MITSUBISHI

三菱可编程控制器

MELSEC Q 系列　MELSEC L 系列　MELSEC-F

MELSEC-Q/L/F结构体编程手册

基础篇

QSERIES
LSERIES
FSERIES

● 安全注意事项 ●

（使用之前请务必阅读）

在使用 MELSEC-Q 系列、MELSEC-L 系列、MELSEC-F 系列可编程控制器之前，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。

请妥善保管产品附带手册，放置于操作人员易于取阅的地方，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保险功能。

(2) 三菱可编程控制器是以一般工业用途等为对象设计和制造的通用产品。因此，三菱可编程控制器不应用于以下设备·系统等特殊用途。如果用于以下特殊用途，对于三菱可编程控制器的质量、性能、安全等所有相关责任（包括但不限于债务未履行责任、瑕疵担保责任、质量保证责任、违法行为责任、制造物责任），三菱将不负责。

- 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代表机构协商。

修订记录

＊ 本手册号在封底的左下角。

印刷日期	＊手册编号	修改内容
2010 年 4 月	SH(NA)-080903CHN-A	第一版

日文手册原稿： SH-080735-G

前言

在此感谢贵方购买了三菱 MELSEC-Q、MELSEC-L、MELSEC-F 系列通用可编程控制器。
在使用之前应熟读本书，在充分了解 MELSEC 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。

目录

关联手册

与本产品有关的手册如下所示。

请根据需要参考本表订购。

(1) 结构体编程

手册名称	手册编号
MELSEC-Q/L 结构体编程手册（公共指令篇） 对结构体程序中可使用的顺控指令、基本指令以及应用指令等的公共指令相关的规格、功能等有关内容进行说明。 （另售）	SH-080904CHN
MELSEC-Q/L 结构体编程手册（应用函数篇） 对结构体程序中可使用的应用函数相关的规格、功能等有关内容进行说明。 （另售）	SH-080905CHN
MELSEC-Q/L 结构体编程手册（特殊指令篇） 对结构体程序中可使用的模块专用指令、PID 控制指令以及内置 I/O 功能用用指令等的特殊指令相关的规格、功能等有关内容进行说明。 （另售）	SH-080906CHN
FXCPU 结构体编程手册（软元件 / 通用说明篇） 对 GX Works2 中提供的结构体程序用软元件、参数进行说明。 （另售）	JY997D26001
FXCPU 结构体编程手册（顺控程序指令篇） 对 GX Works2 中提供的结构体程序用顺控程序指令进行说明。 （另售）	JY997D34701
FXCPU 结构体编程手册（应用函数篇） 对 GX Works2 中提供的结构体程序用应用函数进行说明。 （另售）	JY997D34801

(2) GX Works2 的操作

手册名称	手册编号
GX Works2 Version1 操作手册（公共篇） 对 GX Works2 的系统配置及参数设置、在线功能的操作方法等、简易工程及结构体工程的通用功能等有关内容进行说明。 （另售）	SH-080932CHN
GX Works2 Version1 操作手册（结构体工程篇） 对 GX Works2 的结构体工程中的程序创建、监视等的操作方法等有关内容进行说明。 （另售）	SH-080934CHN
GX Works2 入门指南（结构体工程篇） 对初次使用 GX Works2 的用户介绍结构体工程中的程序创建及编辑、监视、调试的基本操作方法等有关内容进行说明。 （另售）	SH-080936CHN

要 点

各操作手册以 PDF 文件被存储在软件包的 CD-ROM 中。
备有用于另售的印刷品，希望单独购买手册的情况下，请通过上述表格中的手册编号购买。

1 概要

1.1 概要

在本手册中，记载了使用结构体编程方法创建顺控程序的必要程序的构成及内容、程序记述方法等基础知识。

1.2 本手册的定位

在本手册中，对结构体程序创建时必要的编程方法、程序语言的种类等有关内容进行了说明。

以目的进行分类的参阅手册如下所示。

关于各手册的记载内容、手册编号等请参阅 “关联手册”的列表。

(1) GX Works2 的操作

目的		GX Works2 安装步骤说明书	GX Works2 入门		GX Works2 Version1 操作手册			
		—	简易工程篇	结构体工程篇	公共篇	简易工程篇	结构体工程篇	智能功能模块操作篇
安装	希望了解运行环境、安装方法	详细						
简易工程的操作	希望了解基本操作及步骤		详细		概要	概要		
	希望了解编程用的功能及操作方法				概要	详细	详细 *1	
	希望了解除编程以外的所有功能及操作方法				详细			
结构体工程的操作	希望了解基本操作及步骤			详细	概要		概要	
	希望了解编程用的功能及操作方法				概要	详细	详细	
	希望了解除编程以外的所有功能及操作方法				详细			
智能功能模块的操作	希望了解智能功能模块的数据设置方法							详细

*1: 仅 ST 程序。

(2) 编程 (QCPU(Q 模式)/LCPU 时)

目的		MELSEC-Q/L/F 结构体编程手册	MELSEC-Q/L 结构体编程手册			MELSEC-Q/L 编程手册	MELSEC-Q/L/QnA 编程手册	智能功能模块用户手册 / 网络模块参考手册
								
		基础篇	公共指令篇	特殊指令篇	应用函数篇	公共指令篇	PID 控制指令篇	-
简易工程中的编程	希望了解公共指令的种类及详细内容、出错代码、特殊继电器・特殊寄存器的内容					详细		
	希望了解智能功能模块用指令的种类及详细内容							详细
	希望了解网络模块用指令的种类及详细内容							详细
	希望了解 PID 控制用指令的种类及详细内容						详细	
结构体工程的编程	希望了解初次进行结构体编程的基础知识	详细						
	希望了解公共指令的种类及详细内容		详细					
	希望了解智能功能模块用指令的种类及详细内容			详细				详细
	希望了解网络模块用指令的种类及详细内容			详细				详细
	希望了解 PID 控制用指令的种类及详细内容			详细			详细	
	希望了解出错代码、特殊继电器・特殊寄存器的内容					详细		
	希望了解应用函数的种类及详细内容				详细			

(3) 编程 (FXCPU 时)

栈担		MELSEC-Q/L/F 结构体编程手册	FXCPU 结构体编程手册			FXCPU 编程手册		
								
		基础篇	软元件 / 通用说明篇	顺控程序指令篇	应用函数篇	FX0, FX0S, FX0N, FX1, FX2, FX2C	FX1S, FX1N, FX2N, FX1NC, FX2NC	FX3G, FX3U, FX3UC
简易工程的编程	希望了解基本・应用指令的种类及详细内容、软元件及参数的内容							
结构体编程	希望了解初次进行结构体编程的基础知识							
	希望了解软元件及参数、出错代码的内容							
	希望了解顺控指令的种类及详细内容							
	希望了解函数的种类及详细内容							

1.3 本手册中使用的总称・略称

在本手册中，将软件包、可编程控制器 CPU 等以如下所示的总称・略称表示。在需要标明相关型号的情况下，将记载模块型号。

总称 / 略称	总称・略称的内容
GX Works2	产品型号 SWnDNC-GXW2 的总称产品名。 (n= 版本)
GX Developer	产品型号 SWnD5C-GPPW、SWnD5C-GPPW-A、SWnD5C-GPPW-V、SWnD5C-GPPW-VA 的总称产品名。 (n= 版本)
基本型 QCPU	Q00J、Q00、Q01 的总称。
高性能型 QCPU	Q02、Q02H、Q06H、Q12H、Q25H 的总称。
通用型 QCPU	Q00UJ、Q00U、Q01U、Q02U、Q03UD、Q03UDE、Q04UDH、Q04UDEH、Q06UDH、Q06UDEH、 Q10UDH、Q10UDEH、Q13UDH、Q13UDEH、Q20UDH、Q20UDEH、Q26UDH、Q26UDEH 的总称
QCPU(Q 模式)	基本型 QCPU、高性能型 QCPU、通用型 QCPU 的总称。
LCPU	L02CPU、L26-BTCPU 的总称。
FXCPU	MELSEC-FX 系列可编程控制器的总称。 (对象可编程控制器 CPU 为 FX0、FX0S、FX0N、FX2、FX2C、FX1S、FX1N、FX1NC、FX2N、FX2NC、FX3G、 FX3U、FX3UC)
CPU 模块	QCPU(Q 模式)、LCPU、FXCPU 的总称。
个人计算机	基于 Windows® 的个人计算机的总称。
IEC61131-3	国际标准规格 IEC61131-3 规格的略称。
公共指令	顺控程序指令、基本指令、应用指令、数据链接指令、多 CPU 专用指令、多 CPU 高速通信专用指令的总称。
特殊指令	模块专用指令、PID 控制指令、Socket(套接字) 通信功能用指令、内置 I/O 功能用指令、数据记录功能用指令的总称。

1.4 结构体程序的特点

结构体程序的特点如下所示。

(1) 结构体设计

结构体设计是指，将通过可编程控制器 CPU 进行控制的内容分为较小的处理单位（部件）以构成分级结构后进行编程的方法。在结构体程序中，可以对顺控程序进行结构体设计。

将程序分级的优点如下所示。

- 可以首先对程序的总体概要进行研究审核，然后逐步进行详细设计。
- 进行了分级设计的最低位的程序成为及其单纯且高独立性的程序。

将程序部件化的优点如下所示。

- 各部件的处理较为明确，因此程序总体易于理解。
- 可以将程序分割后通过多人进行创建。
- 提高了程序的再利用性及开发效率。

(2) 多种程序语言

在结构体程序中配备了多种程序语言。 用户可以根据用途选择最合适的程序语言组合使用。此外，可以对各个程序部件采用不同的语言进行编程。

表 1.4-1 结构体程序中可使用的程序语

名称	说明
ST（结构化文本）	类似于 C 语言等面向计算机技术人员的文本语言。
结构体梯形图	通过触点及线圈等表示梯形图的图形语言。

关于程序语言的概要请参阅以下内容。

☞ 4.2.6 项 程序部件的程序语言

关于各程序语言的记述方法请参阅以下内容。

☞ 第 5 章 程序记述方法

可以使用以前的 GX Developer 或 GX Works2 的简易工程中使用的梯形图 /SFC 语言。

关于记述方法请参阅以下内容。

☞ 各 CPU 对应的编程手册

(3) 提高了程序的再利用性

部件化后的程序可以保存到库中。通过将程序进行库化可以实现程序资源的共享化，提高程序的再利用性。

1.5 对应的 CPU 模块

结构体工程的程序对应于下述 CPU 模块。

表 1.5-1 对应的 CPU 模块

可编程控制器类型	
基本型 QCPU	Q00J, Q00, Q01
高性能型 QCPU	Q02, Q02H, Q06H, Q12H, Q25H
通用型 QCPU	Q00UJ, Q00U, Q01U, Q02U, Q03UD, Q03UDE, Q04UDH, Q04UDEH, Q06UDH, Q06UDEH, Q10UDH, Q10UDEH, Q13UDH, Q13UDEH, Q20UDH, Q20UDEH, Q26UDH, Q26UDEH
LCPU	L02, L26-BT
FXCPU	FX0, FX0S, FX0N, FX2, FX2C, FX1S, FX1N, FX1NC, FX2N, FX2NC, FX3G, FX3U, FX3UC

1.6 对应的软件版本

结构体工程的程序的创建・编辑・监视是通过下述编程工具进行的。

表 1.6-1 对应软件包

软件包名称	型号
GX Works2	SW1DNC-GXW2-J

(1) 关于 GX Works2

GX Works2 是进行顺控程序的编辑、调试、可编程控制器 CPU 的监视等的软元件包。在个人计算机的 Microsoft® Windows® 操作系统环境下运行。

在各个使用的可编程控制器中，用户创建的顺控程序以 “工程”为单位被管理，工程被大致分为 “简易工程”及 “结构体工程”。

要 点

在本手册中，以 GX Works2 的结构体工程为例对编程的基本内容进行说明。

备忘录

2 顺控程序的结构体设计

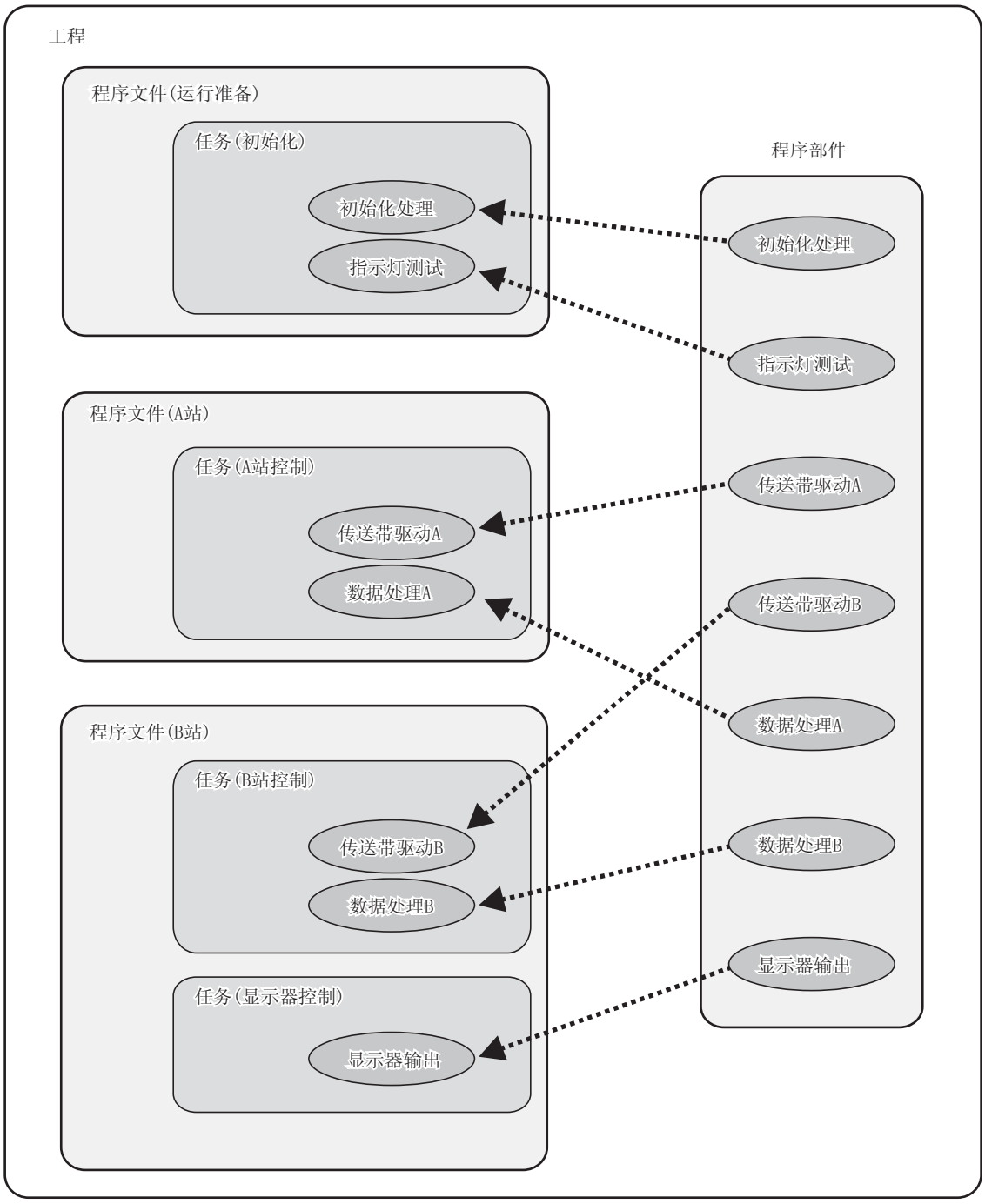
2.1 顺控程序的分级

分级是指，将通过可编程控制器进行的控制分为若干个阶层后，创建顺控程序。

在高位阶层中，对确定范围内处理的顺序及时机进行控制。

随着从高位至低位的移动，确定范围内的控制内容及处理被细分，在低位阶层中记述具体的处理。

在结构体工程中，将最高位的阶层作为工程，由程序文件、任务、程序部件所构成，对顺控程序进行分级。



2.2 顺控程序的部件化

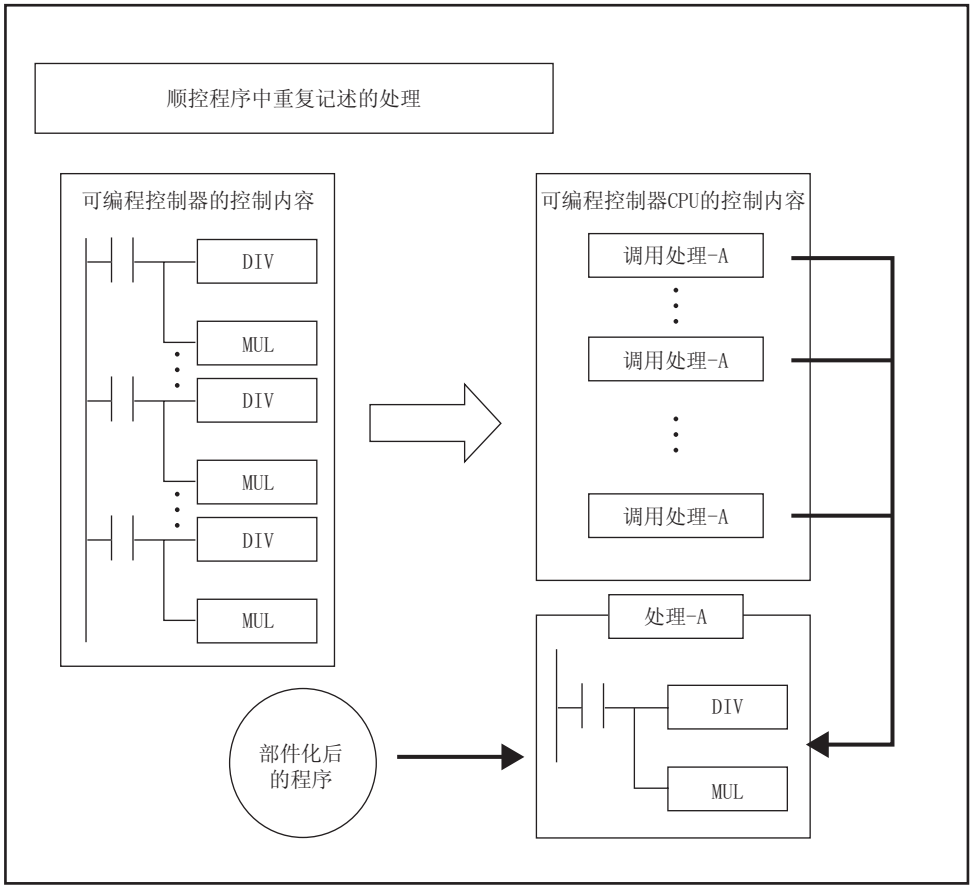
部件化是指，将顺控程序分级后的低位的处理按处理内容及功能分为若干个单位，对各单位的程序进行创建。

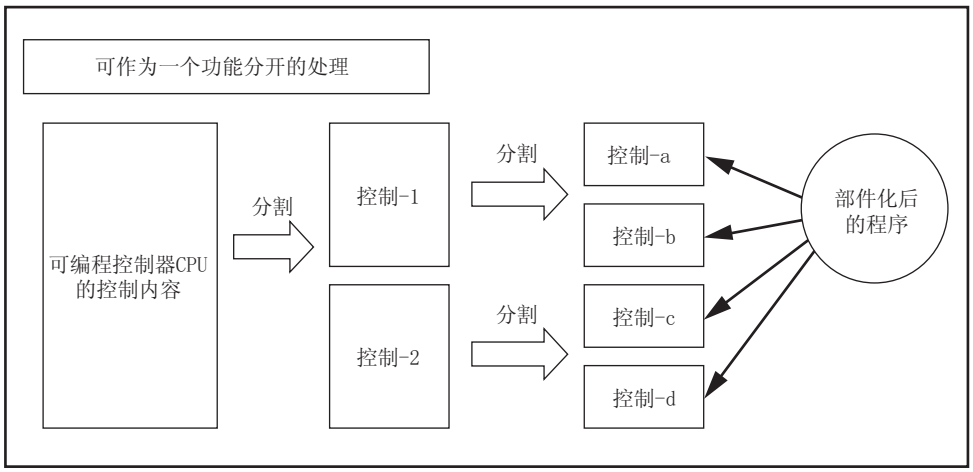
在结构体设计中，建议将低位阶层的处理尽可能地部件化。

部件应设计为独立性高、添加及更换容易。

部件化处理有以下几种。

- 顺控程序中重复记述的处理
- 可作为一个功能分开的处理





备忘录

3 编程步骤

3.1 结构体工程的顺控程序创建步骤

创建结构体工程的顺控程序时的基本步骤如下所示。

1. 程序构成的创建

步骤
创建程序文件。
创建任务。

2. 程序部件的创建

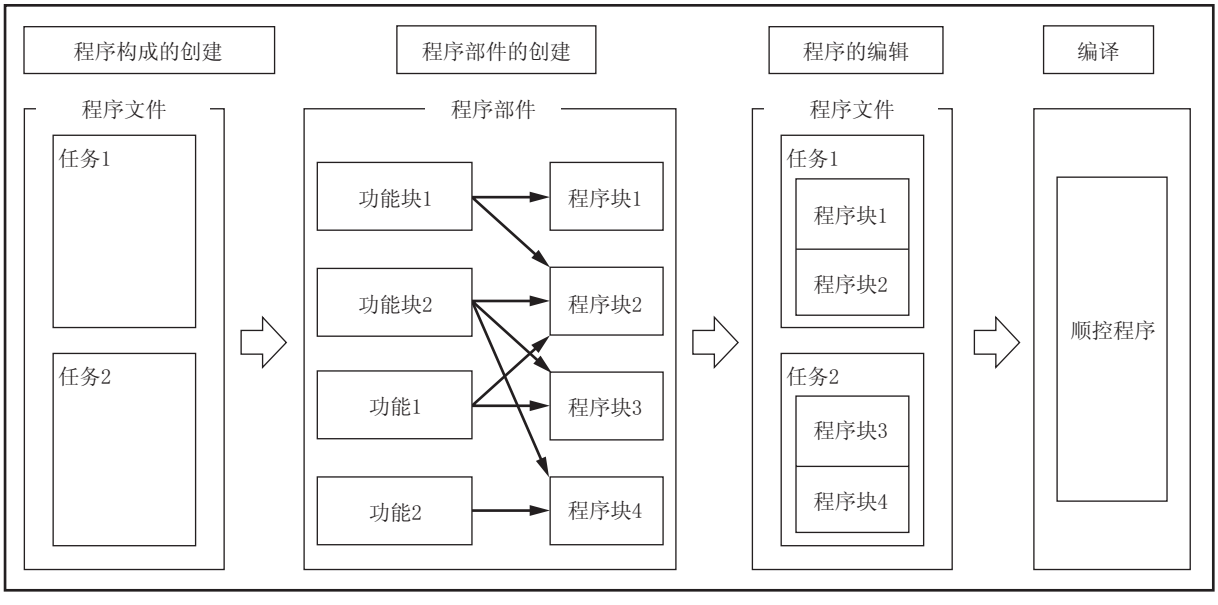
步骤
创建程序部件。
对全局标签进行定义。
对局部标签进行定义。

3. 程序的编辑

步骤
编辑各程序部件的程序。

4. 编译

步骤
将程序部件登录到任务中。
进行编译。



4 程序构成

4.1 程序构成的概要

结构体工程中创建的顺控程序是由程序文件、任务、程序部件所构成。

各详细内容请参阅以下章节。

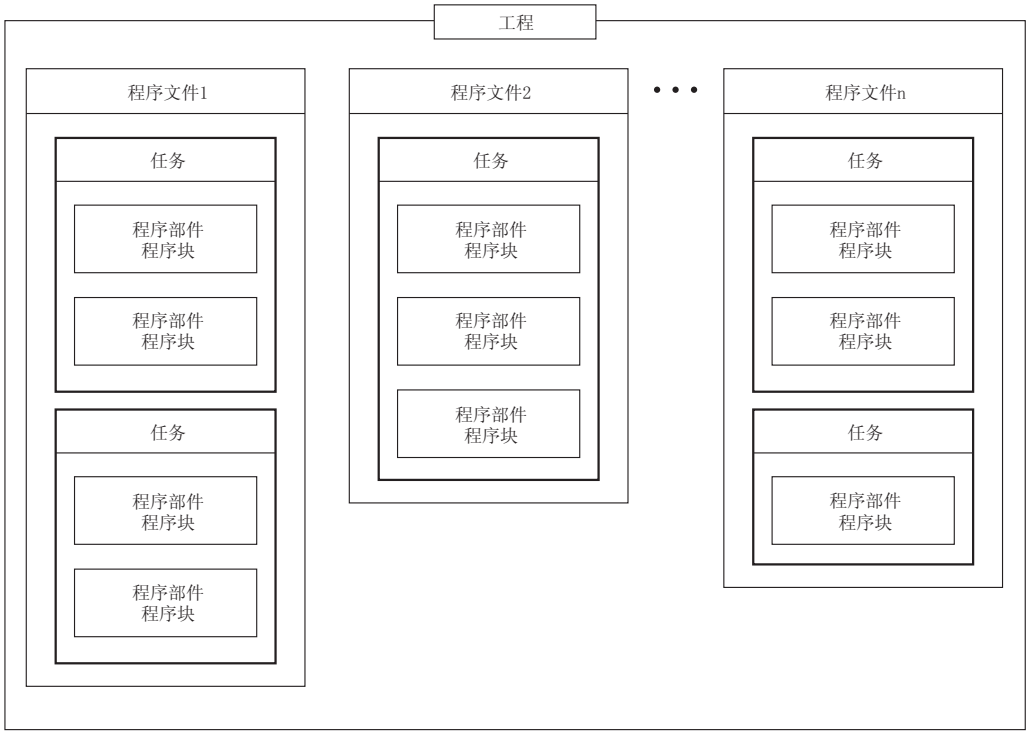
关于工程 ☞ 4.1.1 项 工程

关于程序文件 ☞ 4.1.2 项 程序文件

关于任务 ☞ 4.1.3 项 任务

关于程序部件 ☞ 4.2 节 程序部件

工程与文件、任务、程序部件的关系如下图所示。



4.1.1 工程

工程是可编程控制器 CPU 中执行的数据（程序、参数等）总称。

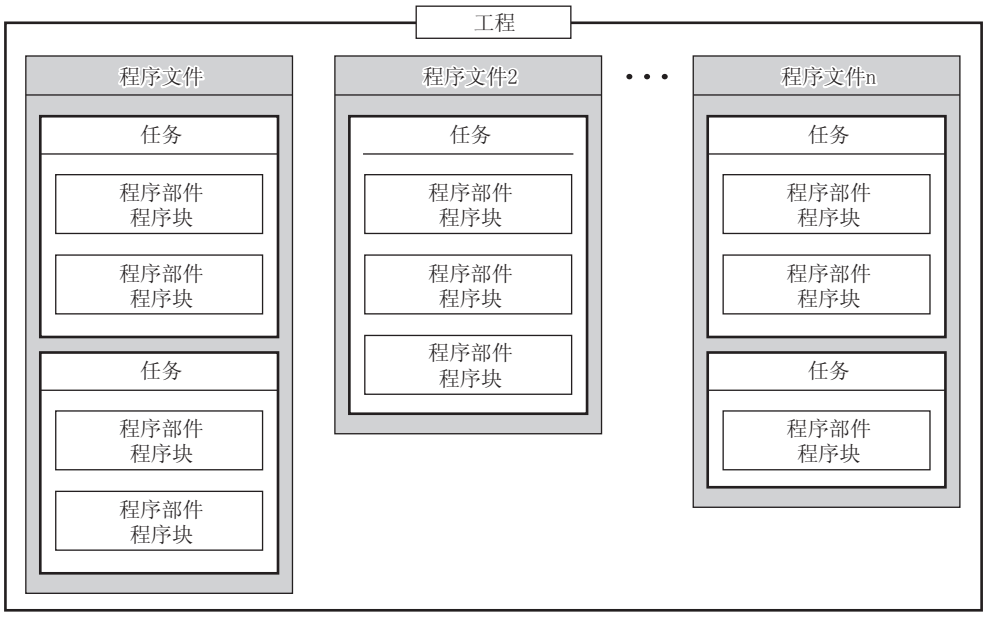
工程中需要创建一个以上的程序文件。

4.1.2 程序文件

程序文件中需要创建一个以上的任务。（创建的任务在程序文件的控制下执行。）

程序文件在可编程控制器 CPU 中执行时的执行类型（扫描执行、恒定周期执行等）是在参数的程序设置中进行设置。

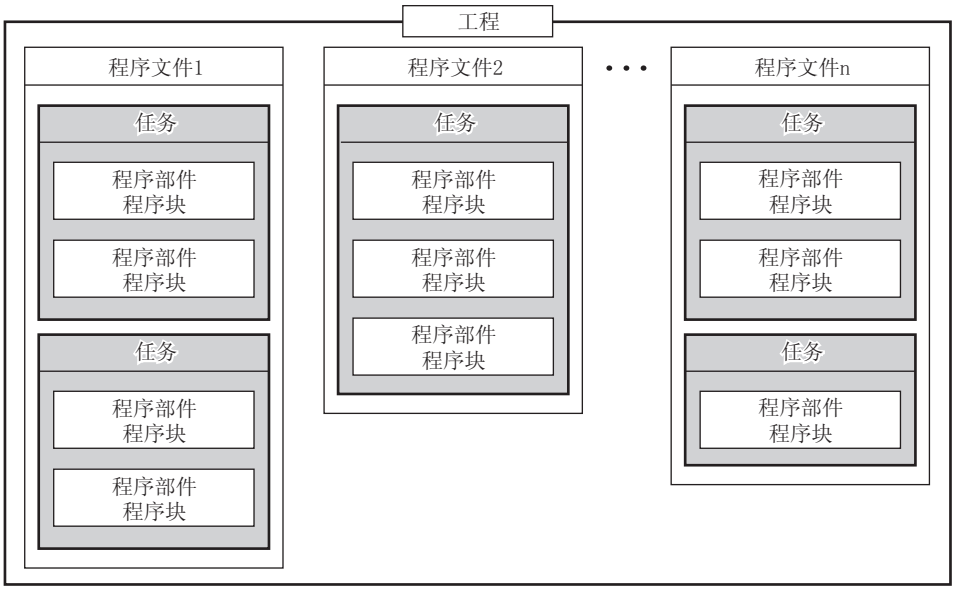
关于参数中设置的执行类型，请参阅各 CPU 模块的用户手册。



4.1.3 任务

任务是指，将多个程序部件汇集后登录到程序文件中的要素。

任务中需要登录一个以上的程序部件中的程序块。（功能及功能块不能登录到任务中。）



(1) 任务的执行条件

在程序文件中登录的各个任务中，对可编程控制器 CPU 中的执行条件进行设置。通过设置执行条件，确定各任务的执行方法。

任务的执行条件中有以下几种。

(a) 常时执行（默认的执行条件）

常时执行的任务在每次扫描中执行所登录的程序块。

(b) 事件执行

在对应的软元件或者标签中设置了值时执行。

(c) 恒定周期执行

以一定的周期执行任务。

可以对各个任务设置优先度。

- 优先度

在多个任务中，执行条件同时成立的情况下，可以设置任务的执行优先度。
从优先度值最小的任务开始按顺序执行。
优先度值相同的情况下，按任务的数据名顺序执行。

4.2 程序部件

程序部件是按各功能分类定义的程序单位。

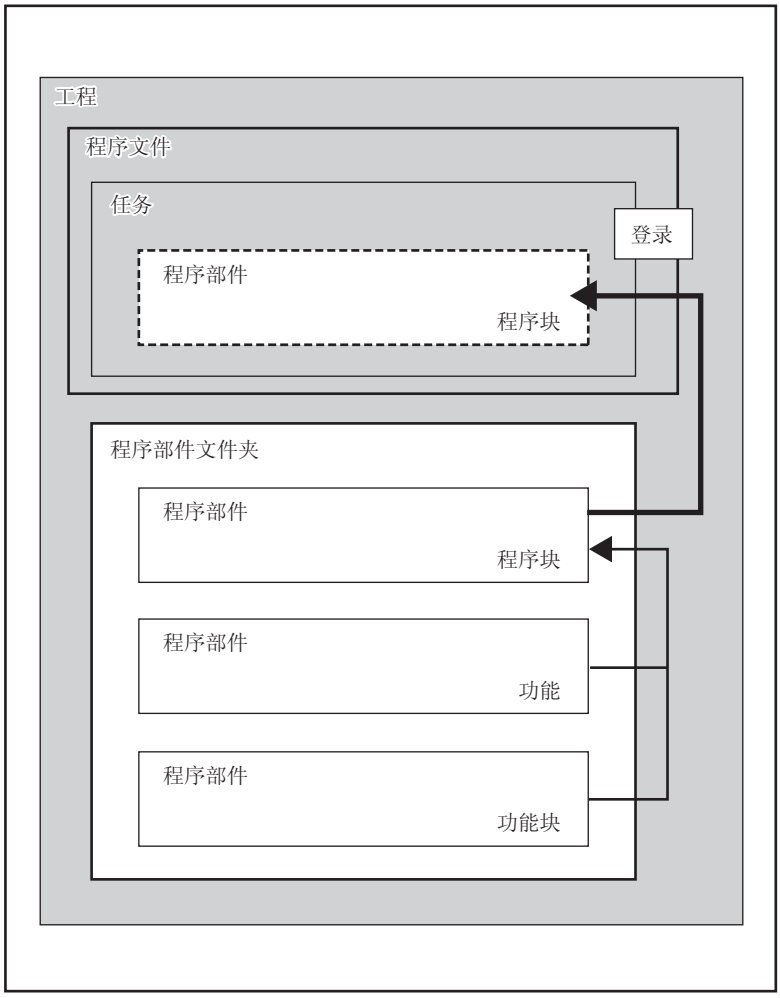
4.2.1 程序部件的种类

对于程序部件，可根据定义内容从以下 3 种类型中选择。

- 程序块
- 功能
- 功能块

各程序部件由局部标签 *1 及程序所构成。

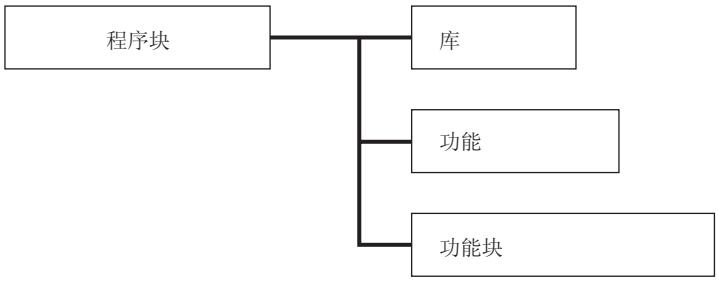
在各程序部件中，以符合控制的程序语言对处理进行记述。



*1: 局部标签是指，只能在该程序部件的程序中使用的标签。关于局部标签请参阅以下章节。
☞ 4.3.2 项 局部标签

4.2.2 程序块

程序块是程序部件中最高位的要素。使用库、功能及功能块进行编辑。



可编程控制器 CPU 中执行的顺控程序是在程序部件的程序块中创建。

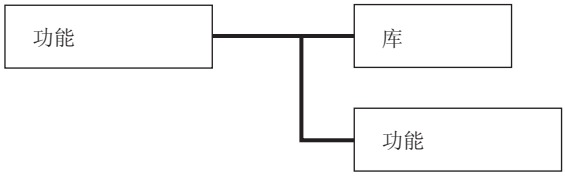
最单纯的顺控程序的情况下，通过仅创建一个程序块并登录到一个任务中后，可在可编程控制器 CPU 中执行。

程序块可以通过程序语言 ST、结构体梯形图进行记述。

4.2.3 功能

功能是使用库及功能进行编辑。

可以从程序块及功能块、功能中进行调用使用。



对于相同的输入，功能的处理结果总是输出相同的值。

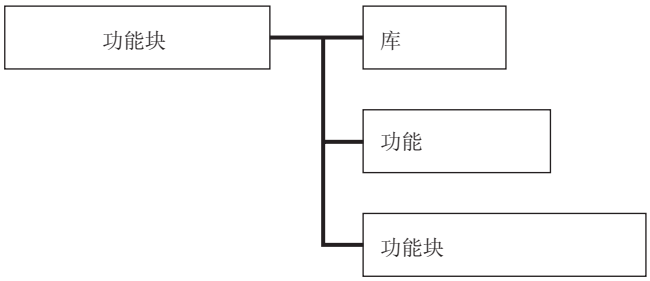
通过将单纯、独立且经常使用的计算程序进行定义，可以有效地重复利用。

功能可以通过程序语言 ST、结构体梯形图进行记述。

4.2.4 功能块

功能块是使用库、功能及其它功能块进行编辑。

对于功能块，可以从程序块及功能块中调用后使用。不能从功能中调用。



功能块可以在内部变量及输出变量中进行值的保存，因此可以对输入状态进行保持。保持的值被用于下一次的处理中，因此即使相同的输入值也不限于每次都输出相同的结果。

功能块可以通过程序语言 ST、结构体梯形图进行记述。

- 实例化
  为了在程序块中使用功能块，需要对功能块进行实例化。
  关于实例化的详细内容，请参阅 ☞ 4.2.7 项 功能与功能块的区别

要 点

实例是指，将功能块的标签中分配的软元件统称为实例的变量。
在局部标签中创建了实例的情况下，软元件将被自动分配。

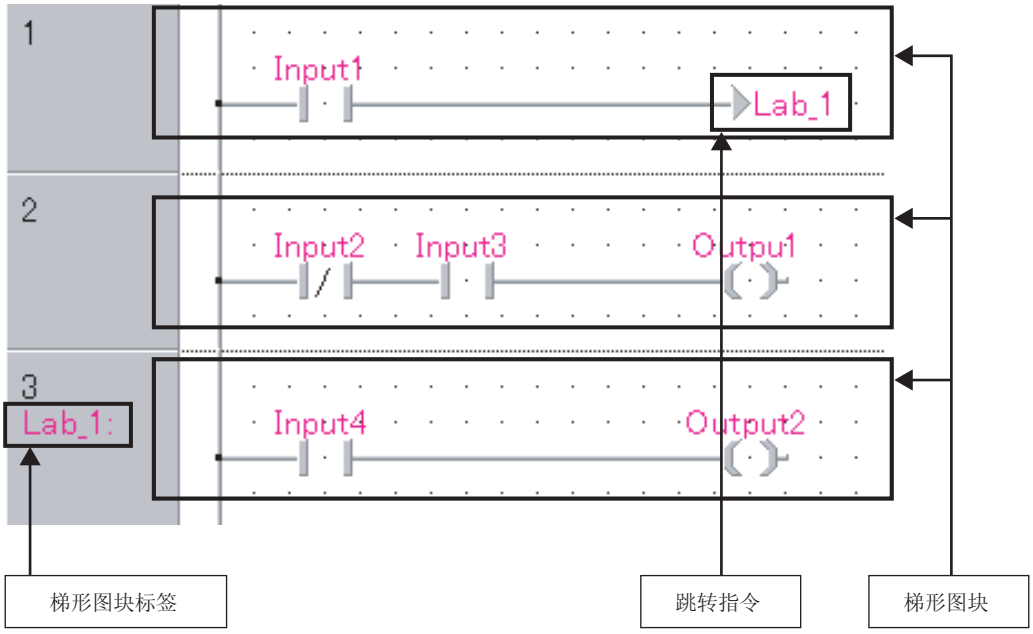
4.2.5 梯形图块

在结构体梯形图中，程序以梯形图块为单位被分割。

在 ST 中，不存在梯形图块。

- 梯形图块标签

  在梯形图块中，可以附加梯形图块标签。梯形图块标签被用作跳转指令的跳转目标。



4.2.6 程序部件的程序语言

在程序部件的程序中，可以使用 2 种程序语言。

各程序语言的特点如下所示。

1. ST：结构化文本

对于 ST 语言，可以与 C 语言等高级语言一样，通过条件语句记述选择分支，通过重复语句记述重复等对控制加以记述。由此，可以简便地记述易于看懂的程序。

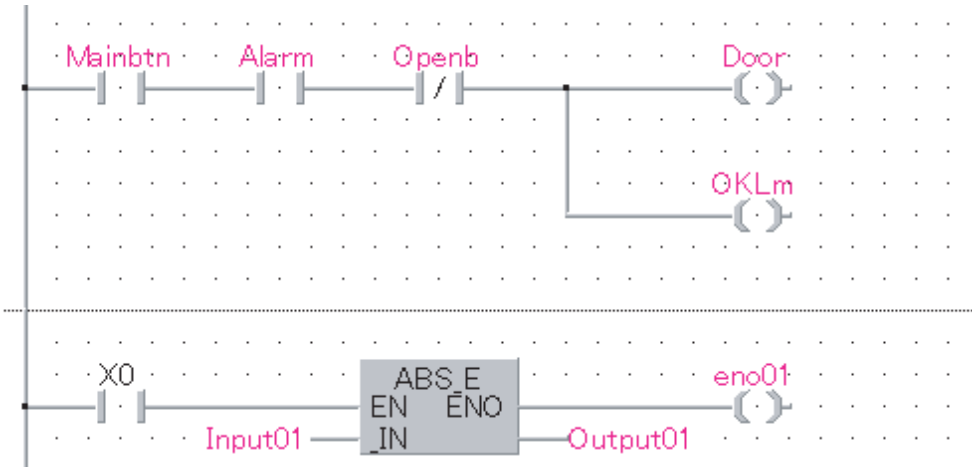
```
intV2 := ABS( intV1);

IF M1 THEN
    btn01 := TRUE;
ELSE
    btn01 := FALSE;
END_IF;

Output_ENO := ENEG(btn01, Input1);
```

(2) 结构体梯形图（电路）

是基于继电器电路的设计技术而创建的图形语言。由于较为直观且易于理解，因此常用于顺控程序中。

梯形图总是从位于左侧的母线开始。

结构体梯形图是由触点、线圈、功能块、功能所构成。这些要素通过垂直线及水平线相连接。



4.2.7 功能与功能块的区别

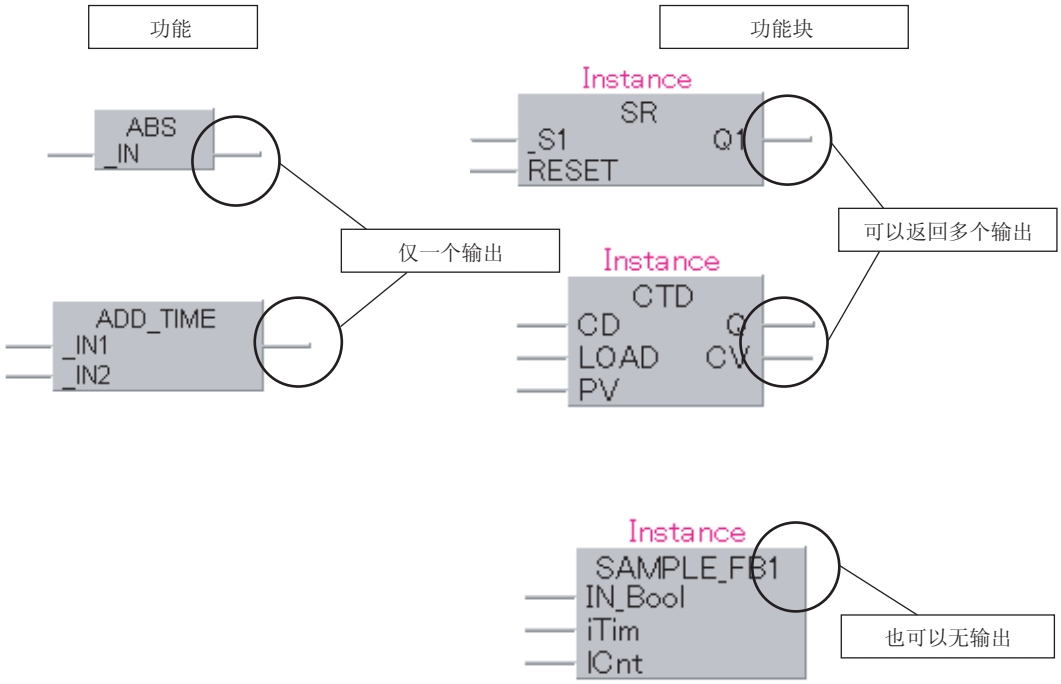
功能与功能块的区别如下所示。

表 4.2.7-1 功能与功能块

项目	功能	功能块
输出变量的分配	不能	可以
内部变量	不使用	使用
实例的创建	不需要	需要

1. 关于输出变量的分配

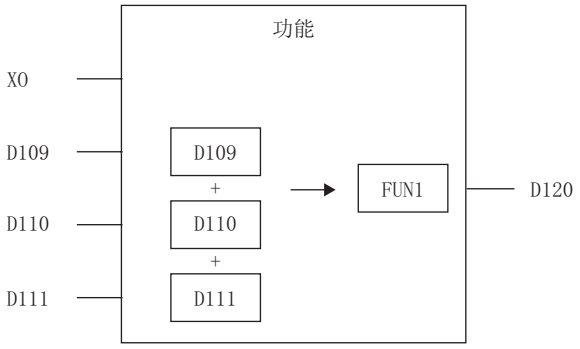
对于功能，必须输出一个运算结果。不能创建无输出或者输出多个运算结果的功能。
对于功能块，可以输出多个运算结果。此外，也可不进行输出。



(2) 关于内部变量

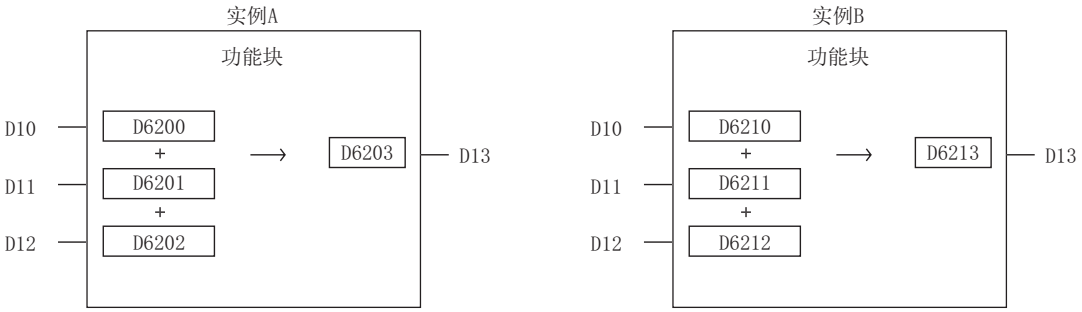
功能不使用内部变量。对于功能，对各输入变量中分配的各个软元件直接使用，重复进行运算。

(a) 输出 3 个输入变量的总和的程序（功能（功能名为 FUN1）的情况下）



功能块使用内部变量。内部变量中，对功能块的各个实例分配不同的软元件。

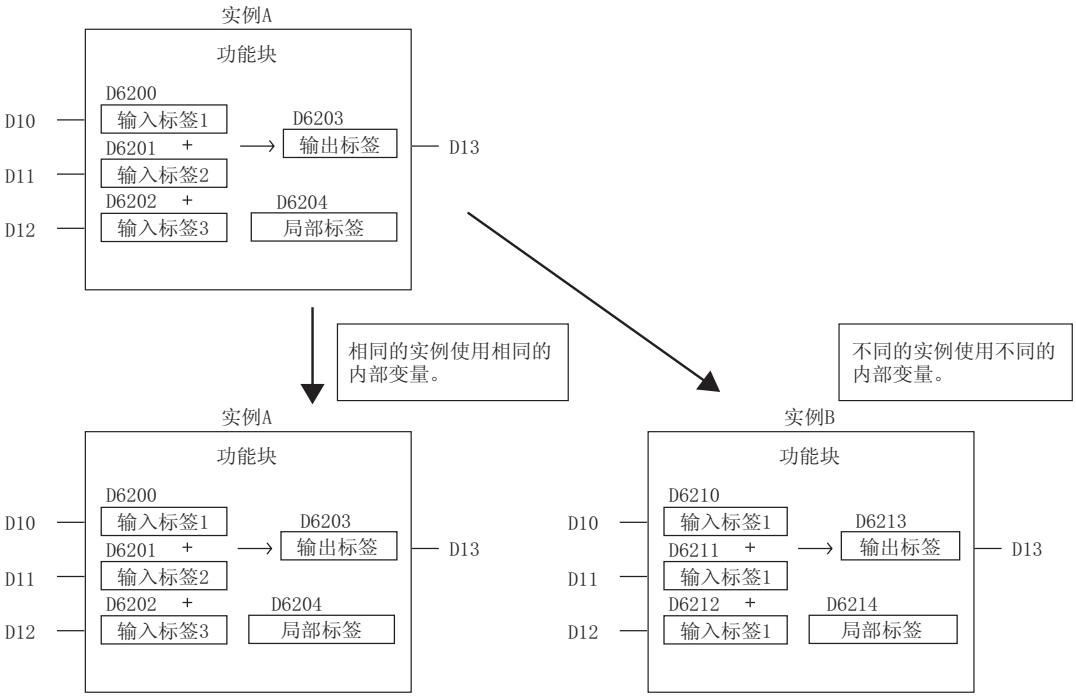
(b) 输出 3 个输入变量的总和的程序（功能块的情况下）



(3) 关于实例的创建

使用功能块时，由于需要预留内部变量，因此进行实例创建。
通过创建功能块的实例，可以从程序块及其它功能块中进行调用后使用。

进行实例化时，在全局标签或者使用功能块的程序部件的局部标签中，作为标签进行宣言。
在一个程序部件中，可以将同一个功能块以不同的名称进行多次实例化后使用。



功能块使用各实例中分配的内部变量进行运算。

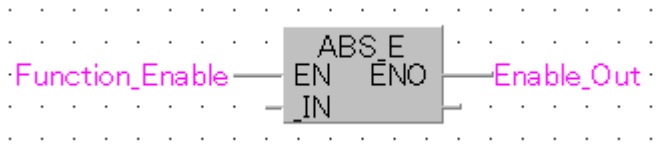
4.2.8 关于 EN/ENO

通过对功能、功能块附加 EN（使其输入）、ENO（使其输出），可以进行执行控制。

在 EN 中作为功能的执行条件对布尔变量进行设置。

对于带 EN 的功能，仅在 EN 的执行条件为 TRUE 的情况下才执行。

在 ENO 中对输出功能的执行状态的布尔变量进行设置。



根据 EN 的状态，ENO 及运算结果内容如下表所示。

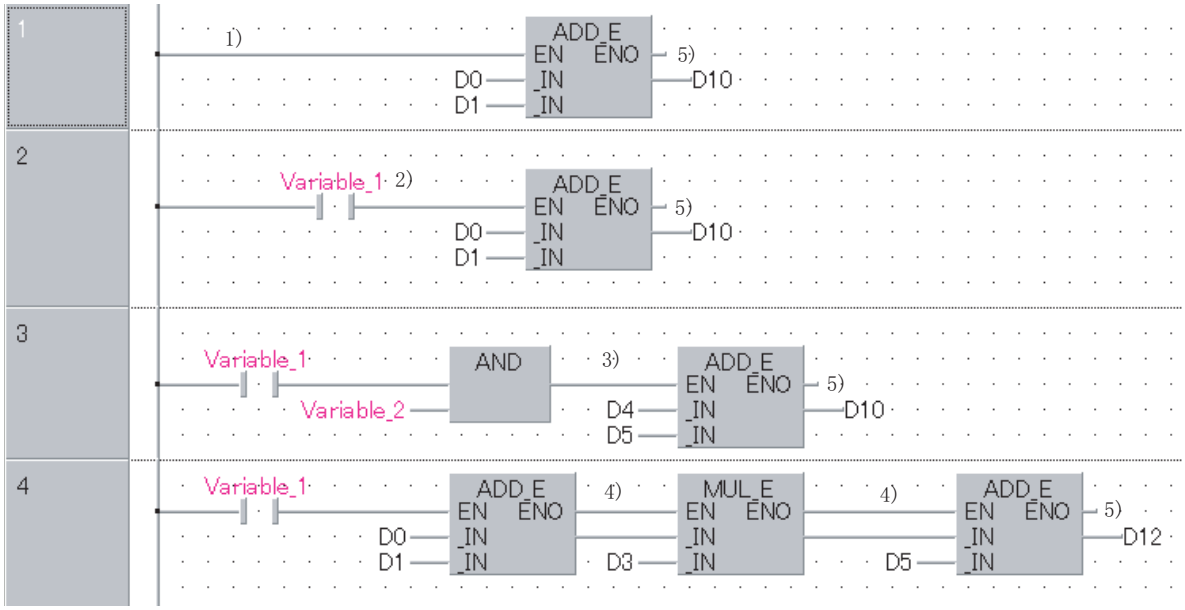
表 4.2.8-1　根据 EN 的状态的 ENO 及运算结果

EN	ENO	运算结果
TRUE（执行运算）	TRUE（无运算出错）	运算输出值
	FALSE（有运算出错）	不定值
FALSE（停止运算）	FALSE	不定值

要 点

1. 并非一定要对至 ENO 的输出标签进行设置。
2. 应用函数的情况下，带 EN 函数将变为 “函数名 _E”。

• EN、ENO 的使用示例



编号	控制内容
1)	从左侧母线开始直接连接 EN 输入的情况下，EN 输入一直为 TRUE，指令常时执行。 如果 ADD_E 指令也象这样被使用，则与无 EN 输入的 ADD 指令的结果相同。
2)	从 Variable_1 开始直接连接 EN 输入的情况下，Variable_1 为 TRUE 时执行指令。
3)	将运算的布尔结果与 EN 输入直接连接的情况下，运算结果为 TRUE 时执行指令。
4)	从 ENO 输出开始直接连接 EN 输入的情况下，Variable_1 为 TRUE 时执行 3 个指令。
5)	未连接 ENO 输出的情况下，不输出指令的执行状态。

4.3 标签

标签中包含有全局标签及局部标签。

4.3.1 全局标签

全局标签是可在程序块及功能块中使用的标签。

在全局标签的设置中，对标签名及分类、数据类型及软元件建立关联。

4.3.2 局部标签

局部标签是只能在已宣言的程序部件中使用的标签。对各程序部件分别进行定义。

在局部标签的设置中，对标签名及分类、数据类型进行设置。
在局部标签中，用户无需对软元件进行指定。在编译时将自动进行至软元件的分配。

4.3.3 标签的分类

标签的分类表示标签来自于哪个程序部件、可以如何使用。
根据程序部件的种类，可选择的分类有所不同。

标签的分类如下表所示。

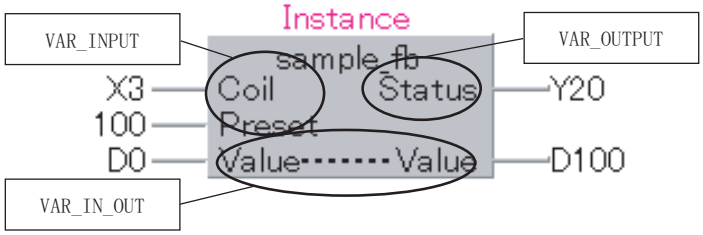
表 4.3.3-1 标签的分类

分类	内容	可使用的程序部件		
		程序块	功能	功能块
VAR_GLOBAL	在程序块及功能块中可以使用的通用标签。	○	×	○
VAR_GLOBAL_CONSTANT	在程序块及功能块中可以使用的通用常数。	○	×	○
VAR	是在已宣言的程序部件范围内使用的标签。在其它的程序部件中不能使用。	○	○	○
VAR_CONSTANT	是在已宣言的程序部件范围内使用的常数。在其它的程序部件中不能使用。	○	○	○
VAR_RETAIN*1	是在已宣言的程序部件范围内使用的锁存型标签。在其它的程序部件中不能使用。	○	×	○
VAR_INPUT	是接收值的标签，在程序部件内不能进行变更。	×	○	○
VAR_OUTPUT	是从功能块输出的标签。	×	×	○
VAR_IN_OUT	是接收值、从程序部件输出的局部标签。在程序部件内不能进行变更。	×	×	○

*1: FXCPU 不支持。

要 点

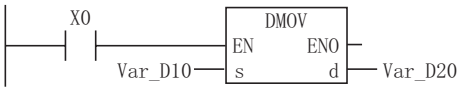
- 关于输入变量、输出变量、输入输出变量
  VAR_INPUT 变为功能及功能块的输入。
  VAR_OUTPUT 变为功能块的输出。
  VAR_IN_OUT 承担着输入及输出的任务。



4.3.4 标签的设置

对于在程序中使用的标签，需要进行全局标签或者局部标签的设置。

DMOV 指令的自变量 Var_D10、Var_D20 的设置示例如下所示。



- 将 DMOV 指令的自变量作为全局标签使用的情况下
  对分类、标签名、数据类型、软元件 / 地址进行设置。

	Class	Label Name	Data Type		Constant	Device	Address	Comment	Remark
1	VAR_GLOBAL	g_bool1	Bit	...		M0	%MX0.0		
2	VAR_GLOBAL	g_int1	Word[Signed]	...		D0	%MW0.0		
3				...					

- 将 DMOV 指令的自变量作为局部标签使用的情况下
  对分类、标签名、数据类型进行设置。

	Class	Label Name	Data Type		Constant	Device	Address	Comment
1	VAR	g_bool1	Bit	...				
2	VAR	g_int1	Word[Signed]	...				
3				...				

4.3.5 数据类型

标签根据位长、处理方法、值的范围等被按数据类型进行分类。

1. 基本数据类型

基本数据类型中有具有下述性质的数据类型。*1

- 布尔型（位）：表示 ON 或 OFF 等的二者选一状态的类型。
- 位串型（字［无符号］/ 位串［16 位］、双字［无符号］/ 位串［32 位］）：表示位的数组的类型。
- 整数型（字［带符号］、双字［带符号］）：处理正及负的整数值的类型。
- 实数型（单精度、双精度）：处理小数点以下数值的类型。
- 字符串型（字符串）：处理字符串（字符）的类型。
- 时间型（时间）：处理日时分秒（毫秒）数值的类型。

表 4.3.5-1　基本数据类型

基本数据类型	内容	值的范围	位长
位	布尔	0(FALSE),1(TRUE)	1 位
字［带符号］	整数	-32768 ～ 32767	16 位
双字［带符号］	双精度整数	-2147483648 ～ 2147483647	32 位
字［无符号］/ 位串［16 位］	16 位串	0 ～ 65535	16 位
双字［无符号］/ 位串［32 位］	32 位串	0 ～ 4294967295	32 位
单精度实数	实数	-2^{128} ～ -2^{-126}, 0, 2^{-126} ～ 2^{128}	32 位
双精度实数*2	双精度实数	-2^{1024} ～ -2^{-1022}, 0, 2^{-1022} ～ 2^{1024}	64 位
字符串	字符串	最多 255 字符	可变
时间*3	时间值	T#-24d20h31m23s648ms ～ T#24d20h31m23s647ms	32 位

*1:　下述数据类型在结构体梯形图 /ST 中不能使用。
只能在梯形图中使用。
- 定时器型：处理可编程控制器 CPU 的定时器软元件（T）的类型。
- 累计定时器型：处理可编程控制器 CPU 的累计定时器软元件（ST）的类型。
- 计数器型：处理可编程控制器 CPU 的计数器软元件（C）的类型。
- 指针型：处理可编程控制器 CPU 的指针软元件（P）的类型。

*2:　只能在通用型 QCPU/LCPU 中使用。

*3:　在时间型、应用函数的时间型运算指令中使用。关于应用函数请参阅以下手册。

MELSEC-Q/L 结构体编程手册（应用函数篇）

FXCPU 结构体编程手册（应用函数篇）

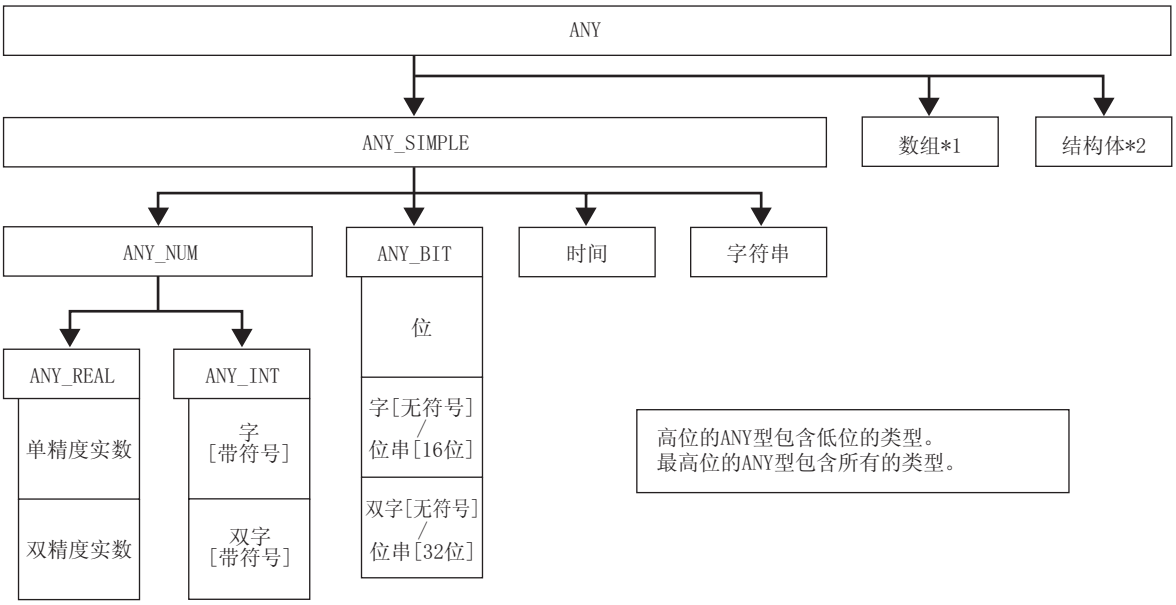
(2) 普通的数据类型

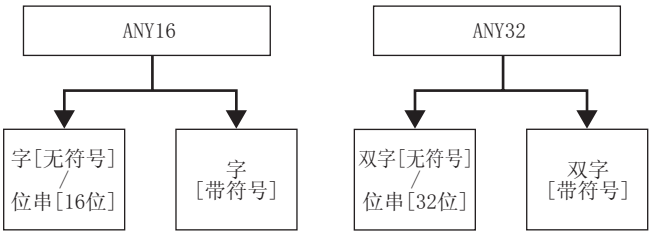
普通的数据类型是汇集了若干个基本数据类型的标签的数据类型。数据型号以 “ANY” 开始。

在允许函数的自变量、恢复值等多种数据类型的情况下，使用 ANY 型。
对于以普通的数据类型定义的标签，低位的数据类型任意类型均可使用。
例如，函数的自变量为 ANY_NUM 的情况下，自变量可指定为字 [带符号] 型、双字 [带符号] 型、单精度实数型、双精度实数型中的任意一种数据类型。

对于函数及指令的自变量，为了能够使用各种不同类型的数据，因此使用普通的数据类型进行记述。

普通的数据类型的种类及对应的基本数据类型如下所示。





*1: 关于数组请参阅 ☞ 4.4.7 项 数组

*2: 关于结构体请参阅 ☞ 4.4.8 项 结构体

4.3.6 常数的表示方法

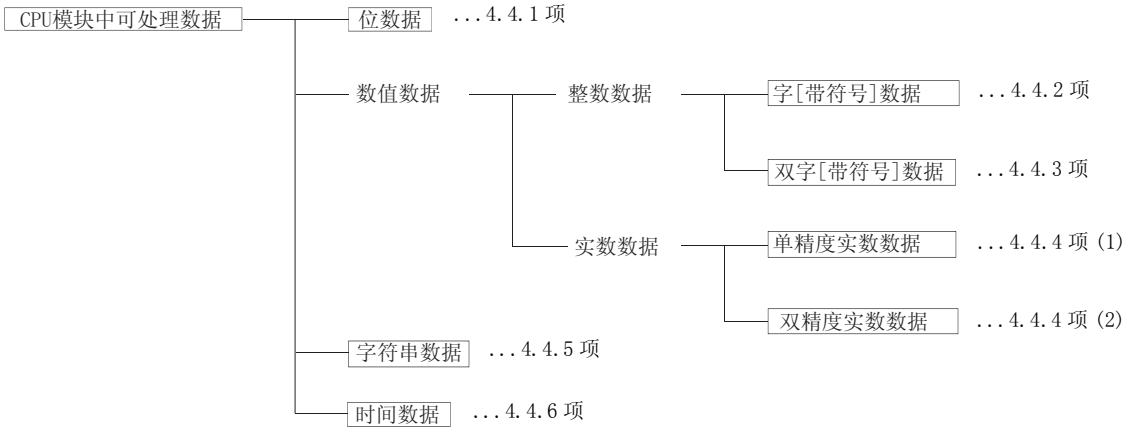
将常数设置到标签中时的表示方法如下所示。

表 4.3.6-1 常数的表示方法

常数的种类	表示方法	示例
布尔	输入 FALSE 或 TRUE。或者输入 0 或 1。	TRUE,FALSE
二进制数	在二进制数的前面附加 “2#”。	2#0010,2#01101010
八进制数	在八进制数的前面附加 “8#”。	8#0,8#337
十进制数	直接输入十进制数。或者在十进制数的前面附加 “K”。	123,K123
十六进制数	在十六进制数的前面附加 “16#”。或者附加 “H”。 “H”的情况下，字母将自动转换为大写字母。	16#FF,HFF
实数	直接输入实数。或者在实数的前面附加 “E”。	2.34,E2.34
字符串	将字符串用单引号（ ‘）或者双引号（ “）围住。	‘ABC’, “ABC”
时间	在起始处附加 “T#”。	T#1h T#1d2h3m4s5ms

4.4 数据指定方法

在 CPU 模块的指令中可使用的数据有以下 6 种。



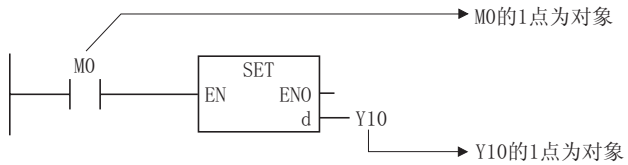
4.4.1 位数据

位数据是以触点・线圈等 1 位单位处理的数据。

“位软元件”及“位指定后的字软元件”可以作为位数据使用。

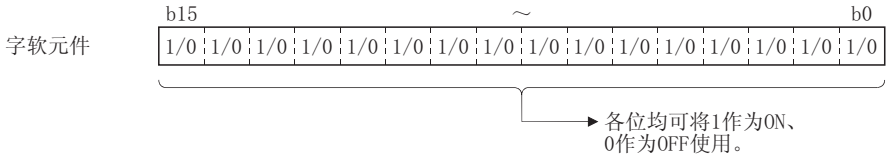
(1) 使用位软元件时

位软元件以 1 点为单位进行指定。



(2) 使用字软元件时

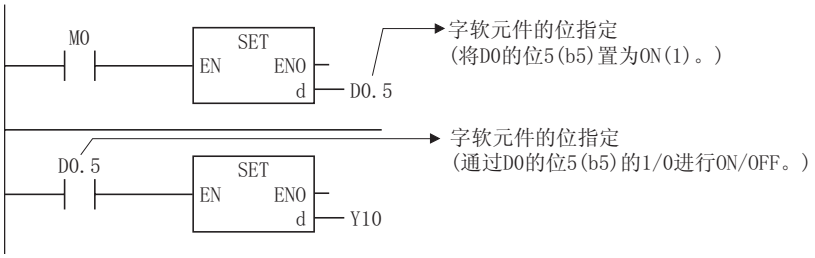
(a) 通过对字软元件进行位号指定，可以将指定位号的 1/0 作为位数据使用。



(b) 字软元件的位指定通过 “ 字软元件 . 位号 ”进行指定。(位号通过十六进制数指定。)

例如 D0 的位 5(b5) 指定为 D0.5，D0 的位 10(b10) 指定为 D0.A。

但是，定时器 (T)、累计定时器 (ST)、计数器 (C)、变址寄存器 (Z) 不能进行位指定。(例 : 不能指定 Z0.0)



要 点

FXCPU 的情况下，FX3U、FX3UC 中可以使用字软元件的位指定。

4.4.2 字（16 位）数据

字数据是基本指令・应用指令中使用的 16 位的数值数据。

CPU 模块中可处理的字数据有以下 2 种。

・十进制数 ..... -32768 ～ 32767
・十六进制数 .... 0000H ～ FFFFH

字数据可以用于字软元件及进行了位数指定的位软元件。

但是，直接访问输入 (DX)、直接访问输出 (DY) 不能通过位数指定进行字数据指定。（关于直接访问输入、直接访问输出的有关内容请参阅所使用的 CPU 模块的用户手册（功能解说 / 程序基础篇）。）

(1) 使用位软元件时

(a) 位软元件通过位数指定可以处理字数据。

位数据的位数指定是通过 “ 位数 | 位软元件的起始编号 ” 进行指定。
位数指定以 4 点（4 位）为单位可在 K1 ～ K4 的范围内进行指定。
（链接直接软元件的情况下，
以 “J | 网络号 | \ | 位数 | | 位软元件的起始编号 | ” 进行指定。指定网络号 2 的 X100 ～ X10F 时，变为 J2\K4X100。）
例如，将位数指定为 X0 时的对象点数如下所示。

・K1X0..... X0 ～ X3 的 4 点为对象
・K2X0..... X0 ～ X7 的 8 点为对象
・K3X0..... X0 ～ XB 的 12 点为对象
・K4X0..... X0 ～ XF 的 16 点为对象

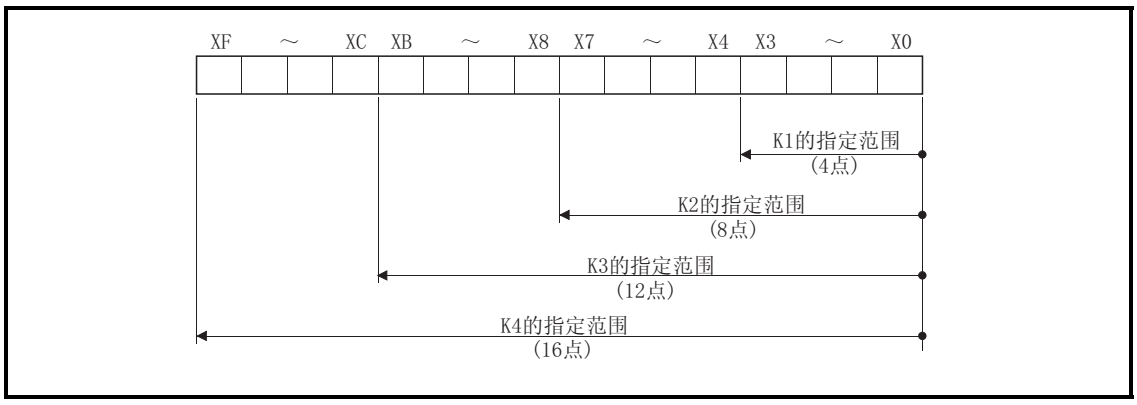


图 4.4.2-1 字数据（16 位）的位数指定设置范围

(b) 源 ⓢ 侧中有位数指定的情况下，可作为源数据使用的数值如表 4.4.2-1 所示。

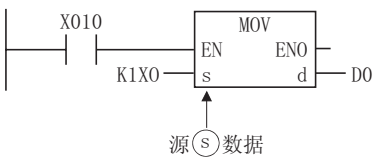
表 4.4.2-1 位数指定及可处理的数值列表

指定位数	值的范围
K1(4 点)	0 ～ 15
K2(8 点)	0 ～ 255
K3(12 点)	0 ～ 4095
K4(16 点)	-32768 ～ 32767

(c) 目标为字软元件的情况下

对于目标侧的字软元件，在源侧中进行了位数指定的位以后的位状态中将存储 0。

梯形图示例	处理
16 位数据的处理指令	



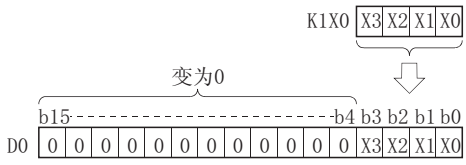
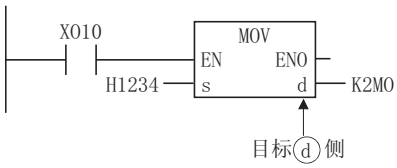


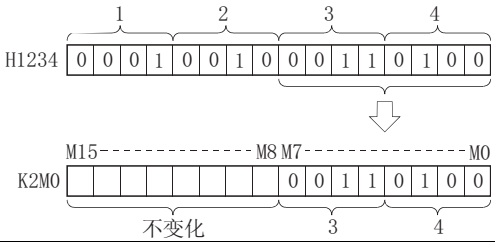
图 4.4.2-2　梯形图示例及处理内容

(d) 目标ⓓ侧中有位数指定的情况下，位数指定的点数将变为目标侧的对象。

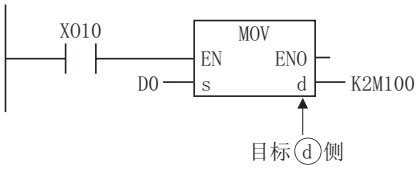
位数指定的点数以后的位软元件不变化。

梯形图示例	处理
源Ⓢ数据为数值的情况下	





源Ⓢ数据为字软元件的情况下



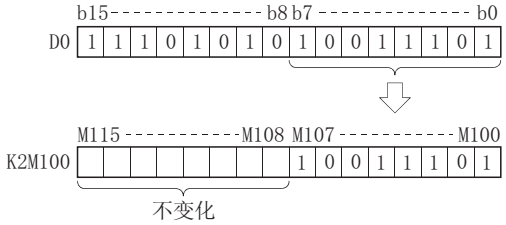
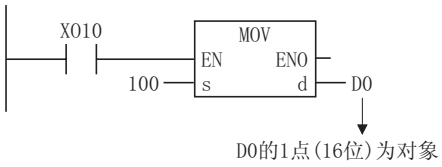


图 4.4.2-3　梯形图示例及处理内容

(2) 使用字软元件时

字软元件以 1 点（16 位）为单位进行指定。



要 点

1. 位数指定处理的情况下，位软元件的起始软元件号可以使用任意值。
2. 对于直接访问输入输出（DX、DY）不能进行位数指定。

4.4.3 双字（32 位）数据

双字数据是基本指令・应用指令中使用的 32 位的数值数据。

CPU 模块中可处理的字数据有以下 2 种。

・十进制数 ..... -2147483648 ～ 2147483647
・十六进制数 .... 00000000H ～ FFFFFFFFH

双字数据可以用于字软元件及进行了位数指定的位软元件。

但是，直接访问输入（DX）、直接访问输出（DY）不能通过位数指定进行双字数据指定。

1. 使用位软元件时

(a) 位软元件通过位数指定可以处理双字数据。

位数据的位数指定是通过“位数 位软元件的起始编号”进行指定。

（链接直接软元件的情况下，以

“J 网络号 \ 位数 位软元件的起始编号 ”进行指定。指定网络号 2 的 X100 ～ X11F 时，变为 J2\K8X100。）

位数指定以 4 点（4 位）为单位可在 K1 ～ K8 的范围内进行指定。

例如，将位数指定为 X0 时的对象点数如下所示。

・K1X0.... X0 ～ X3 的 4 点为对象
・K2X0.... X0 ～ X7 的 8 点为对象
・K3X0.... X0 ～ XB 的 12 点为对象
・K4X0.... X0 ～ XF 的 16 点为对象
・K5X0.... X0 ～ X13 的 20 点为对象
・K6X0.... X0 ～ X17 的 24 点为对象
・K7X0.... X0 ～ X1B 的 28 点为对象
・K8X0.... X0 ～ X1F 的 32 点为对象

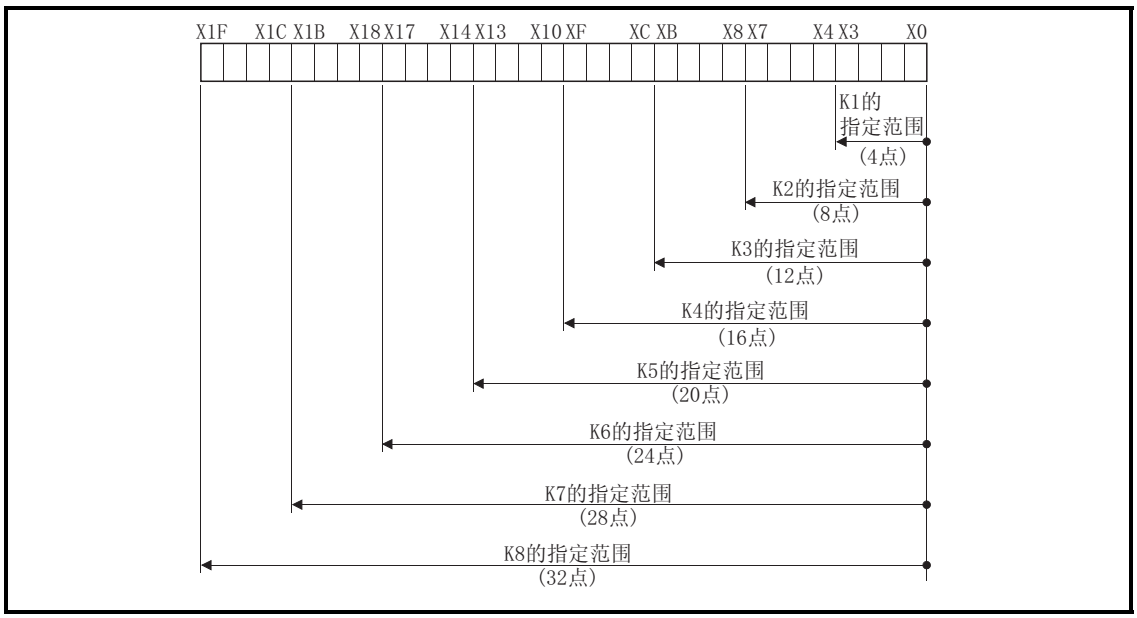


图 4.4.3-1 字数据（32 位）的位数指定设置范围

(b) 源Ⓢ侧中有位数指定的情况下，可作为源数据使用的数值如表 4.4.3-1 所示。

表 4.4.3-1 位数指定及可处理的数值列表

指定位数	值的范围	指定位数	值的范围
K1(4 点)	0 ～ 15	K5(20 点)	0 ～ 1048575
K2(8 点)	0 ～ 255	K6(24 点)	0 ～ 16777215
K3(12 点)	0 ～ 4095	K7(28 点)	0 ～ 268435455
K4(16 点)	0 ～ 65535	K8(32 点)	-2147483648 ～ 2147483647

(c) 目标为字软元件的情况下

对于目标侧的字软元件，在源侧中进行了位数指定的位以后的位状态中将存储 0。

(Data_s:K1X0，Data_d:D0 的情况下)

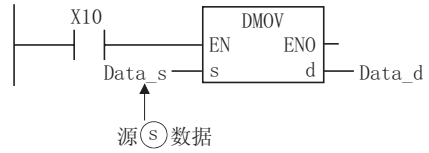
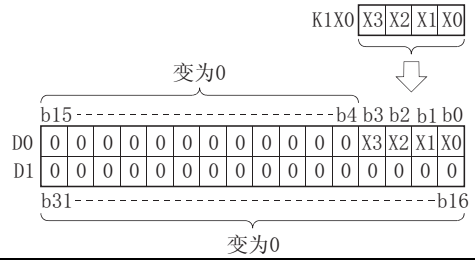
梯形图示例	处理
32 位数据的处理指令 	

图 4.4.3-2　梯形图示例及处理内容

(d) 目标ⓓ侧中有位数指定的情况下，位数指定的点数将变为目标侧的对象。

(Data_d1:K5M0, Data_d2:K5M10, Data_s:D0 的情况下) 位数指定的点数以后的位软元件不变化。

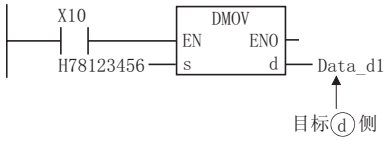
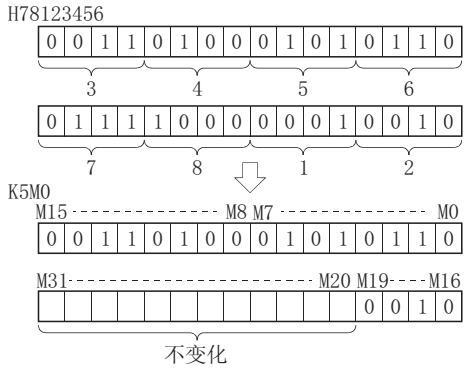
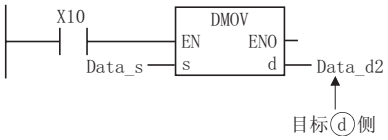
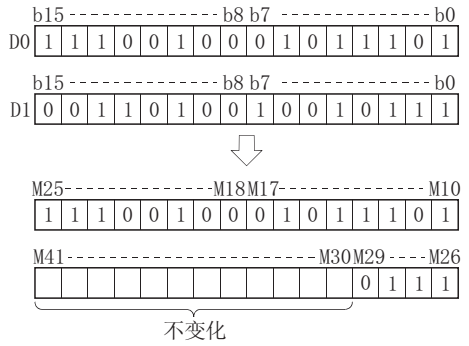
梯形图示例	处理
源Ⓢ数据为数值的情况下 	
源Ⓢ数据为字软元件的情况下 	

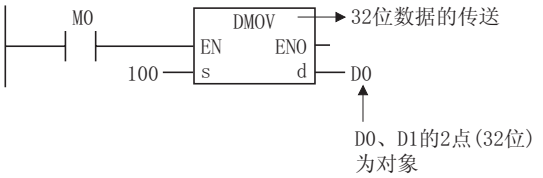
图 4.4.3-3　梯形图示例及处理内容

要 点

1. 位数指定处理的情况下，位软元件的起始软元件号可以使用任意值。
2. 对于直接访问输入输出 (DX、DY) 不能进行位数指定。

(2) 使用字软元件时

字软元件以低 16 位中使用的软元件进行指定。
在 32 位数据处理指令中，使用（指定软元件号）及（指定软元件号 +1）。

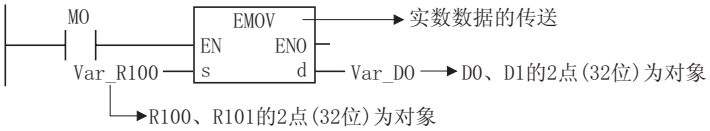


4.4.4 单精度实数 / 双精度实数数据

单精度实数 / 双精度实数数据是基本指令、应用指令中使用的 32 位的浮动小数点数据。
只有字软元件可以存储实数数据。

1. 单精度实数（单精度浮动小数点）

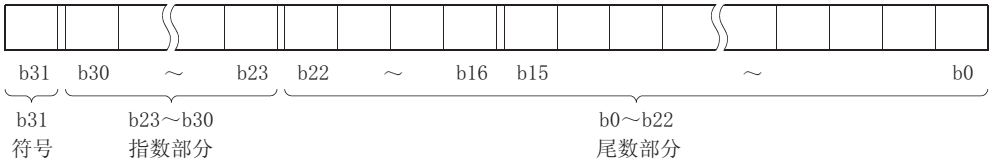
在处理实数的指令中，对低 16 位中使用的软元件进行指定。
实数被存储在（指定软元件号）及（指定软元件号 +1）的 32 位中。



备 注

浮动小数点数据使用 2 个字软元件以下述方式表示。
[符号] 1.[尾数部分]×2 [指数部分]

将浮动小数点数据内部表示时的位构成及其含义如下所示。



- 符号　在 b31 中表示符号。
  0：正
  1：负

- 指数部分　b23 ～ b30 中表示 2^n 的 n。
  根据 b23 ～ b30 的 BIN 值 n 的情况如下所示。

b23～b30	FFH	FEH	FDH			81H	80H	7FH	7EH			02H	01H	00H
n	未使用	127	126			2	1	0	-1			-125	-126	未使用

- 尾数部分　在 b0 ～ b22 的 23 位中，表示二进制数中 1.XXXXXX…时的 XXXXXX…的值。

(2) 双精度实数（双精度浮动小数点）

在处理实数的指令中，对低 16 位中使用的软元件进行指定。

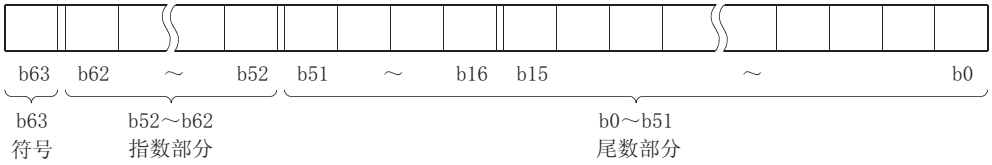
实数被存储在（指定软元件号）及（指定软元件号 +3）的 64 位中。

备 注

浮动小数点数据使用 4 个字软元件以下述方式表示。

[符号] 1.[尾数部分]×2 [指数部分]

将浮动小数点数据内部表示时的位构成及其含义如下所示。



- 符号　在 b63 中表示符号。
  0：正
  1：负

- 指数部分 在 b52 ～ b62 中表示 2^n 的 n。
  根据 b52 ～ b62 的 BIN 值 n 的情况如下所示。

b52～b62	7FFH	7FEH	7FDH			400H	3FFH	3FEH	3FDH	3FCH			02H	01H	00H
n	未使用	1023	1022			2	1	0	-1	-2			-1021	-1022	未使用

- 尾数部分　在 b0 ～ b51 的 52 位中，表示二进制数中 1.XXXXXX…时的 XXXXXX…的值。

要 点

1. 在编程工具的监视功能中，可以对 CPU 模块的浮动小数点数据进行监视。
2. 浮动小数点数据表示为 0 时，下述范围全部置为 0。
   (a) 单精度浮动小数点数据时，b0 ～ b31
   (b) 双精度浮动小数点数据时，b0 ～ b63
3. 实数的设置范围如下所示。*1
   (a) 单精度浮动小数点数据时

   $-2^{128} < \text{软元件} \leqq -2^{-126}$，0 ，$2^{-126} \leqq \text{软元件} < 2^{128}$

   (b) 双精度浮动小数点数据时

   $-2^{1023} \leqq \text{软元件} \leqq -2^{-1022}$，0 ，$-2^{-1022} \leqq \text{软元件} \leqq 2^{1023}$
4. 在浮动小数点数据中，不要指定 -0（仅浮动小数点型实数的最高位为 1 时）。
   （以 -0 进行浮动小数点运算时将变为运算出错状态。）
   对于将浮动小数点运算的内部运算以双精度进行的 CPU 模块，指定为 -0 时将在 CPU 模块内部转换为 0 后进行浮动小数点运算，因此不会变为运算出错状态。
   对于将浮动小数点运算的内部运算以单精度进行的 CPU 模块，指定为 -0 时处理速度优先，-0 直接被用于浮动小数点运算，因此将变为运算出错状态。
   (a) 指定了 -0 时不发生运算出错的 CPU 模块如下所示。
      • 将内部运算设置为双精度的高性能型 QCPU *2（浮动小数点运算的内部运算的默认状态为双精度。）
   (b) 指定了 -0 时将发生运算出错的 CPU 模块如下所示。
      • 基本型 QCPU*3
      • 将内部运算设置为单精度的高性能型 QCPU *2
      • 通用型 QCPU
      • LCPU

*1: 关于上溢及下溢时的动作、输入了特殊值时的动作，请参阅所使用的 CPU 模块的用户手册（功能解说 / 程序基础篇）。

*2: 对于浮动小数点运算的内部运算的单精度与双精度的切换，是在可编程控制器参数的可编程控制器系统设置中进行。关于浮动小数点运算的单精度及双精度的有关内容，请参阅所使用的 CPU 模块的用户手册（功能解说 / 程序基础篇）。

*3: 对于基本型 QCPU，在序列号的前 5 位数为 “04122” 以后的产品中可以进行浮动小数点运算。

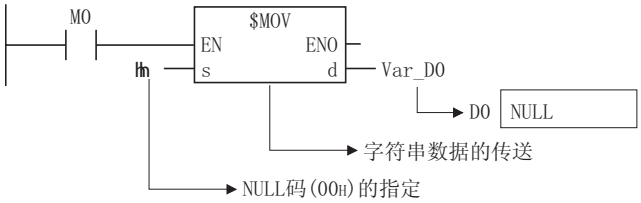
4.4.5 字符串数据

字符串数据是基本指令、应用指令中使用的字符数据。

从指定字符开始至表示字符串的最后的 NULL 码 (00H) 为止将成为对象。

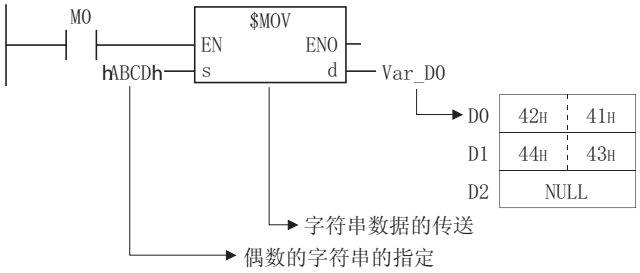
1. 指定字符为 NULL 码时

使用 1 个字存储 NULL 码。



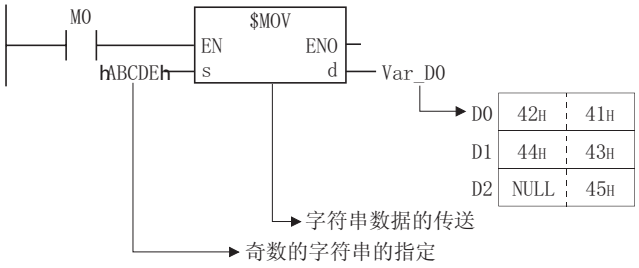
(2) 字符数为偶数时

使用 (字符数 /2+1) 个字存储字符串及 NULL 码。
例如，将 “ABCD”传送至 D0 ～时，在 D0 及 D1 中存储字符串 (ABCD)，在 D2 中存储 NULL 码。(NULL 码被存储在最后的 1 个字中。)



(3) 字符数为奇数时

使用 (字符数 /2) 个字 (小数点以下进位) 存储字符串及 NULL 码。
例如，将 “ABCDE”传送至 D0 ～时，在 D0 ～ D2 中存储字符串 (ABCDE) 及 NULL 码。(NULL 码被存储在最后的 1 个字的高 8 位中。)

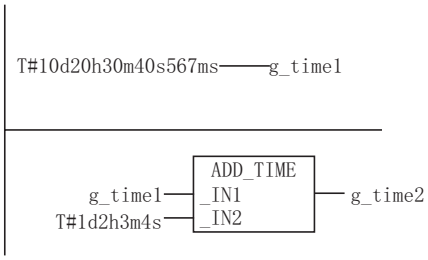


4.4.6 时间数据

时间数据是应用函数的时间型运算指令中使用的数据。

T#10d20h30m40s567ms 的格式进行指定。

例如，在“10 日 20 时 30 分 40 秒 567 毫秒”中加上“1 日 2 时 3 分 4 秒”时的情况如下所示。



时间数据的各个值可在下述范围内指定。

表 4.4.6-1 时间数据的允许指定范围

值	范围
d(日)	0 ～ 24
h(时)	0 ～ 23
m(分)	0 ～ 59
s(秒)	0 ～ 59
ms(毫秒)	0 ～ 999

关于应用函数的有关内容请参阅以下手册。

☞ MELSEC-Q/L 结构体编程手册（应用函数篇）

☞ FXCPU 结构体编程手册（应用函数篇）

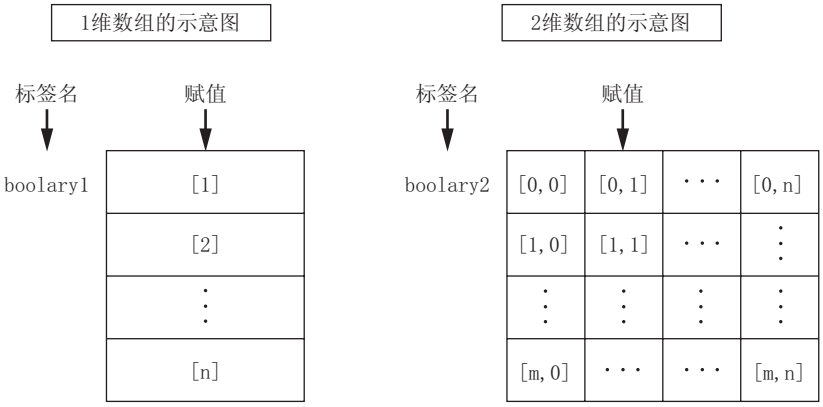
4.4.7 数组

数组是指，将相同数据类型的标签的连续集合体以一个名称表示。

数组可在基本数据类型及结构体中定义。

(☞ GX Works2 Version1 操作手册（结构体工程篇）)

根据数据类型，数组的最大数有所不同。



(1) 数组的定义

定义的格式如下表所示。

表 4.4.7-1 数组定义的格式

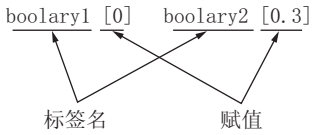
数组的维数	格式	备注
1 维	基本数据类型 / 结构体名的数组（数组开始值 .. 数组结束值）	关于基本数据类型请参阅 ☞ 4.3.5 项 关于结构体名请参阅 ☞ 4.4.8 项
	（定义示例）位 (0..2)	
2 维	基本数据类型 / 结构体名的数组（数组开始值 .. 数组结束值，数组开始值 .. 数组结束值）	
	（定义示例）位 (0..2，0..1)	
3 维	基本数据类型 / 结构体名的数组（数组开始值 .. 数组结束值，数组开始值 .. 数组结束值，数组开始值 .. 数组结束值）	
	（定义示例）位 (0..2，0..1，0..3)	

(2) 数组的表示

为了识别数组的各个标签，在标签名的后面用 “[ ]” 将赋值围住表示。

赋值可指定的范围为 -32768 ～ 32767。

例)

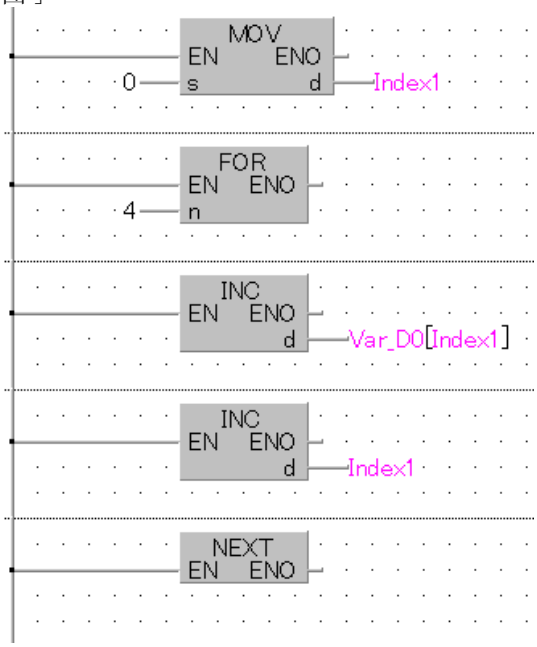


2 维以上的数组的情况下，将 “[ ]” 的赋值用 “，” 分开表示。

在程序语言 ST、结构体梯形图中，也可按如下所示在赋值中使用标签（字 [带符号] 型或者双字 [带符号] 型）。

但是，赋值中使用了标签的情况下，在程序中不要使用 Z0、Z1。

[结构体梯形图]



〔ST〕

```
FOR Index1:=0
    TO 4
    BY 1 DO
    INC(TRUE,Var_D0[Index1]);
END_FOR;
```

(3) 数组要素数的上限

数组的最大数根据数据类型而异，如下表所示。

表 4.4.7-2 数组的最大数

数据类型	最大数
位、字 [带符号]、字 [无符号]/ 位串 [16 位]、定时器、计数器、累计定时器	32768
双字 [带符号]、双字 [无符号]/ 位串 [32 位]、单精度实数、时间	16384
双精度实数	8192
字符串	32768 ÷字符串长

4.4.8 结构体

结构体是不同数据类型的标签的集合体。

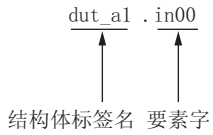
可以在所有程序部件中使用。

使用结构体时，首先创建结构体的构成，然后需要将创建的结构体作为新数据类型定义结构体标签名。

(☞ GX Works2 Version1 操作手册（结构体工程篇）)

使用构成结构体的各个要素时，在结构体标签名的后面用 “.” 分开后附加要素名。

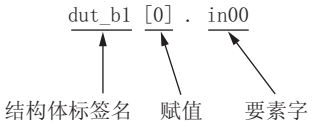
例)使用结构体的要素时

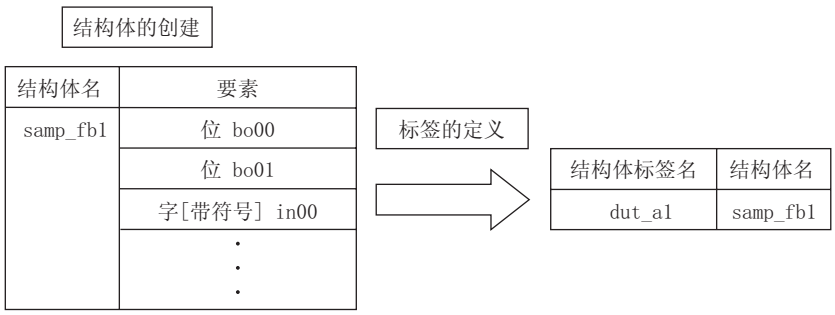


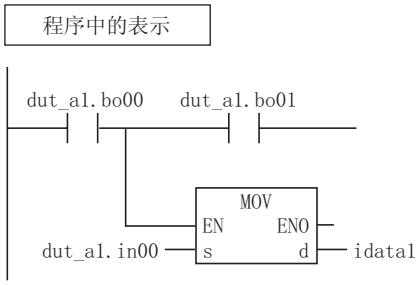
也可以将结构体排列为数组后使用。作为数组进行了宣言的情况下，结构体标签名的后面用“[ ]”将赋值围住表示。

也可将结构体的数组指定为功能或功能块的自变量。

例)使用排列为数组后的结构体的要素时







4.5 软元件及地址

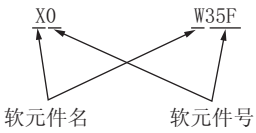
可编程控制器 CPU 的软元件的表示方法有以下 2 种。

- 软元件： 由软元件名及软元件号构成。
- 地址： 是 IEC61131-3 的表示方法。以 % 开始。

4.5.1 软元件

软元件是通过软元件名及软元件号进行表示的方法。

例)



关于软元件的详细内容请参阅以下手册。

☞ 所使用的 CPU 模块的用户手册（功能解说 / 程序基础篇）

☞ FXCPU 结构体编程手册（软元件 / 通用说明篇）

4.5.2 地址

地址是 IEC61131-3 中定义的表示方法。

IEC61131-3 的表示方法如下表所示。

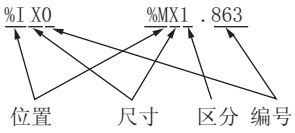
表 4.5.2-1 地址定义规格

起始	第 1 个字符：位置		第 2 个字符：尺寸		第 3 个字符以后：区分	编号
%	I	输入	（省略）	位	是用于详细分类的数字。后面的编号用“.”（点号）分开表示。有时被省略。	软元件号中的编号（以十进制数表示）。
	Q	输出	X	位		
	M	内部	W	字（16 位）		
			D	双字（32 位）		
			L	长字（64 位）*1		

*1：在 FXCP 中不能使用。

- 位置
  将数据分配位置分为输入 / 输出 / 内部这 3 大类表示。
  软元件表示的相应表示原则如下所示。
  - X，J\X(X 软元件）:I（输入）
  - Y，J\Y(Y 软元件）:Q（输出）
  - 除上述以外的软元件：M（内部）

例）



- 尺寸
  是表示数据的大小的分类。
  软元件表示的对应表示原则如下所示。
  - 位软元件：X（位）
  - 字软元件：W（字）、D（双字）、L（长字）
- 区分
  是用于表示仅通过上述位置及尺寸无法区别的软元件的类别的小分类。
  软元件的 X、X、Y 不能分开使用。
  软元件表示的对应表示请参阅以下内容。
  ☞ 4.5.3 项 软元件表示及地址表示

要 点

长字是在通用型 QCPU/LCPU 的双精度实数运算指令中使用。

4.5.3 软元件表示及地址表示

软元件及地址的表示方法及对应关系如下所示。

(1) 软元件与地址的对应

软元件与地址的表示方法的对应如下所示。

(a) QCPU(Q 模式)/LCPU

表 4.5.3-1 软元件与地址的对应 (1/2)

软元件			表示方法		软元件与地址的对应示例	
			软元件	地址	软元件	地址
输入		X	Xn	%IXn	X7FF	%IX2047
输出		Y	Yn	%QXn	Y7FF	%QX2047
内部继电器		M	Mn	%MX0.n	M2047	%MX0.2047
锁存继电器		L	Ln	%MX8.n	L2047	%MX8.2047
报警器		F	Fn	%MX7.n	F1023	%MX7.1023
特殊继电器		SM	SMn	%MX10.n	SM1023	%MX10.1023
功能输入		FX	FXn	无对应	FX10	无对应
功能输出		FY	FYn	无对应	FY10	无对应
变址继电器		V	Vn	%MX9.n	V1023	%MX9.1023
直接访问输入		DX	DXn	%IX1.n	DX7FF	%IX1.2047
直接访问输出		DY	DYn	%QX1.n	DY7FF	%QX1.2047
定时器	触点	TS	Tn	%MX3.n	TS511	%MX3.511
	线圈	TC	Tn	%MX5.n	TC511	%MX5.511
	当前值	TN	Tn	%MW3.n %MD3.n	TN511 T511	%MW3.511 %MD3.511
计数器	触点	CS	Cn	%MX4.n	CS511	%MX4.511
	线圈	CC	Cn	%MX6.n	CC511	%MX6.511
	当前值	CN	Cn	%MW4.n %MD4.n	CN511 C511	%MW4.511 %MD4.511
累计计数器	触点	STS	STn	%MX13.n	STS511	%MX13.511
	线圈	STC	STn	%MX15.n	STC511	%MX15.511
	当前值	STN	STn	%MW13.n %MD13.n	STN511 ST511	%MW13.511 %MD13.511
数据寄存器		D	Dn	%MW0.n %MD0.n	D11135	%MW0.11135 %MD0.11135
特殊寄存器		SD	SDn	%MW10.n %MD10.n	SD1023	%MW10.1023 %MD10.1023
功能寄存器		FD	FDn	无对应	FD0	无对应
链接继电器		B	Bn	%MX1.n	B7FF	%MX1.2047
链接特殊继电器		SB	SBn	%MX11.n	SB3FF	%MX11.1023
链接寄存器		W	Wn	%MW1.n %MD1.n	W7FF	%MW1.2047 %MD1.2047
链接特殊寄存器		SW	SWn	%MW11.n %MD11.n	SW3FF	%MW11.1023 %MD11.1023
智能功能模块软元件		G	Ux\Gn	%MW14.x.n %MD14.x.n	U0\G65535	%MW14.0.65535 %MD14.0.65535
文件寄存器		R	Rn	%MW2.n %MD2.n	R32767	%MW2.32767 %MD2.32767
指针		P	Pn	“”(空字符)	P299	无对应
中断指针		I	In	无对应	-	-
嵌套		N	Nn	无对应	-	-
变址寄存器		Z	Zn	%MW7.n %MD7.n	Z9	%MW7.9 %MD7.9

表 4.5.3-1　软元件与地址的对应 (2/2)

软元件		表示方法		软元件与地址的对应示例	
		软元件	地址	软元件	地址
步进继电器	S	Sn	%MX2.n	S127	%MX2.127
SFC 转移软元件	TR	TRn	%MX18.n	TR3	%MX18.3
SFC 块	BL	BLn	%MX17.n	BL3	%MX17.3
链接输入	J	Jx\Xn	%IX16.x.n	J1\X1FFF	%IX16.1.8191
链接输出		Jx\Yn	%QX16.x.n	J1\Y1FFF	%QX16.1.8191
链接继电器		Jx\Bn	%MX16.x.1.n	J2\B3FFF	%MX16.2.1.16383
链接寄存器		Jx\Wn	%MW16.x.1.n %MD16.x.1.n	J2\W3FFF	%MW16.2.1.16383 %MD16.2.1.16383
链接特殊继电器		Jx\SBn	%MX16.x.11.n	J2\SB1FF	%MX16.2.11.511
链接特殊寄存器		Jx\SWn	%MW16.x.11.n %MD16.x.11.n	J2\SW1FF	%MW16.2.11.511
文件寄存器	ZR	ZRn	%MW12.n %MD12.n	ZR32767	%MW12.32767 %MD12.32767

(b) FXCPU

表 4.5.3-2　软元件与地址的对应

软元件			表示方法		软元件与地址的对应示例	
			软元件	地址	软元件	地址
输入		X	Xn	%IXn	X367	%IX247
输出		Y	Yn	%QXn	Y367	%QX247
辅助继电器		M	Mn	%MX0.n	M499	%MX0.499
定时器	触点	TS	Tn	%MX3.n	TS191	%MX3.191
	线圈	TC	Tn	%MX5.n	TC191	%MX5.191
	当前值	TN	Tn	%MW3.n %MD3.n	TN191 T190	%MW3.191 %MD3.190
计数器	触点	CS	Cn	%MX4.n	CS99	%MX4.99
	线圈	CC	Cn	%MX6.n	CC99	%MX6.99
	当前值	CN	Cn	%MW4.n %MD4.n	CN99 C98	%MW4.99 %MD4.98
数据寄存器		D	Dn	%MW0.n %MD0.n	D199 D198	%MW0.199 %MD0.198
智能功能模块软元件		G	Ux\Gn	%MW14.x.n %MD14.x.n	U0\G09	%MW14.0.10 %MD14.0.9
扩展寄存器		R	Rn	%MW2.n %MD2.n	R32767 R32766	%MW2.32767 %MD2.32766
扩展文件寄存器		ER	ERn	无对应	-	-
指针		P	Pn	“”（空字符）	P4095	无对应
中断指针		I	In	无对应	-	-
嵌套		N	Nn	无对应	-	-
变址寄存器		Z	Zn	%MW7.n %MD7.n	Z7 Z6	%MW7.7 %MD7.6
		V	Vn	%MV6.n	V7	%MW6.7
状态		S	Sn	%MX2.n	S4095	%MX2.4095

(2) 位软元件的位数指定

进行位软元件的位数指定时，软元件与地址的对应关系如下所示。

表 4.5.3-3　位数指定中表示的对应

软元件	地址
K[位数][软元件名][软元件号] (位数 : 1 ～ 8)	%[存储器区域的位置][尺寸]19.[位数].[区分].[编号] (位数 : 1 ～ 8)

• 对应示例

软元件	地址
K1X0	%IW19.1.0
K4M100	%MW19.4.0.100
K8M100	%MD19.8.0.100
K2Y7E0	%QW19.2.2016

(3) 字软元件的位指定

进行字软元件的位指定的情况下，软元件与地址的对应关系如下所示。

表 4.5.3-4　位指定中表示的对应

软元件	地址
[软元件名][软元件号].[位号] (位号 : 0 ～ F)	%[存储器区域的位置] × [区分].[软元件号].[位号]

• 对应示例

软元件	地址
D11135.C	%MX0.11135.12
SD1023.F	%MX10.1023.15

要 点

• 关于变址修饰、位数指定、位指定
不能对标签进行变址修饰、位数指定、位指定。

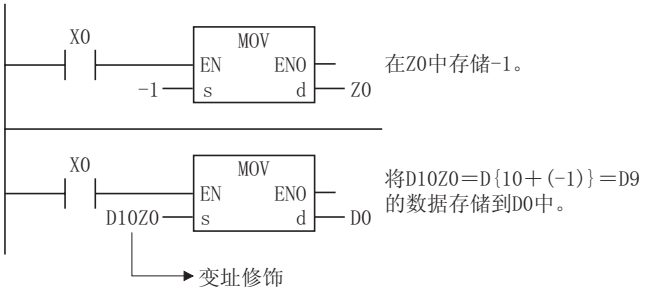
(1) 变址修饰的概要

(a) 变址修饰是使用了变址寄存器的间接设置。
如果在顺控程序中使用变址修饰，使用的软元件将变为（直接指定的软元件号）+（变址寄存器的内容）。
例如，指定了 D2Z2 时，Z2 的内容为 3 的情况下，D(2+3)=D5 将成为对象。

(b) 通用型 QCPU、LCPU、FXCPU 的情况下，除在 16 位的范围内修饰以外，也可在 32 位的范围内进行修饰。

(2) 通过 16 位进行的变址修饰

(a) 在 16 位的范围内进行修饰时
各变址寄存器的设置范围为 -32768 ～ 32767。
变址修饰的情况如下所示。



(b) 可以进行变址修饰的软元件（QCPU(Q 模式）、LCPU 时）

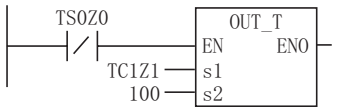
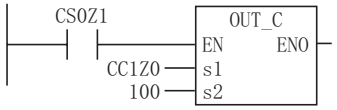
变址修饰除下述限制以外，可在触点、线圈、基本指令、应用指令中使用的软元件中使用。不能对标签进行变址修饰。

1) 禁止变址修饰的软元件

软元件	内容
E	浮动小数点数据
$	字符串数据
□.□（D0.1 等）	字软元件的位指定
FX,FY,FD	功能软元件
P	作为标签的指针
I	作为标签中断指针
Z	变址寄存器
S	步进继电器
TR	SFC 转移软元件 *1
BL	SFC 块软元件 *1

*1: SFC 转移软元件、SFC 块软元件是用于 SFC 的软元件。
关于使用方法，请参阅以下手册。
• MELSEC-Q/L/QnA 编程手册（SFC 篇）

2）使用变址寄存器有限制的软元件

软元件	内容	使用示例
T	定时器的触点、线圈中只能使用 Z0、Z1。	
C	计数器的触点、线圈中只能使用 Z0、Z1。	

(c) 允许变址修饰的软元件 (FXCPU 时)

允许变址修饰的软元件如下所示。

软元件	内容
M, S, T, C, D, R, KnM, KnS, P, K	十进制数软元件・数值
X, Y, KnX, KnY	八进制数软元件
H	十六进制数值

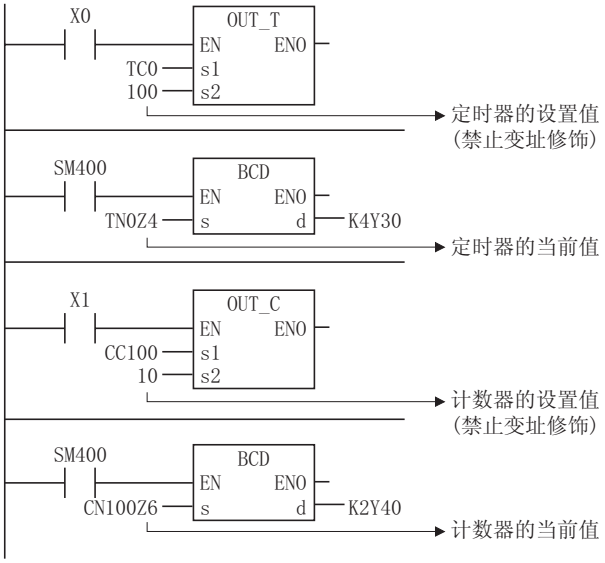
1）使用变址寄存器有限制的软元件

FXCPU 的情况下，应注意以下内容。

- 只有在 FX3U、FX3UC 中才可以对基本指令中使用的软元件进行变址修饰。
- 不能对 32 位计数器、特殊辅助继电器进行变址修饰。

备 注

定时器、计数器的当前值中，变址寄存器编号的使用无限制。



(d) 进行了变址修饰的情况与实际的处理软元件如下所示。

(Z0=20,Z1=-5 时)

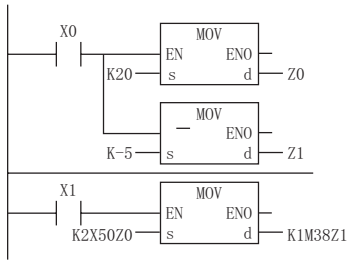
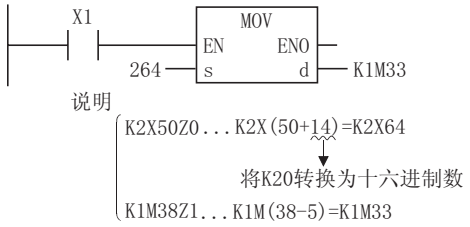
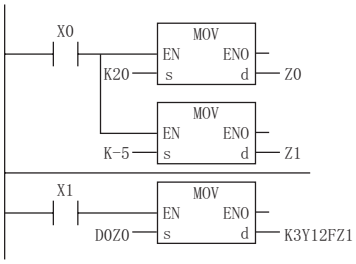
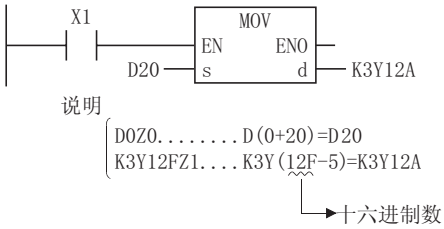
梯形图示例	实际的处理软元件
	
	

图 4.6-1　梯形图示例及实际的处理软元件

(3) 通过 32 位进行的变址修饰

(通用型 QCPU(Q00UJCPU 除外)、LCPU、FXCPU 时)

通用型 QCPU(Q00UJCPU 除外)、LCPU 的情况下，通过 32 位进行变址修饰时的变址寄存器的指定方法有以下 2 种可选。

• 对 32 位变址修饰中使用的变址寄存器的范围进行指定。

• 对 “ZZ”表示的 32 位变址修饰进行指定。

FXCPU 的情况下，32 位变址修饰是通过变址寄存器 V(V0 ～) 与 Z(Z0 ～) 的组合进行修饰。

要 点

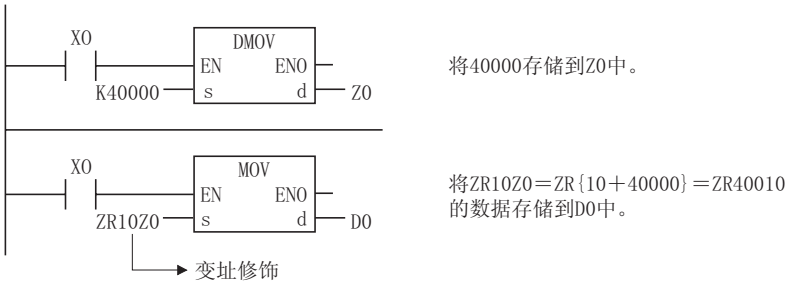
“ZZ”表示的 32 位变址修饰只能用于以下 CPU 模块。

- 序列号的前 5 位数为 “10042”以后的 QnU(D)(H)CPU(Q00UJCPU 除外)
- QnUDE(H)CPU
- LCPU

(a) 指定 32 位变址修饰中使用的变址寄存器的范围时

1) 各变址寄存器的可设置范围为 -2147483648 ～ 2147483647。

变址修饰的情况如下所示。



2) 指定方法

在 32 位的范围内进行修饰的情况下，在可编程控制器参数的 << 软元件设置 >> → “ZR 软元件的变址修饰设置”中，对所使用的变址寄存器的起始编号进行指定。

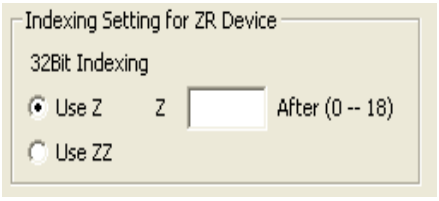


图 4.6-2　ZR 软元件的变址修饰设置参数的设置画面

要 点

在可编程控制器的软元件设置中，对所使用的变址寄存器的起始编号进行了变更的情况下，不要只对参数进行变更或者只对参数进行可编程控制器写入。必须与程序一道进行可编程控制器写入。

如果进行强制写入，将变为 CAN'T EXE. PRG.（出错代码 : 2500）状态。

3) 可进行变址修饰的软元件

变址修饰只能使用以下软元件。

软元件	内容
ZR	连号访问方式文件寄存器
D	扩展数据寄存器
W	扩展链接寄存器

4）变址寄存器的使用范围

以 32 位的范围进行修饰时的变址寄存器的使用范围如下表所示。

在 32 位范围的变址修饰中，由于使用指定的变址寄存器（Zn）及连续的下一个变址寄存器（Zn+1），因此应注意不要使变址寄存器重叠。

设置值	使用的变址寄存器	设置值	使用的变址寄存器
Z0	Z0，Z1	Z10	Z10，Z11
Z1	Z1，Z2	Z11	Z11，Z12
Z2	Z2，Z3	Z12	Z12，Z13
Z3	Z3，Z4	Z13	Z13，Z14
Z4	Z4，Z5	Z14	Z14，Z15
Z5	Z5，Z6	Z15	Z15，Z16
Z6	Z6，Z7	Z16	Z16，Z17
Z7	Z7，Z8	Z17	Z17，Z18
Z8	Z8，Z9	Z18	Z18，Z19
Z9	Z9，Z10	Z19	禁止使用

5）进行了变址修饰的情况及实际的处理软元件如下所示。

（Z0(32 位)=100000, Z2(16 位)=-20 时）

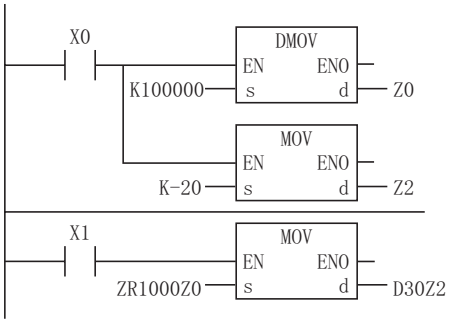
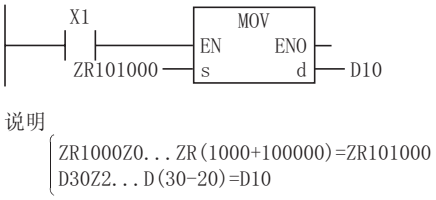
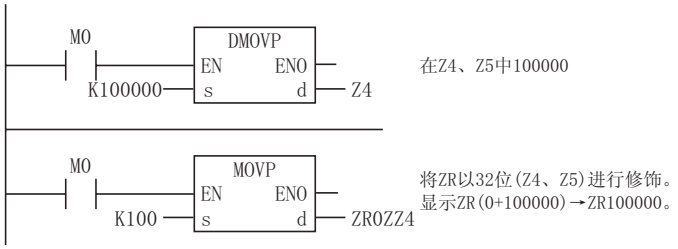
梯形图示例	实际的处理软元件
	

图 4.6-3　梯形图示例及实际的处理软元件

(a) 指定 “ZZ”表示的 32 位变址修饰时

1) 通过指定 “ZR0ZZ4”之类的 “ZZ”表示的变址修饰，可以通过任意的变址寄存器指定 32 位变址修饰。

“ZZ”表示的 32 位变址修饰的情况如下所示。



2) 指定方法

进行 “ZZ”表示的 32 位变址修饰的情况下，在可编程控制器参数的 << 软元件设置 >> → “ZR 软元件的变址修饰设置”中对 “使用 ZZ”进行设置。

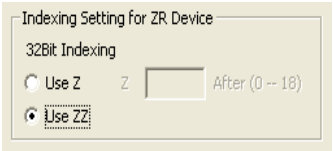


图 4.6-4　ZR 软元件的变址修饰设置参数的设置画面

3) 可进行变址修饰的软元件

变址修饰只能使用以下的软元件。

软元件	内容
ZR	连号访问方式文件寄存器
D	扩展数据寄存器
W	扩展链接寄存器

4) 变址寄存器的使用范围

“ZZ”表示的 32 位变址修饰时的变址寄存器的使用范围如下表所示。

指定 “ZZ”表示的 32 位变址修饰时，以 ZRmZZn 的格式进行指定。

通过指定 ZRmZZn，将 ZRm 的软元件号以 Zn、Zn+1 的 32 位值进行修饰。

“ZZ”表示 *2	使用的变址寄存器	“ZZ”表示 *2	使用的变址寄存器
□ZZ0	Z0，Z1	□ZZ10	Z10，Z11
□ZZ1	Z1，Z2	□ZZ11	Z11，Z12
□ZZ2	Z2，Z3	□ZZ12	Z12，Z13
□ZZ3	Z3，Z4	□ZZ13	Z13，Z14
□ZZ4	Z4，Z5	□ZZ14	Z14，Z15
□ZZ5	Z5，Z6	□ZZ15	Z15，Z16
□ZZ6	Z6，Z7	□ZZ16	Z16，Z17
□ZZ7	Z7，Z8	□ZZ17	Z17，Z18
□ZZ8	Z8，Z9	□ZZ18	Z18，Z19
□ZZ9	Z9，Z10	□ZZ19	禁止使用

*2 ：□表示修饰对象的软元件名 (ZR、D、W)。

5）指定了 “ZZ”表示的 32 位变址修饰的情况及实际的处理软元件如下所示。
(Z0(32 位)=100000, Z2(16 位)=-20 时)

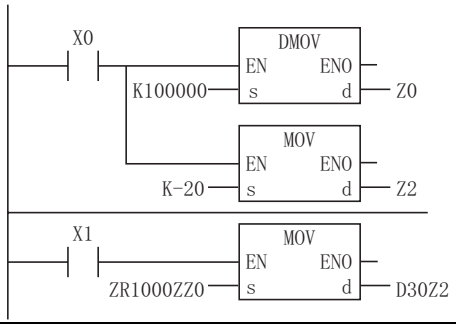
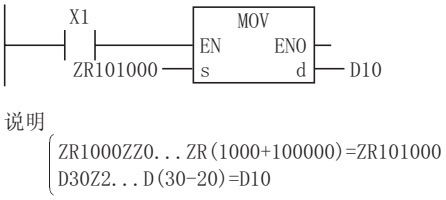
梯形图示例	实际的处理软元件
	

图 4.6-5 梯形图示例及实际的处理软元件

6）可使用 “ZZ”表示的功能

对于 “ZZ”表示的 32 位变址修饰的指定，在以下功能中可以使用。

编号	功能名称・说明
1	通过程序中的指令进行的软元件指定
2	看门狗
3	当前值变更
4	附带执行条件软元件测试
5	采样跟踪（跟踪设置（数据获取时机）、跟踪对象软元件）

要 点

不能象 “DMOV K100000 ZZ0”这样，将 ZZn 单独作为软元件处理。为了指定 “ZZ”表示的 32 位变址修饰而将值设置到变址修饰寄存器中时，应对 Zn(Z0 ～ Z19) 进行设置。

不能对各功能单独输入 ZZn。

(a) FXCPU 中的 32 位变址修饰

将变址寄存器 V(V0 ～) 与 Z(Z0 ～) 组合进行 32 位变址修饰。

V 侧变为高位、Z 侧变为低位。如果指定低位侧的 Z，则作为包含与 Z 侧组合的 V 侧在内的 32 位寄存器执行动作。

即使对高位侧的 V 进行指定也不能进行修饰。

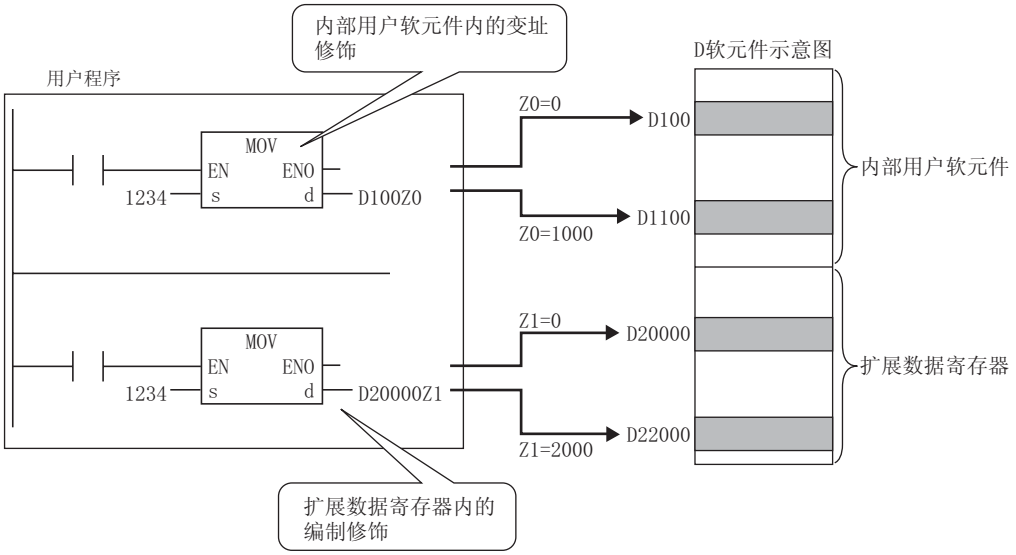
例 : 如果指定 Z4，则 V4、Z4 作为 32 位寄存器进行修饰。

设置值	使用的变址寄存器
Z0	V0，Z0
Z1	V1，Z1
Z2	V2，Z2
Z3	V3，Z3
Z4	V4，Z4
Z5	V5，Z5
Z6	V6，Z6
Z7	V7，Z7

(4) 通过扩展数据寄存器 (D)、扩展链接寄存器 (W) 进行变址修饰时

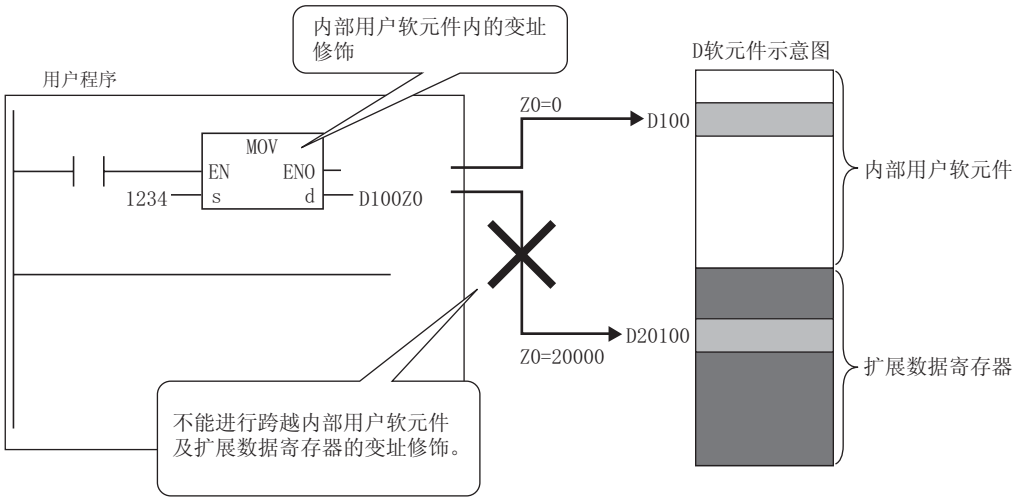
(通用型 QCPU(Q00UJCPU 除外)、LCPU)

与内部用户软元件的数据寄存器 (D)、链接寄存器 (W) 中的变址修饰一样，在扩展数据寄存器 (D)、扩展链接寄存器 (W) 的范围内可以通过变址修饰使用软元件指定。



1) 跨越内部用户软元件及扩展数据寄存器 (D)、扩展链接寄存器 (W) 的变址修饰

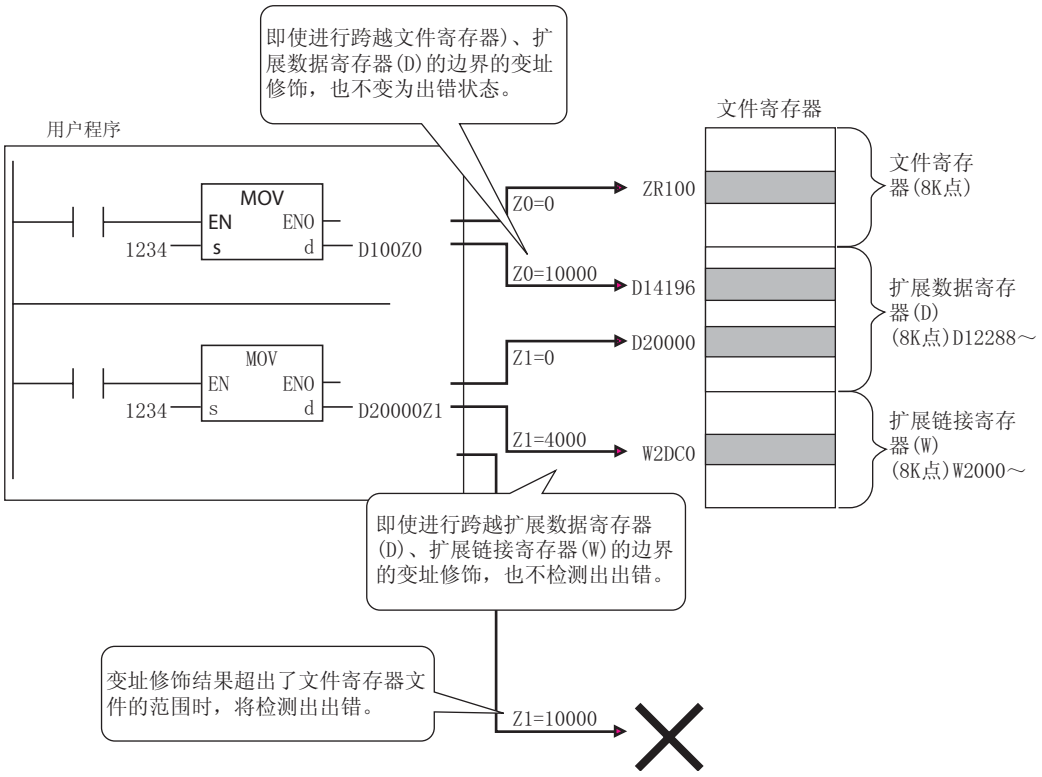
不能指定跨越内部用户软元件及扩展数据寄存器 (D)、扩展链接寄存器 (W) 的变址修饰。变址修饰时的软元件范围检查有效的情况下，将变为出错状态。(出错代码： 4101)



2) 跨越文件寄存器 (ZR)、扩展数据寄存器 (D)、扩展链接寄存器 (W) 的变址修饰

即使进行跨越文件寄存器 (ZR)、扩展数据寄存器 (D)、扩展链接寄存器 (W) 的变址修饰，也不变为出错状态。

但是，文件寄存器 (ZR)、扩展数据寄存器 (D)、扩展链接寄存器 (W) 的变址修饰结果超出了文件寄存器文件的范围时，将变为出错状态。(出错代码：4101)

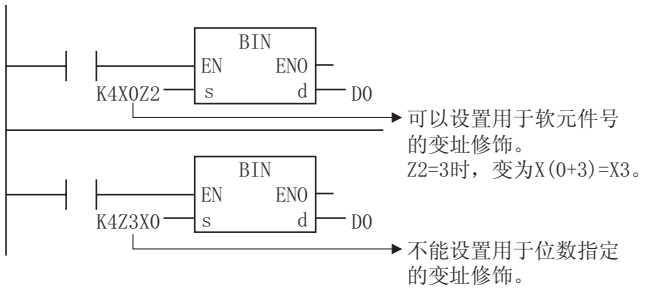


(5) 进行其它的变址修饰时

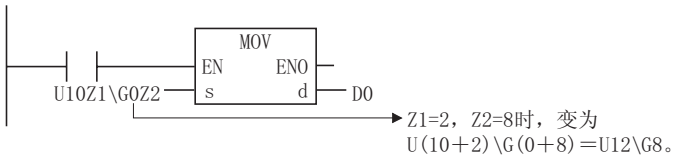
(a) 位数据

进行了位数指定的情况下，可以对软元件号进行变址修饰。

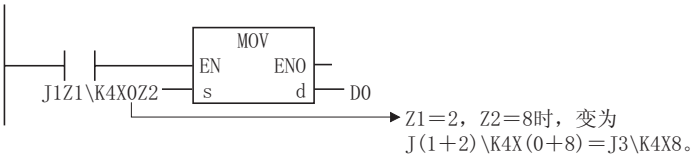
但是对于位数指定不能进行变址修饰。



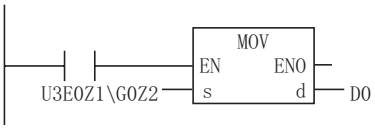
(b) 在智能功能模块软元件中 *3，对智能功能模块的起始输入输出地址号及缓冲存储器地址均可进行变址修饰。



(c) 在链接直接软元件 *3 中，对网络号及软元件号均可进行变址修饰。



(d) 在多 CPU 共享软元件 *4 中，对 CPU 模块的起始输入输出地址号及 CPU 共享存储器地址均可进行变址修饰。



*3: 关于智能功能模块软元件、链接直接软元件，请参阅所使用的 CPU 模块的用户手册（功能解说／程序基础篇）。

*4: 关于多 CPU 共享软元件，请参阅所使用的 CPU 模块的用户手册（功能解说／程序基础篇）。

(e) 进行扩展数据寄存器 (D)、扩展链接寄存器 (W) 的 32 位变址修饰时
（通用型 QCPU(Q00UJCPU 除外）、LCPU 时）
进行扩展数据寄存器 (D)、扩展链接寄存器 (W) 的变址修饰时，与文件寄存器 (ZR) 的变址修饰一样，可以通过以下 2 种方法在 32 位的范围内进行变址修饰。

- 对 32 位变址修饰中使用的变址寄存器的范围进行指定。
- 对 “ZZ”表示的 32 位变址修饰进行指定。

要 点

“ZZ”表示的 32 位变址修饰只能用于以下 CPU 模块。
- 序列号的前 5 位数为 “10042”以后的 QnU(D)(H)CPU(Q00UJCPU 除外）
- QnUDE(H)CPU
- LCPU

(6) 注意事项

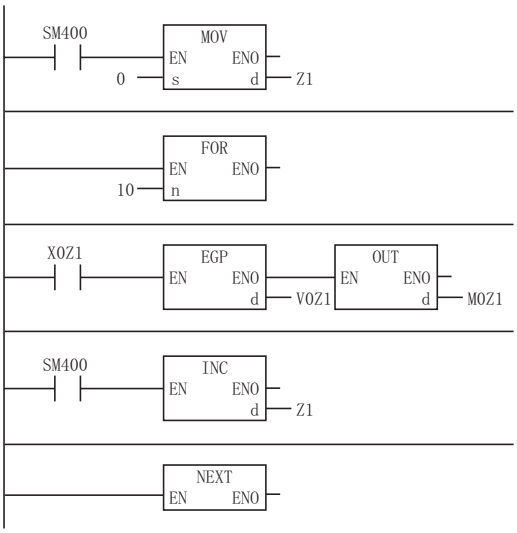
(a) 在 FOR ～ NEXT 指令之间进行变址修饰时

在 FOR ～ NEXT 指令之间通过使用变址继电器 (V)，可以进行脉冲输出。

但是，不能通过 PLS/PLF/ 脉冲化 (□ P) 指令进行脉冲输出。

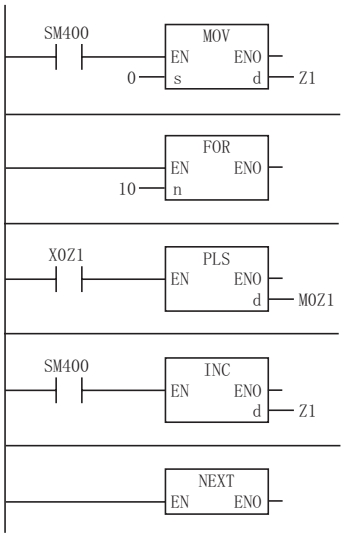
［使用变址继电器时］

(MOZ1 正常地脉冲输出。)



［未使用变址继电器时］

(MOZ1 不能正常地脉冲输出。)



备 注

X0Z1 的 ON/OFF 信息被存储在变址继电器的 V0Z1 中。

例如，X0 的 ON/OFF 信息存储在 V0 中，X1 的 ON/OFF 信息被存储在 V1 中。

要 点

在 FOR ～ NEXT 指令之间的数组的赋值中使用标签时，不要使用 Z0、Z1。

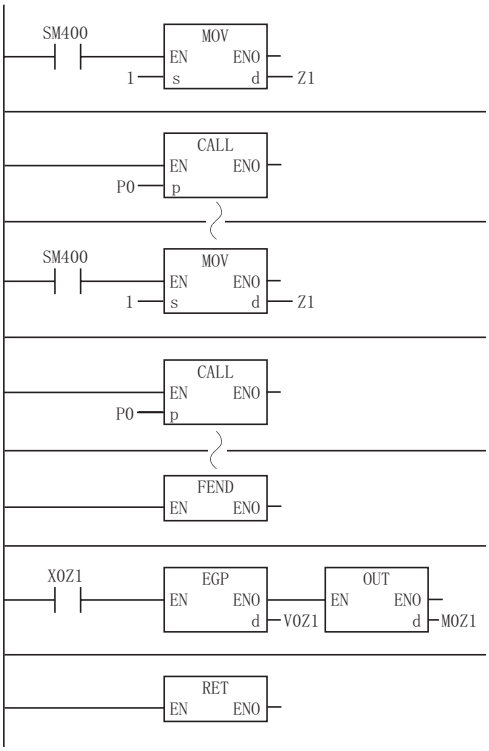
(b) 通过 CALL 指令进行变址修饰时

通过在 CALL 指令中使用变址继电器 (V)，可以进行脉冲输出。

但是，在 PLS/PLF/ 脉冲化 (□ P) 指令中不能进行脉冲输出。

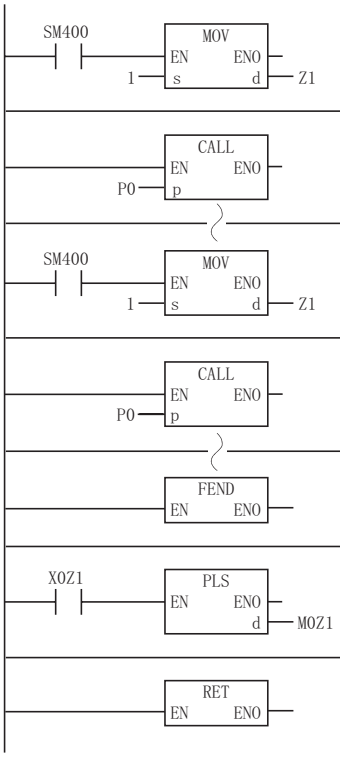
［使用变址继电器时］

(MOZ1 正常地脉冲输出。)



［未使用变址继电器时］

(MOZ1 不能正常地脉冲输出。)



(c) 变址修饰时的软元件范围检查

1) 对于基本型 QCPU、高性能型 QCPU、FXCPU

变址修饰时不进行软元件范围检查。

基本型 QCPU、高性能型 QCPU 时进行变址修饰的结果超出了用户指定的软元件范围的情况下，不变为出错状态，而是将数据写入到其它的软元件中。(但是，变址修饰的结果超出了用户指定的软元件范围后被写入到系统用的软元件中的情况下将变为出错状态。)

(出错代码 : 1103)

FXCPU 时，将变为运算出错状态。(出错代码 : 6706)

创建使用了变址修饰的程序时，应充分注意。

2) 通用型 QCPU、LCPU 时

变址修饰时将进行软元件范围检查。

此外，也可以根据可编程控制器参数的情况设置为不进行软元件范围检查。

(d) 16 位←→ 32 位变址修饰范围的变更

进行 16 位←→ 32 位变址修饰范围变更的情况下，应对程序内的变址修饰位置重新进行审核。

在 32 位范围的变址修饰中，由于使用指定的变址寄存器 (Zn) 及紧接着的下一个变址寄存器 (Zn+1)，因此应注意避免变址寄存器重叠。

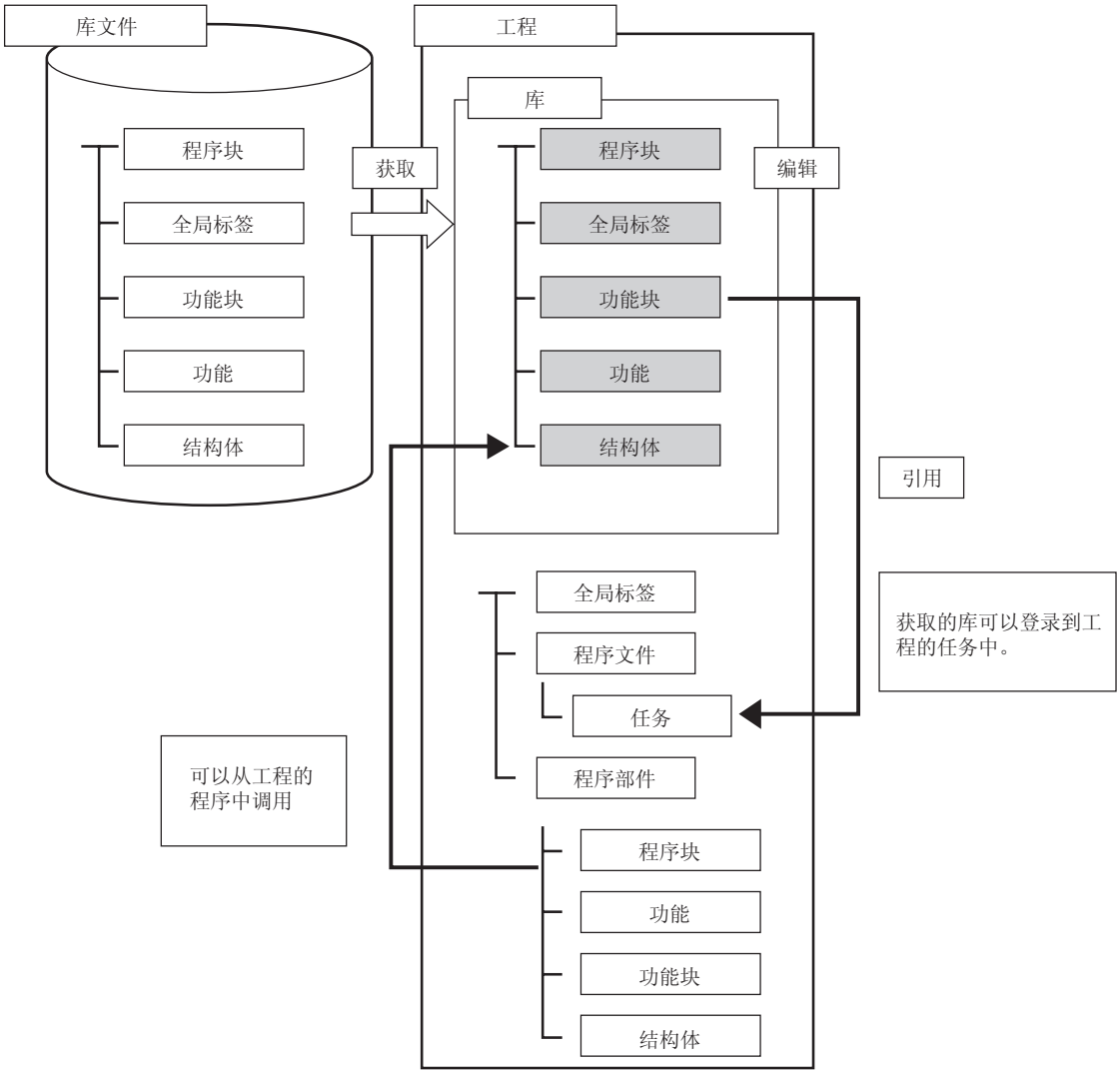
4.7 库

库是指，将程序部件及全局标签、结构体等汇集到一个文件中，使其可以供各个工程共享使用的数据集合。

如果使用了库，将会带来以下优点。

- 在各工程中获取库，可以使库内的数据共享使用。
- 可以将库与部件的各个功能分开创建，因此希望重复利用的部件易于确认。
- 对库中登录的部件进行了变更的情况下，变更结果还可被反映到使用了该部件的工程中。

将库中的部件用于工程中时的数据流程如下所示。



4.7.1 用户库

用户库是指，将用户创建的可共享使用的结构体、全局标签、程序部件等汇集后保存的库。

(1) 用户库的构成

以下内容可登录到用户库中。

表 4.7.1-1 用户库的构成

构成名称	内容
结构体	对库内的程序部件文件夹中使用的结构体的定义及工程内的程序中使用的结构体的定义进行存储。
全局标签	对库内的程序部件文件夹中使用的全局标签的定义进行存储。
程序部件	对可作为库使用的程序块、功能、功能块进行存储。

4.8 附加名称时的注意事项

本节介绍对标签、功能块的实例、结构体标签附加名称时的条件。

• 条件

(1) 应以 32 字以内的字符串进行指定。

(2) 不要使用保留字。
关于保留字请参阅以下内容。
☞ 附录 2 标签名及数据名中可使用的字符串

(3) 应使用以下字符。
半角英文数字、下划线 (_)、全角字符。

(4) 不要在名称最后面使用下划线。
此外，不要连续使用 2 条或以上的下划线。

(5) 不要使用空格。

(6) 起始字符不要使用半角的数字。

(7) 不能使用常数。
(在以 “H” 或 “h” 开始的识别符时，“H” 或 “h” 的紧后面连接了 16 进制数 (0 ～ F) 的情况下 (包括 “H” 或 “h” 在内最多 9 位数 (“H” 或 “h” 的紧后面连接了 0 时除外)) 也将被处理为常数。(例 : “hab0 “))

(8) 不能使用基本数据类型名。

(9) 不能使用函数 /FB 的部件名。

5 程序记述方法

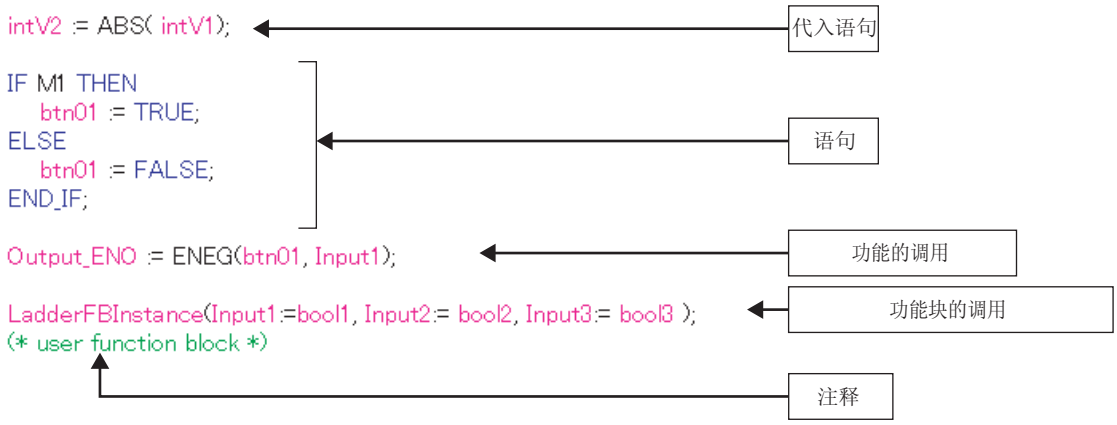
5.1 ST

ST 是具有类似于 C 语言等语法结构的文本格式的程序语言。

可以通过语句记述条件判断及重复等的控制。

ST 适合于对图形语言（结构体梯形图）中难以表述的复杂处理进行编程时使用。

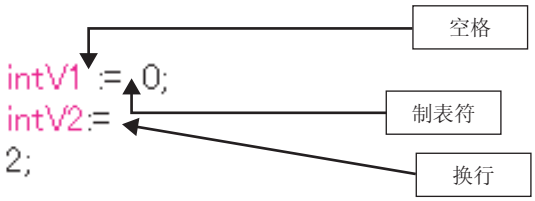
5.1.1 基本格式



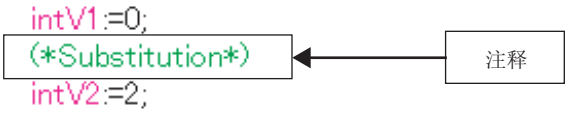
ST 的编程是由运算符及语句所构成。

语句的最后必须附加 “；”。

空格、制表符、换行等可以插入到关键字与识别符之间的任意位置。



程序中可以插入注释。注释的前面记述 “(*”，注释的后面记述 “*)”。



5.1.2 ST 运算符

ST 程序中使用的运算符及运算符的优先顺序如下所示。

表 5.1.2-1 ST 的运算符

运算符	内容	示例	优先顺序
( )	圆括弧式	(1+2)*(3+4)	最高位
函数 ( )	函数 (参数列表)	ADD_E(bo01, in01, in02, in03)	
**	幂	re01:= 2.0 ** 4.4	
NOT	位取反后的值	NOT bo01	
* / MOD	乘法 除法 余数	3 * 4 12 / 3 13 MOD 3	
+ -	加法 减法	in01 + in02 in01 - in02	
<, >, <=, =>	比较	in01 < in02	
= <>	等式 不等式	in01 = in02 in01 <> in02	
AND, &	逻辑积	bo01 & bo02	
XOR	排他逻辑和	bo01 XOR bo02	
OR	逻辑和	bo01 OR bo02	最低位

一个语句中，存在多个优先顺序相同的运算符的情况下，从左侧的运算符开始运算。

运算符及对象数据类型、运算结果的数据类型如下所示。

表 5.1.2-2 运算符中使用的数据类型

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(μÞ) ANY_NUM(指数)	ANY_REAL

5.1.3 ST 语句

ST 程序中可使用的语句如下所示。

表 5.1.3-1 ST 语句

语句的种类	内容
代入语句	代入语句
条件语句	IF THEN 条件语句、IF ELSE 条件语句、IF ELSIF 条件语句
	CASE 条件语句
重复语句	FOR DO 语句
	WHILE DO 语句
	PEPEAT UNTIL 语句
其它的控制语句	RETURN 语句
	EXIT 语句

(1) 代入语句

(a) 格式

〈左边〉 := 〈右边〉;

(b) 说明

代入语句具有将右边的式子的结果代入到左边的标签及软元件中的功能。
在代入语句中，右边式子的结果需要与左边的数据类型相同。

(c) 记述示例

```
intV1:=0;
intV2:=2;
```

(2) IF THEN 条件语句

(a) 格式

IF ＜布尔式＞ THEN
＜语句...＞；
END_IF;

(b) 说明

布尔式（条件式）为真（TRUE）时，执行语句。布尔式为假（FALSE）时，不执行语句。
对于布尔式，作为单一的位型变量的状态或者包含数量较多的变量的复杂式子的布尔运算的结果，只要是返回真（TRUE）或假（FALSE）的式子，无论什么样式子均可使用。

(c) 记述示例

```
IF bool1 THEN
   intV1:=intV1 +1;
END_IF;
```

(3) IF ... ELSE 条件语句

(a) 格式

IF ＜布尔式＞ THEN
＜语句 1...＞；
ELSE
＜语句 2...＞；
END_IF;

(b) 说明

布尔式（条件式）为真（TRUE）时，执行语句 1。
布尔式的值为假（FALSE）的情况下执行语句 2。

(c) 记述示例

```
IF bool1 THEN
   intV3 :=intV3 +1;
ELSE
   intV4 :=intV4 +1;
END_IF;
```

(4) IF ... ELSIF 条件语句

(a) 格式

IF ＜布尔式 1＞ THEN
＜语句 1...＞；
ELSIF ＜布尔式 2＞ THEN
＜语句 2...＞；
ELSIF ＜布尔式 3＞ THEN
＜语句 3...＞；
END_IF;

(b) 说明

布尔式（条件式）1 为真（TRUE）时，执行语句 1。布尔式 1 的值为假（FALSE）而布尔式 2 的值为真（TRUE）的情况下执行语句 2。
布尔式 1、2 的值为假（FALSE）而布尔式 3 的值为真（TRUE）的情况下执行语句 3。

(c) 记述示例

```
IF bool1 THEN
   intV1 :=intV1 +1;
ELSIF bool2 THEN
   intV2 :=intV2 +2;
ELSIF bool3 THEN
   intV3 :=intV3 +3;
END_IF;
```

(5) CASE 条件语句

(a) 格式

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;

(b) 说明

CASE 条件语句的式子的结果将返回整数值。根据诸如单一的整数值或复杂公式的结果的整数值，执行选择语句的情况下可以使用该条件语句。
具有与整数式的值一致的整数选择值的语句将被执行，无一致的选择值的情况下，执行 ELSE 语句的下一个语句。

(c) 记述示例

```
CASE intV1 OF
   1:bool1 :=TRUE;
   2:bool2 :=TRUE;
ELSE
   intV1 :=intV1 +1;
END_CASE;
```

(6) FOR ... DO 语句

(a) 格式

FOR ＜重复变量初始化＞
TO ＜最终值＞
BY ＜增加式＞ DO
＜语句...＞;
END_FOR;

(b) 说明

FOR...DO 语句根据重复变量的值，对若干个语句反复执行。

(c) 记述示例

```
FOR intV1 := 0
   TO 30
   BY 1 DO
   intV3 :=intV1 +1;
END_FOR;
```

(7) WHILE...DO 语句

(a) 格式

WHILE 〈布尔式〉 DO
〈语句...〉;
END_WHILE;

(b) 说明

WHILE...DO 语句在布尔式（条件式）为真（RTUE）期间执行一个以上的语句。
对于布尔式，在执行语句之前事先判断，布尔式为假（FALSE）的情况下，不执行 WHILE...DO 内的语句。对于 WHILE 语句中的〈布尔式〉，无论结果为真还是假只要能返回就可以，因此 IF 条件语句中的〈布尔式〉中可指定的式子均可使用。

(c) 记述示例

```
WHILE intV1 = 30 DO
   intV1 :=intV1 +1;
END_WHILE;
```

(8) REPEAT...UNTIL 语句

(a) 格式

REPEAT
〈语句...〉
UNTIL 〈布尔式〉
END_REPEAT;

(b) 说明

对于 REPEAT...UNTIL 语句，在布尔式（条件）为假（FALSE）期间执行一个以上的语句。
对于布尔式，在执行语句之后进行判断，值为真（TRUE）的情况下，不执行 REPEAT...UNTIL 内语句。
对于 PEPEAT 语句中的〈布尔式〉，无论结果为真还是假只要能返回就可以，因此 IF 条件语句中的〈布尔式〉中可指定的式子均可使用。

(c) 记述示例

```
REPEAT
    intV1 :=intV1 +1;
    UNTIL intV1 = 30
END_REPEAT;
```

(9) RETURN 语句

(a) 格式

RETURN;

(b) 说明

RETURN 语句用于在程序的执行过程中结束程序。
如果在程序中使用 RETURN 语句，RETURN 语句以后的处理将全部被忽略，从 RETURN 被执行的位置开始跳转至程序的最终行。

(c) 记述示例

```
IF bool1 THEN
    RETURN;
END_IF;
```

(10)EXIT 语句

(a) 格式

EXIT;

(b) 说明

EXIT 语句是只能在重复语句中使用的语句，用于使重复语句中途结束。
在循环回路的执行过程中如果到达 EXIT 语句，则 EXIT 语句以后的循环回路处理将不执行。从重复语句结束的下一行开始继续执行程序。

(c) 记述示例

```
FOR intV1 := 0
    TO 10
    BY 1 DO
    IF intV1 > 10 THEN
        EXIT;
    END_IF;
END_FOR;
```

5.1.4 ST 中功能的调用

在 ST 中调用功能时，进行以下记述。

功能名（变量 1、变量 2、...）；

在功能名的后面用 “（）”将自变量围住。

有多各变量的情况下用 “，”分开。

通过代入变量，对功能的执行结果进行存储。

1）输入变量为一个功能（例： ABS）时

```
Output1 := ABS(Input1);
```

2）输入变量为三个功能（例： MAX）时

```
Output1 := MAX(Input1, Input2, Input3);
```

5.1.5 ST 中功能块的调用

在 ST 中调用功能块时，进行以下记述。

```
实例名（输入变量 1: = 变量 1，... 输出变量 1: = 变量 2，...）；
```

实例名的后面将从变量至输入变量、输出变量的代入语句用 “（ ）”围住。

有多个变量的情况下，各代入语句之间用 “，”（逗号）分开。

对于功能块的执行结果，通过在实例名的后面附加 “.”（点号）并指定输出变量数后，代入到变量中进行存储。

1）输入变量为一个、输出变量为一个的功能块

```
FB 定义
FB 名： FBADD
FB 实例名： FBADD1
输入变量 1: IN1
输出变量 1: OUT1
```

调用上述功能块时的记述如下所示。

```
FBADD1(IN1:=Input1);
Output1:=FBADD1.OUT1;
```

2）输入变量为三个、输出变量为二个的功能块

```
FB 定义
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
```

调用上述功能块时的记述如下所示。

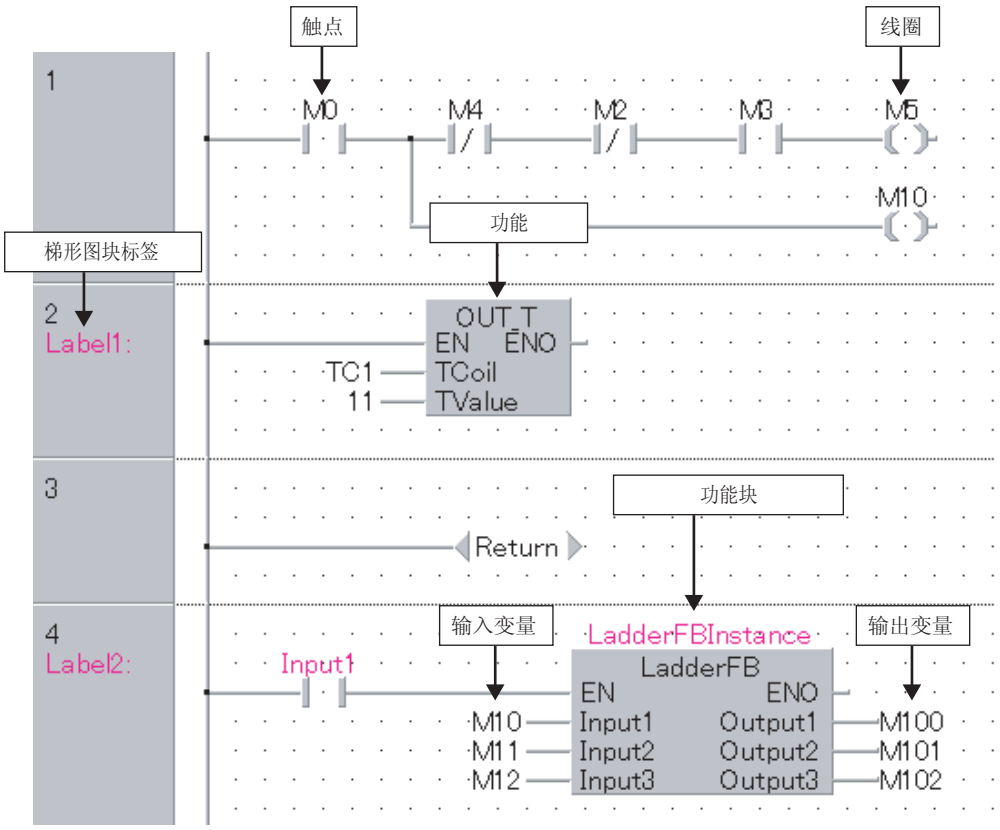
```
FBADD1(IN1:=Input1, IN2:=Input2, IN3:= Input3);
Output1:=FBADD1.OUT1;
Output2:=FBADD1.OUT2;
```

5.2 结构体梯形图

结构体梯形图是使用触点、线圈、功能、功能块等的梯形图符号，将程序用图形进行记述的语言。

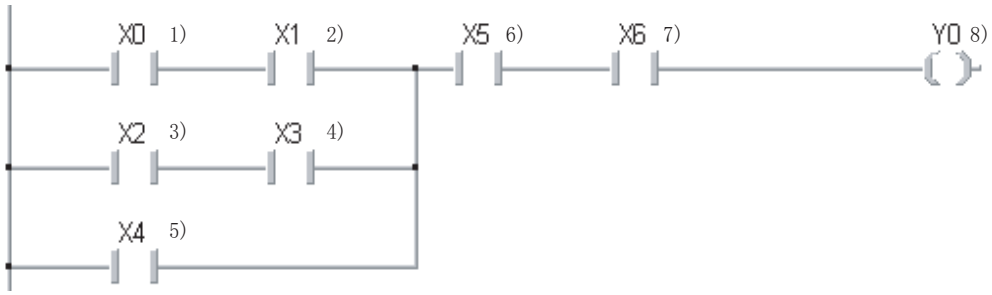
编辑器的左端有母线，通过从母线向连接的电路供应电源的示意图进行编程。

5.2.1 基本格式



结构体梯形图的编程是以梯形图块为单位进行。

梯形图块中的运算顺序是从母线开始向右、由上至下执行。



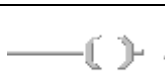
5.2.2 结构体梯形图的梯形图符号

结构体编程中可使用的梯形图符号如下表所示。

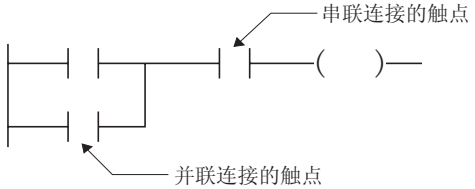
有关详细内容轻参阅以下手册。

☞ MELSEC-Q/L 结构体编程手册（公共指令篇）

表 5.2.2-1 结构体梯形图的梯形图符号 (1/2)

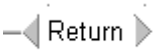
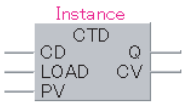
要素	梯形图符号	说明
常开触点 *1		指定软元件或者标签为 ON 时导通。
常闭触点 *1		指定软元件或者标签为 OFF 时导通。
上升沿脉冲 *1,*2		指定软元件或者标签的上升沿时 (OFF → ON) 导通。
下降沿脉冲 *1,*2		指定软元件或者标签的下降沿时 (ON → OFF) 导通。
上升沿脉冲否 *1,*2		指定软元件或者标签的 OFF 时、ON 时以及下降沿时 (ON → OFF) 导通。
下降沿脉冲否 *1,*2		指定软元件或者标签的 OFF 时、ON 时以及上升沿时 (OFF → ON) 导通。
线圈		将运算结果输出到指定软元件或者标签中。
取反型线圈		运算结果为 OFF 时指定软元件或者标签变为 ON。
设置		运算结果为 ON 时指定软元件或者标签变为 ON。 变为 ON 的软元件或者标签即使运算结果变为 OFF 也仍将保持 ON 状态不变。
复位		运算结果为 ON 时指定软元件或者标签变为 OFF。运算结果为 OFF 时软元件或者标签的状态不变化。

*1: 对于触点符号，根据梯形图的连接状态进行 AND 运算、OR 运算后，反映到运算结果中。
- 串联连接时，与到此为止的运算结果进行 AND 运算作为运算结果。
- 并联连接时，与到此为止的运算结果进行 OR 运算作为运算结果。



*2: 与 GX Works2 的版本 1.15R 以后的产品相对应。
关于 GX Works2 的版本的确认方法请参阅以下手册。
☞ GX Works2 Version1 操作手册（公共篇）

表 5.2.2-2　结构体梯形图的梯形图符号 (2/2)

要素	梯形图符号	说明
跳转		是指针分支指令。 无条件执行同一程序文件内指令的指针号的程序。
返回		表示子程序的结束。返回到子程序调用指令的下一步。
功能		执行功能。
功能块		执行功能块。
功能自变量输入		将自变量输入到功能、功能块。
功能恢复值输出		从功能、功能块输出恢复值。
功能自变量取反输入		将自变量取反后输入到功能、功能块。
功能恢复值取反输出		将来自于功能、功能块的恢复值取反后输出。

备忘录

附 附录

附 1 对应于普通数据类型的软元件

对应于普通数据类型的软元件如下所示。

附表 1-1 对应于普通数据类型的软元件

软元件				
分类	类别	软元件名	软元件符号	
内部用户软元件	位软元件	输入	X	
		输出	Y	
		内部继电器	M	
		锁存继电器	L	
		报警器	F	
		变址继电器	V	
		步进继电器	S	
		链接特殊继电器	SB	
		链接继电器	B	
		定时器触点 *1	TS	
		定时器线圈 *1	TC	
		累计定时器触点 *1	STS	
		累计定时器线圈 *1	STC	
		计数器触点 *1	CS	
		计数器线圈	CC	
	字软元件	定时器当前值	T 或者 TN*1	
		累计定时器当前值	ST 或者 STN*1	
		计数器当前值	C 或者 CN*1	
		数据寄存器	D	
		链接寄存器	W	
		链接特殊寄存器	SW	
内部系统软元件	位软元件	功能输入	FX	
		功能输出	FY	
		特殊继电器	SM	
	字软元件	功能寄存器	FD	
		特殊寄存器	SD	

*1 ： 位数指定时可以使用。
*2 ： 位指定时可以使用。

	普通的数据类型												
	ANY											ANY	
	ANY_SIMPLE									数组	结构体		
	ANY_BIT			ANY_NUM				时间	字符串				
				ANY_INT		ANY_REAL							
	位	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]	单精度实数	双精度实数					ANY16	ANY32
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	×	×	×	×	×	×	×	×	×	×	×	×
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	×	○	×	○	×	×	×	×	×	×	×	○	×
	×	○	×	○	×	×	×	×	×	×	×	○	×
	×	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	–	–	–	–	–	–	–	–	–	–	–	–	–
	–	–	–	–	–	–	–	–	–	–	–	–	–
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	–	–	–	–	–	×	×	×	×	×	×		–
	○*2	○	×	○	×	×	×	×	×	×	×	○	×

软元件				
分类	类别	软元件名	软元件符号	
链接直接软元件	位软元件	链接输入	Jn\X	
		链接输出	Jn\Y	
		链接继电器	Jn\B	
		链接特殊继电器	Jn\SB	
	字软元件	链接寄存器	Jn\W	
		链接特殊寄存器	Jn\SW	
智能功能模块软元件	字软元件	智能功能模块软元件	Un\G	
变址寄存器	字软元件	变址寄存器	Z	
文件寄存器	字软元件	文件寄存器	R 或者 ZR	
嵌套	-	嵌套	N	
指针	J	指针	P	
		中断指针	I	
常数	-	-	K, H	
			E	
字符串型常数	-	-	‘字符串’或者“字符串”	

*1 ： 位数指定时可以使用。
*2 ： 位指定时可以使用。

	普通的数据类型											
	ANY										ANY	
	ANY_SIMPLE								数组	结构体		
	ANY_BIT			ANY_NUM				时间	字符串			
				ANY_INT		ANY_REAL						
	位	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]	单精度实数	双精度实数				ANY16	ANY32
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○	○*1	○*1	○*1	○*1	×	×	×	×	×	×	○*1	○*1
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	×	○	×	○	×	×	×	×	×	×	×	○	×
	○*2	○	×	○	×	×	×	×	×	×	×	○	×
	－	－	－	－	－	－	－	－	－	－	－	－	－
	－	－	－	－	－	－	－	－	－	－	－	－	－
	－	－	－	－	－	－	－	－	－	－	－	－	－
	○	○	○	○	○	○	×	×	×	×	×	○	○
	×	×	×	×	×	○	○	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	○	×	×	×	×

附

附录

附 2　标签名及数据名中不能使用的字符串

将应用函数名、公共指令名、特殊指令名、指令语等中使用的字符串称为保留字。

保留字不能作为标签名及数据名使用。如果在标签名及数据名中使用了被定义为保留字的字符串，在执行登录 / 编译时将变为出错状态。

标签名及数据名中不能使用的字符串如下所示。

附表 2-1　标签名及数据名中不能使用的字符串 (1/2)

分类	字符串
分类识别符	VAR, VAR_RETAIN, VAR_ACCESS, VAR_CONSTANT, VAR_CONSTANT_RETAIN, VAR_INPUT, VAR_INPUT_RETAIN, VAR_OUTPUT, VAR_OUTPUT_RETAIN, VAR_IN_OUT, VAR_IN_EXT, VAR_EXTERNAL, VAR_EXTERNAL_CONSTANT, VAR_EXTERNAL_CONSTANT_RETAIN, VAR_EXTERNAL_RETAIN, VAR_GLOBAL, VAR_GLOBAL_CONSTANT, VAR_GLOBAL_CONSTANT_RETAIN, VAR_GLOBAL_RETAIN
数据类型	BOOL, BYTE, INT, SINT, DINT, LINT, UINT, USINT, UDINT, ULINT, WORD, DWORD, LWORD, ARRAY, REAL, LREAL, TIME, STRING
数据类型分级	ANY, ANY_NUM, ANY_BIT, ANY_REAL, ANY_INT, ANY_DATE, ANY_SIMPLE, ANY16, ANY32
软元件名	X, Y, D, M, T, B, C, F, L, P, V, Z, W, I, N, U, J, K, H, E, A, SD, SM, SW, SB, FX, FY, DX, DY, FD, TR, BL, SG, VD, ZR, ZZ
被识别为软元件的字符串（软元件名＋数字）	X0 等
ST 运算符	NOT, MOD
IL 运算符	LD, LDN, ST, STN, S, S1, R, R1, AND, ANDN, OR, ORN, XOR, XORN, ADD, SUB, MUL, DIV, GT, GE, EQ, NE, LE, LT, JMP, JMPC, JMPCN, CAL, CALC, CALCN, RET, RETC, RETCN, LDI, LDP, LDPI, LDF, LDFI, ANI, ANDP, ANDPI, ANDF, ANDFI, ANB, ORI, ORP, ORPI, ORF, ORFI, ORB, MPS, MRD, MPP, INV, MEP, MEF, EGP, EGF, OUT(H), SET, RST, PLS, PLF, FF, DELTA(P), SFT(P), MC, MCR, STOP, PAGE, NOP, NOPLF
GX Works2 中的应用指令	DMOD, PCHK, INC(P) 等的应用指令 ☞ MELSEC-Q/L 编程手册（公共指令篇）、MELSEC-Q/L 结构体编程手册（公共指令篇） ☞ FXCPU 结构体编程手册（顺控程序指令篇）、FXCPU 结构体编程手册（应用函数篇）
SFC 指令	SFCP, SFCPEND, BLOCK, BEND, TRANL, TRANO, TRANA, TRANC, TRANCA, TRANOA, SEND, TRANOC, TRANOCA, TRANCO, TRANCOC, STEPN, STEPD, STEPSC, STEPSE, STEPST, STEPR, STEPC, STEPG, STEPI, STEPID, STEPISC, STEPISE, STEPIST, STEPIR, TRANJ, TRANOJ, TRANOCJ, TRANCJ, TRANCOJ, TRANCOCJ
ST 载码体	RETURN, IF, THEN, ELSE, ELSIF, END_IF, CASE, OF, END_CASE, FOR, TO, BY, DO, END_FOR, WHILE, END_WHILE, REPEAT, UNTIL, END_REPEAT, EXIT, TYPE, END_TYPE, STRUCT, END_STRUCT, RETAIN, VAR_ACCESS, END_VAR, FUNCTION, END_FUNCTION, FUCTION_BLOCK, END_FUCTION_BLOCK, STEP, INITIAL_STEP, END_STEP, TRANSITION, END_TRANSITION , FROM, TO, UNTILWHILE
标准功能名	AND_E, NOT_E 等的应用函数的功能名

附表 2-1　标签名及数据名中不能使用的字符串 (2/2)

分类	字符串
标准功能块名	CTD, CTU 等的应用函数的功能块名
符号	/, \, *, ?, <, >, |, ″, :, [,], ;, ,, =, +, %, ’, ~, @ {, }, !, #, $, &
日期时间文字	DATE, DATE_AND_TIME, DT, TIME, TIME_OF_DAY, TOD
其它	ACTION, END_ACTION, CONFIGURATION, END_CONFIGURATION, CONSTANT, F_EDGE, R_EDGE, AT, PROGRAM, WITH, END_PROGRAM, TRUE, FALSE, READ_ONLY, READ_WRITE, RESOURCE, END_RESOURCE, ON, TASK, EN, ENO, BODY_CCE, BODY_FBD, BODY_IL, BODY_LD, BODY_SFC, BODY_ST, END_BODY, END_PARAMETER_SECTION, PARAM_FILE_PATH, PARAMETER_SECTION, SINGLE, TRUE, FALSE, RETAIN, INTERVAL, L, P
以 K1 ～ K8 开始的字符串	K1AAA 等
地址	%IX0 等
梯形图语言的声明	;FB BLK START, ;FB START, ;FB END, ;FB BLK END, ;FB IN, ;FB OUT, ;FB_NAME;,INSTANCE_NAME, ;FB, ;INSTANCE
公共指令	MOV 等
Windows 保留字	COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9, AUX, CON, PRN, NUL

(1) 使用标签时的其它注意事项

- 不能使用与任务、结构体、程序部件等的数据名相同的标签名、实例名。
- 不能使用空格。
- 起始字符不能使用半角的数字。
- 标签名不区分字母的大小写。编译时将变为出错状态。
- 在结构体梯形图及 ST 中，通过 GX Works2 的下述选项设置 *1 全局标签及局部标签可以使用相同的标签名。

*1:　［工具］→［选项］→“编译”→“结构体梯形图 /ST”→“编译条件 1”中“全局标签与局部标签使用相同的标签名”

附 3　梯形图的替换

将与程序语言的梯形图中创建的程序相同内容的结构体程序通过 GX Works2 创建时的示例如下所示。

附 3.1　创建步骤

以程序语言的梯形图中创建的程序为基础，创建结构体程序时的基本步骤如下所示。

1. 从软元件替换为标签

步骤
标签中有全局标签及局部标签。 将软元件替换为标签时，确定标签的种类（全局标签、局部标签）。

2. 标签设置

步骤
程序中使用的全局标签、局部标签需要进行定义。 对程序中使用的所有标签进行定义。

3. 程序创建

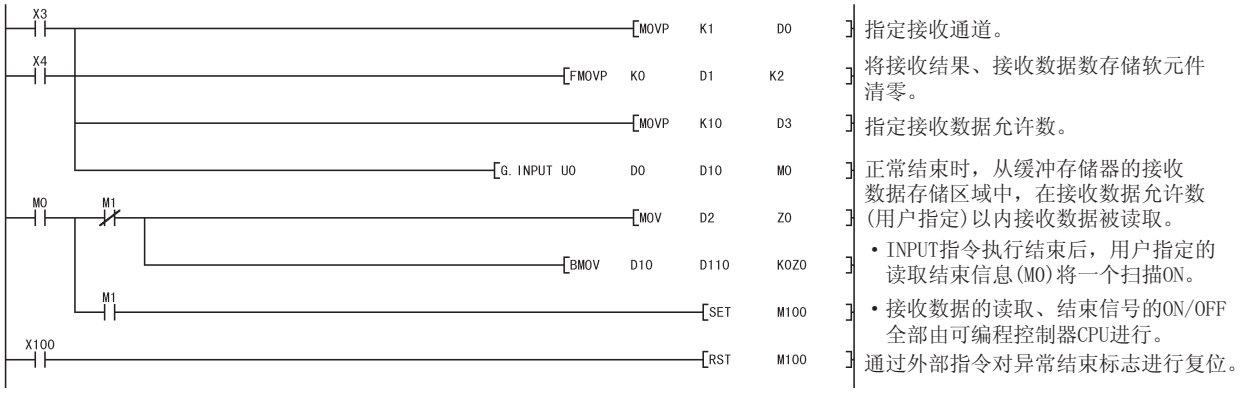
步骤
通过使用的程序语言创建结构体程序。

附 3.2 创建示例

将与 GX Developer 中创建的顺控程序相同内容的结构体程序通过 GX Works2 创建时的示例如下所示。

作为创建示例，将与 Q 系列串行通信模块的数据接收程序相同内容的结构体程序，通过程序语言的结构体梯形图及 ST 创建的示例如下所示。

以下为创建的源程序。



(1) 从软元件替换为标签

将使用的软元件替换为标签。

输入输出软元件等可被替换为全局标签。内部继电器等可以被替换为局部标签。

附表 3.2-1 从软元件替换为标签的示例

软元件	用途		标签	
			数据类型	标签名
X3	CH1 接收读取请求		位	CH1ReadRequest
X4	CH1 接收异常检测		位	CH1AbnormalDetection
D0	控制数据	接收通道	字 [无符号]/位串 [16 位] [0] ～ [3]	ControlData
D1		接收结果		
D2		接收数据数		
D3		接收数据允许数		
D10 ～ D109	接收数据		字 [无符号]/位串 [16 位] [0] ～ [99]	RecieveData
D110 ～ D209	接收数据存储区域		字 [无符号]/位串 [16 位] [0] ～ [99]	Data
M0	接收结束标志	结束标志	位 [0] ～ [1]	Completion
M1		结束时的状态标志		
M100	异常结束标志		位	AbnormalCompletion
X100	异常结束标志复位指令		位	ResetAbnormalCompletion

附录

(2) 标签设置

进行全局标签及局部标签的设置。

• 全局标签的设置示例

	Class	Label Name	Data Type		Constant	Device	Address
1	VAR_GLOBAL	CH1 ReadRequest1	Bit	...		X3	%IX3
2	VAR_GLOBAL	CH1 AbnormalDetection	Bit	...		X4	%IX4
3	VAR_GLOBAL	ResetAbnormalCompletion	Bit	...		X100	%IX256

• 局部标签的设置示例 *1

Class	Label Name	Data Type		Constant
VAR	ControlData	Word[Unsigned]/Bit String[16-bit](0..3)	...	
VAR	ReceiveData	Word[Unsigned]/Bit String[16-bit](0..1)	...	
VAR	Completion	Bit(0..1)	...	
VAR	Data	Word[Unsigned]/Bit String[16-bit](0..9)	...	
VAR	AbnormalCompletion	Bit	...	

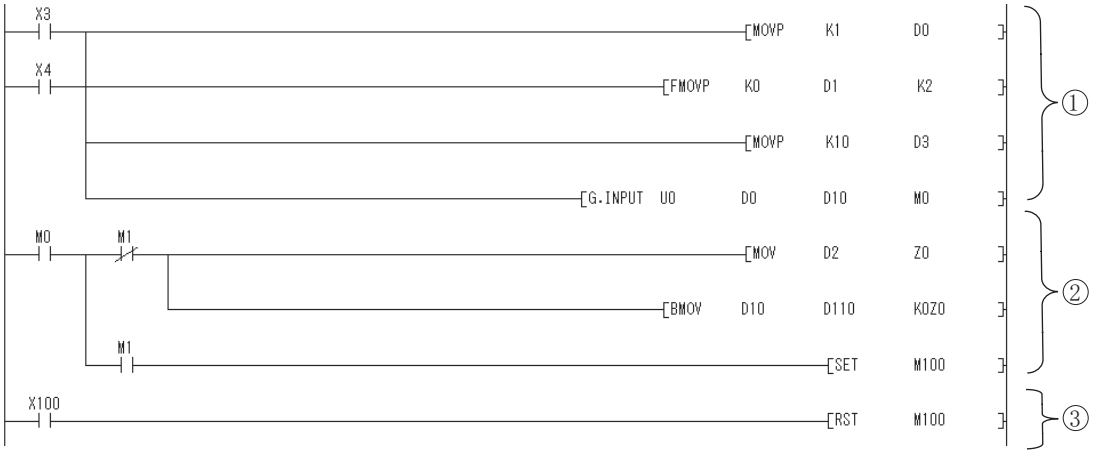
*1 ：对于局部标签的软元件，在 GX Works2 的自动分配软元件设置中指定的范围内被自动分配。

分配与以前的梯形图程序相同软元件的情况下，应通过全局标签进行设置。

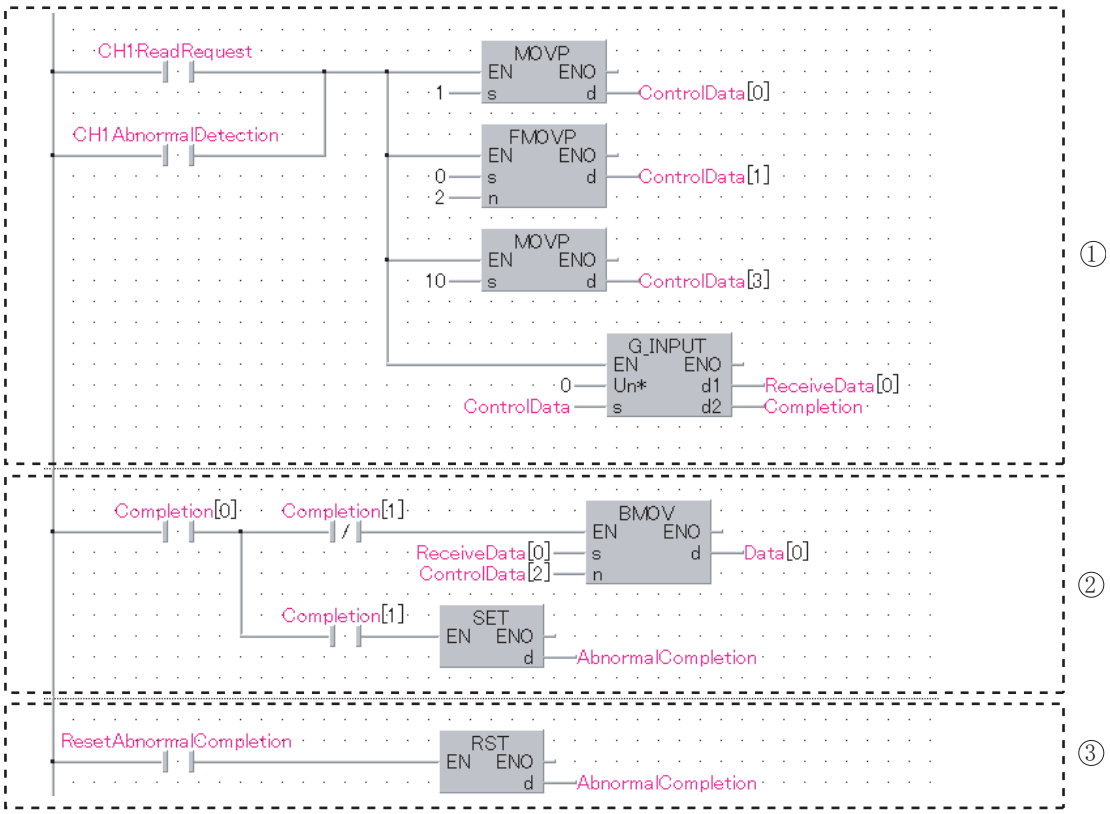
(3) 结构体程序的创建

从创建的源程序创建结构体程序。

• 创建的源程序（程序语言： 梯形图）



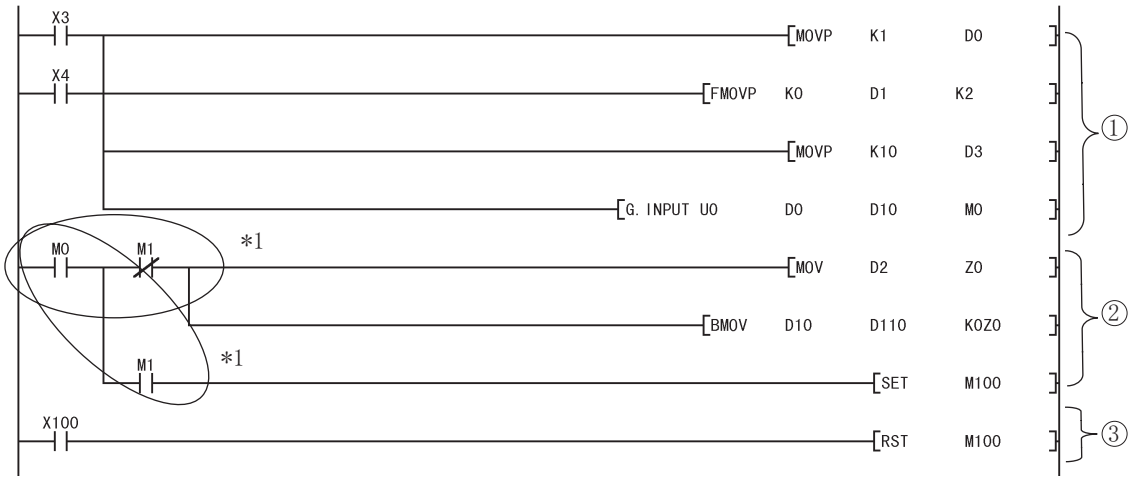
• 结构体程序（程序语言： 结构体梯形图）



附

附录

• 创建的源程序（程序语言： 梯形图）



• 结构体程序（程序语言： ST）

```
IF CH1ReadRequest OR CH1AbnormalDetection THEN

    ControlData[0] := 1;
    ControlData[1] := 0;
    ControlData[2] := 0;
    ControlData[3] := 10;

    G_INPUT(TRUE, 0, ControlData, ReceiveData[0], Completion);

END_IF;                                                          ①

                                        *1
BMOV(Completion[0] AND NOT Completion[1], ReceiveData[0], ControlData[2], Data[0]);
                                                                 ②
                                    *1
SET(Completion[0] AND Completion[1], AbnormalCompletion);

RST(ResetAbnormalCompletion, AbnormalCompletion);                ③
```

*1: 有多个执行条件触点的情况下，可以用 “( )” 将其围住，对多个触点进行批量编程。

索引

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责任的故障或缺陷（以下称“故障”），则经销商或三菱服务公司将负责免费维修。

注意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货到目的地日的一年内。

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

[免费质保范围]

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

(2) 以下情况下，即使在免费质保期内，也要收取维修费用。

1. 因不适当存储或搬运、用户粗心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技术公告等方式予以通告。

(2) 产品停产后，将不再提供产品（包括维修零件）。

3. 海外服务

在海外，维修由三菱在当地的海外 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，对于任何非三菱责任的原因而导致的损失、机会损失、因三菱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。

6. 产品应用

(1) 在使用三菱 MELSEC 通用可编程控制器时，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保险功能。

(2) 三菱通用可编程控制器是以一般工业用途等为对象设计和制造的。因此，可编程控制器的应用不包括那些会影响公共利益的应用，如核电厂和其它由独立供电公司经营的电厂以及需要特殊质量保证的应用如铁路公司或用于公用设施目的的应用。

另外，可编程控制器的应用不包括航空、医疗应用、焚化和燃烧设备、载人设备、娱乐及休闲设施、安全装置等与人的生命财产密切相关以及在安全和控制系统方面需要特别高的可靠性时的应用。

然而，对于这些应用，假如用户咨询当地三菱代表机构，提供有特殊要求方案的大纲并提供满足特殊环境的所有细节及用户自主要求，则可以进行一些应用。

MELSEC-Q/L/F结构体编程手册

基础篇

三菱电机自动化
400-821-3030
CALL CENTER 技术支持热线
周一至周五 9:00~17:00(法定节假日除外)

三菱电机自动化(中国)有限公司
地址：上海市黄浦区南京西路288号创兴金融中心17楼
邮编：200003
电话：021-23223030 传真：021-23223000
网址：www.meas.cn

书号	SH(NA)-080903CHN-A(1004)STC
印号	STC-MELSEC-Q/L/F(B)-S-PM(1004)