MITSUBISHI ELECTRIC

三菱微型可编程控制器

MELSEC iQ-F series

MELSEC iQ-F
FX5编程手册(程序设计篇)

安全方面注意事项

（使用之前请务必阅读）

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

前言

此次承蒙购入MELSEC iQ-F系列可编程控制器产品，诚表谢意。

本手册是帮助用户理解进行FX5编程时所需的指令与函数的手册。

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

此外，希望本手册能够送达至最终用户处。

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

使用时的请求

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

预先通知

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

关联手册

对象模块的用户手册

手册名称<手册编号>	内容
MELSEC iQ-F FX5用户手册(入门篇) <JY997D59501>	记载FX5 CPU模块的性能规格、运行前的步骤、故障排除相关的内容。
MELSEC iQ-F FX5U用户手册(硬件篇) <JY997D58601>	记载FX5U CPU模块的输入输出规格、配线、安装及维护等的硬件相关的详细事项。
MELSEC iQ-F FX5UC用户手册(硬件篇) <JY997D61501>	记载FX5UC CPU模块的输入输出规格、配线、安装及维护等的硬件相关的详细事项。
MELSEC iQ-F FX5用户手册(应用篇) <JY997D58701>	记载程序设计中必要的基础知识、CPU模块的功能、软元件/标签、参数的说明等内容。
MELSEC iQ-F FX5编程手册(程序设计篇) <JY997D58801>(本手册)	记载梯形图、ST、FBD/LD等程序的规格以及标签的内容。
MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇) <JY997D58901>	记载在程序中可使用的命令及函数的规格的内容。
MELSEC iQ-F FX5用户手册(串行通信篇) <JY997D59001>	记载简易PLC间链接、MC协议、变频器通信、无顺序通信、通信协议支持相关的内容。
MELSEC iQ-F FX5用户手册(MODBUS通信篇) <JY997D59201>	记载MODBUS串行通信相关的内容。
MELSEC iQ-F FX5用户手册(以太网通信篇) <JY997D59301>	记载内置以太网端口通信功能相关的内容。
MELSEC iQ-F FX5用户手册(SLMP篇) <JY997D59101>	对对方设备采用基于SLMP的通信对CPU模块的数据进行读取、写入等的方法进行说明。
MELSEC iQ-F FX5用户手册(定位篇) <JY997D59401>	记载内置定位功能相关的内容。
MELSEC iQ-F FX5用户手册(模拟量篇) <JY997D60601>	记载模拟量功能相关的内容。
GX Works3操作手册 <SH-081271CHN>	记载GX Works3的系统配置、参数设置、在线功能的操作方法等简单工程及结构化工程通用的功能相关的内容。

术语

除特别注明的情况外，本手册中使用下列术语进行说明。

- □表示多个型号及版本等的总称时的可变部分。

  (例)FX5U-32MR/ES、FX5U-32MT/ES ⇨ FX5U-32M□/ES

- 关于能够与FX5连接的FX3的设备，请参照FX5用户手册(硬件篇)。

术语	内容
■设备	
FX5	FX5U、FX5UC可编程控制器的总称
FX3	FX3S、FX3G、FX3GC、FX3U、FX3UC可编程控制器的总称
FX5 CPU模块	FX5U CPU模块、FX5UC CPU模块的总称
FX5U CPU模块	FX5U-32MR/ES、FX5U-32MT/ES、FX5U-32MT/ESS、FX5U-64MR/ES、FX5U-64MT/ES、FX5U-64MT/ESS、FX5U-80MR/ES、FX5U-80MT/ES、FX5U-80MT/ESS的总称
FX5UC CPU模块	FX5UC-32MT/D、FX5UC-32MT/DSS的总称
扩展模块	FX5扩展模块、FX3扩展模块的总称
•FX5扩展模块	I/O模块、FX5扩展电源模块、FX5智能功能模块的总称
•FX3扩展模块	FX3扩展电源模块、FX3智能功能模块的总称
•扩展模块(扩展电缆型)	输入模块(扩展电缆型)、输出模块(扩展电缆型)、总线转换模块(扩展电缆型)、智能功能模块的总称
•扩展模块(扩展连接器型)	输入模块(扩展连接器型)、输出模块(扩展连接器型)、输入输出模块、总线转换模块(扩展连接器型)、连接器转换模块(扩展连接器型)的总称
I/O模块	输入模块、输出模块、输入输出模块、电源内置输入输出模块的总称
输入模块	输入模块(扩展电缆型)、输入模块(扩展连接器型)的总称
•输入模块(扩展电缆型)	FX5-8EX/ES、FX5-16EX/ES的总称
•输入模块(扩展连接器型)	FX5-C32EX/D、FX5-C32EX/DS的总称
输出模块	输出模块(扩展电缆型)、输出模块(扩展连接器型)的总称

输出模块	输出模块(扩展电缆型)、输出模块(扩展连接器型)的总称
•输出模块(扩展电缆型)	FX5-8EYR/ES、FX5-8EYT/ES、FX5-8EYT/ESS、FX5-16EYR/ES、FX5-16EYT/ES、FX5-16EYT/ESS的总称
•输出模块(扩展连接器型)	FX5-C32EYT/D、FX5-C32EYT/DSS的总称
输入输出模块	FX5-C32ET/D、FX5-C32ET/DSS的总称
电源内置输入输出模块	FX5-32ER/ES、FX5-32ET/ES、FX5-32ET/ESS的总称
扩展电源模块	FX5扩展电源模块、FX3扩展电源模块的总称
•FX5扩展电源模块	FX5-1PSU-5V的别称
•FX3扩展电源模块	FX3U-1PSU-5V的别称
智能模块	智能功能模块的简称
智能功能模块	FX5智能功能模块、FX3智能功能模块的总称
•FX5智能功能模块	FX5智能功能模块的总称
•FX3智能功能模块	FX3智能功能模块的别称
简单运动模块	FX5-40SSC-S的别称
扩展板	FX5U CPU模块用板的总称
•通信板	FX5-232-BD、FX5-485-BD、FX5-422-BD-GOT的总称
扩展适配器	FX5 CPU模块用适配器的总称
•通信适配器	FX5-232ADP、FX5-485ADP的总称
•模拟量适配器	FX5-4AD-ADP、FX5-4DA-ADP的总称
总线转换模块	总线转换模块(扩展电缆型)、总线转换模块(扩展连接器型)的总称
•总线转换模块(扩展电缆型)	FX5-CNV-BUS的别称
•总线转换模块(扩展连接器型)	FX5-CNV-BUSC的别称
电池	FX3U-32BL的别称
外围设备	工程工具、GOT的总称
GOT	三菱图形操作终端 GOT1000、GOT2000系列的总称
■软件包	
工程工具	MELSEC可编程控制器软件包的产品名
GX Works3	SWnDND-GXW3的总称产品名(n表示版本)
■手册	
用户手册	另附手册的总称
•用户手册(入门篇)	MELSEC iQ-F FX5用户手册(入门篇)的简称
•FX5用户手册(硬件篇)	MELSEC iQ-F FX5U用户手册(硬件篇)、MELSEC iQ-F FX5UC用户手册(硬件篇)的总称
•FX5U用户手册(硬件篇)	MELSEC iQ-F FX5U用户手册(硬件篇)的简称
•FX5UC用户手册(硬件篇)	MELSEC iQ-F FX5UC用户手册(硬件篇)的简称
•用户手册(应用篇)	MELSEC iQ-F FX5用户手册(应用篇)的简称
编程手册(程序设计篇)	MELSEC iQ-F FX5编程手册(程序设计篇)的简称
编程手册(指令/通用FUN/FB篇)	MELSEC iQ-F FX5编程手册(指令/通用FUN/FB篇)的简称
通信手册	MELSEC iQ-F FX5用户手册(串行通信篇)、MELSEC iQ-F FX5用户手册(MODBUS通信篇)、MELSEC iQ-F FX5用户手册(以太网通信篇)、MELSEC iQ-F FX5用户手册(SLMP篇)的总称
•串行通信手册	MELSEC iQ-F FX5用户手册(串行通信篇)的简称
•MODBUS通信手册	MELSEC iQ-F FX5用户手册(MODBUS通信篇)的简称
•以太网通信手册	MELSEC iQ-F FX5用户手册(以太网通信篇)的简称
•SLMP手册	MELSEC iQ-F FX5用户手册(SLMP篇)的简称
定位手册	MELSEC iQ-F FX5用户手册(定位篇)的简称
模拟量手册	MELSEC iQ-F FX5用户手册(模拟量篇)的简称
■程序相关	
操作数	是在各个指令及函数的内部配置中使用的源数据(s)、接收数据(d)、软元件数(n)等的软元件部分的总称。
软元件	CPU模块内部含有的软元件(X、Y、M、D等)。
缓冲存储器	是用于存储设置值、监视值等数据的智能功能模块的存储器。
程序部件	按功能分别定义的程序单位。通过程序的部件化，可以将程序分级时的低位处理按处理内容及功能分为若干个单位，创建各单位的程序。

1 概要

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

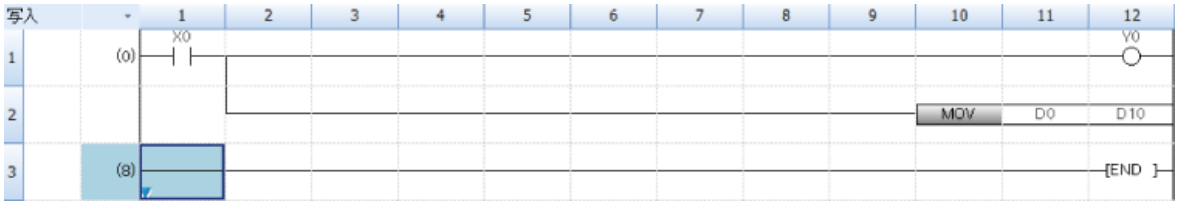
GX Works3操作手册

程序语言的类型

FX5系列中，可以根据用途选择最合适的程序语言使用。

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回路的图表语言。 梯形图语言是为了执行简单易懂的顺控程序控制而使用符号化的触点及线圈等记述逻辑回路的语言。
结构化文本语言 (ST语言)	是使用IF语句或运算符等记述程序的文本语言。 ST语言与梯形图语言相比，因为可以对难以记述的运算处理简洁而直观地进行记述，因此适用于进行复杂的算术运算及比较运算等的领域。此外，可以与C语言等一样，通过条件语句进行选择分支，通过循环语句进行重复等的语句对控制进行记述，因此可以简洁地编写程序。
功能块图表/梯形图语言 (FBD/LD语言)	是通过将实施特定处理的块(功能部件、FB部件)、变量部件、常数部件等沿着数据和信号的流动进行连接，记述程序的图表语言。 能够轻松地创建利用梯形图程序创建时较为复杂的程序，改善程序生产效率。

■ 梯形图语言



使用梯形图语言的情况下，请参阅下述内容。

5 梯形图语言

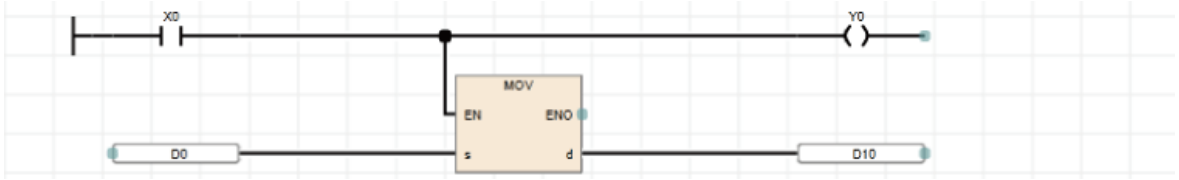
■ ST语言

```
IF X0 THEN
    Y0 := TRUE ;
    D0 := D10 ;
END_IF ;
```

使用ST语言的情况下，请参阅下述内容。

6 ST语言

■ FBD/LD语言



使用FBD/LD语言的情况下，请参阅下述内容。

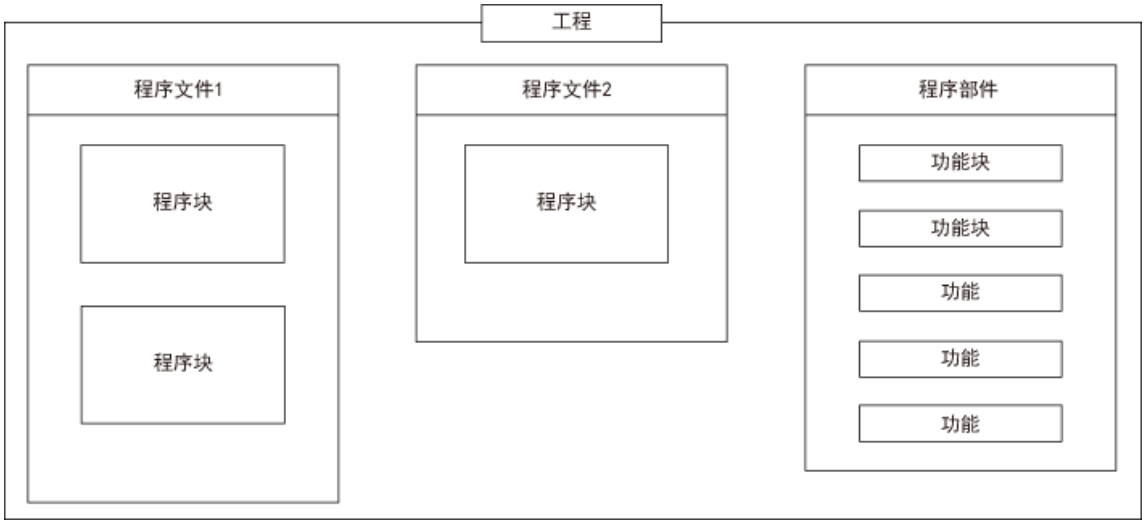
7 FBD/LD语言

Point

- 梯形图语言和FBD/LD语言适合具有顺控程序控制和逻辑回路的相关知识和经验的客户。
- ST语言适合具有C语言等编程知识及经验的用户。
- 通过在程序中使用标签，可以提高程序的可读性，将程序简单地转变至模块配置不同的系统中。

2 程序配置

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



关于程序部件，请参阅下述内容。

☞3 程序部件

工程

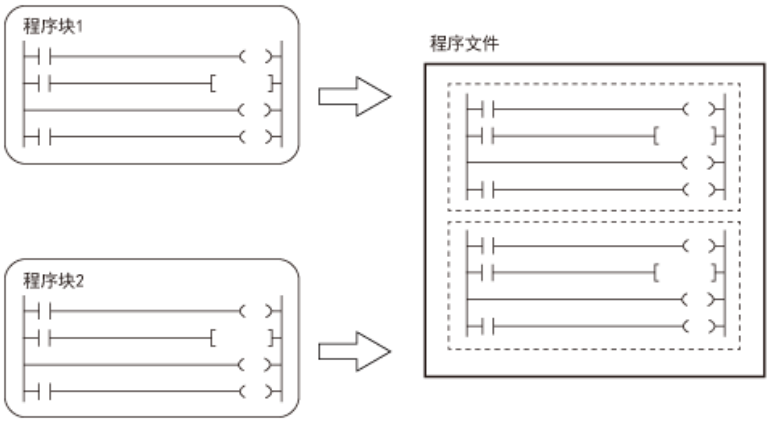
工程是在CPU模块中执行的数据(程序、参数等)的集合。
每一个CPU模块中只可写入一个工程。
工程中需要创建一个以上的程序文件。

程序文件

程序文件是程序与程序部件的集合。
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扫描执行类型替换为待机类型，以及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。
可以在程序文件中创建多个程序块并按照登录顺序执行。
如果按各功能及处理来划分程序块，可以设计可容易进行程序的顺序更改及交换的程序。
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。



通过对各程序块分别创建主程序、子程序、中断程序，可以创建易懂的程序。

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。

关于主程序、子程序、中断程序的详细内容，请参阅下述手册。

用户手册(应用篇)

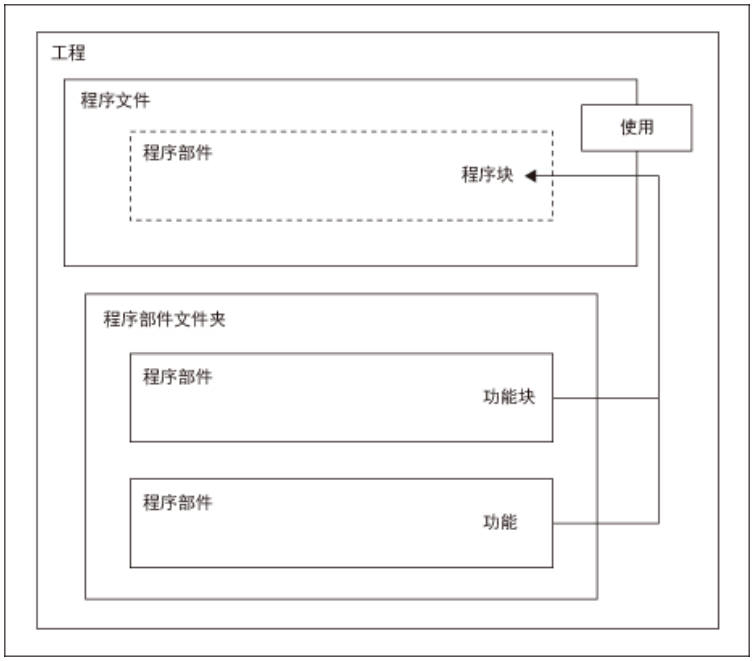
- 子程序以及中断程序是在FEND指令以后进行创建。FEND指令以后的程序不作为主程序进行执行。例如，在第二个程序块的最后使用了FEND指令的情况下，第三个程序块之后将变为子程序或中断程序。
- 为了创建易懂的程序，应将成对的FOR指令与NEXT指令、MC指令与MCR指令在一个程序块中使用。
- 简单程序的情况下，在一个程序块内仅记述主程序，即可以使其在CPU模块中执行。

3 程序部件

程序部件有下述几种类型。

- 功能
- 功能块

各程序部件中，可以使用与控制相适合的程序语言记述处理。程序部件是从程序块中调用后执行。



Point

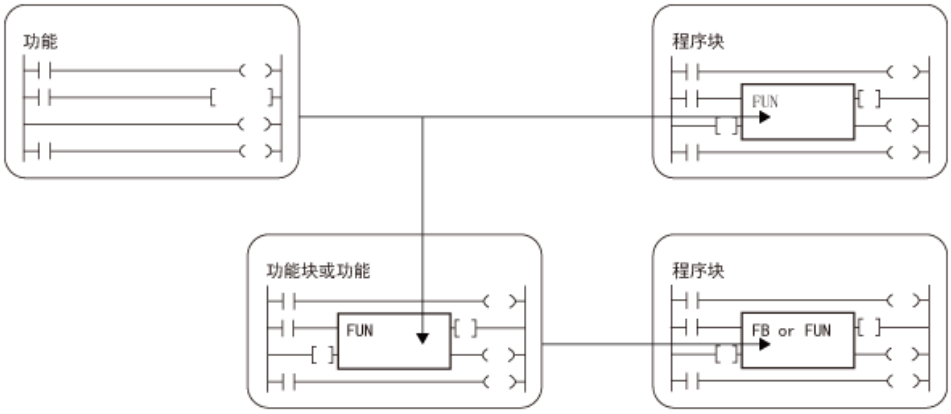
程序的部件化是指将程序阶层化时的低位处理按照各处理内容及功能分为若干单位，创建各单位的程序。
通过程序的部件化，可提高独立性，设计更容易进行添加及交换的程序。
如果进行部件化会更好的处理方法，有以下几种。

- 在程序中被循环记述的处理
- 被划分为一个功能的处理

本项使用标签进行各程序部件的说明。
功能及功能块中的程序本体还可以使用软元件。关于软元件的详细内容，请参阅下述手册。
用户手册(应用篇)

3.1 功能(FUN)

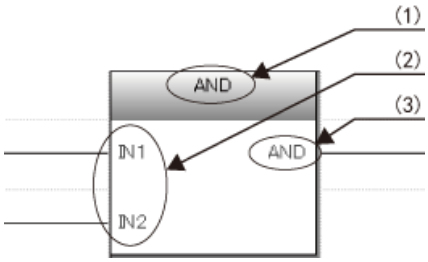
功能是在程序块、功能块以及其它的功能中使用的程序部件。
功能执行完成后将值交接至调用源。该值称为返回值。
功能对于同样的输入，将作为处理结果始终输出相同的返回值。
如果预先定义经常使用的单纯独立的程序算法，可以有效地再利用。



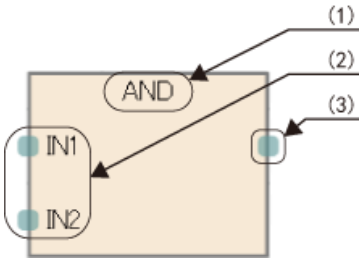
关于输入变量与输出变量

功能可以定义输入变量与输出变量。输出变量可制作与返回值不同的输出数据。

■梯形图语言的情况下



■FBD/LD语言的情况下



(1)功能名
(2)输入变量
(3)输出变量

关于可以设置输入变量、输出变量的分类，请参阅下述内容。

☞4.2 分类

Point

在功能中定义的变量在每次功能调用时被覆盖。

希望每次调用时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至不同的变量等以进行编程。

关于EN/ENO

通过在功能中附加EN(启动输入)、ENO(启动输出)，可以控制执行处理。

- EN设置作为功能的执行条件的布尔型变量。
- 附带EN的功能只在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，请参阅下述内容。

☞4.3 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

Point

- 无需设置至ENO的输出标签。
- 在通用功能中使用EN/ENO的情况下，附带EN的功能就变为“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，请参阅下述内容。

项目	参照目标
功能的创建方法	GX Works3操作手册
能写入CPU模块的FB/FUN文件数	用户手册(入门篇)

■ 可使用的软元件/标签

功能程序中可使用的软元件及标签一览如下所示。

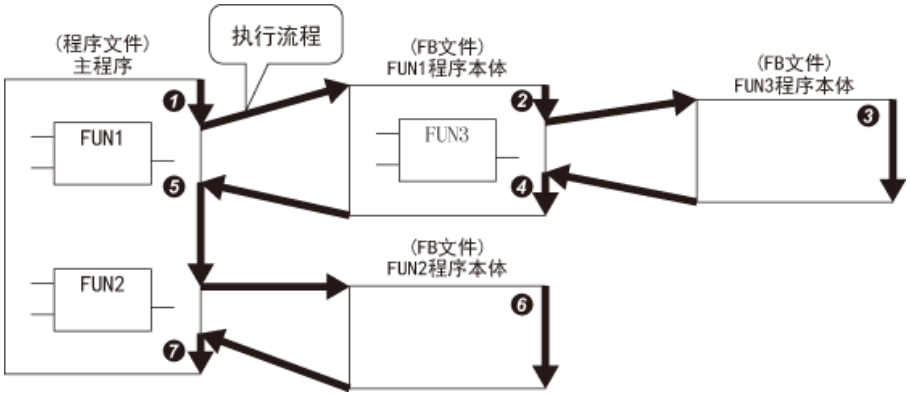
○:可以使用、△:只可在指令中使用(禁止作为表示程序的步的标签使用)、×:禁止使用

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

*1 请勿使用定时器型、累计定时器型、计数器型、长计数器型。

动作概要

功能把程序本体存储在FB/FUN文件内，在执行时从调用源程序中调用出FB/FUN文件内的程序本体以执行。

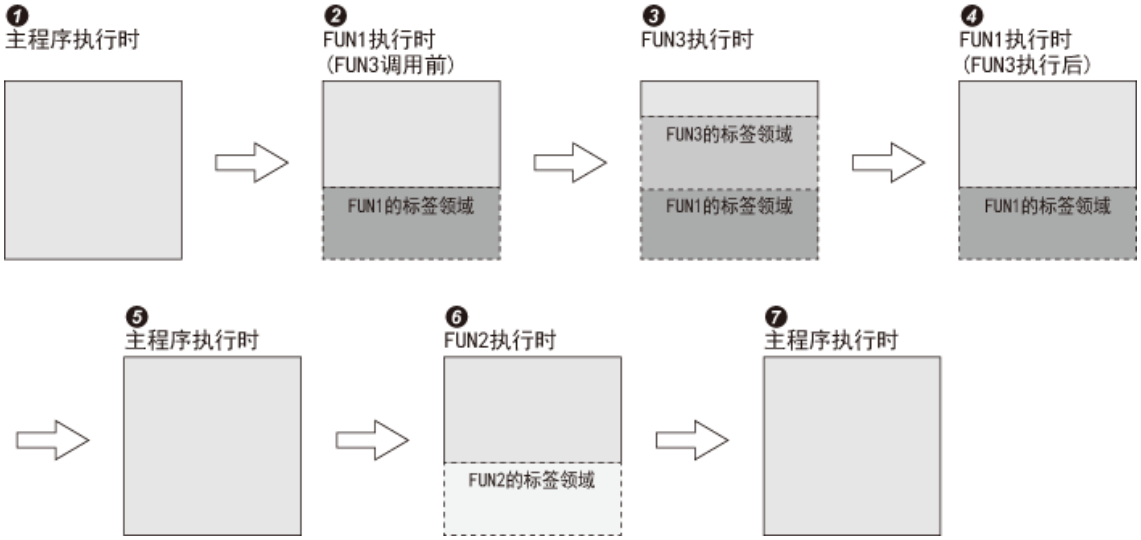


加上所有的功能块、功能，最多可嵌套32次。

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占用存储器内的临时领域，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。



关于在功能中可以定义的标签的类型，请参阅下述内容。

☞4.2 分类

Point

由于在功能中定义的标签为不定值，需要在最初访问时在程序中进行初始化。

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、调用程序本体及由系统使用追加步的步数。

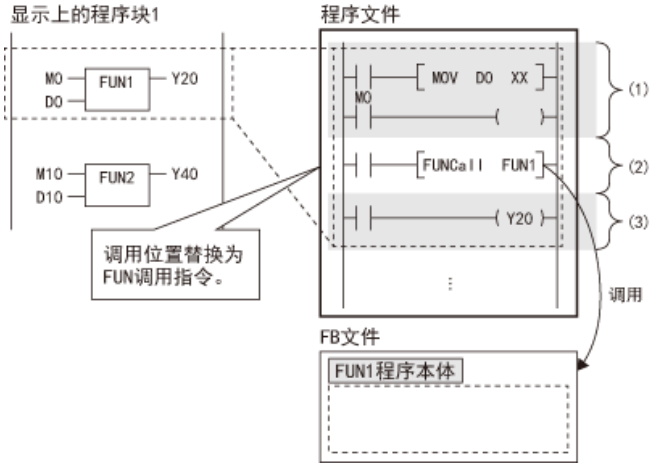
■ 程序本体

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

编程手册(指令/通用FUN/FB篇)

■ 调用侧

调动功能的情况下，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。



(1) 自变量交接

(2) FUN1程序本体调用

(3) 返回值交接

自变量交接

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

自变量的分类	数据类型	使用指令	步数
VAR_INPUT	位	LD+OUT LD+MOVB (根据使用的程序语言、功能的种类、输入自变量的种类的组合，使用其中的某一种。)	关于各指令的步数，请参阅下述手册。 编程手册(指令/通用FUN/FB篇)
	字[无符号]/位列[16位] 双字[无符号]/位列[32位] 字[带符号] 双字[带符号]	LD+MOV LD+DMOV	
	单精度实数	LD+EMOV	
	时间	LD+DMOV	
	字符串(32)	LD+$MOV	
	数组、结构体	LD+BMOV	

程序本体调用

功能的程序本体的调用至少需要16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	6
ENO	4

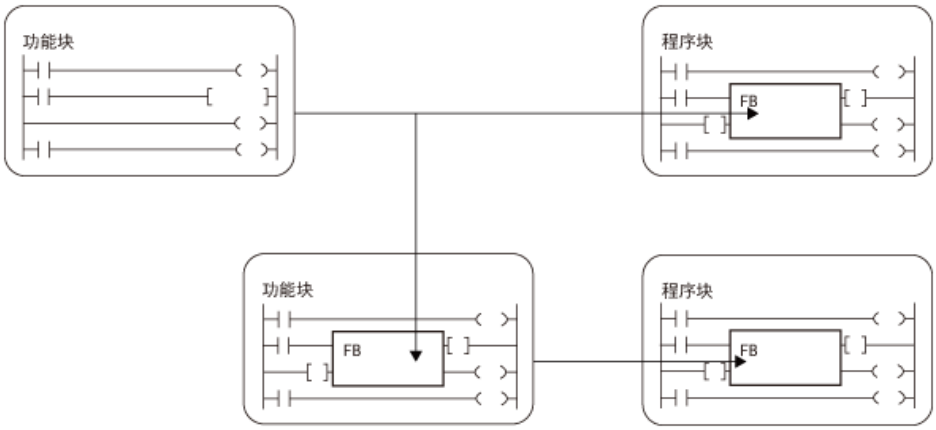
注意事项

■ 全局指针/指针型的全局标签

功能的程序中，不能将全局指针及指针型的全局标签作为表示程序的步的标签使用。

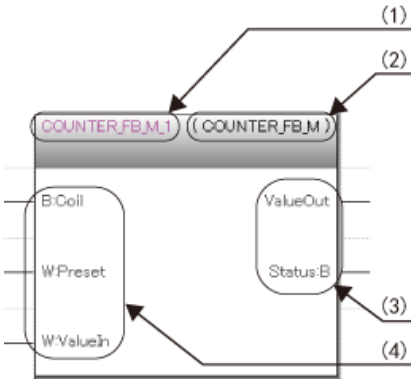
3.2 功能块(FB)

功能块是通过程序块及其他的功能块被使用的程序部件。

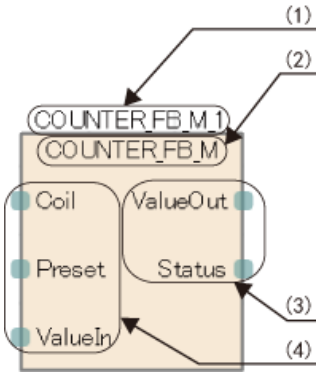


功能块与功能不同，不能保持返回值。
因为功能块能将值保存在变量中，因此也能保持输入状态及处理结果。
因为要在下一次处理中用到保持的值，所以即使是同样的输入值也不一定每次都输出同样的结果。

■梯形图语言的情况下



■FBD/LD语言的情况下



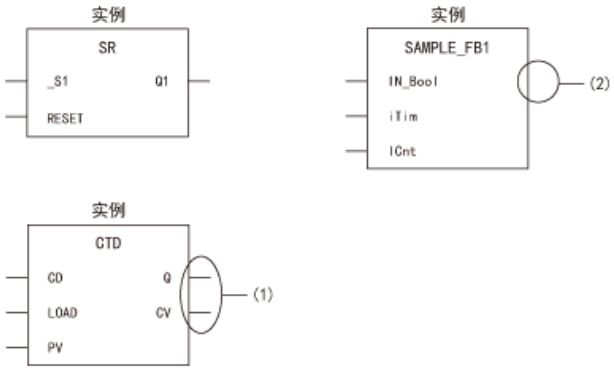
(1)功能名
(2)功能块名
(3)输出变量
(4)输入变量

此外，为了在程序上使用功能块，需要定义实例。

☞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。



(1) 可以返回多个输出。

(2) 也可以不输出。

关于可以设置的输入变量、输出变量、输入输出变量的分类，请参阅以下内容。

☞4.2 分类

关于内部变量

功能块可以使用内部变量。

关于可以设置的内部变量的分类，请参阅下述内容。

☞4.2 分类

关于外部变量

功能块可以使用外部变量。

关于可以设置的外部变量的分类，请参阅下述内容。

☞4.2 分类

实例

■ 实例含义

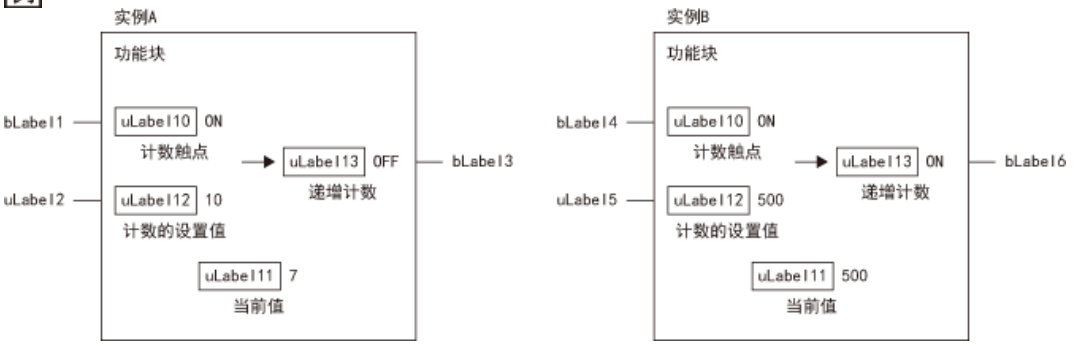
要使用功能块，需要创建实例。

通过创建功能块的实例，可以从程序及其他的功能块调用并使用。从一个功能块定义可以创建多个实例。

创建实例时，定义为全局标签或使用功能块的程序部件的局部标签。实例还可以作为数组进行定义。

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。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



以上是输入变量(计数触点)为ON时递增计数当前值，并且当当前值达到计数的设置值时，将输出变量(输出触点)置为ON的功能块。实例A与B是相同的功能块，但实例不同，因此实例A和B保持不同的值。上述示例的情况下，实例B的输出变量(输出触点)已经为ON，但实例A的输出变量(输出触点)的当前值未达到设置值，因此并未ON。

■ 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。

■ 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)”+“保留领域容量”

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。
保留领域容量	通过在RUN中写入添加锁存属性以外的局部标签以及局部实例的保留领域。(固定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)”+“保留领域容量”

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。
保留领域容量	通过在RUN中写入添加锁存属性以外的局部标签以及局部实例的保留领域。(固定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

关于EN/ENO

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

☞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，请参阅下述内容。

项目	参照目标
功能的创建方法	GX Works3操作手册
能写入CPU模块的FB/FUN文件数	用户手册(入门篇)

■ 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。

- 宏类型功能块
- 子程序类型功能块

关于详细内容，请参阅以下内容。

动作概要

模块FB、通用功能、通用功能块无法进行上述选择。

■ 可使用的软元件/标签

在功能块的程序中可使用的软元件及标签一览如下所示。

○：可以使用，△：只可在指令中使用(禁止作为表示程序的步的标签使用)

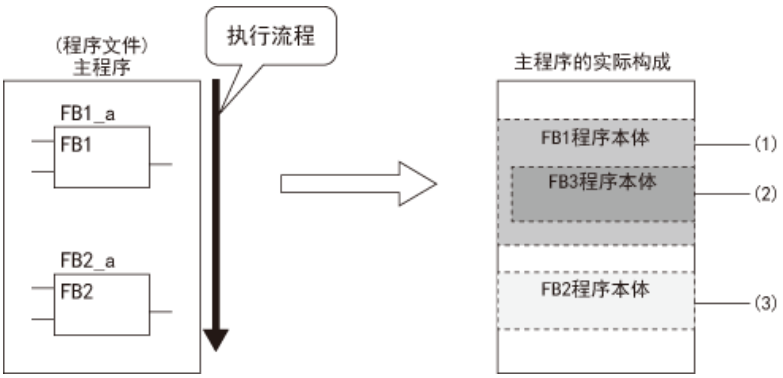
软元件/标签的类型		能否使用
标签(指针型以外)	全局标签	○
	局部标签	○
标签(指针型)	指针型全局标签	△
	指针型局部标签	○
软元件	全局软元件	○
指针	全局指针	△

动作概要

■ 宏类型功能块

宏类型功能块对于编程时的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一样执行展开的程序。

希望程序的处理速度优先的情况下，应使用宏类型功能块。



(1) 展开主程序中的FB1程序本体后执行。

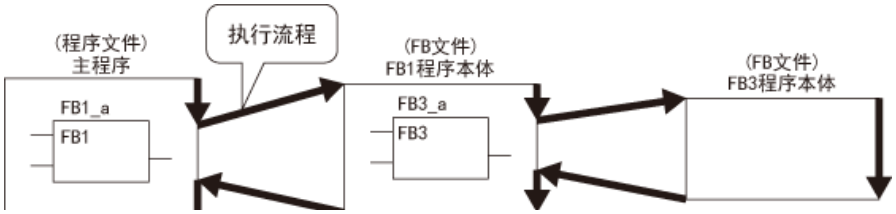
(2) FB3在FB1的程序本体中展开。

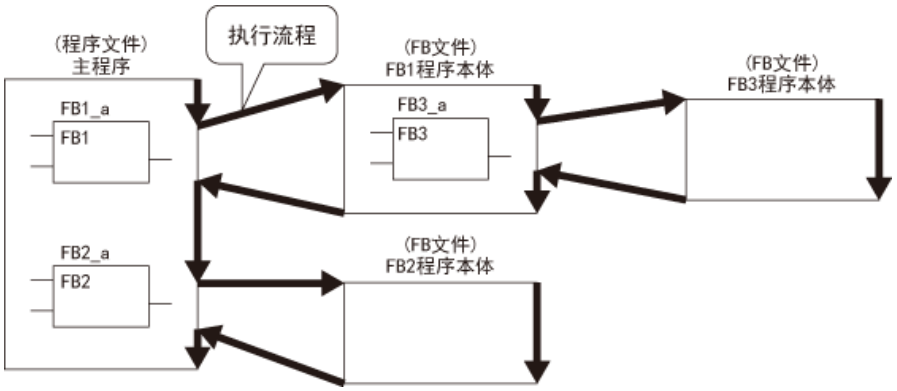
(3) FB2与FB1一样，FB2程序本体在主程序中展开后执行。

■ 子程序类型功能块

子程序类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文件内的程序本体进行执行。

希望使程序的容量变小的情况下，应选择子程序类型功能块。



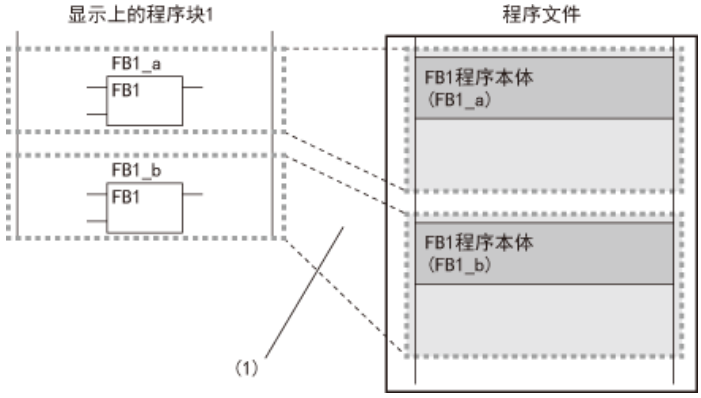


加上所有的功能块、功能，最多可嵌套32次。

宏类型功能块

■ 调用侧

调用宏类型功能块的情况下，转换时展开调用对象的程序本体。



(1) 程序本体在多个调用位置展开。

■ 程序本体

功能块的程序本体的步数与普通的程序一样为指令步数的合计。

关于各指令的步数，请参阅下述手册。

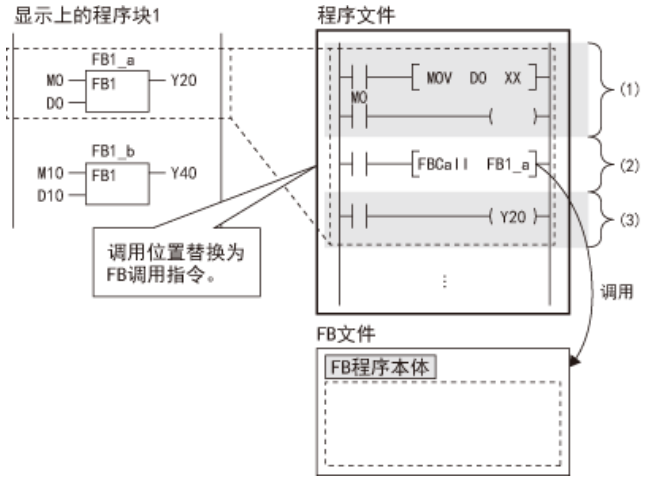
编程手册(指令/通用FUN/FB篇)

子程序类型功能块

■ 调用侧

调用子程序类型功能块的情况下，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。

但是，关于VAR_IN_OUT的变量，虽然将地址交接处理生成为自变量处理，但是不生成返回值交接处理。



(1) 自变量交接

(2) FB1程序本体调用

(3) 返回值交接

自变量交接

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

自变量的分	数据类型	使用指令	步数

类			
VAR_INPUT	位	LD+OUT LD+MOVB (根据使用的程序语言、功能的种类、输入自变量的种类的组合，使用其中的某一种。)	关于各指令的步数，请参阅下述手册。 编程手册(指令/通用FUN/FB篇)
	字[无符号]/位列[16位] 双字[无符号]/位列[32位] 字[带符号] 双字[带符号]	LD+MOV LD+DMOV	
	单精度实数	LD+EMOV	
	时间	LD+DMOV	
	字符串(32)	LD+$MOV	
	数组、结构体	LD+BMOV	

程序本体调用

调用功能块的程序本体需要12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	6
ENO	4

Point

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被变址修饰的情况
- 指定软元件的地址超过16位的情况
- 位数指定的情况

■ 程序本体

功能块的程序本体的步数与普通的程序一样为指令步数的合计。

关于各指令的步数，请参阅下述手册。

编程手册(指令/通用FUN/FB篇)

注意事项

■ 全局指针/指针型的全局标签

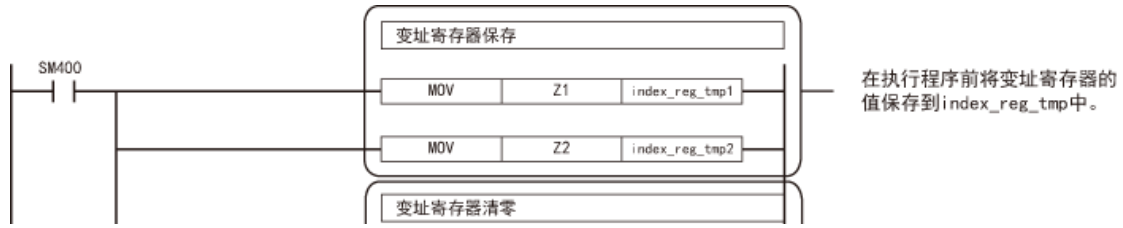
功能块的程序中，不能将全局指针及指针型的全局标签作为表示程序的步的标签使用。

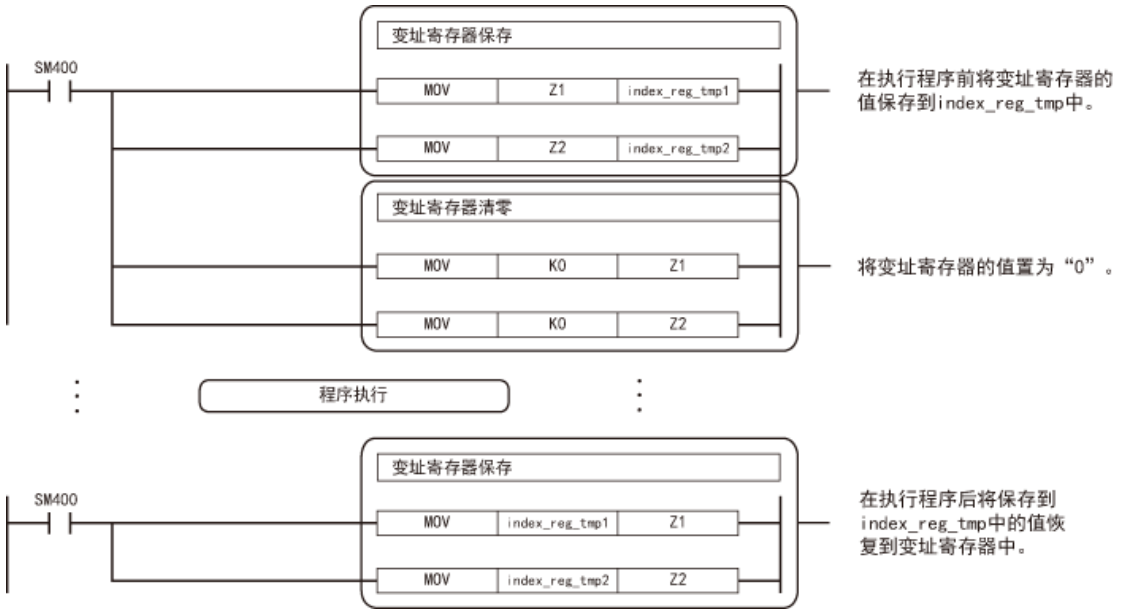
■ 使用变址寄存器的情况

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，需要保存回路与恢复回路。

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




变址寄存器保存
SM400
MOV
Z1
index_reg_tmp1
MOV
Z2
index_reg_tmp2
在执行程序前将变址寄存器的值保存到index_reg_tmp中。
变址寄存器清零
MOV
K0
Z1
MOV
K0
Z2
将变址寄存器的值置为“0”。
程序执行
变址寄存器保存
SM400
MOV
index_reg_tmp1
Z1
MOV
index_reg_tmp2
Z2
在执行程序后将保存到index_reg_tmp中的值恢复到变址寄存器中。

4 标签

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

4.1 类型

本手册针对下述的标签进行说明。

- 全局标签
- 局部标签

全局标签

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

■ 软元件的分配

全局标签可以分配任意的软元件。

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

局部标签

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

Point

作为标签类型，全局标签与局部标签以外有下述几种类型。

■系统标签

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

关于详细内容，请参阅以下内容。

📖开始吧iQ Workss

■模块标签

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

关于详细内容，请参阅以下内容。

📖MELSEC iQ-F FX5CPU模块FB参考

4.2 分类

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

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

4.3 数据类型

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

- 基本数据类型
- 总称数据类型(ANY型)

基本数据类型

基本数据类型包括如下所示的数据类型。

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

*1 时间类型在函数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。

编程手册(指令/通用FUN/通用FB篇)

*2 在时间类型的标签中使用常数的情况下，应在起始添加“T#”。

■ 关于定时器与计数器的数据类型

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

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

*1 当前值的值通过指令名指定单位

*1 当前值的值通过指令名指定单位。

*2 利用OUT LC指令使用时:0～4294967295

利用UDCNTF指令使用时:-2147483648～+2147483647

关于各软元的动作的详细内容，请参阅下述手册。

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。(4.5 结构体)

总称数据类型(ANY型)

是汇总若干个基本数据类型标签的数据类型。数据类型名以“ANY”开始。

在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意一种。

关于与总称数据类型的类型相对应的基本数据类型，请参阅下述手册。

编程手册(指令/通用FUN/FB篇)

可以定义的数据类型

能否设置在标签的各分类中可以定义的数据类型如下所示。

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

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

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

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

*1 不可以定义指针类型。

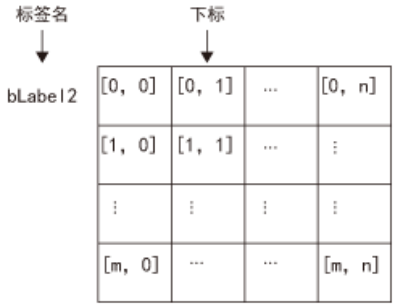
*2 不可以定义定时器型、累计定时器型、计数器型、长计数器型。

4.4 数组

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

■1次元数组的图像

■2次元数组的图像



数组的定义

■ 数组的要素

定义数组时，应决定要素数(数组的长度)。要素数的范围请参照下述内容。

☞ 数组要素数的范围

■ 定义的格式

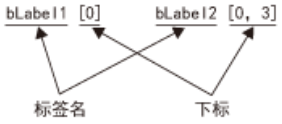
下面以直到3次元数组为例，对定义的格式进行说明。

数组开始值～数组结束值之间的范围即为要素数。

数组的次元数	格式	备注
1次元数组	基本数据类型/结构体名的数组(数组开始值..数组结束值)	•关于基本数据类型: ☞ 基本数据类型 •关于结构体名: ☞ 4.5 结构体
	【定义示例】 位(0..2)	
2次元数组	基本数据类型/结构体名的数组(数组开始值..数组结束值，数组开始值..数组结束值)	
	【定义示例】 位(0..2，0..1)	
3次元数组	基本数据类型/结构体名的数组(数组开始值..数组结束值，数组开始值..数组结束值，数组开始值..数组结束值)	
	【定义示例】 位(0..2，0..1，0..3)	

使用方法

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



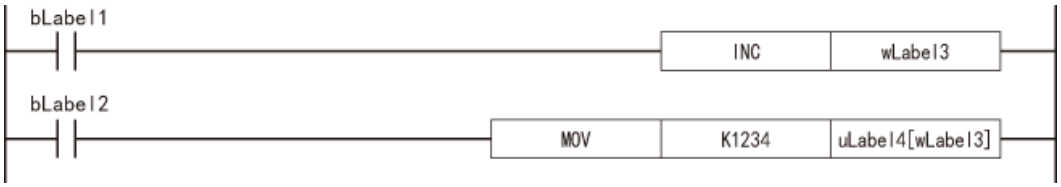
数组中的下标可以指定为下述类型。

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

Point

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

```
bLabel1
```



- 梯形图语言的情况下，使用数组时可省略要素编号。使用时省略了要素编号的情况下，作为数组要素的起始编号进行转换。例如所定义的标签名为“boolAry”，数据类型为“位(0..2,0..2)”的数组时，“boolAry[0,0]”和“boolAry”会进行同样处理。
- 能够在使用数组的指令或功能、功能块的设置数据中指定多次元的数组。此时，数组要素中最右侧的要素会作为一次元数组加以处理。

数组要素数的范围

数组的最多个数根据数据类型而不同。

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

注意事项

■ 使用中断程序的情况下

在数组的下标中指定了标签或软元件的情况下，组合多个指令进行运算。

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

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

关于DI/EI 指令的详细内容，请参阅下述内容。

编程手册(指令/通用FUN/FB篇)

■ 关于数组的要素

对于定义的数组的要素数，请勿访问要素编号以外的范围。

用常数指定通过数组定义的范围外的下标的情况下，将变为转换出错。

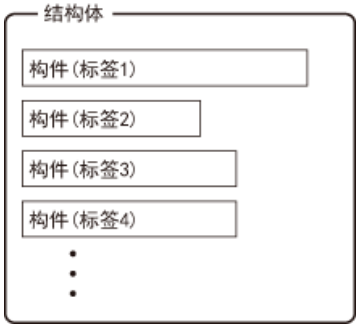
此外，用常数以外指定数组的下标的情况下不会变为转换出错，而是在执行时访问其他标签区域、锁存标签区域的领域后进行处理。

4.5 结构体

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

结构体的创建

创建结构体首先要创建结构体的定义，其次在创建的结构体中定义构件。



使用方法

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

例 使用结构体构件的情况下

Point

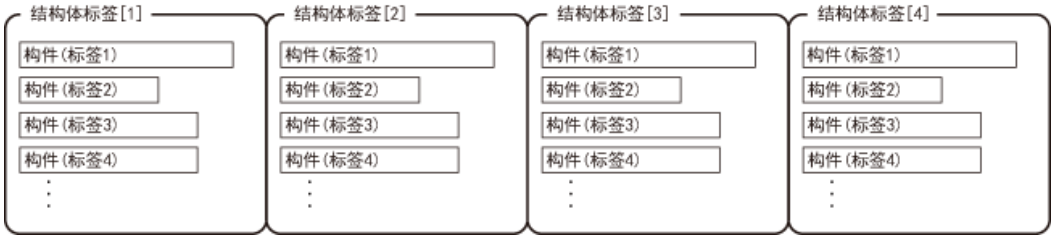
- 在结构体中定义多个数据类型后登录标签，在程序中使用的情况下，转换后的数据存储的顺序不会变为定义了数据类型的顺序。利用工程工具进行转换时，分类为标签类型与数据类型后进行分配。(根据填充块进行存储器分配)

  GX Works3操作手册

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

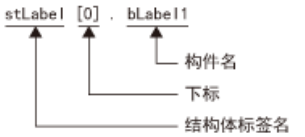
结构体的数组

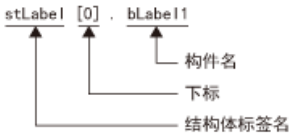
可以将结构体数组化后使用。



作为数组声明的情况下应在结构体标签名后用“[]”将下标括起来表示。
可以将结构体的数组作为功能及功能块的自变量进行指定。

例 使用数组化的结构体的要素的情况下





可以指定的数据类型

下述数据类型可以作为结构体的构件进行指定。

- 基本数据类型
- 指针型
- 数组
- 其他结构体

结构体的类型

下述数据类型预先被定义为结构体。

类型	参照目标
定时器类型	☞ 4.3 数据类型
累计定时器类型	
计数器类型	
长计数器类型	

4.6 常数

常数的类型

在标签中设置常数时的标记如下所示。

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

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

在字符串类型的常数中使用“$”的情况下

“$”作为转换序列使用。

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

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

但是，紧接着“$”的字符在下述情况下不会变为错误。

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

4.7 注意事项

有限制的功能

在下述功能中使用标签时有限制。

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

■ 定义分配软元件的全局标签并使用的情况下

按照下述顺序定义全局标签后应使用对标签的使用有限制的功能。

另外，由于是在软元件区域消耗软元件/标签存储器，所以应确保软元件区域。

1. 确保所使用的软元件区域。

   CPU参数 ⇨ 存储器/软元件设置 ⇨ 软元件/标签存储器区域容量设置

2. 在全局标签中定义标签后手动分配软元件。

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

■ 将所使用的标签的值暂时复制到其他软元件中的情况下

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

另外，由于是在软元件区域消耗软元件/标签存储器，所以应确保软元件区域。

1. 确保所使用的软元件区域。

   CPU参数 ⇨ 存储器/软元件设置 ⇨ 软元件/标签存储器区域容量设置

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

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

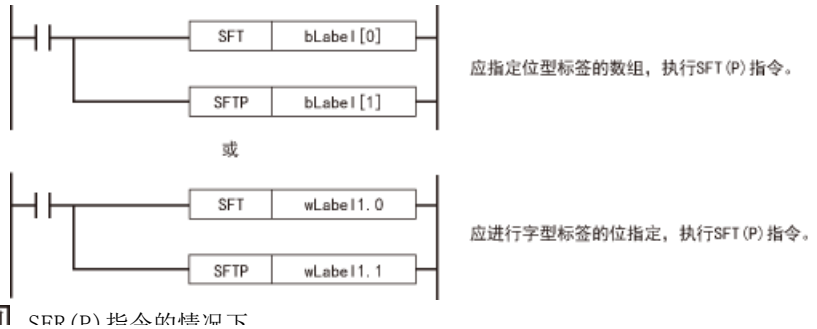
Point

应在考虑将值写入标签的时机及功能的执行时机后决定传送指令的添加位置。

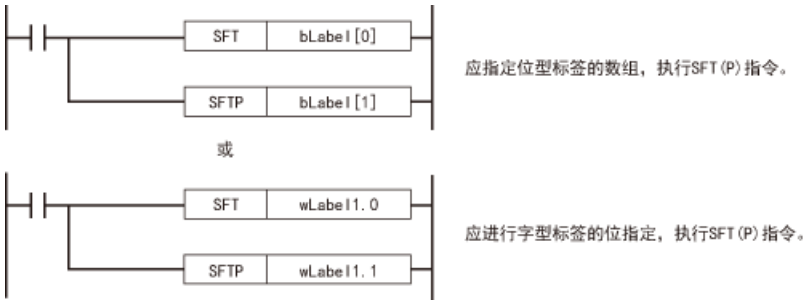
创建程序时的注意事项

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

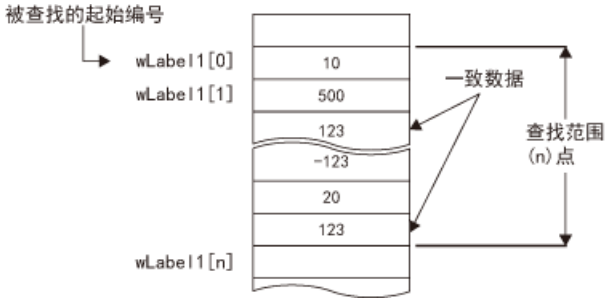
例 SFT(P)指令的情况下



例 SFR(P)指令的情况下



例 SFR(P)指令的情况下



指定的标签的数组应指定具有比查找范围(n)点更大的范围的标签。

标签名的限制

标签名中存在以下限制。

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

关于保留字的详细内容，请参阅下述手册。

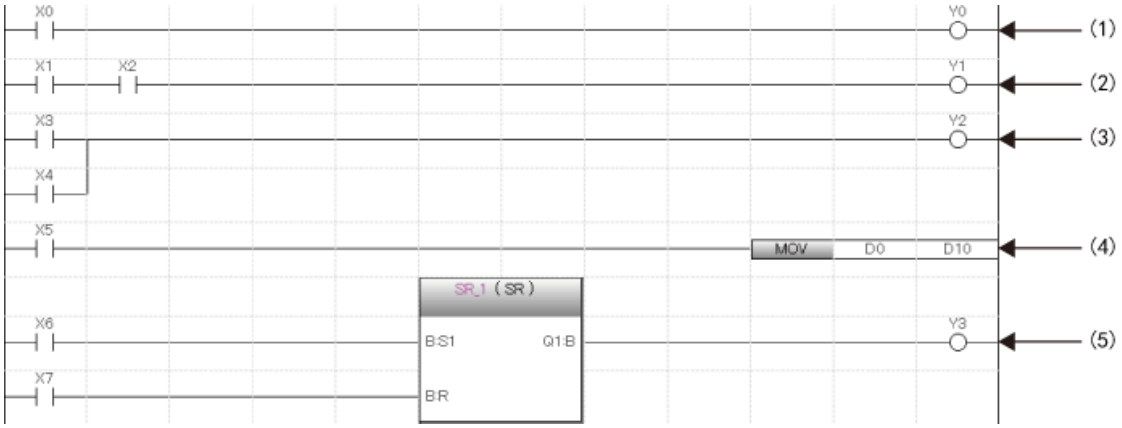
GX Works3操作手册

5 梯形图语言

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

5.1 配置

使用梯形图语言可以创建下述的回路。



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

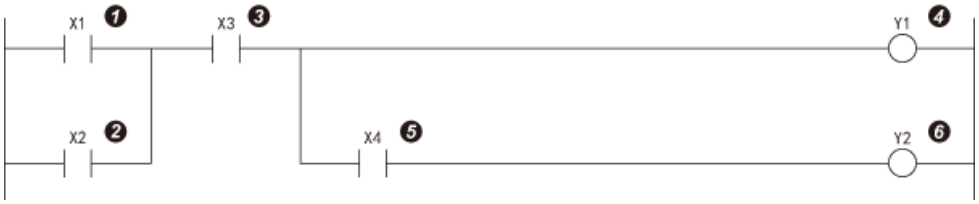
回路符号

可以在梯形图语言的编程中使用的回路符号如下所示。

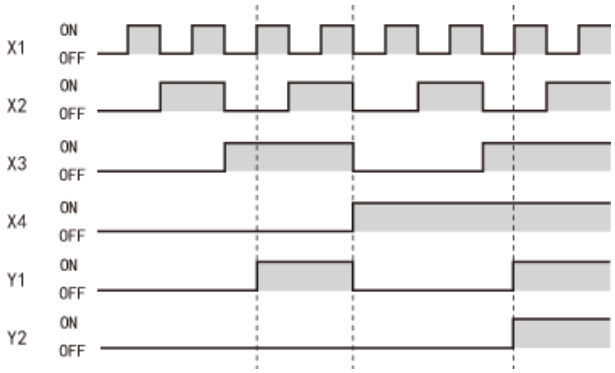
要素	符号	说明
常开触点		指定软元件或标签变为ON时导通。
常闭触点		指定软元件或标签变为OFF时导通。
上升沿脉冲		指定软元件或标签上升沿时(OFF → ON)导通。
下降沿脉冲		指定软元件或标签下降沿时(ON → OFF)导通。
非上升沿脉冲		指定软元件或标签OFF时、ON时以及下降沿时(ON → OFF)导通。
非下降沿脉冲		指定软元件或标签OFF时、ON时以及上升沿时(OFF → ON)导通。
运算结果上升沿脉冲化		运算结果上升沿时(OFF → ON)导通。 运算结果在上升沿以外的情况下不导通。
运算结果下降沿脉冲化		运算结果下降沿时(ON → OFF)导通。 运算结果在下降沿以外的情况下不导通。
运算结果取反		将在开始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	SET M0	在[]内执行指定指令。
换行	K0	超过了一个回路行中可以创建的触点数的情况下，创建换行处的符号及换行目标的符号，执行回路的换行。
功能	ABS IN ABS	执行功能。 •功能的创建方法(GX Works3操作手册) •通用功能(编程手册(指令/通用FUN/FB篇))
功能块	Instance (CTD) B:CD Q:B B:LD CV:W W:PV	执行功能块。 •功能块的创建方法(GX Works3操作手册) •通用功能块(编程手册(指令/通用FUN/FB篇)) •模块标签(MELSEC iQ-F FX5CPU模块FB参考)

程序执行顺序

按照下述的编号顺序执行。



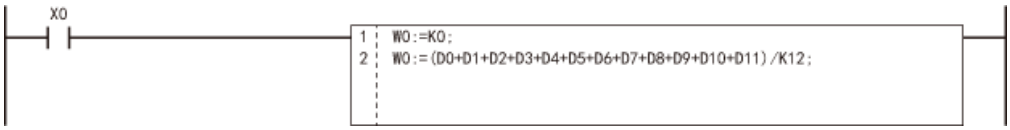
执行了上述程序的情况下，根据X1～X4的ON/OFF，Y1、Y2的ON时机如下所示。



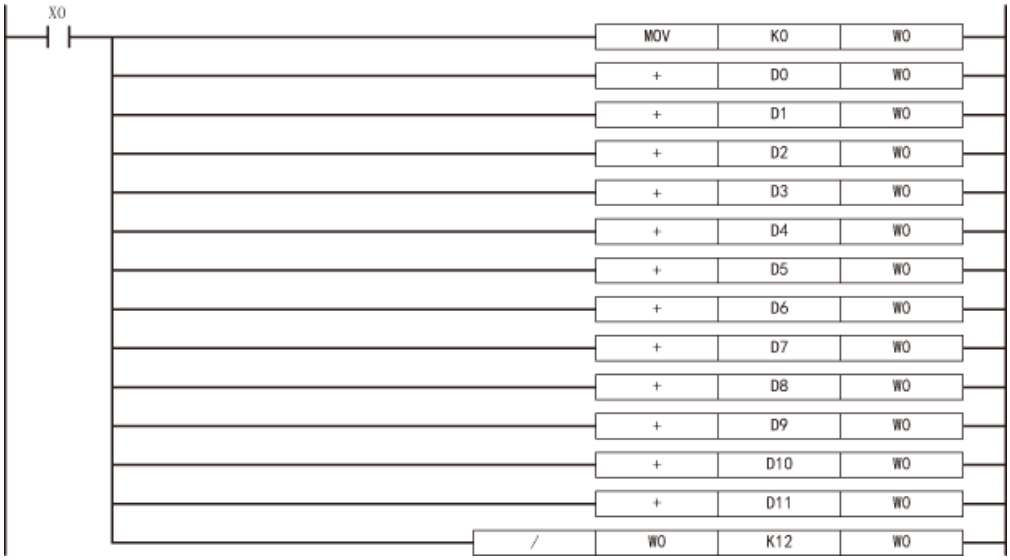
5.2 内嵌ST

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

- 使用了内嵌ST情况下的程序



- 不使用内嵌ST情况下的程序



规格

关于内嵌ST的规格，请参阅ST语言的规格。

☞6 ST语言

注意事项

- 梯形图程序的1行中只能创建一个内嵌ST。
- 在梯形图程序的1行中无法同时使用功能块与内嵌ST框。
- 如果在触点相应的指令位置创建内嵌ST框，在线圈相应的指令位置也会创建内嵌ST框。
- 内嵌ST内最多可输入字符数为2048个。(换行作为一个字符计数。)
- 内嵌ST内，有时上升执行指令、下降执行指令、特殊定时器指令、通用功能块的边缘检测功能块以及计数器功能块不能正常运行，因此请勿使用。
- 如果在内嵌ST内使用“RETURN语句”，则不会结束程序块的处理，而是结束内嵌ST框内的处理。

5.3 声明/注解

在梯形图回路中可以显示声明与注解。

声明

通过使用声明，可以对回路块添加注释。通过添加注释，处理等流程变得易懂。

声明中有行间声明/P 声明/I 声明。

行间声明可以在导航窗口的树状结构上显示。

■ 行间声明

对整个回路块添加注释。

■ P声明

对指针编号添加注释。

■ I声明

对中断指针编号添加注释。

注解

通过使用注解，可以对程序中的线圈及指令添加注释。

通过添加注释，线圈及指令的内容等变得易懂。

声明/注解的类别

声明与注解的类别分为“PLC”与“外围”。

类别	类型	内容
PLC	•行间声明 •P 声明 •I 声明 •注解	可以将声明/注解存储在CPU模块中。 PLC 声明需要消耗下述步数。(全部是半角输入的情况。小数点以后进位。) •无字符:3步 •有字符:4+(字符数+2+14)/5+字符数
外围	•行间声明 •P 声明 •I 声明 •注解	不可以将声明/注解存储在CPU模块中。(仅存储位置信息。) 有必要保存在外围设备中。 每一行消耗一步。 在输入的文本前自动添加*印记。

6 ST语言

ST语言是在规定逻辑记述方式的国际标准IEC61131-3中定义的语言。ST语言是具有与C语言等相似的语法结构的文本形式的程序语言。适用于对梯形图语言难以表现的复杂处理进行编程的情况。

ST语言支持控制语法、运算式、功能块(FB)、功能(FUN)，可以进行如下的记述。

例 通过条件语句进行选择分支，通过重复语句进行重复等的控制语法

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

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

例 使用运算符(*、/、+、-、<、>、=等)的表达式

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

例 定义的功能块的调用

```
//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

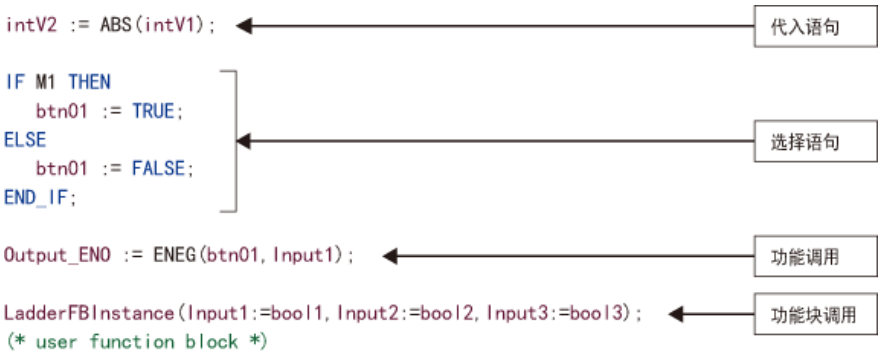
```
(*将BOOL型数据转换为INT型/DINT型数据*)
wLabel2 := BOOL_TO_INT (bLabel1);
```

例 汉字等全角字符的使用

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

6.1 配置

ST语言中的编程由运算符与语句组成。



语句的终端必须添加“;”(分号)

语句的终端必须添加“;”(分号)。

空格、制表、换行可以插入到运算符及数据中。

可以在程序中插入注释。

程序的结构要素

ST程序由以下要素构成。

项目		示例	参照页
段落符号		; , ( )	段落符号
运算符		+、-、<、>、=	运算符
保留字	语句	IF、CASE、WHILE、RETURN	语句
	软元件	X0、Y10、M100	用户手册(应用篇)
	数据类型	BOOL、DWORD	4.3 数据类型
	函数	ADD、REAL_TO_STRING_E	编程手册(指令/通用FUN/FB篇)
常数		123、‘abc’	常数
标签		Switch_A	标签与软元件
注释		(*置为ON*)、//置为ON、/*置为ON*/	注释
其他符号		半角空格、换行代码、TAB代码	—

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

GX Works3操作手册

段落符号

在ST语言中，为了明确程序的结构，设有下述的段落符号。

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

运算符

在ST程序中使用的运算符、对象数据类型与运算结果的数据类型如下所示。

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

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
功能()	功能的自变量	CONCAR(‘AB’、‘CD’)	2
**	幂运算	3.0**4	3
-	符号取反	-10	4

-	符号取反	-10	4
NOT	位型补数	NOT TRUE	
*	乘法运算	10 * 20	5
/	除法运算	20 / 10	
MOD	求余数运算	17 MOD 10	
+	加法运算	1.4 + 2.5	6
-	减法运算	3 - 2	
<、>、<=、=>	比较	10 > 20	7
=	一致	T#26h = T#1d2h	8
<>	不一致	8#15 <> 13	
&、AND	逻辑积	TRUE AND FALSE	9
XOR	逻辑异或	TRUE XOR FALSE	10
OR	逻辑或	TRUE OR FALSE	11

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

语句

可以在ST程序中使用的语句如下所示。

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

应用半角字符记述语句。

代入语句

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

使用数组型标签及结构体标签的情况下，应注意代入语句的左边与右边的数据类型。

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

例 intAry1 := intAry2;

结构体标签的情况下，需要使数据类型(结构体的数据类型)的左边与右边相同。

例 dutVar1 := dutVar2;

■ 数据类型的自动转换

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

例 自动转换的示例

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

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

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

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

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

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

换为单精度实数。

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

代入算数运算的结果时的注意事项请参阅下述内容。

☞■ 使用算数运算公式时

子程序控制语句

■ 功能块调用语句

格式	内容
实例名(输入变量1:=变量1,...输出变量1=>2,...);	在实例名后，用“()”括住输入变量、输出变量的代入语句。 多个变量的情况下，各代入语句之间用“,”(逗号)隔开。
实例名.输入变量1:=变量1; ⋮ 实例名(); 变量2:=实例名.输出变量1;	在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使用的符号与可分配公式如下所示。

类型	内容	属性	使用符号	可分配公式
VAR_INPUT	输入变量	无，或RETAIN	:=	所有的公式
VAR_OUTPUT	输出变量	无，或RETAIN	=>	只有变量
VAR_IN_OUT	输入输出变量	无	:=	所有的公式
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“.”(句点)指定输出变量，代入变量被存储。

功能块	FB定义	记述示例
1个输入变量、1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;

■ 功能调用语句

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“,”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)	Outout1 := ABS(Input1);
输入变量为3个功能的情况下(例：MAX)	Outout1 := MAX(Input1, Input2, Input3);
具有EN/ENO的功能(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。)

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

■ RETURN语句

语句	格式	内容	记述示例
■RETURN	RETURN;	为了使程序、功能块、功能在中途结束而使用。 如果在程序中使用了RETURN语句，将跳转到程序的最后语句的下一步。 如果在功能块中使用了RETURN语句，将从功能块返回。 如果在功能中使用了RETURN语句，将从功能返回。 相对于1个RETURN语句，在系统使用1点指针型标签。	IF bool1 THEN RETURN; END_IF;

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

选择语句

语句	格式	内容	记述示例

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

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

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

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

重复语句

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

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

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

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

注意事项

■ 使用代入语句时

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

■ 使用算数运算公式时

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

的数据类型之后再进行运算。

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

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

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

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

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

■ 在算术运算公式中使用符号取反的运算符时

如果相对于数据类型的最小值使用符号取反的运算符(-)，则会为相同值。

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

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

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

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

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

例 在算术运算之前自动转换数据类型时

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

例 不使用符号取反的运算符时

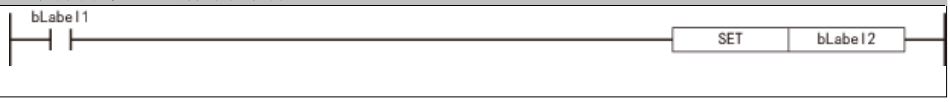
```
varDint2 := varDint1 - varInt1;
```

■ 使用位型标签时

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

例 始终ON程序

ST 程序	ST 程序同等处理的梯形图程序

IF bLabel1 THEN
  bLabel2 := TRUE;
END_IF;
	

为避免始终ON，应按下述方式添加将位型标签置为OFF 的程序。

例 避免始终ON 的程序

ST 程序*1	ST 程序同等处理的梯形图程序

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

*1 上述程序可以按下述方式记述。

bLabel2 := bLabel1;

或

OUT(bLabel1,bLabel2);

但是，在<语句>内使用了OUT 指令的情况下，变为与始终ON程序相同的状态。

■ 使用定时器功能块、计数器功能块时

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

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并且bLabel2=ON时，开始计数。                    *)
(*bLabel1=ON并且bLabel2=OFF时，计数清零。                   *)
(*bLabel1=OFF并且bLabel2=ON时，停止计数。计数值不清零。      *)
```

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

■更改后程序示例

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

计数器功能块的情况下

■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并且bLabel2=ON/OFF时，计数+1。  *)
(*bLabel1=OFF并且bLabel2=ON/OFF时，不计数。*)
(*bLabel1=ON/OFF不与计数+1联动。            *)
```

■更改后程序示例

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

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

■ 使用FOR...DO语句时

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

■ 使用上升执行指令、下降执行指令时

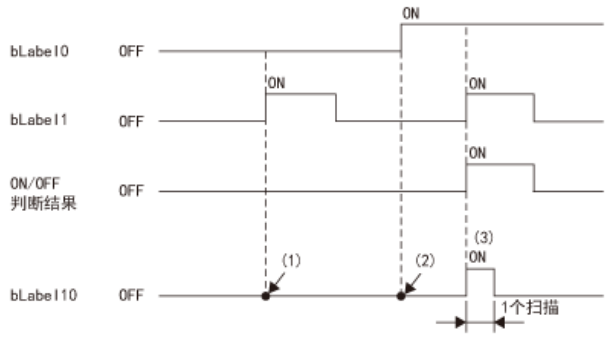
以下显示在IF语句和CASE语句中使用上升执行指令、下降执行指令时的动作。

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

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

例 IF语句中使用了PLS指令(上升执行指令)时

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



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

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

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

■ 使用主控指令时

显示主控OFF时的动作。

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

例 选择语句(IF语句)内的语句

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

例 选择语句或者重复语句以外的语句(位代入语句时)

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

例 选择语句/重复语句以外的语句 (OUT指令时)

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

常数

常数的标记方法

ST程序中字符串的标记方法如下所示。

数据类型		标记方法	标记示例
字符串(32)	STRING	将字符串用单引号(’)括起来。	Stest :=‘ABC’;

上述以外的常数的标记方法请参阅下述内容。

☞4.6 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内容。

☞4 标签

关于可使用的软元件请参阅下述内容。

用户手册(应用篇)

■ 附带类型指定的软元件标记

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

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

可以使用软元件类型指定符的软元件如下所示。

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

■ 软元件的指定方法

关于软元件的指定可以使用下述方法。

- 变址修饰
- 位指定

- 位指定
- 位数指定
- 间接指定

关于详细内容，请参阅以下内容。

用户手册(应用篇)
编程手册(指令/通用FUN/FB篇)

注意事项

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

例 D0 := K5X0;

在上述情况下，因为K5X0为双字型、D0为字型，程序出错。

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

例 K5X0 := 2#1011_1101_1111_0111_0011_0001;

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

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

例 varInt := WORD_TO_INT(TN0); (*使用类型转换功能*)

注释

可以在ST程序中使用的注释如下所示。

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

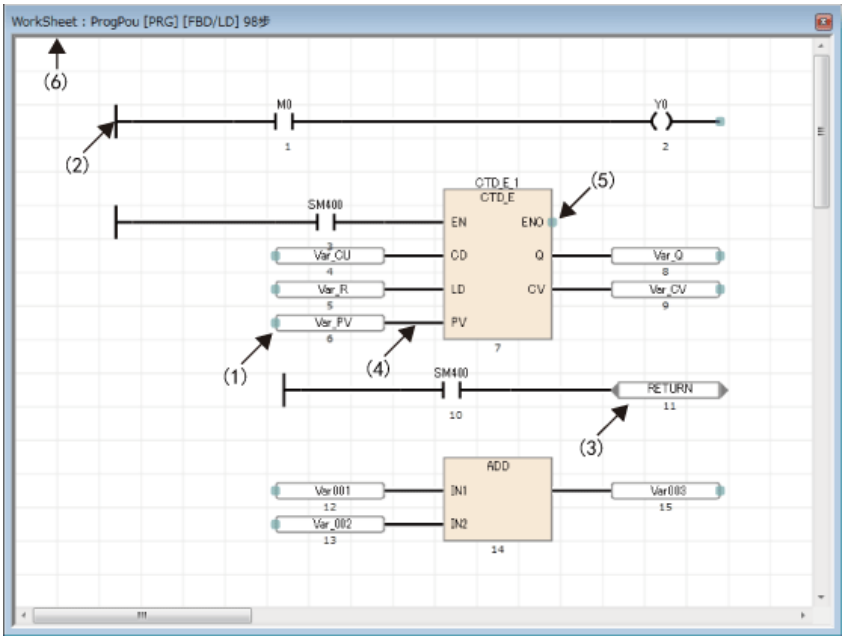
在多行注释中请勿记述含有结束符号的注释。

7 FBD/LD语言

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

7.1 配置

使用FBD/LD语言可以创建下述的程序。



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

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

程序部件

FBD部件

显示构成FBD/LD程序的部件。

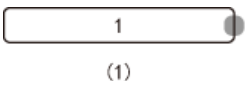
要素	符号	说明
变量	D0 bLabel1	为了存储各个值(数据)，会使用变量。变量中事先规定了数据类型，仅存储该数据类型的值(数据)。 可在变量中指定标签或者软元件。
常数	100	输出所指定的常数值。
功能(FUN)	ADD IN1 IN2	执行功能。 •功能的创建方法(GX Works3操作手册) •通用功能(编程手册(指令/FUN/通用FB篇))
功能块(FB)	CTU_1 CTU CU Q R CV PV	执行功能块。 •功能块的创建方法(GX Works3操作手册) •通用功能块(编程手册(指令/通用FUN/FB篇)) •模块标签(MELSEC iQ-F FX5CPU模块FB参考)

■ **常数部件的数据类型**

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

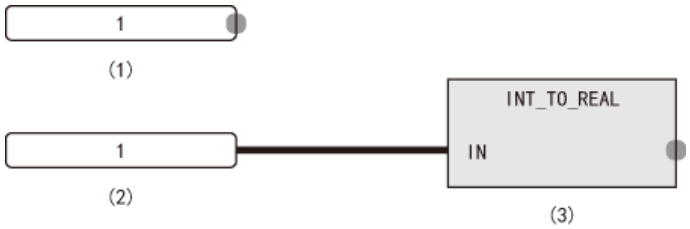
例 常数值中输入了1时

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



(1)

(1)数据类型未决定
(2)INT型



(1)数据类型未决定
(2)INT 型
(3)INT 型

■ 数据类型的自动转换

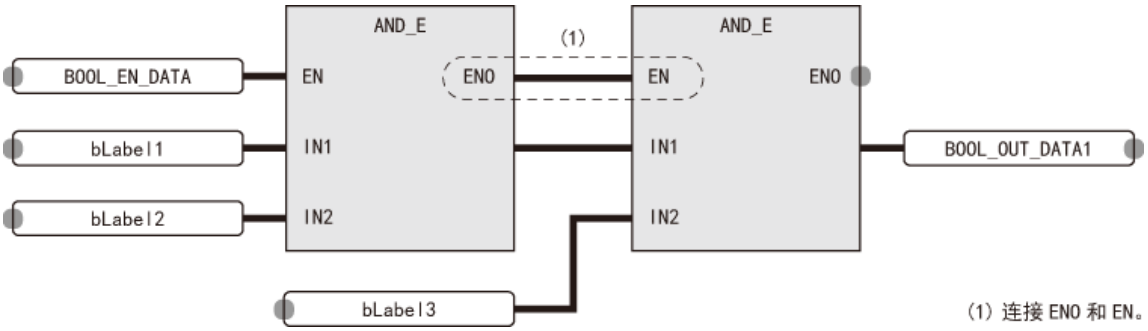
要连接的部件的数据类型不同时，有时会自动转换数据类型。

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

☞ ■ 数据类型的自动转换

■ 功能的输入输出点

- 功能必须事先将输入点与所有其他FBD 部件连接。
- 功能的输入变量与输出变量中，必须事先决定数据类型，并使连接至输入点和输出点的FBD 部件也与其一致。
- 将CPU 模块用指令，模块专用指令的输出变量(不包括ENO)连接至其他功能(或者功能块)的输入变量时，请通过变量部件。
- 从附带EN 的功能连接至功能的程序中，请将功能置为附带EN 功能，使其成为连接ENO 和EN 的程序，以免功能使用不定值。



(1) 连接 ENO 和 EN。

LD部件

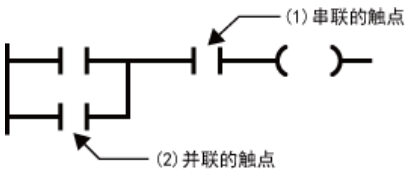
显示能够在FBD/LD 语言的程序中使用的梯形图语言的部件。

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

■ 触点符号的AND运算和OR运算

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

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



通用部件

显示配置在FBD/LD 编辑器上的通用部件。

要素	符号	说明
跳转	Jump	将执行处理从跳转部件跳转至跳转标签。跳转的部分不执行。 根据对跳转部件的ON/OFF信息，控制是否实施跳转。 ON:将执行处理跳转至跳转标签。 OFF:不进行跳转，而是实施执行处理。
跳转标签	JUMP	成为来自同一程序内的跳转指令的跳转目标。根据跳转标签以后的执行顺序的程序，执行处理。
连接器	CONNECTOR	将其作为连接线的替代来使用。 处理会移动至成对的连接器部件。 对于一个输出连接器，能够使用一个或多个输入连接器。
返回	RETURN	中断程序上的返回部件以后的处理。用于不执行返回部件以后的程序和功能、功能块的处理时。 根据对返回部件的ON/OFF信息，控制是否实施返回处理。 ON:执行返回处理。 OFF:不实施返回处理，而是实施通常的执行处理。
注释	Comment	用于记载注释时。

■ 跳转部件的注意事项

- 通过跳转部件使线圈为ON的定时器跳转时，无法实施正常的测量。
- 能够将跳转标签配置在跳转部件的上侧(执行顺序为前)。此时，请包含从重复中抽出的方法在内创建程序，以免超出监视时钟的设置值。
- 跳转部件和跳转标签中，仅能指定指针型的局部标签。不能使用指针软元件。
- 不能使用指针分支指令(CJ)。跳转时，请使用跳转部件。
- 不能向程序块的外侧跳转或从外侧跳转。以下显示不能执行的有关跳转的动作。
  - 向程序块的外侧跳转*1
  - 来自程序块的外侧跳转*1
  - 调用子程序
  - 作为子程序进行调用

*1 包括通过BREAK指令进行的分支。

■ 返回部件的动作

根据使用的程序和功能、功能块，返回部件的动作会有所不同。

使用的程序部件	内容
程序	结束程序部件的执行。
功能	结束功能，并且返回调用了功能的指令的下一步。
功能块	结束功能块，并且返回调用了功能块的指令的下一步。

通过宏类型功能块使用了返回部件时，请勿配置多个实例名相同的功能块部件。

转换使用了返回部件的程序后，局部标签"_SYSTEM_RETURN"会自动登录。

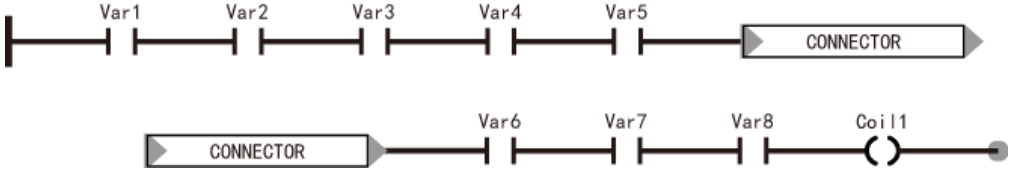
对于"_SYSTEM_RETURN"，存有下述操作限制。

对于自动登录的标签的操作	可否操作
变更标签名	不可操作*1
变更数据类型	不可操作
变更分类	不可操作
删除标签	不可操作*1
变更登录行	可操作

*1 变更或删除后再次进行转换时，要登录新的局部标签。

■ 关于连接器部件

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



连接线

是在FBD部件、LD部件、通用部件的连接点间进行连接的线。

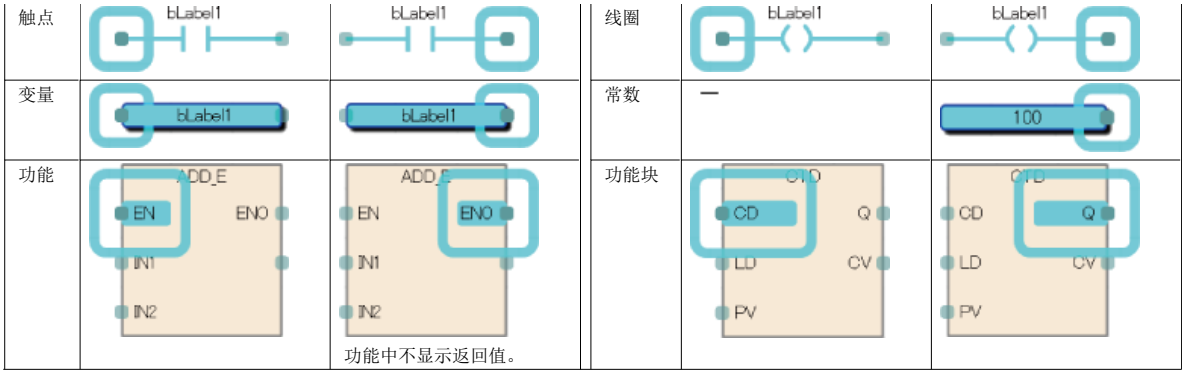
连接部件，将数据从左端交向右端。所连接的部件的数据类型必须一致。

连接点

是通过连接线连接FBD部件、LD部件、通用部件时的端点。

各部件的左侧的点表示输入侧，右侧的点表示输出侧。

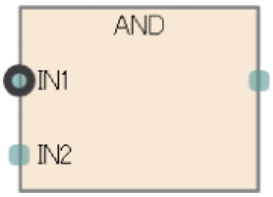
部件	输入连接点	输出连接点	部件	输入连接点	输出连接点
触点	bLabel1	bLabel1	线圈	bLabel1	bLabel1



连接点在连接后会隐藏。

■ **输入输出点的取反**

能够在连接点将至部件的输入或者来自部件的输出取反。
取反状态的连接点会被黑圈包围，并将输入或输出的数据取反(FALSE → TRUE或者TRUE → FALSE)。
可取反的数据类型为BOOL、WORD、DWORD、ANY_BIT、ANY_BOOL。



工作表

工作表是用来插入程序部件并连接的作业领域。

常数

常数的标记方法

FBD/LD程序中字符串的标记方法如下所示。

数据类型		标记方法	标记示例
字符串(32)	STRING	将字符串用单引号(')括起来。	'ABC'

上述以外的常数的标记方法请参阅下述内容。

☞ 4.6 常数

标签与软元件

指定方法

在FBD/LD程序中可以直接记述并使用标签与软元件。标签与软元件可以在部件的输入点、输出点、通用功能/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内容。

☞ 4 标签

关于可使用的软元件请参阅下述内容。

用户手册(应用篇)

■ **附带类型指定的软元件标记**

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

关于可与软元件型指定符使用的软元件，请参阅下述内容。

☞ ■ 附带类型指定的软元件标记

不指定字软元件的数据类型时，根据软元件的种类来决定数据类型。

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

注意事项

■ **使用标签时**

- 不能使用数组的下标中名称的末尾为"_"的标签。但是，将用于下标中的软元件/标签代入到其他软元件/标签中，并且将代入目标的软元件/标签指定为下标后，即可使用。

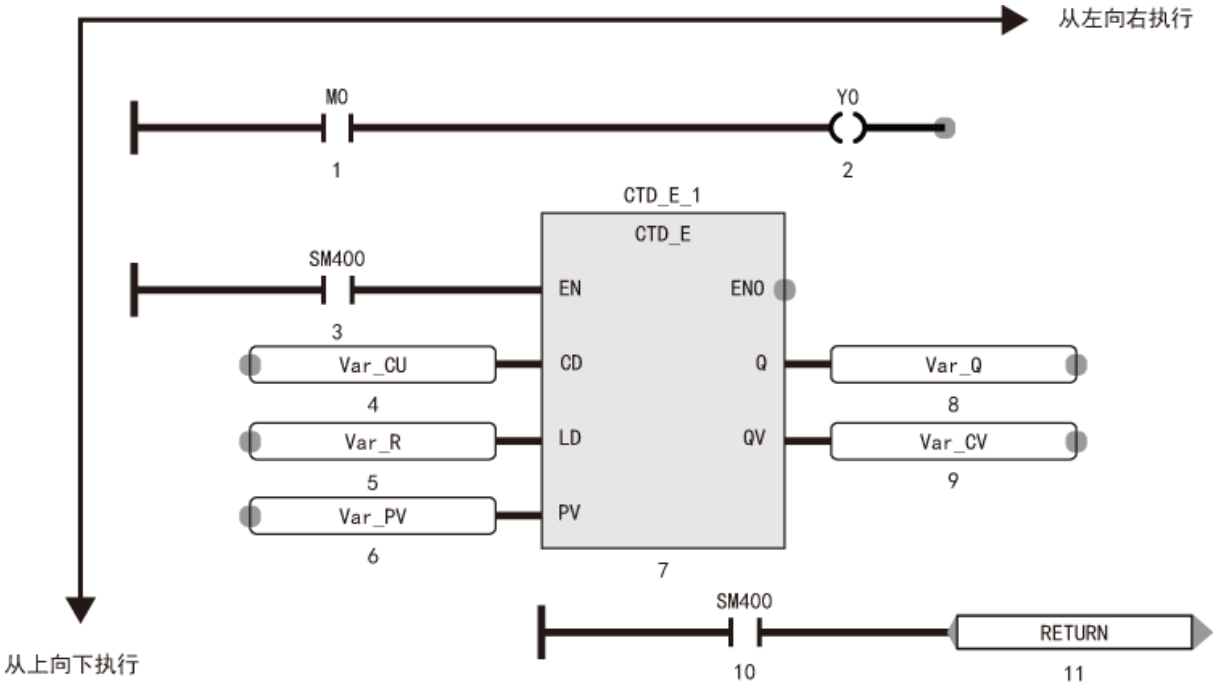
并且将代入目标的软元件/标签指定为下标后，即可使用。

- 不能指定名称的末尾为"_"的标签(结构体、功能块)的构件。
- 不能在名称的末尾为"_"的标签(数组)中指定下标。

7.2 程序执行顺序

程序部件的执行顺序

根据部件的位置关系和连接状况，决定FBD/LD编辑器上的部件的执行顺序。



配置在FBD/LD编辑器上的各部件中，会显示执行顺序的编号。

修订记录

制作日期	版本号	内容
2015年2月	A	制作初版

关于保修

在使用时，请务必确认一下以下的有关产品保证方面的内容。

1. 免费保修期和免费保修范围

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

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

【免费保修范围】

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

(2) 即使是在免费保修期内，但是如果属于下列的情况的话就变成收费的维修。

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

②由于用户擅自改动产品而引起的故障。

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

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

⑤即使按照正常的使用方法，但是继电器触点或是触点到寿命的情况。

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

⑦在本公司产品出厂时的科学技术水平下不能预见的原因引起的故障。

⑧其他、认为非公司责任而引起的故障。

2. 停产后的收费保修期

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

(2) 不提供停产后的产品（包括附属品）。

3. 在海外的服务

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

4. 对于机会损失、二次损失等保证责任的免除

无论是否在保修期内，对于不是由于本公司的责任而导致的损害；以及由于本公司产品的故障导致用户或第三方的机会损失、利益损失，无论本公司是否可以预见，由于特别的原因导致出现的损害、二次损害、事故赔偿，损坏到本公司以外产品，以及对于用户的更换产品工作，现场机械设备的重新调试、启动试运行等其他业务的补偿，本公司都不承担责任。

5. 产品规格的变更

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

6. 关于产品的适用范围

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

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

商标

Microsoft®、 Windows®是美国Microsoft Corporation的美国以及其他国家中的注册商标或者商标。

Ethernet是美国Xerox Corporation的注册商标。

MODBUS®是Schneider Electric SA的注册商标。

其他的公司名称、产品名称都是各个公司的商标和注册商标。