

三菱可编程控制器

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生产的通用产品。

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、生产物责任），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。关于软元件也可以像标签一样使用。

备忘录

目录

| | |
|------------------|-----------|
| 安全注意事项 | 1 |
| 关于产品的应用 | 1 |
| 前言 | 2 |
| 关联手册 | 6 |
| 术语 | 7 |
| 第1章 概要 | 8 |
| 第2章 程序配置 | 10 |
| 2.1 程序块 | 11 |
| 第3章 程序部件 | 12 |
| 3.1 功能(FUN) | 13 |
| 3.2 功能块(FB) | 18 |
| 3.3 注意事项 | 28 |
| 3.4 使用安全程序的情况下 | 31 |
| 安全功能(安全FUN) | 31 |
| 安全功能块(安全FB) | 32 |
| 第4章 标签 | 33 |
| 4.1 类型 | 33 |
| 4.2 分类 | 34 |
| 4.3 数据类型 | 35 |
| 4.4 排列 | 38 |
| 4.5 结构体 | 41 |
| 4.6 常数 | 43 |
| 4.7 注意事项 | 44 |
| 4.8 使用安全程序的情况下 | 46 |
| 安全标签的类型 | 46 |
| 分类 | 47 |
| 数据类型 | 48 |
| 结构体 | 48 |
| 第5章 梯形图语言 | 49 |
| 5.1 配置 | 49 |
| 梯形图符号 | 49 |
| 程序执行顺序 | 50 |
| 5.2 直接ST | 51 |
| 5.3 声明/注释 | 53 |
| 第6章 ST语言 | 54 |
| 6.1 配置 | 55 |
| 分隔符 | 56 |
| 运算符 | 56 |
| 构文 | 57 |
| 常数 | 65 |
| 标签与软元件 | 66 |

| | |
|---------------------------|------------|
| 注释 | 67 |
| 第7章 FBD/LD语言 | 68 |
| 7.1 配置 | 68 |
| 部件 | 69 |
| 工作表 | 76 |
| 常数 | 76 |
| 标签与软元件 | 76 |
| 7.2 程序执行顺序 | 78 |
| 部件的执行顺序 | 78 |
| 第8章 SFC程序 | 79 |
| 8.1 规格 | 81 |
| 8.2 配置 | 82 |
| 块 | 83 |
| 步 | 84 |
| 动作输出 | 95 |
| 转移条件 | 98 |
| 8.3 SFC控制指令 | 106 |
| 8.4 SFC用信息软元件 | 108 |
| 8.5 SFC设置 | 116 |
| CPU参数 | 116 |
| SFC块设置 | 122 |
| 8.6 SFC程序的执行顺序 | 123 |
| 整个程序的处理 | 123 |
| SFC程序的处理顺序 | 125 |
| 8.7 SFC程序的执行 | 128 |
| SFC程序的启动及停止 | 128 |
| 块的启动及结束 | 129 |
| 块的暂时停止及重启 | 130 |
| 步的启动及结束(激活及非激活) | 131 |
| 步双重启动时的注意点 | 132 |
| 程序更改时的动作 | 133 |
| SFC程序的动作确认 | 134 |
| 索引 | 135 |
| 修订记录 | 138 |
| 质保 | 139 |
| 商标 | 140 |

关联手册

最新的e-Manual、EPUB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 | 提供形态 |
|---|--|-------------------------|
| MELSEC iQ-R编程手册(程序设计篇) [SH-081319CHN](本手册) | 记载梯形图、ST、FBD/LD、SFC的程序规格以及标签有关内容。 | e-Manual EPUB PDF |
| MELSEC iQ-R编程手册(指令/通用FUN/通用FB篇) [SH-081322CHN] | 记载CPU模块的指令、智能功能模块的专用指令、通用功能/通用功能块有关内容。 | e-Manual EPUB 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 EPUB PDF |

要点

e-Manual是使用专用工具可阅览的三菱电机FA电子书籍手册。

e-Manual有如下所示特点。

- 可以通过一次查找从多个手册中查找出希望搜索的信息(手册横向查找)
- 可以通过手册内的链接参阅其它手册
- 可以通过产品插图的各部件阅览希望了解的硬件规格
- 可以将经常浏览的信息登录到收藏夹中

术语

在本手册中，除非特别标明，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的总称产品名。(n表示版本。) |
| 一般/安全共享标签 | 是一般程序及安全程序中使用的标签。在安全程序及一般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。 <ul style="list-style-type: none"> 与对象设备相符合的协定的设置 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。 <ul style="list-style-type: none"> 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。 CPU模块的情况下，是指用于存储以太网功能的设置值、监视值等的的数据及多CPU功能的数据通信中使用的数据等的存储器。 |
| 程序部件 | 是各功能分开定义的程序单位。通过程序的部件化，可以将程序分级时的低位处理按处理内容及功能分为若干个单位，创建各单位的程序。 |
| 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 | 是将各模块固有定义的存储器(输入输出信号及缓冲存储器)以任意字符串表示的标签。可以从使用的模块由工程工具自动生成，作为全局标签使用。 |
| 标签 | 是以任意字符串表示软元件的标签。 |

此外，使用安全CPU的情况下，将使用下述术语进行说明。

| 术语 | 内容 |
|-------|---|
| 安全控制 | 执行安全程序及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。 |
| 一般CPU | 是执行一般控制的MELSEC iQ-R系列的各CPU模块(安全CPU以外的CPU模块)的总称。(与安全CPU进行区别时使用。) |
| 一般控制 | 执行一般程序及一般通信实施机械的控制。除安全可编程控制器以外仅保有一般控制。(与安全控制进行区别时使用。) |
| 一般通信 | 是除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)。 |
| 一般软元件 | 是CPU模块内部具有的除安全软元件以外的软元件(X、Y、M、D等)。仅在一般程序中使用。(与安全软元件进行区别时使用。) |
| 一般程序 | 是用于执行顺控程序控制的除安全程序以外的程序。(与安全程序进行区别时使用。) |

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。

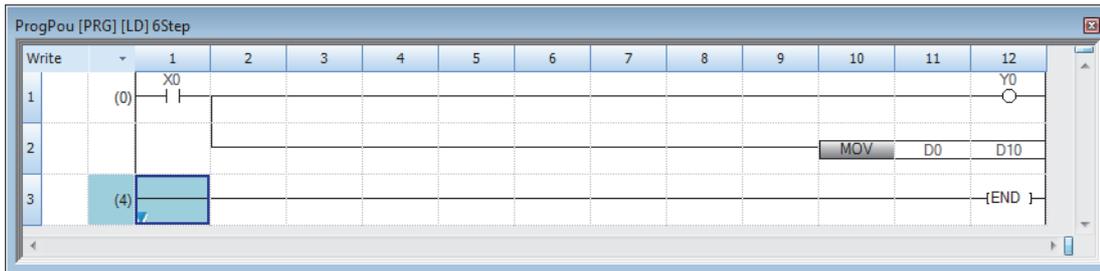
📖 GX Works3操作手册

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。 ST语言与梯形图语言相比，由于可以对难以记述的运算处理简洁而容易地进行记述，因此适用于进行复杂的算术运算及比较运算等的领域。此外，可以与C语言等一样，对通过条件语句进行的选择分支及，通过循环语句进行的重复等的构文的控制进行记述，因此可以简洁地进行易懂的程序编写。 |
| 功能块图表/梯形图语言(FBD/LD语言) | 是通过按照数据及信号的流向将进行特定处理的块、变量部件、常数部件接线，对程序进行记述的图表语言。 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，提高程序生产率。 |
| 顺序功能图(SFC程序) | 是将一系列的控制动作分割为多个步，置为能够清楚地表示各程序的执行顺序及执行条件的程序的记述形式。 |

n 梯形图语言



使用梯形图语言的情况下，请参阅下述章节。

📖 49页 梯形图语言

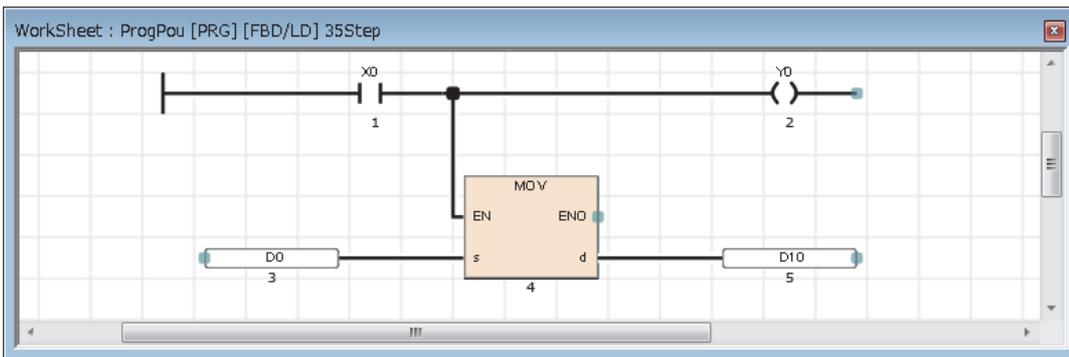
n ST语言

```
ProgPou1 [PRG] [ST] 13Step
1 IF X0 THEN
2   Y0 := TRUE ;
3   D0 := D10;
4 END_IF;
5
```

使用ST语言的情况下，请参阅下述章节。

📖 54页 ST语言

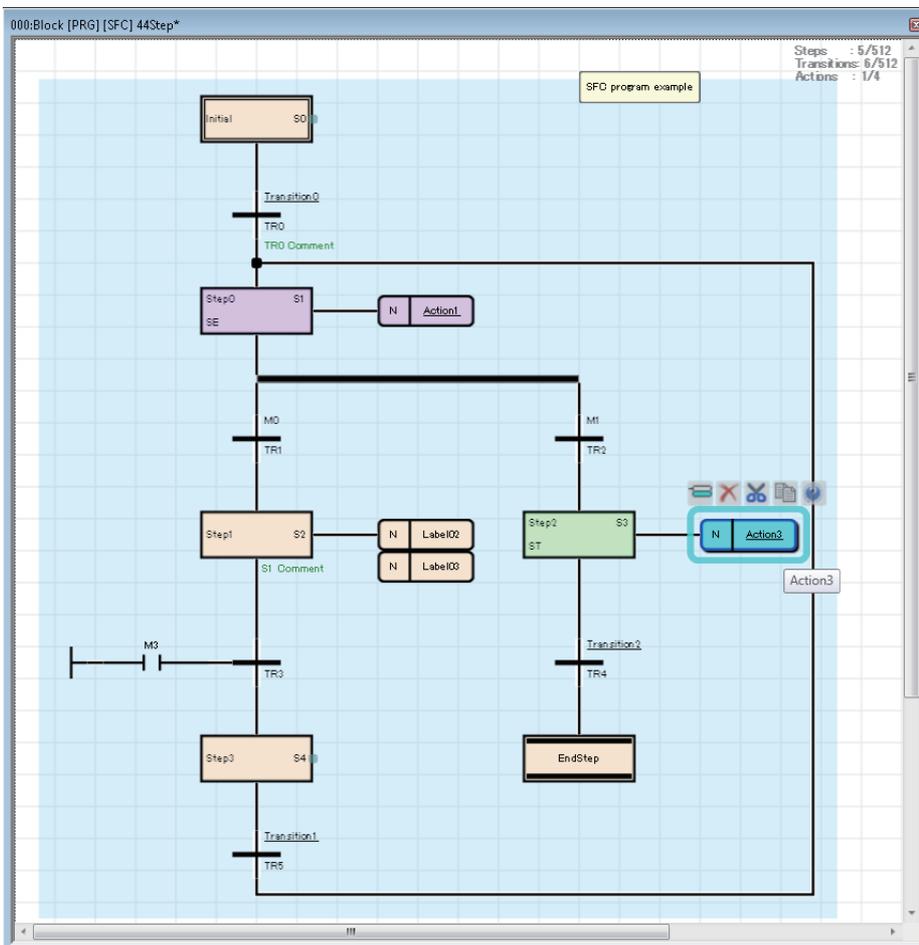
n FBD/LD语言



使用FBD/LD语言的情况下，请参阅下述章节。

☞ 68页 FBD/LD语言

n SFC程序



使用SFC程序的情况下，请参阅下述章节。

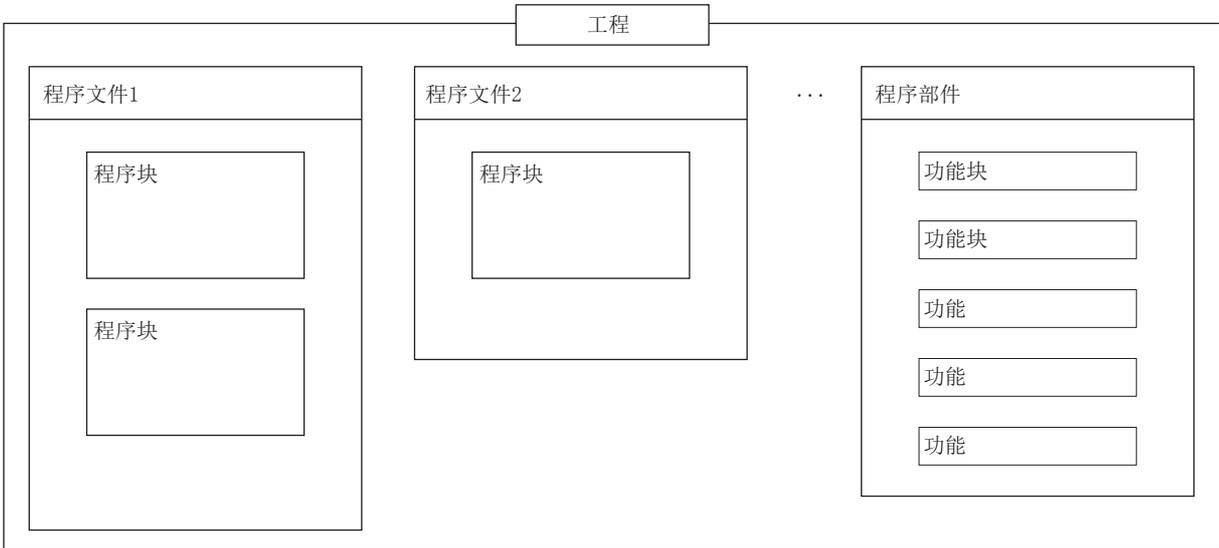
☞ 79页 SFC程序

要点

- 梯形图语言适用于具有顺控程序控制、逻辑梯形图知识及经验的用户。ST语言适用于具有C语言等编程知识及经验的用户。FBD/LD语言适用于与过程控制相关的用户。SFC程序适用于每当各机械的实际控制时分割程序，对作业的切换进行管理的情况。
- 通过在程序中使用标签，可以提高程序的可读性，将程序简单地转移至模块配置不同的系统中。

2 程序配置

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



关于程序部件，请参阅下述章节。

☞ 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。
程序文件由一个以上的程序块构成。
通过程序文件单位的操作，可以将程序的执行类型由恒定周期执行类型替换为待机类型，对是否将数据写入至CPU模块中进行更改。

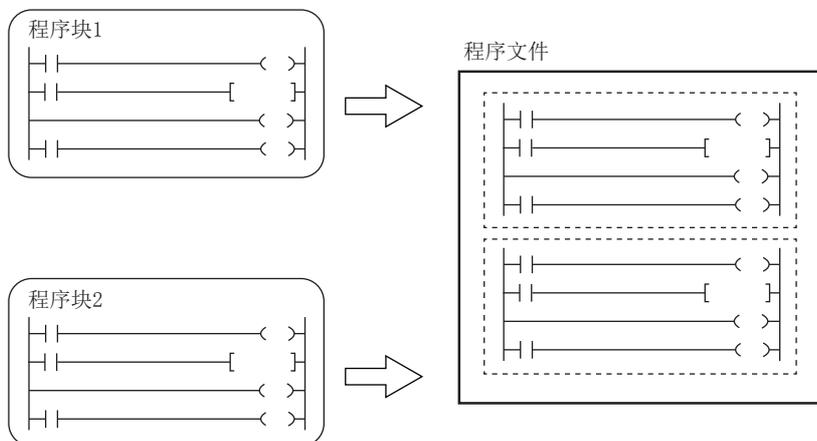
2.1 程序块

程序块将变为构成程序的单位。

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被执行。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)被执行。

如果对各功能及处理划分程序块，设计时确保方便地进行程序的顺序更改及更换。

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。



通过对各程序块分开创建主程序、子程序、中断程序，可以创建易懂的程序。^{*1}

| 类型 | 内容 |
|------|--|
| 主程序 | 是从程序步0到FEND指令为止的程序。 |
| 子程序 | 是从指针(P)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)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。

关于主程序、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)

- 子程序以及中断程序是在FEND指令以后进行创建。FEND指令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，应将成对的FOR指令与NEXT指令、MC指令与MCR指令在一个程序块中使用。
- 简单程序的情况下，在一个程序块内仅记述主程序，可以使其在CPU模块中执行。

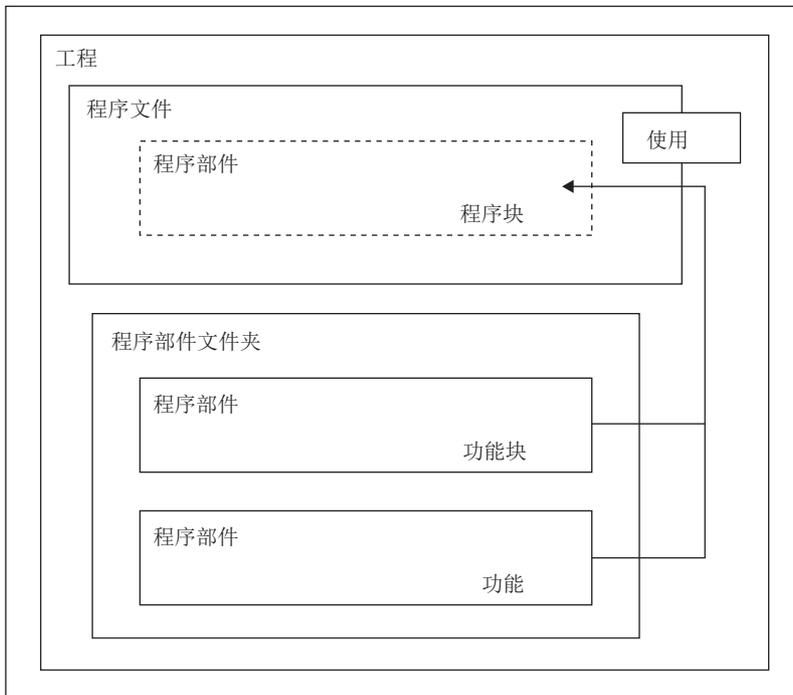
3 程序部件

程序部件有下述几种类型。

- 功能
- 功能块

在功能及功能块中，可以通过符合控制的程序语言(梯形图语言、ST语言、FBD/LD语言)对处理进行记述。

程序部件是从程序块中调用后执行。



要点

程序的部件化是指将程序阶层化时的低位处理按照各处理内容及功能分为若干单位，创建各单位的程序。

通过程序的部件化，可提高独立性，设计添加及更换更容易进行的程序。

如果进行部件化，更好的处理方法，有以下几种。

- 在程序中被循环记述处理
- 作为一个功能被分开处理

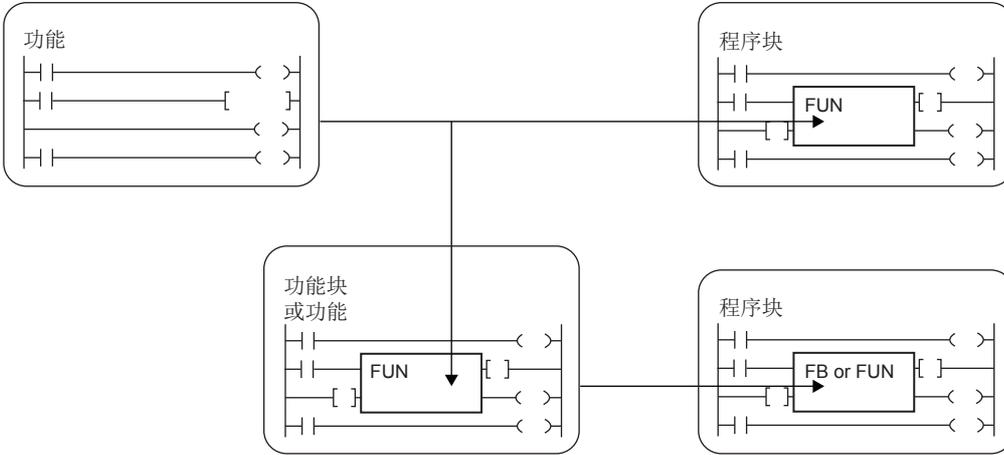
本项使用标签对各程序部件进行说明。

功能及功能块中的程序本体也可以使用软元件。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)

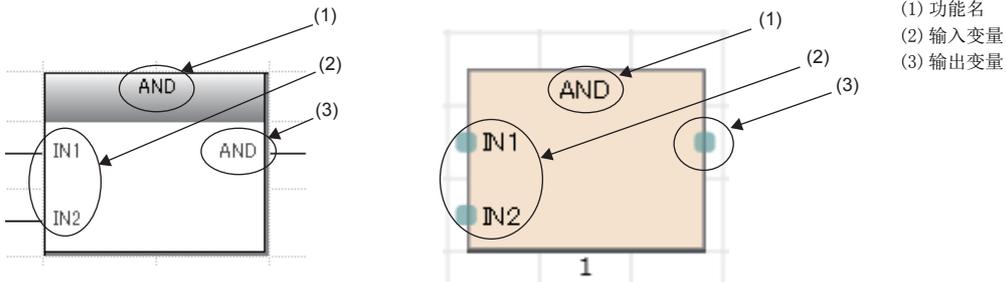
3.1 功能(FUN)

功能是在程序块、功能块以及其它的功能中使用的程序部件。
功能执行完成后将值交接至调用源。该值称为返回值。
功能对于同样的输入，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。



关于输入变量与输出变量

功能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。
梯形图语言的情况下 FBD/LD语言的情况下



不显示功能的返回值名称。

关于可设置的输入变量、输出变量的分类，请参阅下述章节。

☞ 34页 分类

要点 🔍

在功能中定义的变量在每次功能调用时被覆盖。
每次调用时希望保持变量值的情况下，应通过使用功能块或将输出变量保存为不同的变量等进行编程。

关于EN/ENO

通过在功能中附加EN(允许输入)、ENO(允许输出)，可以控制执行处理。

- EN设置功能的执行条件的布尔型变量。
- 带EN功能只在EN的执行条件为TRUE的情况下执行。
- ENO设置输出功能的执行结果的布尔型变量。

关于布尔型，请参阅下述章节。

☞ 35页 数据类型

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，不需要进行至ENO的输出标签的设置。
- 在通用功能中使用EN/ENO的情况下，带EN功能将变为“功能名_E”。

创建程序

创建功能程序的情况下，执行下述操作。

☞ 导航窗口⇒“FB/FUN”⇒右击⇒“新建数据”

已创建的程序存储在FB/FUN文件中。

☞ [CPU参数]⇒“程序设置”⇒“FB/FUN文件设置”

一个FB/FUN文件中最多可以存储64个创建的程序。

关于程序创建的关联内容，请参阅下述章节。

| 项目 | 参照目标 |
|-------------------|------------------------------|
| 功能的创建方法 | ☞ GX Works3操作手册 |
| 写入CPU模块的FB/FUN文件数 | ☞ MELSEC iQ-R CPU模块用户手册(入门篇) |

n 可使用软元件/标签

功能程序中可使用的软元件及标签一览如下所示。

○：可以使用，△：只通过指令可以使用(作为表示程序的步的标签禁止使用)，×：禁止使用

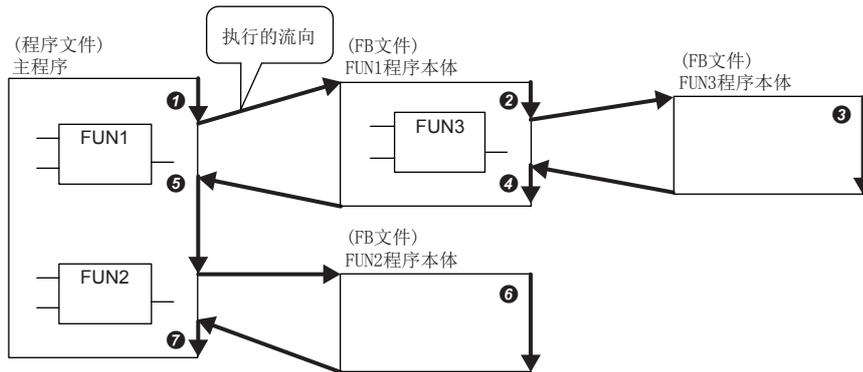
| 软元件/标签的类型 | | 能否使用 |
|-----------|---------|------|
| 标签(指针型以外) | 全局标签 | × |
| | 局部标签 | ○*1 |
| 标签(指针型) | 指针型全局标签 | △ |
| | 指针型局部标签 | ○ |
| 软元件 | 全局软元件 | ○ |
| | 局部软元件 | × |
| 指针 | 全局指针 | △ |
| | 局部指针 | × |

*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

动作概要

功能是将程序本体存储在FB/FUN文件内，在执行时从调用源程序中调用FB/FUN文件内的程序本体后再执行。

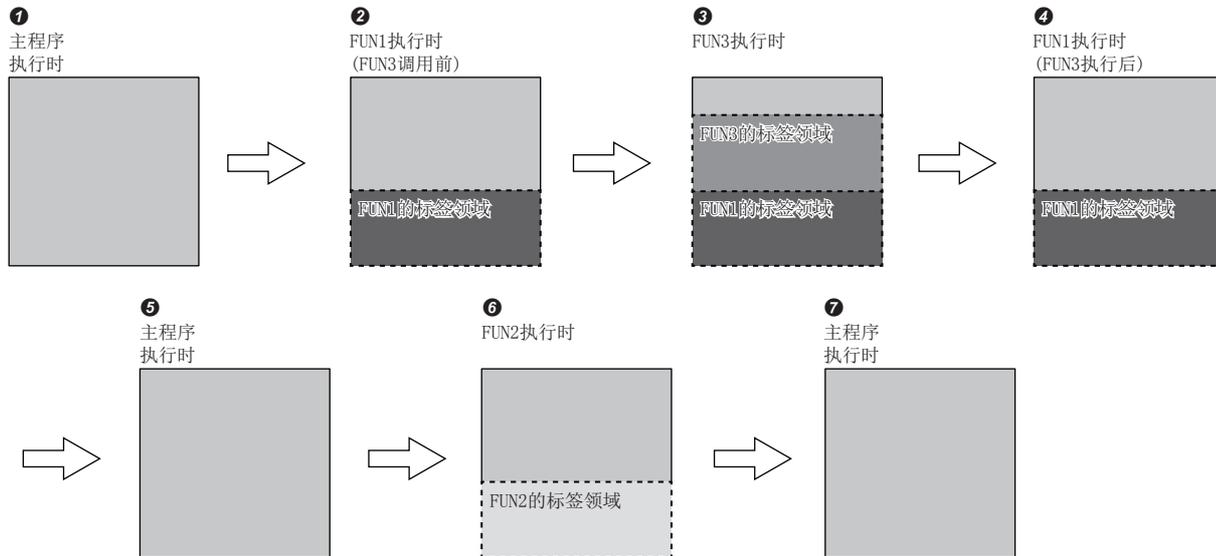


结合子程序型功能块、宏型功能块、全部功能块，可进行32次嵌套。

在功能中定义的标签

在功能中定义的标签的分配目标在功能执行时确保到存储器内的暂时领域中，在执行完成时解除。

对于上述功能执行动作的标签分配状态如下所示。



关于在功能中可定义的标签的类型，请参阅下述章节。

☞ 34页 分类

要点

由于在功能中定义的标签变为不定值，因此在最初访问时需要通过程序进行初始化。

步数

调用功能的情况下，除了程序本体的步数，还需要进行自变量及返回值的交接处理及调用程序本体的步数。

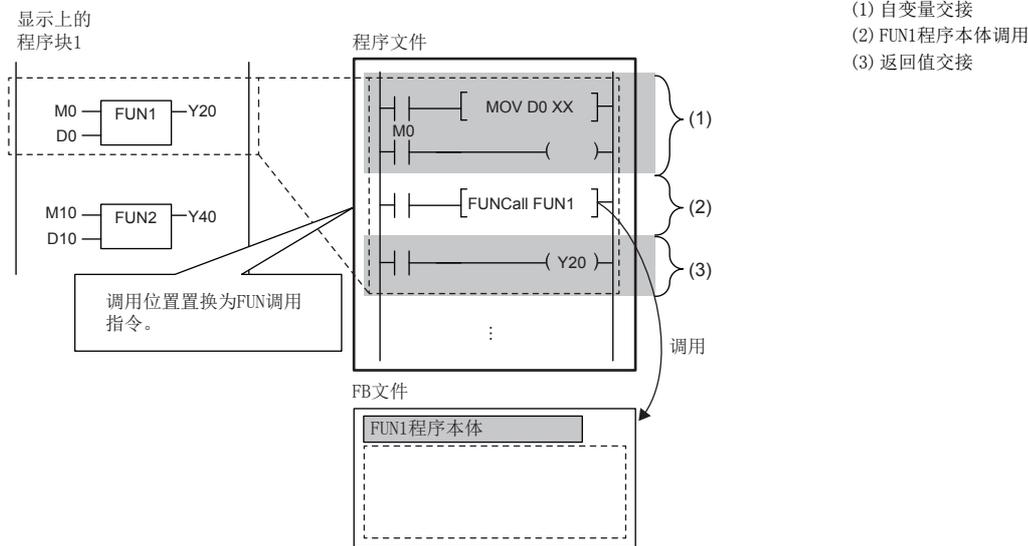
n 程序本体

所使用功能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(指令/通用FUN/通用FB篇)

n 调用侧

调用功能的情况下，在功能调用前后生成功能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、功能的类型、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(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] | LD+MOV LD+DMOV | |
| | 字[带符号] 双字[带符号] | | |
| | 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 | |
| | 排列、结构体 | LD+BMOV | |

• 程序本体调用

功能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 |

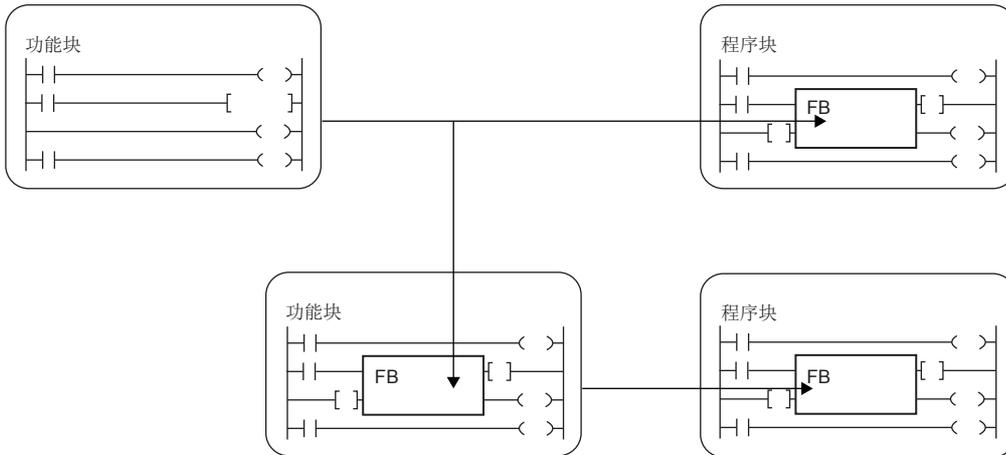
- EN/ENO

EN/ENO所需步数如下所示。

| 项目 | 步数 |
|-----|----|
| EN | 3 |
| ENO | 2 |

3.2 功能块(FB)

功能块是通过程序块及其它的功能块被使用的程序部件。

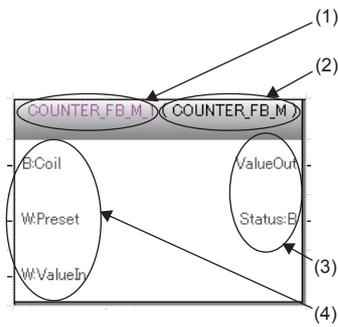


功能块与功能不同，不能保持返回值。

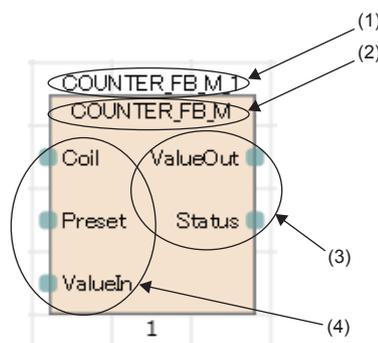
因为功能块能将值保存在变量中，因此也能保持输入状态及处理结果。

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。

梯形图语言的情况下



FBD/LD语言的情况下



- (1) 实例名
- (2) 功能块名
- (3) 输出变量
- (4) 输入变量

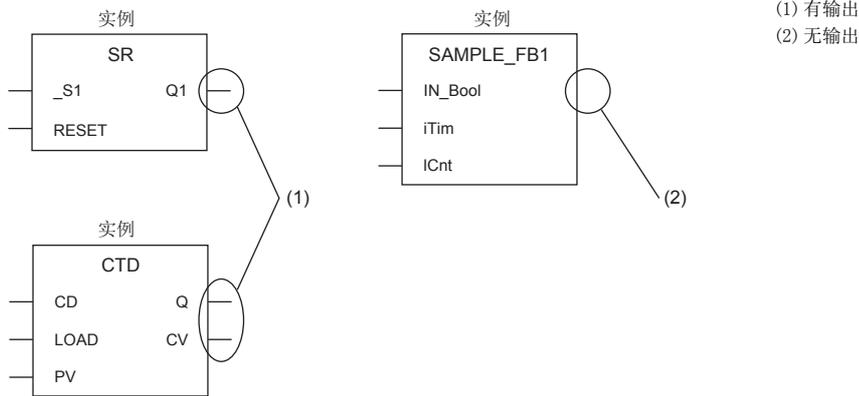
此外，为了在程序上使用功能块，需要定义实例。

☞ 20页 实例

关于输入变量、输出变量、输入输出变量

功能块中需要定义输入变量、输出变量、输入输出变量。

功能块可以输出多个运算结果。此外，也可以不输出。



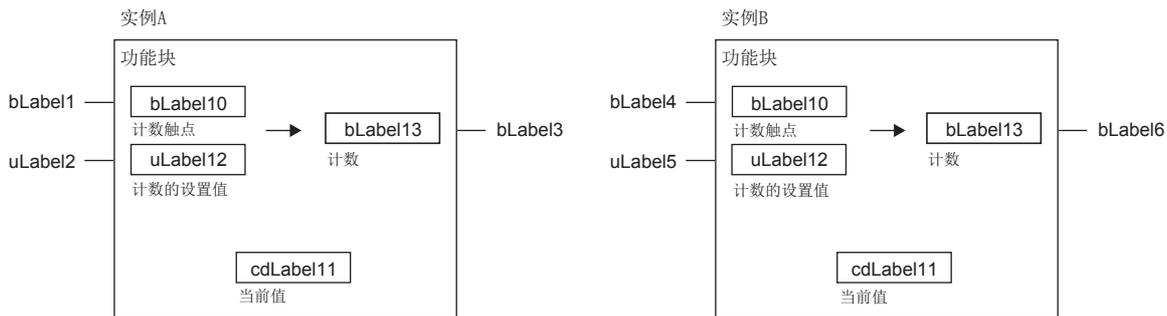
关于可设置的输入变量、输出变量、输入输出变量的分类，请参阅下述章节。

☞ 34页 分类

关于内部变量

功能块使用内部变量。内部变量的功能块的各实例在不同的领域分配了标签。即使是同样的标签名，各实例都保持着不同的状态。

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的功能块。即使是同一个功能块，因为实例A与实例B保持着各自独立的状态，所以输出的时机有所不同。

关于可设置的内部变量的分类，请参阅下述章节。

☞ 34页 分类

关于外部变量

功能块可以使用外部变量。

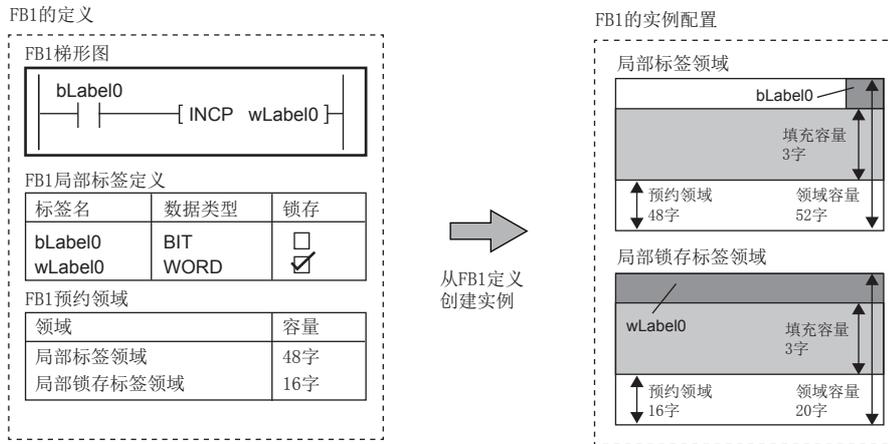
关于可设置的外部变量的分类，请参阅下述章节。

☞ 34页 分类

实例

n 实例含义

功能块的实例是指在功能块的定义的基础上分配的局部标签。从一个功能块定义可以创建多个实例。
实例的配置如下所示。

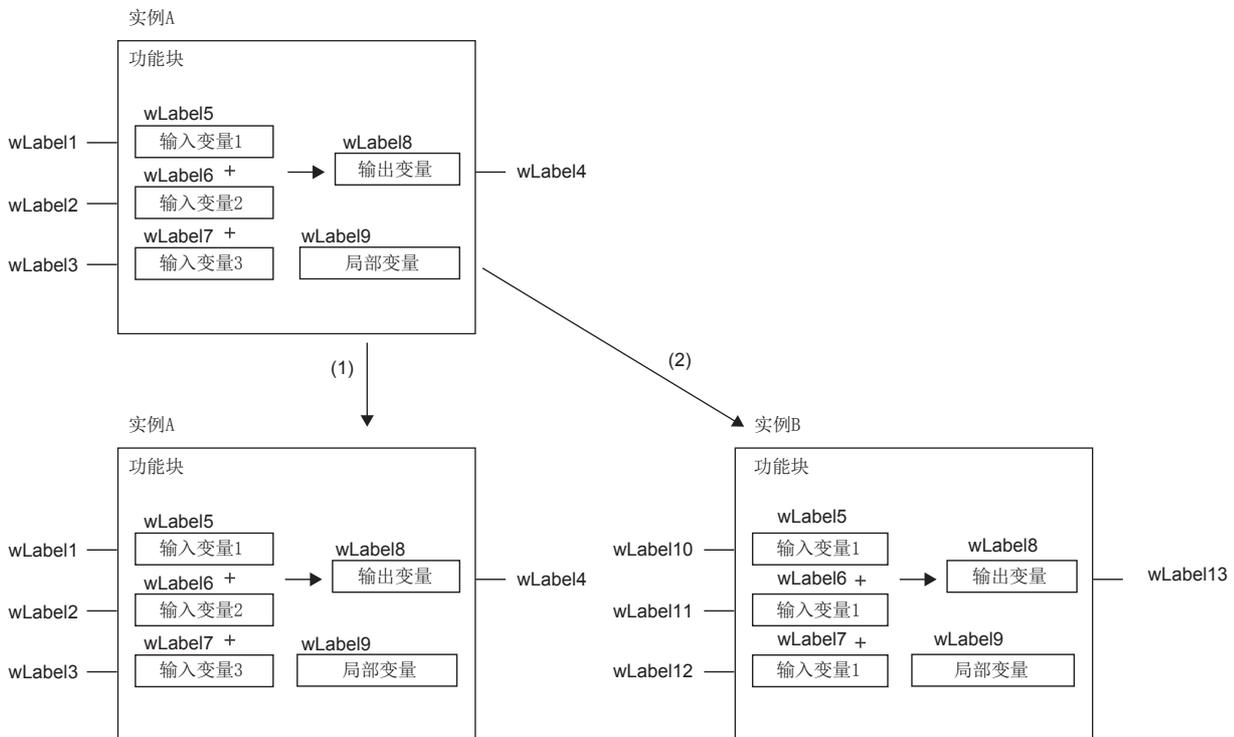


(由于以4字单位确保标签的使用领域，因此在上述示例中确保3字领域(填充容量)。)

n 创建实例

为了使用功能块，需要创建实例。
根据创建功能块的实例，可以从程序及其他的功能块调用使用。
此外，实例可以作为排列进行定义。

创建实例时，在全局标签或使用功能块的程序部件的局部标签中作为标签声明。在一个程序部件中，同一个功能块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，使用不同领域的内部变量。

n 实例的配置

实例是由下述的数据领域构成。

| 实例的数据领域 | 内容 |
|----------|---------------------|
| 局部标签领域 | 分配功能块的局部标签的领域。 |
| 局部锁存标签领域 | 分配功能块的锁存属性的局部标签的领域。 |

n 实例的容量

关于实例的各数据领域的容量，计算方法如下所示。

- 局部标签领域的容量

“实例的局部标签领域的容量” = “锁存属性以外的局部标签的数据容量(总和)” + “预约领域容量”

| 明细 | 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领域的总和。 |
| 预约领域容量 | 用于在RUN中写入添加锁存属性以外的局部标签以及局部实例的预约领域。(固定为48字) |

- 局部锁存标签领域的容量

“实例的局部锁存标签领域的容量” = “全部锁存属性的局部标签的数据容量(总和)” + “预约领域容量”

| 明细 | 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领域的总和。 |
| 预约领域容量 | 用于在RUN中写入添加锁存属性的局部标签以及局部实例的预约领域。(固定为16字) |

实例的局部标签容量应按照工程工具中的标签分配方法。关于工程工具中的标签分配方法，请参阅下述手册。

 GX Works3操作手册

初始值的设置

n 功能块的局部标签的初始值

功能块的局部标签可以设置功能块定义及各实例的初始值。

可设置初始值的局部标签根据类型与属性而不同。

☞ 37页 可定义的数据类型与初始值

n 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 | <p>是各数据类型决定的初始值。功能块的局部标签中未设置初始值的情况下，默认初始值也适用。</p> <p>全局标签定义</p> <table border="1"> <thead> <tr> <th>标签名</th> <th>初始值</th> </tr> </thead> <tbody> <tr> <td>FB1_a</td> <td>-</td> </tr> <tr> <td>-- AAA</td> <td>0</td> </tr> <tr> <td>-- BBB</td> <td>0</td> </tr> <tr> <td>-- CCC</td> <td>0</td> </tr> </tbody> </table> <p>FB1的局部标签定义</p> <table border="1"> <thead> <tr> <th>标签名</th> <th>初始值</th> </tr> </thead> <tbody> <tr> <td>AAA</td> <td></td> </tr> <tr> <td>BBB</td> <td></td> </tr> <tr> <td>CCC</td> <td></td> </tr> </tbody> </table> <p>程序块1的局部标签定义</p> <table border="1"> <thead> <tr> <th>标签名</th> <th>初始值</th> </tr> </thead> <tbody> <tr> <td>FB1_b</td> <td>-</td> </tr> <tr> <td>-- AAA</td> <td>0</td> </tr> <tr> <td>-- BBB</td> <td>0</td> </tr> <tr> <td>-- CCC</td> <td>0</td> </tr> </tbody> </table> <p>(1) 未根据FB1定义在局部标签中设置初始值。 (2) 默认初始值也适用。</p> | 标签名 | 初始值 | FB1_a | - | -- AAA | 0 | -- BBB | 0 | -- CCC | 0 | 标签名 | 初始值 | AAA | | BBB | | CCC | | 标签名 | 初始值 | FB1_b | - | -- AAA | 0 | -- BBB | 0 | -- CCC | 0 |
| 标签名 | 初始值 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FB1_a | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- AAA | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- BBB | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- CCC | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 标签名 | 初始值 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AAA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BBB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CCC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 标签名 | 初始值 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FB1_b | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- AAA | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- BBB | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- CCC | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FB定义初始值 | <p>是在定义功能块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。</p> <p>全局标签定义</p> <table border="1"> <thead> <tr> <th>标签名</th> <th>初始值</th> </tr> </thead> <tbody> <tr> <td>FB1_a</td> <td>-</td> </tr> <tr> <td>-- AAA</td> <td>1111</td> </tr> <tr> <td>-- BBB</td> <td>2222</td> </tr> <tr> <td>-- CCC</td> <td>3333</td> </tr> </tbody> </table> <p>FB1的局部标签定义</p> <table border="1"> <thead> <tr> <th>标签名</th> <th>初始值</th> </tr> </thead> <tbody> <tr> <td>AAA</td> <td>1111</td> </tr> <tr> <td>BBB</td> <td>2222</td> </tr> <tr> <td>CCC</td> <td>3333</td> </tr> </tbody> </table> <p>程序块1的局部标签定义</p> <table border="1"> <thead> <tr> <th>标签名</th> <th>初始值</th> </tr> </thead> <tbody> <tr> <td>FB1_b</td> <td>-</td> </tr> <tr> <td>-- AAA</td> <td>1111</td> </tr> <tr> <td>-- BBB</td> <td>2222</td> </tr> <tr> <td>-- CCC</td> <td>3333</td> </tr> </tbody> </table> <p>(1) 根据FB1定义在局部标签中设置初始值。 (2) FB1的所有实例根据相同的定义初始值进行初始化。</p> | 标签名 | 初始值 | FB1_a | - | -- AAA | 1111 | -- BBB | 2222 | -- CCC | 3333 | 标签名 | 初始值 | AAA | 1111 | BBB | 2222 | CCC | 3333 | 标签名 | 初始值 | FB1_b | - | -- AAA | 1111 | -- BBB | 2222 | -- CCC | 3333 |
| 标签名 | 初始值 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FB1_a | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- AAA | 1111 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- BBB | 2222 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- CCC | 3333 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 标签名 | 初始值 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AAA | 1111 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BBB | 2222 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CCC | 3333 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 标签名 | 初始值 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FB1_b | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- AAA | 1111 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- BBB | 2222 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| -- CCC | 3333 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

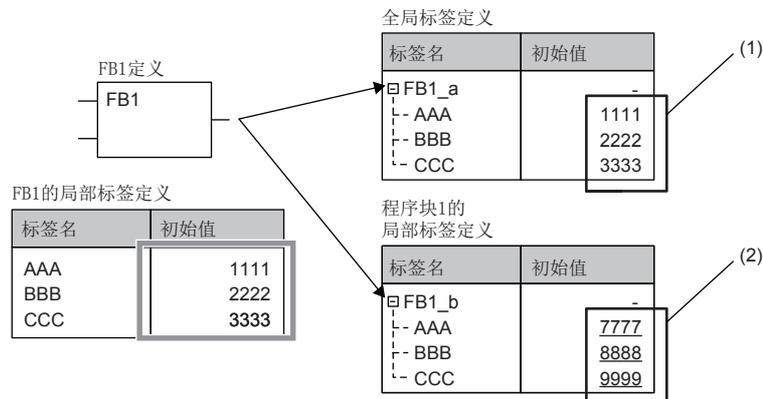
| 类型 | 内容 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|-------|--|-----|-----|-------|---|-----|------|-----|------|-----|------|-----|-----|-----|------|-----|------|-----|------|-----|-----|-------|---|-----|------|-----|------|-----|------|
| 实例初始值 | <p>是在包含全局标签及程序块的局部标签定义的实例中所设置的初始值。</p> <p>全局标签定义</p> <table border="1"> <thead> <tr> <th>标签名</th> <th>初始值</th> </tr> </thead> <tbody> <tr> <td>FB1_a</td> <td>-</td> </tr> <tr> <td>AAA</td> <td>3333</td> </tr> <tr> <td>BBB</td> <td>4444</td> </tr> <tr> <td>CCC</td> <td>5555</td> </tr> </tbody> </table> <p>FB1的局部标签定义</p> <table border="1"> <thead> <tr> <th>标签名</th> <th>初始值</th> </tr> </thead> <tbody> <tr> <td>AAA</td> <td>1111</td> </tr> <tr> <td>BBB</td> <td>2222</td> </tr> <tr> <td>CCC</td> <td>3333</td> </tr> </tbody> </table> <p>程序块1的局部标签定义</p> <table border="1"> <thead> <tr> <th>标签名</th> <th>初始值</th> </tr> </thead> <tbody> <tr> <td>FB1_b</td> <td>-</td> </tr> <tr> <td>AAA</td> <td>7777</td> </tr> <tr> <td>BBB</td> <td>8888</td> </tr> <tr> <td>CCC</td> <td>9999</td> </tr> </tbody> </table> <p>(1) 可以在各FB定义的实例中设置初始值。</p> | 标签名 | 初始值 | FB1_a | - | AAA | 3333 | BBB | 4444 | CCC | 5555 | 标签名 | 初始值 | AAA | 1111 | BBB | 2222 | CCC | 3333 | 标签名 | 初始值 | FB1_b | - | AAA | 7777 | BBB | 8888 | CCC | 9999 |
| 标签名 | 初始值 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FB1_a | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AAA | 3333 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BBB | 4444 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CCC | 5555 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 标签名 | 初始值 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AAA | 1111 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BBB | 2222 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CCC | 3333 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 标签名 | 初始值 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FB1_b | - | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| AAA | 7777 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| BBB | 8888 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| CCC | 9999 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

功能块的初始值可以设置FB定义初始值与实例初始值两者。
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 | 备注 |
|------|---------|-------------------------|
| 高 | 实例初始值 | — |
| ↑ | FB定义初始值 | — |
| ↓ | 默认初始值 | 适用于未设置实例初始值与FB定义初始值的情况。 |
| 低 | | |

要点

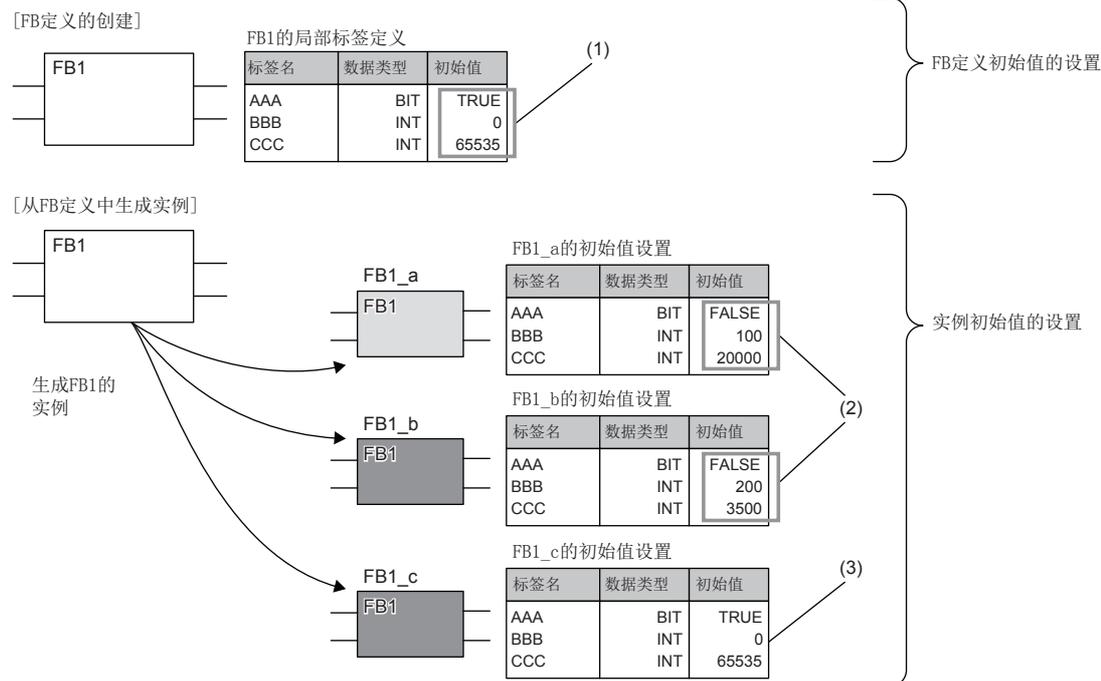
创建两个设置了FB定义初始值的功能块的实例，其中只有一个设置了实例初始值的情况下，FB定义初始值适用于未设置实例初始值的实例，实例初始值适用于设置了实例初始值的实例。



- (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。

n 使用示例

功能块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) 可以在各实例中设置个别的初始值。

(3) 未设置个别的初始值的情况下，通用的初始值也适用。

关于EN/ENO

功能块与功能一样通过附带EN(允许输入)、ENO(允许输出)，可以进行执行处理的控制。

☞ 14页 关于EN/ENO

调用附带EN/ENO的功能块的实例时，必须对EN的实际自变量进行分配。

创建程序

创建功能块的程序的情况下，实施下述操作。

🔍 导航窗口⇒“FB/FUN”⇒右击⇒“新建数据”

已创建的程序存储在FB/FUN文件中。

🔍 [CPU参数]⇒“程序设置”⇒“FB/FUN文件设置”

一个FB/FUN文件中最多可以存储64个创建的程序。

关于程序创建的关联内容，请参阅下述手册。

| 项目 | 参照目标 |
|-------------------|------------------------------|
| 功能的创建方法 | 📖 GX Works3操作手册 |
| 写入CPU模块的FB/FUN文件数 | 📖 MELSEC iQ-R CPU模块用户手册(入门篇) |

n 程序的类型

功能块有下述几种类型，功能块的程序本体的存储方式不同。

- 宏型功能块
- 子程序型功能块

详细内容，请参阅下述章节。

☞ 25页 动作概要

模块FB、通用功能、通用功能块无法进行上述选择。

n 可使用软元件/标签

在功能块程序中可使用的软元件及标签一览如下所示。

○：可以使用， △：只通过指令可以使用(作为表示程序的步的标签禁止使用)， ×：禁止使用

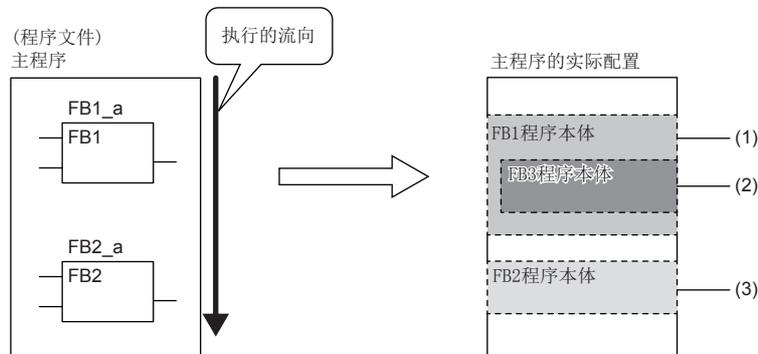
| 软元件/标签的类型 | | 能否使用 |
|-----------|---------|------|
| 标签(指针型以外) | 全局标签 | ○ |
| | 局部标签 | ○ |
| 标签(指针型) | 指针型全局标签 | △ |
| | 指针型局部标签 | ○ |
| 软元件 | 全局软元件 | ○ |
| | 局部软元件 | × |
| 指针 | 全局指针 | △ |
| | 局部指针 | × |

动作概要

n 宏型功能块

宏型功能块对于编程时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。

希望优先程序的处理速度的情况下，应使用宏型功能块。

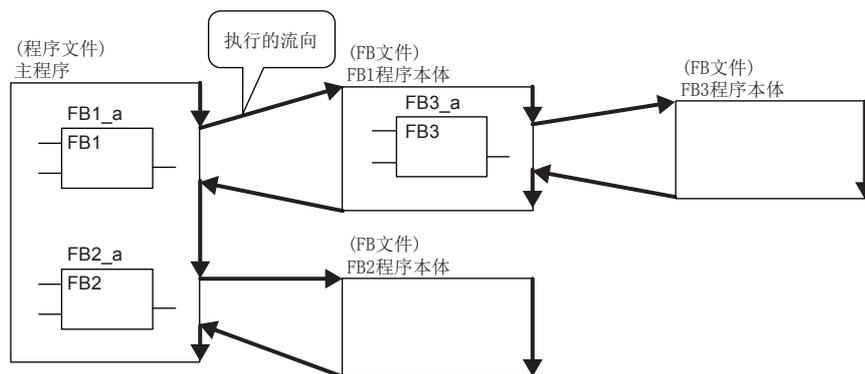


- (1) 展开主程序中的FB1程序本体后执行。
- (2) FB3在FB1的程序本体中展开。
- (3) FB2与FB1同样，FB2程序本体在主程序中展开后执行。

n 子程序型功能块

子程序型功能块将程序本体存储在FB/FUN文件中，执行时从调用源程序中调用FB/FUN文件内的程序本体后执行。

希望使程序的容量变小的情况下，应选择子程序型功能块。

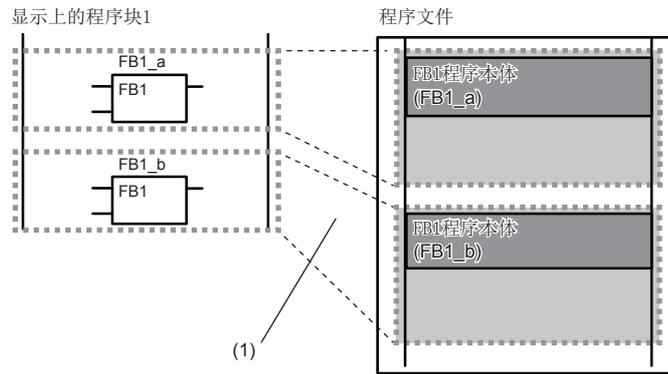


结合子程序型功能块、宏型功能块、全部功能块，可进行32次嵌套。

步数(宏型功能块)

n 调用侧

调用宏型功能块的情况下，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。

n 程序本体

功能块程序本体的步数与普通的程序一样为指令步数的总计。

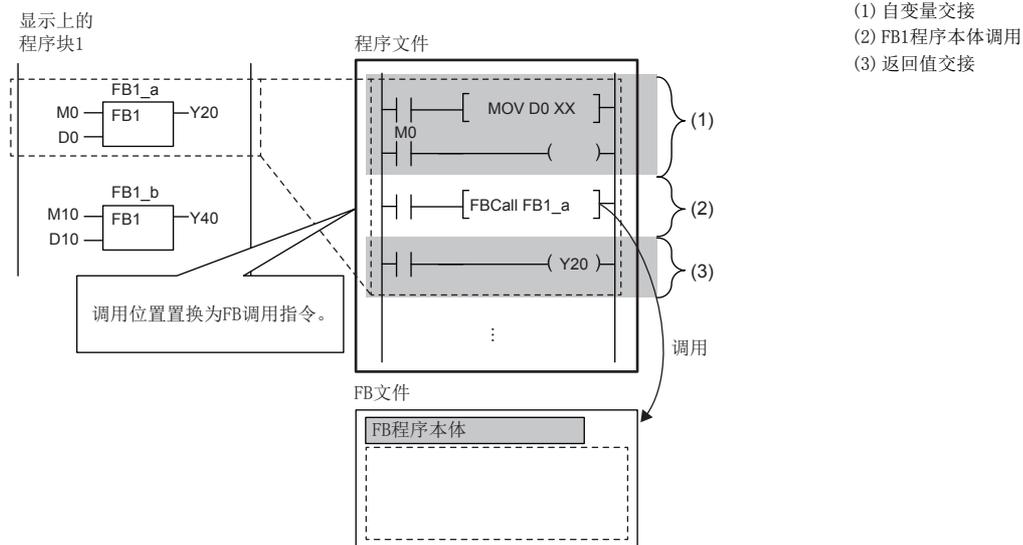
关于各指令的步数，请参阅下述手册。

📖 MELSEC iQ-R编程手册(指令/通用FUN/通用FB篇)

步数(子程序型功能块)

n 调用侧

调用子程序型功能块的情况下，在功能块调用前后生成功能块的自变量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 VAR_IN_OUT | 位 | LD+OUT LD+MOVB (根据所使用的程序语言、功能的类型、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(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 | |
| | 排列、结构体 | LD+BMOV | |

- 程序本体调用

功能块程序本体的调用需要10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 VAR_IN_OUT | 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 |

要点

步数根据下述条件增减。

- 功能块调用的实际自变量及实际返回值变址修饰的情况下
- 指定软元件地址超过16位的情况下
- 位数指定的情况下

n 程序本体

功能块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(指令/通用FUN/通用FB篇)

3.3 注意事项

使用功能的情况下

n 全局指针/局部指针/指针型的全局标签

功能程序中，不可以将全局标签、局部标签及指针型的全局标签作为表示程序的步的标签使用。

使用功能块的情况下

n 全局指针/局部指针/指针型的全局标签

在功能块程序中，不能将全局标签、局部标签及指针型的全局标签作为表示程序的步的标签使用。

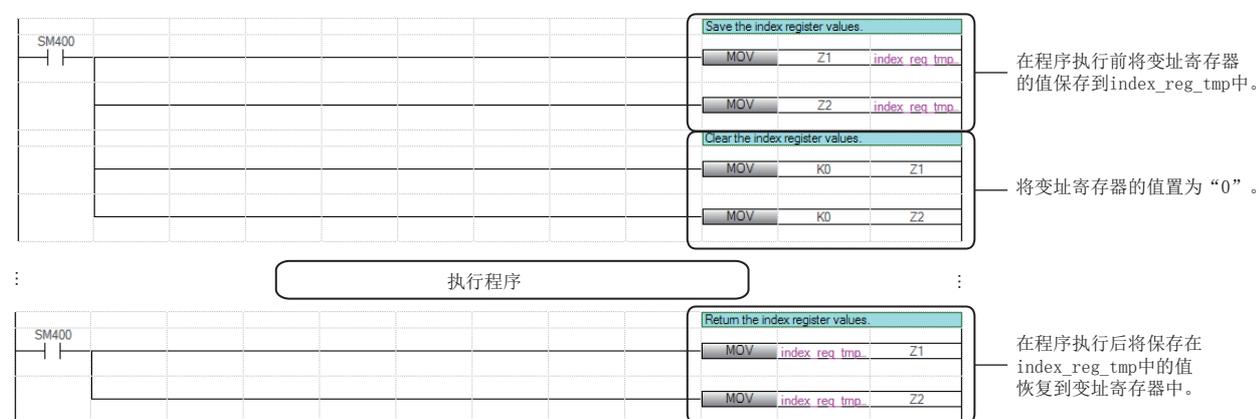
n 使用变址寄存器的情况下

在功能块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，应安装保存电路与恢复电路。

变址寄存器保存时通过将变址寄存器的值设为0，可以根据变址修饰的整合性检查(软元件编号是否超过了软元件范围)防止出错。

例

执行程序前保存变址寄存器Z1、Z2，在程序执行后恢复已保存的变址寄存器的情况下



n 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及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。

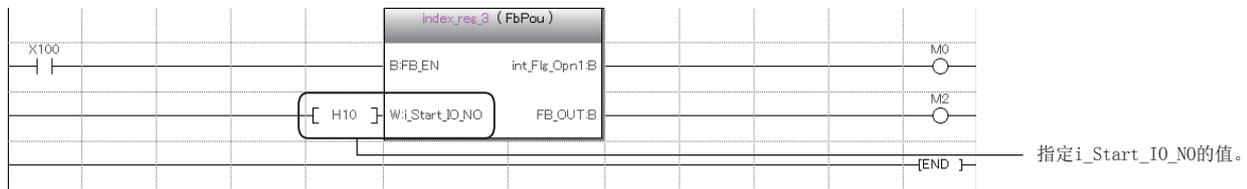
通过将起始输入输出编号作为输入自变量进行接收，在安装位置不同的多个智能功能模块中可以不用更改起始输入输出编号而使用通用功能块。

例

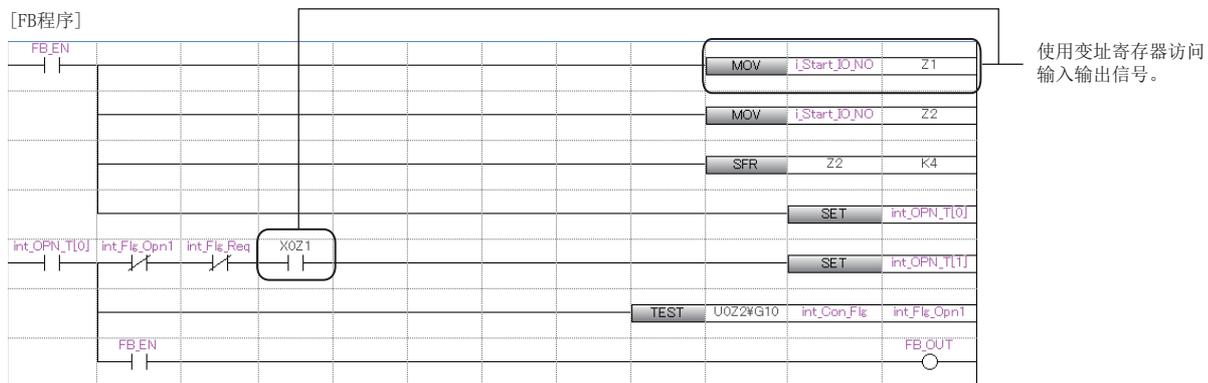
访问智能功能模块的输入输出信号的情况下

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]



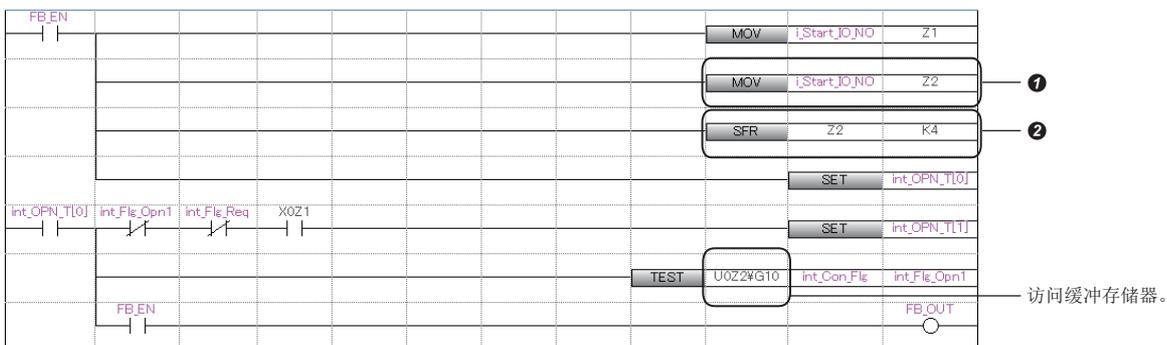
[FB程序]



例

访问智能功能模块的缓冲存储器的情况下

- ① 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。
- ② 使用SFR指令，把值向右移四位，或者使用将值除16所得的商。



n 更改模块FB的对象模块的管理CPU的情况下

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，应将程序复制到其它机号CPU工程中。

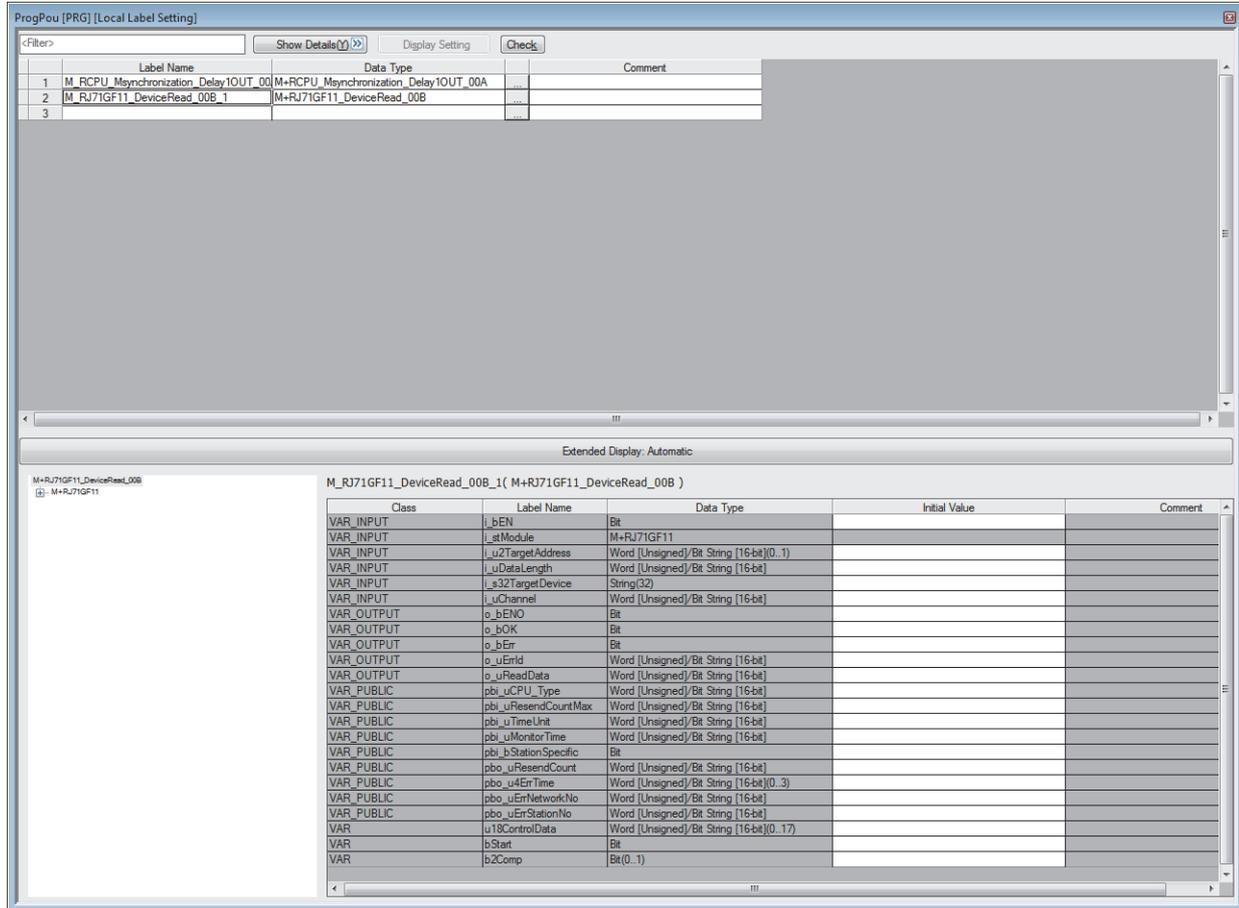
n 关于模块FB的动作参数的更改

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。
- 🔗 工程视图⇒“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。
- 🔗 工程视图⇒“标签”⇒“全局标签”

例

设置局部标签的情况下



在“初始值”栏中设置动作参数。

3.4 使用安全程序的情况下

将在安全程序中使用的功能称之为安全功能，将功能块称之为安全功能块。本节中未记载的内容与普通的功能、功能块相同。
( 13页 功能(FUN)、 18页 功能块(FB))

安全功能(安全FUN)

以下对安全功能有关内容进行说明。

程序的创建

n 可使用软元件及标签

在安全功能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 | × |
| | 一般/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

n 自变量交接

调用安全功能的情况下，在安全功能调用的前后生成安全功能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] | LD+MOV | ○ |
| | 双字[无符号]/位串[32位] | LD+DMOV | |
| | 字[带符号] | | |
| | 双字[带符号] | | |
| | 单精度实数 | LD+EMOV | × |
| | 双精度实数 | LD+EDMOV | × |
| | 时间 | LD+DMOV | × |
| 字符串 | 字符串 | LD+\$MOV | × |
| | 字符串[Unicode] | LD+\$MOV_WS | × |
| | 排列、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(指令/通用FUN/通用FB篇)

安全功能块(安全FB)

以下对安全功能块有关内容进行说明。

实例

n 实例的配置

安全功能块的实例由下述数据领域构成。

○：可以使用， ×：禁止使用

| 实例的数据领域 | 内容 | 能否使用 |
|----------|---------------------|------|
| 局部标签领域 | 分配功能块的局部标签的领域。 | ○ |
| 局部锁存标签领域 | 分配功能块的锁存属性的局部标签的领域。 | × |

程序的创建

n 可使用软元件/标签

在安全功能块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 | × |
| | 一般/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(子程序型功能块)

n 自变量交接

调用安全功能块的情况下，在安全功能块调用的前后生成安全功能块的自变量及返回值的交接处理。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 | ○ |
| VAR_IN_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 | × |
| | 排列、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(指令/通用FUN/通用FB篇)

4 标签

标签是指在输入输出数据及内部处理中指定任意字符串的变量。

在编程中使用标签时，无需软元件及缓冲存储器容量，可以创建程序。

因此，使用了标签的程序即使是模块配置不同的系统也可以简单再利用。

在使用标签时，编程及使用的功能中的一部分需要注意。详细内容，请参阅下述章节。

☞ 44页 注意事项

4.1 类型

在本手册中，对下述标签有关内容进行说明。

- 全局标签
- 局部标签

全局标签

是在一个工程中变为相同数据的标签。可以在工程内的所有程序中使用。

在程序中，可以通过程序块与功能块使用。

在全局标签的设置中进行标签名、分类及数据类型的关联。

此外，全局标签通过公开设置，可以通过GOT及其它站CPU模块进行参阅，监视及数据访问。

n 软元件的分配

全局标签可以分配任意的软元件。

| 项目 | 内容 |
|-----------|---|
| 不分配软元件的标签 | <ul style="list-style-type: none">• 无需软元件也可以进行编程。• 定义的标签被配置到软元件/标签存储器中的标签区或锁存标签区。 |
| 分配软元件的标签 | <ul style="list-style-type: none">• 对于在输入及输出等中使用的软元件，希望作为标签进行编程的情况下，可以直接分配软元件。• 定义的标签被配置到软元件/标签存储器内的软元件区中。 |

局部标签

只能在各程序部件中使用的标签。不可以使用程序部件外的局部标签。

在局部标签的设置中进行标签名、分类与数据类型的设置。

要点

作为标签类型，全局标签与局部标签以外有下述几种类型。

[系统标签]

在与iQ Works相对应的全部工程中变为相同数据的标签。可以通过GOT及其它站的CPU模块、运动控制器进行参阅，监视及数据访问时使用。

关于详细内容，请参阅下述手册。

☞ iQ Works入门指南

[模块标签]

是各模块固有定义的标签。从所使用的模块中自动生成工程工具后可以作为全局标签使用。

关于详细内容，请参阅下述手册。

☞ 所使用模块的FB参考

4.2 分类

标签的分类显示标签在程序的哪个部件以及怎样使用。

根据程序部件的类型，可选择的分类也不同。

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

| 局部标签 | | | | |
|-------------------|---|----------|-----|----|
| 分类 | 内容 | 可使用的程序部件 | | |
| | | 程序块 | 功能块 | 功能 |
| VAR | 是在声明的程序部件的范围内使用的标签。 不可以在其它程序部件中使用。 | ○ | ○ | ○ |
| VAR_CONSTANT | 是在声明的程序部件的范围内使用的常数。 不可以在其它程序部件中使用。 | ○ | ○ | ○ |
| VAR_RETAIN | 是在声明的程序部件的范围内使用的锁存类型的标签。不可以在其它程序部件中使用。 | ○ | ○ | × |
| VAR_INPUT | 是向功能及功能块中输入的标签。 在接收数值的标签中，不可以在程序部件中更改。 | × | ○ | ○ |
| VAR_OUTPUT | 是从功能或功能块中输出的标签。 | × | ○ | ○ |
| VAR_OUTPUT_RETAIN | 是从功能及功能块中输出的锁存类型的标签。 | × | ○ | × |
| VAR_IN_OUT | 是接收数值、从程序部件中输出的局部标签。可以在程序部件内更改。 | × | ○ | × |
| VAR_PUBLIC | 是可以从其它程序部件进行访问的标签。 | × | ○ | × |
| VAR_PUBLIC_RETAIN | 是可以从其它程序部件进行访问的锁存类型的标签。 | × | ○ | × |

4.3 数据类型

标签根据位长、处理方法、值的范围等，分类数据类型。

数据类型有下述几种类型。

- 基本数据类型
- 总称数据类型 (ANY型)

基本数据类型

基本数据类型有下述几种数据类型。

| 数据类型 | | 内容 | 值的范围 | 位长 |
|-----------------|-----------------|--|--|-----|
| 位 | BOOL | 是表示ON或OFF等二者择一的状态的类型。 | 0 (FALSE)、1 (TRUE) | 1位 |
| 字[无符号]/位串[16位] | WORD | 是表示16位排列的类型。 | 0~65535 | 16位 |
| 双字[无符号]/位串[32位] | DWORD | 是表示32位排列的类型。 | 0~4294967295 | 32位 |
| 字[带符号] | INT | 是处理正或负的整数值的类型。 | -32768~32767 | 16位 |
| 双字[带符号] | DINT | 是处理正或负的双精度整数值的类型。 | -2147483648~2147483647 | 32位 |
| 单精度实数 | REAL | 是处理小数点以后的数值(单精度实数值)的类型。 有效位数: 7位(小数点以后6位)*1 | $-2^{128} \sim -2^{-126}$ 、0、 $2^{-126} \sim 2^{128}$ E-3. 402823+38~E-1. 175495-38、 0、E1. 175495-38~E3. 402823+38 | 32位 |
| 双精度实数 | LREAL | 是处理小数点后的数值(双精度实数值)的类型。 有效位数: 15位(小数点以后14位)*1 | $-2^{1024} \sim -2^{-1022}$ 、0、 $2^{-1022} \sim 2^{1024}$ E-1. 79769313486231+308~E- 2. 22507385850721-308、0、 E2. 22507385850721-308~ E1. 79769313486231+308 | 64位 |
| 时间*2 | TIME | 是作为d(日)、h(时)、m(分)、s(秒)、ms(毫秒)处理数值的类型。 | T#-24d20h31m23s648ms~ T#24d20h31m23s647s*3 | 32位 |
| 字符串 | STRING | 是处理ASCII代码、移位JIS代码的字符串的类型。 | 半角最多255字符 | 可变 |
| 字符串[Unicode] | WSTRING | 是处理Unicode字符串的类型。 | 最多255个字符 | 可变 |
| 定时器 | TIMER | 是与软元件的定时器(T)相对应的结构体。 | ☞ 36页 关于定时器与计数器的数据类型 | |
| 累计定时器 | RETENTIVETIMER | 是与软元件的累计定时器(ST)相对应的结构体。 | | |
| 长定时器 | LTIMER | 是与软元件的长定时器(LT)相对应的结构体。 | | |
| 长累计定时器 | LRETENTIVETIMER | 是与软元件的定时器(LST)相对应的结构体。 | | |
| 计数器 | COUNTER | 是与软元件的计数器(C)相对应的结构体。 | | |
| 长计数器 | LCOUNTER | 是与软元件的计数器(LC)相对应的结构体。 | | |
| 指针 | POINTER | 是与软元件的指针(P)相对应的类型。 (☞ MELSEC iQ-R CPU模块用户手册(应用篇)) | | |

*1 单精度实数数据的输入值超出7位的情况下，将第8位取整为四舍五入后的值。双精度实数数据的输入值超出15位的情况下，将第16位取整为四舍五入后的值。

*2 时间类型在通用功能的时间数据类型功能中使用。关于通用功能，请参阅下述手册。

☞ MELSEC iQ-R编程手册(指令/通用FUN/通用FB篇)

*3 在时间类型的标签中使用常数的情况下，应在起始添加“T#”。

- 在字型标签中，通过进行位No. 指定，可以处理指定位No. 的位数据。
- 对于字型数组的标签，通过进行位数指定，可以作为16位数据或32位数据处理。

关于位指定及位数指定的标记方法，请参阅下述手册。

 MELSEC iQ-R编程手册 (指令/通用FUN/通用FB篇)

n 关于定时器与计数器的数据类型

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

| 数据类型 | 构件名 | 构件名 | 构件的数据类型 | 内容 | 值的范围 |
|--------|-----------------|-----|-----------------|-------------------------------------|--------------------|
| 定时器 | TIMER | S | 位 | 触点如下所示。是与定时器软元件的触点(TS)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | C | 位 | 线圈如下所示。是与定时器软元件的线圈(TC)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | N | 字[无符号]/位串[16位] | 当前值如下所示。是与定时器软元件的当前值(TN)同样的动作。 | 0~65535*1 |
| 累计定时器 | RETENTIVETIMER | S | 位 | 触点如下所示。是与累计定时器软元件的触点(STS)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | C | 位 | 线圈如下所示。是与累计定时器软元件的线圈(STC)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | N | 字[无符号]/位串[16位] | 当前值如下所示。是与累计定时器软元件的当前值(STN)同样的动作。 | 0~65535*1 |
| 长定时器 | LTIMER | S | 位 | 触点如下所示。是与长定时器软元件的触点(LTS)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | C | 位 | 线圈如下所示。是与长定时器软元件的线圈(LTC)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | N | 双字[无符号]/位串[32位] | 当前值如下所示。是与长定时器软元件的当前值(LTN)同样的动作。 | 0~4294967295*1 |
| 长累计定时器 | LRETENTIVETIMER | S | 位 | 触点如下所示。是与长累计定时器软元件的触点(LSTS)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | C | 位 | 线圈如下所示。是与长累计定时器软元件的线圈(LSTC)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | N | 双字[无符号]/位串[32位] | 当前值如下所示。是与长累计定时器软元件的当前值(LSTN)同样的动作。 | 0~4294967295*1 |
| 计数器 | COUNTER | S | 位 | 触点如下所示。是与计数器软元件的触点(CS)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | C | 位 | 线圈如下所示。是与计数器软元件的线圈(CC)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | N | 字[无符号]/位串[16位] | 当前值如下所示。是与计数器软元件的当前值(CN)同样的动作。 | 0~65535 |
| 长计数器 | LCOUNTER | S | 位 | 触点如下所示。是与长计数器软元件的触点(LCS)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | C | 位 | 线圈如下所示。是与长计数器软元件的线圈(LCC)同样的动作。 | 0 (FALSE)、1 (TRUE) |
| | | N | 双字[无符号]/位串[32位] | 当前值如下所示。是与长计数器软元件的当前值(LCN)同样的动作。 | 0~4294967295 |

*1 当前值的值变为在CPU参数的定时器时限设置中设置的单位。

关于各软元件动作的有关内容，请参阅下述手册。

 MELSEC iQ-R CPU模块用户手册 (应用篇)

各构件的指定方法与结构体数据类型的构件指定相同。( 41页 结构体)

总称数据类型 (ANY型)

是总结若干个基本数据类型标签的数据类型。

在功能及功能块的自变量、返回值等中允许多个数据类型的情况下，使用总称数据类型。

在总称数据类型中定义的标签，可以使用低位的数据类型的任何一种。

关于与总称数据类型的类型相对应的基本数据类型，请参阅下述手册。

📖 MELSEC iQ-R编程手册 (指令/通用FUN/通用FB篇)

可定义的数据类型与初始值

在标签的各分类中可定义的数据类型与能否设置初始值如下所示。

| 全局标签 | | |
|---------------------|-------------------|---------|
| 分类 | 可定义的数据类型 | 能否设置初始值 |
| VAR_GLOBAL | 基本数据类型、排列、结构体、功能块 | ○ |
| VAR_GLOBAL_CONSTANT | 基本数据类型*1 | × |
| VAR_GLOBAL_RETAIN | 基本数据类型*1、排列、结构体 | ○ |
| 局部标签 (程序块) | | |
| 分类 | 可定义的数据类型 | 能否设置初始值 |
| VAR | 基本数据类型、排列、结构体、功能块 | ○ |
| VAR_CONSTANT | 基本数据类型*1 | × |
| VAR_RETAIN | 基本数据类型*1、排列、结构体 | ○ |
| 局部标签 (功能) | | |
| 分类 | 可定义的数据类型 | 能否设置初始值 |
| VAR | 基本数据类型*2、排列、结构体 | × |
| VAR_CONSTANT | 基本数据类型*1 | × |
| VAR_INPUT | 基本数据类型*1*2、排列、结构体 | × |
| VAR_OUTPUT | | × |
| 返回值 | | × |
| | | |
| 局部标签 (功能块) | | |
| 分类 | 可定义的数据类型 | 能否设置初始值 |
| VAR | 基本数据类型、排列、结构体、功能块 | ○ |
| VAR_CONSTANT | 基本数据类型*1 | × |
| VAR_RETAIN | 基本数据类型*1、排列、结构体 | ○ |
| VAR_INPUT | | ○ |
| VAR_OUTPUT | | ○ |
| VAR_OUTPUT_RETAIN | | ○ |
| VAR_IN_OUT | | × |
| VAR_PUBLIC | | ○ |
| VAR_PUBLIC_RETAIN | | ○ |
| | | |

*1 不可以定义指针类型。

*2 不可以定义定时器型、累计定时器型、长定时器型、长累计定时器型、计数器型、长计数器型。

要点

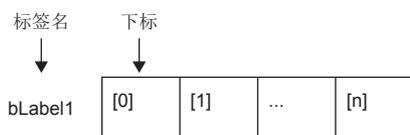
- 在分配软元件的全局标签的情况下，初始值的设置应按照软元件侧的设置。
- 功能块的初始值应按照功能块内的局部标签的设置。
- 结构体类型的初始值应按照结构体定义侧的设置。

4.4 排列

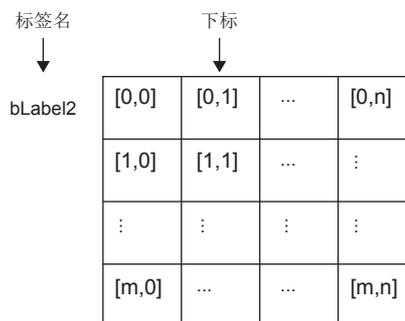
排列是指将相同数据类型的标签的连续集合用一个名称表示。

可以将基本数据类型、结构体及功能块作为排列进行定义。

• 1次元排列的图像



• 2次元排列的图像



排列的定义

n 排列的要素

定义排列时，应决定要素数(排列的长度)。要素数的范围，请参阅下述章节。

☞ 39页 排列要素数的范围

n 多次元排列的次元数

多次元排列最多可定义3次元的排列。

n 定义的书写格式

定义的书写格式如下所示。

排列开始值~排列结束值为止的范围将变为要素数。

| 排列的次元数 | 书写格式 | 备注 |
|--------|--|---|
| 1次元排列 | 基本数据类型/结构体名的排列(排列开始值..排列结束值) | • 关于基本数据类型: ☞ 35页 基本数据类型 • 关于结构体名: ☞ 41页 结构体 |
| | [定义示例]位(0..15) | |
| 2次元排列 | 基本数据类型/结构体名的排列(排列开始值..排列结束值, 排列开始值..排列结束值) | |
| | [定义示例]位(0..1, 0..15) | |
| 3次元排列 | 基本数据类型/结构体名的排列(排列开始值..排列结束值, 排列开始值..排列结束值, 排列开始值..排列结束值) | |
| | [定义示例]位(0..2, 0..1, 0..15) | |

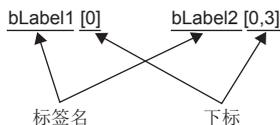
n 初始值

对于一个排列定义，只能设置一个初始值。(不能在各要素中设置不同的初始值。)

初始值存储在排列的全部要素中已设置的值。

使用方法

使用排列时，为了识别各个标签，在标签名后用“[]”将下标括起来。
此外，2次元以上的排列的情况下，“[]”内的下标要用“逗号(,)”隔开。

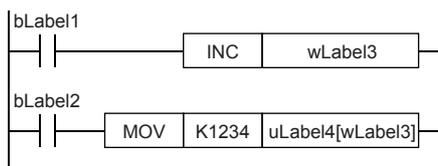


排列中的下标可以指定为下述类型。

| 类型 | 指定示例 | 备注 |
|-----|------------------|--|
| 常数 | bLabel1[0] | 可以指定整数。 |
| 软元件 | bLabel1[D0] | 可以指定字软元件、双字软元件、10进制常数、16进制常数。(不能指定ST、LST、G、HG。) |
| 标签 | bLabel1[uLabel2] | 可以指定下述的数据类型。 • 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] |
| 表达式 | bLabel1[5+4] | 只能通过ST语言指定。 |

要点

- 通过在排列的下标中指定标签，由于数据存储目标变为动态，可以在执行重复处理的程序上使用。下述为在“uLabel4”的排列中连续“1234”进行存储的程序。



- 梯形图语言的情况下，对于排列可以省略要素编号使用。省略要素编号使用的情况下，将作为排列要素的起始编号被转换。例如定义的标签名为“boolAry”，数据类型为位“(0..2,0..2)”排列的情况下，“boolAry[0,0]”与“boolAry”进行相同的处理。
- 将多次元的排列指定为使用排列的指令及通用功能、通用功能块的设置数据时，在排列要素中最右边的要素被作为一次元排列进行处理。

排列要素数的范围

排列要素数的最大数根据数据类型而有所不同。

| 数据类型 | 设置范围 |
|---|--------------------|
| 位 字[无符号]/位串[16位] 字[带符号] 定时器 计数器 累计定时器 | 1~2147483648 |
| 双字[无符号]/位串[32位] 双字[带符号] 单精度实数 时间 长计数器 长累计定时器 长定时器 | 1~1073741824 |
| 双精度实数 | 1~536870912 |
| 字符串 | 1~2147483648÷字符串长度 |
| 字符串[Unicode] | 1~1073741824÷字符串长度 |
| 功能块 | 1~32768 |

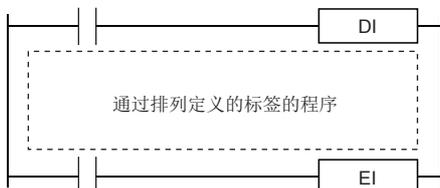
注意事项

n 使用中断程序的情况下

在排列的下标中指定了标签或软元件的情况下，组合多个指令进行运算。

因此，如果在排列中定义的标签的运算中发生中断，将发生数据的背离，变为意料外的运算结果。

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



关于DI/EI指令的详细内容，请参阅下述手册。

📖 MELSEC iQ-R编程手册 (指令/通用FUN/通用FB篇)

n 关于排列的要素

对于定义的排列的要素数，请勿访问要素编号以外的范围。

排列中定义的的范围外的下标在常数中指定的情况下，将变为编译出错。

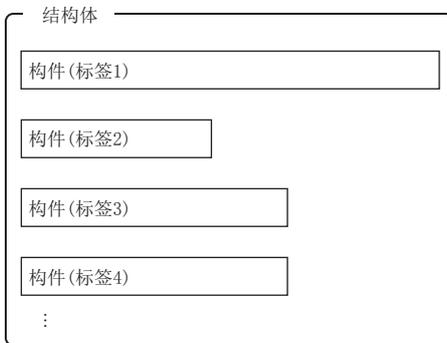
此外，在常数以外指定排列的下标的情况下不会变为编译出错，在执行时访问其它标签区、锁存标签区的领域后进行处理。

4.5 结构体

结构体可以在包含一个以上的标签的数据类型中及所有的程序部件中使用。
包含在结构体中的各个构件(标签)即使数据类型不同也可以定义。

结构体的创建

创建结构体首先要创建结构体的定义，其次在创建的结构体中定义构件。

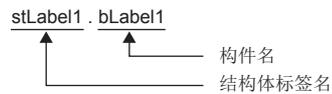


使用方法

使用结构体的情况下，登录预先将定义的结构体置为数据类型的标签。
对于指定配置的各构件，应在结构体标签名后用“句号(.)”断开并附上构件名。

例

使用结构体构件的情况下

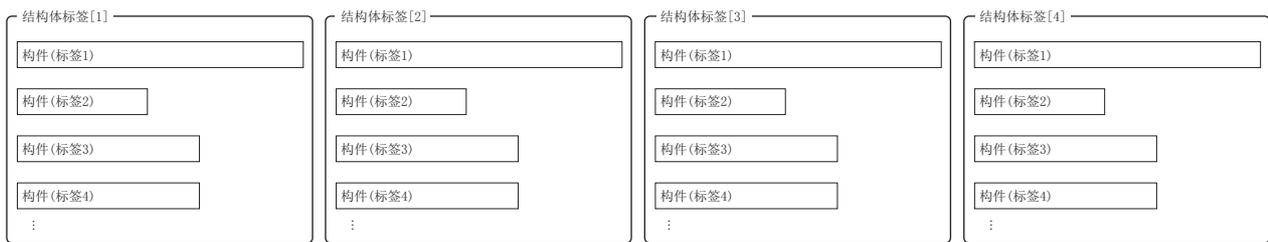


要点

- 在结构体中定义多个数据类型后登录标签，在程序中使用的情况下，运算后的数据存储的顺序不能变为定义数据类型的顺序。编译工程工具时，分类为标签类型与数据类型后进行分配。(根据填充块进行存储器分配)
-  GX Works3操作手册
- 对于使用控制数据(设置指令动作的操作数)的指令，如果指定结构体的标签，根据包装块进行存储器分配，将不可以变为定义的顺序。

结构体的排列

可以排列结构体后进行使用。

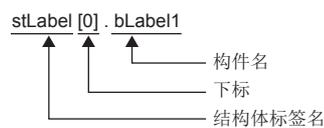


作为排列声明的情况下应在结构体标签名后用“[]”将下标括起来表示。

可以将结构体的排列作为功能及功能块的自变量进行指定。

例

使用置为排列的结构体的要素的情况下



可指定的数据类型

下述数据类型可以作为结构体的构件进行指定。

- 基本数据类型
- 指针型
- 排列
- 其它结构体

结构体的类型

下述标签作为结构体预先被定义。

| 类型 | 参照目标 |
|---------|---------------|
| 模块标签 | ☞ 所使用的模块的FB参考 |
| 定时器型 | ☞ 35页 数据类型 |
| 累计定时器型 | |
| 计数器型 | |
| 长定时器型 | |
| 长累计定时器型 | |
| 长计数器型 | |

4.6 常数

常数的类型

在标签中设置常数时的标记如下所示。

| 可对应的数据类型 | 类型 | 标记方法 | 标记示例 |
|--|--------------|--|---------------------------------------|
| 位 | 引导 | 输入“FALSE”或“TRUE”。 | TRUE、FALSE |
| | 2进制数 | 在所使用的2进制数的数值前附上“2#”。 | 2#0、2#1 |
| | 8进制数 | 在所使用的8进制数的数值前附上“8#”。 | 8#0、8#1 |
| | 10进制数 | 直接输入所使用的10进制数。或在数值前附上“K”。 | 0、1、K0、K1 |
| | 16进制数 | 在所使用的16进制数的数值前附上“16#”。或在数值前附上“H”。 | 16#0、16#1、H0、H1 |
| <ul style="list-style-type: none"> • 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] | 2进制数*1 | 在2进制数前附上“2#”。 | 2#0010、 2#01101010、 2#1111_1111 |
| | 8进制数*1 | 在8进制数前附上“8#”。 | 8#0、8#337、8#1_1 |
| | 10进制数*1 | 直接输入10进制数，或在数值前附上“K”。 | 123、K123、K-123、 12_3 |
| | 16进制数*1 | 在16进制数前附上“16#”。或在数值前附上“H”。 | 16#FF、HFF、16#1_1 |
| <ul style="list-style-type: none"> • 单精度实数 • 双精度实数 | 实数*1 | 对实数进行直接输入且包括小数点及小数点以下的数值。或者在数值前面附上“E”。 | 2.34、E2.34、E- 2.34、3.14_15 |
| | 实数(指数表现) | 指数表现或者在实数值前附上“E”后，在指数部前附上“+”或者“-”。 | 1.0E6、E1.001+5、 1.0E-6、E1.001-5 |
| <ul style="list-style-type: none"> • 字符串 • 字符串[Unicode] | 字符串 | 将字符串用单引号(')或双引号(")括起来。 | 'ABC'、“ABC” |
| | 字符串[Unicode] | | |
| 时间 | 时间 | 在起始附上“T#”。 | T#1h、 T#1d2h3m4s5ms、 TIME#1h |

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

在字符串类型的常数中使用“\$”的情况下

“\$”作为转换序列使用。

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

紧接着“\$”的两个16进制数字符与ASCII代码不相对应的情况下，将变为转换出错。

但是，紧接着“\$”的字符在下述情况下不会变为出错。

| 标记 | 字符串中使用的符号或打印机代码 |
|---------|-----------------|
| \$\$ | \$ |
| \$' | ' |
| \$" | " |
| \$L或\$I | 移行 |
| \$N或\$n | 换行 |
| \$P或\$p | 传送页面 |
| \$R或\$r | 复位 |
| \$T或\$t | 制表 |

4.7 注意事项

有限的功能

在下述功能中使用标签时有限制。

| 项目 | 内容 | |
|-----------------------------|---------------------------------------|--|
| CPU参数 | • 事件执行类型程序的触发 • 多CPU间的刷新设置 | 下述功能中因不可以指定全局标签/局部标签，应使用软元件。*1 |
| 模块参数 | • 通信协议支持功能 | 在下述功能中可以使用模块标签。在没有使用模块标签的情况下应使用软元件。*1 |
| | • 智能功能模块的刷新设置 • 网络模块的刷新设置(仅限SB/SW) | |
| | • 网络模块的刷新设置(SB/SW以外) | 下述功能中因不可以指定全局标签/局部标签，应使用软元件。*1 |
| 数据记录功能 存储器转储功能 实时监视功能 | | 由于不可以指定全局标签/局部标签，在假定预先使用数据记录功能的情况下，应使用软元件。*1 另外，在不能在全局标签中分配软元件的情况下，为了常时执行，应暂时将全局标签复制到其它软元件的传送指令添加到扫描程序中，使用该软元件。 |

*1 全局标签通过分配任意的软元件，可以作为软元件使用。

n 定义分配软元件的全局标签并使用的情况下

按照下述顺序定义全局标签后应使用对标签的使用有限的功能。

另外，由于是在软元件区消耗软元件/标签存储器，所以应确保软元件区。

1. 确保所使用的软元件区。

 CPU参数⇒存储器/软元件设置⇒软元件/标签存储器区设置

2. 在全局标签中定义标签后手动分配软元件。

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

n 将所使用的标签的值暂时复制到其它的软元件中的情况下

按照下述步骤暂时将标签值复制到其它软元件中，应在对标签有限的功能下使用该软元件。

另外，由于是在软元件区消耗软元件/标签存储器，所以应确保软元件区。

1. 确保所使用的软元件区。

 CPU参数⇒存储器/软元件设置⇒软元件/标签存储器区设置

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



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

要点

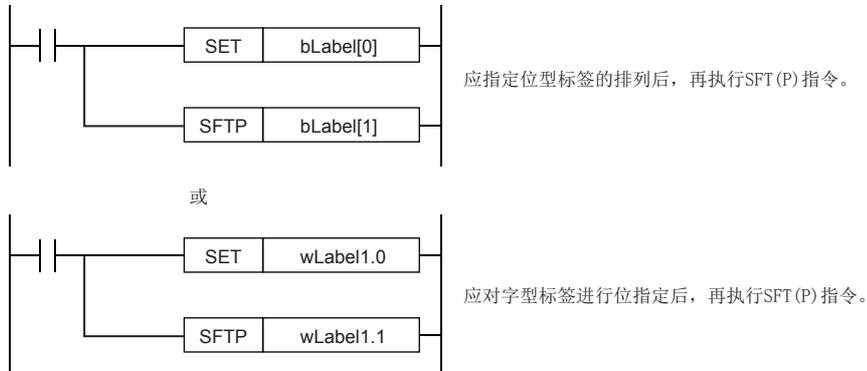
- 执行传送指令时程序的步数会增加。(扫描时间延长)
- 应在考虑至标签写入值的时机及功能的执行时机后决定传送指令的添加位置。

创建程序时的注意事项

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

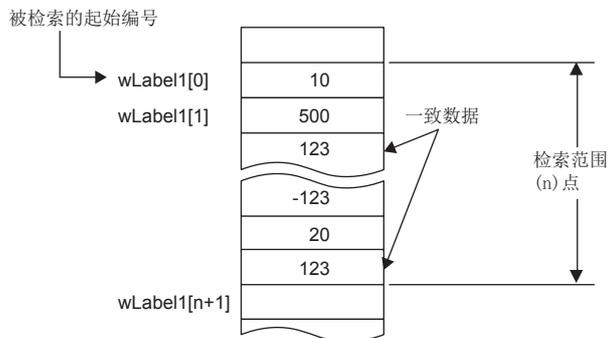
例

SFT(P) 指令的情况下



例

SER(P) 指令的情况下



指定的标签的排列应指定具有比检索范围 (n) 点更大的范围的标签。

标签名的限制

标签名有下述的限制。

- 标签名以字符或下划线 () 开始。不可以定义以数字开始的标签名。
- 不可以将保留字定义为标签名。

关于保留字的有关内容，请参阅下述手册。

GX Works3操作手册

4.8 使用安全程序的情况下

将在安全程序中使用的标签称之为安全标签。本节中未记载的内容与普通标签相同。(☞ 33页 标签)

安全标签的类型

安全标签的类型如下所示。在安全程序中，只能使用下述标签。

- 安全全局标签*1
- 一般/安全共享标签*2
- 安全局部标签

*1 可以分配安全软元件。

*2 一般程序及一般功能块中也可使用。

限制事项

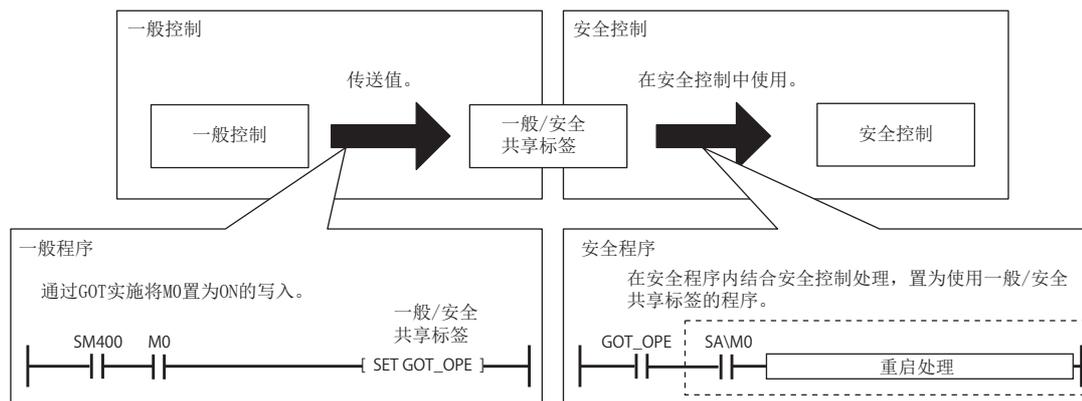
安全标签的设置中不能设置初始值。详细内容，请参阅下述手册。

☞ MELSEC iQ-R CPU模块用户手册(应用篇)

一般/安全共享标签的使用方法

安全程序↔一般程序中交接软元件数据时使用此标签。但是，按以下方式在安全程序中使用一般/安全共享标签的情况下，应创建程序以确保可以确认安全状态。

n 通过来自于GOT的指示重启安全控制的情况下



n 显示报警器的情况下

在一般程序中使用报警器(F)管理安全状态信号的状态的程序示例如下所示。通过一般/安全共享标签(safe_state)，从安全程序接收安全状态信号，通过报警器编号5进行管理。在报警器中检测出发生异常的情况下，将报警器的编号输出到Y20中。

- 一般程序



- (0) 安全状态信号变为OFF之后将报警器编号5置为ON。
- (4) 将SM62(报警器)中检测出的报警器编号输出到Y20中。
- (8) 安全状态信号变为ON之后将报警器编号5置为OFF。

分类

安全全局标签及一般/安全共享标签的分类能否使用如下所示。

○：可以使用， ×：禁止使用

| 分类 | 能否使用 | |
|---------------------|--------|-----------|
| | 安全全局标签 | 一般/安全共享标签 |
| VAR_GLOBAL | ○ | ○ |
| VAR_GLOBAL_CONSTANT | ○ | ○ |
| VAR_GLOBAL_RETAIN | × | × |

安全局部标签的分类能否使用如下所示。

○：可以使用， ×：禁止使用

| 分类 | 能否使用 | | |
|-------------------|------|------|-------|
| | 安全程序 | 安全功能 | 安全功能块 |
| VAR | ○ | ○ | ○ |
| VAR_CONSTANT | ○ | ○ | ○ |
| VAR_RETAIN | × | × | × |
| VAR_INPUT | × | ○ | ○ |
| VAR_OUTPUT | × | ○ | ○ |
| VAR_OUTPUT_RETAIN | × | × | × |
| VAR_IN_OUT | × | × | ○ |
| VAR_PUBLIC | × | × | ○ |
| VAR_PUBLIC_RETAIN | × | × | × |

数据类型

基本数据类型

基本数据类型的能否使用如下所示。

○：可以使用， ×：禁止使用

| 数据类型 | | 能否使用 |
|-----------------|-----------------|------|
| 位 | BOOL | ○ |
| 字[无符号]/位串[16位] | WORD | ○ |
| 双字[无符号]/位串[32位] | DWORD | ○ |
| 字[带符号] | INT | ○ |
| 双字[带符号] | DINT | ○ |
| 单精度实数 | REAL | × |
| 双精度实数 | LREAL | × |
| 时间 | TIME | ○ |
| 字符串 | STRING | × |
| 字符串[Unicode] | WSTRING | × |
| 定时器 | TIMER | ○ |
| 累计定时器 | RETENTIVETIMER | ○ |
| 长定时器 | LTIMER | × |
| 长累计定时器 | LRETENTIVETIMER | × |
| 计数器 | COUNTER | ○ |
| 长计数器 | LCOUNTER | × |
| 指针 | POINTER | × |

结构体

在一般程序与安全程序中共用结构体的定义。但是，下述所示情况下不能使用。

- 在安全程序中无法使用的基本数据类型的构件存在的情况下
- 结构体的定义中设置了初始值的情况下

5 梯形图语言

是通过由触点与线圈构成的电路且串联与并联的组合中由AND/OR组成的逻辑运算，记述顺控程序控制的语言。

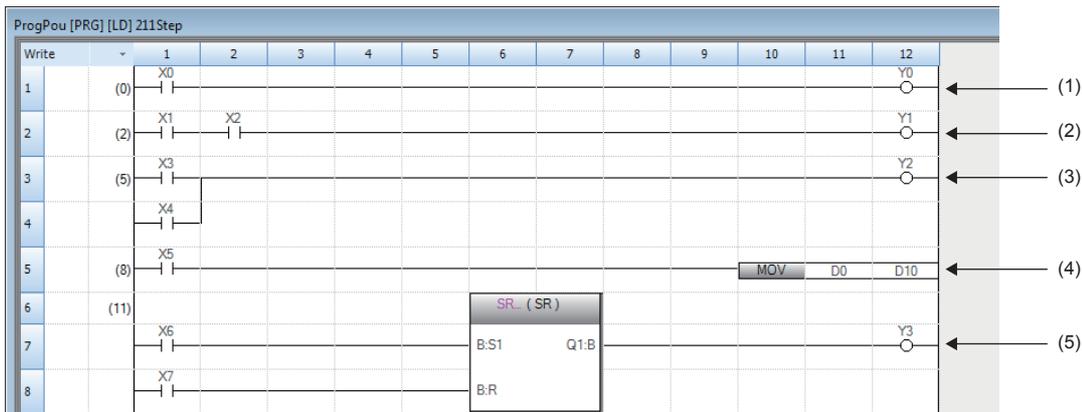
要点

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

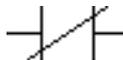
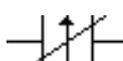
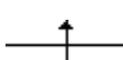
梯形图语言中，可以创建下述电路。

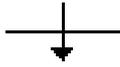
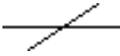
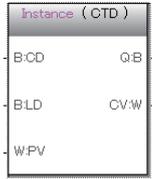


- (1) 由触点与线圈组成的电路
- (2) 由串联组成的电路
- (3) 由并联组成的电路
- (4) 使用了指令的电路
- (5) 使用了通用功能/功能块的电路

梯形图符号

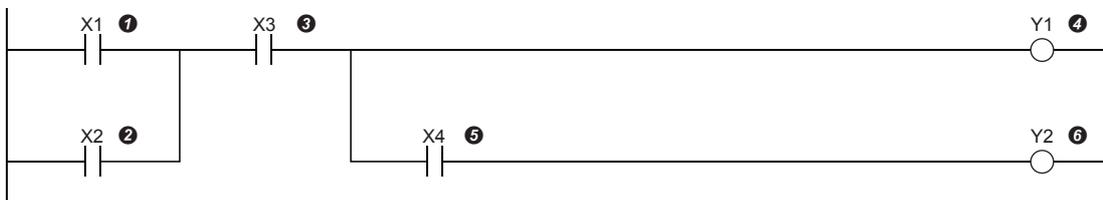
可以在梯形图语言的编程中使用的梯形图符号如下所示。

| 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时以及下降沿时 (OFF→ON) 导通。 |
| 运算结果上升沿脉冲化 |  运算结果上升沿时 (OFF→ON) 导通。运算结果在上升沿以外的情况不导通。 |

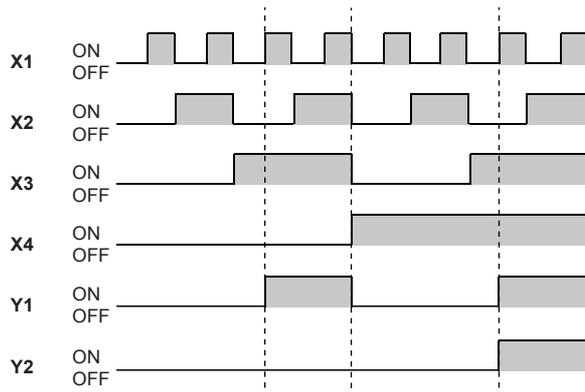
| 项目 | | 内容 |
|------------|---|--|
| 运算结果下降沿脉冲化 |  | 运算结果下降沿时 (ON→OFF) 导通。运算结果在下降沿以外的情况不导通。 |
| 运算结果取反 |  | 对之前的运算结果取反。 |
| 线圈 |  | 将运算结果输出到指定软元件或标签中。 |
| 指令 |  | 执行 [] 内指定的指令。 |
| 返回 |  | 在一个梯形图行中超过了可创建触点数的情况下，创建返回源的符号及返回目标的符号，执行梯形图的返回。 |
| 功能 |  | 执行功能。 <ul style="list-style-type: none"> 功能的创建方法 (GX Works3操作手册) 通用功能 (MELSEC iQ-R编程手册 (指令/通用FUN/通用FB篇)) |
| 功能块 |  | 执行功能块。 <ul style="list-style-type: none"> 功能块的创建方法 (GX Works3操作手册) 通用功能块 (MELSEC iQ-R编程手册 (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的ON的时机如下所示。



注意事项

- 梯形图程序的1行中只能创建一个直接ST。
- 在梯形图程序的1行中无法使用功能块与直接ST盒两者。
- 如果在触点相应的指令位置创建直接ST盒，在线圈相应的指令位置也创建直接ST盒。
- 直接ST内最多可输入的字符数为2048个字符。（换行作为一个字符计数。）
- 由于在直接ST内，上升沿执行指令、下降沿执行指令、特殊定时器指令、边缘检测功能块、计数器功能块可能会无法正常进行动作，因此请勿使用。
- 如果在直接ST内使用“RETURN语句”，不是结束程序块的处理，而是结束直接ST盒内的处理。
- 无法从直接ST内进行功能块的调用。

5.3 声明/注释

在梯形图电路中，可以显示声明及注释。

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。

n 行间声明

对整个梯形图块添加注释。

n P声明

对指针编号添加注释。

n I声明

对中断指针编号添加注释。

注释

通过使用注释，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/注释的类型

作为声明与注释的类型，包括“全体”与“外围”。

| 类型 | 种类 | 内容 |
|----|---|--|
| 全体 | <ul style="list-style-type: none">• 行间声明• P声明• I声明• 注释 | 可以将声明/注释存储到CPU模块中。 全体声明需要下述步数。(全部半角输入的情况下。小数点以后进位。) • 2+字符数÷2(步) |
| 外围 | <ul style="list-style-type: none">• 行间声明• P声明• I声明• 注释 | 不可以将声明/注释存储到CPU模块中。(仅存储位置信息。) 需要在外围设备中进行保存。 在一行中需要一步。 在输入的文本前自动添加*印记。 |

6 ST语言

ST语言是关于逻辑记述方式所规定的国际标准IEC61131-3中定义的语言。ST语言是具有与C语言等相似的语法结构的文本形式的程序语言。适用于梯形图语言中对难以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语言支持控制构文、运算式、功能块(FB)、功能(FUN)，可以按如下方式记述。

例

通过条件语句进行选择分支，通过重复语句进行重复等的控制构文

(*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

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

例

定义的功能块的调用

```
//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

通用功能的调用

(* 将BOOL型数据转换为INT型/DINT型数据 *)

```
wLabel12 := BOOL_TO_INT(bLabel11);
```

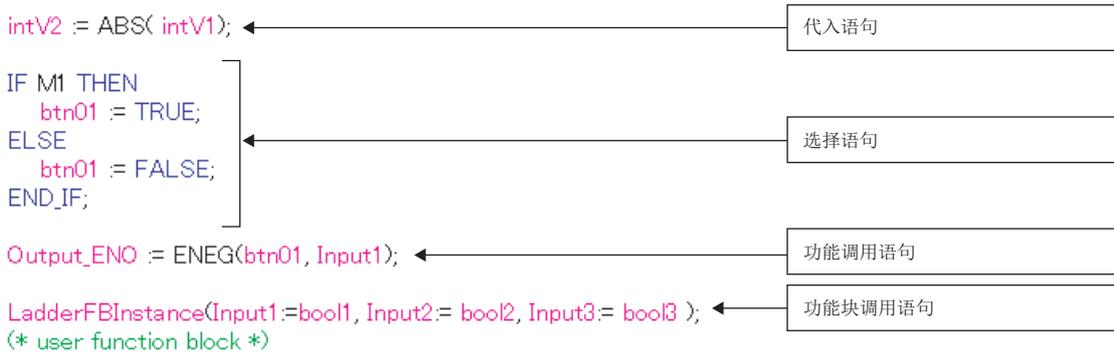
例

汉字等全角字符的使用

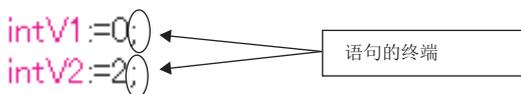
```
//油箱限制ON时关闭阀，OFF时打开阀
IF 油箱限制 = TRUE THEN
  阀 := FALSE ; (* 由于限制变为ON，因此应关闭阀 *)
ELSE
  阀 := TRUE ; /* 由于限制变为OFF，因此应打开阀 */
END_IF;
```

6.1 配置

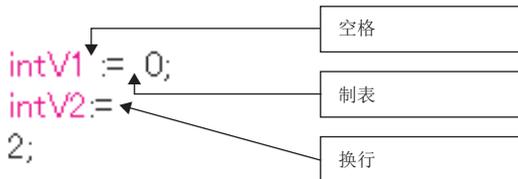
ST语言中的编程由运算符与构文组成。



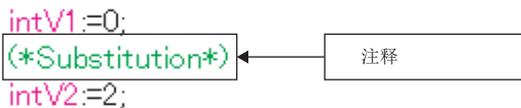
句子的终端必须添加“;”（分号）。



空格、制表、换行可以插入到运算符及数据中。



可以在程序中插入注释。应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 | ;、(、) | ☞ 56页 分隔符 | |
| 运算符 | +、-、<、>、= | ☞ 56页 运算符 | |
| 保留字 | 构文 | IF、CASE、WHILE、RETURN | ☞ 57页 构文 |
| | 软元件 | X0、Y10、M100、Z0 | ☞ MELSEC iQ-R CPU模块用户手册(应用篇) |
| | 数据类型 | BOOL、DWORD | ☞ 35页 数据类型 |
| | 通用功能 | ADD、REAL_TO_STRING_E | ☞ MELSEC iQ-R编程手册(指令/通用FUN/通用FB篇) |
| 常数 | 123、“abc” | ☞ 65页 常数 | |
| 标签 | Switch_A | ☞ 66页 标签与软元件 | |
| 注释 | (*置为ON*) | ☞ 67页 注释 | |
| 其它符号 | 半角空格、换行代码、TAB代码 | — | |

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

☞ GX Works3操作手册

分隔符

在ST语言中为了明确程序的结构，有下述的分隔符。

| 符号 | 内容 |
|----------|---------------------|
| () | 圆括弧式 |
| [] | 排列要素的指定 |
| 。(句号) | 结构体、功能块构件的指定 |
| ，(逗号) | 自变量的分隔 |
| : (冒号) | 软元件型指定符 |
| ; (分号) | 语句的终端 |
| "(双引号) | Unicode字符串的表记 |
| '(单引号) | 字符串(ASCII、移位JIS)的表记 |
| 。。(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 | ANY_NUM |
| <, >, <=, >=, =, <> | ANY_ELEMENTARY*1 | 位 |
| MOD | ANY_INT | ANY_INT |
| AND, &, XOR, OR, NOT | ANY_BIT | ANY_BIT |
| ** | ANY_REAL(底) ANY_NUM(指数) | ANY_REAL |

*1 不可以指定WSTRING型的Unicode字符串。

运算符的优先顺序如下所示。

| 运算符 | 内容 | 示例 | 优先顺序 |
|--------------|----------|-----------------------|------|
| () | 圆括弧式 | (2+3)*(4+5) | 1 |
| 通用功能() | 通用功能的自变量 | CONCAT('AB' , 'CD') | 2 |
| ** | 次方 | 3.0**4 | 3 |
| - | 符号取反 | -10 | 4 |
| NOT | 位型补数 | NOT TRUE | 4 |
| * | 乘法 | 10*20 | 5 |
| / | 除法 | 20/10 | 5 |
| MOD | 求余数 | 17 MOD 10 | 5 |
| + | 加法 | 1.4+2.5 | 6 |
| - | 减法 | 3-2 | 6 |
| <, >, <=, >= | 比较 | 10>20 | 7 |
| = | 一致 | T#26h=T#1d2h | 8 |
| <> | 不一致 | 8#15<>13 | 8 |
| &, AND | 逻辑积 | TRUE AND FALSE | 9 |
| XOR | 异或 | TRUE XOR FALSE | 10 |
| OR | 逻辑或 | TRUE OR FALSE | 11 |

- 在一个公式中有多个优先顺序一样的运算符的情况下，从左侧的运算符开始运算。
- 一个公式中可以记述的运算符的使用个数最多为1024个。

构文

可在ST程序中使用的构文如下所示。

| 项目 | 内容 | 参照页 |
|---------|--------------------------------|-------------|
| 代入语句 | 代入语句 | 57页 代入语句 |
| 子程序控制语句 | 功能块调用语句、功能调用语句 | 59页 子程序控制语句 |
| | RETURN语句 | |
| 选择语句 | IF语句(IF THEN、IF ELSE、IF ELSIF) | 60页 选择语句 |
| | CASE语句 | |
| 重复语句 | FOR语句 | 60页 重复语句 |
| | WHILE语句 | |
| | REPEAT语句 | |
| | EXIT语句 | |

应用半角字符记述构文。

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。 需要使右边公式的结果与左边的数据类型相同。 | intV1:=0; intV2:=2; |

使用排列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。

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

例

```
intAry1:=intAry2;
```

结构体标签的情况下，需要使数据类型(结构体的数据类型)的左边与右边相同。

例

```
dutVar1:=dutVar2;
```

n 数据类型的自动转换

在ST语言中，在记述不同的数据类型的代入及算术运算公式时，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，代入左边的DINT型(dintLabel1)  
dintLabel1 := dintLabel2 + intLabel1 ;  
//算术运算公式： 将INT型(intLabel1)的值自动转换为DINT型，执行DINT型的加法运算  
DMOV(TRUE, wordLabel1, dwordLabel1);  
//指令、功能、功能块调用语句： 将WORD型输入自变量(wordLabel1)的值自动转换为DWORD型，执行传送
```

类型转换通过向代入语句、功能块及功能(包括指令、通用功能、通用功能块)的输入自变量交接(VAR_INPUT部)、算术运算公式进行。

应仅从容量小的数据类型向容量大的数据类型进行转换以确保在类型转换时数据不被丢失。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] | 转换后变为双字[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 将16位数据(字[带符号]或字[无符号]/位串[16位])过渡为数据类型为ANY_REAL输入变量的情况下，自动转换为单精度实数。

*2 将字[无符号]/位串[16位]数据过渡为数据类型为ANY32输入变量的情况下，自动转换为双字[无符号]/位串[32位]。

上述以外的数据类型，应使用类型转换功能。

此外在下述情况下因为无法进行类型转换，应使用类型转换功能。

- 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。

☞ 61页 代入算术运算式结果的情况下

子程序控制语句

n 功能块调用语句

| 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; | 在功能块调用的前后列举输入自变量、输出自变量的代入语句。 |

在功能块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 | 属性 | 使用符号 | 可分配表达式 |
|------------|--------|----------|------|--------|
| VAR_INPUT | 输入变量 | 无或RETAIN | := | 所有的表达式 |
| VAR_OUTPUT | 输出变量 | 无或RETAIN | => | 只有变量 |
| VAR_IN_OUT | 输入输出变量 | 无 | := | 所有的表达式 |
| VAR_PUBLIC | 外部变量 | 无或RETAIN | 禁止指定 | — |

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

| 功能块 | FB定义 | 记述示例 |
|-----------------------|---|---|
| 一个输入变量、一个输出变量的功能块的情况下 | FB名: FBADD FB实例名: FBADD1 输入变量1: IN1 输出变量1: OUT1 | FBADD1(IN1:= Input1); Output1 := FBADD1.OUT1; |
| 3个输入变量、2个输出变量的功能块的情况下 | FB名: FBADD FB实例名: FBADD1 输入变量1: IN1 输入变量2: IN2 输入变量3: IN3 输出变量1: OUT1 输出变量2: OUT2 | FBADD1(IN1:= Input1,IN2:= Input2,IN3:= Input3); Output1 := FBADD1.OUT1; Output2 := FBADD1.OUT2; |

n 功能调用语句

| 书写格式 | 内容 |
|------------------|---|
| 功能名(变量1、变量2、……); | 用“()”将紧接在功能名后的自变量括起来。 多个自变量的情况下用“,”(逗号)隔开。 |

通过向变量代入,存储执行功能的结果。

| 功能 | 记述示例 |
|----------------------------------|--|
| 输入变量为1个功能的情况下(例: ABS) | Output1 := ABS(Input1); |
| 输入变量为3个功能的情况下(例: MAX) | Output1 := MAX(Input1, Input2, Input3); |
| 具有EN/ENO功能(通用功能以外)的情况下(例: MAX_E) | Output1 := MAX_E(boolEN, boolENO, Input1, Input2, Input3); |
| 通用功能的情况下(例: MOV) | boolENO := MOV(boolEN, Input1, Output1); (执行功能的结果变为ENO,第一自变量(变量1)变为EN。) |

不能返回值的用户定义功能及在调用语句的参数中包含VAR_OUTPUT变量的功能,通过将“;”(分号)连接在后面,可作为语句执行。

n RETURN语句

| 构文 | 书写格式 | 内容 | 记述示例 |
|----------|---------|---|-------------------------------------|
| n RETURN | RETURN; | 为了使程序、功能块、功能在中途结束使用。 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。 如果在功能块中使用RETURN语句,将从功能块返回。 如果在功能中使用RETURN语句,将从功能返回。 对一个RETURN语句,在系统中使用1点指针型标签。 | IF bool1 THEN RETURN; END_IF; |

选择语句

| 构文 | 书写格式 | 内容 | 记述示例 |
|-----------------|--|--|---|
| n 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,则执行语句。若布尔表达式为假(FALSE)时,则不执行语句。 在布尔表达式中,作为在单一的类型变量的状态下或包含多个变量的复杂的表达式的布尔运算结果,如果是返回真(TRUE)或假(FALSE)的表达式,可以使用任意表达式。 | IF bool1 THEN intV1 := intV1 + 1; END_IF; |
| n 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,则执行语句1。 布尔表达式的值为假(FALSE)时,则执行语句2。 | IF bool1 THEN intV3 := intV3 + 1; ELSE intV4 := intV4 + 1; END_IF; |
| n IF.....ELSEIF | IF<布尔表达式1>THEN <语句1...>; ELSE<布尔表达式2>THEN <语句2...>; ELSE<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,则执行语句1。布尔表达式1的值为假(FALSE)而布尔表达式2的值为真(TRUE)时,则执行语句2。 布尔表达式1、2的值都为假(FALSE)而布尔表达式3的值为真(TRUE)时则执行语句3。 | IF bool1 THEN intV1 := intV1 + 1; ELSIF bool2 THEN intV2 := intV2 + 2; ELSIF bool3 THEN intV3 := intV3 + 3; END_IF; |
| n CASE | CASE<整数表达式> <整数选择值1>: <语句1...>; <整数选择值2>: <语句2...>; : <整数选择值n>: <语句n...>; ELSE <语句n+1...>; END_CASE; | 执行具有与整数表达式(条件表达式)的值一致的整数的选择值的语句,在无一致的情况下,则执行ELSE语句的下一语句。 对于CASE语句,例如可以在根据单一的整数值及复杂的表达式的结果的整数值执行选择语句的情况下使用。 | CASE intV1 OF 1 : bool1 := TRUE; 2 : bool2 := TRUE; ELSE intV1 := intV1 + 1; END_CASE; |

重复语句

| 构文 | 书写格式 | 内容 | 记述示例 |
|--------------------|--|---|--|
| n 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; | 进行作为反复变量使用数据的初始化。 根据增加公式对初始化后的反复变量进行加法运算或减法运算,直至超出最终值,将重复执行以DO开始END_FOR内的一个以上的语句。 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; |
| n 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,则执行超过1个的语句。 在执行语句之前判定布尔表达式,布尔表达式为假(FALSE)的情况下,不执行DO...END_WHILE中的语句。对于WHILE语句中的<布尔表达式>,由于与结果返回真或假无关,因此IF语句中的<布尔表达式>中的可指定的表达式则全部可以使用。 | WHILE intV1 = 30 DO intV1 := intV1 + 1; END_WHILE; |
| n 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,则执行超过1个的语句。 布尔表达式在语句执行后判定,值为真(TRUE)时则不执行REPEAT.....UNTIL内的语句。REPEAT语句中的<布尔表达式>,由于与结果返回真或假无关,因此IF语句中的<布尔表达式>中的可指定的表达式则全部可以使用。 | REPEAT intV1 := intV1 + 1; UNTIL intV1 = 30 END_REPEAT; |
| n EXIT | EXIT; | 通过只在重复语句中可使用的构文,使重复语句在中途结束。 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。终止重复语句后从下一行继续程序的执行。 | FOR intV1 := 0 TO 10 BY 1 DO IF intV1 > 10 THEN EXIT; END_IF; END_FOR; |

注意事项

n 使用代入语句时

- 字符串的代入的最大字符串长255字符。代入超过最大字符串长的字符串时，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。
- 功能块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。

n 使用步继电器(S)及SFC块软元件(BL)的情况下

将步继电器(S)及SFC块软元件(BL)在代入式的右边、功能及功能块的输入自变量中使用的情况下，有可能变为转换出错状态。该情况下，应替换代入式。

例

替换示例如下所示。

| 替换前 | 替换后 |
|-----------|--|
| MO := S0; | IF S0 THEN MO := TRUE; ELSE MO := FALSE; END_IF; |

此外，将步继电器(S)及带块指定步继电器(BL□\S□)的位数指定在程序中使用的情况下，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 |
|--|---|
| (*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); |

n 代入算术运算式结果的情况下

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，应预先将算术运算表达式的变量转换为左边的数据类型之后再行运算。

例

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

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

算术运算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，上溢或下溢的运算结果被代入到varDint1中。

在这种情况下，应预先将运算表达式的操作数转换到左边的数据类型之后再行运算。

```
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
```

n 在算术运算式中使用符号取反运算符的情况下

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的情况下

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

该情况下，(-varInt1)的值将保持为-32768不变，varDint2中代入-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下

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

例

不使用符号取反运算符的情况下

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

n 从单精度实数至双精度实数转换数据型的情况下

从单精度实数至双精度实数的型转换 (REAL_TO_LREAL) 中, 转换结果可能会发生误差。

因此进行数据类型自动转换的情况下及返回值在代入语句的右边、算术运算式的操作数中使用了实数型功能 (SIN等) 的情况下, 可能无法变为希望的结果。

例

发生误差的情况下

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

上述情况下, ABS(varReal1) 的返回值将变为单精度实数, 由于将该值型转换为双精度实数并代入 varLReal 中, 因此将发生误差。

这种情况下, 应通过与代入目标相同的数据类型 (双精度实数) 创建执行功能的程序。

例

不发生误差的情况下

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

n 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 |
|---|----------------|
| <pre>IF bLabel1 THEN bLabel2 := TRUE; END_IF;</pre> | |

为避免常时 ON, 应按下述方式添加将位型标签置为 OFF 的程序。

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 |
|--|----------------|
| <pre>IF bLabel1 THEN bLabel2 := TRUE; ELSE bLabel2 := FALSE; END_IF;</pre> | |

*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。

n 使用定时器功能块、计数器功能块时

对于选择语句中的布尔表达式(条件表达式)，定时器功能块、计数器功能块的执行条件不同。

例

定时器功能块的情况下

更改前程序示例

```
IF bLabel1 THEN
    TIMER_100_FB_M_1(Coil:= bLabel2, Preset:= wLabel3, ValueIn:= wLabel4, ValueOut=> wLabel5, Status=> bLabel6);
END_IF;
(* bLabel1=ON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

计数器功能块的情况下

更改前程序示例

```
IF bLabel1 THEN
    COUNTER_FB_M_1(Coil:= bLabel2, Preset:= wLabel3, ValueIn:= wLabel4, ValueOut=> wLabel5, Status=> bLabel6);
END_IF;
(* bLabel1=ON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);
```

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

n 使用FOR...DO语句时

- 无法在重复自变量中使用结构体构件及排列要素。
- 应使在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构文以下可能不执行或变为无限循环。
- FOR……DO构文中FOR构文中的<语句…>在执行后进行重复变量的计数处理。超过重复变量的数据类型的最大值或低于最小值的计数处理被执行的情况下，发生无限循环。

n 使用上升沿执行指令、下降沿执行指令时

- IF语句以及CASE语句中，使用上升沿执行指令、下降沿执行指令时的动作如下所示。

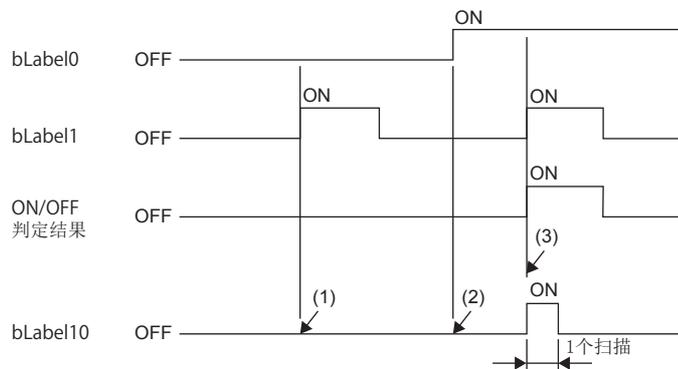
| 条件 | | | 动作结果 | | |
|-------------------|------------|--------------------|--------------|---------|---------|
| IF语句、CASE语句的条件表达式 | 指令执行条件(EN) | 上一次扫描时指令ON/OFF判定结果 | 指令ON/OFF判定结果 | 上升沿执行指令 | 下降沿执行指令 |
| TRUE或CASE一致 | TRUE | ON | ON | 不执行 | 不执行 |
| | | OFF | ON | 执行 | 不执行 |
| | FALSE | ON | OFF | 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

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

例

在IF语句中使用了PLS指令(上升沿执行)的情况下

```
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。(仅bLabel10为1扫描ON)

- 以重复语句 (FOR语句、WHILE语句或REPEAT语句)，执行上升沿执行指令或下降沿执行指令的情况下，使用变址继电器 (V) 以及变址修饰。该情况下，通过使用变址继电器 (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，确保在使用中指令数的变址继电器 (V)。

例

在FOR语句中使用上升沿执行指令及下降沿执行指令的情况下

在1个地方使用变址继电器 (V) 的示例
(可最多使用变址继电器 (V) 合计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
(可最多使用变址继电器 (V) 合计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;
```

n 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) 中，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，代入语句的情况下为无处理，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 (位代入语句的情况下)

```
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指令的情况下)

```
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

ST程序中的字符串的标记方法如下所示。

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

上述以外常数的标记方法，请参阅下述章节。

☞ 43页 常数

标签与软元件

指定方法

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

关于可使用的标签，请参阅下述章节。

 33页 标签

关于可使用的软元件，请参阅下述手册。

 MELSEC iQ-R CPU模块用户手册(应用篇)

n 带类型指定软元件表记

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的自变量部分中作为无类型指定的软元件被指定的情况下，则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下 |
| :U | 字[无符号]/位串[16位] | D0:U | 将D0作为字[无符号]/位串[16位]的值 |
| :D | 双字[带符号] | D0:D | 将D0、D1作为双字[带符号]的值 |
| :UD | 双字[无符号]/位串[32位] | D0:UD | 将D0、D1作为双字[无符号]/位串[32位]的值 |
| :E | 单精度实数 | D0:E | 将D0、D1作为单精度实数的值 |
| :ED | 双精度实数 | D0:ED | 将D0、D1、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□)
- 文件寄存器(R/ZR)
- 刷新数据寄存器(RD)

进行了位数指定或变址修饰的软元件不能赋予软元件型指定符。

n 软元件的指定方法

关于软元件的指定可以使用下述方法。

- 变址修饰
- 位指定
- 位数指定
- 间接指定

关于详细内容，请参阅以下手册。

 MELSEC iQ-R CPU模块用户手册(应用篇)

 MELSEC iQ-R编程手册(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变为字[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变为双字[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相一致。

例

D0:=K5X0;

在上述情况下，因为K5X0变为双字型，D0变为字型，所以程序出错。

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

例

K5X0:=2#1011_1101_1111_0111_0011_0001;

在上述情况下，由于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)时，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，应使用类型转换功能。

例

varInt:=WORD_TO_INT(T0);(*使用类型转换功能*)

- 定时器及计数器的软元件的线圈(TC、STC、LTC、LSTC、CC、LCC)作为代入式的右边及功能、功能块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下

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。*)

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 |
|------|------|--|--|
| 单行注释 | // | 将从开始符号“//”到行尾的内容作为注释。 | //注释内容 |
| 多行注释 | (**) | 将从开始符号“(**)”到结束符号“(**)”的内容作为注释。 可以在注释中换行输入。 | n 无换行 (*注释内容*) n 有换行 (*第1行注释内容 第2行注释内容*) |
| | /**/ | 将从开始符号“/**/”到结束符号“/**/”的内容作为注释。 可以在注释中换行输入。 | n 无换行 /*注释内容*/ n 有换行 /*第1行注释内容 第2行注释内容*/ |

在多行注释中请勿记述含有结束符号的注释。

7 FBD/LD语言

是通过按照数据及信号的流向将进行特定处理的块、变量部件、常数部件接线，对程序进行记述的图表语言。

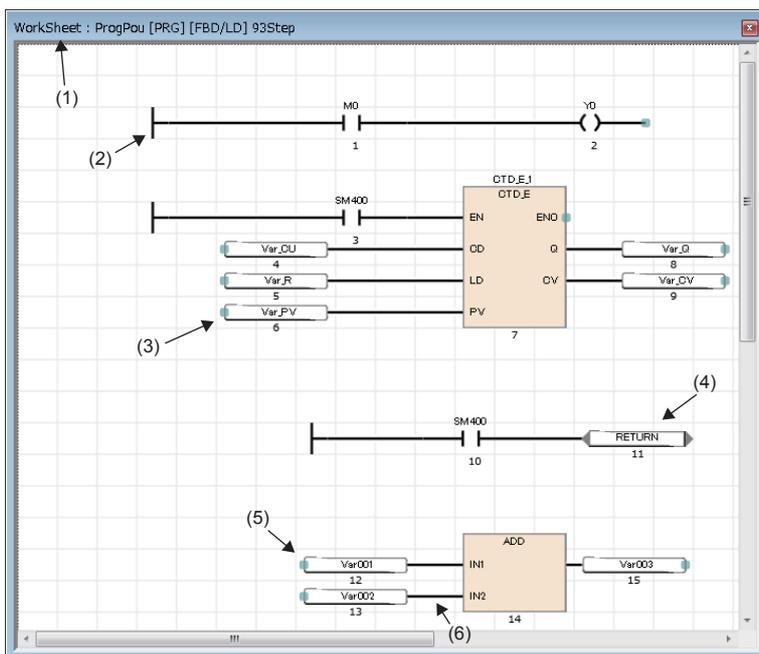
要点

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

7.1 配置

FBD/LD语言中，可创建下述所示程序。



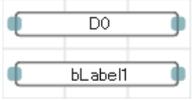
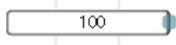
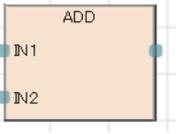
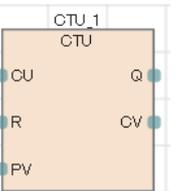
- (1) 工作表
- (2) LD部件
- (3) FBD部件
- (4) 通用部件
- (5) 连接点
- (6) 连接线

在FBD/LD语言的程序中，数据从功能块(FB)、功能(FUN)、变量部件(标签与软元件)、常数部件的输出点流向其它功能块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。

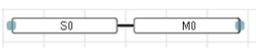
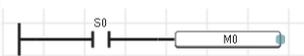
| 项目 | | 内容 |
|---------|---|---|
| 变量 |  | 用于存储各个值(数据)时使用变量。变量中规定数据类型，仅存储该数据类型的值(数据)。在变量中，可以指定标签或软元件。 |
| 常数 |  | 输出所指定的常数值。 |
| 功能(FUN) |  | 执行功能。 <ul style="list-style-type: none"> 功能的创建方法(《GX Works3操作手册》) 通用功能(《MELSEC iQ-R编程手册(指令/通用FUN/通用FB篇)》) |
| 功能块(FB) |  | 执行功能块。 <ul style="list-style-type: none"> 功能块的创建方法(《GX Works3操作手册》) 通用功能块(《MELSEC iQ-R编程手册(指令/通用FUN/通用FB篇)》) 模块FB(《所使用模块的FB参考》) |

n 使用步继电器(S)及SFC块软元件(BL)的情况下

将步继电器(S)及SFC块软元件(BL)变量部件中使用的情况下，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

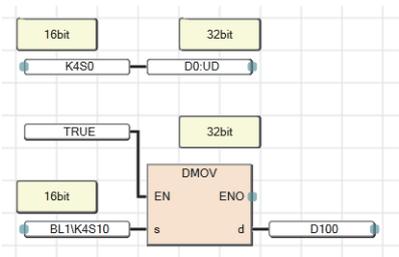
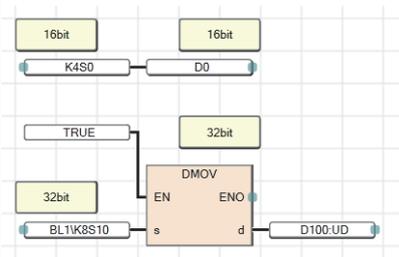
替换示例如下所示。

| 替换前 | 替换后 |
|---|--|
|  |  |

此外，将步继电器(S)及带块指定步继电器(BL□\S□)的位数指定在程序中使用的情况下，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 |
|---|--|
|  |  |

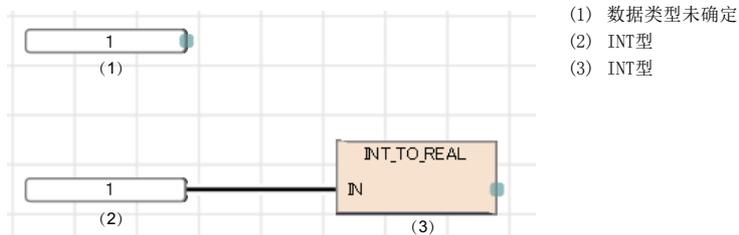
n 常数部件的数据类型

常数部件的情况下，在输入常数值时，不可以确定常数值的数据类型。通过连接线将常数部件与FBD部件连接在一起时，数据类型被确定。常数值的数据类型变为与通过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的情况下

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，因此不可以确定数据类型。通过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。



n 数据类型的自动转换

被连接的部件的数据类型不同时，可能会有自动转换数据类型的情况。

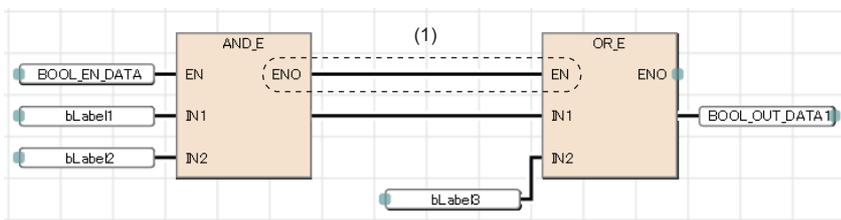
类型转换时，为了确保不丢失数据，只从容量小的数据类型转换至容量大的数据类型。FBD/LD语言中数据类型的自动转换的动作与ST语言相同。详细内容，请参阅下述章节。

☞ 58页 数据类型的自动转换

n 功能的输入输出点

- 功能需要预先将输入点全部与其它的FBD部件接线。
- 功能的输入变量与输出变量确定数据类型，连接至输入点与输出点的FBD部件也需要与此相符合。
- 将CPU模块用指令、模块专用指令的输出变量(除去ENO)连接至其它功能(或功能块)的输入变量时，应经由变量部件。
- 从带EN功能接线至功能的程序中，为了确保功能不使用不定值，应将功能作为带EN功能且置为将ENO与EN接线的程序。

(1) 将ENO与EN接线。



LD部件

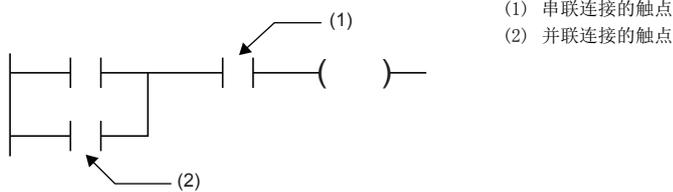
在FBD/LD程序中可使用的梯形图部件如下所示。

| 项目 | | 内容 |
|---------|--|--|
| 左母线 | | 是表示母线的部件。变为创建梯形图电路时的起点。 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。 |
| 反转型线圈 | | 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。 |

n 触点符号的AND运算与OR运算

触点符号根据梯形图形的连接状态，进行AND运算、OR运算，并反映至运算结果。

- 串联连接(1)时，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)时，进行之前的运算结果与OR运算，并作为运算结果。



n 将其它部件连接到线圈的输出连接点上的情况下

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 |  | 变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。 对成对的连接器部件，处理将转移。 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 |  | 中断程序上返回部件以后的处理。不执行返回部件以后的程序以及功能、功能块处理的情况下使用。 根据至返回部件的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。

n 关于跳转部件

- 在跳转部件中使线圈处于ON状态的定时器跳转时，将变得无法进行正常的测量。
- 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，应包括从循环进行脱落的方法在内创建程序，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。不可以使用指针软元件。
- 不可以使用指针分支指令(CJ、SCJ、JMP)。进行跳转的情况下，请勿使用跳转部件。
- 不可以进行至程序块外侧的跳转及来自于外侧的跳转。

| 跳转关联的动作 | 执行可否 |
|---------------|-------|
| 至程序块外侧的跳转*1 | 不可以执行 |
| 来自于程序块外侧的跳转*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 包含根据BREAK指令的分支。

n 关于返回部件

- 返回部件的动作根据使用的程序及功能、功能块而有所不同。

| 所使用的程序部件 | 内容 |
|----------|-----------------------|
| 程序 | 结束程序部件的执行。 |
| 功能 | 结束功能，返回调用了功能指令的下一步。 |
| 功能块 | 结束功能块，返回调用了功能块指令的下一步。 |

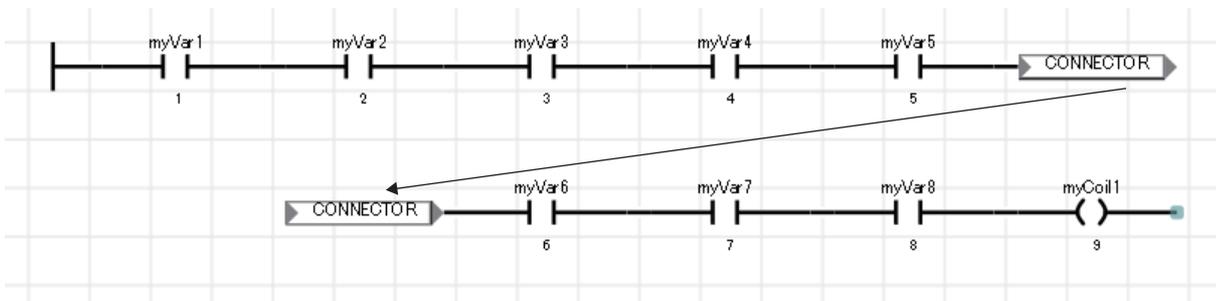
- “FB类型”在“宏型”功能块中使用返回部件的情况下，请勿对相同FB实例名的功能块部件进行多个配置。
- 转换使用了返回部件的程序时，局部标签将被自动地登录。对该标签，有下述操作限制。

| 对被自动登录的标签进行操作 | 操作可否 |
|---------------|---------|
| 标签名的更改 | 不可以操作*1 |
| 数据类型的更改 | 不可以操作 |
| 分类的更改 | 不可以操作 |
| 标签的删除 | 不可以操作*1 |
| 登录行的更改 | 可以操作 |

*1 更改或删除后再次进行转换时，局部标签将被重新登录。

n 关于连接器部件

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过渡数据。需要使处于连接状态的部件的数据类型一致或使数据类型为可进行自动转换的类型。

连接点

是根据连接线连接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。

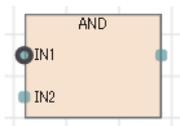
| 项目 | 输入连接点 | 输出连接点 |
|-----|-------|-------|
| 触点 | | |
| 线圈 | | |
| 变量 | | |
| 常数 | — | |
| 功能 | | |
| 功能块 | | |

连接点被接线时，将不显示功能块。

n 输入输出点的反转

通过连接点可对至部件的输入或来自于部件的输出进行反转。

用黑圆圈括住反转状态的连接点，反转进行输入或输出的数据 (FALSE→TRUE或TRUE→FALSE)。



可反转数据类型为BOOL、WORD、DWORD、ANY_BIT、ANY_BOOL。

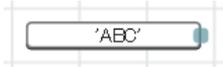
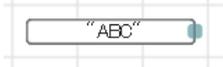
工作表

工作表是用于插入、接线程序部件的作业领域。

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，请参阅下述章节。

☞ 43页 常数

标签与软元件

指定方法

在FBD/LD程序上，可以直接记述标签与软元件使用。对部件的输入点、输出点、通用功能/功能块的自变量、返回值等可以使用标签与软元件。

关于可使用标签的详细内容，请参阅下述章节。

☞ 33页 标签

关于可使用软元件的详细内容，请参阅下述手册。

📖 MELSEC iQ-R CPU模块用户手册 (应用篇)

n 带类型指定软元件表记

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。

☞ 66页 带类型指定软元件表记

不指定字软元件的数据类型时，数据类型由软元件的类型确定。

| 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

n 使用标签的情况下

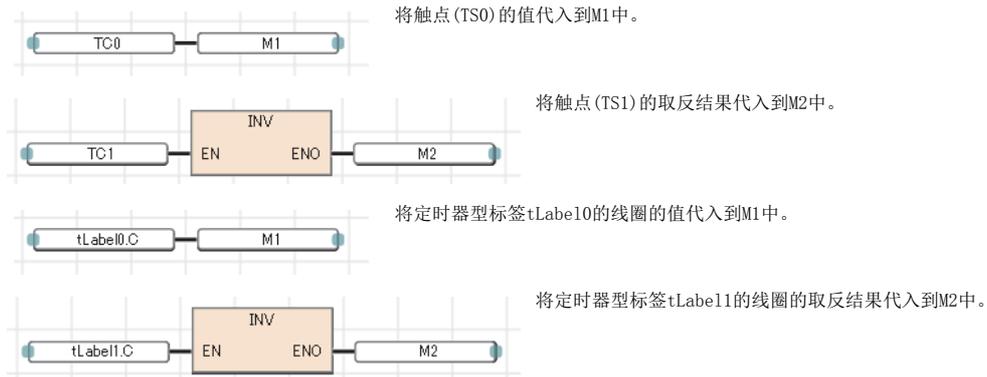
- 在排列的下标中，不能使用局部软元件。但是，将下标中使用的软元件代入其它软元件中，将代入目标的软元件指定为下标时将变为可以使用。

n 使用定时器、计数器的情况下

- 定时器及计数器的软元件的线圈 (TC、STC、LTC、LSTC、CC、LCC) 作为代入及功能、功能块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

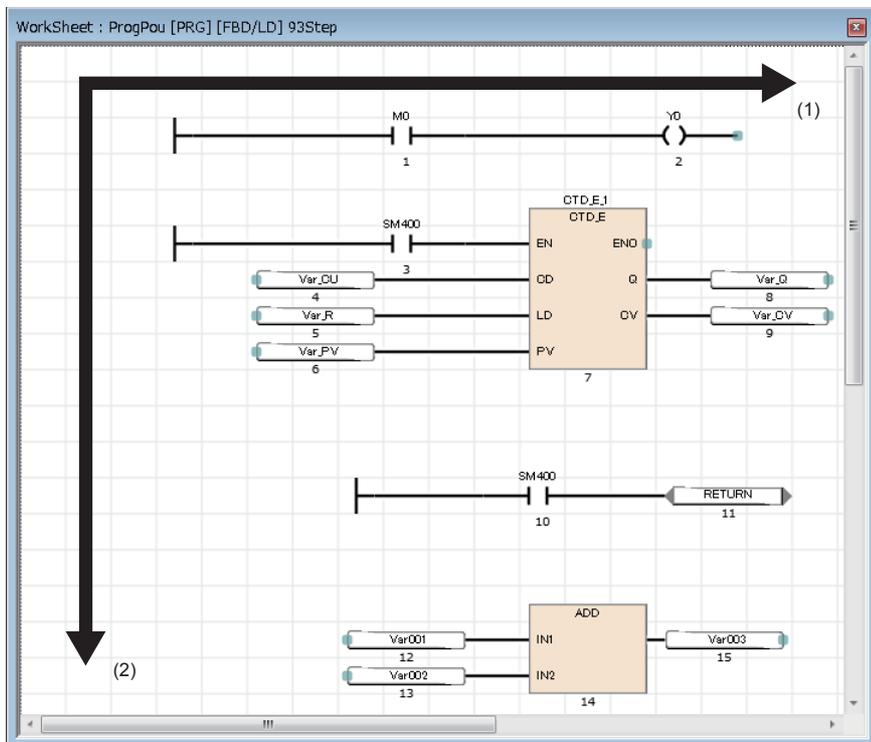
定时器软元件及定时器型标签的情况下



7.2 程序执行顺序

部件的执行顺序

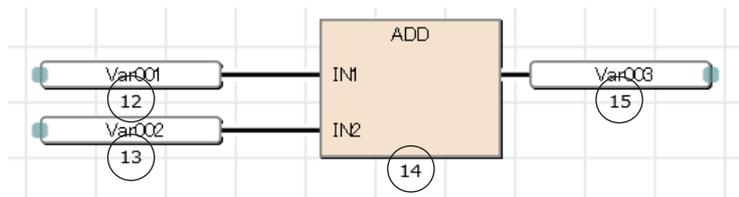
FBD/LD编辑器上部件的执行顺序由部件位置关系与接线状况决定。



(1) 从左至右执行。

(2) 从上至下执行。

对配置在FBD/LD编辑器上的各部件，将显示执行顺序的编号。



8 SFC程序

是将一系列的控制动作分割为多个步，以能够清楚地表示各程序的执行顺序及执行条件的程序的记述形式。

要点

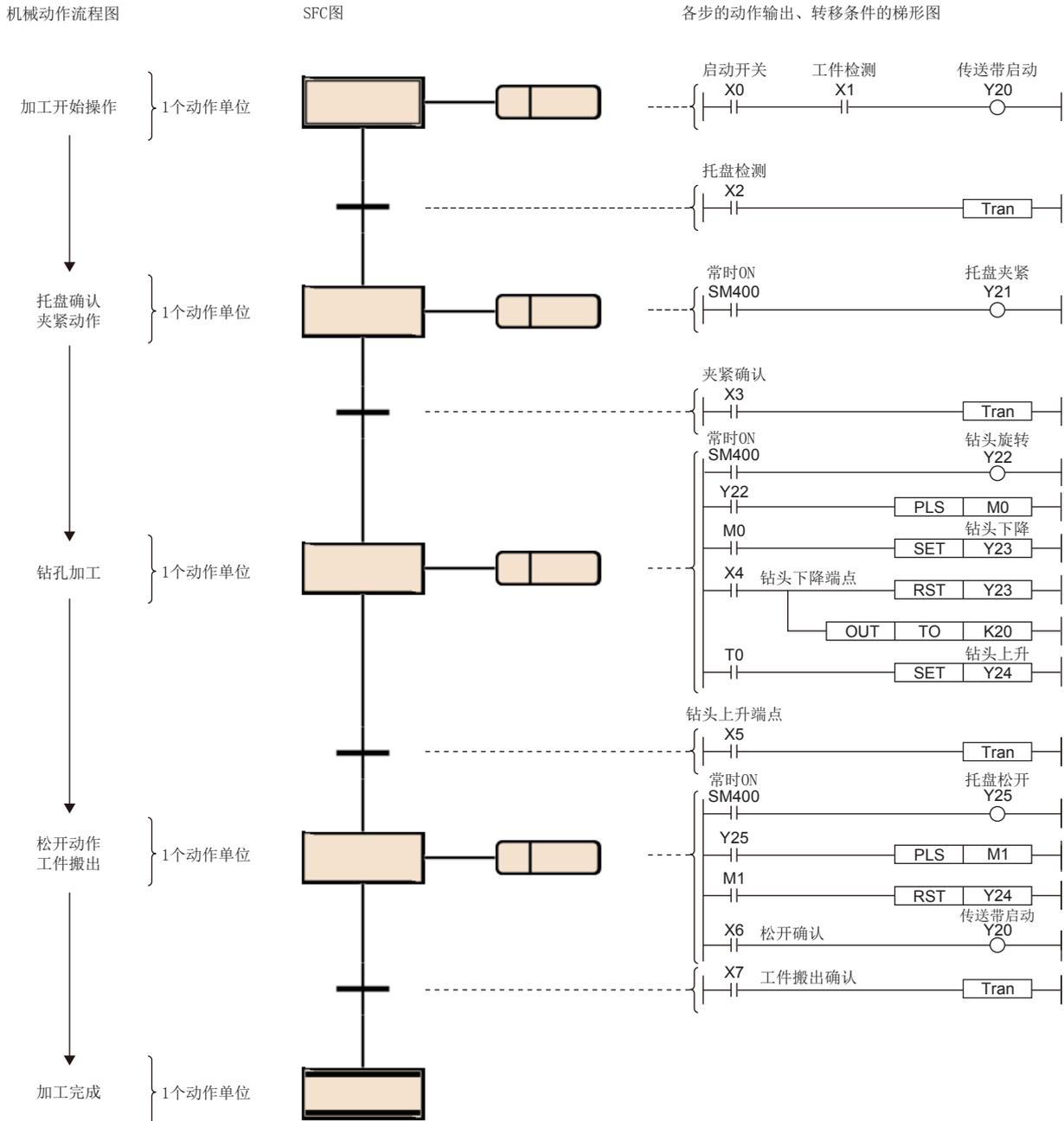
• 在本章中，对SFC程序的动作及规格有关内容进行说明。关于本章中未记载的内容，请参阅下述手册。

📖 GX Works3操作手册

📖 MELSEC iQ-R CPU模块用户手册(应用篇)

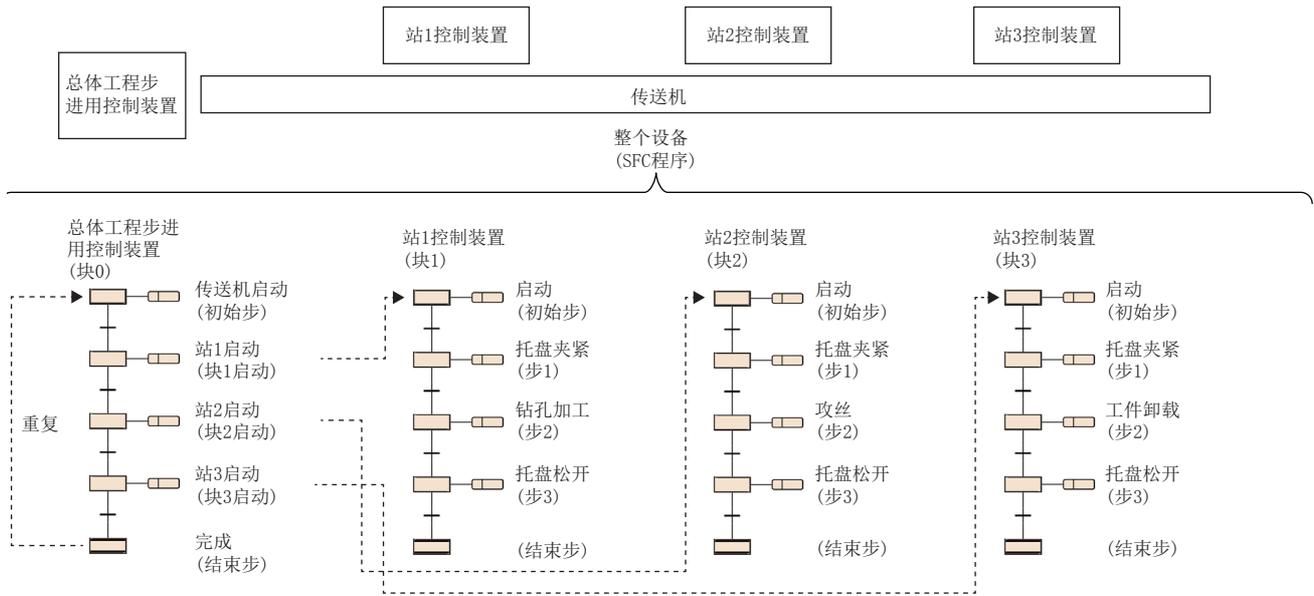
SFC程序将机械的一系列动作中的各动作单位表示为1个步。

在各步中对实际的精细控制的程序进行创建。



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

可以将整个设备、各站的机械装置、各机械的实际控制与SFC程序的各块、各步以1对1对应。



8.1 规格

SFC程序相关的性能规格如下所示。

| 项目 | 规格 | |
|-----------|--------------------------|---------------------------|
| SFC程序执行个数 | 1个 | |
| 块数 | 最多320块 | |
| SFC步数 | 所有块最多16384步 1个块最多512步 | |
| 分支数 | 最多32分支 | |
| 同时激活步数 | 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 (无每1步的限制) |
| | 转移条件 | 仅1个梯形图块 |

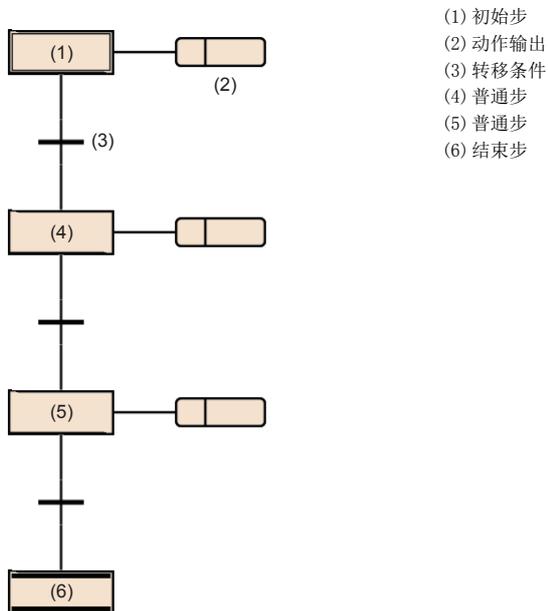
要点

- 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。



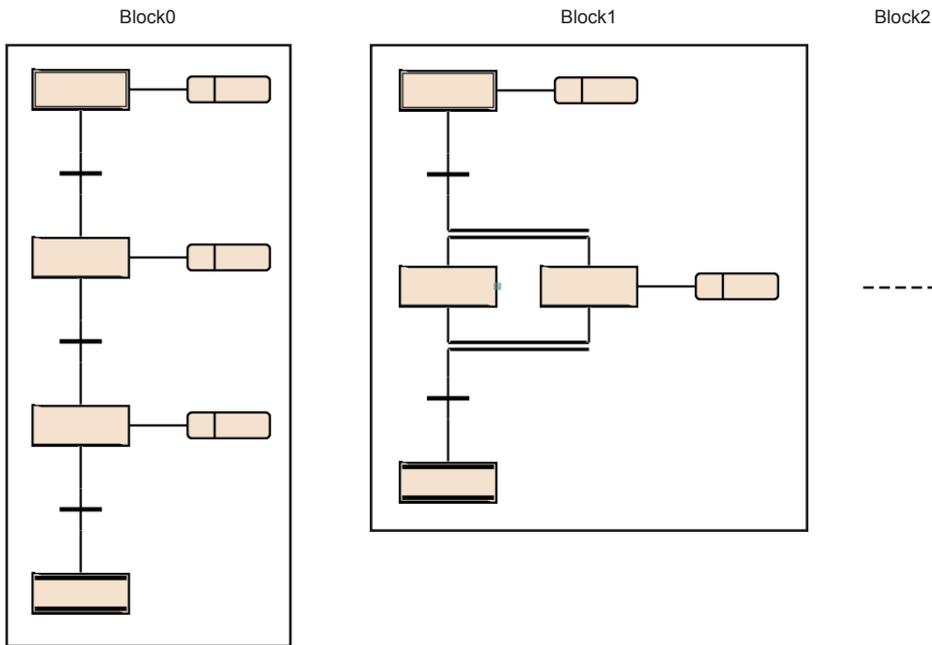
1. 块启动时，首先执行初始步(1)的动作输出(2)。初始步(1)的动作输出(2)执行后将检查下一个转移条件(3)是否成立。
2. 在转移条件(3)成立之前，仅执行动作输出(2)。转移条件(3)成立时将结束初始步(1)的动作输出(2)，执行下一个普通步(4)。
3. 在执行普通步(4)的动作输出后，检查下一个转移条件是否成立。如果下一个转移条件成立，将重复普通步(4)的动作输出的执行。
4. 转移条件成立时将结束步(4)的动作输出，执行下一个步(5)。
5. 每当转移条件成立时将执行下一个步，最后转移到结束步(6)时结束块。

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。(☞ 95页 动作输出)

块

块由步及转移条件构成，是表示一系列动作的单位。



SFC程序内最多可创建320个块。

块内变为从初始步开始，交互连接步及转移条件，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

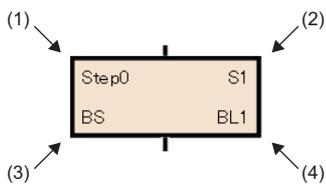
块从非激活变为激活时，初始步将变为激活，依次执行处理。（☞ 125页 各块的执行顺序）

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。在这种情况下转移到结束步结束块0时，将自动被再启动并再次从初始步开始执行。（☞ 119页 启动条件设置）
- 非激活块的步中存在启动请求的情况下，通过SET指令（步启动）使块激活后，从指定的步开始执行处理。

步

步是指用于构成块的基本单位。



- (1) 步名
- (2) 步No.
- (3) 属性
- (4) 属性指定目标

步有下述特征。

- 步激活时，执行相关的动作输出。
- 对于步，1个块中最多可以创建512个。
- 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。
(☞ 94页 至步继电器(S)的步的分配)
- 步名及步No. 在各块内是特有的。(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进行更改。

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

步的一览如下所示。

| 项目 | | 内容 |
|-------------------|--|--|
| 普通步 | | 无属性的步。 也可以置为不创建动作输出的步。 |
| 初始步 | | 表示块的起始的步。 也可以置为不创建动作输出的步。 |
| 线圈保持步 [SC] | | 转移条件成立时，在保持通过OUT指令变为ON的线圈的输出状态下，进行转移的步。 |
| 动作保持步(无转移检查) [SE] | | 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。 |
| 动作保持步(有转移检查) [ST] | | 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。 |
| 复位步 [R] | | 将指定步置为非激活的步。 |
| 块启动步(有结束检查) [BC] | | 使指定块激活的步。 指定块变为非激活且转移条件成立时，将转移到下一个步。 |
| 块启动步(无结束检查) [BS] | | 使指定块激活的步。 转移条件成立时，将转移到下一个步。 |
| 结束步 | | 使块结束的步。 |

要点

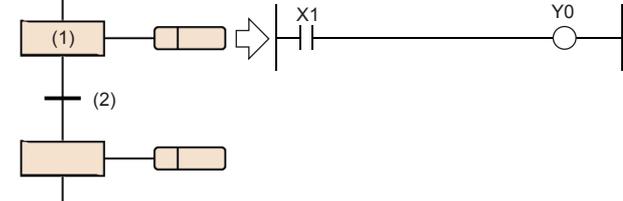
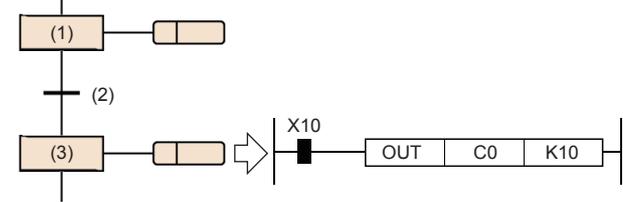
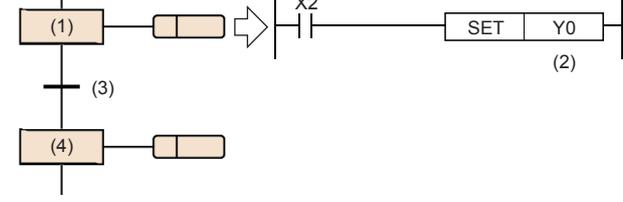
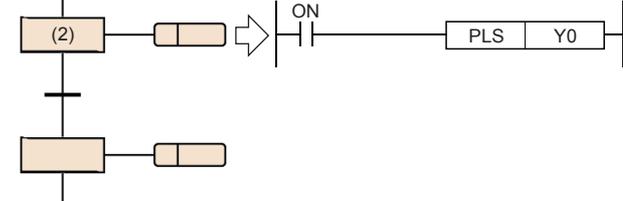
- 通过在步的属性画面中，对“属性”的设置进行更改可以更改步的类型。
- 对于复位步 [R]、块启动步(有结束检查) [BC]、块启动步(无结束检查) [BS]，在属性画面的“属性指定目标”中对步名或块No. 进行指定。

关于设置方法有关内容，请参阅下述手册。

GX Works3操作手册

普通步(无属性)

普通步(无属性)是步处于激活中时, 始终对该步的下一个转移条件进行检查, 转移条件成立时下一个步中转移激活的步。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。 |  <p>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。</p> |
| 使用OUT C指令时 | 位于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。 |  <p>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。</p> |
| 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。 |  <p>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的ON仍将被保持。</p> |
| 使用PLS指令、上升沿执行指令时 | 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。 |  <p>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。</p> |

n 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。

- 步的激活过程中, 始终对转移条件进行检查, 转移条件成立时下一个步中对激活进行转移。
- 创建动作输出时, 将作为普通的步进行动作。

初始步

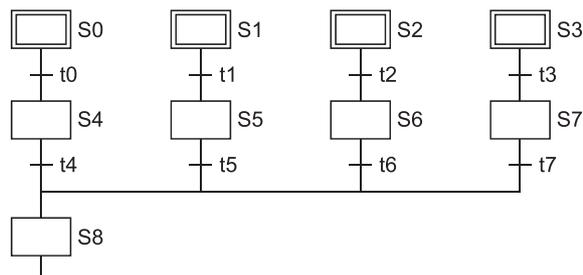
初始步是表示各块的起始的步，各块中最多可记述32个。(P81页 规格) 对多个初始步进行合并时只能进行选择合并。初始步的执行方法与初始步以外的步相同。

n 块启动时的激活步

在初始步为多个步的情况下，块中开始启动时的激活步根据启动方法将变为如下所示。

| 激活步的动作 | 启动方法 |
|----------|---------------------------|
| 初始步全部被激活 | 通过块启动步开始启动时 |
| | 通过SFC控制指令的块启动指令开始启动时 |
| | 通过SFC用信息软元件的块启动结束位强制开始启动时 |
| | 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的某个时 |

n 初始步进行了多个步激活时的转移处理



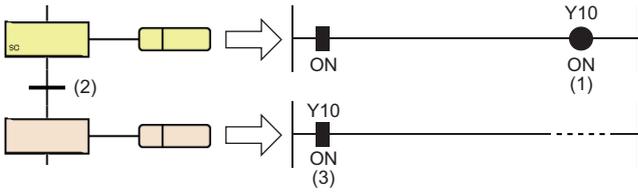
对初始步进行了多个步激活的块执行了选择合并的情况下，如果合并之前的某个转移条件成立，则合并之后的步将被激活。在上述程序示例中，如果转移条件t4~t7的某个成立则步8(S8)将被激活。此外，合并之后的步(在上述程序示例中为S8)被激活后，合并之前的其它转移条件(在上述程序示例中为t4~t7)成立时，合并之后的步将再次被激活。

n 初始步中附加步属性时的动作

对于初始步，可以对线圈保持、动作保持(无/有转移检查)、复位的各属性进行附加。此外，附加了属性的情况下，块启动时自动被激活的动作以外与初始步以外的步相同。也可置为无动作输出的步。

线圈保持步[SC]

线圈保持步是转移条件成立的情况下，在保持通过OUT指令变为ON的线圈输出状态下进行转移的步。



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。

n 线圈OFF的时机

在转移后的线圈保持步中，保持为ON状态的线圈变为OFF的时机如下所示。

- 执行了块的结束步的情况下 (SM327为ON时除外)
- 通过SFC控制指令的RST指令(块结束)对块进行了强制结束的情况下
- 通过SFC控制指令的RST指令(步结束)对步进行了复位的情况下
- 对SFC用信息软件的块启动结束位中指定的软件进行了复位的情况下
- 所设置的用于复位线圈保持步的复位步被激活的情况下
- 将SM321(SFC程序的启动/停止)置为了OFF的情况下
- 通过程序对线圈进行了复位的情况下
- 在停止时输出模式为OFF的状态下开始了停止指令的情况下
- 在块内的复位步中指定了S999的情况下

n 块停止/重启时的动作

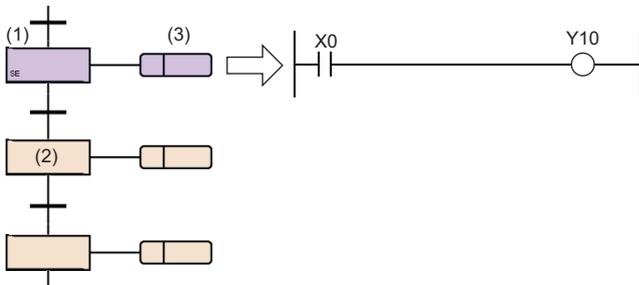
块停止/重启时的动作，根据SM325(块停止时的输出模式设置)与SFC用信息软件的停止时模式位的设置、步的保持/非保持的组合确定。(☞ 120页 块停止重启时的动作)

动作保持步(无转移检查)[SE]

动作保持步(无转移检查)是转移后仍继续进行动作输出的运算的步。

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，但是动作输出(3)将继续执行。
在这种情况下，根据X0的ON/OFF，Y10将变为ON/OFF。

n 变为非激活的时机

动作保持步(无转移检查)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下
- 通过SFC控制指令的RST指令(块结束)对块进行了强制结束的情况下
- 通过SFC控制指令的RST指令(步结束)对步进行了复位的情况下
- 对SFC用信息软件的块启动结束位中指定的软件进行了复位的情况下
- 所设置的用于复位动作保持步(无转移检查)的复位步被激活的情况下
- 将SM321(SFC程序的启动/停止)置为了OFF的情况下
- 在块内的复位步中指定了S999的情况下

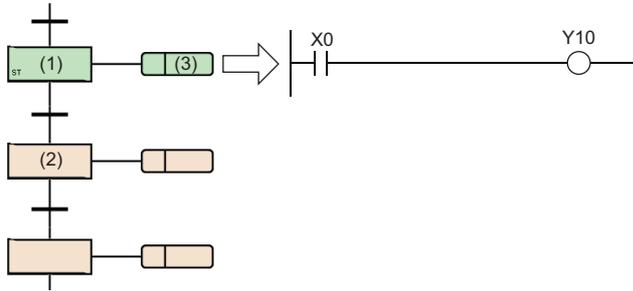
n 块停止/重启时的动作

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用信息软件的停止时模式位的设置、步的保持/非保持的组合确定。(☞ 120页 块停止重启时的动作)

动作保持步(有转移检查)[ST]

动作保持步(有转移检查)是转移后仍继续进行动作输出的运算的步。

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, 再次使下一个步激活的同时, 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。

n 变为非激活的时机

动作保持步(有转移检查)变为非激活状态的时机如下所示。

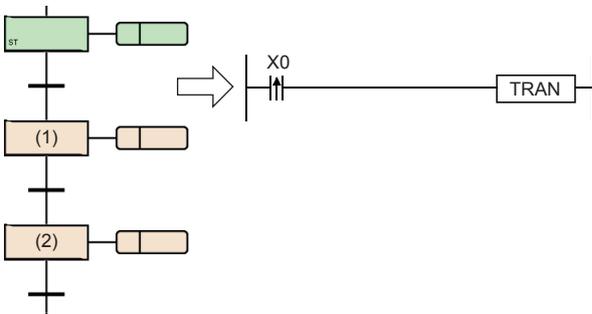
- 执行了块的结束步的情况下
- 通过SFC控制指令的RST指令(块结束)对块进行了强制结束的情况下
- 通过SFC控制指令的RST指令(步结束)对步进行了复位的情况下
- 对SFC用信息元件的块启动结束位中指定的软元件进行了复位的情况下
- 所设置的用于复位动作保持步(无转移检查)的复位步被激活的情况下
- 将SM321(SFC程序的启动/停止)置为了OFF的情况下
- 在块内的复位步中指定了S999的情况下

n 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用信息元件的停止时模式位的设置、步的保持/非保持的组合确定。(120页 块停止重启时的动作)

n 注意事项

- 对于动作保持步(有转移检查), 之后的转移条件成立期间, 每个扫描下一个步将被启动。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不再变为OFF→ON步(1)将不启动。

- SM328(END步到达时清除处理模式)为ON的情况下, 应置为动作保持步(有转移检查)之后的转移条件不常时成立。下一个步始终变为非保持的激活状态, 因此不可以结束块。

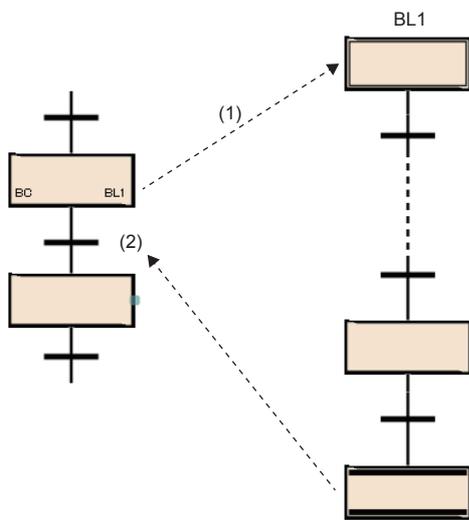
复位步[R]

复位步是将指定步置为非激活的步。

- 复位步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，将本块内保持中的线圈保持步、动作保持步(无转移检查)、动作保持步(有转移检查)全部置为非激活状态。在这种情况下，仅保持中的步可进行非激活。保持步以非保持进行动作时，将不变为非激活的对象。
- 本步No. 不可以指定到指定步No. 中。

块启动步(有结束检查)[BC]

块启动步(有结束检查)是指启动指定块，之后在启动目标块变为非激活状态时，开始进行至下一个步的转移条件的检查的步。



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个块同时进行了启动的情况下及，对已启动的块进行了启动时的动作，按照块重复启动时的运行设置进行。(☞ 122页 块双重启动时的运行设置)

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

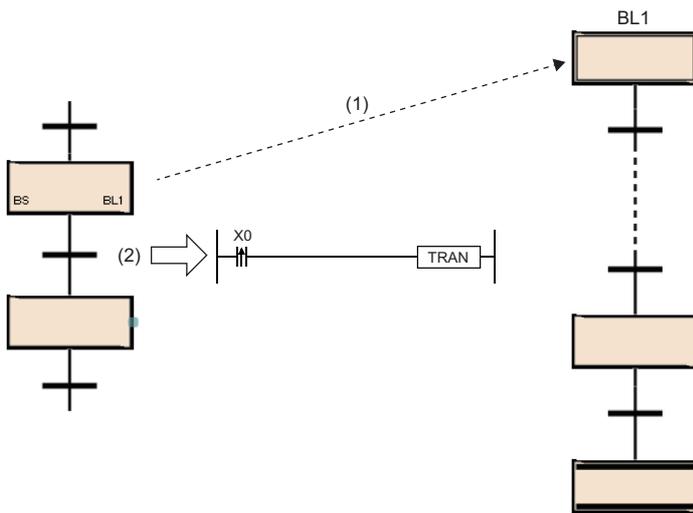
n 注意事项

- 不可以将动作输出创建到块启动步(有结束检查)中。
- 在并行合并的合并之前不可以创建块启动步(有结束检查)。在并行合并的合并之前进行创建的情况下，使用块启动步(有结束检查)。

块启动步(无结束检查) [BS]

块启动步(无结束检查)是指,启动指定块后,与启动目标块的激活状态无关,进行至下一个步的转移条件的检查的步。

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,如果条件成立则不进行启动目标块(BL1)的结束等待,转移至下一步。



对1个块同时进行了启动的情况下及,对已启动的块进行了启动时的动作,按照块重复启动时的运行设置进行。(☞ 122页 块双重启动时的运行设置)

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

n 注意事项

- 不可以将动作输出创建到块启动步(无结束检查)中。

结束步

结束步是使块内的一系列处理全部结束的步。

- 转移到结束步的情况下，块内不存在保持中以外的激活步时，将块内的全部保持中步置为非激活状态后，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 | 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指令的线圈输出全部置为OFF。但是，对于保持中步的线圈输出，根据SM327 (END步执行时的输出) 的状态进行下述处理。

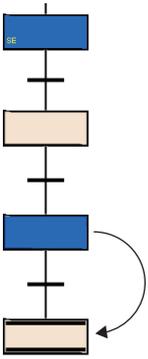
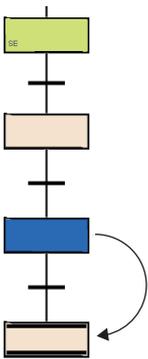
| SM327的状态 | 内容 |
|----------|---|
| OFF | 将保持中步的输出全部置为OFF。 |
| ON | 将保持中步的输出全部保持。 仅保持中的保持步SM327的设置有效。转移条件未成立且不处于保持中的保持步的输出将全部OFF。此外即使SM327为ON时，步也将由激活状态变为非激活状态。 但是，通过块结束指令等进行强制结束的情况下，所有步的线圈输出将OFF。 |

- 块结束后使块再次启动的方法如下所示。

| 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“进行自动启动” |
| | 在参数的SFC设置中将块0的启动条件设置为“不进行自动启动” |
| 块0以外的所有块 | 自动地再次对初始步进行激活，重复执行处理。 通过下述方法，在存在有对指定块的启动请求时进行再启动。 <ul style="list-style-type: none"> • 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用信息软件的块启动结束位置为ON。 |

n 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该保持中步将变为非激活状态。不希望将保持中步的线圈输出置为OFF的情况下，应将SM327置为ON。SM328与保持步的动作的关系如下所示。

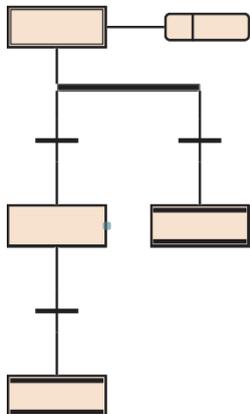
| 普通的激活步剩余的情况下，或转移不成立的保持步剩余的情况下 (非保持) | 保持中的激活步剩余的情况下 |
|---|---|
|  <ul style="list-style-type: none"> • SM328为OFF时，进行清除后结束块。 • SM328为ON时，不进行清除而继续进行处理。 |  <ul style="list-style-type: none"> • 与SM328的设置无关，进行清除后结束块。 |

- SM328为0N的情况下，通过块启动步启动的块中，在非保持的激活中步不存在于块内的时刻，将返回到原来的块处理。
- 应置为动作保持步(有转移检查)之后的转移条件不常时成立。动作保持步(有转移检查)之后的转移条件常时成立的情况下，下一个步始终变为激活形态，因此SM328为0N时可以不结束块。

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，选择菜单的[编辑]⇒[更改]⇒[结束]时，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被分配。

- 对于步继电器, 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进行分配。
- 对于不存在的块No., 不分配到步继电器中。
- 在1个块内, 缺少编号的步No. 由步继电器进行分配。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

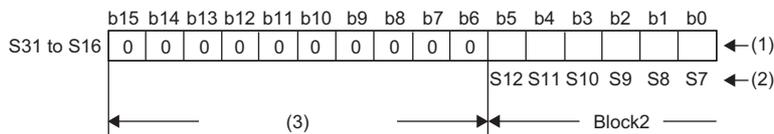
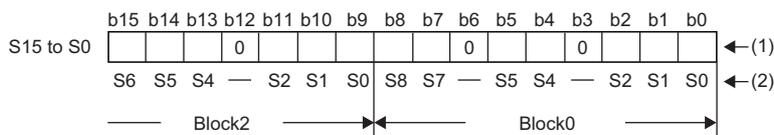
下述块配置时的步继电器分配如下所示。

Block0: 最大步No. 为8, 且步No. 3及步No. 6不存在。

Block1: 不存在。

Block2: 最大步No. 为12, 且步No. 3不存在。

Block3以后: 不存在。



(1) 存储的数据

(2) 块内的步No.

(3) 由于不存在全部为0

要点

对于各步(除结束步以外), 可以自由分配步No.。

- 步No. 中缺少编号时, 可创建的最大步数变少, 因此应尽可能以小编号顺序进行创建。
- 在最上行的左端初始步中, 不可以使用步No. 0(S0)以外。

对于块的最初初始步, 块No. 0被分配。

对于步No. 每1个块可以在0~511的范围内使用。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的情况下

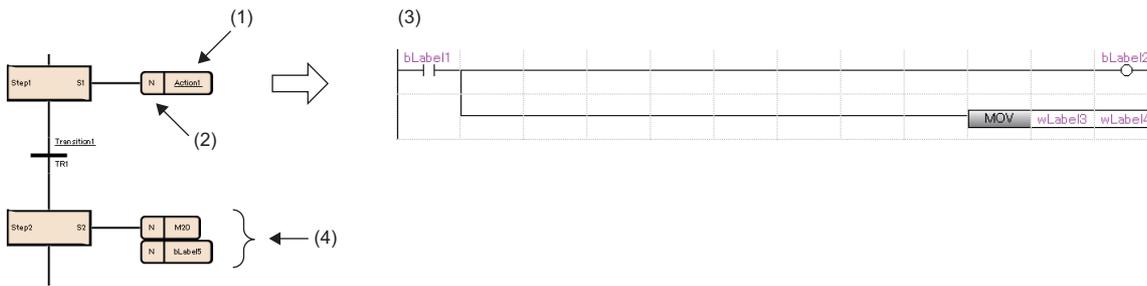
| 程序类型 | | 软元件表记 | 内容 |
|--------------|------------------|----------|-----------------------|
| SFC程序 | 同一块内 | S23 | 指定同一块内的步的情况下, 可以省略块名。 |
| | 块12以外 | BL12\S23 | 指定块No. 及步No.。 |
| SFC程序以外的顺控程序 | 指定当前对象块的情况下 | S23 | 指定对象块内的步的情况下, 可以省略块名。 |
| | 指定与当前对象块不同的块的情况下 | BL12\S23 | 指定块No. 及步No.。 |

n 注意事项

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

动作输出

对于动作输出，步在激活中被执行的程序如下所示。



- (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表示步在激活中被执行。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。步进行非激活时，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。

要点

关于详细表示及标签/软元件的详细内容，请参阅下述手册。

GX Works3操作手册

无法使用的指令

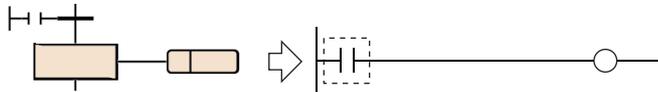
在动作输出内一部分的指令不可以使用。无法使用的指令如下所示。

| 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 |
| 程序执行控制指令 | IRET |
| 结构化指令 | BREAK* ¹ |
| | RET |

*1 在动作输出内的功能/功能块内可以使用。

要点

对于详细表示内的梯形图，必须对各指令的输入条件的触点进行创建。



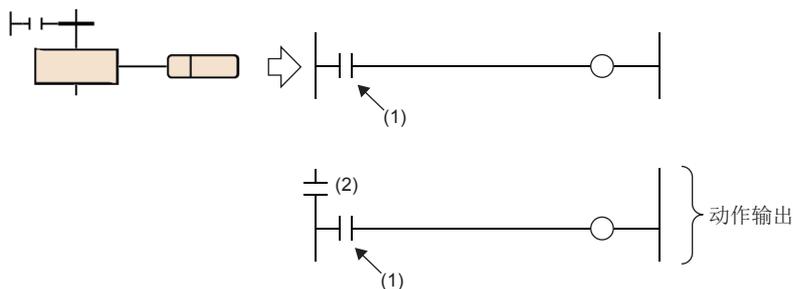
n 限制事项

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 | <p>指针及中断指针，不可以输入到指针输入区中。</p> <p>n 无法使用的功能/功能块</p> <ul style="list-style-type: none"> • 动作输出中包括无法使用的指令的功能/功能块 • 包括指针的功能/功能块 • “EN的控制中使用MC/MCR”为“是”，“使用EN/ENO”为“否”且“FB的类型”为“宏型”的FB |
| ST语言 | ☞ 54页 ST语言 |
| FBD/LD语言 | ☞ 68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OFF。在转移条件成立步变为非激活状态时希望使CALL目标的输出OFF的情况下, 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为常时ON, 步为非激活状态时及动作输出OFF被执行后, 被视为输入条件为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指令等的上升沿执行指令中, 将输入条件置为了常时ON的情况下, 每当步被激活时执行指令。
- 在OUT C指令、SET指令、基本指令及应用指令等中, 通过动作输出的OFF软元件不变为OFF。将软元件置为OFF时需要另外执行RST指令等。
- 对于PLS指令及PLF指令, 通常指定软元件仅1个扫描变为ON之后变为OFF, 但是与线圈保持步的转移成立的同时指定软元件变为了ON的情况下, 将继续保持ON状态。在这种情况下, 通过在该线圈保持步中执行OFF或, 其它步等执行RST指令, 再次对步进行激活, 将变为OFF。
- 在PLF指令的输入条件为ON的情况下, 如果执行OFF, 指定软元件将保持为ON状态不变。
- 线圈保持步中转移条件成立的情况下及SM325(块停止时的输出模式)为保持的设置中停止了步的情况下, 仅保持线圈输出时不进行动作的情况下, 处于非执行的状态, 因此动作重启时的各指令的动作被非执行前的执行条件所左右。

转移条件

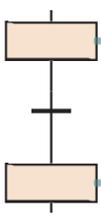
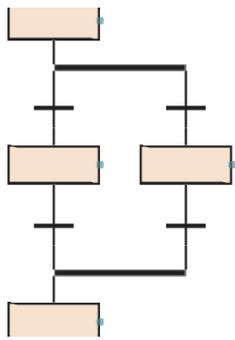
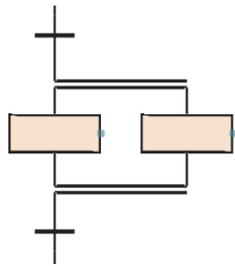
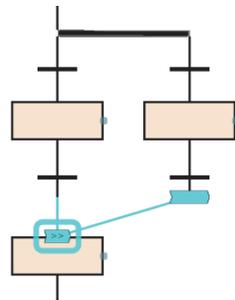
转移条件是用于转移到下一个步的条件，通过条件成立转移到下一步。



- (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 (☞ 105页 转移条件的直接表示)
- (5) 转移条件的标签/软元件 (☞ 105页 转移条件的标签/软元件)

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，则从现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转移。 合并：转移条件最先成立的列的合并之前的转移条件成立时，将转移至下一个步。 |
| 并行转移(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转移。 合并：如果合并之前的步全部被激活，则通过通用的转移条件成立转移至下一个步。 |
| 跳转 |  | 通过转移条件成立，转移至同一块内的指定的步。 |

要点

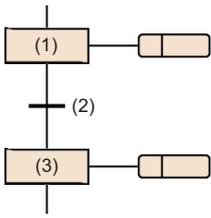
关于转移至已激活的步时的动作有关内容，请参阅下述章节。

 132页 步双重启动时的注意点

串行转移

串行转移是指对于1个步，下一个步在转移条件中串行结合的连接。通过转移条件成立，激活步转移到下一个步。

执行步(1)的动作输出时，如果转移条件(2)成立，则将步(1)的动作输出置为非执行后，执行步(3)的动作输出。



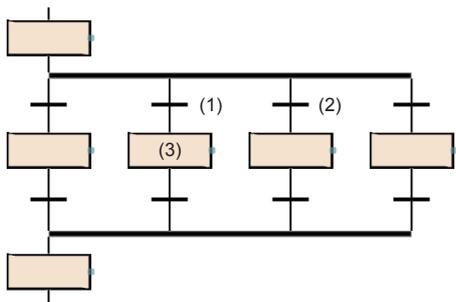
选择转移(分支/合并)

选择转移是指从1个步，在多个转移条件及与其连接的步中进行分支的连接。对于激活步，在多个转移条件之中仅最先成立的步进行转移。

| 项目 | 内容 |
|----|--|
| 分支 | <p>The diagram shows a vertical line starting with a box labeled (1). From the bottom of box (1), the line splits into two horizontal paths. The left path goes through a transition labeled (2) to a box labeled (4), which then leads to a box labeled (6). The right path goes through a transition labeled (3) to a box labeled (5), which then leads to a box labeled (7). Below boxes (4) and (5), there are two separate vertical lines, each ending in a small horizontal bar.</p> <p>执行步(1)的动作输出时，选择转移条件(2)或转移条件(3)之中条件先成立的步((4)或(5))，执行该步的动作输出((6)或(7))。</p> <ul style="list-style-type: none"> • 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。 • 步(1)的动作输出将变为非执行。但是，保持步的情况下，将按照属性继续进行动作。 • 选择后，依次执行所选择列的各步直至进行合并为止。 |
| 合并 | <p>The diagram shows two parallel vertical lines. The left line starts with a box labeled (3) and the right line starts with a box labeled (4). From the bottom of box (3), a horizontal line goes to the right and then down to a transition labeled (1). From the bottom of box (4), a horizontal line goes to the left and then down to a transition labeled (2). From the center of transition (1), a horizontal line goes to the right and then down to a box labeled (5). From the center of transition (2), a horizontal line goes to the left and then down to the same box labeled (5). Below box (5), there is a vertical line ending in a small horizontal bar.</p> <p>如果在分支中执行一方的列的转移条件((1)或(2))成立，将执行的步的动作输出((3)或(4))置为非执行(但是保持步按照该属性)，执行步(5)的动作输出。</p> |

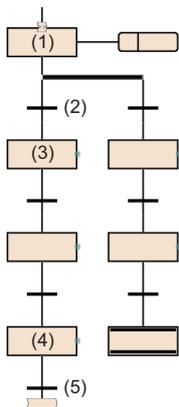
- 在选择转移中，最多可以分支为32个转移条件。
- 多个转移条件同时成立的情况下，将优先执行左侧的转移条件。

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行合并及并行分支与选择合并组合的SFC图。
- 在选择转移中，可以通过跳转及结束步对合并进行省略。

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，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

并行转移(分支/合并)

并行转移是从1个转移条件转移到多个步的连接。如果转移条件成立，进行了合并的步中，激活同时进行转移。

| 项目 | 内容 |
|----|--|
| 分支 | <ul style="list-style-type: none"> • 执行步(1)的动作输出时，如果转移条件(2)成立，则同时执行步(3)的动作输出、步(4)的动作输出。 • 步(1)的动作输出将变为非执行。但是，保持步的情况下，将按照属性继续进行动作。 • 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 | <ul style="list-style-type: none"> • 在执行步(1)与步(2)的动作输出时，如果转移条件(3)、转移条件(4)成立，将步(1)及步(2)的动作输出分别置为非执行(但是保持步按照该属性)，转移到步(5)、步(6)中。 • 通过将合并之前的步(步(5)及步(6))全部激活，检查转移条件(7)，如果转移条件(7)成立，则执行步(8)的动作输出。 |

- 在并行转移中，最多可以转移到32个步中。
- 通过并行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。
- 在并行分支之后，必须进行并行合并。

n 注意事项

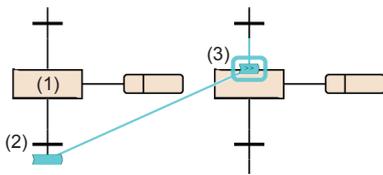
- 在并行合并中，合并的步中存在有保持中的步的情况下，将进行下述动作。

| 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。 |

- 在并行合并中，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

跳转是通过转移条件的成立，转移到同一块内的指定的步中的连接。

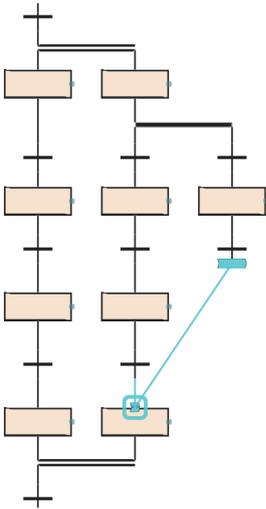


执行步(1)的动作输出时，如果转移条件(2)成立，将步(1)的动作输出置为非执行(但是保持步按照该属性)，执行步(3)的动作输出。

- 跳转的使用个数无限制。
- 并行转移内的跳转，仅在同一分支内进行。不可以创建至并行分支内的不同分支的跳转及从并行分支中脱离的跳转、从并行分支外至并行分支的跳转。

例

并行分支内可指定的跳转示例



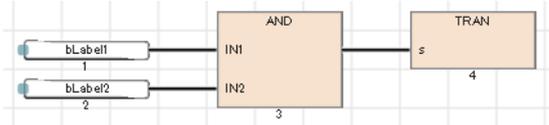
n 注意事项

下述情况下，不可以作为跳转的指定目标进行指定。

- 指定了从并行转移中脱离的位置中的步的情况下
- 指定了进入并行转移的位置中的步的情况下
- 指定了先行的转移条件之前的步的情况下
- 指定了本步的情况下

转移条件的详细表示

将下一个步中使激活转移的条件在Zoom编辑器内进行创建。可以通过下述程序语言对条件进行创建。

| 类型 | 内容 |
|----------|--|
| 梯形图语言 | <p>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。</p>  <p>n 限制事项</p> <ul style="list-style-type: none"> • 不可以使用直接ST。 • 线圈时仅TRAN指令可以输入。 |
| ST语言 | <p>可以创建下述转移条件程序。</p> <p>n 对TRAN功能(转移条件虚拟输出)的调用语句进行记述的方法</p> <pre>TRAN(bLabel1 & bLabel2);</pre> <p>//输入自变量的BOOL式为真(TRUE)的情况下，转移条件成立。</p> <p>n 对保留字“TRAN”的BOOL式的代入语句进行记述的方法</p> <pre>TRAN := bLabel1 & bLabel2;</pre> <p>//右边的BOOL式为真(TRUE)的情况下，转移条件成立。</p> <p>n 对转移条件名的BOOL式的代入语句进行记述的方法</p> <pre>Transition1 := bLabel1 & bLabel2;</pre> <p>//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。</p> |
| FBD/LD语言 | <p>在单一网络中，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。</p>  <p>n 限制事项</p> <ul style="list-style-type: none"> • TRAN指令只能使用1个。 • 不可以创建代入至软元件/标签的程序。 • 不可以使用线圈部件、功能块部件、功能部件(一部分可以)、跳转部件/跳转标签部件、返回部件。 <p>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。</p> <p>☞ 104页 转移条件的详细表示</p> |

要点

- 相同转移条件的详细表示可以在多个转移条件中使用。
- 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)

n可使用指令

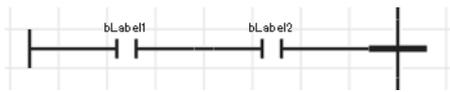
转移条件的程序中可使用的指令如下所示。

| 分类 | 指令符号 |
|---------|---------------------------------------|
| 触点指令 | LD、LDI、AND、ANI、OR、ORI |
| | LDP、LDF、ANDP、ANDF、ORP、ORF |
| | LDPI、LDPI、ANDPI、ANDFI、ORPI、ORFI |
| 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\$□ |

*1 在以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。

转移条件的直接表示

可以将下一个步中使激活转移的条件在SFC图上直接进行创建。对转移条件连接FBD/LD部件的触点。



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

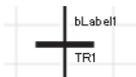
要点

通过选择转移条件后，对菜单的[编辑]⇒[更改]⇒[转移条件的直接表示]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)

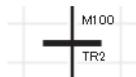
转移条件的标签/软元件

下一个步中使激活转移的条件中，可以指定位型标签及位软元件或BOOL值。

位型标签的情况下



位软元件的情况下



BOOL值的情况下



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)

n 注意事项

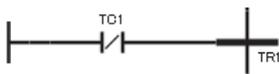
- 转移条件中使用了定时器及计数器的软元件(T、ST、LT、LST、C、LC)的情况下，将作为触点(TS、STS、LTS、LSTS、CS、LCS)进行动作。使用了定时器及计数器的软元件的线圈(TC、STC、LTC、LSTC、CC、LCC)的情况下，也同样作为触点进行动作。
- 希望在转移条件中使用定时器及计数器的线圈的情况下，应使用定时器型及计数器型的标签。

例

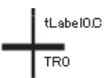
定时器软元件及定时器型标签的情况下



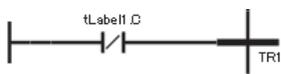
触点(TS0)为ON时，转移至下一个步。



触点(TS1)为OFF时，转移至下一个步。



定时器型标签tLabel0的线圈为ON时，转移至下一个步。



定时器型标签tLabel1的线圈为OFF时，转移至下一个步。

8.3 SFC控制指令

SFC控制指令是指，可以进行块、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程序的動作輸出內對SFC程序進行控制。

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 | 检查指定块的指定步是否被激活。 |
| | LD、LDI、AND、ANI、OR、ORI [BL□\S□] | |
| 块激活检查 | LD、LDI、AND、ANI、OR、ORI [BL□] | 检查指定块是否被激活。 |
| 激活步批量读取 | MOV(P) [K4S□]*1 | 将指定块的激活步状态作为位信息读取至指定软元件中。 |
| | MOV(P) [BL□\K4S□] | |
| | DMOV(P) [K8S□]*1 | |
| | DMOV(P) [BL□\K8S□] | |
| | BMOV(P) [K4S□]*1 | |
| BMOV(P) [BL□\K4S□] | | |
| 块启动 | SET [BL□] | 对指定块进行单独启动，并从初始步开始执行。 |
| 块结束 | RST [BL□] | 将指定块置为非激活。 |
| 块停止 | PAUSE [BL□] | 将指定块置为暂时停止状态。 |
| 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 | 启动指定步。 |
| | 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 (指令/通用FUN/通用FB篇)

n 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 |

对于步继电器及SFC块软元件，也包括进行变址修饰的情况，在下述范围内进行指定。

| 软元件 | 范围 |
|--------|-------------------------|
| S□ | 0~16383 (最大值通过CPU参数的设置) |
| BL□\S□ | BL□ 0~319 |
| | S□ 0~511 |
| BL□ | 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)

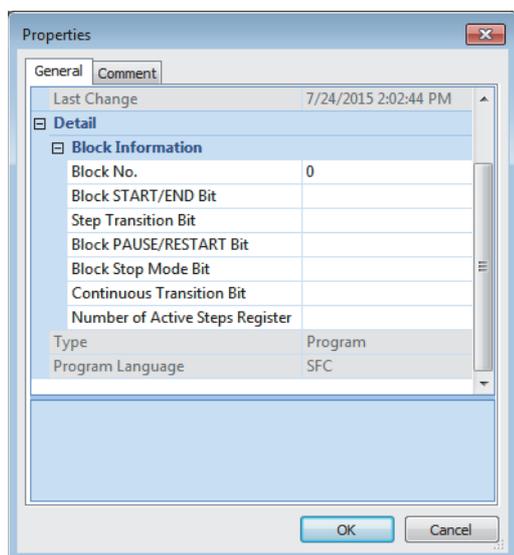
8.4 SFC用信息软元件

SFC用信息软元件是对块指示强制启动/结束与暂时停止/重启的指示以及步的转移条件的成立与激活步数的确认，或转移条件的连续转移动作的软元件或标签。

SFC用信息软元件对各块进行设置。

🔗 “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。 设置的位ON时可以使块启动，OFF时可以使块结束。 | 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 | 对确定使块停止的时机的软元件或标签进行设置。 设置的位ON时在各步的转移后使其停止，OFF时使所有步立即停止。 | | |
| 连续转移位 | 转移条件成立时，设置对连续转移的动作进行确定的软元件或标签。 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1步1步执行。 | | |
| 激活步数寄存器 | 设置对块的当前激活中的步数进行存储的软元件或标签。 | | |

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

要点

SFC用信息软元件的设置仅使用SFC用信息软元件的情况下需要。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。

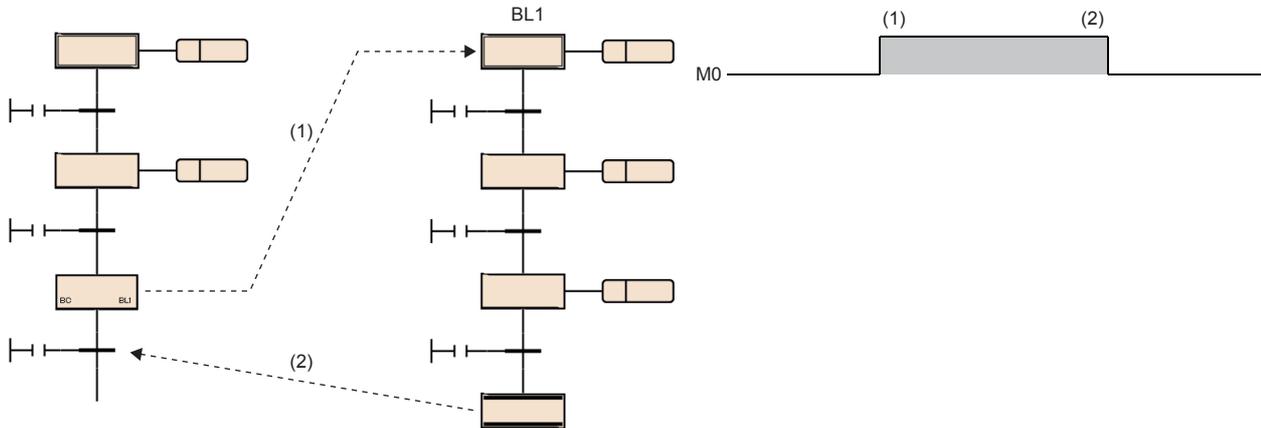
设置的软元件或标签ON时可以使块启动，OFF时可以使块结束。

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为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为OFF状态不变。

例

在块1 (BL1) 的块启动结束位中指定了M0的情况下



(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》)

n 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“对块0进行自动启动”的情况下 | 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“不对块0进行自动启动”的情况下 | 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用信息软件中设置的所有块启动结束位将OFF。但是，继续运行启动设置时仅允许继续运行启动的情况下SFC程序启动时，所有块启动结束位将恢复。

步转移位

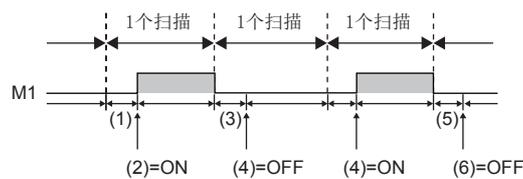
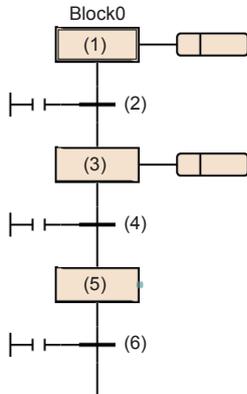
检查激活中的步的转移条件是否成立。

执行各步的动作输出后，至下一个步的转移条件成立的情况下，步转移位将自动ON。

变为ON的转移位在再次执行指定的块的处理时，将自动OFF。

例

在Block0的步转移位中指定了M1的情况下



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。

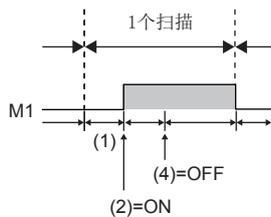
下一个扫描的Block0的处理时M1将OFF。

在步(3)的执行后，转移条件(4)不成立的情况下，M1将保持为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将保持为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，步转移位也将保持为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也将保持为ON状态不变。

下一个扫描的Block0的处理时M1将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ON。

n 注意事项

- 如果执行结束步，块的步转移位将ON。然后步转移位将保持为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OFF。

块停止重启位

块停止重启位是在指定块的激活中，使处理暂时停止的位。

| 块停止重启位的设置 | 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。 <ul style="list-style-type: none">在动作保持状态下变为停止的动作保持步(有转移检查/无转移检查)，在动作保持状态下重启执行。对于线圈保持步，线圈输出OFF的设置(SM325 = OFF)中进行了停止的情况下将变为非激活状态，因此不可以重启保持状态。线圈输出保持的设置(SM325 = ON)中进行了停止的情况下，将维持保持状态，因此重启后将直接为保持状态不变。 |

- 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为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，在块变为了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为不变。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/非保持的组合确定。(☞ 120页 块停止重启时的动作)

n 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OFF。

块停止模式位

块停止模式位是用于存在有通过块停止重启位的ON或PAUSE指令(块停止)进行的停止请求时，确定指定块的停止时机的位。

| 停止时模式位的设置 | 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。 无法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。 对于保持中的步，与块停止模式位的设置无关，停止请求后将立即停止。 |

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/非保持的组合确定。(☞ 120页 块停止重启时的动作)

n 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OFF。

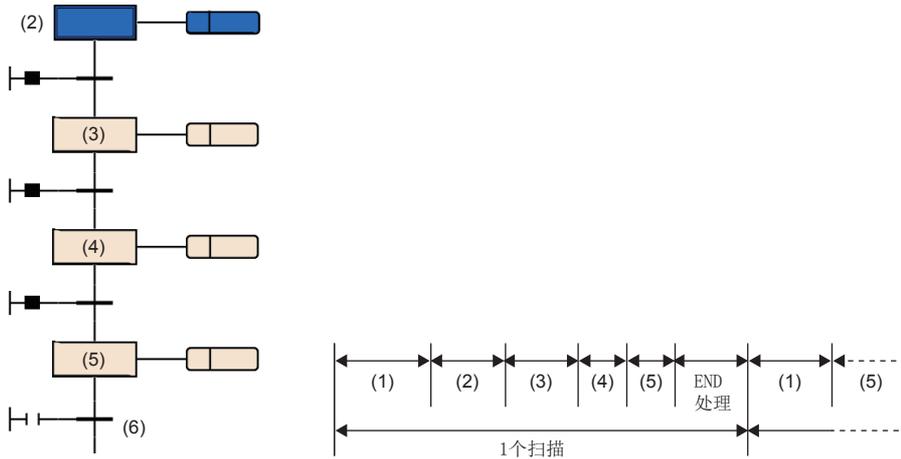
连续转移位

连续转移位是用于指定在转移条件成立时是否在同一扫描内执行下一个步的动作输出的位。

| 连续转移位的设置 | 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。 |

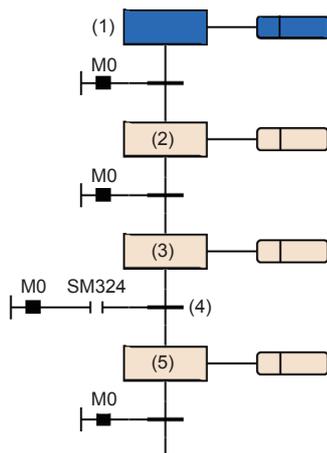
例

有SFC用信息软元件的连续转移位的指定的情况下



| 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。不对连续转移位进行设置的情况下, 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。(127页 连续转移有/无的动作)
- 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。

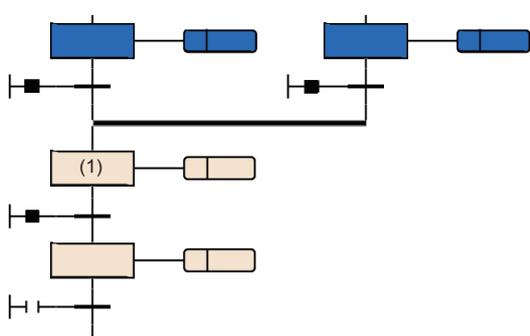


MO为ON时, 1个扫描中从步(1)到步(3)变为连续转移。

通过在转移条件(4)中作为AND条件对SM324进行附加, 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。

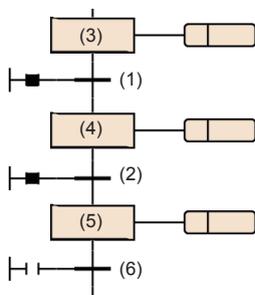
n 注意事项

- 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OFF。
- 通过跳转及选择合并从多个步转移至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，将步(1)在1个扫描中执行2次。

- 有连续转移的设置中步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将不被反映，在其它程序中无法检测线圈的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输出(Y)的ON。因此，也无法检测步继电器的ON。为了反映OUT指令的输入输出刷新，应创建将1个步在多个扫描中执行的程序。



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，因此将步(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，因此执行步(4)的动作输出。
- 由于转移条件(2)成立，因此将步(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，因此执行步(5)的动作输出。
- 由于转移条件(6)不成立，因此步(5)的动作输出不OFF。

- 在使用跳转进行循环的程序时，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。

激活步数寄存器

激活步数寄存器是对指定的块的当前激活中的步数进行存储的寄存器。

激活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

n 注意事项

- 在块结束时，激活步数寄存器将变为0。
- 在SFC程序结束时不变为0，在SFC程序启动时激活步数寄存器将变为0。

8.5 SFC设置

在CPU参数及SFC块设置内，对SFC程序的启动条件等进行设置。

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 | 对在SFC程序启动时，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(继续运行启动)进行设置。 |
| | 启动条件设置 | 对在SFC程序启动时，是使块0自动启动后再进行激活，还是保持非激活状态不变直至有启动请求为止进行设置。 |
| | 块停止时的输出模式设置 | 对在块停止时，是使线圈输出OFF，还是使线圈输出保持进行设置。 |

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/标签存储器区详细设置]⇒[软元件设置]中，在1024~16384点的范围(1024点单位)内设置步继电器(S)的点数。

SFC程序启动模式设置

对在SFC程序启动时，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(继续运行启动)进行设置。

☞ [CPU参数]⇒“SFC设置”⇒“SFC程序启动模式设置”

画面显示



显示内容

| 设置 | 内容 |
|--------------|---|
| 初始启动 (默认) | 对上次停止时的激活信息进行清除并启动。 启动后的动作按照SFC设置的启动条件设置进行。(☞ 119页 启动条件设置) |
| 继续运行启动 | 保持上次停止时的激活状态不变的情况下进行启动。 |

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，确定是进行初始启动还是进行继续运行启动。

| 动作 | SFC程序启动模式设置为初始启动 | | SFC程序启动模式设置为继续运行启动 | |
|---|------------------------|----------------------|-----------------------|-----------------------|
| | 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 | | 继续运行启动 | 初始启动 |
| (2) 将电源置为OFF→ON | | | 继续运行启动/初始启动*4 | |
| (3) 将SM321置为ON→OFF或RUN→STOP 后将电源置为OFF→ON | | | 继续运行启动 | |
| (4) 复位操作→RUN | | | 继续运行启动/初始启动*4 | |
| (5) 将SM321置为ON→OFF或RUN→STOP 后进行复位操作→RUN | | | 继续运行启动 | |
| (6) STOP→RUN | 继续运行启动*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时确定初始状态。

*2 将SFC程序启动模式设置设置为继续运行启动，且在程序的写入前后无更改的情况下将继续运行启动。

*3 动作输出的ON/OFF，按照参数设置的“STOP→RUN时的输出模式”的设置内容进行。

*4 根据时机将变为禁止继续运行启动状态，且有可能进行初始启动。

n 注意事项

- 继续运行启动时，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，在进行继续运行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。
- 线圈保持步变为OFF的条件(表中(1)、(3)、(5))以外的继续运行启动时，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时或复位操作时，智能功能模块将被初始化。继续运行启动的情况下，至智能功能模块的初始化程序，建议创建至常时变为激活状态的块或顺控程序上。
- 电源OFF时或复位操作时，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后或复位操作后的继续运行启动，根据时机有可能不可以继续运行启动。在继续运行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运行启动的事件将被存储。确实希望进行继续运行启动的情况下，应将SM321置为ON→OFF或RUN→STOP后再将电源置为OFF或进行复位操作。

启动条件设置

对在SFC程序启动时，是使块0自动启动后再进行激活，还是保持非激活状态不变直至有启动请求为止进行设置。

 [CPU参数]⇒“SFC设置”⇒“启动条件设置”

画面显示



显示内容

| 设置 | 内容 | |
|-------------------|--|------------------------------------|
| | SFC程序启动时 | 块0结束时 |
| 对块0进行自动启动 (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。 |
| 不对块0进行自动启动 | 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将无法自动进行再启动，保持非激活状态不变直至再次有启动请求为止。 |

对于启动条件设置，在希望根据产品类型等对SFC程序启动时的启动块进行指定时使用。

“对块0进行自动启动”在按以下方式使用块0的情况下有效。

- 作为管理块使用
- 作为前处理块使用
- 作为常时监视块使用

n 注意事项

- 设置为“不对块0进行自动启动”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“对块0进行自动启动”的情况下，必须对块0进行创建。

块停止时的输出模式设置

对在块停止时，是使线圈输出OFF，还是使线圈输出保持进行设置。

 [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。 |
| 保持为ON不变 | 将线圈输出在停止之前的状态下进行保持(线圈输出保持)。 |

- 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。

n 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/非保持的组合确定。

块停止/重启时的动作一览如下所示。

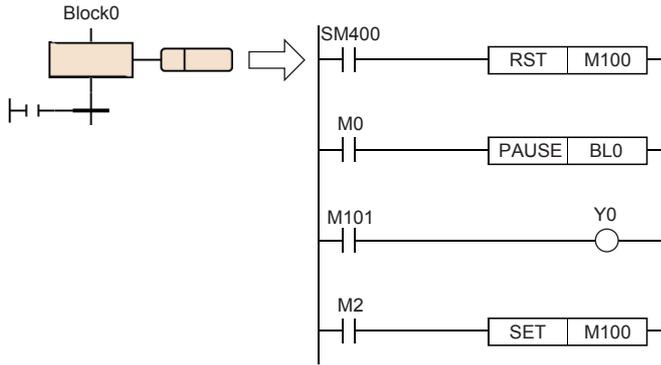
| 块停止时的输出模式的设置 | 块停止模式位的设置 | 动作 | | | |
|------------------------|-------------------|---|--|--|------------------------|
| | | 保持中以外的激活中步 (也包括转移条件不成立的SC、SE、ST) | 保持中步 | | |
| | | | 线圈保持步[SC] | 动作保持步(无转移检查)[SE] | 动作保持步(有转移检查)[ST] |
| SM325=OFF (线圈输出OFF) | OFF或无设置 (立即停止) | 有停止请求之后，将动作输出的线圈输出置为OFF后进行停止。状态保持为激活不变。 | 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。 | 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。状态保持为激活不变。 | |
| | ON (转移后停止) | 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。 | | | |
| SM325=ON (线圈输出保持) | OFF或无设置 (立即停止) | 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。状态保持为激活不变。 | 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。状态保持为激活不变。 | | |
| | ON (转移后停止) | 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。 | | | |
| 重启时 | | 返回普通的动作。 | 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。 | 在保持状态下重启动作输出的执行。 | 在保持状态下，重启动作输出后，检查转移条件。 |

n 注意事项

- 即使为停止中块，在LD指令(块激活检查)等中也作为ON被检测。此外，即使在步的停止中，在LD指令(步激活检查)等中也作为ON被检测。
- 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，对于非激活块执行了SET指令(步启动)的情况下，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，当前执行中的步也执行直至最后为止，之后执行停止请求。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，转移后停止模式中需要停止请求。

例

M100为停止时模式位，且M101为块停止重启位的情况下

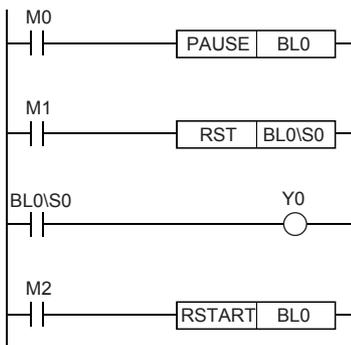


执行上述动作输出时，如果将M0置为ON则执行PAUSE指令，Block0的块停止重启位(M101)将变为ON，但是在动作输出的最后之前执行，因此Y0将变为ON状态。此外，M2为ON时，即使在PAUSE指令的执行后停止时模式位也将变为ON，在执行了全部动作输出后，转移后停止模式中需要停止请求。

- 在块停止中执行RST指令(步结束)时，指定的步继电器将OFF。但是，工程工具的监视画面将保持为激活状态不变，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。
- 对于SET指令(步启动)，即使在块停止中也立即被执行且指定的步继电器将变为ON，工程工具的监视画面中也变为激活状态。但是，动作输出不执行直至块被重启为止。

例

使用RST指令(步结束)时的块的停止重启的情况下



- (1) 如果将M0置为ON，块0将停止。
- (2) 如果将M1置为ON，步No. 0中需要结束请求，步继电器的BL0\S0将变为OFF，但是在工程工具的监视上步No. 0将保持激活中不变。
- (3) BL0\S0变为OFF，因此Y0也变为OFF。
- (4) 在M0、M1处于OFF的状态下将M2置为ON时，重启块0之后结束步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。从此状态对转移后停止等待状态进行解除后立即停止时，重启块后，在停止模式位为OFF的状态下需要再次开始停止请求。
- 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，将在工程工具的块一览表显示中进行监视，或对块停止重启位中设置的位进行监视。但是，不可以通过工程工具的监视确认步是否处于停止中或是否以停止等待进行动作中。
- 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，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。

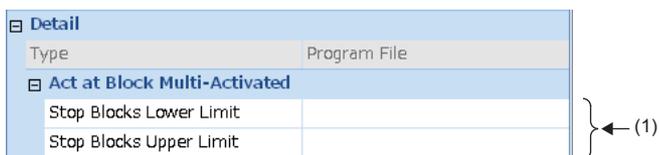
SFC块设置

块双重启动时的运行设置

对于已激活的块通过块启动步[BC或BS]发出了启动请求时，希望停止CPU模块的运算的情况下进行设置。设置范围对希望停止的块范围进行设置。

“导航窗口” ⇒ “程序” ⇒ 希望设置的SFC程序文件的属性

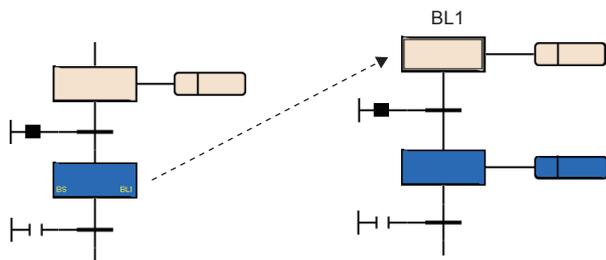
画面显示



(1) 对希望停止的块范围进行设置。

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进行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，将无法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。 |



n 注意事项

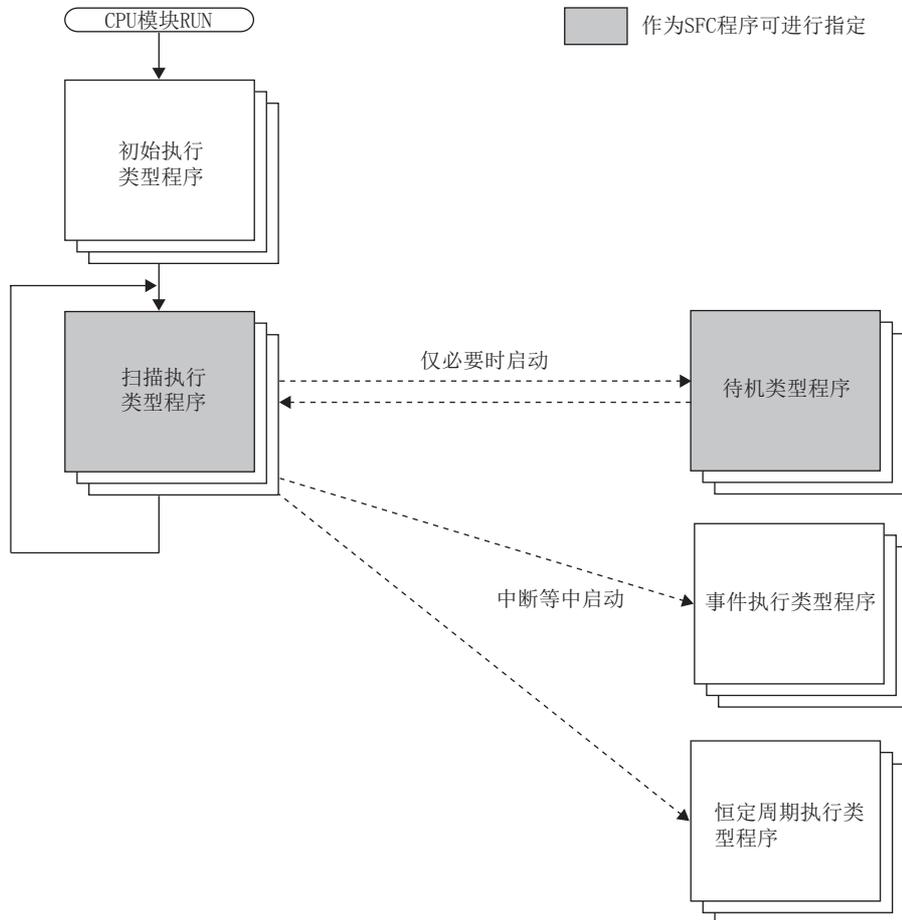
- 对于已处于激活中的块，执行了SFC控制指令的SET指令(块启动)时，将忽略启动请求直接继续运行SFC程序的处理。
- 试图转移至激活中的块启动步的情况下，块启动步的启动将被忽略。不可以再次从初始步开始执行。

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

对于SFC程序，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 | ○ | 通过以PSCAN指令对SFC程序进行指定，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 | × | — |

n 注意事项

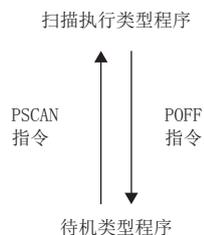
不存在扫描执行类型的SFC程序(仅待机类型程序)的情况下，请勿对SFC程序执行SFC控制指令及监视。

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 | ○ | 将指定的SFC程序的执行类型更改为扫描执行类型。 在已存在有变为扫描执行类型的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 | × | 对于SFC程序执行的情况下，将变为出错状态。 |
| POFF | ○ | 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。 |



n 注意事项

- 来自于CPU模块的文件读取/文件写入及，使用数据记录功能时请勿执行PSCAN指令。如果执行PSCAN指令，扫描时间有可能延长数个100ms。
- 在指定继续运行启动时，将与上次动作的SFC程序不同的SFC程序通过PSCAN指令进行了动作的情况下，指定的SFC程序将初始启动。在这种情况下，事件履历中“禁止SFC程序的继续运行启动”（事件代码：0430）将被保存。

SFC程序的执行顺序

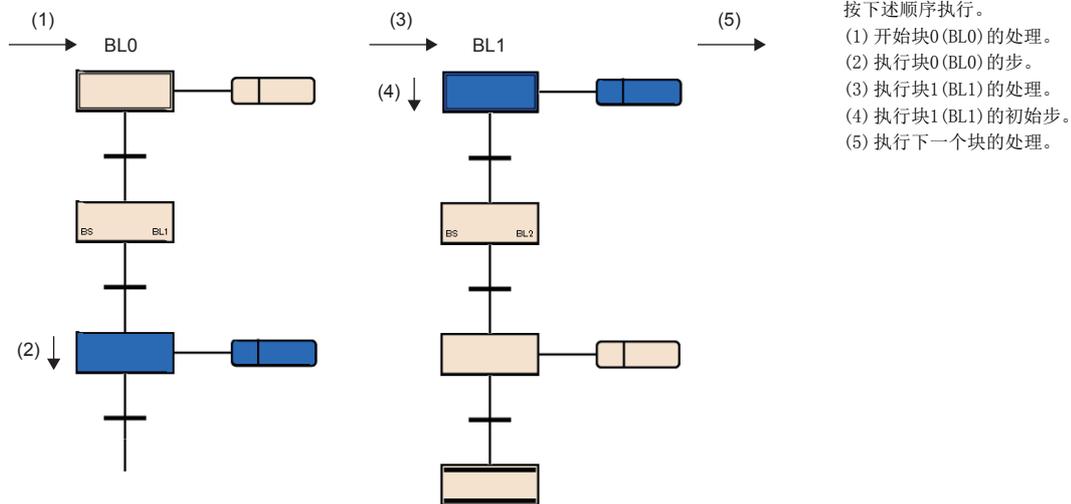
各块的执行顺序

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。如果有启动请求将块激活后，执行块内的激活步。

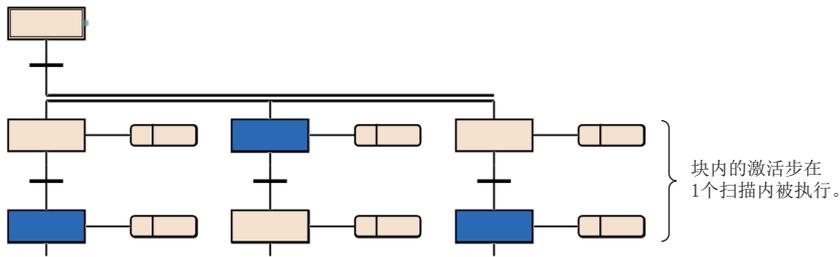


SFC设置的启动条件设置中指定了块0的自动启动的情况下，只可以使块0自动启动。在此设置的情况下，即使到达结束步变为非激活状态，块0也将在下一个扫描中再次被启动。(☞ 119页 CPU参数)

此外，对于块的结束、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

在SFC程序中，将所有激活步的动作输出在1个扫描内进行处理。

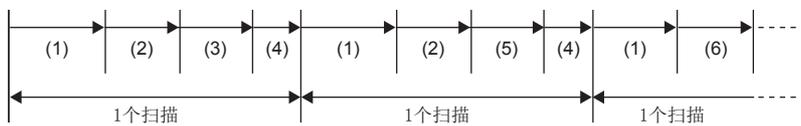


各步的动作输出结束时，检查至下一个步的转移条件的成立状态。

- 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，也将步的属性设置为线圈保持步[SC]时，不变为非激活状态而按照属性进行处理。（☞ 88页 线圈保持步[SC]）

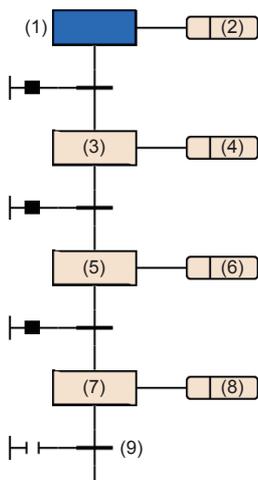
STOP→RUN
(SM321=ON)



- (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

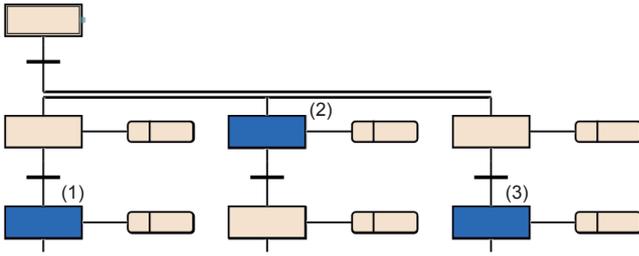
无SFC用信息软元件的连续转移的指定的情况下



| 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个扫描以后 | 在转移条件(9)成立之前激活步(7)，执行动作输出(8)。 |

n 注意事项

- 在首次执行时转移条件已成立的步的情况下，1个扫描中结束步，因此线圈输出等的输入输出刷新将无法被反映，在其它程序中不可以检测线圈输出的ON。为了反映输入输出刷新，应创建将1个步在多个扫描中执行的程序。
- 块内的激活步的动作输出同时(同一扫描内)被执行。因此，请勿创建取决于动作输出的执行顺序的SFC程序。
(1)、(2)、(3)的动作输出的执行顺序将变为不定。



连续转移有/无的动作

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。

连续转移无/有的设置根据SFC用信息软件元件的连续转移位的设置及SM323(所有块连续转移的有无)确定。

| 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“启动条件设置”设置为“进行自动启动”时，CPU模块的电源ON时、RESET解除时或STOP→RUN时启动SFC程序，并启动块0。(☞ 119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)通过系统自动被ON。

- 通过将SM321置为OFF，可以全部结束SFC程序的处理。
- 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程序启动模式设置”进行设置，可以继续启动SFC程序。(☞ 117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，可以对SFC程序进行启动或停止。(☞ 124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“启动条件设置”设置为“对块0进行自动启动”，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。 | ☞ 119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。 | ☞ 90页 块启动步(有结束检查)[BC] ☞ 91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)。 | 在异常检测时等对出错恢复处理块开始启动、执行中断处理时等有效。 | ☞ 106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“块启动结束位”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、试运行时有有效。 | ☞ 109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。 | ☞ 92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)使指定的块结束，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了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有效。 | ☞ 106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“块启动结束位”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、试运行时有有效。 | ☞ 109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程序执行中使指定的块停止的方法如下所示。

| 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)使指定的块的执行暂时停止。 | 在异常检测时等情况下使机械暂时停止通过手动运行对异常位置进行修复时有效。 | ☞ 106页 SFC控制指令 |
| 通过SFC用信息软件元件进行的停止 | 通过将各块中设置的“块停止重启位”作为SFC用信息软件元件置为ON，停止指定块。 | 也可通过外部设备进行停止，因此在调试、试运行同时进行确认的同时进行控制等情况下有效。 | ☞ 112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用信息软件元件的块停止模式位的设置、步的保持/非保持的组合确定。(☞ 120页 CPU参数)

块的重启方法

在SFC程序执行中使暂时停止的块的处理重启的方法如下所示。

| 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。 | ☞ 106页 SFC控制指令 |
| 通过SFC用信息软件元件进行的重启 | 通过将各块中设置的“块停止重启位”作为SFC用信息软件元件置为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、试运行时有有效。 | ☞ 112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用信息软件元件的块停止模式位的设置、步的保持/非保持的组合确定。(☞ 120页 CPU参数)

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。 | — | ☞ 106页 SFC控制指令 |
| 通过工程工具进行的启动 | <ul style="list-style-type: none">• 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, 结束保持步的情况下有效。 | ☞ 90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。 | ☞ 106页 SFC控制指令 |
| 通过工程工具进行的结束 | <ul style="list-style-type: none">• 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有效。 | ☞ GX Works3操作手册 |

步双重启动时的注意点

对步双重启动时的动作如下所示。

串行转移的情况下



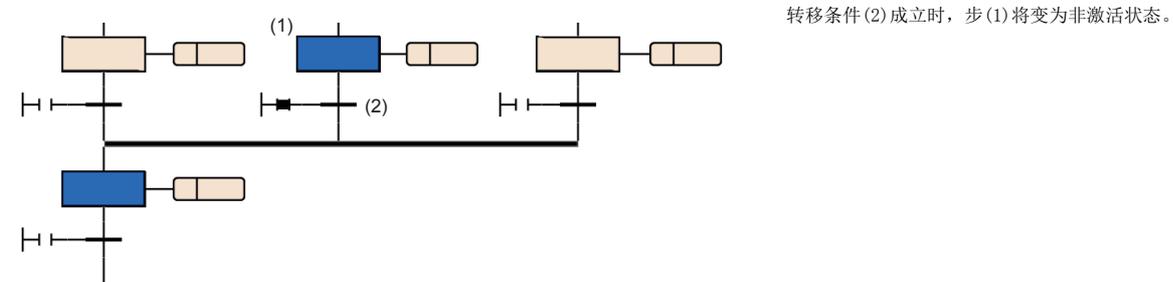
选择转移的情况下

n 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时, 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。

n 选择合并

选择合并时的双重启动的动作与串行转移的情况下相同。

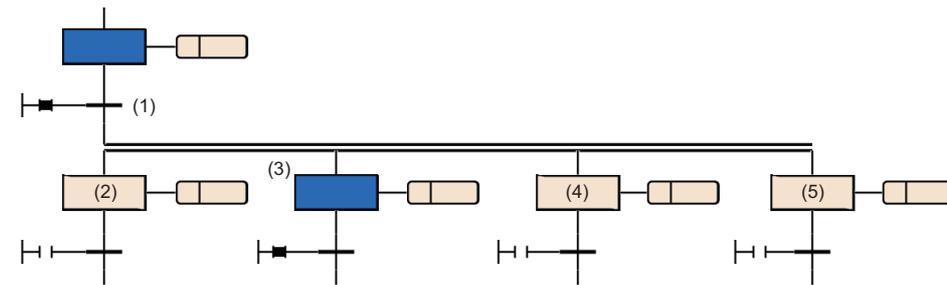


并行转移的情况下

n 并行分支

并行分支的情况下, 在多个转移目标中即使1个为激活状态, 在下一个扫描中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。



n 并行合并

转移源将变为非激活(保持步为保持状态)状态。

程序更改时的动作

对于SFC程序，至可编程控制器的写入及RUN中写入中进行了程序更改时的动作如下所示。

通过至可编程控制器的写入进行的程序更改

通过至可编程控制器的写入进行的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 (继续运行启动) | 初始启动 | 继续运行启动 |

关于SFC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。 <ul style="list-style-type: none">• 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的设置仅在至可编程控制器的写入后SFC程序存在时有效。此外，不仅SFC程序的写入，存在程序文件及参数文件的写入的情况下也有效。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 文件寄存器 (R/ZR) 即使未进行锁存设置的情况下，也不会变为SM326的清除对象。

n 注意事项

- 通过至可编程控制器的写入进行的程序更改后，应在进行了一次复位操作后，执行SFC程序。
- CPU参数的SFC程序启动模式设置为继续运行启动的情况下，应将SM322 (SFC程序的启动状态) 置为OFF (初始启动) 后，进行通过至可编程控制器的写入进行的程序更改。之后，应使SFC程序初始启动之后，再次将SM322置为ON (继续运行启动)。

通过RUN中写入进行的程序更改

通过RUN中写入进行的程序更改后，与CPU参数的SFC程序启动模式设置无关，必须进行继续运行启动。

STOP→RUN操作

在SFC程序的行动中 (RUN中) 使CPU模块STOP的情况下，在STOP→RUN时软元件的状态与SFC程序的激活信息均恢复为STOP前的状态。与CPU参数的SFC程序启动模式设置无关，将变为继续运行启动。

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) 的某个写入到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运行启动。(P.117页 SFC程序启动模式设置)

SFC程序的动作确认

SFC程序的动作确认中可使用的工程工具的功能如下所示。

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

要点

关于通过工程工具进行的SFC程序的动作确认有关内容，请参阅下述手册。

 GX Works3操作手册

索引

[符号]

| | |
|----|----|
| - | 56 |
| * | 56 |
| ** | 56 |
| / | 56 |
| & | 56 |
| + | 56 |
| < | 56 |
| <= | 56 |
| <> | 56 |
| = | 56 |
| > | 56 |
| >= | 56 |
| \$ | 43 |

[A]

| | |
|-------------|------------|
| AND | 56 |
| ASCII | 35, 65, 76 |
| 安全程序 | 7 |
| 安全功能块(安全FB) | 32 |
| 安全功能(安全FUN) | 31 |
| 安全局部标签 | 46 |
| 安全控制 | 7 |
| 安全全局标签 | 46 |
| 安全软元件 | 7 |
| 安全通信 | 7 |

[B]

| | |
|-------------|-----|
| BC | 85 |
| BOOL | 35 |
| BS | 85 |
| 保留字 | 55 |
| 标签 | 7 |
| 标签/软元件 | 98 |
| 并行转移(分支/合并) | 99 |
| 步数 | 16 |
| 步转移位 | 108 |

[C]

| | |
|---------|------------|
| CASE | 60 |
| COUNTER | 35, 36 |
| 长定时器 | 35, 36 |
| 长计数器 | 35, 36 |
| 长累计定时器 | 35, 36 |
| 常数 | 43 |
| 程序 | 10, 14, 24 |
| 程序块 | 11, 34 |

| | |
|------|----|
| 程序文件 | 10 |
| 程序语言 | 8 |
| 初始步 | 85 |
| 串行转移 | 99 |

[D]

| | |
|--------------|--------|
| DINT | 35 |
| DWORD | 35 |
| 代入语句 | 57 |
| 单精度实数 | 35 |
| 定时器 | 35, 36 |
| 动作保持步(无转移检查) | 85 |
| 动作保持步(有转移检查) | 85 |

[E]

| | |
|------|--------|
| EN | 14, 24 |
| ENO | 14, 24 |
| EXIT | 60 |

[F]

| | |
|----------|----------------|
| FBD/LD语言 | 8 |
| FB/FUN文件 | 14, 15, 24, 25 |
| FOR……DO | 60, 64 |
| 分类 | 37 |
| 复位步 | 85 |

[G]

| | |
|---------|------------|
| 工程 | 10 |
| 功能调用语句 | 59 |
| 功能块调用语句 | 59 |
| 功能块(FB) | 12, 18, 34 |
| 功能(FUN) | 12, 13, 34 |

[H]

| | |
|-------|--------|
| 宏型功能块 | 25, 26 |
| 缓冲存储器 | 7 |

[I]

| | |
|------------|----|
| IF THEN | 60 |
| IF……ELSE | 60 |
| IF……ELSEIF | 60 |
| INT | 35 |

[J]

| | |
|---------|--------|
| 激活步数寄存器 | 108 |
| 计数器 | 35, 36 |

| | |
|------------------|------------|
| 结构体 | 36, 41 |
| 结构体的排列 | 42 |
| 结束步 | 85 |
| 局部标签. | 33, 34, 37 |

[K]

| | |
|-----------------------|-----|
| 块启动步(无结束检查) | 85 |
| 块启动步(有结束检查) | 85 |
| 块启动结束位 | 108 |
| 块停止模式位 | 108 |
| 块停止时的输出模式设置 | 116 |
| 块停止重启位 | 108 |

[L]

| | |
|---------------------------|--------|
| LCOUNTER. | 35, 36 |
| LREAL. | 35 |
| LRETENTIVETIMER | 35, 36 |
| LTIMER | 35, 36 |
| 累计定时器 | 35, 36 |
| 类型指定. | 66, 76 |
| 类型转换. | 58, 70 |
| 连续转移位 | 108 |

[M]

| | |
|---------------|----|
| MOD | 56 |
| 模块标签. | 33 |

[N]

| | |
|---------------|----|
| NOT | 56 |
| 内部变量. | 19 |

[O]

| | |
|--------------|----|
| OR | 56 |
|--------------|----|

[P]

| | |
|-------------------|----|
| POINTER | 35 |
| 排列要素数 | 39 |
| 普通步 | 85 |

[Q]

| | |
|------------------|------------|
| 启动条件设置 | 116 |
| 全局标签. | 33, 34, 37 |

[R]

| | |
|--------------------------|--------|
| R | 85 |
| REAL | 35 |
| REPEAT……UNTIL | 60 |
| RETENTIVETIMER | 35, 36 |
| RETURN | 59 |

| | |
|------------------|----|
| 软元件 | 7 |
| 软元件的分配 | 33 |

[S]

| | |
|---------------------------|------------|
| SC | 85 |
| SE | 85 |
| SFC程序 | 8 |
| SFC程序启动模式设置. | 116 |
| ST | 85 |
| STRING | 35, 65, 76 |
| ST语言 | 8 |
| 声明 | 53 |
| 时间 | 35 |
| 实例 | 20 |
| 输出变量. | 13, 19 |
| 数据类型. | 35, 36, 37 |
| 输入变量. | 13, 19 |
| 输入输出变量 | 19 |
| 双精度实数 | 35 |
| 双字[带符号] | 35 |
| 双字[无符号]/位串[32位] | 35 |

[T]

| | |
|-----------------|--------|
| TIME | 35 |
| TIMER | 35, 36 |
| 梯形图语言 | 8, 49 |
| 跳转 | 99 |

[U]

| | |
|-------------------|--------|
| Unicode | 35, 65 |
|-------------------|--------|

[W]

| | |
|--------------------|------------|
| WHILE……DO. | 60 |
| WORD | 35 |
| WSTRING | 35, 65, 76 |
| 外部变量. | 19 |
| 位 | 35 |

[X]

| | |
|----------------------|----|
| XOR. | 56 |
| 系统标签. | 33 |
| 线圈保持步 | 85 |
| 详细表示. | 98 |
| 选择转移(分支/合并). | 99 |

[Y]

| | |
|-----------------|---|
| 一般CPU | 7 |
| 一般程序. | 7 |
| 一般控制. | 7 |
| 一般软元件 | 7 |

| | |
|---------------------|------------|
| 一般通信. | 7 |
| 一般/安全共享标签 | 46 |
| 移位JIS | 35, 65, 76 |

[Z]

| | |
|-------------------------|------------|
| 直接表示. | 98 |
| 指针 | 35 |
| 中断程序. | 11 |
| 主程序 | 11 |
| 注释 | 53 |
| 转移条件名. | 98 |
| 转移条件No. | 98 |
| 子程序 | 11 |
| 子程序型功能块 | 25, 26 |
| 字符串 | 35, 65, 76 |
| 字符串[Unicode] | 35, 65, 76 |
| 字[带符号] | 35 |
| 字[无符号]/位串[16位]. | 35 |
| 总称数据类型(ANY型) | 37 |

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08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02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06月 | SH (NA) -081319CHN-E | 第五版 部分修改 |

日文原稿手册：SH-081225-E

本手册不授予工业产权或任何其它类型的权利，也不授予任何专利许可。三菱电机对于使用了本手册中的内容而引起的涉及工业产权的任何问题不承担责任。

© 2014 MITSUBISHI ELECTRIC CORPORATION

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前提下。

(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、大风和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，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. 意外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，对于任何非三菱电机责任的原因而导致的损失、机会损失、因三菱电机产品故障而引起的用户利润损失、无论能否预测的特殊损失和间接损失、事故赔偿、除三菱电机以外产品的损失赔偿、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，三菱电机将不承担责任。

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

Ethernet是Fuji Xerox Corporation在日本的注册商标。

本手册中的公司名、系统名和产品名等是相应公司的注册商标或商标。

本手册中，有时未标明商标符号(™、®)。

SH (NA) -081319CHN-E (1606) 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

内容如有更改 恕不另行通知