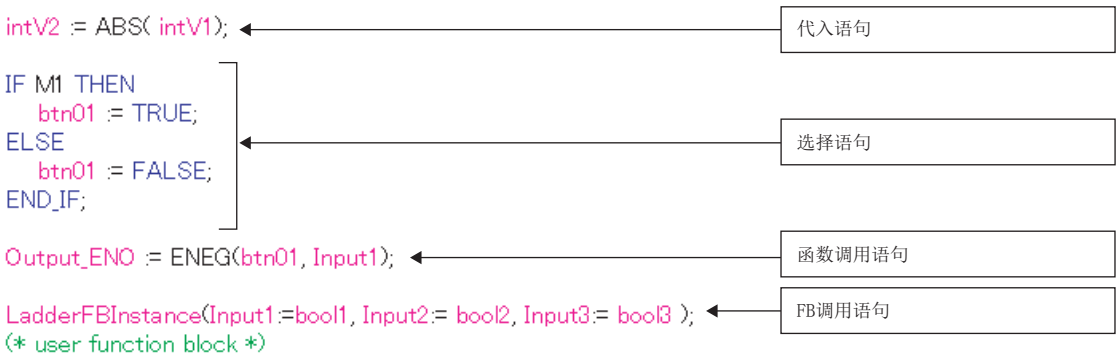
例

汉字等全角字符的使用

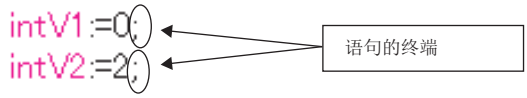
```
//油箱限制器ON时关闭阀，OFF时打开阀
IF  油箱限制器 = TRUE  THEN
     阀 := FALSE ;  (* 由于限制器变为ON，关闭阀 *)
    ELSE
     阀 := TRUE ;  /* 由于限制器变为OFF，打开阀 */
END_IF;
```

6.1 配置

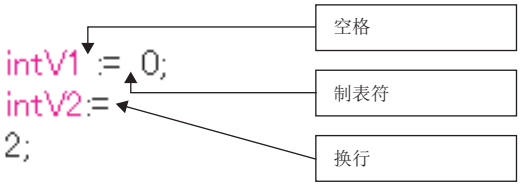
ST语言中的编程由运算符与语法组成。



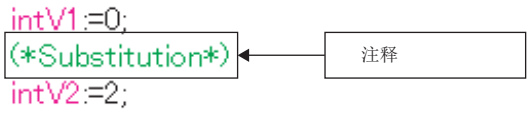
句子的终端必须添加“;”(分号)。



空格、制表符、换行可以插入到运算符及数据中。



可以在程序中插入注释。在注释语句的前后记述“(*注释语句*)”。



程序的结构要素

ST程序由以下要素构成。

项目		示例	参照目标
分隔符		; 、(,)	☞ 49页 分隔符
运算符		+、-、<、>、=	☞ 49页 运算符
保留字	语法	IF、CASE、WHILE、RETURN	☞ 50页 语法
	软元件	X0、Y10、M100、ZR0	📖MELSEC iQ-R CPU模块用户手册(应用篇)
	数据类型	BOOL、DWORD	📖MELSEC iQ-R CPU模块用户手册(应用篇)
	通用函数	ADD、REAL_TO_STRING_E	📖MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)
常数		123、"abc"	☞ 59页 常数
标签		Switch_A	☞ 60页 标签与软元件
注释		(*置为ON*)	☞ 62页 注释
其它符号		半角空格、换行代码、TAB代码	—

- 分隔符、运算符号、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

分隔符

在ST语言中为了明确表示程序的结构，有下述的分隔符。

符号	内容
()	圆括弧式
[]	数组要素编号的指定
.(句号)	结构体、FB构件的指定
,(逗号)	自变量的分隔
:(冒号)	软元件型指定符、CASE语句的分隔
;(分号)	语句的终端
″(双引号)	Unicode字符串的标记
’(单引号)	字符串(ASCII、移位JIS)的标记
..(两个句号)	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*1	ANY_NUM
<、>、<=、>=、=、<>	ANY_ELEMENTARY*2	位
MOD	ANY_INT	ANY_INT
AND、&、XOR、OR、NOT	ANY_BIT	ANY_BIT
**	ANY_REAL(底) ANY_NUM(指数)*1	ANY_REAL

*1　不可以指定“WORD#”或“DWORD#”的常数标记。(☞ 59页 常数的标记方法)

*2　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( )	函数调用方式	CONCAT(’AB’,’CD’)	2
-	符号取反	-10	3
NOT	位型补数	NOT TRUE	
**	次方	3.0**4	4
*	乘法	10*20	5
/	除法	20/10	
MOD	求余数	17 MOD 10	
+	加法	1.4+2.5	6
-	减法	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	异或	TRUE XOR FALSE	10
OR	逻辑或	TRUE OR FALSE	11

- 一个公式中有多个相同优先顺序的运算符的情况下，从左侧的运算符开始运算。
- 一个公式中，最多可记述的运算符使用个数为1024个。

语法

可在ST程序中使用的语法如下所示。

项目	内容	参照页
代入语句	代入语句	50页 代入语句
子程序控制语句	FB调用语句、函数调用语句	52页 子程序控制语句
	RETURN语句	
选择语句	IF语句(IF THEN、IF ELSE、IF ELSIF)	53页 选择语句
	CASE语句	
重复语句	FOR语句	53页 重复语句
	WHILE语句	
	REPEAT语句	
	EXIT语句	

应用半角字符记述语法。

代入语句

书写格式	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。 右边公式的结果需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，左边与右边的数据类型与要素数需要相同。此外，请勿指定要素。

例

```
intAry1:=intAry2;
```

结构体标签的情况下，左边与右边的数据类型(结构体的数据类型)需要相同。

例

```
dutVar1:=dutVar2;
```

右边为函数调用方式的情况下，函数的返回值将代入左边。将返回值代入标签的示例如下所示。

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

■数据类型的自动转换

在ST语言中记述不同数据类型的代入或算术运算公式时，可以自动转换数据类型。

例

自动转换示例

```
dintLabel1 := intLabel1 ;
//代入语句： 将INT型(intLabel1)的值自动转换为DINT型，代入至左边的DINT型(dintLabel1)
dintLabel1 := dintLabel2 + intLabel1 ;
//算术运算公式： INT型(intLabel1)的值自动转换为DINT型，执行DINT型的加法运算
DMOV(TRUE, wordLabel1, dwordLabel1);
//指令、函数、FB调用语句： 将WORD型输入自变量(wordLabel1)的值自动转换为DWORD型，执行传送
```

类型转换在代入语句、向FB及函数(包括指令、通用函数、通用FB)交接输入自变量(VAR_INPUT部)、算术运算公式中进行。

从数据类型容量小的开始往容量大的顺序进行，以确保类型转换时不丢失数据。类型转换以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]	转换后变为双字[带符号]的情况下，自动转换为符号扩展值。 单精度实数或双精度实数的情况下，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]或双字[带符号]的情况下，自动转换为零扩展值。*2 单精度实数或双精度实数的情况下，自动转换为与转换前的整数相同的值。*1
双字[带符号]	转换后为双精度实数的情况下，自动转换为与转换前的整数相同的值。
双字[无符号]/位串[32位]	
单精度实数	转换后为双精度实数的情况下，自动转换为相同的值。

*1　数据类型若将16位数据(字[带符号]或字[无符号]/位串[16位])过渡为ANY_REAL的输入自变量，则自动转换为单精度实数。

*2　数据类型若将字[无符号]/位串[16位]数据过渡为ANY32的输入自变量下，则自动转换为双字[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，下述情况也无法进行类型转换，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

代入算术运算的结果时的注意事项，请参阅下述章节。

☞54页 代入算术运算式结果的情况

使用软元件时的注意事项，请参阅下述章节。

☞61页 使用软元件时的数据类型自动转换

子程序控制语句

■FB调用语句

书写格式	内容
实例名(输入变量1:=变量1,...输出变量1=>变量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所使用的符号与可分配表达式如下所示。

类型	内容	使用符号	可分配表达式
EN，VAR_INPUT	输入变量	:=	所有的表达式
ENO，VAR_OUTPUT， VAR_OUTPUT_RETAIN	输出变量	=>	只有变量
VAR_IN_OUT	输入输出变量	:=	只有变量

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

FB	FB定义	记述示例
具有1个输入变量、1个输出变量的FB的情况	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;
具有3个输入变量、2个输出变量的FB的情况	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

无返回值的函数或参数中包含VAR_OUTPUT变量的函数，通过在函数调用方式的后方加上“;”(分号)，可作为语句执行。

书写格式	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“,”(逗号)隔开。

■RETURN语句

语法	书写格式	内容	记述示例
■RETURN	RETURN;	在中途结束程序、FB、函数时使用。 如果在程序中使用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RETURN语句，将从FB返回。 如果在函数中使用RETURN语句，将从函数返回。 在系统中，一个RETURN语句使用1点指针型标签。	IF bool1 THEN RETURN; END_IF;

选择语句

语法	书写格式	内容	记述示例
■IF THEN	IF<布尔表达式>THEN 　<语句…>; END_IF;	布尔表达式(条件表达式)为真(TRUE)时，执行语句。布尔表达式为假(FALSE)时，不执行语句。 对于布尔表达式，只要单一的位型变量的状态或包含多个变量的复杂表达式的布尔运算结果为返回真(TRUE)或假(FALSE)的表达式，则可以在任意表达式中使用。	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(条件表达式)为真(TRUE)时，执行语句1。
		布尔表达式的值为假(FALSE)时，执行语句2。
			IF　bool1　THEN
			　intV3 := intV3 + 1;
			　ELSE
			　intV4 := intV4 + 1;
			END_IF;

■IF...ELSEIF	IF<布尔表达式1>THEN
	　<语句1…>;
	　ELSEIF<布尔表达式2>THEN
	　<语句2…>;
	　ELSEIF<布尔表达式3>THEN
	　<语句3…>;
	END_IF;
		布尔表达式(条件表达式)1为真(TRUE)时，执行语句1。布尔表达式1的值为假(FALSE)且布尔表达式2的值为真(TRUE)时，执行语句2。
		布尔表达式1、2的值都为假(FALSE)且布尔表达式3的值为真(TRUE)时，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<整数表达式>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;
		具有与整数表达式(条件表达式)的值一致的整数的选择值的语句被执行后，若无一致，则执行ELSE语句的下一语句。
		例如，根据单一的整数值及复杂表达式的结果的整数值，可在执行选择语句时使用CASE语句。
			CASE　intV1　OF
			　1 :
			　bool1 := TRUE;
			　2 :
			　bool2 := TRUE;
			　ELSE
			　intV1 := intV1 + 1;
			END_CASE;

重复语句

语法	书写格式	内容	记述示例
■FOR...DO	FOR<重复变量初始化> 　TO<最终值> 　BY<增加表达式>DO 　<语句…>; END_FOR;	对作为反复变量使用的数据初始化。 根据增加公式对初始化后的反复变量进行加法或减法运算，重复执行自DO到END_FOR间的一个及其以上的语句，直至超出最终值为止。 FOR...DO语句结束后的反复变量，保持在结束时点的值。	FOR　intV1 := 0 　TO 30 　BY 1 DO 　intV3 := intV1 + 1; END_FOR;
■WHILE...DO	WHILE<布尔表达式>DO 　<语句…>; END_WHILE;	布尔表达式(条件表达式)为真(TRUE)时，执行1个及其以上的语句。 布尔表达式在执行语句之前被判定，布尔表达式为假(FALSE)时，不执行DO...END_WHILE中的语句。WHILE语句中的<布尔表达式>，无论返回的结果为真或假，只要有返回即可，因此IF语句中的<布尔表达式>中可指定的表达式全可以使用。	WHILE　intV1 = 30　DO 　intV1 := intV1 + 1; END_WHILE;

■REPEAT...UNTIL	REPEAT
	　<语句…>;
	　UNTIL<布尔表达式>
	END_REPEAT;
		布尔表达式(条件表达式)为假(FALSE)时，执行1个及其以上语句。
		布尔表达式在执行语句后被判定，值为真(TRUE)时则不执行REPEAT...UNTIL内的语句。REPEAT语句中的<布尔表达式>，无论返回的结果为真或假，只要有返回即可，因此IF语句中的<布尔表达式>中的可指定的表达式全可以使用。
			REPEAT
			　intV1 := intV1 + 1;
			　UNTIL　intV1 = 30
			END_REPEAT;

■EXIT	EXIT;	只可在重复语句中使用的语法，于中途结束重复语句。 如果在执行重复环路过程中达到EXIT语句，则不执行EXIT语句之后的重复环路处理。从终止重复语句后的下一行开始继续执行程序。	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

- 字符串的代入的最大字符串长为255字符。代入的字符串超过最大字符串长时，将变为转换出错。
- 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。应在代入式的左边使用实例的输入变量、输出变量、外部变量。

■使用步继电器(S)及SFC块软元件(BL)的情况

将步继电器(S)及SFC块软元件(BL)在代入式的右边、函数及FB的输入自变量中使用的情况下，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

MO := SO;	IF SO THEN
	MO := TRUE;
	ELSE
	MO := FALSE;
	END_IF;

此外，在程序中使用步继电器(S)或带块指定步继电器(BL□\S□)的位数指定的情况下，应指定正确的数据容量。由于步继电器(S)或带块指定步继电器(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);

■代入算术运算式结果的情况

将算术运算表达式的结果代入到数据容量较大的数据类型的变量中的情况下，应预先把算术运算表达式的变量转换为左边的数据类型之后再进行运算。

例

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

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

算术运算表达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，varInt1*10的运算结果超出了INT型的范围(-32768～32767)的情况下，会将上溢或下溢的运算结果代入到varDint1中。

在这种情况下，应预先将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
varDint1 := varDint2 * 10; //以DINT型进行乘法运算，代入运算结果
```

■在算术运算式中使用符号取反运算符的情况

对数据类型的最小值，使用符号取反运算符(-)时，将变为相同的值。

例如，INT型最小值的情况下，变为-(-32768)=-32768。

因此，在数据类型的自动转换的对象变量中使用符号取反运算符时，可能无法变为希望的结果。

例

varInt1(INT型)的值为-32768、varDint1(DINT型)的值为0的情况

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

此时，(-varInt1)的值将保持-32768不变，varDint2中将被代入-32768。

在算术运算式中使用符号取反运算符的情况下，应预先在算术运算前进行数据类型的自动转换或创建不使用符号取反运算符的程序。

例

算术运算之前进行数据类型自动转换的情况

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

例

不使用符号取反运算符的情况

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

■从单精度实数转换成双精度实数数据型的情况

从单精度实数转换成双精度实数型(REAL_TO_LREAL)时，转换结果可能会发生误差。

因此，在进行数据类型自动转换、在代入语句的右边或算术运算式的操作数中使用了实数型函数(SIN等)的返回值的情况下，可能无法变为希望的结果。

例

发生误差的情况

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

上述情况下，ABS(varReal1)的返回值变为单精度实数，由于将该值型转换为双精度实数后代入varLReal1中，因此发生误差。

这种情况下，应使用与代入目标相同的数据类型(双精度实数)创建执行函数的程序。

例

不发生误差的情况

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

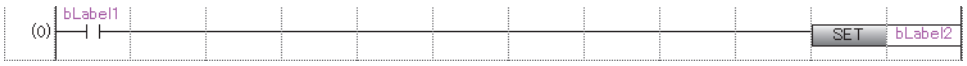
■使用位型标签时

选择语句或重复语句中布尔表达式(条件表达式)一旦成立，将<语句>内的位型标签置为ON状态时，则该位型标签将变为常时ON。

例

常时ON的程序

ST程序	ST程序同等处理的梯形图程序

IF bLabel1 THEN
bLabel2 := TRUE;
END_IF;
	

为避免常时ON，应按下述方式对将位型标签置为OFF的程序进行添加。

例

避免常时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，变为与常时ON程序同样的状态。

■使用定时器FB、计数器FB时

选择语句中的布尔表达式(条件表达式)，与定时器FB、计数器FB的执行条件不同。

例

定时器FB的情况

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且bLabel2=ON时，开始计数。*)
(* bLabel1=ON且bLabel2=OFF时，清除计数。*)
(* bLabel1=OFF且bLabel2=ON时，停止计数。不清除计数值。*)
(* bLabel1=OFF且bLabel2=OFF时，停止计数。不清除计数值。*)
```

更改后的程序示例

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

例

计数器FB的情况

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且bLabel2=ON/OFF时，将计数+1。*)
(* bLabel1=OFF且bLabel2=ON/OFF时，不进行计数。*)
(* bLabel1=ON/OFF与计数+1不联动。*)
```

更改后的程序示例

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

上述更改前的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。
通过bLabel1条件与bLabel1的AND条件，使定时器、计数器动作的情况下，不使用控制语句，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

- 无法在重复自变量中使用结构体构件及数组要素。
- 应让重复自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循环。
- FOR...DO语法中，执行FOR语法中的<语句…>后进行重复变量的计数处理。执行了超过重复变量的数据类型的最大值或低于最小值的计数处理的情况下，发生无限循环。

■使用上升沿指令、下降沿指令时

- 在IF语句及CASE语句中使用上升沿指令、下降沿指令时的动作如下所示。

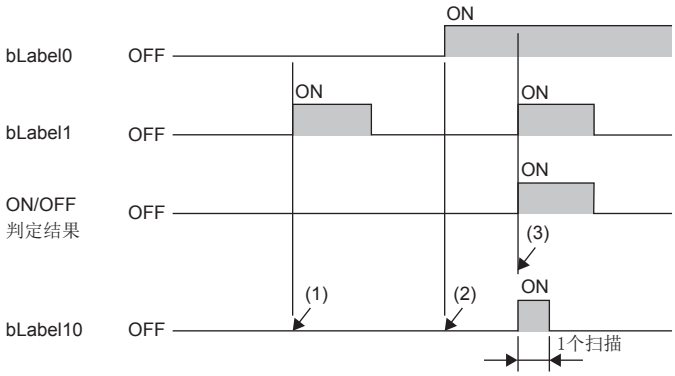
条件			动作结果		
IF语句、CASE语句的条件表达式	指令执行条件(EN)	上一次扫描时指令ON/OFF判定结果	指令ON/OFF判定结果	上升沿指令	下降沿指令
TRUE或CASE一致	TRUE	ON	ON	不执行	不执行
		OFF	ON	执行	不执行
	FALSE	ON	OFF	不执行	执行
		OFF	OFF	不执行	不执行
FALSE或CASE不一致	TRUE	ON	OFF	不执行	不执行*1
		OFF	OFF	不执行	不执行
	FALSE	ON	OFF	不执行	不执行*1
		OFF	OFF	不执行	不执行

*1 虽然为下降沿(ON→OFF)，但是由于IF或CASE语句条件不成立，指令不执行。

例

在IF语句中使用了PLS指令(上升沿执行)的情况

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



(1) bLabel0 = OFF的情况下(IF语句的条件表达式为FALSE)，ON/OFF判定结果变为OFF，不执行PLS指令。(保持bLabel10 = OFF不变)

(2) bLabel0 = ON(IF语句的条件表达式为TRUE)且，bLabel1 = OFF(指令执行条件为OFF)的情况下，ON/OFF判定结果变为OFF，不执行PLS指令。(保持bLabel10 = OFF不变)

(3) bLabel0 = ON(IF语句的条件表达式为TRUE)且bLabel1 = ON(指令执行条件为ON)的情况下，ON/OFF判定结果变为OFF→ON(上升沿条件成立)，执行PLS指令。(bLabel10仅1个扫描ON)

• 以重复语句(FOR语句、WHILE语句或REPEAT语句)，执行上升沿指令或下降沿指令的情况下，使用变址继电器(V)以及变址修饰。此时，对每个使用了变址继电器(V)指令，在系统中使用1点变址继电器(V)。因此，在重复语句中加入使用的点数，确保正在使用的指令数的变址继电器(V)。

例

在FOR语句中使用了上升沿指令及下降沿指令的情况

```
在1个地方使用变址继电器(V)的示例
(最多可使用的变址继电器(V)合计为11点(INC指令中V0～V10)。)
FOR  Z0 := 0  TO  9  BY  1  DO
   INC(EGP(M100Z0 , V0Z0) , D100Z0);
END_FOR;
在2个地方使用变址继电器(V)的示例
(最多可使用的变址继电器(V)合计为22点(INC指令中V0～V10、DEC指令中V11～V21)。)
FOR  Z0 := 0  TO  9  BY  1  DO
   INC(EGP(M100Z0 , V0Z0) , D100Z0);
   DEC(EGF(M200Z0 , V11Z0) , D200Z0);
END_FOR;
```

■使用主控制指令时

主控制OFF时的动作如下所示。

• 选择语句(IF语句或CASE语句)或重复语句(FOR语句、WHILE语句或REPEAT语句)中的语句变为无处理。

• 选择语句或重复语句之外的情况下，代入语句变为无处理、代入语句以外的语句变为非执行处理。

例

选择语句(IF语句)中的语句

```
MC(M0 , N1 , M1);  //主控制OFF
IF  M2  THEN
   M3 := M4;  //主控制OFF时为无处理，因此M3保持之前扫描时的值
END_IF;
M20 := MCR(M0, N1);
```

例

选择语句或重复语句之外的语句(位代入语句的情况)

```
MC(M0 , N1 , M1);  //主控制OFF
M3 := M4;  //主控制OFF时为无处理，因此M3保持之前扫描时的值
M20 := MCR(M0, N1);
```

例

选择语句或重复语句之外的语句(OUT指令的情况)

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

常数

常数的标记方法

在ST程序中，除了可用普通的常数标记外，还可用下述常数标记。

可对应的数据类型	类型	标记方法*1	标记示例
位	引导	在使用的引导值前添加“BOOL#”。	BOOL#1、BOOL#0 BOOL#TRUE、BOOL#FALSE
字[无符号]/位串[16位]	2进制数	在2进制数前添加“UINT#2#”或“WORD#2#”。*2	UINT#2#10101010 WORD#2#10101010
	8进制数	在8进制数前添加“UINT#8#”或“WORD#8#”。*2	UINT#8#3370 WORD#8#3370
	10进制数	在10进制数前添加“UINT#”或“WORD#”。*2	UINT#123 WORD#123
	16进制数	在16进制数前添加“UINT#16#”或“WORD#16#”。*2	UINT#16#FF WORD#16#FF
双字[无符号]/位串[32位]	2进制数	在2进制数前添加“UDINT#2#”或“DWORD#2#”。*2	UDINT#2#1100110011001100 DWORD#2#1100110011001100
	8进制数	在8进制数前添加“UDINT#8#”或“DWORD#8#”。*2	UDINT#8#33703370 DWORD#8#33703370
	10进制数	在10进制数前添加“UDINT#”或“DWORD#”。*2	UDINT#456789 DWORD#456789
	16进制数	在16进制数前添加“UDINT#16#”或“DWORD#16#”。*2	UDINT#16#FFFF DWORD#16#FFFF
字[带符号]	2进制数	在2进制数前添加“INT#2#”。	INT#2#01010101
	8进制数	在8进制数前添加“INT#8#”。	INT#8#3370
	10进制数	在10进制数前添加“INT#”。	INT#-123
	16进制数	在16进制数前添加“INT#16#”。	INT#16#1F
双字[带符号]	2进制数	在2进制数前添加“DINT#2#”。	DINT#2#0011001100110011
	8进制数	在8进制数前添加“DINT#8#”。	DINT#8#33703370
	10进制数	在10进制数前附上“DINT#”。	DINT#-456789
	16进制数	在16进制数前添加“DINT#16#”。	DINT#16#1F1F
单精度实数	实数	在实数前添加“REAL#”。	REAL#2.34
	实数(指数表现)		REAL#1.0E6
双精度实数	实数	在实数前添加“LREAL#”。	LREAL#-2.34
	实数(指数表现)		LREAL#1.001E16
字符串	STRING	用单引号(')括住字符串(ASCII、移位JIS)。	'ABC'
字符串[Unicode]	WSTRING	用双引号(")括住Unicode字符串。	"ABC"

*1 大小写无区别。此外，不可与使用K、H、E的常数标记同时使用。

*2 ANY_NUM算术运算符的操作数、函数调用语句、FB调用语句、函数调用方式的自变量的情况下，应常数标记为“UINT#”或“UDINT#”。常数标记为“WORD#”或“DWORD#”的情况下，将被认为是位串，转换时会发生出错。

上述以外的常数标记方法，请参阅下述手册。

MELSEC iQ-R CPU模块用户手册(应用篇)

要点

2进制数、8进制数、10进制数、16进制数、实数的标记时，可使用下划线(_)划分数值使程序更易读。例如，双字[无符号]的2进制标记时，可记载如下。

UDINT#2#1100_1100_1100_1100

程序处理时，下划线(_)的数值划分将被忽略。

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/FB的自变量、返回值等中使用。

标签与软元件的详细内容，请参阅下述手册。

MELSEC iQ-R CPU模块用户手册(应用篇)

■带类型指定软元件标记

通过在软元件名中添加软元件型指定符，可将字软元件作为任意数据类型使用。

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 在算术运算式等中只使用软元件的情况下，变为字[带符号]。 但是,在FUN/FB的自变量部分被指定作为无类型指定的软元件的情况下，变为自变量定义的数据类型。	D0	D0中不附加类型指定的情况
: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
:ED	双精度实数	D0:ED	将D0、D1、D2、D3作为双精度实数的值

软元件类型指定可使用的软元件如下所示。

- 数据寄存器(D)
- 链接寄存器(W)
- 链接直接软元件(J□\W□)
- 模块访问软元件(U□\G□)
- CPU缓冲存储器访问软元件(U3E□\G□/U3E□\HG□)
- 文件寄存器(R/ZR)
- 刷新数据寄存器(RD)

位数指定或间接指定的软元件中不能赋予软元件型指定符。

■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，请参阅下述手册。

MELSEC iQ-R CPU模块用户手册(应用篇)
MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)

注意事项

- 在ST程序中无法使用指针型。
- 用当前值使用定时器、计数器、累计定时器软元件时的数据类型为字[无符号]/位串[16位]。用当前值使用长定时器、长计数器、长累计定时器软元件时的数据类型为双字[无符号]/位串[32位]。
- 使用位数指定代入的情况下，应使右边和左边的数据类型一致。

例

```
D0:=K5X0;
```

在上述情况下，K5X0为双字型、D0变为字型，因此程序出错。

- 使用位数指定代入的情况下，右边>左边时，数据将传送至左边的对象点数范围内。

例

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

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

- 以字[无符号]/位串[16位]以外的类型使用计数器(C)、定时器(T)、累计定时器(ST)的当前值(TNn等)，或以双字[无符号]/位串[32位]以外的类型使用长计数器(LC)、长定时器(LT)、长累计定时器(LST)的当前值(LTNn等)的情况下，应使用类型转换函数。

例

```
varInt:=WORD_TO_INT(T0);(*使用类型转换函数*)
```

- 将定时器及计数器的软元件的线圈(TC、STC、LTC、LSTC、CC、LCC)用于代入式的右边及函数、FB的输入自变量的情况下，作为触点(TS、STS、LTS、LSTS、CS、LCS)进行动作。
- 希望将定时器及计数器的线圈用于输入自变量的情况下，应使用定时器型及计数器型的标签。

例

定时器软元件及定时器型标签的情况

```
M1 := TC0; (* 将触点(TS0)的值代入到M1中。 *)
M2 := INV(TC1); (* 将触点(TS1)的取反结果代入到M2中。 *)
M1 := tLabel0.C; (* 将定时器型标签tLabel0的线圈的值代入到M1中。 *)
M2 := INV(tLabel1.C); (* 将定时器型标签tLabel1的线圈的取反结果代入到M2中。 *)
```

■使用软元件时的数据类型自动转换

以字[带符号]以外的数据类型使用字软元件的情况下，应赋予软元件类型指定符。(☞60页 带类型指定软元件标记)

例

将D2、D3的值传送至双字[无符号]的标签dwordLabel1的情况

```
//赋予软元件类型指定符，以正确的数据类型传送的示例
dwordLabel1 := D2:UD;
//赋予软元件类型指定符的D2:UD为双字[无符号]，因此会将D2、D3的值传送至dwordLabel1。

//出现意料外的传送结果的示例
dwordLabel1 := D2;
//没有软元件类型指定符的D2为字[带符号]，因此会将数据类型自动转换为双字[无符号]后，传送至dwordLabel1。
//因此，仅传送D2的值，不传送D3的值。
```

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“(*”起到结束符号“*)”为止的内容作为注释处理。 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“/*”起到结束符号“*/”为止的内容作为注释处理。 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

RnCPU RnENCPU RnPCPU(过程) RnPCPU(冗余) RnPSFCPU(常规) RnPSFCPU(安全) RnSFCPU(常规) RnSFCPU(安全)

是通过按照数据及信号的流向对特定处理的块、变量部件、常数部件进行连接，记述程序的图表语言。

要点

- 在本章中，对FBD/LD语言的动作及规格有关内容进行说明。关于创建FBD/LD程序时的操作方法，请参阅下述手册。

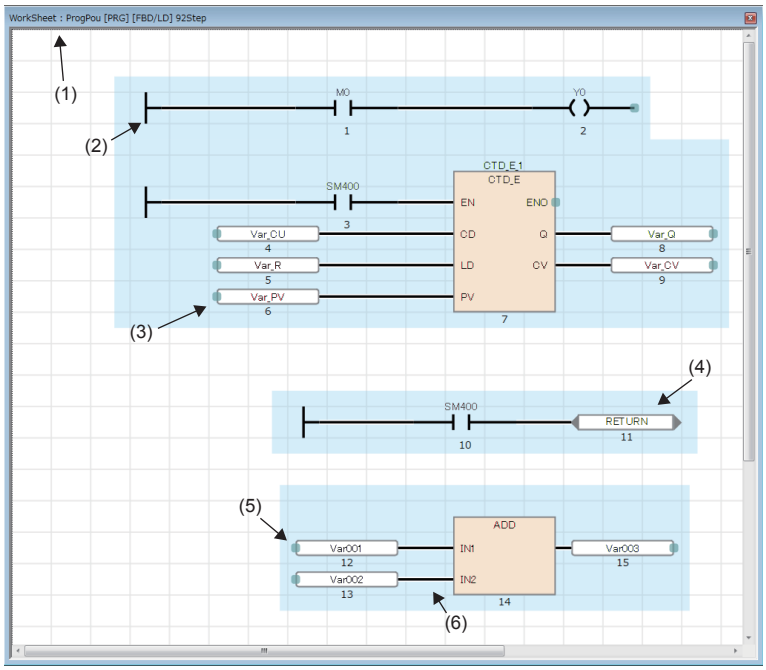
📖GX Works3 操作手册

- 通过过程CPU的过程控制FB对FB部件进行配置和连接，使创建程序变得更简易，且用于执行过程控制的FB种类丰富，使得编程变得更简单。在过程CPU中使用过程控制FB时，请参阅下述手册。

📖MELSEC iQ-R 编程手册(过程控制FB/指令篇)

7.1 配置

FBD/LD语言中，可创建以下程序。



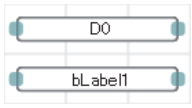
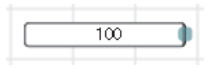
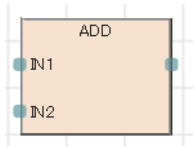
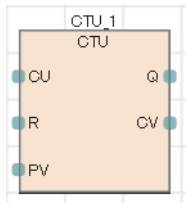
(1) 工作表
(2) LD部件
(3) FBD部件
(4) 通用部件
(5) 连接点
(6) 连接线

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

部件

FBD部件

配置FBD/LD程序的FBD部件如下所示。

项目		内容
变量		用于存储各个值(数据)时使用变量。变量中规定数据类型，仅存储该数据类型的值(数据)。 在变量中，可以指定标签或软元件。
常数		输出所指定的常数值。
函数(FUN)		执行函数。 • 函数的创建方法(GX Works3 操作手册) • 通用函数(MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇))
功能块(FB)		执行FB。 • FB的创建方法(GX Works3 操作手册) • 通用FB(MELSEC iQ-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)) • 模块FB(所使用的模块的FB参考)

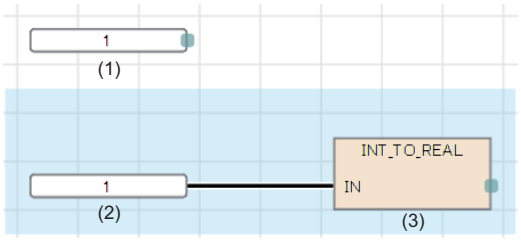
■常数部件的数据类型

常数部件的情况下，在输入常数值时尚未决定常数值的数据类型。用连接线将常数部件与FBD部件连接在一起时，决定数据类型。常数值的数据类型变为与用连接线连接的目标的FBD部件相同的数据类型。

例

在常数值中输入1的情况

由于数据类型后补中存在BOOL型、WORD型、DWORD型、INT型、DINT型、REAL型、LREAL型，因此无法决定数据类型。用连接线将常数部件与FBD部件连接时，将变为连接目标中的部件输入点的数据类型。



(1) 数据类型未决定
(2) INT型
(3) INT型

■数据类型的自动转换

连接的部件的数据类型不同时，可能会自动转换数据类型。

类型转换时，只从容量小的数据类型转换至容量大的数据类型，以确保不丢失数据。FBD/LD语言中数据类型的自动转换的动作与ST语言相同。详细内容，请参阅下述章节。

☞51页 数据类型的自动转换

■函数的输入输出点

- 需要预先将函数的所有输入点与其它部件连接。
- 函数的输入变量与输出变量的数据类型确定后，连接至输入点与输出点的部件也需与这些数据类型相符。

■带EN的函数、带EN的FB的输出变量与其它部件连接时的注意事项

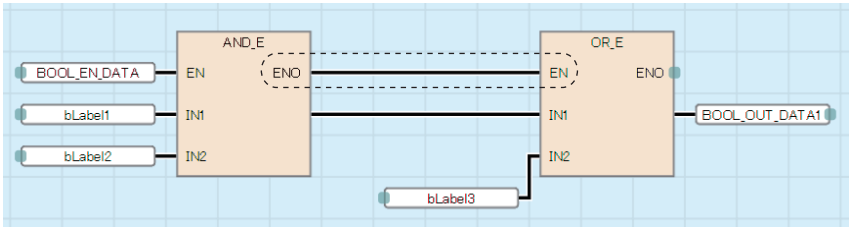
带EN的函数及带EN的FB的ENO为FALSE(运算停止)时，根据连接部件的不同，程序动作也有所不同。

连接部件	带EN的函数及带EN的FB的ENO为FALSE(运算停止)时的程序动作
• 函数 • FB	连接至带EN的函数及带EN的FB的输出变量的输入变量值变为不定值。
上述以外	连接至带EN的函数及带EN的FB的输出变量的部件值不被更改。(变为前次值。)

因此，在将带EN的FB的输出变量直接连接至另外的函数或FB时,可能发生预想之外的动作。
为防止使用不定值，应通过下述示例中的任意一种方法进行连接。

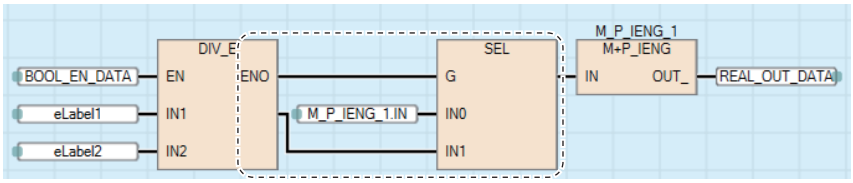
例

将连接目标作为带EN的函数或带EN的FB，对连接源的ENO与连接目标的EN进行连接。



例

使用选择函数SEL，在ENO变为FALSE时使之输入前次值。



■使用步继电器(S)及SFC块软元件(BL)的情况

在变量部件中使用将步继电器(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

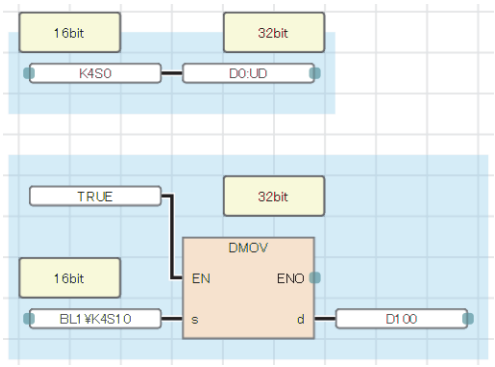
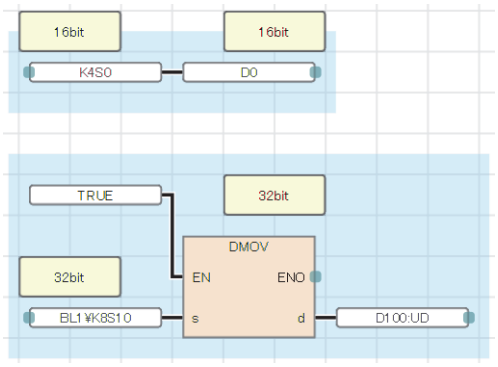
替换示例如下所示。

替换前	替换后
	

此外，在程序中使用步继电器(S)或带块指定步继电器(BL□\S□)的位指定的情况下，应指定正确的数据容量。步继电器(S)及带块指定步继电器(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。

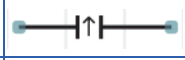
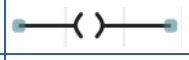
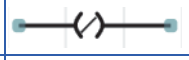
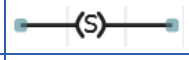
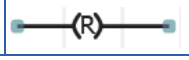
例

替换示例如下所示。

替换前	替换后
	

LD部件

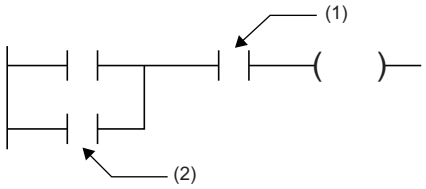
在FBD/LD程序中可使用的梯形图部件如下所示。

项目		内容
左母线		是表示母线的部件。变为创建梯形图时的起点。 左母线的输出将变为常时ON。
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。
上升沿脉冲否定		指定软元件或标签OFF时、ON时以及下降沿时(ON→OFF)导通。
下降沿脉冲否定		指定软元件或标签OFF时、ON时以及上升沿时(OFF→ON)导通。
线圈		将运算结果输出至指定软元件或标签。
取反型线圈		运算结果变为OFF时，指定软元件或标签变为ON。
设置线圈		运算结果变为ON时，指定软元件或标签变为ON。 即使运算结果变为OFF，已ON的软元件或标签也将保持ON状态不变。
复位线圈		运算结果变为ON时，指定软元件或标签变为OFF。 运算结果变为OFF的情况下，软元件或标签的状态不变化。

■触点符号的AND运算与OR运算

触点符号，根据梯形图的连接状态进行AND运算、OR运算，并反映至运算结果。

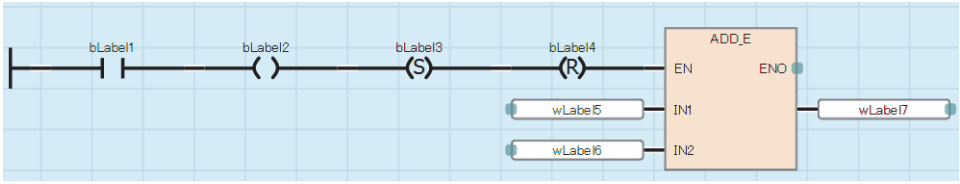
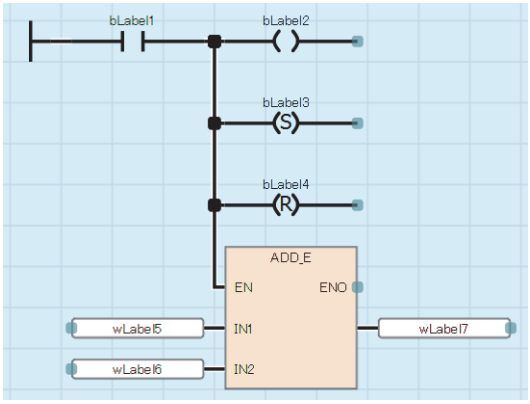
- 串联连接(1)的情况下，进行之前的运算结果与AND运算，并作为运算结果。
- 并联连接(2)的情况下，进行之前的运算结果与OR运算，并作为运算结果。



(1) 串联连接的触点
(2) 并联连接的触点

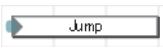
■将其它部件连接到线圈的输出连接点上的情况

将其它部件直接连接到线圈的输出连接点上时，动作将变为与并联连接时相同。

项目	内容
程序示例	
上述示例的动作	

通用部件

配置在FBD/LD程序上的通用部件如下所示。

项目		内容
跳转*1		从跳转部件开始到跳转标签为止跳转执行处理。不执行已跳转的部分。 根据至跳转部件的ON/OFF信息，控制是否进行跳转。 ON：在跳转标签中跳转执行处理。 OFF：不跳转，进行普通的执行处理。
跳转标签*1		为来自于同一程序内的跳转部件的跳转目标。进行了跳转的情况下，通过跳转标签及其以后的执行顺序的程序执行处理。
连接器		作为连接线的替代品使用。 处理将转移到成对的连接器部件中。 对一个输出连接器，可以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及其以后的处理。在不执行返回部件及其以后的程序、函数及FB处理的情况下使用。 根据返回至部件的ON/OFF信息，控制是否进行返回处理。 ON：执行返回处理。 OFF：不进行返回处理，进行普通的执行处理。
注释		记载注释的情况下使用。

*1　在SFC程序的Zoom编辑器内不能使用。

■跳转部件

- 在跳转部件中使线圈处于ON状态的定时器跳转时，将变得无法正常进行测量。
- 在跳转部件的上侧(执行顺序在前)可配置跳转标签。该情况下，应创建包含脱离循环方法的程序，以确保不超出看门狗定时器的设置值。
- 在跳转部件与跳转标签中仅可指定指针型的局部标签。而且不可以使用结构体的构件。
- 不可以使用指针分支指令(CJ、SCJ、JMP)。进行跳转的情况下，应使用跳转部件。
- 不可以跳转至程序块的外侧或从外侧跳转。

跳转关联的动作	执行可否
跳转至程序块的外侧*1	不可以执行
从程序块的外侧跳转*1	不可以执行
子程序的调用	可以执行
作为子程序被调用	不可以执行

*1　包含通过BREAK指令进行的分支。

■返回部件

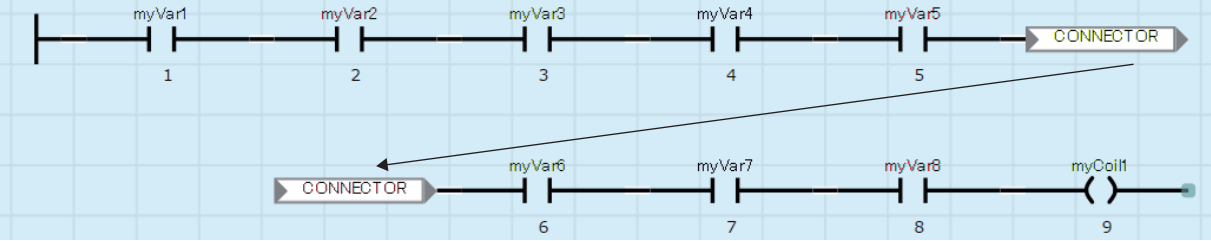
• 返回部件的动作根据使用的程序及函数、FB而有所不同。

所使用的程序部件	内容
程序	结束程序部件的执行。
函数	结束函数，返回至调用了函数指令的下一步。
FB	结束FB，返回至调用了FB指令的下一步。

• 宏型FB中使用返回部件的情况下，请勿对相同FB实例名的FB部件进行多个配置。

■连接器部件

在FBD/LD编辑器的显示范围内或印刷范围内，希望配置程序的情况下使用连接器部件。



连接线

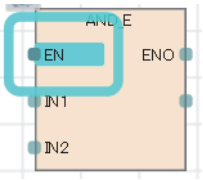
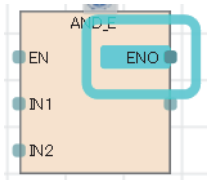
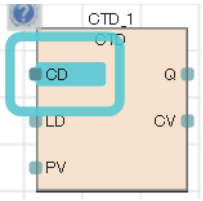
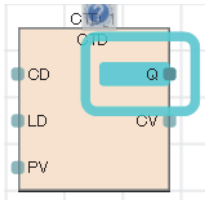
是连接FBD部件、LD部件、通用部件的连接点间的线。

连接部件从左端向右端交接数据。处于连接状态的部件的数据类型需要一致，或数据类型为可自动转换的类型。

连接点

是用连接线连接FBD部件、LD部件、通用部件时的端点。

各部件左侧点表示输入侧，右侧点表示输出侧。

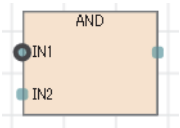
项目	输入连接点	输出连接点
触点		
线圈		
变量		
常数	—	
函数		 不显示函数的返回值名称。
FB		

连接点被连接时，将变为隐藏。

■输入输出点的取反

通过连接点可对至部件的输入或来自于部件的输出进行取反。

用黑圆圈括住取反状态的连接点，取反输入或输出的数据(FALSE→TRUE或TRUE→FALSE)。

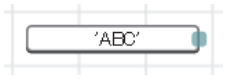
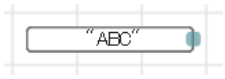


可取反的数据类型为BOOL、WORD、DWORD、ANY_BIT、ANY_BOOL。

常数

常数的显示方法

FBD/LD程序中的字符串的显示方法如下所示。

数据类型		显示方法	显示示例
字符串	STRING	将字符串(ASCII、移位JIS)用单引号(')括住。	
字符串[Unicode]	WSTRING	用双引号(")括住Unicode字符串。	

上述以外常数的显示方法，请参阅下述手册。

MELSEC iQ-R CPU模块用户手册(应用篇)

标签与软元件

指定方法

在FBD/LD程序上，可以直接记述且使用标签与软元件。可以对部件的输入点、输出点、通用函数/FB的自变量、返回值等使用标签与软元件。

标签与软元件的详细内容，请参阅下述手册。

MELSEC iQ-R CPU模块用户手册(应用篇)

■带类型指定软元件标记

通过对软元件名附加软元件型指定符，可以以任意数据类型使用字软元件。

FBD/LD语言中的软元件型指定符与可使用的软元件与ST语言相同。详细内容，请参阅下述章节。

☞ 60页 带类型指定软元件标记

不指定字软元件的数据类型时，数据类型由软元件的类型决定。

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
长定时器软元件的当前值(LTN)、长累计定时器软元件的当前值(LSTN)、长计数器软元件的当前值(LCN)	DWORD
长变址寄存器(LZ)	DINT
上述以外	ANY16

注意事项

■使用标签的情况

- 在数组的下标中，不能使用局部软元件。但是，将下标中使用的软元件代入其它软元件中、将代入目标的软元件指定为下标后，变为可使用。

■使用软元件时的数据类型自动转换

以字[带符号]以外的数据类型使用字软元件的情况下，应赋予软元件类型指定符。(☞ 72页 带类型指定软元件标记)

例

将D2、D3的值传送至双字[无符号]的标签dwordLabel1的情况

赋予软元件类型指定符，以正确的数据类型传送的示例	出现意料外的传送结果的示例
	
赋予了软元件类型指定符的D2：UD为双字[无符号]，因此会将D2、D3的值传送至dwordLabel1。	没有软元件类型指定符的D2为字[带符号]，因此会在将数据类型自动转换为双字[无符号]后，传送至dwordLabel1。 因此，仅传送D2的值，不传送D3的值。

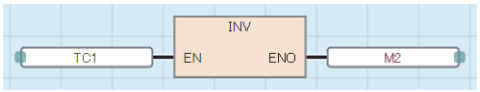
■使用定时器、计数器的情况

- 将定时器及计数器的软元件的线圈(TC、STC、LTC、LSTC、CC、LCC)作为变量、函数及FB的输入使用的情况下，将作为触点(TS、STS、LTS、LSTS、CS、LCS)进行动作。
- 希望将定时器及计数器的线圈作为输入使用的情况下，应使用定时器型及计数器型的标签。

例

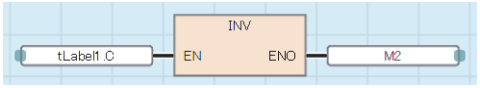
定时器软元件及定时器型标签的情况

将触点(TS0)的值代入到M1中。



将触点(TS1)的取反结果代入到M2中。

将定时器型标签tLabel0的线圈的值代入到M1中。

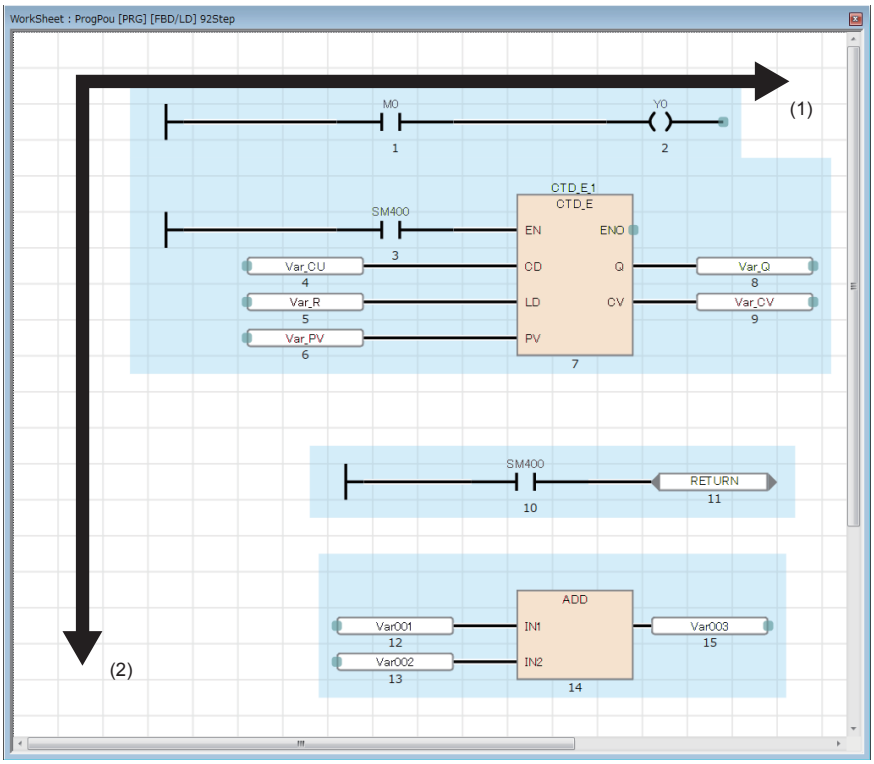


将定时器型标签tLabel1的线圈的取反结果代入到M2中。

7.2 程序执行顺序

部件的执行顺序

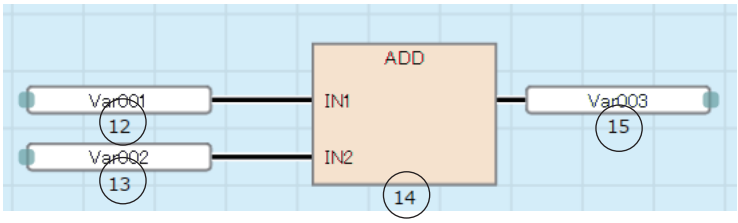
FBD/LD编辑器上部件的执行顺序依部件位置关系与连接状况而定。



(1) 从左至右执行。

(2) 从上至下执行。

配置在FBD/LD编辑器上的各部件显示执行顺序的编号。如果转换程序，则显示确定的执行顺序。

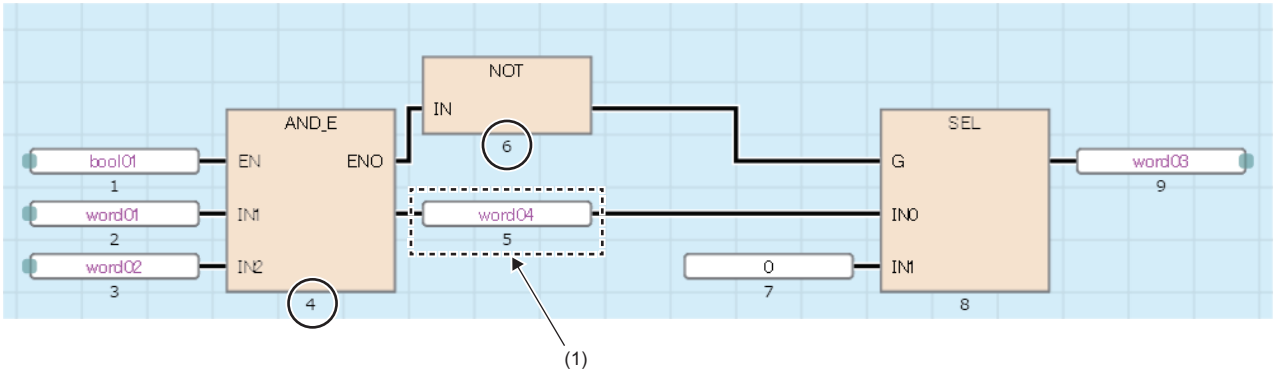


注意事项

如果使用函数的程序，请勿将连接函数的返回值与其他函数的输入变量直接连接，而是在之间连接变量部件。

例

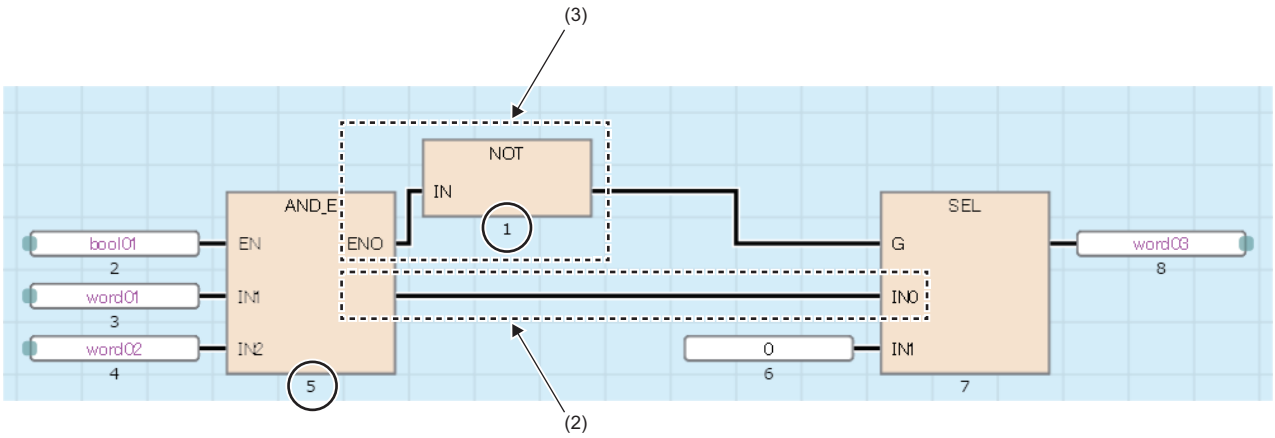
在返回值与输入变量之间连接了变量部件(1)的情况



如果直接连接函数的返回值与输入变量，可能不会变为预想内的执行顺序。

例

不变为预想内的执行顺序的情况



配置在左方的部件的返回值(2)与经由其他部件的输出变量(3)连接了配置在右方的部件的输入，所以执行顺序不同。

8 SFC程序

RnCPU RnENCPU RnPCPU(过程) RnPCPU(冗余) RnPSFCPU(常规) RnPSFCPU(安全) RnSFCPU(常规) RnSFCPU(安全)

是将一系列的控制动作分割为多个步，以便各程序的执行顺序及执行条件能清楚表示的程序的记述形式。

要点

在本章中，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。

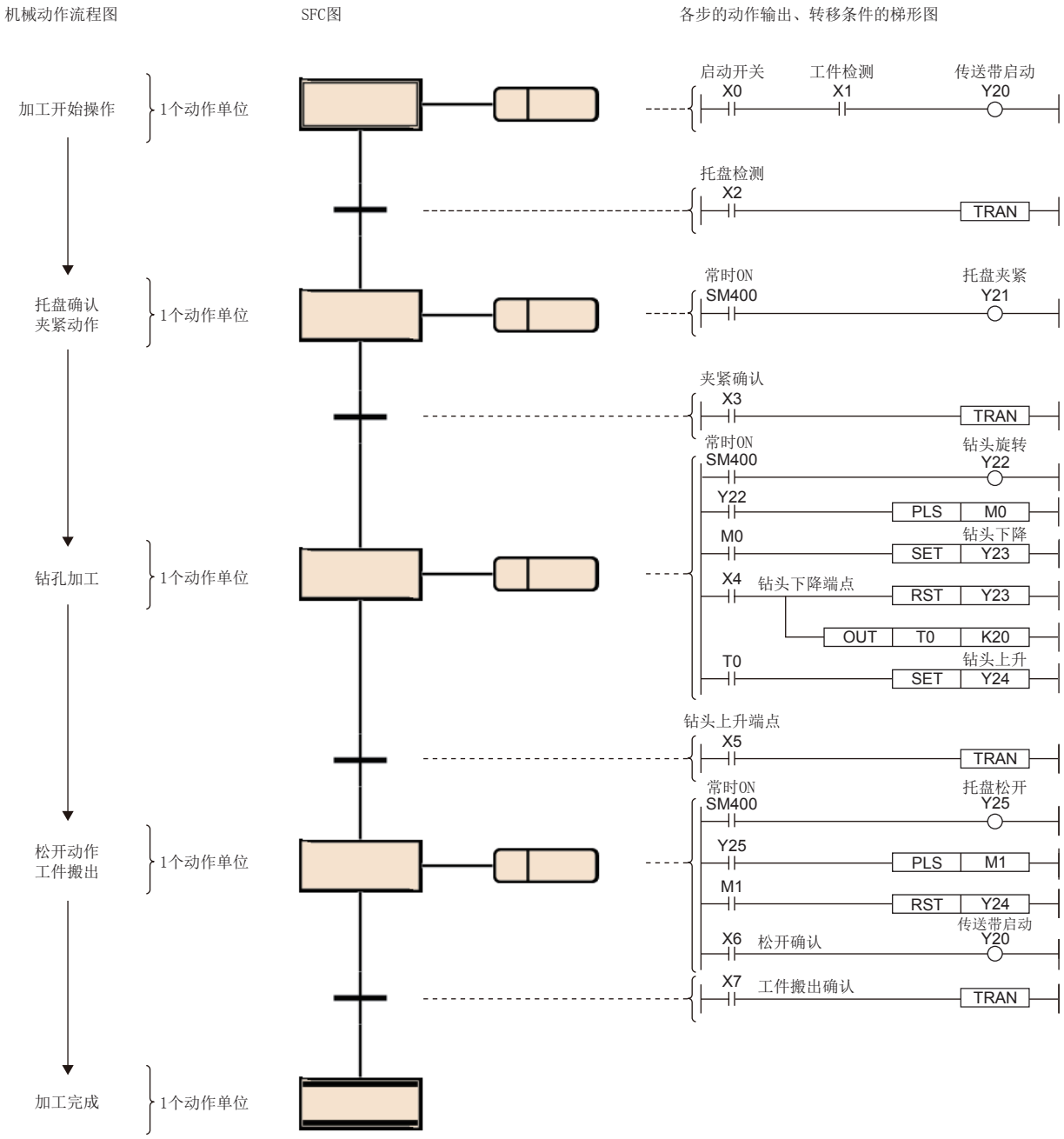
GX Works3 操作手册
MELSEC iQ-R CPU模块用户手册(应用篇)

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模块及工程工具的版本，请参阅下述手册。

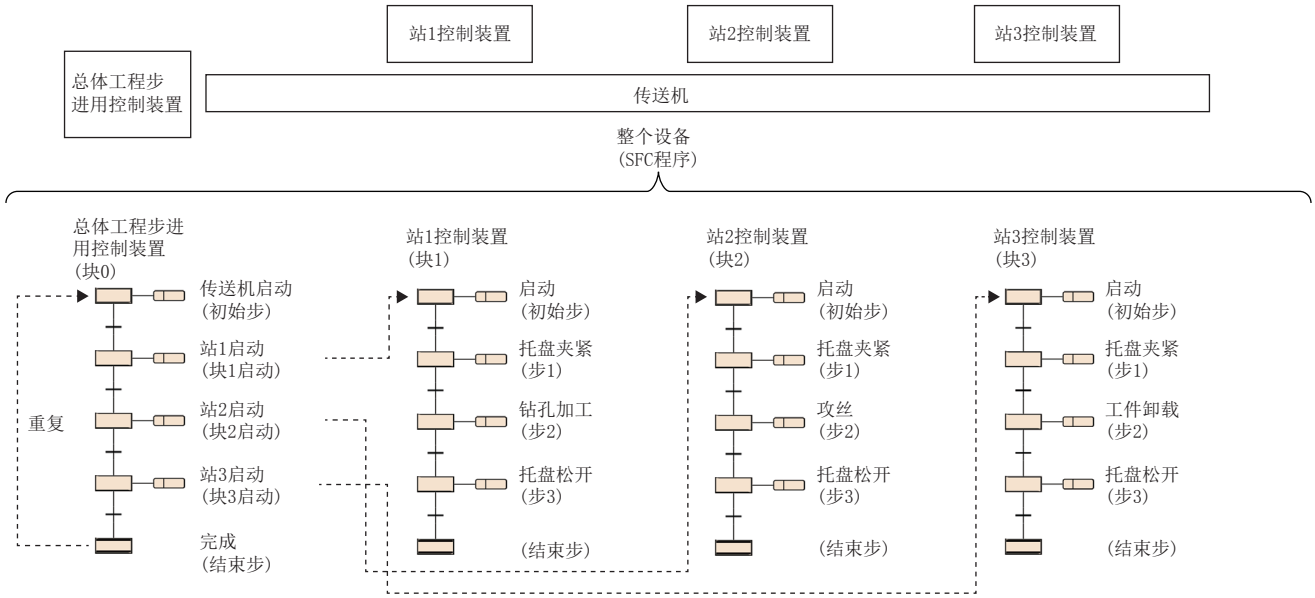
MELSEC iQ-R CPU模块用户手册(应用篇)

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、各机械的实际控制以1对1与SFC程序的各块、各步对应。



8.1 规格

SFC程序相关的性能规格如下所示。

项目		规格
软元件点数(SFC关联)	步继电器(S)	R00CPU、R01CPU、R02CPU： 最多8192点 上述以外的CPU模块： 最多16384点
	SFC块软元件(BL)	R00CPU、R01CPU、R02CPU： 128点 上述以外的CPU模块： 320点
	SFC转移软元件(TR)	0点
SFC程序执行个数		1个
块数		R00CPU、R01CPU、R02CPU： 最多128块 上述以外的CPU模块： 最多320块
SFC步数		R00CPU、R01CPU、R02CPU： 所有块最多1024步，1个块最多128步 上述以外的CPU模块： 所有块最多16384步，1个块最多512步
步No.		R00CPU、R01CPU、R02CPU： 每一个块为0～127 上述以外的CPU模块： 每一个块为0～511
分支数		最多32分支
同时激活步数		R00CPU、R01CPU、R02CPU： 所有块最多1024步，1个块最多128步 上述以外的CPU模块： 所有块最多1280步，1个块最多256步
初始步数		最多32个/块
动作输出数		最多4个/步
顺控程序步数	动作输出	1个块约32K顺控程序步 (对每1步无限制)
	转移条件	仅1个梯形图块
SFC块RUN中写入对象块数		1块*1

*1 关于SFC块RUN中写入，请参阅下述章节。
☞ 137页 SFC块RUN中写入
且使用SFC块RUN中写入的情况下，应确认CPU模块及工程工具的版本。(MELSEC iQ-R CPU模块用户手册(应用篇))

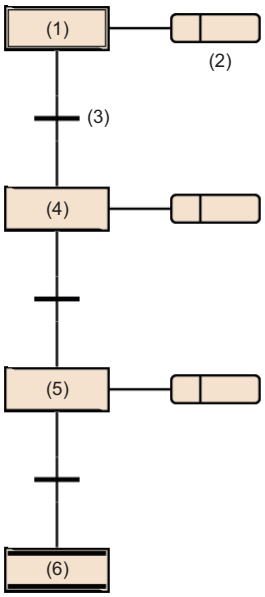
要点

关于SFC程序的处理时间有关内容，请参阅下述手册。
MELSEC iQ-R CPU模块用户手册(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

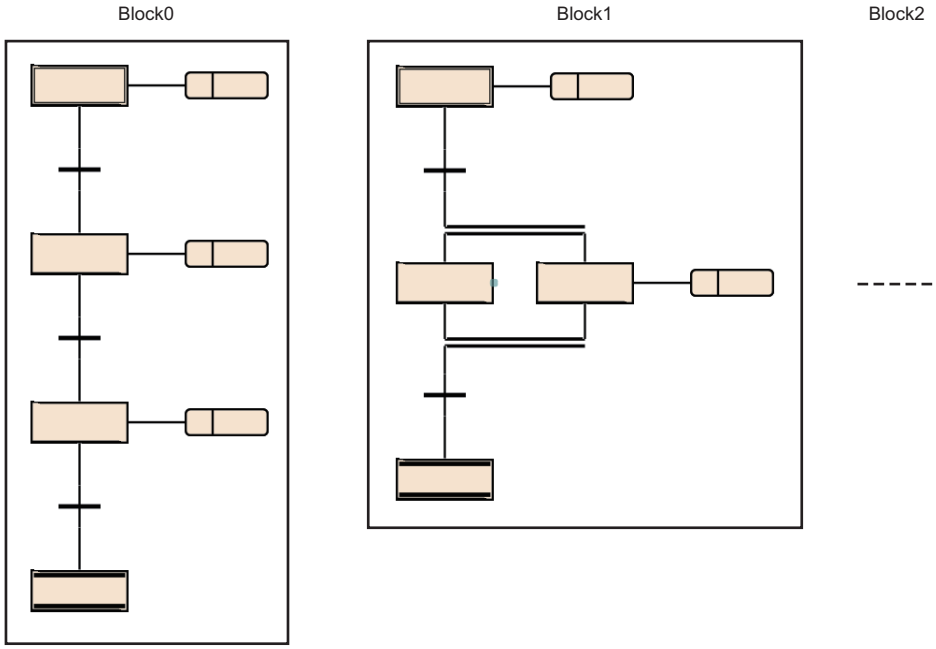
1. 块启动时，首先激活初始步(1)，执行动作输出(2)。动作输出(2)执行后检查下一个转移条件(3)是否成立。
2. 在转移条件(3)成立之前，仅执行动作输出(2)。转移条件(3)成立时将结束动作输出(2)，初始步(1)变为非激活状态，激活下一个普通步(4)。
3. 在执行普通步(4)的动作输出后，检查下一个转移条件是否成立。如果下一个转移条件成立，将重复执行普通步(4)的动作输出。
4. 转移条件成立时将结束动作输出，初始步(4)变为非激活状态，激活下一个步(5)。
5. 每当转移条件成立时将激活下一个步，最后激活结束步(6)时结束块。

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，将从上开始按顺序执行。(☞94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。(☞83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。

☞79页 规格

在块内，变为从初始步开始交互连接步及转移条件，并以结束步或跳转转移结束的构成。

块具有激活/非激活的状态。

- 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

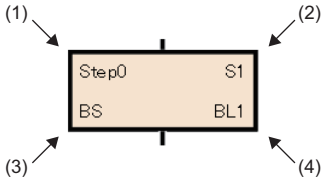
块从非激活变为激活时，初始步将变为激活，依次执行处理。(☞127页 各块的执行顺序)

要点

- 通过CPU参数的设置，仅块0启动SFC程序时可以自动启动。在这种情况下激活结束步、结束块0时，块0将自动被再启动，再次从初始步开始执行。(☞121页 启动条件设置)
- 通过SET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，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。

☞79页 规格

步有下述特征。

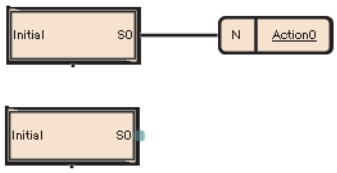
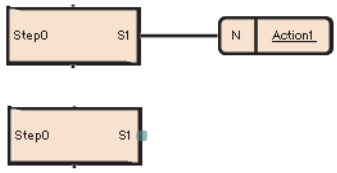
- 步激活时，执行相关的动作输出。
- 各步时，步No.被添加。步No.用于对执行步进行监视的情况及通过SFC控制指令进行强制启动及强制停止的情况。(☞92页 至步继电器(S)的步的分配)
- 步名及步No.在各块内是固有的。(不可以空栏。)

要点

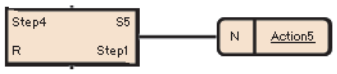
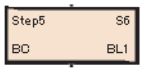
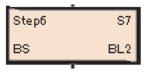
步名、步No.、属性、属性指定目标可以通过步的属性画面更改。
选择步后，对菜单的[编辑]⇨[属性]进行选择时，将显示步的属性画面。(📖GX Works3 操作手册)

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、SE、ST、R的属性。 也可以置为不创建动作输出的步。
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、SE、ST、R、BC、BS的属性。 也可以置为不创建动作输出的步。
结束步		结束块的步。 无法创建动作输出。

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，动作输出为ON的线圈仍保持输出的步。
SE	动作保持步(无转移检查)[SE]		激活转移后继续执行动作输出的步。 转移条件成立，下一个步激活后将不进行转移条件的检查。
ST	动作保持步(有转移检查)[ST]		激活转移后继续执行动作输出的步。 转移条件成立，下一个步被激活后，也将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“属性”的设置可以更改步的类型。
- 复位步[R]、块启动步(有结束检查)[BC]、块启动步(无结束检查)[BS]，对属性画面中的“属性指定目标”指定步名或块No.。

关于设置方法有关内容，请参阅下述手册。

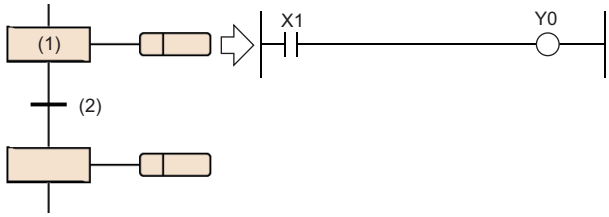
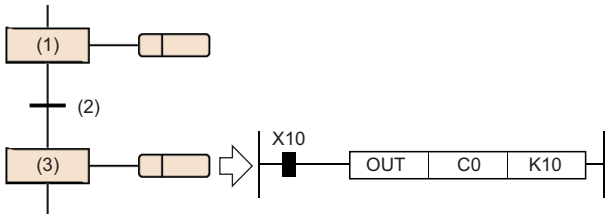
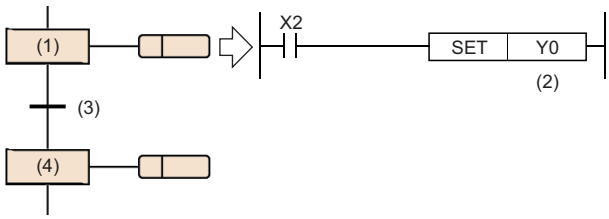
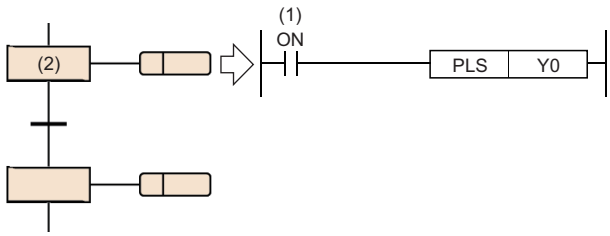
GX Works3 操作手册

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，转移条件成立且该步激活时，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，即使该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，在其它步中将执行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，即使该步变为非激活状态，也将保持ON状态或软元件/标签中存储的数据。 ON状态的软元件/标签的OFF或软元件/标签中存储的数据的清除时，应在其它步中通过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，每当该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

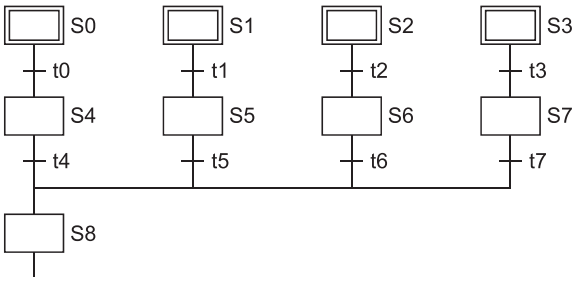
初始步是表示各块的起始的步，各块中最多可记述32个。(☞ 79页 规格) 对多个初始步进行合并时只能进行选择合并。初始步的执行方法与初始步以外的步相同。

■块启动时的激活步

多个初始步的情况下，在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	通过块启动步开始了启动时
	通过SFC控制指令的块启动指令开始了启动时
	通过SFC用信息软元件的块启动结束位强制开始了启动时
	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

■初始步为多个激活步时的转移处理



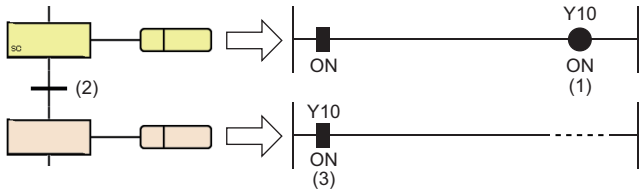
选择合并初始步为多个激活步的块时，如果在合并之前任意1个转移条件成立，则合并之后的步将被激活。在上述程序示例中，如果转移条件t4～t7的其中一个成立，则步8(S8)将被激活。此外，在合并之后的步(在上述程序示例中为S8)被激活后，如果合并之前的其它转移条件(在上述程序示例中为t4～t7)成立，则合并之后的步将再次被激活。

■将步属性附加到初始步时的动作

可以附加SC(线圈保持)、SE(动作保持(无转移检查))、ST(动作保持(有转移检查))、R(复位)的各属性到初始步中。此外，附加了属性的情况下，与初始步以外的步相同，但块启动时自动被激活的动作除外。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，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(SM327为ON时除外)
- 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
- 通过SFC控制指令的RST指令(步结束)对步进行了复位的情况
- 对指定为SFC用信息软元件的块启动结束位中的软元件进行了复位的情况
- 设置了用于复位线圈保持步[SC]的复位步[R]被激活的情况
- 将SM321(SFC程序的启动/停止)置为OFF的情况
- 通过程序对线圈进行了复位的情况
- 在停止时输出模式为OFF的状态下开始了停止指令的情况
- 在块内的复位步[R]中指定了S999的情况

■块停止/重启时的动作

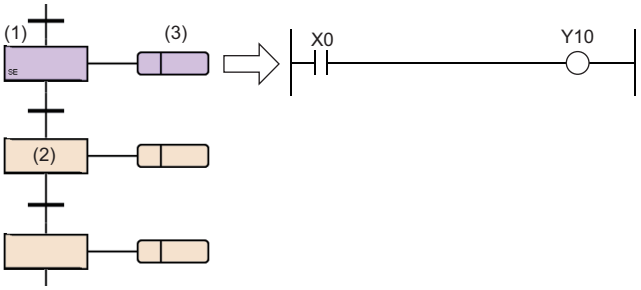
块停止/重启时的动作，根据SM325(块停止时的输出模式设置)与SFC用信息软元件的停止时模式位的设置、步的保持/非保持的组合决定。(☞ 122页 块停止重启时的动作)

动作保持步(无转移检查)[SE]

激活转移后继续执行动作输出的步。

通过转移条件的成立转移到下一个步后，仍将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发生变化。

转移条件成立且下一个步被激活后不进行转移条件的检查，即使转移条件再次成立，也不进行至下一个步的转移。



从步(1)转移至步(2)时，步(1)将变为保持中。
保持中时，不进行转移检查，但是将继续执行动作输出(3)。
在这种情况下，根据X0的ON/OFF，Y10将变为ON/OFF。

■变为非激活的时机

动作保持步(无转移检查)[SE]变为非激活状态的时机如下所示。

- 执行了块的结束步的情况
- 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
- 通过SFC控制指令的RST指令(步结束)对步进行了复位的情况
- 对指定为SFC用信息软元件的块启动结束位中的软元件进行了复位的情况
- 设置了用于复位动作保持步(无转移检查)[SE]的复位步[R]被激活的情况
- 将SM321(SFC程序的启动/停止)置为OFF的情况
- 在块内的复位步[R]中指定了S999的情况

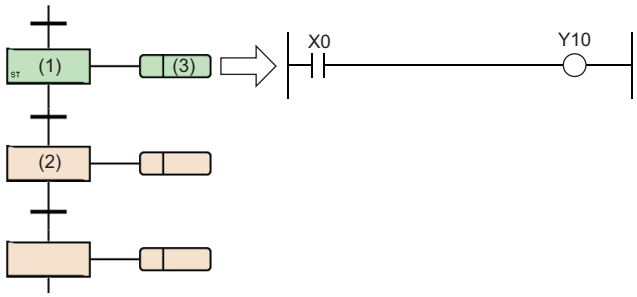
■块停止/重启时的动作

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用信息软元件的停止时模式位的设置、步的保持/非保持的组合决定。(☞ 122页 块停止重启时的动作)

动作保持步(有转移检查)[ST]

激活转移后继续执行动作输出的步。

通过转移条件的成立转移到下一个步后，仍将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发生变化。转移条件成立，下一个步被激活后，也将重复进行转移条件的检查。转移条件再次成立时，再次激活下一个步的同时，继续进行动作输出内的运算的继续运行。



从步(1)转移至步(2)时，步(1)将变为保持中。
保持中时与普通的激活步一样，动作输出(3)也将继续执行。
在这种情况下，根据X0的ON/OFF，Y10将变为ON/OFF。
此外也进行转移检查，转移条件成立时，将下一个步激活。

■变为非激活的时机

动作保持步(有转移检查)[ST]变为非激活状态的时机如下所示。

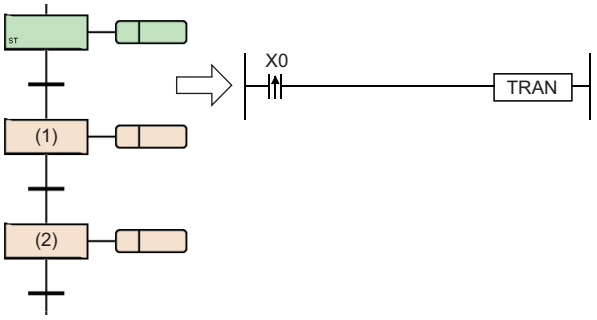
- 执行了块的结束步的情况
- 通过SFC控制指令的RST指令(块结束)对块进行了强制结束的情况
- 通过SFC控制指令的RST指令(步结束)对步进行了复位的情况
- 对指定为SFC用信息软元件的块启动结束位中的软元件进行了复位的情况
- 设置了用于复位动作保持步(有转移检查)[ST]的复位步[R]被激活的情况
- 将SM321(SFC程序的启动/停止)置为OFF的情况
- 在块内的复位步[R]中指定了S999的情况

■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用信息软元件的停止时模式位的设置、步的保持/非保持的组合决定。(☞ 122页 块停止重启时的动作)

■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，每个扫描的下一个步将被启动。置为每个扫描不进行转移时，应在转移条件中使用PLS指令等上升沿执行的指令。



通过将转移条件置为上升沿脉冲运算开始的条件，仅X0变为ON的瞬间的1个扫描可启动步(1)。
即使从步(1)转移至步(2)，步(1)变为非激活状态，如果X0不再次变为OFF→ON，则步(1)将不启动。

- SM328(END步到达时清除处理模式)为ON的情况下，应常时将动作保持步(有转移检查)[ST]之后的转移条件置为不成立。下一个步始终变为非保持的激活状态，因此无法结束块。

复位步[R]

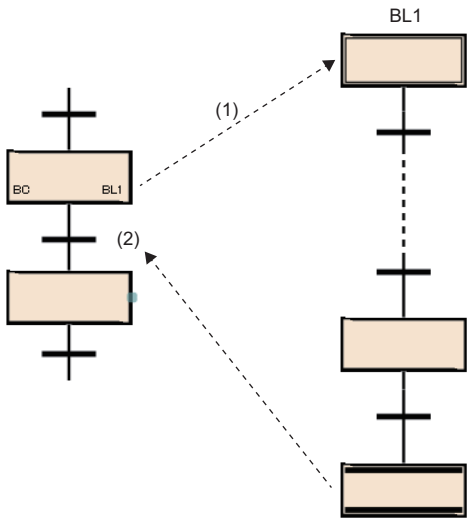
是将指定步置为非激活的步。

- 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。复位指定步以外将与普通的步(无属性)相同。
- 指定的步No.为S999的情况下，将本块内保持中的保持步[SC、SE、ST]全部置为非激活。在这种情况下，仅保持中的保持步[SC、SE、ST]可进行非激活。动作保持步[SE、ST]正在以非保持进行动作时，将不为非激活的对象。
- 不可以指定本步No.到指定步No.中。

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的动作，按照块冗余启动时的运行设置进行。(☞124页 块冗余启动时的运行设置)

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

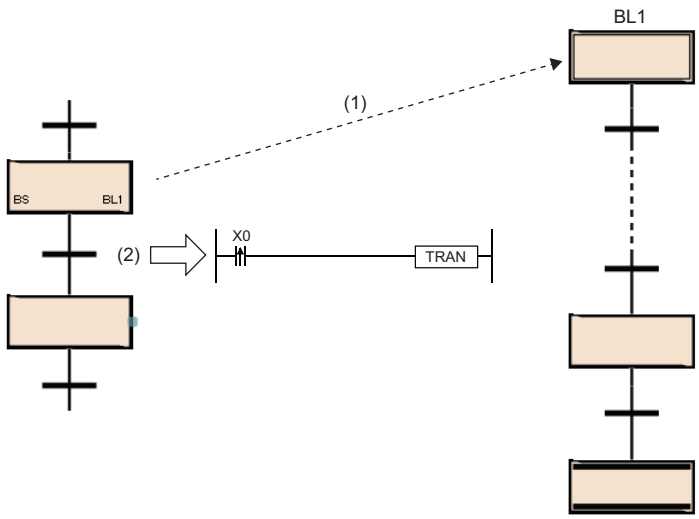
■注意事项

- 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]。在并联合并的合并之前进行创建的情况下，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的动作，按照块冗余启动时的运行设置进行。(☞124页 块冗余启动时的运行设置)

可指定的块仅1个。同时启动多个块的情况下，在使用并联分支后，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[SC、SE、ST]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。 在保持状态不变的情况下继续块的执行，不结束块。

- 在执行清除处理时，将通过OUT指令进行的线圈输出全部置为OFF。但是，关于保持中步[SC、SE、ST]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。

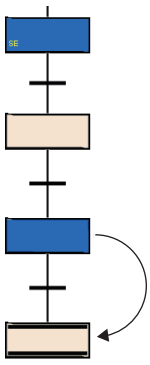
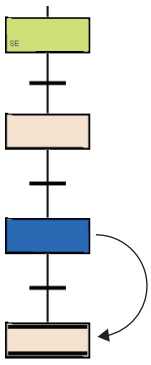
SM327的状态	内容
OFF(默认)	将保持中步[SC、SE、ST]的输出全部置为OFF。
ON	将保持中步[SC、SE、ST]的输出全部保持。 SM327的设置仅对保持中的保持步[SC、SE、ST]有效。转移条件未成立且不处于保持中的保持步[SC、SE、ST]的输出将全部OFF。此外即使SM327为ON时，步也将由激活状态变为非激活状态。 但是，通过块结束指令等进行强制结束的情况下，所有步的线圈输出将变为OFF。

- 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设置中将块0的启动条件设置为“自动起动”	自动再次激活初始步，重复执行处理。
	在参数的SFC设置中将块0的启动条件设置为“不自动起动”	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控制指令的SET指令(块启动)。 • 将SFC用信息软元件的块启动结束位置为ON。
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RST指令(块结束)等进行强制结束的情况下，将所有步的线圈输出置为OFF。
- 在激活转移到结束步中的时刻仅保持中步[SC、SE、ST]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[SC、SE、ST]将变为非激活。不希望将保持中步[SC、SE、ST]的线圈输出置为OFF的情况下，应将SM327置为ON。SM328与保持步[SC、SE、ST]的动作的关系如下所示。

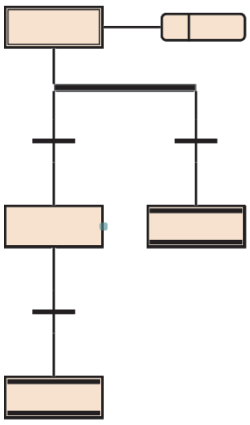
普通的激活步有剩余或转移不成立的保持步[SC、SE、ST]有剩余的情况(非保持)	保持中的激活步剩余的情况
	
• SM328为OFF时，进行清除后结束块。 • SM328为ON时，不进行清除，而是继续进行处理。	• 与SM328的设置无关，进行清除后结束块。

- SM328为ON的情况下，通过块启动步被启动的块将在非保持的激活中步不存在于块内时返回到原来的块处理。
- 动作保持步(有转移检查)[ST]之后的转移条件应置为不常时成立。动作保持步(有转移检查)[ST]之后的转移条件为常时成立的情况下，下一个步始终变为激活状态，因此SM328为ON时将无法结束块。

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。



至步继电器(S)的步的分配

步继电器是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。

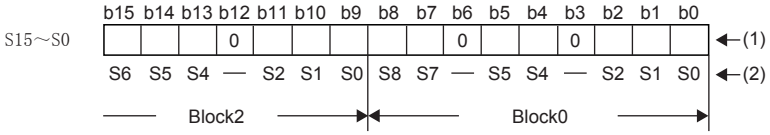
步继电器按下述方式分配。

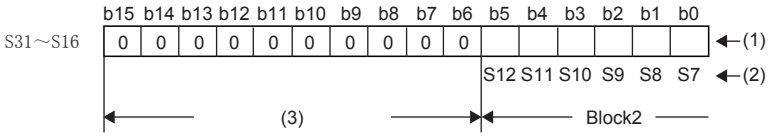
- 从SFC程序的块0开始按块No.顺序，在1个块内按步No.顺序从起始开始向末尾分配步继电器。
- 不分配步继电器到不存在的块No.中。
- 在1个块内，步继电器被分配到缺少编号的步No.中。该位始终变为OFF。
- 最后的块中，分配的步继电器以后的位将全部变为OFF。

例

下述的块配置时的步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。





(1) 存储的数据
(2) 块内的步No.
(3) 不存在，全部为0

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块No.0。
每一个块可使用的步No.，请参阅下述章节。
☞79页 规格
不可以进行超出上限的步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.。
指定其它块的步继电器的情况下，按下述形式进行指定。

例
指定块No.12的步No.23的情况

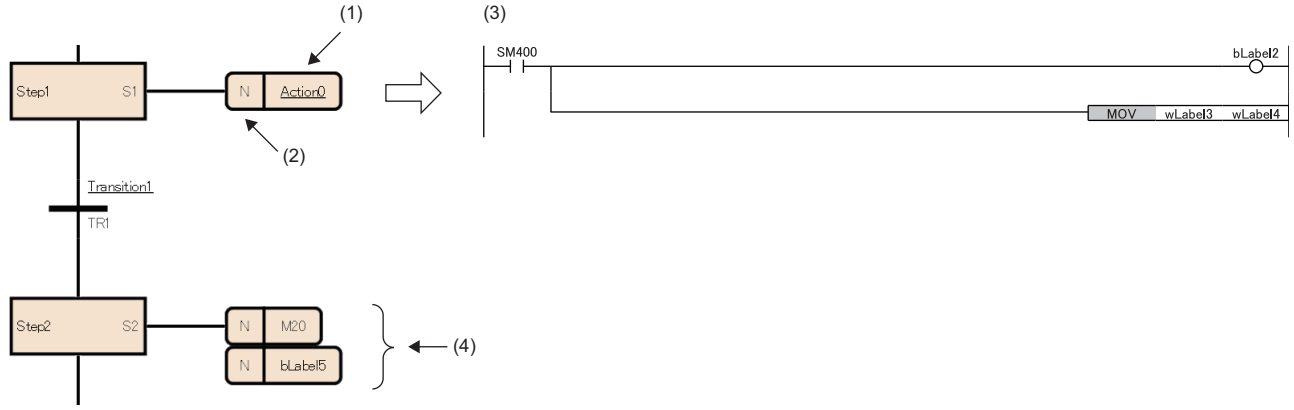
程序类型		软元件表记	内容
SFC程序	同一块内	S23	指定同一块内的步的情况下，可以省略块名。
	块12以外	BL12\S23	指定块No.及步No.。
SFC程序以外的顺控程序	指定当前对象块的情况	S23	指定对象块内的步的情况下，可以省略块名。
	指定与当前对象块不同的块的情况	BL12\S23	指定块No.及步No.。

■注意事项

• 即使SFC设置的“块停止时的输出模式”设置变为OFF，停止中步继电器也变为ON。

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，变为非执行，直至下一个步被激活为止。

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，将从上开始按顺序执行。

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。如果是梯形图语言，则可以切换详细表示与MELSAP-L(指令形式)。(☞95页 MELSAP-L(指令形式)的动作输出)

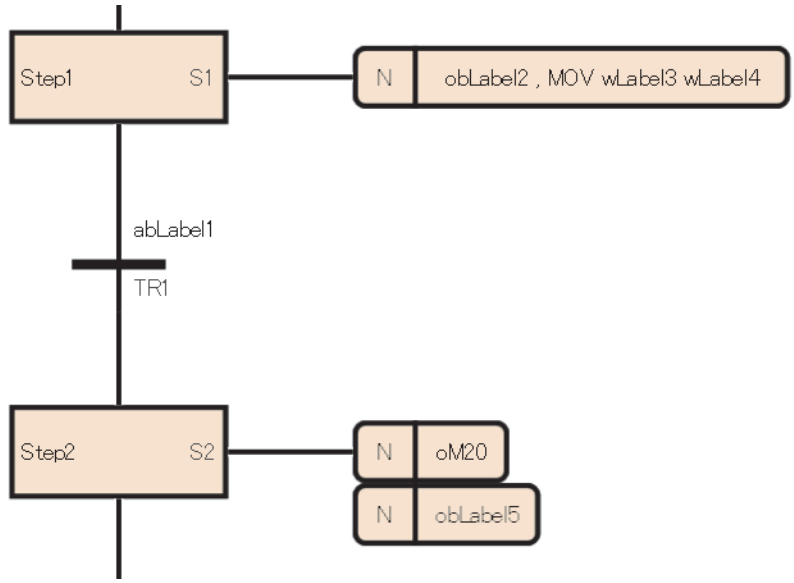
要点

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

MELSAP-L(指令形式)的动作输出

MELSAP-L(指令形式)是使用文本在SFC图内记述动作输出指令的形式。



要点

从梯形图的详细表示切换为MELSAP-L(指令形式)的情况下，选择菜单的[显示]⇨[梯形图显示切换]⇨[MELSAP-L(指令格式)]。(GX Works3 操作手册)

在通过MELSAP-L(指令形式)的动作输出中，不记述作为各指令输入条件的触点，而是记述指令或希望输出的线圈。

MELSAP-L(指令形式)采用下述形式记述程序。

□：可使用的标签/软元件，Kn：定时器/计数器的设置值

项目	MELSAP-L(指令形式)	记述示例
线圈输出(OUT指令)	o□	oY0
软元件的设置(SET指令)	s□	sM0
软元件的复位(RST指令)	r□	rM0
低速定时器(OUT T指令)*1	o□ Kn	oT0 K100
高速定时器(OUTH T指令)	h□ Kn	hT1 K10
计数器(OUT C指令)*1	o□ Kn	oC0 K10
上述以外的指令*2	与梯形图语言中的指令输入以同样方式记述	MOV D10 D120

*1 长定时器、长计数器也以同样方式指定。

*2 部分指令无法使用。(☞ 96页 无法使用的指令)

记述多个指令时，使用“,”(逗号)间隔。在动作输出的最后记述IMASK、NOPLF。

无法使用的指令

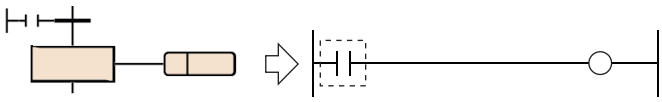
在动作输出内，有一部分的指令无法使用。无法使用的指令如下所示。

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
	SCJ*1
	JMP*1
	GOEND
程序执行控制指令	IRET
结构化指令	BREAK*1
	RET
转移条件虚拟输出	TRAN

*1　在动作输出内的函数/FB内，可以使用。

要点

必须在详细表示内的梯形图中创建作为各指令的输入条件的触点。



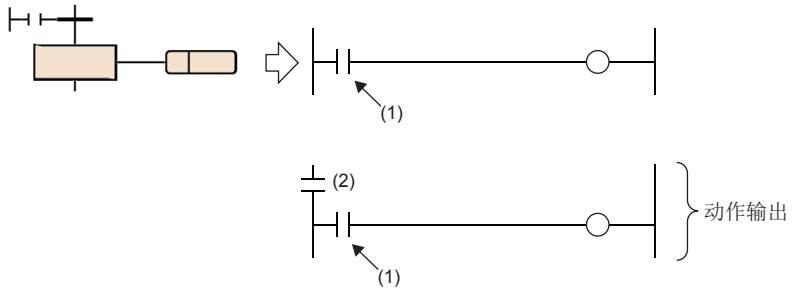
■限制事项

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/FB ・包括动作输出中无法使用的指令的函数/FB ・包括指针的函数/FB ・“使用MC/MCR控制EN”设置为“是”，且“使用EN/ENO”设置为“否”的宏型FB
	MELSAP-L(指令形式)	不可以记述相当于触点的指令(包含LD<等比较运算指令)、NOP、MPS、MRD、MPP、指针、中断指针、函数、FB。
ST语言		☞47页 ST语言
FBD/LD语言		☞63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。



(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，若希望将CALL目标的输出置为OFF，应在CALL指令后记述FCALL指令或使用XCALL指令。在步的动作输出内进行子程序调用的情况下，如果使用XCALL指令可以减少步数。

• 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，通过将线圈保持步[SC]的线圈输出置为OFF的条件，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，请参阅下述章节。

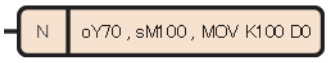
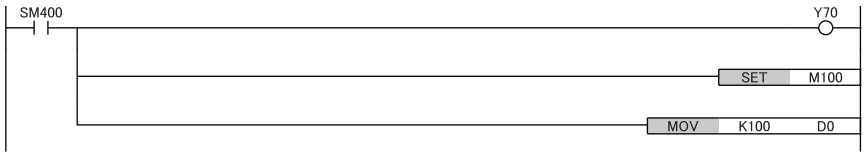
☞86页 线圈输出OFF的时机

• 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。

• 线圈保持步[SC]中转移条件成立或SM325(块停止时的输出模式)设置为保持，且停止了步的情况下，如果仅在保持线圈输出时不进行动作的情况下，由于处于非执行的状态，动作重启时的各指令的动作将被变为非执行前的执行条件所左右。

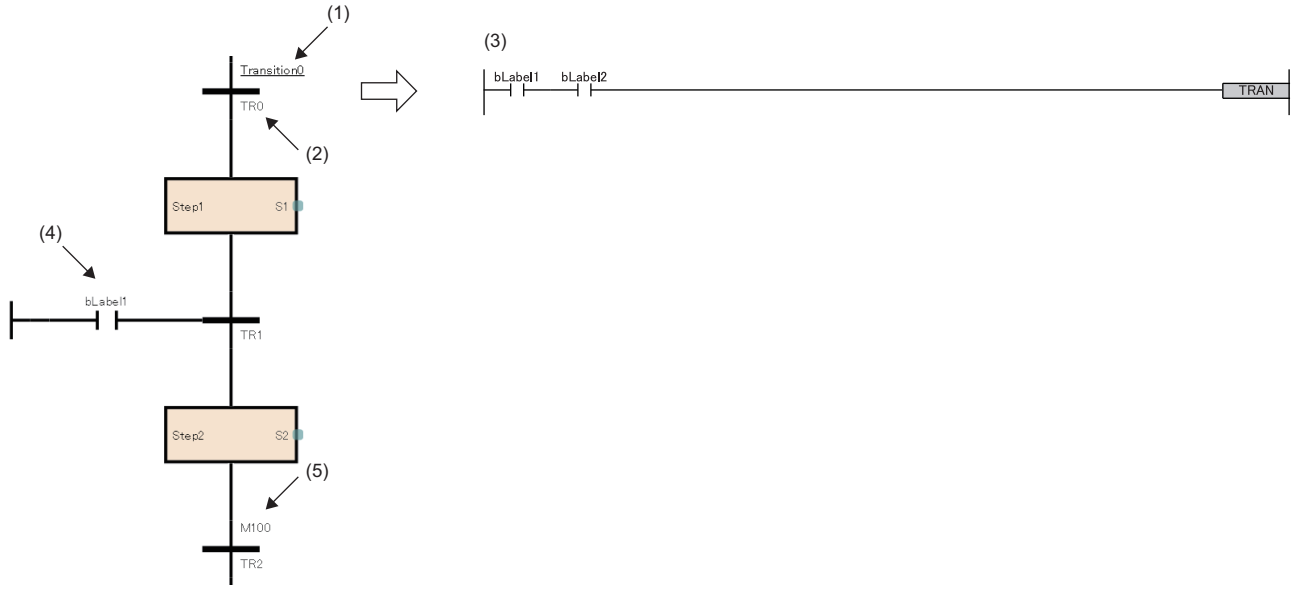
• 从梯形图的详细表示切换为MELSAP-L(指令形式)时，在详细表示中创建了MELSAP-L(指令形式)无法记述的程序的情况下，在MELSAP-L(指令形式)中显示为“????????”。此外，如果程序内使用的标签的定义被删除，也以同样方式显示。希望进行程序的确认或修正的情况下，应切换为详细表示。

• 以MELSAP-L(指令形式)创建的程序切换为梯形图的详细表示的情况下，作为指令的执行条件，追加SM400(常时ON)触点。

MELSAP-L(指令形式)	梯形图的详细表示
	

转移条件

转移条件是构成块的基本单位，通过条件成立将激活转移到下一个步。

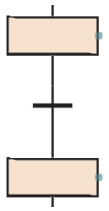
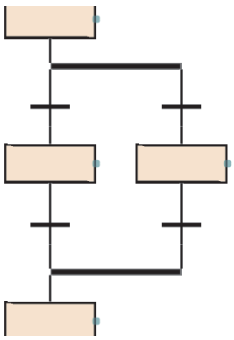
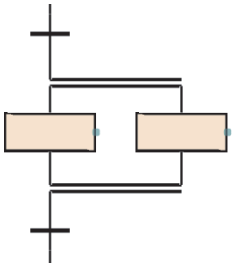
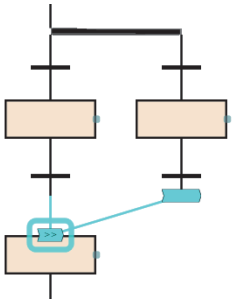


(1) 转移条件名
(2) 转移条件No.
(3) 转移条件的详细表示(☞ 104页 转移条件的详细表示)
(4) 转移条件的直接表示(☞ 107页 转移条件的直接表示)
(5) 转移条件的标签/软元件(☞ 107页 转移条件的标签/软元件)

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。如果是梯形图语言，则可以切换详细表示与MELSAP-L(指令形式)。(☞ 106页 MELSAP-L(指令形式)中的转移条件)

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，则在通用的转移条件成立时，激活将转移至下一个步。
跳转转移		通过转移条件成立，激活转移至同一块内的指定的步。

要点

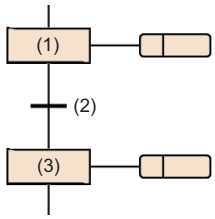
关于转移至已激活的步时的动作有关内容，请参阅下述章节。

☞134页 步冗余启动时的注意点

串行转移

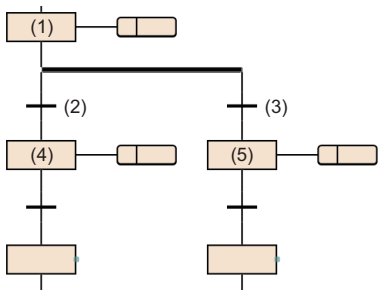
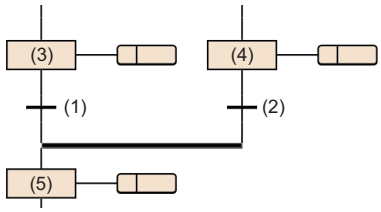
如果转移条件成立，则激活将从先行的步转移至后续的步。

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。



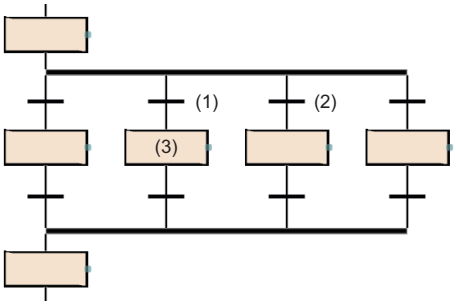
选择转移(分支/合并)

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，将转移条件(2)或转移条件(3)中条件先成立的步((4)或(5))置为激活。 步(1)将变为非激活。但是，保持步[SC、SE、ST]的情况下，将按照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，将步(5)置为激活。 已激活的步(3)或(4)将变为非激活。但是，保持步[SC、SE、ST]的情况下，将按照属性保持线圈输出或动作输出。

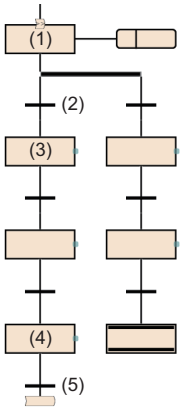
- 在选择转移中，最多可以分支为32个转移条件。
- 多个转移条件同时成立的情况下，将优先执行左侧的转移条件。

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及并联分支与选择合并组合的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

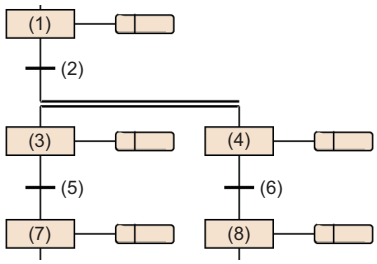
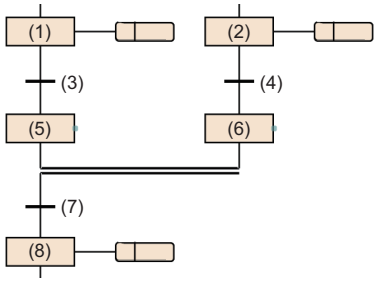
通过将选择分支的左端以外的步更改为结束步，并将位于选择分支的左端的结束步更改为跳转转移，可以创建上述程序。

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)

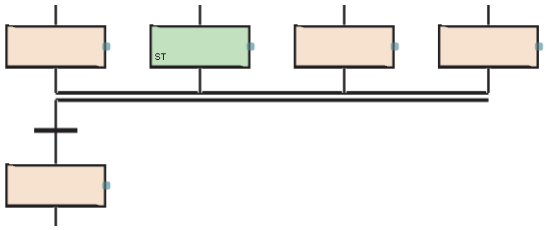
从1个步进行了分支的多个步全部同时进行激活转移。如果合并之前的步全部被激活，则在通用的转移条件成立时，激活将转移至下一个步。

项目	内容	
分支		步(1)处于激活时，转移条件(2)成立时，将步(3)与步(4)同时置为激活。 步(1)将变为非激活。但是，保持步[SC、SE、ST]的情况下，将按照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		步(1)与步(2)处于激活时，转移条件(3)、转移条件(4)成立时，将步(5)、步(6)置为激活。 通过将合并之前的步(5)及步(6)全部激活后，检查转移条件(7)，如果转移条件(7)成立，则将步(8)置为激活。 步(5)与步(6)将变为非激活。但是，保持步[SC、SE、ST]的情况下，将按照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，必定进行并联合并。

■注意事项

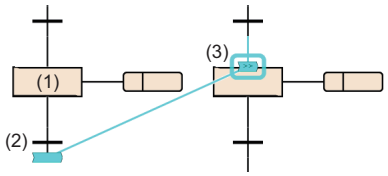
- 在并联合并中，合并的步中存在有保持中步[SC、SE、ST]的情况下，将进行下述动作。

项目	内容
线圈保持步[SC]	与非激活步一样，不转移到下一个步中。
动作保持步(无转移检查)[SE]	
动作保持步(有转移检查)[ST]	其它合并的步处于激活状态时，转移到下一个步中。 

- 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

通过转移条件成立，激活转移至同一块内的指定的步。

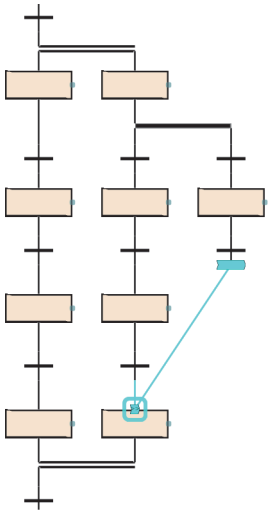


步(1)处于激活时，转移条件(2)成立时，将步(3)置为激活。
步(1)将变为非激活。但是，保持步[SC、SE、ST]的情况下，将按照属性保持线圈输出或动作输出。

- 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。不可以创建至并联分支内的不同分支的跳转转移及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



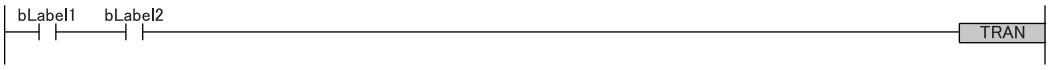
■注意事项

下述情况下，不可以作为跳转转移的指定目标进行指定。

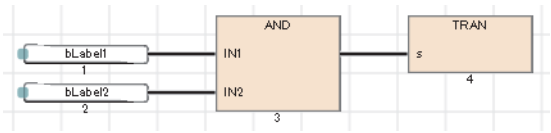
- 指定处于脱离并联转移位置的步的情况下
- 指定处于进入并联转移位置的步的情况下
- 指定先行转移条件的前一个步的情况下
- 指定本步的情况下

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。可以通过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，对由触点的梯形图及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。
	MELSAP-L(指令形式)	☞ 106页 MELSAP-L(指令形式)中的转移条件

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 • 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、FB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指令以外其它可使用的指令有关内容，请参阅下述章节。 ☞ 105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。
- 创建后的详细表示，可以从Zoom一览表中确认一览。(GX Works3 操作手册)

■可使用指令

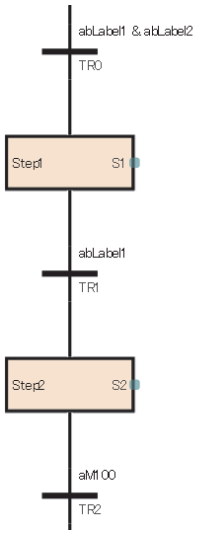
转移条件的程序中可使用的指令如下所示。

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*2
合并指令	ANB、ORB
	INV
	MEP、MEF
	EGP、EGF*1
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□
	LDED□、ANDED□、ORED□
字符串处理指令	LD$□、AND$□、OR$□
转移条件虚拟输出	TRAN*2

*1　在ST语言、FBD/LD语言的转移条件程序中，不可以使用EGP指令及EGF指令。
*2　在MELSAP-L(指令形式)的转移条件程序中，不可以使用LDPI、LDFI、ANDPI、ANDFI、ORPI、ORFI、TRAN。

MELSAP-L(指令形式)中的转移条件

MELSAP-L(指令形式)是使用文本在SFC图内记述转移条件的形式。



要点

从梯形图的详细表示切换为MELSAP-L(指令形式)的情况下，选择菜单的[显示]⇨[梯形图显示切换]⇨[MELSAP-L(指令格式)]。(GX Works3 操作手册)

在MELSAP-L(指令形式)中，以相当于触点的指令记述转移条件。转移条件的BOOL式为真(TRUE)的情况下，转移条件成立。

MELSAP-L(指令形式)采用下述形式记述程序。

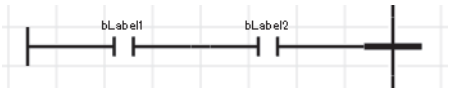
□：可使用的标签/软元件

项目	MELSAP-L(指令形式)	记述示例
常开触点(LD指令)	a□	aX0
常闭触点(LDI指令)	b□	bX1
上升沿常开触点(LDP指令)	p□	pM2
下降沿常开触点(LDF指令)	f□	fM3
运算结果取反(INV指令)	& INV	aM0 & INV
运算结果脉冲化指令(上升沿)(MEP指令)	& MEP	aM1 & MEP
运算结果脉冲化指令(下降沿)(MEF指令)	& MEF	aM2 & MEF
变址继电器运算结果脉冲化指令(上升沿)(EGP指令)	& EGP □	aM3 & EGP V0
变址继电器运算结果脉冲化指令(下降沿)(EGF指令)	& EGF □	aM4 & EGF V1
相当于触点的比较运算指令	与梯形图语言中的指令输入以同样方式记述。 可以使用下述比较运算指令。 BIN16位数据比较(带符号)：<，<=，<>，=，>，>= BIN32位数据比较(带符号)：D<，D<=，D<>，D=，D>，D>= 单精度实数比较：E<，E<=，E<>，E=，E>，E>= 双精度实数比较：ED<，ED<=，ED<>，ED=，ED>，ED>= BIN16位数据比较(无符号)：<_U，<=_U，<>_U，=_U，>_U，>=_U BIN32位数据比较(无符号)：D<_U，D<=_U，D<>_U，D=_U，D>_U，D>=_U 字符串比较：$<，$<=，$<>，$=，$>，$>=	< D10 D20
并联连接(OR)	|	aX0 | aM0
串联连接(AND)	&	aX0 & aM0
括弧	()	(aX0 | aM0) & aX1

同时使用&和|的情况下，优先运算&。希望更改优先顺序的情况下，使用( )。

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。对转移条件连接FBD/LD部件的触点。



不可以使用线圈部件、FB部件、函数部件、跳转部件/跳转标签部件、返回部件。

要点

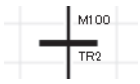
通过选择转移条件后，对菜单的[编辑]⇨[更改]⇨[直接显示转移条件]进行选择，可以将FBD/LD部件连接到转移条件的左侧中。(GX Works3 操作手册)

转移条件的标签/软元件

下一个步中转移激活的条件中，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，输入希望指定的位型标签及位软元件或BOOL值。(GX Works3 操作手册)

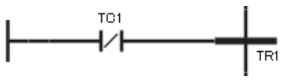
■注意事项

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

例

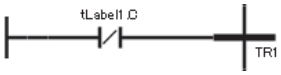
定时器软元件及定时器型标签的情况

触点(TS0)为ON时，转移条件成立。



触点(TS1)为OFF时，转移条件成立。

定时器型标签tLabel0的线圈为ON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。

8.3 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)［K4S□］*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。
	MOV(P)［BL□\K4S□］	
	DMOV(P)［K8S□］*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。
	DMOV(P)［BL□\K8S□］	
	BMOV(P)［K4S□］*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。
	BMOV(P)［BL□\K4S□］	
块启动	SET［BL□］	单独激活指定块激活，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□］	将指定块置为暂时停止。
块重启	RSTART［BL□］	对指定块的暂时停止进行解除，并从停止步开始重启执行。
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□］	
块切换	BRSET	指定SFC控制指令的对象块。

*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-R 编程手册(CPU模块用指令/通用FUN/通用FB篇)

■注意事项

- 在中断程序内请勿使用SFC控制指令。
- SFC控制指令只有在SM321(SFC程序启动/停止)为ON时才执行。

变址修饰

通过SFC控制指令指定的步继电器及SFC块软元件可以作为变址修饰进行指定。

软元件	变址修饰对象位置
S□Z□	步继电器
BL□\S□Z□	带块指定步继电器的步部分
BL□Z□\S□	带块指定步继电器的块部分
BL□Z□\S□Z□	带块指定步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□		R00CPU、R01CPU、R02CPU： 0～8191 上述以外的CPU模块： 0～16383 (最大值根据CPU参数的设置而定)
BL□\S□	BL□	R00CPU、R01CPU、R02CPU： 0～127 上述以外的CPU模块： 0～319
	S□	R00CPU、R01CPU、R02CPU： 0～127 上述以外的CPU模块： 0～511
BL□		R00CPU、R01CPU、R02CPU： 0～127 上述以外的CPU模块： 0～319

要点

关于变址修饰的详细内容，请参阅下述手册。

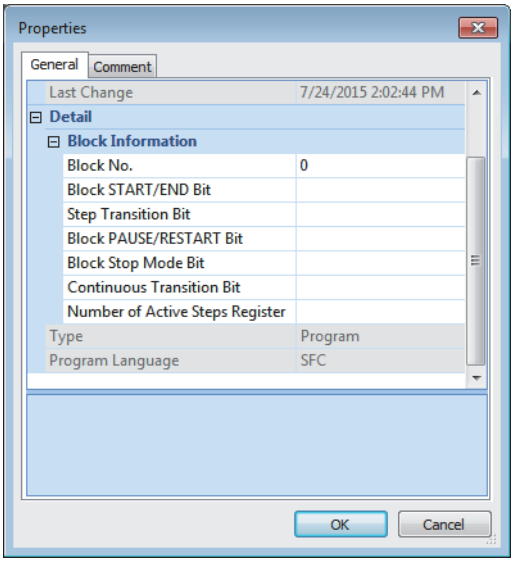
MELSEC iQ-R CPU模块用户手册(应用篇)

8.4 SFC用信息软元件

SFC用信息软元件是对块指示强制启动/结束与暂时停止/重启及对步的转移条件的成立与激活步数进行确认，或对转移条件的连续转移动作进行指示的软元件或标签。
对各块设置SFC用信息软元件。

[导航窗口]⇨[程序]⇨SFC程序文件⇨希望设置的块属性

画面显示



显示内容

项目	内容	可使用的数据	
		软元件	数据类型(标签)
块启动结束位	设置对块的激活状态进行确认的软元件或标签。 另外，在设置的位为ON时可以启动块，OFF时可以结束块。	位：Y、M、L、F、V、B 字：D、W、RD的位指定	BOOL、BOOL的数组、INT的位指定、WORD的位指定
步转移位	设置对执行中的步的转移条件的成立进行确认的软元件或标签。 各步的动作输出执行后，至下一个步的转移条件成立时将ON。		
块停止重启位	设置使激活中的块暂时停止或重启的软元件或标签。 设置的位ON时块通过执行中步进行停止，OFF时将从停止了块的步重启执行。		
块停止模式位	对决定停止块的时机的软元件或标签进行设置。 设置的位为ON时在各步的转移后停止块，为OFF时立即停止所有步。		
连续转移位	设置在转移条件成立时决定连续转移动作的软元件或标签。 设置的位ON时将变为有连续转移，在同一扫描内执行下一个步的动作输出。OFF时将变为无连续转移，在1个扫描中逐步执行。		
活动步数寄存器	设置对块的当前激活中的步数进行存储的软元件或标签。	D、W、R、ZR、RD	INT、WORD

除全局软元件及局部软元件以外，也可以对SFC用信息软元件指定全局标签或局部标签。不可以间接指定、位指定、变址修饰(Z、LZ)。

要点

仅在使用SFC用信息软元件的情况下，需要设置SFC用信息软元件。不使用的情况下，无需设置SFC用信息软元件。

块启动结束位

块启动结束位是对块的激活状态进行确认的软元件或标签。

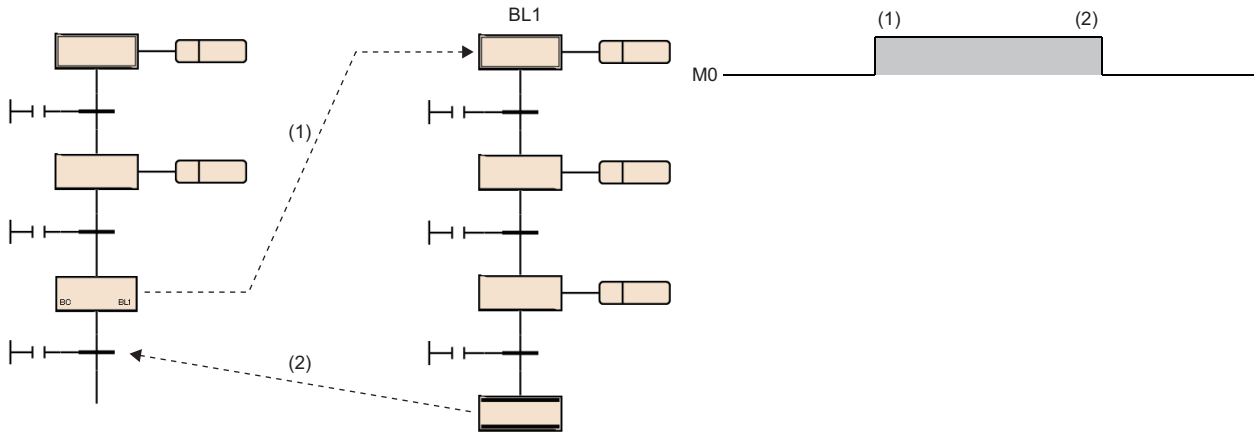
另外，在设置的位为ON时可以启动块，OFF时可以结束块。

无块启动程序的情况下，通过工程工具也可对块的启动/结束进行控制，因此块单位中的调试及试运行中可以使用。

- 设置的块启动时，块启动结束位将自动ON。设置的块处于激活中时，块启动结束位将保持ON状态不变。
- 设置的块变为非激活状态时，块启动结束位将自动OFF。设置的块为非激活状态时，块启动结束位将保持OFF状态不变。

例

在块1(BL1)的块启动结束位中指定了M0的情况



(1) 启动块1(BL1)，M0变为ON。

(2) 块1(BL1)变为非激活状态，M1变为OFF。

- 在设置的块为非激活时，如果将块启动结束位置为ON，则单独启动设置的块。
- 在设置的块处于激活中时，如果将块启动结束位置为OFF，则结束设置的块。

块启动结束位的ON/OFF也可通过工程工具的测试操作进行。(GX Works3 操作手册)

将块启动结束位置为OFF，并将设置的块置为非激活状态的情况下，进行下述处理。

- 停止设置的块的执行，执行的步的输出也全部OFF。但是，通过SET指令变为ON的软元件不变为OFF。
- 在设置的块内通过块启动步启动了其它块的情况下，设置的块将结束，但是启动目标的块保持激活状态不变继续运行处理。

要点

通过从工程工具的查看对BL□或BL□\S□的当前值进行更改，也可以对块进行启动/结束或对步进行激活/非激活。

通过菜单的[调试]⇨[SFC步控制]，也可以对选择的步进行激活/非激活。(GX Works3 操作手册)

■注意事项

• 将设置的块置为非激活之后的重启动作如下所示。

设置的块		内容
块0	在CPU参数的SFC设置中，启动条件设置为“自动起动块0”的情况	结束步处理后，从初始步开始重启。
	在CPU参数的SFC设置中，启动条件设置为“不自动起动块0”的情况	结束步处理后，将设置的块置为非激活状态，如果再次对设置的块发出了启动请求则从初始步开始重启。
块0以外		

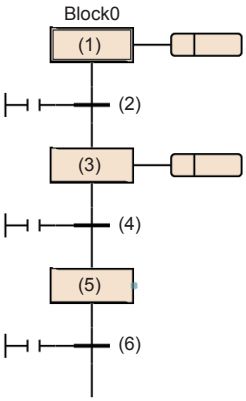
• 在SFC程序结束时，SFC用信息软元件中设置的所有块启动结束位将变为OFF。但是，继续启动设置时仅允许继续启动的情况下SFC程序启动时，所有块启动结束位将恢复。

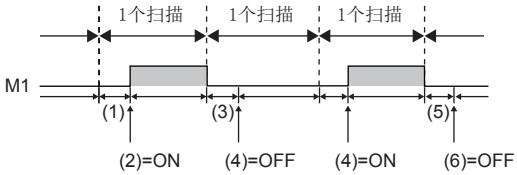
步转移位

对执行中的步的转移条件的成立进行确认的软元件或标签。
各步的动作输出执行后，至下一个步的转移条件成立时将ON。
变为ON的步转移位在执行再次指定的块的处理时，将自动OFF。

例

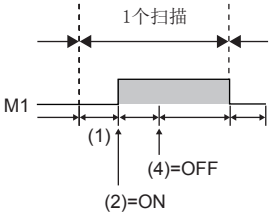
在Block0的步转移位中指定了M1的情况





在步(1)的执行后转移条件(2)成立时，执行其它块期间M1将变为ON。
下一个扫描的Block0的处理时M1将变为OFF。
在步(3)的执行后，转移条件(4)不成立的情况下，M1将保持OFF状态不变。
转移条件(4)成立时，执行其它块期间M1将变为ON。
在步(5)的执行后，转移条件(6)不成立的情况下，M1将保持OFF状态不变。

将连续转移位置为ON设置了“有连续转移”的情况下，转移条件成立后下一个步的动作输出执行中及执行多个步后转移条件未成立时，步转移位也将保持ON状态不变，在执行下一个扫描的指定块时将变为OFF。



在步(1)的执行后转移条件(2)成立时，M1将变为ON。
转移条件(4)不成立的情况下，M1也将保持ON状态不变。
下一个扫描的Block0的处理时M1将变为OFF。

在块内存在多个激活步的情况下，只要其中一个转移条件成立，在该时点步转移位将变为ON。

■注意事项

- 如果执行结束步，块的步转移位将变为ON。接着，步转移位将保持ON状态不变，直至该块再次激活为止。
- 在SFC程序启动时及SFC程序结束时，步转移位不变为OFF。

块停止重启位

对激活中的块执行暂时停止或重启的软元件或标签。
设置的位ON时块通过执行中步进行停止，OFF时将从停止了块的步重启执行。

设置	内容
OFF→ON	如果进行OFF→ON，指定块以执行中的步进行停止。
ON→OFF	如果进行ON→OFF，指定块将从停止的步的动作输出开始重启执行。 • 在动作保持状态下变为停止的动作保持步(无转移检查)[SE]或动作保持步(有转移检查)[ST]，在动作保持状态下重启执行。 • 对于线圈保持步[SC]，通过线圈输出OFF的设置(SM325 = OFF)停止的情况下将变为非激活，因此无法重启保持状态。通过线圈输出保持的设置(SM325 = ON)停止的情况下，将维持保持状态，因此重启后也将保持保持状态不变。

- 通过块启动步启动了其它块的情况下，如果将块停止重启位置为ON则指定的块将停止，但是启动目标的其它块将保持激活状态不变，继续运行处理。启动目标的块也同时停止的情况下，启动目标的块停止重启位也将ON。
- 将非激活的块中设置的块停止重启位置为ON的情况下，在非激活状态中不进行动作，在块变为激活状态的时刻立即变为停止状态。
- 强制结束了指定块的情况下，块停止重启位的状态将被保持不变。在停止中强制结束，不对块停止重启位的状态进行更改的情况下，再启动时立即变为停止状态。

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/非保持的组合决定。(☞ 122页 块停止重启时的动作)

■注意事项

- 在SFC程序启动时及SFC程序结束时，块停止重启位不变为OFF。

块停止模式位

决定停止块的时机的软元件或标签。
设置的位为ON时在各步的转移后停止块，为OFF时立即停止所有步。

设置	内容
OFF时(立即停止)	发出停止请求时，立即变为停止状态。
ON时(转移后停止)	发出停止请求后，执行中的步的转移条件成立，转移时将停止。 不执行转移后的步的动作输出。 在块内存在多个激活步时，将从转移成立的步开始依次停止。 与块停止模式位的设置无关，停止请求后，立即停止保持中的步。

块停止/重启时的动作，根据SM325(块停止时的输出模式设置)与块停止模式位的设置、步的保持/非保持的组合决定。
(☞ 122页 块停止重启时的动作)

■注意事项

- 在SFC程序启动时及SFC程序结束时，块停止模式位不变为OFF。

连续转移位

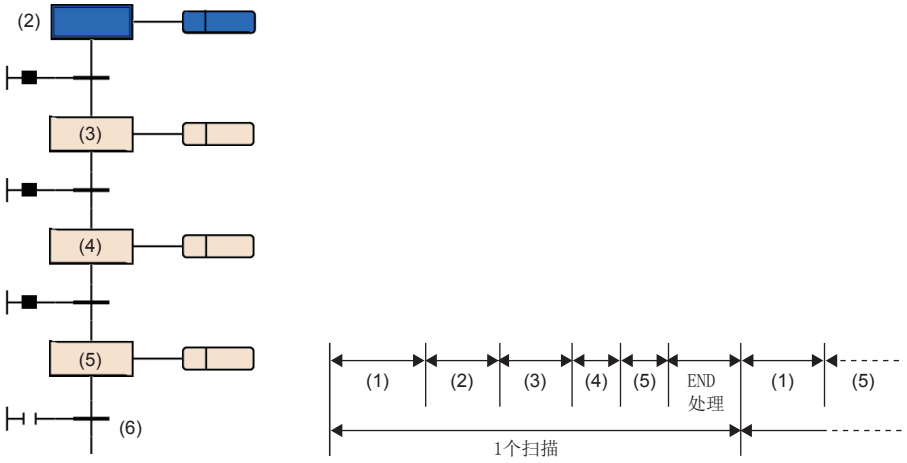
转移条件成立时，决定连续转移的动作的软元件或标签。

设置的位ON时将变为有连续转移，在同一扫描内执行下一个步的动作输出。OFF时将变为无连续转移，在1个扫描中逐步执行。

设置	内容
OFF时(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时(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。

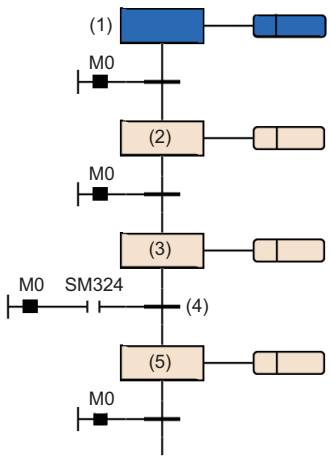
例

有指定SFC用信息软元件的连续转移位的情况



扫描	内容
第1个扫描	顺控程序(1)的执行后，连续执行SFC程序的步(2)～步(5)。
第2个扫描转移	顺控程序(1)的执行后，在转移条件(6)成立之前执行步(5)的动作输出。

- 对连续转移位进行了设置的情况下，与SM323(所有块连续转移的有无)的ON/OFF无关，设置的位软元件为OFF时变为无连续转移的动作，ON时变为有连续转移的动作。未设置连续转移位的情况下，SM323为OFF时变为无连续转移的动作，ON时变为有连续转移的动作。(☞ 129页 有/无连续转移的动作)
- SM324(连续转移阻止标志)在执行SFC程序时系统将自动ON，但是连续转移中时将变为OFF。通过在转移条件中将SM324以AND条件使用，可以禁止连续转移。

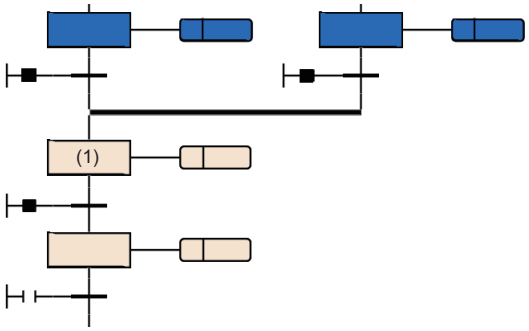


M0为ON时，1个扫描中从步(1)到步(3)变为连续转移。
通过将SM324作为AND条件附加至转移条件(4)中，步(3)的执行后的转移条件(4)将变为不成立。
在下一个扫描中，执行步(3)后SM324变为ON，因此在该扫描内转移到步(5)中。

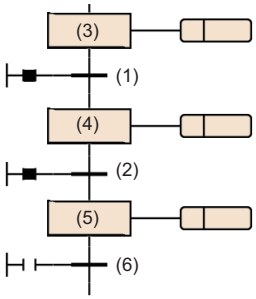
■注意事项

- 设置为有连续转移时，从转移条件成立开始优先于其它处理进行转移目标步的动作输出执行，因此可以缩短节拍时间。但是，在该情况下，其它块及顺控程序的动作有可能变慢。
- 在SFC程序启动时及SFC程序结束时，连续转移位不变为OFF。
- 通过跳转转移及选择合并，激活从多个步转移至1个步的情况下，1个步的动作输出有可能在1个扫描中执行2次。

有连续转移的情况下，在1个扫描中执行2次步(1)。



- 有连续转移的设置中，在步之后的转移条件成立的情况下，在1个扫描内进行步的启动及结束。在这种情况下，由于不执行END处理，因此通过动作输出内的OUT指令进行的线圈输出的输入输出刷新不被反映，在其它程序中无法检测出线圈的ON。例如，输出(Y)的情况下，在END处理未执行的时刻中输出(Y)不被输出，在其它程序中无法检测出输出(Y)的ON。因此，也无法检测出步继电器的ON。为了反映OUT指令的输入输出刷新，应创建程序，使得1个步可在多个扫描中执行。



转移条件(1)与转移条件(2)成立的情况下，在1个扫描内执行下述操作。
- 执行步(3)的动作输出。
- 由于转移条件(1)成立，将步(3)的动作输出置为OFF。
- 步(3)变为非激活，步(4)变为激活。
- 由于有连续转移，执行步(4)的动作输出。
- 由于转移条件(2)成立，将步(4)的动作输出置为OFF。
- 步(4)变为非激活，步(5)变为激活。
- 由于有连续转移，执行步(5)的动作输出。
- 由于转移条件(6)不成立，步(5)的动作输出不变为OFF。

- 在使用跳转转移进行循环的程序时，应置为无连续转移，或置为在执行中环路内的转移条件全部不成立。有连续转移且在执行中环路内的转移条件全部成立时，则在1个扫描内将变为无限循环。

活动步数寄存器

对块的当前激活中的步数进行存储的软元件或标签。

活动步数寄存器中存储的激活步数包括下述步。

- 普通的激活步
- 保持中的线圈保持步[SC]
- 保持中的动作保持步(有转移检查)[ST]
- 保持中的动作保持步(无转移检查)[SE]
- 停止中的步

■注意事项

- 在块结束时，活动步数寄存器将变为0。
- 在SFC程序结束时，活动步数寄存器不变为0，而是在SFC程序启动时变为0。

8.5 SFC设置

在CPU参数及SFC块设置内，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。
	启动条件设置	在SFC程序启动时，设置成是在自动启动块0后激活，还是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	在块停止时，将线圈输出置为OFF或保持不变。

要点

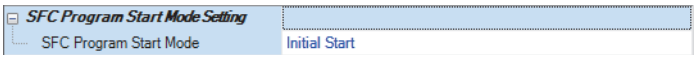
使用SFC程序的情况下，应预先确保步继电器(S)的点数。(步继电器(S)的默认点数为0点。)
在[CPU参数]⇨[存储器/软元件设置]⇨[软元件/标签存储器区域详细设置]⇨[软元件设置]中，在1024点单位内设置步继电器(S)的点数。(☞ 79页 规格)

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程序起动模式设置]

画面显示



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。(☞ 121页 启动条件设置)
继续启动	保持上次停止时的激活状态不变的状况下进行启动。

根据SFC程序启动模式设置及SM322(SFC程序的启动状态)的状态组合，决定是进行初始启动还是继续启动。

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置为OFF→ON	初始启动		继续启动	初始启动
(2)	将电源置为OFF→ON			继续启动/初始启动*4	
(3)	将SM321置为ON→OFF或RUN→STOP后将电源置为OFF→ON			继续启动	
(4)	复位→RUN			继续启动/初始启动*4	
(5)	将SM321置为ON→OFF或RUN→STOP后进行复位→RUN			继续启动	
(6)	STOP→RUN	继续启动*3			
(7)	STOP→程序写入→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，且在程序的写入前后无更改的情况下将继续启动。
*3 动作输出的ON/OFF，将按照参数设置的“STOP→RUN时的输出模式”的设置进行。
*4 根据时机将变为禁止继续启动状态，且有可能进行初始启动。

■注意事项

- 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，在进行继续启动的基础上需要预先保持的标签及软元件，应置为锁存设置。
- 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。希望继续输出的情况下，应将希望继续的标签及软元件置为锁存设置。此外，STOP→RUN时的输出的ON/OFF动作，将按照CPU参数设置的“STOP→RUN时的输出模式设置”的设置进行。(MELSEC iQ-R CPU模块用户手册(应用篇))
- 电源OFF时或复位时，智能功能模块将被初始化。继续启动的情况下，至智能功能模块的初始程序，建议创建至常时激活状态的块或顺控程序上。
- 电源OFF时或复位时，标签及软元件也被清除。SFC用信息软元件设置时，仅进行了锁存设置的情况下保持值。
- 电源OFF后或复位后的继续启动，根据时机有可能无法继续启动。在继续启动的设置时进行了初始启动的情况下，事件履历中禁止继续启动的事件将被存储。希望确实进行继续启动的情况下，应将SM321置为ON→OFF或RUN→STOP后再将电源置为OFF或进行复位。

启动条件设置

在SFC程序启动时，设置成是在自动启动块0后激活，还是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

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也与其它块一样根据SFC控制指令的SET指令(块启动)及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不变，直至再次有启动请求为止。

启动条件设置，在希望根据产品类型等指定SFC程序启动时的启动块的情况下使用。

“自动起动块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

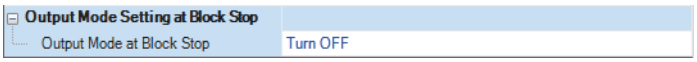
- 设置为“不自动起动块0”的情况下，执行SFC程序时通过顺控程序执行SET指令(块启动)，或将SFC用信息软元件中设置的块启动结束位置为ON。
- 设置为“自动起动块0”的情况下，必须创建块0。

块停止时的输出模式设置

在块停止时，将线圈输出置为OFF或保持不变。

[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	将线圈输出保持在停止之前的状态。

- 已设置的内容在电源ON时及复位时或STOP→RUN时被反映到SM325(块停止时的输出模式)的初始值中，SFC程序动作时按照SM325的设置进行。CPU参数的设置将被忽略。

■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/非保持的组合决定。

块停止/重启时的动作一览如下所示。

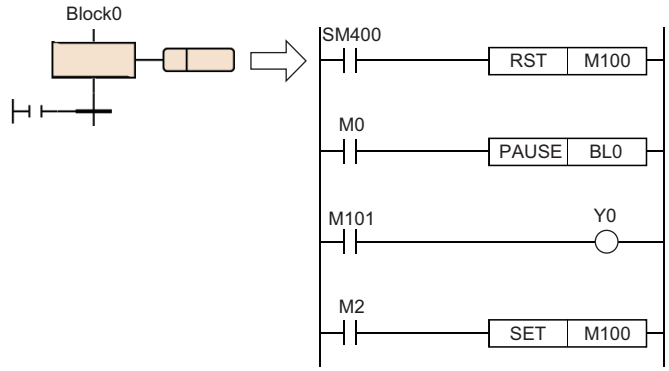
<table>
<tr><th rowspan="3">块停止时的输出模式的设置</th><th rowspan="3">块停止模式位的设置</th><th colspan="4">动作</th></tr>
<tr><th rowspan="2">保持中以外的激活中步
(也包括转移条件不成立的SC、SE、ST)</th><th colspan="3">保持中步</th></tr>
<tr><th>线圈保持步[SC]</th><th>动作保持步(无转移检查)[SE]</th><th>动作保持步(有转移检查)[ST]</th></tr>
<tr><td rowspan="2">SM325=OFF
(线圈输出OFF)</td><td>OFF或无设置
(立即停止)</td><td>有停止请求之后，将动作输出的线圈输出置为OFF后进行停止。状态保持激活不变。</td><td rowspan="2">有停止请求之后，将动作输出的线圈输出置为OFF后变为非激活状态。</td><td colspan="2" rowspan="2">有停止请求之后将动作输出的线圈输出置为OFF后进行停止。状态保持激活不变。</td></tr>
<tr><td>ON
(转移后停止)</td><td>转移成立后，进行步的结束处理，同时转移目标步将变为激活状态，在执行动作输出之前将停止。</td></tr>
<tr><td rowspan="2">SM325=ON
(线圈输出保持)</td><td>OFF或无设置
(立即停止)</td><td>有停止请求之后，在保持动作输出的线圈输出的状态下进行停止。状态保持激活不变。</td><td colspan="3" rowspan="2">有停止请求之后，在保持动作输出的线圈输出的状态下进行停止。状态保持激活不变。</td></tr>
<tr><td>ON
(转移后停止)</td><td>在转移成立之前与普通的动作相同。转移成立后，进行步的结束处理，同时转移目标步将变为激活状态，在执行动作输出之前将停止。</td></tr>
<tr><td colspan="2">重启时</td><td>返回普通的动作。</td><td>线圈输出OFF时： 变为非激活状态，因此禁止重启。
线圈输出保持时： 保持中的状态下直接重启。</td><td>在保持状态下重启动作输出的执行。</td><td>在保持状态下，重启动作输出后，检查转移条件。</td></tr>
</table>

■注意事项

- 使用LD指令(块激活检查)等指定的块为停止中的块的情况下，变为ON。此外，即使使用LD指令(步激活检查)等指定的步为停止中的步，也变为ON。
- 在将SFC用信息软元件的停止重启位置为ON的状态下进行块启动时，在初始步变为激活状态之前将停止。此外，对非激活块执行了SET指令(步启动)的情况下，在指定步变为激活状态之前将停止。
- SM325(块停止时的输出模式)为ON时(线圈输出保持)，可以在保持线圈输出不变的状态下进行停止。在停止中即使将SM325置为ON→OFF线圈输出的状态也不变化，发生块的重启请求时，在保持状态下直接进行重启。
- 在SM325为ON时停止了块的情况下，保持状态的线圈保持步[SC]在重启后也维持保持状态，但是步的动作不重启。将线圈保持步[SC]置为非激活时，应执行RST指令(步结束)。
- 在动作输出内即使对该块有停止请求，当前执行中的步也将执行到最后为止，随后才执行停止请求。因此，在执行中步内，块停止模式位为OFF时(立即停止)即使开始停止请求也不停止。此外，之后在相同步内块停止模式位为ON时(转移后停止)进行了切换的情况下，通过转移后停止模式发出停止请求。

例

M100为停止时模式位，且M101为块停止重启位的情况

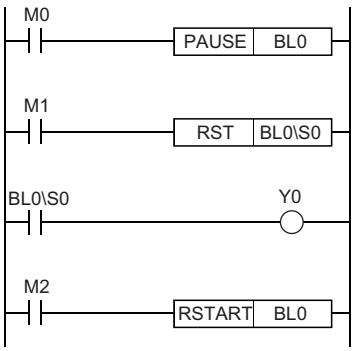


执行上述动作输出时，如果将M0置为ON则执行PAUSE指令，Block0的块停止重启位(M101)将变为ON，但由于执行至动作输出的最后，Y0将变为ON状态。
此外，M2为ON时，即使在PAUSE指令的执行后停止时模式位也将变为ON，在执行了全部动作输出后，转移后停止模式发出停止请求。

- 在块停止中执行RST指令(步结束)时，指定的步继电器将变为OFF。但是，工程工具的监视画面将保持激活状态不变，重启了块时将变为非激活状态。即使在SM325=ON(块停止时的线圈输出保持)且停止中执行也一样，但是线圈输出不变为OFF。
- 即使SET指令(步启动)处于块停止中，也会立即被执行，且指定的步继电器变为ON，工程工具的监视画面也会变为激活状态。但是，不执行动作输出，直至块被重启为止。

例

使用RST指令(步结束)时的块的停止重启的情况



(1) 如果将M0置为ON，则块0将停止。
(2) 如果将M1置为ON，并向步No.0发出结束请求，步继电器的BL0\S0变为OFF，但在工程工具的监视上，步No.0仍保持激活中不变。
(3) 由于BL0\S0变为OFF，Y0也变为OFF。
(4) 如果在M0、M1处于OFF的状态下将M2置为ON，则重启块0，结束步No.0。

- 停止时模式位为ON(转移后停止模式)时，即使存在转移后停止等待状态的步的状态下将停止时模式位置为OFF，也将保持转移后停止等待状态不变。为从此状态对转移后停止等待状态进行解除并立即停止，重启块后，需要在停止模式位为OFF的状态下再次开始停止请求。
- 在转移后停止模式中步的转移目标为结束步的情况下，将执行结束步的处理，因此不变为停止状态。
- 对有停止请求进行确认时，通过工程工具的块一览表显示监视，或对块停止重启位中设置的位进行监视。但是，无法通过工程工具的监视确认步是否处于停止中或停止等待动作中。
- 在转移成立之前，通过将块停止重启位置为OFF或执行RSTART指令，可以对转移后停止状态进行解除。在已停止的步与停止等待的步同时存在的状态下开始了重启请求时，已停止的步将开始动作，停止等待步将直接继续进行动作。停止请求被解除。

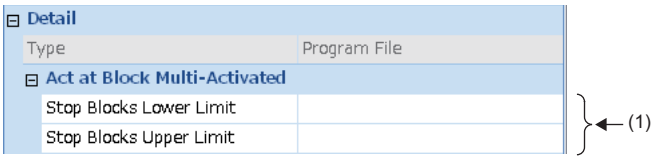
SFC块设置

块冗余启动时的运行设置

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，在希望停止CPU模块的运算的情况下设置。对设置范围设置希望停止的块范围。

[导航窗口]⇨[程序]⇨希望设置的SFC程序文件的属性

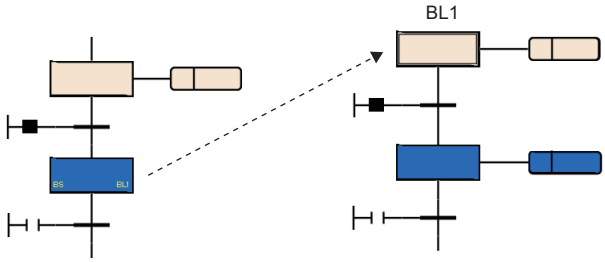
画面显示



(1) 设置希望停止的块范围。

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，保持转移条件成立的状态不变保持待机，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。 如果变为转移等待状态，之前的步将非激活且输出变为OFF，不执行动作输出的运算。
有停止的块范围的设置	停止	变为出错状态。



■注意事项

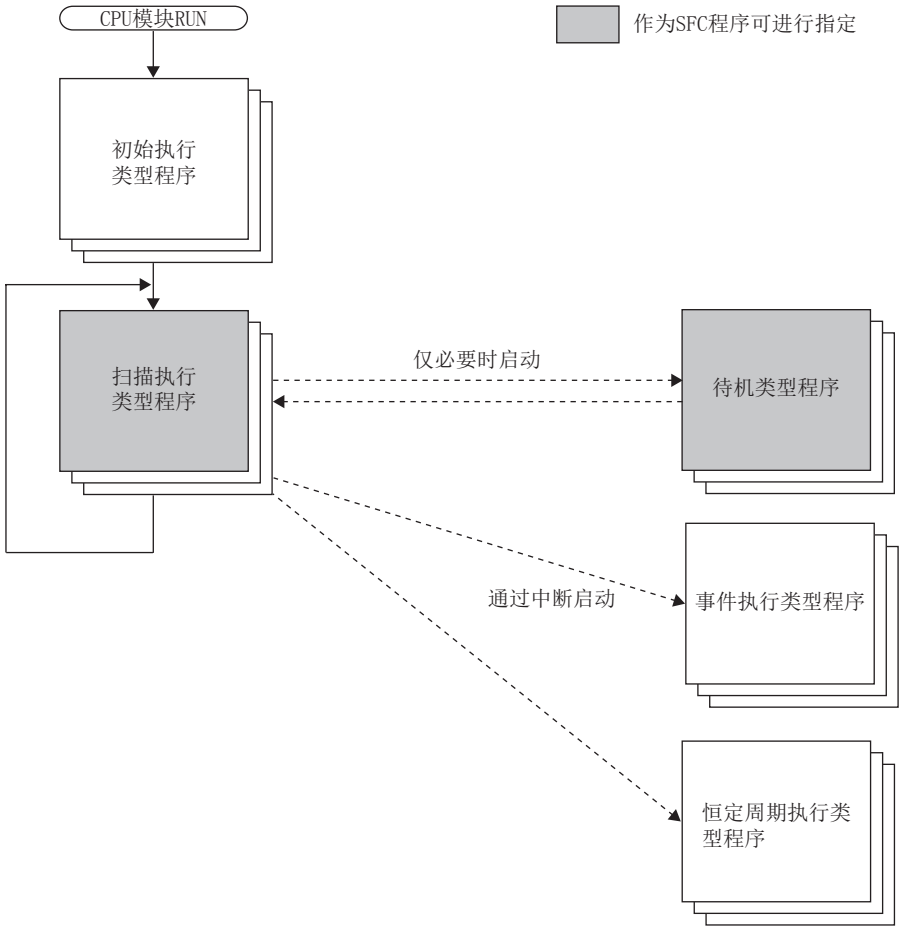
- 对已处于激活中的块，执行了SFC控制指令的SET指令(块启动)时，将忽略启动请求直接继续运行SFC程序的处理。
- 试图转移至激活中的块启动步的情况下，块启动步的启动将被忽略。不会再次从初始步开始执行。

8.6 SFC程序的执行顺序

整个程序的处理

可指定的执行类型

SFC程序的执行类型的指定可否如下所示。



执行类型	指定可否	备注
初始执行类型程序	×	—
扫描执行类型程序	○	SFC程序时仅1个可以执行
待机类型程序	○	通过PSCAN(P)指令指定SFC程序，则可更改为扫描执行类型
事件执行类型程序	×	—
恒定周期执行类型程序	×	—

■注意事项

不存在扫描执行类型的SFC程序(仅待机类型程序)的情况下，请勿对SFC程序执行SFC控制指令及监视。

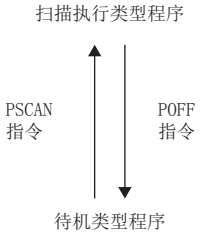
通过指令进行执行类型的更改

通过使用程序控制用指令，可以更改程序的执行类型。

对于程序控制用指令，SFC程序的指定可否如下所示。

指令符号	指定可否	备注
PSCAN(P)	○*1	将指定的SFC程序的执行类型更改为扫描执行类型。 在已存在有扫描执行类型的SFC程序时，指定了其它SFC程序后执行的情况下，将变为出错状态。
PSTOP(P)	×	对SFC程序执行的情况下，将变为出错状态。
POFF(P)	○*1	指定的SFC程序在下一个扫描中执行所有块的结束处理，在其后的下一个扫描中将执行类型更改为待机类型。

*1　在过程CPU(冗余模式)中不可指定。



■注意事项

- 从CPU模块进行文件读取/文件写入，及使用数据记录功能时请勿执行PSCAN(P)指令。如果执行PSCAN(P)指令，扫描时间可能会延长数100ms。
- 在指定继续启动时，将与上次动作的SFC程序不同的SFC程序通过PSCAN(P)指令进行了动作的情况下，指定的SFC程序将进行初始启动。在这种情况下，事件履历中将保存“不可继续启动SFC程序”(事件代码： 0430)。

SFC程序的处理顺序

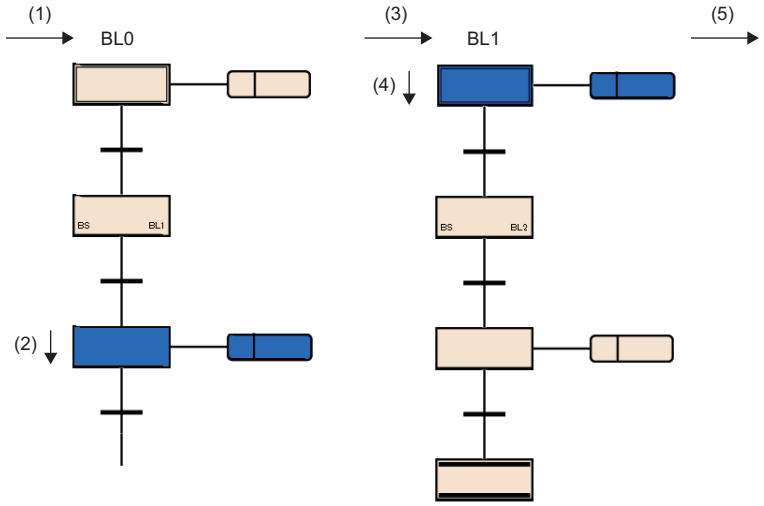
各块的执行顺序

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。

存在多个块的SFC程序的情况下，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。

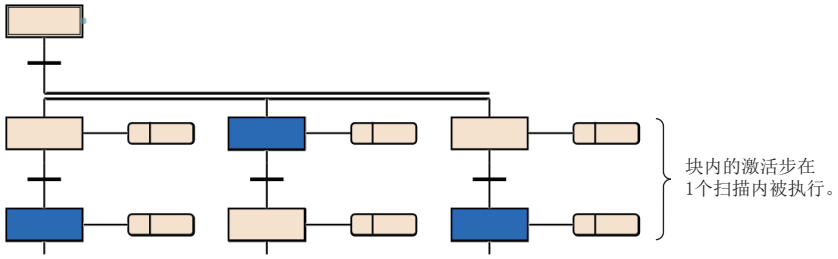
(5) 执行下一个块的处理。

SFC设置的启动条件设置中指定为块0的自动启动的情况下，只可以自动启动块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。(☞ 121页 启动条件设置)

此外，关于块的结束、停止、重启的请求，将在块内的执行处理之前被处理。

各步的执行顺序

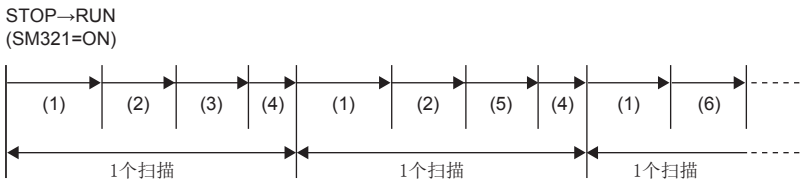
通过SFC程序，在1个扫描内处理所有激活步的动作输出。



各步的动作输出结束时，检查至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的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。上次执行的步将变为非激活状态，动作输出将变为非执行状态。

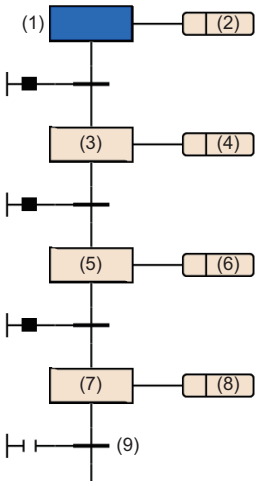
即使转移条件成立，将步的属性设为线圈保持步[SC]时，也不变为非激活状态而是按照属性处理。(☞ 86页 线圈保持步[SC])



(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

例

无指定SFC用信息软元件的连续转移位的情况

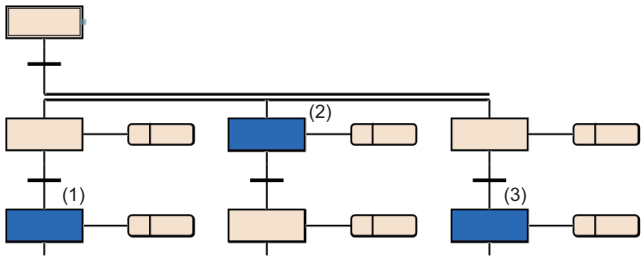


扫描	内容
第1个扫描	激活步(1)，执行动作输出(2)。
第2个扫描	激活步(3)，执行动作输出(4)。
第3个扫描	激活步(5)，执行动作输出(6)。
第4个扫描	激活步(7)，执行动作输出(8)。
第5个扫描及其以后	在转移条件(9)成立之前激活步(7)，执行动作输出(8)。

■注意事项

- 在首次执行时转移条件已成立的步的情况下，由于在1个扫描中结束步，因此线圈输出等的I/O刷新不被反映，在其它程序中将无法检测出线圈输出的ON。为了反映I/O刷新，应创建程序，使得1个步可在多个扫描中执行。
- 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之SFC程序。

(1)、(2)、(3)的动作输出的执行顺序将变为不定。



有/无连续转移的动作

在SFC程序的转移条件中，存在“有连续转移”及“无连续转移”的动作。

连续转移有/无的设置根据SFC用信息软元件的连续转移位的设置及SM323(所有块连续转移的有无)决定。

连续转移位	SM323	内容	
无设置	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。
OFF	ON/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	ON/OFF	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。

要点

通过设置为“有连续转移”，可以缩短节拍时间。因此，可以消除从转移条件成立开始到转移目标步的动作输出执行为止的等待时间。

但是，设置为“有连续转移”时，其它块及顺控程序的动作有可能变慢。

8.7 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
- 通过指令进行的启动及停止

通过CPU参数进行的自动启动

将CPU参数的“起动条件设置”设置为“自动起动”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。(☞121页 启动条件设置)

通过特殊继电器(SM321)进行的启动及停止

在执行SFC程序时，SM321(SFC程序的启动/停止)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，可以再次执行结束的SFC程序。

要点

通过对CPU参数的“SFC程序起动模式设置”进行设置，可以继续启动SFC程序。(☞119页 SFC程序启动模式设置)

通过指令进行的启动及停止

通过程序控制用指令，可以启动或停止SFC程序。(☞126页 通过指令进行执行类型的更改)

- 如果执行PSCAN指令，可以启动待机类型的SFC程序。执行类型将变为扫描执行类型。
- 如果执行POFF指令，将执行中的SFC程序的输出置为OFF后，可以进行停止。执行类型将变为待机类型。

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	参照目标
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。	☞121页 启动条件设置
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。	控制的顺序明确时有效。	☞88页 块启动步(有结束检查)[BC] ☞89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它顺控程序，通过SFC控制指令对指定的块进行启动。 • 从指定的块的初始步开始执行的情况下，使用SET [BL□]指令(块启动)。 • 从指定的块的指定步开始执行的情况下，使用SET [S□/BL□\S□]指令(步启动)。	检测出异常时，开始出错恢复处理块的启动、执行中断处理时有效。	☞108页 SFC控制指令
通过SFC用信息软元件进行的启动	通过将各块中设置的“块起动结束位”作为SFC用信息软元件置为ON，启动指定块。	也可通过外部设备进行启动，因此在块单位的调试、试运行时有效。	☞111页 块启动结束位
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	参照目标
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。	在自动运行时循环停止中停止动作等时有效。	☞90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它顺控程序，通过RST [BL□]指令(块启动)结束指定的块，并置为非激活状态。(通过RST [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时也结束块。)	在与动作状态无关，通过紧急停止等中止处理时有效。	☞108页 SFC控制指令
通过SFC用信息软元件进行的结束	通过将各块中设置的“块起动结束位”作为SFC用信息软元件置为OFF，结束指定块。	也可通过外部设备进行结束，因此在块单位的调试、试运行时有效。	☞111页 块启动结束位
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

块的暂时停止及重启

块的停止方法

在SFC程序执行中停止指定的块的方法如下所示。

项目	停止方法	备注	参照目标
通过SFC控制指令进行的停止	从SFC程序的动作输出或其它顺控程序，通过PAUSE[BL□]指令(块停止)暂时停止执行指定的块。	检测出异常时，暂时停止机械，利用手动运行对异常位置进行修复时有效。	☞108页 SFC控制指令
通过SFC用信息软元件进行的停止	作为SFC用信息软元件，通过将各块中设置的“块停止再启动位”置为ON，停止指定块。	也可通过外部设备进行停止，因此在调试、试运行时的边确认边控制也有效。	☞114页 块停止重启位

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/非保持的组合决定。(☞122页 块停止重启时的动作)

块的重启方法

在SFC程序执行中重启暂时停止的块的处理之方法如下所示。

项目	重启方法	备注	参照目标
通过SFC控制指令进行的重启	从SFC程序的停止块以外的动作输出或其它顺控程序，通过RSTART[BL□]指令(块重启)对指定的块进行重启。	暂时停止中的手动控制完成，返回至自动运行等时有效。	☞108页 SFC控制指令
通过SFC用信息软元件进行的重启	通过将各块中设置的“块停止再启动位”作为SFC用信息软元件置为OFF，重启指定块。	也可通过外部设备进行启动，因此在块单位的调试、试运行时有效。	☞114页 块停止重启位

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/非保持的组合决定。(☞122页 块停止重启时的动作)

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	参照目标
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。	—	☞98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它顺控程序，通过SET [S□/BL□\S□]指令(步启动)对指定的步进行启动。	—	☞108页 SFC控制指令
通过工程工具进行的启动	• 通过将步继电器置为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。

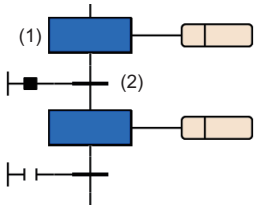
项目	结束方法	备注	参照目标
通过转移条件成立进行的结束	步的下一个转移条件成立时，将自动结束。	—	☞98页 转移条件
通过复位步[R]进行的结束	复位步[R]变为激活时，在属性指定目标中指定的步将结束。	在SFC程序的选择分支中转移至出错处理的步时等，结束保持步[SC、SE、ST]的情况下有效。	☞88页 复位步[R]
通过SFC控制指令进行的结束	从SFC程序的动作输出或其它顺控程序，通过RST [S□/BL□\S□]指令(步启动)对指定的步进行结束。	通过RST指令指定块的全部步变为非激活状态时，块也将结束。	☞108页 SFC控制指令
通过工程工具进行的结束	• 通过将步继电器置为OFF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



选择转移的情况

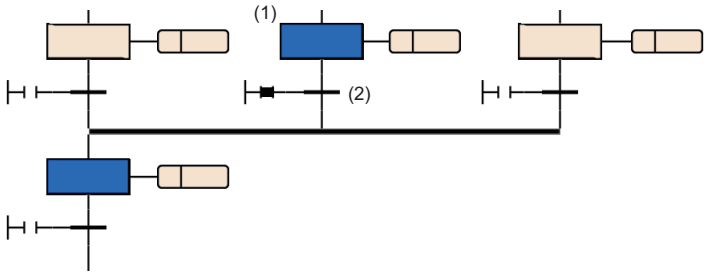
■选择分支

选择分支时，从左开始按顺序对转移条件进行检查，如果转移条件成立的分支的转移目标为激活步，将与串行转移的情况下相同。此时，即使在右分支中条件再次成立的情况下，也不检查该条件。

■选择合并

选择合并时的冗余启动的动作与串行转移的情况下相同。

转移条件(2)成立时，步(1)将变为非激活状态。

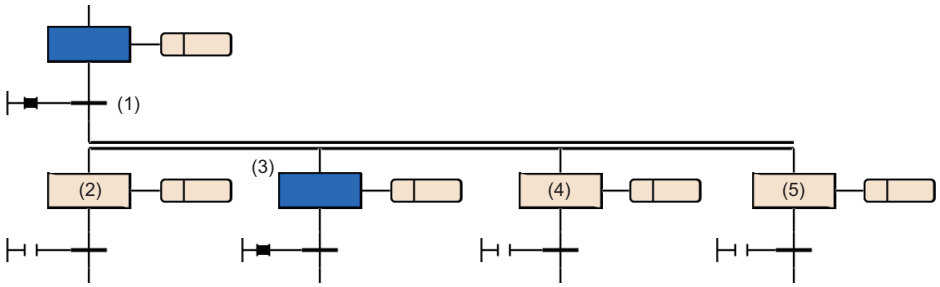


并联转移的情况

■并联分支

并联分支的情况下，在多个转移目标中只要有一个为激活状态，则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，在下一个扫描中步(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。保持步[SC、SE、ST]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，请参阅下述章节。
☞ 137页 SFC块RUN中写入
且使用SFC块RUN中写入的情况下，应确认CPU模块及工程工具的版本。(MELSEC iQ-R CPU模块用户手册(应用篇))

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

关于所有程序内使用的各软元件或标签，通过SM326(SFC的软元件·标签清除模式)的设置的动作如下所示。

SM326(SFC的软元件·标签清除模式)*1	内容
OFF	对除去下述的全部软元件及标签进行清除之后，执行SFC程序。 ・步继电器(S) ・文件寄存器(R/ZR)*2 ・锁存指定的标签
ON	对除去步继电器(S)的全部软元件及标签的值进行保持的状态下，执行SFC程序。

*1 SM326的设置，仅在写入至可编程控制器后，SFC程序存在时有效。此外，不仅SFC程序的写入，程序文件及参数文件的写入存在的情况下也有效。但是，仅通用软元件注释、软元件存储器、软元件初始值的写入时有效。
*2 即使未对文件寄存器(R/ZR)进行锁存设置，也不会变为SM326的清除对象。

■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、参数文件(CPU参数、系统参数等)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继续启动。(☞ 119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，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。之后，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中写入进行的程序更改

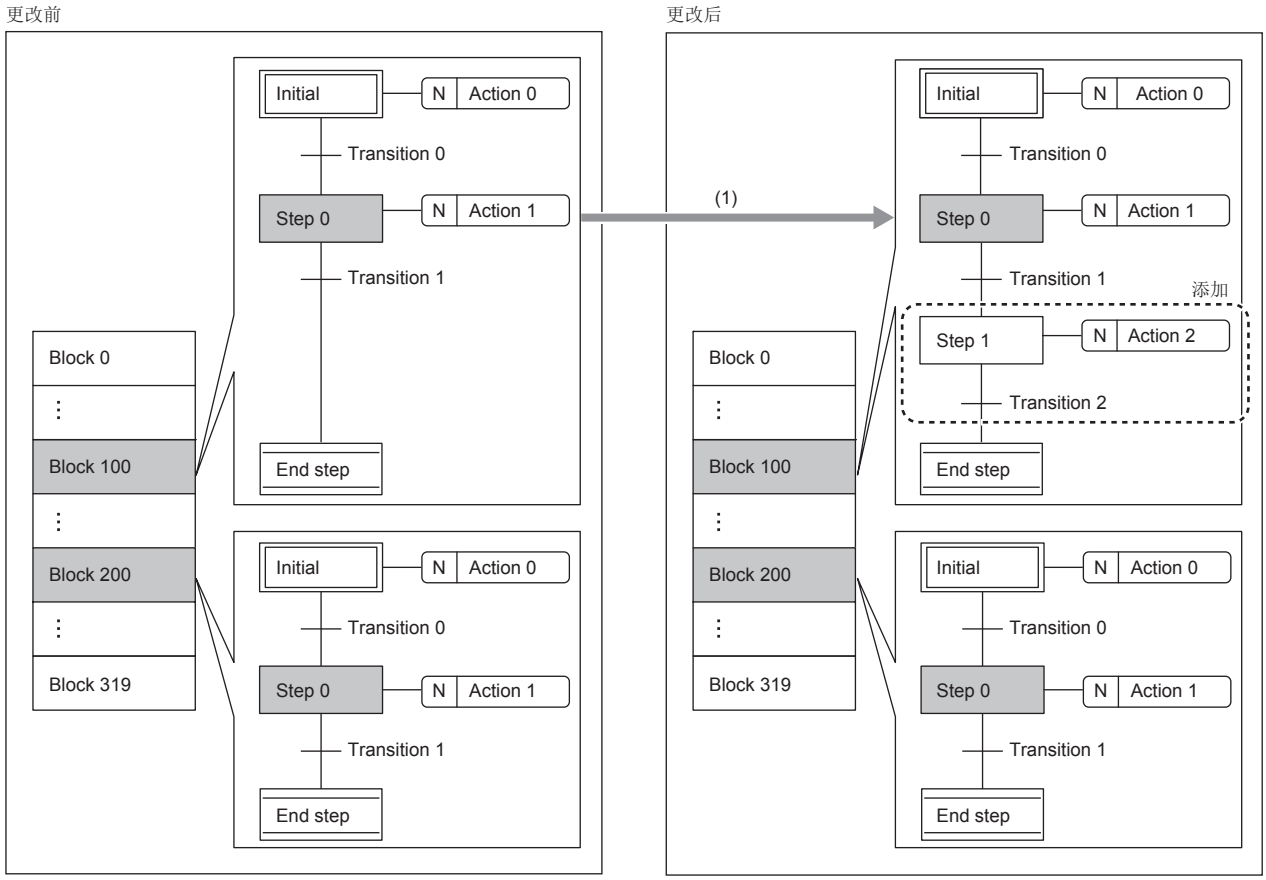
通过RUN中写入进行程序更改后，与CPU参数的SFC程序启动模式设置无关，必须继续启动。

SFC块RUN中写入

RnCPU RnENCPU ~~RnPCPU（过程）~~ ~~RnPCPU（冗余）~~ ~~RnPSFCPU（常规）~~ ~~RnPSFCPU（安全）~~ ~~RnSFCPU（常规）~~ ~~RnSFCPU（安全）~~

• 关于CPU模块及工程工具的版本，请参阅下述手册。(MELSEC iQ-R CPU模块用户手册(应用篇))

可以以块单位更改SFC程序。即使在RUN中也可维持激活状态不变，并能以块单位更改程序，因此可提升SFC程序调试及维护的效率。



(1)可更改激活中的SFC块的激活步以外的程序。

限制事项

使用SFC块RUN中写入的情况下，应确认CPU模块及工程工具的版本。
关于CPU模块及工程工具的版本，请参阅下述手册。
MELSEC iQ-R CPU模块用户手册(应用篇)

■程序的执行类型的执行可否

仅可通过扫描执行类型程序执行。(无法通过待机类型程序执行。)

■SM321(SFC程序的启动/停止)为OFF时的SFC块RUN中写入

SM321(SFC程序的启动/停止)为OFF时，与变为OFF前的对象块的激活状态无关，可执行RUN中写入。此外，在SM321=OFF时执行SFC块RUN中写入的情况下，不论下述的设置如何，必定会变为初始启动。

- CPU参数的SFC继续启动设置(☞ 119页 SFC程序启动模式设置)
- SM322(SFC程序的启动状态)

■更改的对象块

应在对块进行逐个更改后，再执行SFC块RUN中写入。在更改了多个SFC块的状态下，无法执行SFC块RUN中写入。

■块的更改/添加/删除

SFC块RUN中写入的块的更改/添加/删除如下所示。

操作	内容	
	对象块为非激活的情况	对象块为激活中的情况
块的更改	• 可更改CPU模块内的SFC程序中已存在的SFC块的程序。 • 可更改对象SFC块的SFC用信息软元件。	• 可更改CPU模块内的SFC程序中已存在的SFC块的程序。但是，不可进行激活步的删除、属性更改、步No.更改。 • 可更改对象SFC块的SFC用信息软元件。
块的添加	• 将新的SFC块添加至CPU模块内的SFC程序中。 • 可添加对象SFC块的SFC用信息软元件。	—
块的删除	• 可从CPU模块中的SFC程序删除指定的SFC块。 • 可删除对象SFC块的SFC用信息软元件。 • 在CPU模块的SFC程序中不存在对象块的情况下，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，激活的步维持激活状态。因此，在要实施步的删除、属性更改、步No.更改的情况下，应在将维持激活状态的步置为非激活后，再执行SFC块RUN中写入。

■程序改写的范围

在对象块中仅改写添加/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梯形图。此外，添加/更改对象的动作输出及转移条件内的梯形图中的上升沿、下降沿指令的前次执行信息被初始化。

■SFC块RUN中写入中程序的执行类型的更改可否

无法对RUN中写入执行中的程序文件更改程序控制指令(POFF/PSCAN指令)的执行类型。若执行，则指令将变为无处理。

■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-R CPU模块用户手册(应用篇))

■RUN中写入用确保步

为执行SFC块RUN中写入，CPU模块内需要有用于添加・更改容量的RUN中写入用确保步容量。执行SFC块RUN中写入时，若添加・更改容量超过RUN中写入用确保步容量，而程序存储器中有可用空间的情况下，可重新设置RUN中写入用确保步。

关于RUN中写入用确保步，请参阅下述手册。

📖MELSEC iQ-R CPU模块用户手册(应用篇)

■引导运行中的SFC块RUN中写入

在从SD存储卡进行引导运行的过程中，执行SFC块RUN中写入的情况下，也可更改引导源的SD存储卡内的对应文件。

■在SFC块RUN中写入中试图启动对象块或步的情况下的动作

在RUN中写入执行中试图启动SFC块RUN中写入的对象块或步的情况下，对象块或步不启动。各种类型的块或步的启动动作如下所示。

对象块或步的启动类型(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	在SFC块RUN中写入完成之前不启动。在SFC块RUN中写入完成之后自动启动。
块启动步(无结束检查)	• 在SFC块RUN中写入完成之前，不启动对象块，保持待机。即使步中附带的转移条件成立，也不转移至下一个步中。 • 在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)	• 在SFC块RUN中写入完成之前，不启动对象块，保持待机。 • 在SFC块RUN中写入完成后，启动对象块。块结束后，若附带的转移条件成立，则转移至下一个步中。
SFC控制指令(SET BL□，SET S□，SET BL□\S□指令)	不启动对象块。指令触点持续保持ON的情况下，在SFC块RUN中写入完成后启动对象块。
SFC用信息软元件(块启动结束位的启动)	即使块启动结束位为ON，也不启动对象块。块启动结束位为ON的情况下，在SFC块RUN中写入完成后启动对象块。(在SFC块RUN中写入完成之前，系统不启动对应块。)
通过工程工具进行的块启动*1	不启动对象块。忽略请求。(在SFC块RUN中写入完成之前，系统不启动对应块。)
转移条件成立时的步的启动(激活)	• 在SFC块RUN中写入完成之前，根据之前的转移条件的成立，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，转移条件成立的情况下，激活状态进行转移。

*1　显示监看窗口中的BL□、BL□\S□、软元件/缓冲存储器批量监视的启动、软元件→SFC步控制的启动。

■注意事项

- SFC块RUN中写入中(包含对程序存储器的传送)，请勿将电源置为OFF或复位。若执行，应执行写入至可编程控制器的操作。
- SFC块RUN中写入及通过工程工具的下述操作无法同时执行。
  - RUN中的梯形图块更改/写入至可编程控制器、SFC块RUN中写入
  - 写入至可编程控制器(软元件/局部软元件/全局软元件/全局标签/局部标签除外)
  - 存储器的初始化
- 删除控制中不需要的OUT指令(线圈)时，应在确认OUT指令为OFF后再行删除。若在不为OFF的情况下删除OUT指令，则输出将保持不变。
- 在更改梯形图内调用了子程序型FB时，调用的子程序型FB的FB定义内的上升沿、下降沿等前次执行信息不会被初始化。
- 使用SFC块RUN中写入的情况下，中断程序的启动作业可能需要等待。因此，通过使用了模块间同步中断(I44)、多CPU间同步中断(I45)的中断程序监视了中断程序执行时间的情况下(CPU参数的异常检测设置)，有可能检测出CPU模块中有出错。(MELSEC iQ-R CPU模块用户手册(应用篇))
- SFC块RUN中写入时，由于一部分区间为监视对象外，因此即使扫描时间超出扫描时间监视时间(WDT)设置所设置的时间，也有可能检测不出WDT时间超出。
- 在SFC块RUN中写入中通过执行的指令检测出出错的情况下，由于程序位置信息(步No.)会变为写入中的程序的步No.，因此不会变为写入完成后的程序的步No.。
- 在下述情况下，无法执行SFC块RUN中写入。
  - 对象的程序文件未登录到参数设置
  - SFC步数超出CPU参数的软元件设置所设置的步继电器点数
  - 将CPU模块置为STOP状态、指定程序或参数，在写入至可编程控制器时的STOP→RUN期间
  - 程序不可执行(出错代码：3204H)发生时
- 对于激活块的SFC块RUN中写入中使用SM329(SFC块RUN中写入执行中标志)或SD329(SFC块RUN中写入对象块No.)采取互锁等，以使激活步的转移条件不成立。SFC块RUN中写入中至对象块的转移条件成立的情况下，激活状态不转移，而保持待机。SFC块RUN中写入完成后，转移条件成立的情况下，转移至对象步。此外，已设置步转移位的情况下，待机中位不变为ON。

• 实施包含跨越激活步的步移动的SFC块RUN中写入的情况下，应按照下述步骤实施。

❶首先剪切要移动的步。

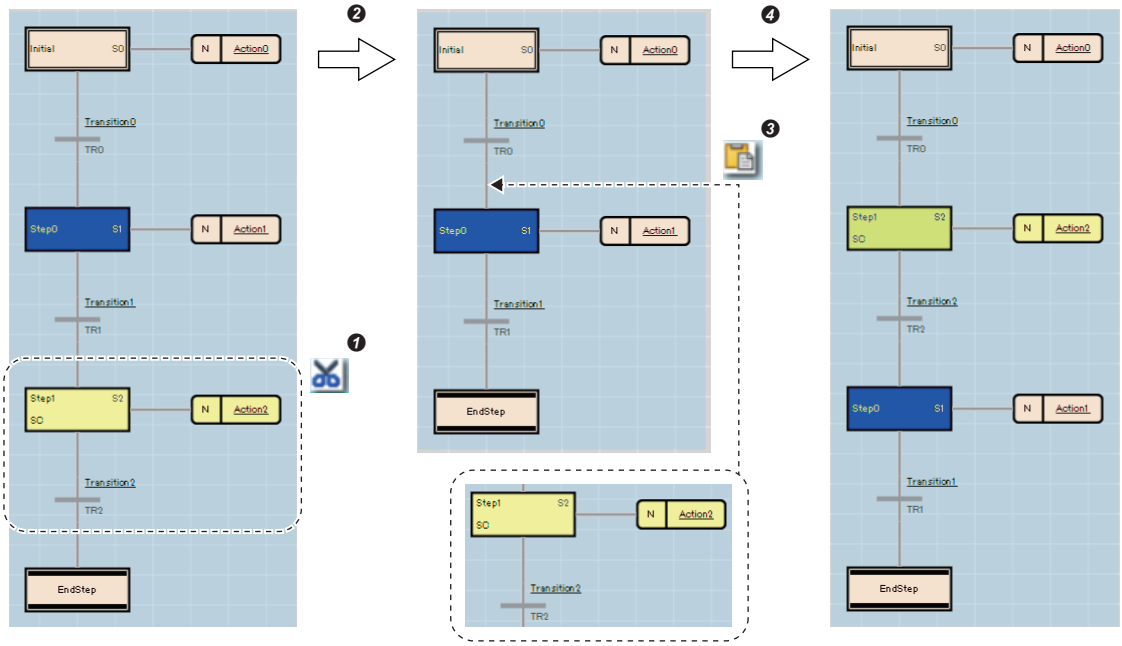
❷执行SFC块RUN中写入。

❸将剪切的步粘贴至目的位置。

❹再次执行SFC块RUN中写入。

省略上述❷，以❶→❸→❹的顺序实施的情况下，有可能无法执行SFC块RUN中写入。

编辑前　　编辑后



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
- SFC已激活步监视

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

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/高速)、长定时器	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/高速)、长累计定时器、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

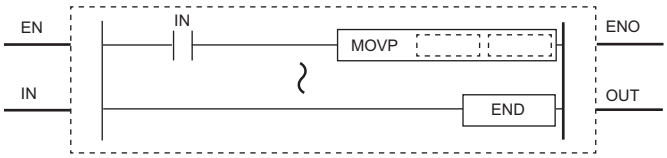
将“使用MC/MCR控制EN”选为“是”的情况下，在该FB处于执行中时，请勿使用MC/MCR指令。使用了MC/MCR指令的情况下，EN的控制可能无法正确动作。

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。

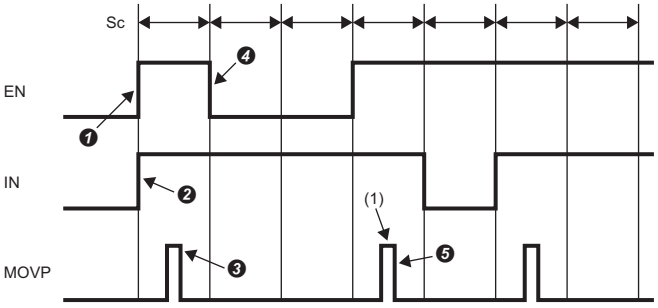
例

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

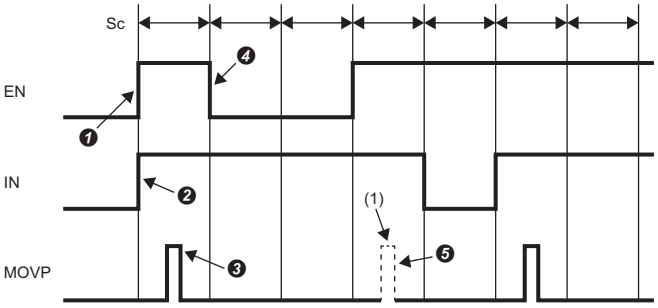
在EN变为ON时，条件触点若成立，则执行指令。(图中(1))



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。)

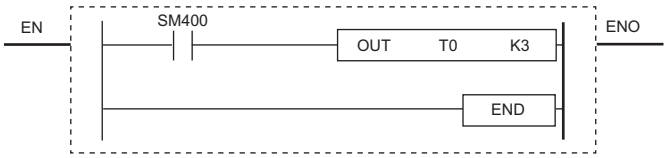
附

定时器(低速/高速)、长定时器的动作

定时器(低速/高速)、长定时器的动作如下所示。

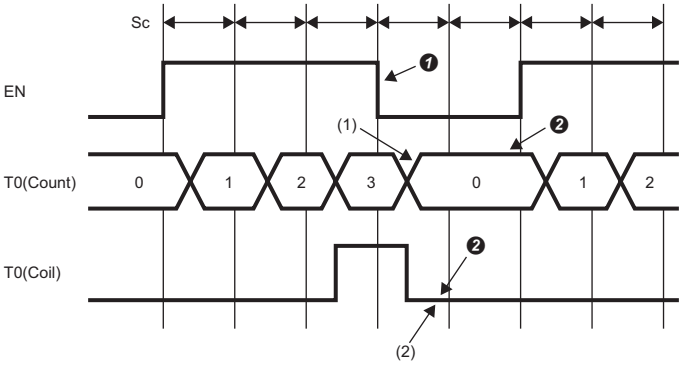
例

使用低速定时器的子程序型FB



■将“使用MC/MCR控制EN”选为“是”时

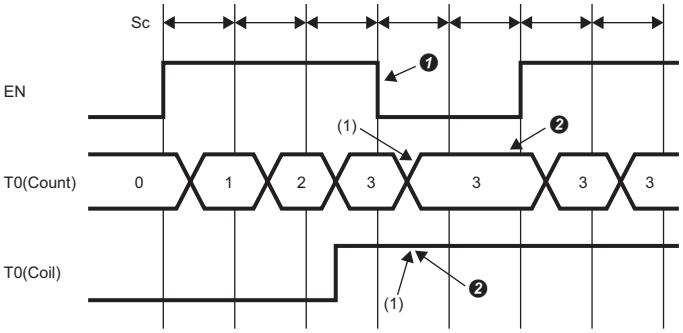
计数值变为0。(图中(1))此外，线圈变为OFF。(图中(2))



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))



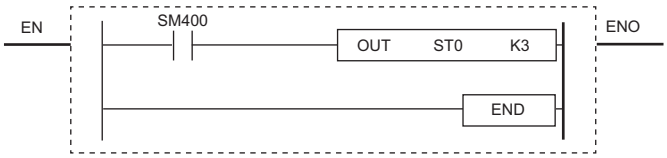
Sc：扫描
T0(Count)：T0(计数值)
T0(Coil)：T0(线圈)
❶将EN置为OFF。(用户操作)
❷保持值。(CPU模块动作)

累计定时器(低速/高速)、长累计定时器、计数器、长计数器的动作

累计定时器(低速/高速)、长累计定时器、计数器、长计数器的动作如下所示。

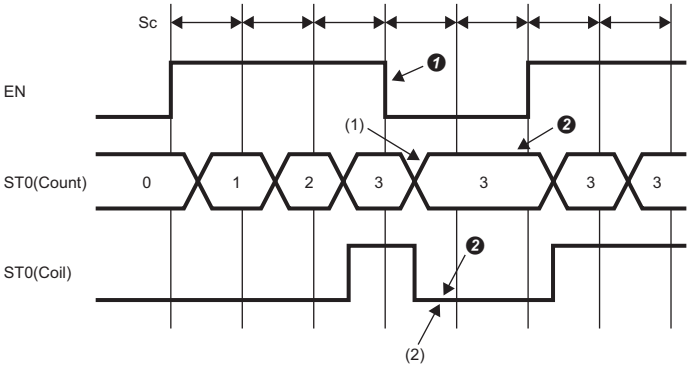
例

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

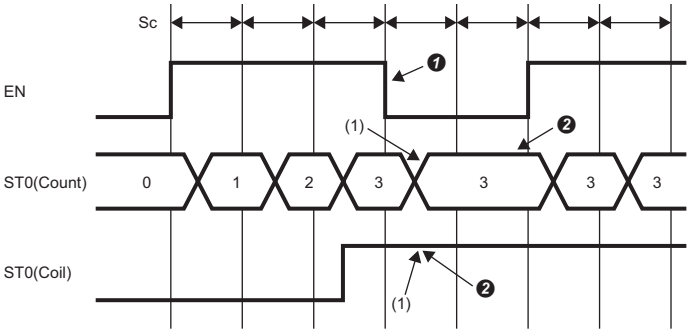
计数值保持现状。(图中(1))此外，线圈变为OFF。(图中(2))



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)

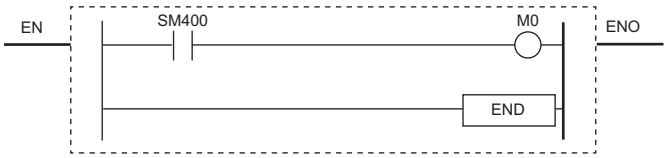
附

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。

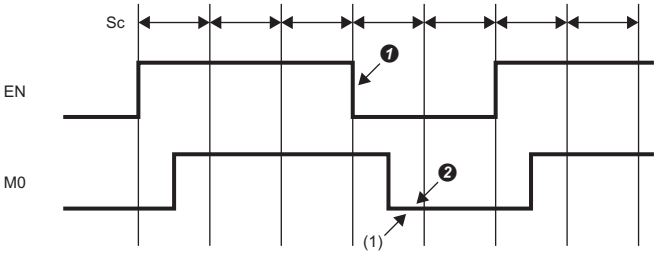
例

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

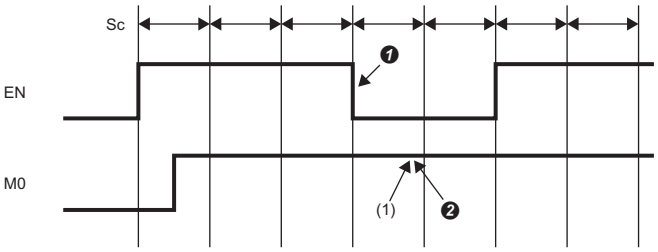
M0强制变为OFF。(图中(1))



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)

备忘录

索引

符号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W

X

Y

Z

索

修订记录

*本手册号在封底的左下角。

修订日期	*手册编号	修改内容
2014年8月	SH(NA)-081319CHN-A	第一版
2014年12月	SH(NA)-081319CHN-B	■第二版 部分修改
2015年2月	SH(NA)-081319CHN-C	■第三版 部分修改
2015年10月	SH(NA)-081319CHN-D	■第四版 部分修改
2016年6月	SH(NA)-081319CHN-E	■第五版 部分修改
2017年2月	SH(NA)-081319CHN-F	■第六版 部分修改
2017年7月	SH(NA)-081319CHN-G	■第七版 部分修改
2017年11月	SH(NA)-081319CHN-H	■第八版 部分修改
2018年6月	SH(NA)-081319CHN-I	■第九版 部分修改
2018年11月	SH(NA)-081319CHN-J	■第十版 部分修改

日语版手册编号：SH-081225-L

质保

使用之前请确认以下产品质保的详细说明。

1. 免费质保期限和免费质保范围

在免费质保期内使用本产品时如果出现任何属于三菱电机责任的故障或缺陷（以下称“ 故障 ”），则经销商或三菱电机服务公司将负责免费维修。

但是如果需要在国内现场或海外维修时，则要收取派遣工程师的费用。对于涉及到更换故障模块后的任何再试运转、维护或现场测试，三菱电机将不负任何责任。

[免费质保期限]

免费质保期限为自购买日或交货的一年内。

注意产品从三菱电机生产并出货之后，最长分销时间为 6 个月，生产后最长的免费质保期为 18 个月。维修零部件的免费质保期不得超过修理前的免费质保期。

[免费质保范围]

(1) 范围局限于按照使用手册、用户手册及产品上的警示标签规定的使用状态、使用方法和使用环境正常使用的情况下。

(2) 以下情况下，即使在免费质保期内，也要收取维修费用。

① 因不适当存储或搬运、用户过失或疏忽而引起的故障。因用户的硬件或软件设计而导致的故障。

② 因用户未经批准对产品进行改造而导致的故障等。

③ 对于装有三菱电机产品的用户设备，如果根据现有的法定安全措施或工业标准要求配备必需的功能或结构后本可以避免的故障。

④ 如果正确维护或更换了使用手册中指定的耗材（电池、背光灯、保险丝等）后本可以避免的故障。

⑤ 因火灾或异常电压等外部因素以及因地震、雷电、大风或水灾等不可抗力而导致的故障。

⑥ 根据从三菱电机出货时的科技标准还无法预知的原因而导致的故障。

⑦ 任何非三菱电机或用户责任而导致的故障。

2. 产品停产后的有偿维修期限

(1) 三菱电机在本产品停产后的 7 年内受理该产品的有偿维修。
停产的消息将以三菱电机技术公告等方式予以通告。

(2) 产品停产后，将不再提供产品（包括维修零件）。

3. 海外服务

在海外，维修由三菱电机在当地的海外 FA 中心受理。注意各个 FA 中心的维修条件可能会不同。

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. 产品规格的改变

目录、手册或技术文档中的规格如有改变，恕不另行通知。

商标

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

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

SH(NA)-081319CHN-J(1811)MEACH
MODEL: R-P-PS-C

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：021-23223030 传真：021-23223000
网址：http://cn.MitsubishiElectric.com/fa/zh/
技术支持热线 400-821-3030

扫描二维码,关注官方微博

扫描二维码,关注官方微信