编号：W447-CN5-02

CX-Programmer
6.1版
WS02-CXPC1-E-V61
CS1-H, CJ1-H, CJ1M, CP1H CPU单元
NSJ, FQM1

功能块

操 作 手 册

OMRON

编号：W447-CN5-02 CX-Programmer 6.1版 功能块
操作手册
OMRON

OMRON

特约经销商

编号：W447-CN5-02

注：规格等随时可能变化，恕不另行通知。

上海印刷
200606S01

CX-Programmer

6.1 版
WS02-CXPC1-E-V61

CS1-H, CJ1-H, CJ1M, CP1H CPU 单元
NSJ, FQM1

操作手册
功能块

于 2006 年 06 月修订

注意：

欧姆龙产品是为合格的操作人员按照正常步骤使用，并只为本手册中所叙述的目的而制造的。

下列约定是用来指出本手册中的注意事项，并对其进行分类。始终注意它们所规定的情况。不注意这些事项可能导致对人体的伤害或危及财产。

⚠ 危险　　表示一个紧迫的危险情况，如不避免可能导致死亡或严重伤害。

⚠ 警告　　表示一个潜在的危险情况，如不避免可能导致死亡或严重伤害。

⚠ 注意　　表示一个潜在的危险情况，如不避免可能导致轻度或中度伤害，或财产损失。

欧姆龙产品附注

所有欧姆龙产品在本手册中都用大写字母表示，当“ 单元 ”表示欧姆龙产品时，它也以大写字母表示，不管它是否以产品的正式名称表示。

缩写“ Ch ”出现在某些显示中和某些欧姆龙产品上，往往表示“ 字 ”，在这个意义上在文件中缩写为“ Wd ”。

缩写“ PLC ”表示可编程序控制器，但是“ PLC ”在某些编程设备的显示中也表示可编程序控制器。

直观标题

列在本手册左侧的下列标题是帮助读者确定各种不同类型的资料。

注　　表示对有效而方便地运用产品特别重要的资料。

1,2,3...　　1.　表示一种或另一种的列举说明，如步骤，检查表等。

目录

目录

关于本手册：

本手册描述功能块及 6.1 版本的 CX-Programmer 与 CP1H CPU 单元和 3.0 版本以上的 CS1-H、 CJ1-H、CJ1M CPU 单元一同使用时的相关功能，并包含下一页所述的部分。

6.1 版本的 CX-Programmer 是允许个人计算机被当作功能块编程设备，并可用于支持功能块的 SYSMAC CS 系列和 CJ 系列 CPU 单元的软件。6.1 版本的 CX-Programmer 功能块功能已作了改进本手册仅描述与功能块相关的 6.1 版本的 CX-Programmer。对于与功能块无关的操作，请参阅 CX-Programmer 操作手册（附带，样本编号 W437）。本手册亦提供 CS1-H、 CJ1-H、 CJ1M 和 CP1H CPU 单元中仅与功能块相关的信息。其它信息请参阅 CS/CJ/CP 系列手册。

请在尝试安装或操作 6.1 版本 CX-Programmer 或 CS1-H、CJ1-H、CJ1M 或 CP1H CPU 单元之前仔细阅读本手册及相关手册并确保理解其中提供的信息。必须阅读以下章节中提供的注意事项。

与 6.1 版 CX-Programmer 有关的手册

名称	样本编号	内容
SYSMAC WS02-CXPC1-E-V60 6.1 版 CX-Programmer 功能块操作手册 （CS1G-CPU □□ H, CS1H-CPU □□ H, CJ1G-CPU □□ H, CJ1H-CPU □□ H, CJ1M-CPU □□ , CP1H-X □□□□ - □ , CP1H-XA □□□□ - □ , CP1H-Y □□□□ - □ CPU 单元）	W447 （本手册）	叙述只有 6.1 版 CX-Programmer 和 CP 系列 CPU 单元或基于功能块的 3.0 版以上的 CS/CJ 系列 CPU 单元才具备的功能性。

与 CS1-H, CJ1-H, CJ1M CPU 单元有关的手册

名称	样本编号	内容
SYSMAC CS 系列 CS1G/H-CPU □□ -EV1, CS1G/H-CPU □□ H 可编程控制器 操作手册	W339	提供了对 CS 系列 PLC 的设计、安装、维护和其它基本操作的概述并作了说明。 包含以下信息： 概述及特性 系统配置 安装及接线 I/O 存储器分配 故障排除 将本手册随 W394 一同使用。
SYSMAC CJ 系列 CJ1G-CPU □□ , CJ1G/H-CPU □□ H, CJ1G-CPU □□ P, CJ1M-CPU □□ 可编程控制器 操作手册	W393	提供了对 CS 系列 PLC 的设计、安装、维护和其它基本操作的概述并作了说明。 包含以下信息： 概述及特性 系统配置 安装及接线 I/O 存储器分配 故障排除 将本手册随 W394 一同使用。
SYSMAC CS/CJ 系列 CS1G/H-CPU □□ -EV1, CS1G/H-CPU □□ H, CJ1G-CPU □□ , CJ1G/H-CPU □□ H, CJ1G-CPU □□ P, CJ1M-CPU □□ 可编程控制器 编程手册	W394	叙述了要使用 CS/CJ 系列 PLC 的功能而须进行的编程和其它方法。 包含以下信息： 编程 任务 文件存储器 其它功能 将本手册随 W339 或 W393 一同使用。
SYSMAC CS/CJ 系列 CS1G/H-CPU □□ -EV1, CS1G/H-CPU □□ H, CJ1G-CPU □□ , CJ1G/H-CPU □□ H, CJ1G-CPU □□ P, CJ1M-CPU □□ 可编程控制器 指令参考手册	W340	叙述了 CS/CJ 系列 PLC 所支持的梯形图编程指令。 编程时，将本手册随操作手册 （CS1：W339 或 CJ1：W393）和编程手册 （W394）一同使用。

内容概述

注意事项提供使用 6.1 版 CX-Programmer 时的一般注意事项。

第 1 章介绍了 CX-Programmer 功能块的功能性，并说明了非功能块版本的 CX-Programmer 中所不包含的特性。

第 2 章提供了使用功能块时的参考规格，包括功能块、实例和所兼容的 PLC 的规格，还有使用注意事项和指南。

第 3 章叙述了在 CX-Programmer 上创建功能块的步骤。

附录提供了关于数据类型、结构文本规格和外部变量的信息。

⚠ 警告	若不阅读和理解本手册中所提供的信息可能导致人身伤害或死亡、产品受损或产品失灵。请在尝试任何操作之前完整阅读所有章节并确保理解各章节及其相关章节中提供的信息。

阅读并理解本手册

请在使用产品前阅读并理解本手册。如有任何问题或意见，请联系您的欧姆龙代表。

保证内容和责任限定

保证内容

欧姆龙的唯一保证是产品自售出起一年 （或其它指定时间）内在材料和工艺上没有缺点。

欧姆龙对产品的无侵权、可售性或特殊用途的适用性均无任何明示或暗示性担保。任何购买者或使用者须承认单独的购买者或使用者决定了产品将适当地符合他们有意使用的需求。欧姆龙拒绝其它所有保证，无论明确的或暗示的。

责任限定

欧姆龙将不为产品以任何方式造成的特殊、间接的或因此产生的损坏、利益损失或商业损失负责，无论此主张是基于契约、保证、疏忽或严格赔偿责任。

欧姆龙对已宣称责任的产品的任何超越单价的行为概不负责。

欧姆龙对产品的授权，修理或其它主张概不负责，除非欧姆龙分析确认产品完全操作、储藏、安装和维护且未遭受污染、滥用、误用或不当更改或修理。

应用注意事项

使用的适宜性

欧姆龙将不对所有应用于客户应用中的产品结合对各个标准、代码或规章的符合性负责。

在客户的需求中，欧姆龙将提供可用的第三方证明文件来确定产品使用的额定值和局限性。该信息本身对于完全确定产品与其它产品、机器、系统或其它应用或使用的适宜性并不充分。

以下为一些必须特别注意的应用示例。这并不是详尽无遗地罗列了所有可能的产品用途的列表，也并不意味着所列用途对产品适用：

- 户外使用、遭受潜在化学污染或电干扰处使用、或未在本手册中提及的条件或用途。
- 核能控制系统、燃烧系统、铁路系统、航空系统、医疗器材、娱乐机械、车辆、安全设备和服从分离工业或政府规章的安装。
- 可能对生命或财产造成风险的系统、机器和设备。

请了解并遵守所有产品可用性的禁止条款。

切勿将本产品用于可能造成严重生命或财产风险且不能保证整个系统设计于从事风险的场合，欧姆龙产品已为了用在整个设备或系统里而适当地额定并已安装。

可编程产品

欧姆龙将不对可编程设备中用户的编程或其任何结果负责。

不承诺事项

规格的变更

产品规格和附件随时可能基于改进和其它原因而更改。

当已公布的额定值或特性改变，或作出重大结构改变时更改型号编号是我们惯例。但是，产品某些规格可能作出更改而不予通知。如有疑问，可指派特殊型号编号以为您的请求固定或建立关键规格。请在任何时候联系您的欧姆龙代表以确认所够产品的实际规格。

尺寸和重量

尺寸和重量仅为名义上的，并不能用作制造用途，即使已说明了公差。

性能数据

本手册所给出的性能数据是用作给用户作为确定适用性的向导，并不予以担保。其将可能表现出欧姆龙测试条件下的结果，用户必须将其与实际应用需求相联系。实际性能服从欧姆龙的保证以及责任限定。

错误和疏忽

本手册中的信息已小心核对并被认为正确；但是对记录、印刷或校对错误或疏忽并不指定责任。

注意事项

本章给出使用 6.0 版 CX-Programmer 和可编程逻辑控制器的一般注意事项。

本章中所包含的资料对 6.0 版 CX-Programmer 和可编程控制器的安全和可靠应用是非常重要的。在着手装配或操作 6.0 版 CX-Programmer 和可编程控制器前务必阅读本章节并理解其内容。

1 面向的读者

本手册是为下列人员编写的，他必须具有电气系统知识 （电气工程师或同等水平者）。

- 从事 FA 系统安装的人员。
- 从事 FA 系统设计的人员。
- 从事 FA 系统及设备管理的人员。

2 一般注意事项

用户必须按照操作手册中给出的性能规格来运用产品。

在将本产品用于本手册中未述及的条件下，或将产品应用于核控制系统、铁路系统、航空系统、车辆、内燃机系统、医疗装置、娱乐机械、安全装置或若使用不当时可能对生命和财产造成严重影响的其它系统、机械和装置之前，请务必咨询欧姆龙的特约经销商。

请确保本产品的额定值和性能特性满足系统、机械和装置的要求，并务必给系统、机械和装置提供双重的安全机制。

本手册编有供单元的编程和操作作用的资料。在着手使用前务必阅读本手册，并将手册备在手边以供操作时参阅。

⚠ 警告 PLC 和所有 PLC 单元用于规定用途和规定条件下是很重要的，特别在能直接或间接影响人的生命的应用中。在将 PLC 系统应用于上述情况前，请务必咨询欧姆龙的特约经销商。

3 安全注意事项

⚠ 警告 将I/O存储器区状态从6.0版CX-Programmer传送到实际CPU单元之前充分确认安全性。否则连接到输出单元上的设备可能发生故障，不论 CPU 单元处于什么操作模式下。涉及以下功能时须多加小心。

- 用 PLC 存储器窗口从 CX-Programmer 传送到 CPU 单元中的真实 I/O （CIO 区）。
- 用存储器卡窗口从文件存储器传送到 CPU 单元中的真实 I/O （CIO 区）。

⚠ 注意 指定操作数中多个字的首地址或末地址时必须用 AT 设定（或外部变量）来指定变量，或者变量必须与要由指令来处理的数据长度相同。

1. 若指定了数据长度不同的非数组变量或没有 AT 设定，CX-Programmer 将在编译时输出一个错误。
2. 数组变量规格

- 要由指令操作数来处理的长度为固定时：
  数组元素的数量必须与要由指令处理的元素数量相同。否则，CX-Programmer 将在编译时输出一个错误。
- 要由指令操作数来处理的长度不固定时：
  数组元素的数量必须大于或等于其它操作数中指定的长度。
  - 若其它用于指定一个长度的操作数是一个常量，6.0版的CX-Programmer将在编译时输出一个错误。
  - 若另一个用于指定一个长度的操作数为变量，6.0 版的 CX-Programmer 在编译时不会输出错误，即使数组变量的长度与由另一个操作数（变量）所指定的长度不同也一样。但是，将会显示一条警告消息。特别地，若数组元素的数量小于由另一操作数所指定的长度（例如，指令操作数长度为 16，实际变量表中注册的元素数为 10），则指令将对超出元素数的那部分执行读 / 写处理。例如，将对实际变量表中注册的元素数之后的 6 个字执行读 / 写处理。若这些字还被用在其它指令（包括内部变量分配）中，将发生意外动作，可能引起严重事故。

    若操作数中指定的变量长度小于变量定义中的长度，必须在开始 PLC 操作之前检查系统，避免其受逆反影响。

⚠ 注意 将程序传送到另一结点或更改 I/O 存储器区内容前应先在目的结点处确认安全性。否则可能导致人身伤害。

⚠ 注意 仅在确认周期时间的延长不会造成不良影响之后再进行在线编辑。否则输入信号可能无法读取。

⚠ 注意 在梯形部分窗口监控导通状态和当前值状态，或在观察窗口监控当前值之前应先充分确认安全性。若不慎按下了快捷键执行了强制设置 / 强制复位或设置 / 复位操作，连接到输出单元的设备可能发生故障，无论CPU处于什么操作模式下。

4 应用注意事项

使用 CX-Programmer 时应遵守以下注意事项。

- 用户程序无法上传到 CX-Programmer 中。
- 启动 CX-Programmer 之前应遵守以下注意事项。
  - 退出所有与CX-Programmer不直接相关的应用程序。尤其是要退出诸如屏幕保护程序、病毒检查程序、E-mail 或其它通信软件、调度程序以及其它定期启动或自动启动的应用程序等软件。
  - 禁用任何网络上的其它计算机上的共享硬盘、打印机或其它设备。

  - 对于某些笔记本计算机，RS-232C端口被默认地分配到调制解调器或红外线路上。根据用于您计算机的文档中的指示来启用 RS-232C 端口作为正常串行端口。
  - 对于某些笔记本计算机，省电的默认设定不支持RS-232C端口的额定电压。可能同时有用于省电的 Windows 设定和用于指定计算机软件和 BIOS 的设定。根据用于您计算机的文档中的指示来禁用所有省电设定。
- 不要在CX-Programmer与PLC在线时关闭PLC的电源或断开连接电缆。否则计算机可能发生故障。
- 尝试以下任何操作前应先确认不会对系统造成不良影响。否则可能引起意外动作。
  - 更改 PLC 的运行模式。
  - 对存储器中的任意位予以强制设置 / 强制复位。
  - 更改存储器中的任何字或任何所设值的当前值。
- 在单元上实际运行用户程序之前应先对其进行检查以便正确执行。否则可能引起意外动作。
- 执行了在线编辑后，CS1-H、CJ1-H、CJ1M 和 CP1H CPU 单元中的用户程序和参数区数据会备份在内置闪存中。正在进行备份操作时 CPU 单元前的 BKUP 显示灯将亮。 BKUP 显示灯亮时不要关闭 CPU 单元的电源。若电源关闭，则数据将不被备份。要在 CX-Programmer 上显示写入闪存的状态，选择 “ 显示 ” 对话框在 PLC 属性中显示 PLC 存储器备份状态，然后从查看菜单中选择 Windows - PLC 存储器备份状态。
- 包括功能块（梯形编程语言或结构化文本 (ST) 语言）在内的程序可以用和不包括功能块的标准程序相同的方法来上传或下载。但是，包括功能块在内的任务无法在任务单元中下载 （上传可以）。
- 若包含用6.0版或以上的CX-Programmer创建的功能块的用户程序被下载到不支持功能块的 CPU 单元（2.0 版或以下的 CS/CJ 系列 CPU 单元）中，所有实例都将被视为非法指令，并且无法编辑或执行用户程序。
- 若输入变量数据不是布尔格式，且参数中只输入了数值（例如 20），CIO 区地址中的实际值 （例如 0020）将被略过。因此，输入数值前必须加入 &、#、或 +、 - 前缀。
- 可在输入参数中设置地址，但地址本身无法被当作输入变量而略过 （即使地址被设为输入参数，被功能块略过的数值将为输入变量中的数据长度与之相同的那个）。因此，由操作数来指定多个字中的第一个或最后一个时，输入变量无法被用作功能块中的指令操作数。应使用带有 AT 设定的内部变量。在内部数组变量中交替地指定第一个或最后一个元素。

- 执行算法之前 （不与算法中的指令执行同时）数值被成批地经过输入参数到达输入变量。因此，要在执行功能块算法中的指令时使数值经过参数到达输入变量，应使用内部变量或外部变量代替输入变量。将数值从输出变量写入参数的时序中也有相同情况。
- 遇到下列情况时总是使用带 AT 设定的内部变量。
  - 分配到基本I/O单元、特殊I/O单元和CPU总线单元的地址无法注册为全局符号，并且这些变量无法被指定为外部变量 （例如，为全局变量所设的数据可能不稳定）。
  - 除了那些预注册到外部变量中的辅助区位以外的辅助区位被注册为全局符号且这些变量未被指定为外部变量时应使用内部变量。
  - 为网络中另一结点指定 PLC 地址（例如，远程结点处用于 SEND(090) 的首目的字和远程结点处用于 RECV(098) 的首源字）时应使用内部变量。
  - 多个字中的第一个或最后一个被指令操作数所指定并且该操作数无法被指定为内部数组变量 （例如，无法指定数组元素）时应使用内部变量。

第 1 章
介绍

本章主要介绍了 CX-Programmer 功能块的功能。此外，对一些未在 CX-Programmer 非功能块描述中所提及的特性作了说明。

1-1 功能块介绍

1-1-1 概述和特点

CX-Programmer 5.0 版（和以后的版本）是一个可以使用标准 IEC 61131-3 功能块的编程工具。CP1H CPU 单元、NSJ- 系列 NSJ 控制器和 FQM1 柔性运动控制器以及 CS/CJ- 系列 3.0 版或以后的版本均支持 CX-Programmer 5.0 版功能块功能。该功能块具有以下特性：

- 通过功能块可以将用户自定义的过程转换成程序段格式。
- 可以采用梯级编程语言或结构化文本 （ST）语言 （见注解）来写功能块运算法则。
  - 当采用梯级编程时，采用复制粘贴的方法可重新使用非 CX-Programmer 4.0 版或更早版本所创建的梯形图程序。
  - 那些采用梯形图编程而难以输入的数学运算步骤如采用 ST 语言进行编程，则编程十分容易。

  注 ST 语言对 IEC61131-3 中所描述的工业控制 （主要是编程逻辑控制器）而言是一个高级语言。CX-Programmer 支持的 ST 语言符合 IEC 61131-3 标准。
- 由于变量未显示于文本中，因此，容易创建功能块。这些变量注册于变量表中。当变量输入于梯级或 ST 程序中时，变量会自动注册。变量注册于变量表后，注册的变量也可输入于梯级程序中。
- 单个功能块可以转换成库功能作为单个文件。这样，便于重复使用功能块进行标准处理。
- 对单个功能块程序进行检查以便确认作为库功能时功能块的可靠性。
- 与不含功能块的程序一样，含功能块的程序 （梯级编程语言或结构化文本（ST）语言）也可上载或下载。但含功能块的任务就不可下载于任务单元中 （可以上载）。
- 支持一维数组变量。因此，在许多应用中数据处理较为方便。

  注 IEC 61131 标准是由国际电工委员会 （IEC）制订的。其作为国际编程逻辑控制器 （PLC）的标准。标准分成 7 个部分。有关 PLC 编程的相关规范请见第 3 部分—文本语言 （IEC 61131-3）。
- 也可从另一功能块 （梯级编程语言或结构化文本 （ST）语言）调用功能块（梯级编程语言或结构化文本 （ST）语言）。功能块可嵌套成 8 级。梯级 / ST 语言功能块可自由组合。

1-1-2 功能块规格

未列在下表中的规格，请参阅 CX-Programmer 6.1 版 操作手册 （W437）。

项目		规格
型号		WS02-CXPC1-E-V5 □
安装盘		CD-ROM
兼容 CPU 单元		CS/CJ- 系列 CS1-H、 CJ1-H 和 CJ1M CPU 单元 --3.0 版或以后的版本 单元类型 CPU 类型 ・ CS1G-H CS1G-CPU42H/43H/44H/45H ・ CS1H-H CS1H-CPU63H/64H/65H/66H/67H ・ CJ1G-H CJ1G-CPU42H/43H/44H/45H ・ CJ1H-H CJ1H-CPU65H/66H/67H ・ CJ1M CJ1M-CPU11/12/13/21/22/23 以下 CP- 系列的 CPU 单元可以兼容。 ・ CP1H CP1H-X40*/XA40* 注 如果含功能块 （通过 CX-Programmer5.0 版或以后版本创建的）的用户程序下载至不支持功能块 （CS/CJ- 系列 CPU 单元—2.0 版或更早版本）的 CPU 单元中，则所有的范例均作为非法命令处理。其不能编辑或执行用户程序。 ・NSJ G5D （用于 NSJ5-TQ0 □ -G5D、 NSJ5-SQ0 □ -G5D、 NSJ8-TV0 □ -G5D、 NSJ10-TV0 □ -G5D 和 NSJ12-TS0 □ -G5D） ・FQM1-CMFQM1-CM002 ・FQM1-MMAFQM1-MMA22 ・FQM1-MMPFQM1-MMP22
		CS/CJ/CP 系列功能约束 • 块程序指令 （BPRG 和 BEND）、子程序指令 （SBS、 GSBS、 RET、 MCRO 和 SBN）、跳转指令 （JMP、 CJP 和 CJPN）、步指令 （STEP 和 SNXT）、立即刷新指令 （!）、 I/O 刷新 （IORF）和 1-MS 定时器 （TMHH） 欲知详情，请参阅 2-3 功能块约束 .
兼容计算机 computers	计算机	IBM PC/AT 或兼容
	CPU	133 MHz 奔腾或更快； Windows 98、 98SE 或 NT 4.0 （有 Service Pack 6 或更高）
	OS	Microsoft Windows 95、98、98SE、Me、2000、XP 或 NT 4.0（有 Service Pack 6 或更高）
	存储器	最小 64 兆字节 --Windows 98、 98SE 或 NT 4.0 （有 Service Pack 6 或更高） 请参阅 CX-Programmer Ver.5.0 版操作手册 (W437)
	硬盘空间	可用磁盘空间—最小 100 兆字节
	监视器	最小 SVGA （800 × 600 像素） 注 字体大小—用 “ 小字体 ”。
	CD-ROM 驱动	最少一个 CD-ROM 驱动
	COM 端口	最少一个 RS-232C 端口

项目			规格	
CX-Programmer4.0 版或更早版本不支持的功能 .	定义并创建功能块	功能块定义数量	CS1-H/CJ1-H CPU 单元： • 后缀 -CPU44H/45H/64H/65H/66H/67H：最大 1,024—每个 CPU 单元 • 后缀 -CPU42H/43H/63H：最大 128—每个 CPU 单元 CJ1M CPU 单元： · CJ1M-CPU11/12/13/21/22/23：最大 128—每个 CPU 单元 CP1H CPU 单元 · 所有型号：最大 128- 每个 CPU 单元 NSJ 控制器： · 所有型号：最大 1,024- 每个控制器 FQM1 柔性运动控制器： · FQM1-CM002/MMA22/MMP22：最大 128—每个控制器	
		功能块名称	最大 64 字节	
		变量	变量名	最大 30,000 字节
			变量类型	输入、输出、内部和外部
			功能块 I/O 变量数	最大 64 个 （不包括 EN 和 ENO）
			变量所用的地址分配	自动分配 （用户可以自己设定分配范围）
			实际地址规格	支持
			数组规格	支持 （仅指一维数组）
		语言	可采用梯级编程语言或结构化文本 （ST，见注释）创建功能块。	
	创建实例	实例数	CS1-H/CJ1-H CPU 单元： • 后缀 -CPU44H/45H/64H/65H/66H/67H：最大 2,048—每个 CPU 单元 • 后缀 -CPU42H/43H/63H：最大 256—每个 CPU 单元 CJ1M CPU 单元： • CJ1M-CPU11/12/13/21/22/23：最大 256—每个 CPU 单元 CP1H CPU 单元 • 所有型号：最大 256- 每个 CPU 单元 NSJ 控制器： · 所有型号：最大 2,048- 每个控制器 FQM1 柔性运动控制器： · FQM1-CM002/MMA22/MMP22：最大 256—每个控制器	
		实例名	最大 30000 字符	
	存储功能块作为文件	项目文件	项目文件 （.cxp/cxt）包含了功能块定义和实例。	
		程序文件	文件存储器程序文件 （*.obj）包含了功能块定义和实例。	
		功能块库文件	可以将每个功能块定义存储为单个文件 （.cxf）以便其他项目中使用。	

注 结构化文本 （ST 语言）符合 IEC 61131-3 标准但 CX-Programmer5.0 版只支持赋值语句、选择语句（CASE 和 IF 语句）、迭代语句（FOR、WHILE、REPEAT 和 EXIT 语句）、 RETURN 语句、算术运算符、逻辑算符、比较函数、数值函数和注释。

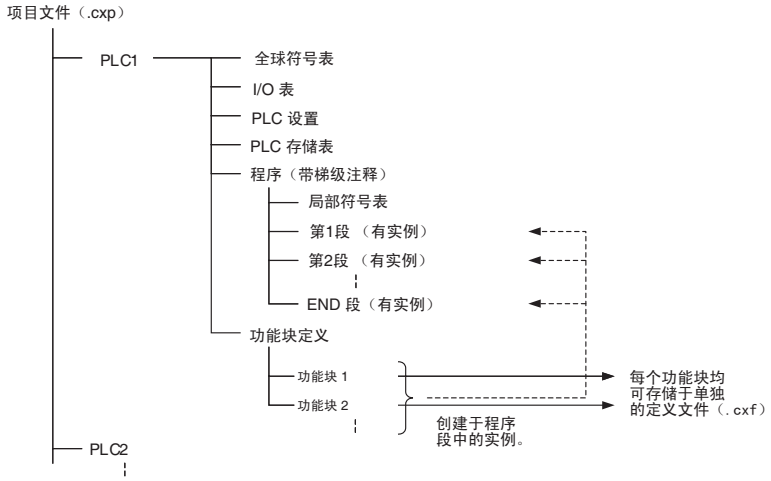
欲知详情，请参阅附录 B 结构化文本 （ST 语言）规格。

1-1-3 用 CX- Programmer 6.0 版创建的文件

工程文件 (*.cxp) 和文件存储器程序文件 (*.obj)

项目文件 （*.cxp）和文件存储器程序文件 （*.obj）使用 CX-Programmer 创建的项目中包含了功能块定义和有实例的项目。这些项目均保存于同一标准项目文件 （*.cxp）和文件存储器程序文件 （*.obj）。

项目内容示于下图。创建功能块定义创建时其目录级可与相关 PLC 目录中程序目录级一样。



功能块库文件 （*.cxf）

采用CX-Programmer6.0版创建的项目功能块定义可以保存为一个可以将定义上载于其他程序中且可反复使用的文件 （1 个定义 = 1 文件）。

注 当功能块嵌套时，所有嵌套 （目的地）功能块定义均包含在此功能块库文件中（.cxf）。

含功能块 （*.cxt）的 项目文本文件

与 CX-Programmer6.0 版创建的项目文件 （*.cxp）相同的数据可以保存为 CXT 文本文件 （*.cxt）。

1-1-4 CX-Programmer 5.0 版 （及以后版本）的功能块菜单

下表列出了 CX- Programmer5.0 版及以后版本中的功能块菜单。所有菜单的详细内容请参阅拿 CX- Programmer. 5.0 版操作手册 （W437）。

主菜单

主菜单	子菜单		快捷	功能
文件	子菜单 功能块	从文件中载入功能块	---	读取保存的功能块库文件 （*.cxf）
		功能块保存于文件中	---	将创建的功能块定义保存于文件中 [功能块库文件 （*.cxf）]。
编辑	更新功能块		---	如在创建实例后修改功能块定义 I/O 变量，则实例左母线显示红色表示出现错误。通过输入新的信息，命令更新实例并清除错误。
	至下层		---	跳转到所选实例的功能块定义。

主菜单	子菜单		快捷	功能
观察	监视 FB 梯级实例		---	当在线监视程序时，监视实例中的梯级程序 I/O 位及状态 （I/O 位监视器）。（只有 CX- Programmer6.0 版和以后版本支持）
	监视 FB 实例		---	当在线监视程序时，监视实例中梯级程序的 ST 变量状态、 I/O 位和字状态 （I/O 位监视器）。（只有 CX- Programmer6.0 版和以后版本支持）
	至下层		---	所选实例的功能块定义内容显示在右侧。（只有 CX- Programmer6.0 版和以后版本支持）
	至上层		---	返回至调用实例 （梯级程序或 ST）。（只有 CX- Programmer6.0 版和以后版本支持）
	窗口	FB 实例浏览器	---	显示 FB 实例浏览器。（当嵌套时，显示器将显示详细的内容，例如，嵌套水平的实例与分配于实例中的变量地址之间的关系）
插入	功能块调用		F	在当前的光标位置处，在程序 （段）中创建实例。
	功能块参数		P	当光标位于输入变量的左面或输出变量的右面时，设置变量输入或输出参数。
PLC	功能块存储器	功能块存储分配	---	设置分配于所选实例变量的地址范围 （功能块实例区域）。
		功能块存储器统计	---	检查分配于所选实例变量的地址状态。
		功能块实例地址	---	检查分配于所选实例的每个变量地址。
		优化功能存储器	---	优化分配于变量变的地址分配情况。

主菜单	子菜单		快捷	功能
工具	模拟	断点 I 设置 / 清除断点	---	设置或清除断点。
		断点 I/ 清除所有断点	---	清除所有断点
		模式 I 运行（监视器模式）	---	进行连续扫描。（梯级执行机构的运行模式设为 MONITOR 模式）。
		模式 I 停止（程序模式）	---	模拟运行模式设为 PROGRAM 模式。
		模式 I 暂停	---	暂停模拟器运行。
		步骤运行	---	模拟单元程序只执行一步。
		步骤运行 I 步骤输入	---	当出现功能块吊调用命令时，该命令开始执行内部程序步骤。
		步骤运行 I 步骤跳出	---	当功能块的内部程序步骤正在执行时，该命令返回至下一高级 （调用源）和暂停执行。
		步骤运行 I 连续步骤运行	---	连续执行程序步骤—执行一定的时间。
		步骤运行 I 扫描运行	---	执行一个周期，暂停执行。
		始终显示当前执行点	---	采用 Step Run 或 Continuous Step Run 命令自动屏幕滚动并一直显示暂停点。
		断点列表	---	显示已设置的断点列表。（操作跳转至指定点）。

主弹出式菜单

功能块定义弹出式菜单

弹出式菜单		功能
插入功能块	梯级	采用梯级编程语言演算法创建功能块定义。
	结构化文本	用 ST 语言演算法创建功能块定义。
	来自文件	从功能块库文件中 （*.cxf）读取功能块定义。

插入的功能块弹出式菜单

弹出式菜单	功能
打开	显示所选功能块定义内容示于窗口右侧。
插入变量	将所选功能块定义保存于文件中。
编辑	编辑所选功能块定义。

功能块变量表弹出式菜单

弹出式菜单		功能
编辑		编辑变量。
插入变量		变量增加到最后一行。
插入变量	上	变量插在当前光标位置上面。
	下	变量插在当前光标位置下面。
剪切		剪切变量。
复制		复制变量。

弹出式菜单	功能
粘贴	粘替变量。
搜索	搜索变量、变量名、变量说明或全部 （正文串） 可以搜索的内容。
替换	替换变量。
删除	删除变量。
重命名	只更改变量名。

实例弹出式菜单

弹出式菜单	功能
编辑	更改变量名。
更新调用	如在创建实例后修改功能块定义 I/O 变量，则实例左母线显示红色表示出现错误。通过输入新的信息，命令更新实例并清除错误。
监视 FB 梯级实例	当在线监视程序时，监视实例中的梯级程序 I/O 位及状态 （I/O 位监视器）。（只有 CX-Programmer6.0 版和以后版本支持）
监视 FB 实例	当在线监视程序时，监视实例中梯级程序的 ST 变量状态、 I/O 位和字状态 （I/O 位监视器）。（只有 CX- Programmer6.0 版和以后版本支持）
注册于观察窗	显示 FB 变量对话框，将所选实例变量注册于观察窗中。
功能块定义	所选实例的功能块定义内容显示在右侧。

快捷键

F 键：显示程序中以前的功能块定义

移动光标至此位置—即，将复制的功能块实例创建于梯级段窗中。点击 F 键。该操作与选择插入 – 功能块调用相同。

P 键：输入参数

光标定位于输入变量的左侧或输出变量的右侧。点击 P 键。该操作与选择插入 – 功能块调用相同。

1-2 功能块

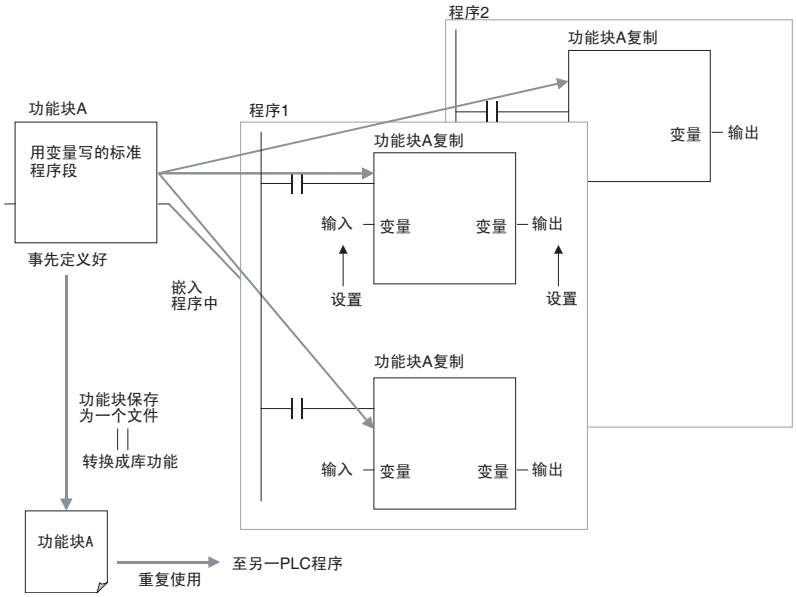
1-2-1 概述

功能块是一个包含标准处理功能的基本程序单元。该标准处理功能事先已定义好。一旦功能块已定义好，用户即可将功能块嵌入程序中，设置 I/O 。这样，即可使用功能。

由于是标准处理功能，因此，功能块不包含实际地址，是变量。用户在变量中设置地址或常数。

这些地址或常数称作参数。变量自身所使用的地址则由 CX- Programmer 自动分配于每个程序。

采用 CX- Programmer 将单个功能块保存为单个文件而且单个功能块还可用于其他 PLC 程序中。因此，标准处理功能可做成库。



1-2-2 功能块优点

功能块便于复杂的编程设备反复使用。一旦在功能块中创建了标准编程并将其保存为文件，便可将功能块嵌入程序中并设置功能块 I/O 参数即可反复使用。当创建 / 调试程序时，反复使用现有功能块的能力讲节省大量的时间并且减少编码错误。此外，使得程序更易于理解。

结构化编程

用功能块创建的结构化程序具有更好的设计质量而所需的开发时间减少。

便于读取的“ 黑匣子 ”设计

I/O 运算域作为程序中的变量名显示。因此，当输入或读取程序时，程序象“ 黑匣子 ”。在理解内部演算法时不会浪费太多的时间。

多程序只使用一块功能块

将标准程序中的参数 （例如，定时器 SVs、控制常数、速度设置和行程）用作输入变量。这样一来，通过一块功能块就很方便地创建许多不同的程序。

减少编码错误

由于反复使用的功能块已经过调试，因此，可以减少编码错误。

数据保护

功能块中的变量不能直接从外面读取。因此，可以保护数据。（数据不能随意修改）。

通过变量编程提高了反复利用率

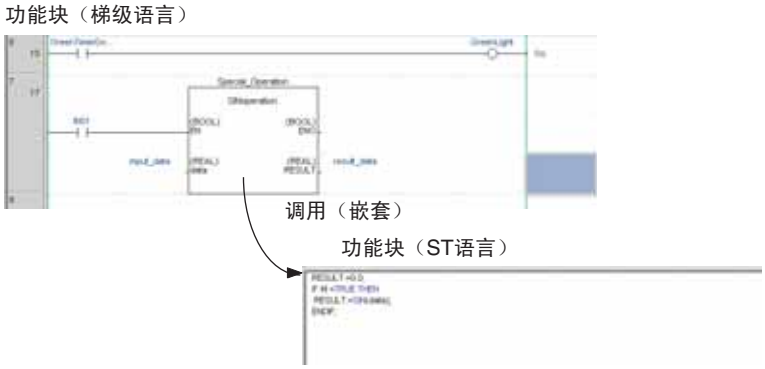
功能块的 I/O 作为变量输入。因此，再使用时无需修改功能块中的数据地址。

创建库

独立可反复使用的程序 （例如，每一步骤的过程、机器、设备或控制系统程序均可保存为功能块定义并转换成库功能。如采用与实际地址无关联的变量名创建功能块，则只要读取功能块定义并将功能块嵌入程序即可很方便地开发新程序。

支持嵌套和多语种

采用结构化文本 （ST）语言输入数学表达式。通过 CX- Programmer6.0 版或以后版本将功能块嵌入程序中。功能块嵌套功能可对嵌入于梯级语言功能块的 ST- 语言功能块进行特殊处理。

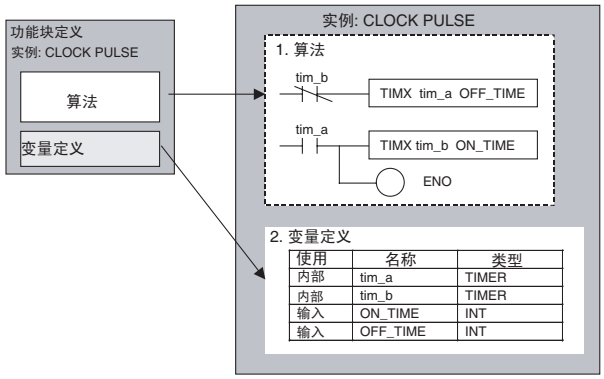


1-2-3 功能块结构

功能块由功能块定义 （事先创建）和嵌入程序中的功能块实例组成。

功能块定义

功能块定义为包含在功能块中的程序。每个功能块定义包含算法和变量定义。算法和变量定义示于下图。



1. 算法

标准化编程采用变量名编写而非实际的 I/O 存储地址。在 CX- Programmer 中，采用梯级编程或结构化文本写算法

2. 变量定义

变量表列出了每个变量用法（输入、输出或内部使用）和属性（数据类型等）。欲知详情，请参阅 1-3 变量。

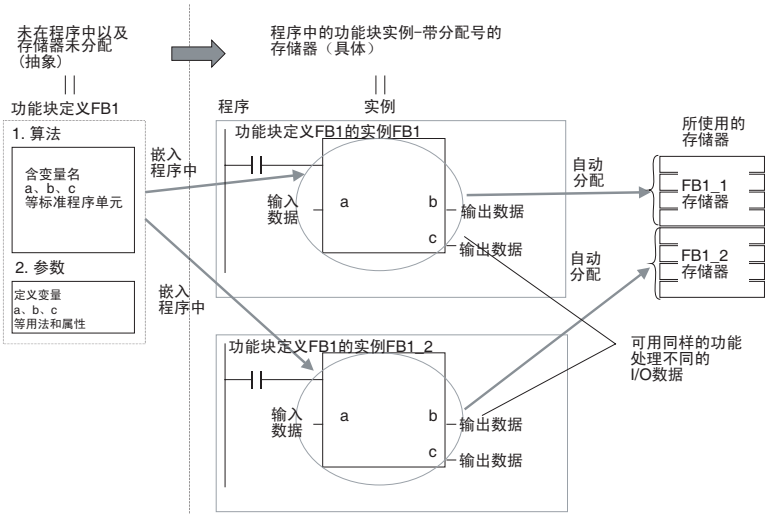
功能块定义数

一个 CPU 单元所创建的功能块定义最大数为 128 或 1,024。具体数取决于 CPU 单元型号。

实例

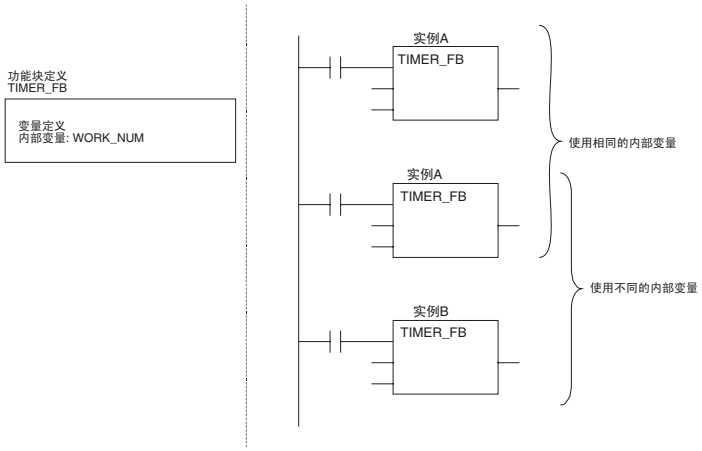
要将实际功能块定义用于程序中，则应创建一份功能块图并将其插入程序中。嵌入程序中的每个功能块定义称作 “ 实例 ” 或 “ 功能块实例 ”。每个实例均有一个标识符—称作 “ 实例名 ”。

要生成实例，则可使用单个功能块定义处理相同功能的不同 I/O 数据。



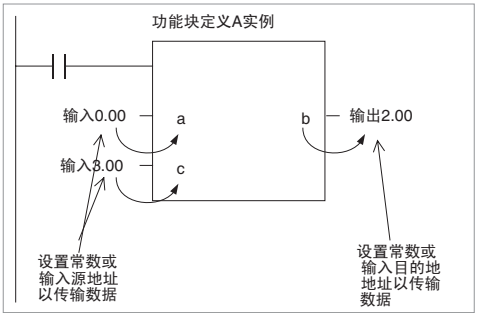
注 实例通过名称来管理。一个以上的具有相同名称的实例也可嵌入程序中。如果有两个或两个以上的实例具有相同名称，则这些实例使用相同的内部变量。不同名称的实例其内部变量也不同。

例如，将定时器用作一个内部变量的多功能块。在这种情况下，所有实例名称均不相同。如果一个以上的实例使用相同的名称，则相同的定时器用于多个位置从而反复使用定时器。但如果内部变量不用或只暂时使用以及仅在下次触发执行一个实例，则相同的实例名可以用来保存存储器。



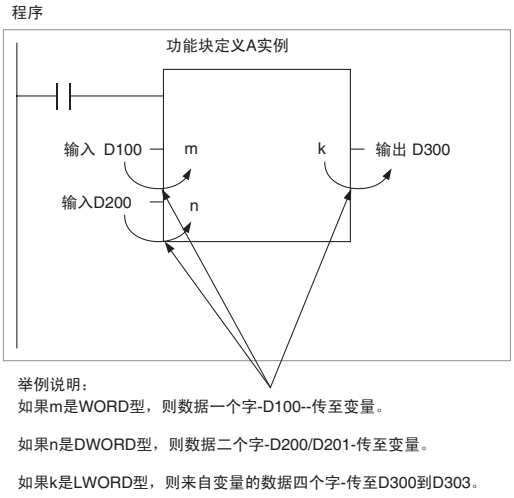
实例数 一个功能块定义可以创建多个实例。根据 CPU 单元的型号，一个 CPU 单元可以创建 256 或 2,048 实例。许可的实例数与功能块数量和实例插入的任务无关。

参数 每当创建一个实例，就需设置 I/O 存储地址或 I/O 变量 （用来传递输入数据值至实例以及从实例中获取输出数据值）常数。这些地址和常数称作参数。



在这里，不是输入源地址但在输入地址常数。其形式和规格大小由通至功能块的变量数据类型来决定。同样，不是输出目的地地址但是输出地址常数。其形式和规格大小由来自功能块的变量数据类型来决定。

即使是输入源地址 （例如，输入参数）或输出目的地地址 （例如，输出参数）是字地址，则数据的形式和规格大小由来自规定字地址的变量数据类型来决定。



注 (1) 以下区域中只有一个地址用作参数：CIO 区域、辅助区域、DM 区域、EM 区域 （存储体从 0 到 C）保持区域和工作区域。

以下内容不能使用：指数和数据寄存器（直接和间接规格）和 DM 区域和 EM 区域的间接地址 （二进制模式和 BCD 模式）。

(2) 用户程序中的局部符号和全局符号也可规定为参数。但要这样做，则局部符号和全局符号的数据大小必须与功能块变量的数据大小一致。

(3) 当执行实例时，在运算处理之前将输入值从参数传至输入变量。在运算处理结束后，输出值从输出变量传至参数中。如必需在运算执行周期内读取或写入数值，则不得将数值传至参数中或从参数上传数值。对内部变量赋值，使用 AT 设置 （规定的地址）。

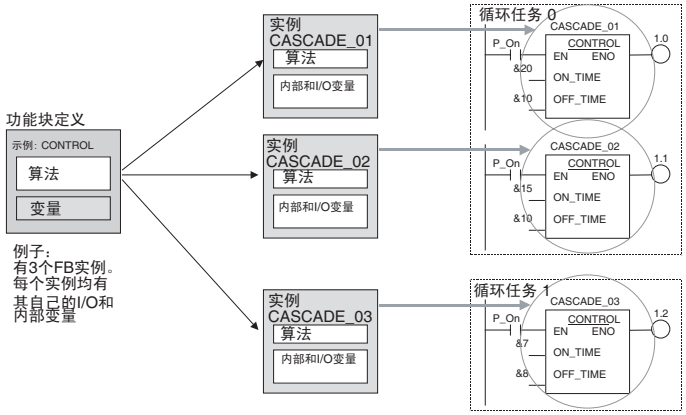
⚠ 注意 如果输入参数中已规定了地址，则地址值传至输入变量。但不能传递实际地址数据。

⚠ 注意 在运算执行周期内，参数不能用来读取或写入数值。使用带 AT 设置的内部变量（规定的地址）。或将全局符号作为外部变量基准。

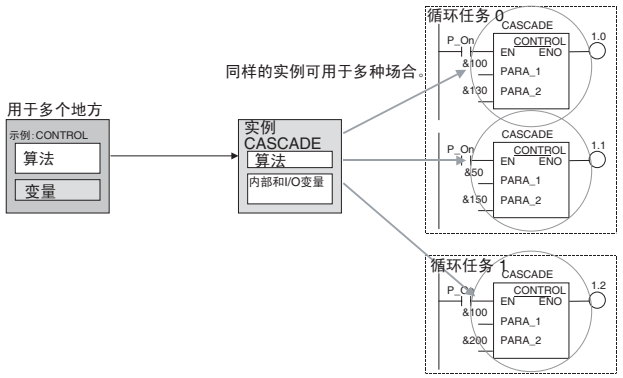
■参考信息

将类似参数的单元 （例如固定值）用作输入变量以及根据每个实例修改传至输入变量的数值从而只需通过单个功能块即可很方便地创建各种程序。

例如：采用 1 个功能块定义创建 3 个实例。



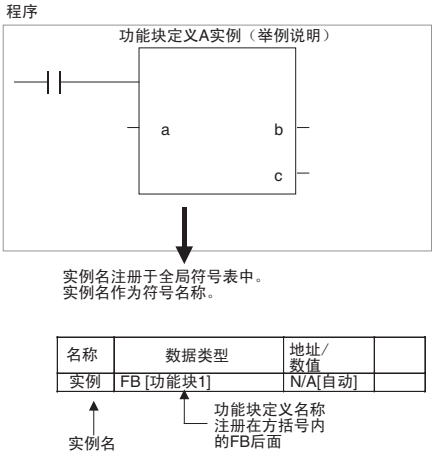
如果不使用内部变量以及处理不受影响或内部变量用于其他地方时，在程序多个地方可以使用相同的实例名。



使用相同存储区域时，需要特别注意。例如，如果含定时指令的相同实例用于一个以上程序时，则会使用相同的定时器号从而导致线圈重复。如两个指令同时执行，则会导致定时器不能正常工作。

实例注册

每个实例名作为一个文件名注册于全局符号表中。

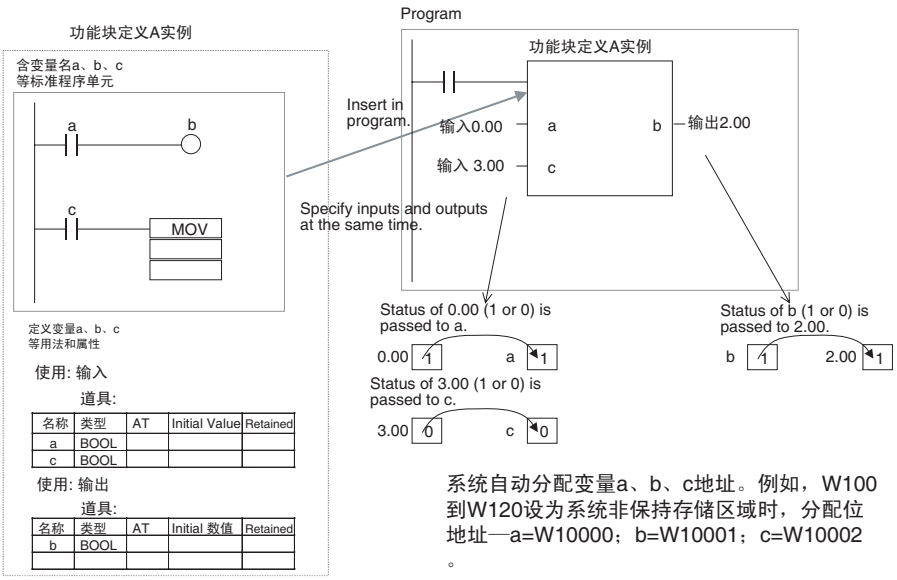


1-3 变量

1-3-1 介绍

在功能块中，地址不作为实际的 I/O 存储地址输入 （见注解）。它们作为变量名输入。每当创建实例时，CX- Programmer 自动将变量所用的实际地址分配在指定的 I/O 存储区域内。此外，用户无需知道功能块实际使用的 I/O 存储地址正如无需知道计算机实际存储器分配。在这种情况下，功能块与子程序不同。例如，功能块使用变量。地址象 “ 黑匣子 ”。

例如：



注 常数不能注册为变量。直接将常数输入至指令操作数。

- 梯级编程语言：在 # 后面输入十六进制数值；在 & 后面输入十进制数。
- 结构化文本 （ST 语言）：在 16# 后面输入十六进制数值；十进制数照原样不动。

例外：直接或间接地将变址寄存器 IR0 到 IR15 指定地址输入指令操作数中；直接将数据寄存器 DR0 到 DR15 输入指令操作数中。

1-3-2 变量用法和属性

变量用法

可以支持以下变量类型 （用法）：

内部： 在实例中只能使用内部变量。这些内部变量不能直接用来传输数据至 I/O 参数或从 I/O 参数中上传数据。

输入： 输入变量可以从实例外的输入参数中输入数据。默认输入变量为传输输入条件数据的 EN （Enable）变量。

输出： 输出变量可以输出数据至实例外的输出参数。默认输出变量为传输实例执行状态的 ENO （Enable Out）变量。

外部： 外部变量是事先由 CX-Programmer 寄存的系统定义变量 （例如，条件标志和一些辅助区域位）或用户定义的全局符号（用于每个实例数中）。

有关变量详细内容，请参阅 2-1-2 功能块单元中变量定义的变量类型 （用法）一节。

下表所示的是可以使用的变量数和为每个变量用法默认创建的变量类型。

1-3-3 变量属性

变量具有以下属性：

变量名

变量名可用来识别功能块中的变量。如果其他功能块也使用了相同的名称，这也没有什么问题。

注 变量名可达 30,000 字符长但不得以数字开始。此外，在一排中，名称不能包含两个下划线字符。字符串不能与变址寄存器相同。例如， IR0 到 IR15。有关其他的限制约束请参阅 2-1-2 功能块单元中变量定义的变量定义。

数据类型

在下面数据类型中，选择其中一个作为变量：

BOOL、 INT、 UINT、 DINT、 UDINT、 LINT、 ULINT、 WORD、 DWORD、 LWORD,

REAL、 LREAL、 TIMER、 COUNTER

有关数据类型的详细情况请参阅 2-1-2 功能块单元中的变量定义。

AT 设置 （分配实际地址）

将变量设于特殊的 I/O 存储地址中而非系统自动分配地址。在这此属性中，用户可输入所需的 I/O 存储地址以规定特殊地址。

该属性仅为内部变量而设。即使设置了具体的地址，变量名还是必须用于算法中。

有关 AT 设置的详细内容请参阅 2-1-2 功能块单元中变量定义。有关 AT 设置使用的详细内容请参阅 2-4-3 内部变量的 AT 设置。

数组设置

变量可作为具有相同属性的单个数据数组来处理。要将变量转换为数组，则需规定数组以及组成数组的最多单元数。该属性仅为内部变量而设。 CX-Programmer5.0 版或以后版本支持一维数组。

- 设置程序
  点击 Advanced 按键，选择 Array Variable 选项，输入最多单元数。
- 当数组变量输入于功能块定义中的算法时，在变量号后面的方括号内输入数组变址号。

有关数组设置的详细内容，请参阅 2-1-2 功能块单元中变量定义。

初始值

在首次执行实例之前，这是设置于变量中的初始值。然后，执行实例时修改数值。

例如，布尔 （BOOL）变量 （位）设为 1 （TRUE）或 0 （FALSE）。WORD 变量设在 0 到 65,535 （0000 到 FFFF 十六进制）值之间。如果未设置初始值，则变量设为 0。例如，布尔变量为 0（FALSE）。WORD 变量为 0000 十六进制。

保持

如果 PLC 再次接通开始运行时需要保持变量数据不变，则选择保持选项。

- 设置程序
  选择保持选项。

1-3-4 变量属性和变量用法

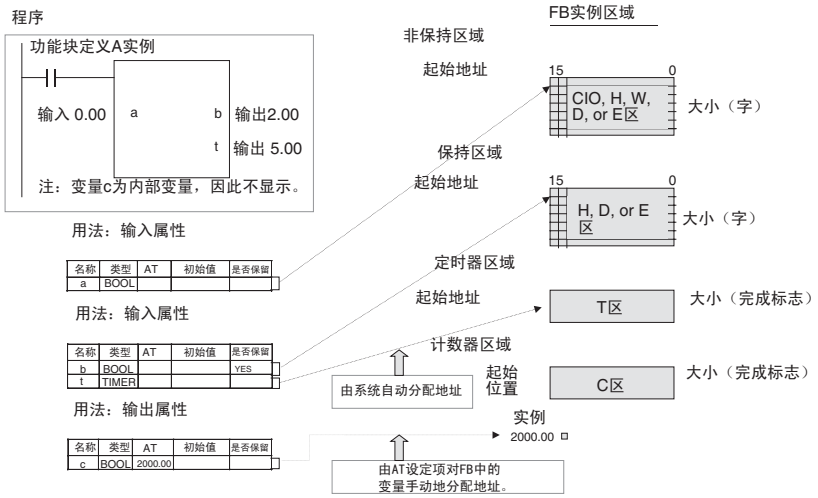
下表所示的是必须设置的、可以设置的以及不能设置的属性 （基于变量用法基础上）：

属性	变量用法		
	内部变量	输入	外部变量
名称	必须设置	必须设置	必须设置
数据类型	必须设置	必须设置	必须设置
AT （指定地址）	可以设置	不能设置	不能设置
初始值	可以设置 .	可以设置 （见注释）	可以设置
保持	可以设置	可以设置 （见注释）	可以设置

注 输入设为初始值但实际输入参数值优先。

1-3-5 变量地址内部分配

当采用功能块定义创建实例时， CX-Programmer 内部分配变量地址。除了通过 AT 设置属性而分配了实际地址的变量之外，所有注册于功能块定义的变量均分配地址。



设置变量内部分配区域 例区域中。

用户可以在系统内部分配的地址处设置功能块实例。系统自动将变量分配于用户所设置的合适实例区域中。

设置程序

在 PLC 菜单中选择 Function Block Memory - Function Block Memory Allocation。在功能块存储器分配对话框中设置这些区域。

功能块实例区域

CS/CJ- 系列 CPU 单元 3.0 版或以后版本和 NSJ 控制器

FB 实例区域	默认值			适用存储区域
	起始地址	终端地址	大小	
Non Retain	H512	H1407	896	CIO, WR, HR, DM, EM
Retain	H1408	H1535	128	HR, DM, EM
Timers	T3072	T4095	1024	TIM
Counters	C3072	C4095	1024	CNT

FQM1 柔性运动控制器

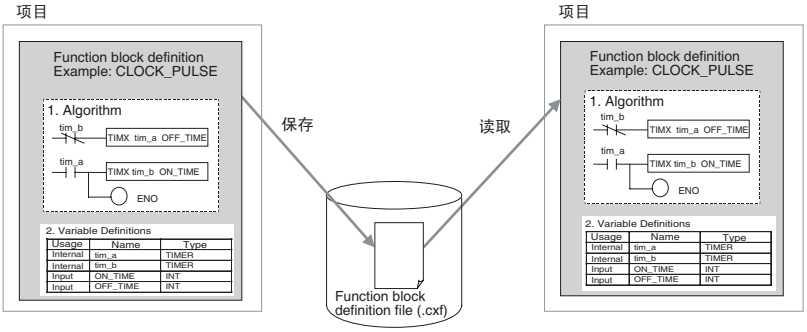
FB 实例区域	默认值			适用存储区域
	起始地址	终端地址	大小	
Non Retain	5000	5999	1000	CIO, WR, DM
Retain	None			
Timers	T206	T255	50	TIM
Counters	C206	C255	50	CNT

功能块保持区域字 （H512 到 H1535）

功能块保持区域字分配在 H512 到 H1535 之间。这些字与程序标准保持区域中的字 （H000 到 H511）是不一样的而且这些字只能用于功能块实例区域 （内部分配的变量区域）。这些字不能作为指令操作数。如果在创建功能块时输入，则显示红色。虽然创建功能块时可以输入字，但在检查程序时会发现字错误。如果该区域仍保留于功能块存储器分配对话框中，则将电源接通 / 切断或在开始操作时清除区域而不保留数值。

1-4 功能块定义转换成功库文件

用 CX-Programmer 创建的功能块定义可以保存为单个文件。该文件以知作为带文件名扩展 *.cxf 的功能块定义文件。这些文件可以在其他项目中使用（PLCs）。



1-5 使用程序

一旦创建了功能块定义以及算法实例，执行时可以调用实例。此外，创建的功能块定义保存为文件。这样，就可用于其他项目中使用 （PLCs）。

1-5-1 创建功能块 / 执行实例

以下程序大致介绍了创建执行功能块所需的步骤。

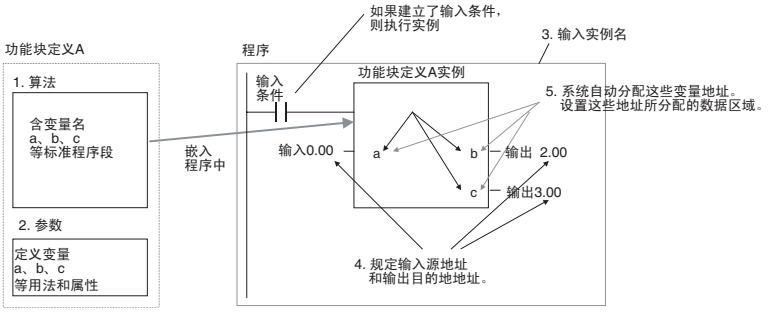
1,2,3...

1. 首先，用梯级程序或 ST 语言创建包含算法和变量定义的功能块。或者，嵌入事先准备好的功能块库文件。

   注 (a) 创建完整的算法 （包括变量名）。

   (b) 当用梯级编程语言输入算法时，通过读取 CX-Programmer5.0 版或更高版本中的项目文件并复制粘贴可用部分从而可以再利用5.0版以前 CX-Programmer 所创建的项目文件。

2. 创建程序时，嵌入全部完整的功能块定义文件。该步骤创建功能块实例。
3. 输入每个实例的实例名。
4. 设置变量输入源地址和 / 或常数，输出目的地地址和 / 或常数作为参数。每个实例数据建立。
5. 选择已创建的实例，从 PLC 菜单中选择 Function Block Memory － Function Block Memory Allocation，设置每种变量的内部数据区域。
6. 程序传输至 CPU 单元中。
7. 开始执行 CPU 单元中的程序。如果输入状态处于 ON 时，调用执行实例。



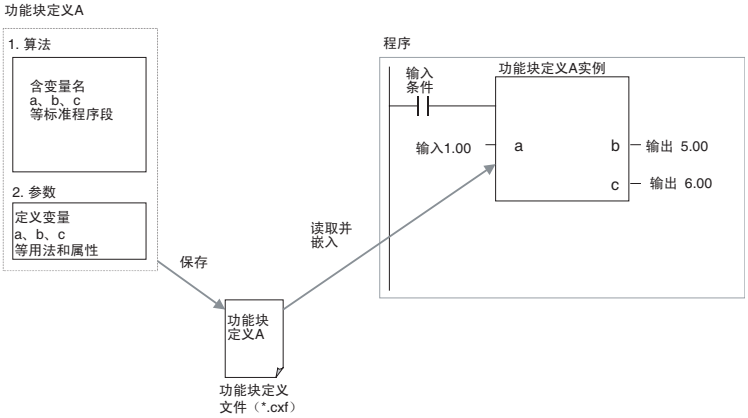
1-5-2 功能块反复使用

采用以下程序将功能块定义保存为文件并用于另一 PLC 程序中。

1,2,3... 1. 选择想保存的功能块，将其保存为功能块定义文件 （*.cxf）。

2. 打开另一 PLC 项目，打开 / 读取已保存的功能块定义文件 （*.cxf）。

3. 创建新程序时，将功能块定义嵌入程序中。



注 在 CX-Programmer 5.0 版中，可将每个功能块定义作为程序来编辑检查。我们推荐编辑程序并在保存或再使用文件之前对每个功能块定义文件进行程序检查。

1-6 版本升级信息

本节大致介绍了 CX-Programmer 功能提高并作为 5.0 版升级到 6.0 版以及 6.0 版升级到 6.1 版的组成部分。

6.0 版升级到 6.1 版信息

适用的 PLC 型号

6.0 版升级到 6.1 版适用于以下 PLC 型号：

- CP- 系列 CP1H （CP1H-XA、 CP1H-X 和 CP1H-Y）

支持 NSJ- 系列 NSJ 控制器

PLC 型号 （“ 设备类型 ”）可以设为 “ NSJ ”。 CPU 类型可以设为 G5D。

支持 FQM1 单元 3.0 版　　目前支持新型的 FQM1 柔性运动控制器 （例如， FQM1-CM002 配位仪模块和 FQM1-MMA22/MMP22 运动控制模块）。

实例 ST/ 梯级程序模拟功能

以前版本 （6.0 版）	最新版本 （6.1 版）
CX- 模拟器可用来执行梯级程序步骤 （Step Run），连续执行步骤 （Continuous Step Run），执行单个循环 （Scan Run）及设置 I/O 断点条件。	执行 Step Run、 Continuous Step Run、 Scan Run 和 Set/Clear Break Point 功能作为 CX-Programmer 功能。所有这些功能与梯级程序和梯级 /ST 程序一起用于功能块中。 注　必须安装 1.6 版本的 CX 模拟器 （另售）以使用这些功能。 注　不可设置 I/O 断开条件。

改进的功能块功能

监视功能块中的 ST 程序

以前版本 （6.0 版）	最新版本 （6.1 版）
在线监控程序时，不能同时监视功能块实例中的 ST 程序操作情况。（只能检查功能块定义程序内容以及监视功能块实例梯形图的 I/O 状态）。	监控程序的同时监视功能块实例中的 ST 程序状态。复核功能块实例或右击实例，在弹出式菜单中选择 Monitor FB Instance 就可监视 ST 程序状态。此时，修改 PV 和强制设置的 / 复位的位。 注　不支持在线编辑。

功能块密码保护

以前版本 （6.0 版）	最新版本 （6.1 版）
设置功能块属性以防显示功能块定义程序。	可以设置以下两种类型的密码保护。 · 密码保护限制读写。 · 密码保护只限制写入。

5.0 版升级到 6.0 版信息

嵌套功能块

以前版本 （5.0 版）	最新版本 （6.0 版）
不能从另一功能块中调用功能块 （不支持嵌套）。	可以从另一功能块中调用功能块 （嵌套）。可支持多达 8 个嵌套级。调用功能块语言和被调用功能块的语言可以是梯级语言或 ST 语言。功能块之间的嵌套级关系可以以目录树格式显示。当功能块嵌套时，调用功能块和被调用 （嵌套）功能块定义只有一个功能块库文件 （.cxf 扩展名）可以保存。

I/O 位监视器支持功能块梯级程序

以前版本 （5.0 版）	最新版本 （6.0 版）
在线监控程序时，不能同时监视功能块实例梯级程序的 I/O 状态。 （只能检查功能块定义中的程序）	在线监控程序的同时监视功能块实例梯级程序的 I/O 状态。复核功能块实例或右击实例，在弹出式菜单中选择 Monitor FB Instance 就可监视 ST 程序状态。此时，修改 PV 和强制设置的 / 复位的位以及近视位差异 （ON/OFF 转换）。 注 不支持在线编辑。不能更改定时器 / 计数器的 SV。

注册监视观察窗中的功能块实例变量

以前版本 （5.0 版）	最新版本 （6.0 版）
必须显示观察窗，双击观察窗并从下拉列表中选择所需的变量来注册功能块实例变量。	功能块实例中的多个变量可以一起注册于观察窗中。 FB 变量注册对话框与以下方法一起显示。变量可以一起注册于对话框中。 • 右击功能块实例，在弹出式菜单中的观察窗内选择 Register。 • 选择程序所需的功能块定义或变量，复制/粘贴或拖/下拉实例至观察窗中。 • 光标移至观察窗中空行，在弹出式菜单中的观察窗内选择 Register。

其他功能块提高内容

- 功能块中的梯级程序支持相互参照弹出功能。
- 从编辑器中的弹出式菜单开始 ST 语言帮助程序。
- 只需双击功能块实例即可打开功能块定义。
- 功能块实例参数输入确认后，光标自动下移。

第 2 章
技术规格

本章提供功能块使用时参考规格。其中包括功能块规格、实例规格、兼容 PLC 规格以及使用注意事项和指南。

2-1 功能块规格

2-1-1 功能块规格

项目	描述
功能块定义数	CS1-H/CJ1-H CPU 单元： ・后缀 -CPU44H/45H/64H/65H/66H/67H：最大 1,024—每个 CPU 单元 ・后缀 -CPU42H/43H/63H：最大 128—每个 CPU 单元 CJ1M CPU 单元： ・CJ1M-CPU11/12/13/21/22/23：最大 128—每个 CPU 单元 CP1H CPU 单元 ・所有型号：最大 128—每个 CPU 单元 NSJ 控制器： ・所有型号：最大 1,024- 每个控制器 FQM1 柔性运动控制器： ・FQM1-CM002/MMA22/MMP22：最大 128—每个控制器
实例数	CS1-H/CJ1-H CPU 单元： ・后缀 -CPU44H/45H/64H/65H/66H/67H：最大 2,048—每个 CPU 单元 ・后缀 -CPU42H/43H/63H：最大 256—每个 CPU 单元 CJ1M CPU 单元： ・CJ1M-CPU11/12/13/21/22/23：最大 256—每个 CPU 单元 CP1H CPU 单元 ・所有型号：最大 256—每个 CPU 单元 NSJ 控制器： ・所有型号：最大 2,048- 每个控制器 FQM1 柔性运动控制器： ・FQM1-CM002/MMA22/MMP22：最大 256—每个控制器
实例嵌套级数	・CX-Programmer 5.0 版：不支持嵌套。 ・CX-Programmer 6.0 版和以后版本：支持可达 8 级的嵌套 （从程序中调用的实例计为一嵌套级。）
I/O 变量数	最多 64 个变量 -- 每个功能块定义。

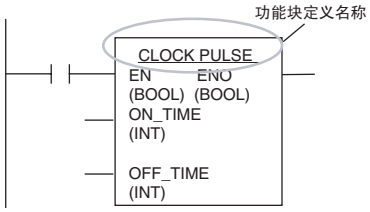
2-1-2 功能块单元

定义功能块时用户必须输入的项目示于下表。

项目	描述
功能块定义名称	功能块定义名称
语言	用于功能块定义的编程语言。选择梯级编程或结构化文本。
变量定义	执行功能块时所要求的变量设置，如操作数和返回值等。 • 变量类型 （用法） • 变量名 • 变量数据类型 • 变量初始值
算法	以梯级或结构化文本输入编程逻辑。 • 用变量输入编程逻辑。 • 无需注册变量即可直接输入常数。
注释	功能块有注释。

功能块定义名称

每个功能块定义有一个名称。名称长度可达 64 个字符。无禁止字符。默认功能块名称为 FunctionBlock@。其中 @ 是数字 （按次序分配）。



语言

选择梯级编程语言或结构化语言 （ST 语言）。

注 (1) 有关 ST 语言的详细内容请参阅附录 B 结构化文本 （ST 语言）规范。

(2) 嵌套时，采用 ST 语言和梯级语言的功能块可以自由组合 （仅指 6.0 版和更高版本）。

变量定义

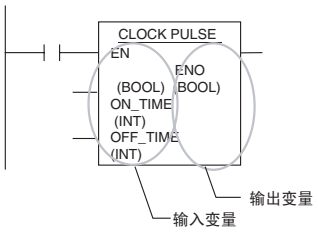
定义用于功能块定义的操作数和变量。

变量名

- 变量名可达 30,000 字符长。
- 变量名中不能有空格或以下的任何字符：
  ！“ # $ % & ‘ （） = - ~ ^ \ | ‘ @ { [+ ; * : }] < ， > . ? /
- 变量名不能以数字开头 （0~9）。
- 变量名一行不能有 2 个下划线字符。
- 以下字符不能用来表示 I/O 存储器中的地址。

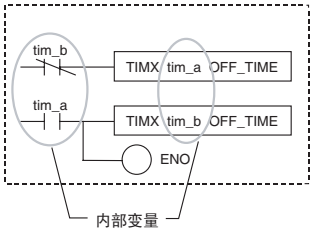
  数字值 （字地址）后面所带的 A、 W、 H （或 HR）、 D （或 DM）、 E （或 EM）、 T （或 TIM）、 C （或 CNT）。

输出变量



变量表

用途	名称	类型
内部	tim_a	TIMER
内部	tim_b	TIMER
输入	ON_TIME	INT
输入	OFF_TIME	INT



变量类型 （用法）

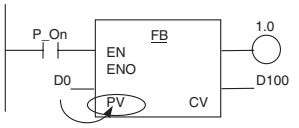
项目	变量类型			
	输入	输出	内部	外部
定义	操作数到实例	实例返回值	仅用于实例的变量	事先通过 CX-Programmer 注册为变量的全局符号或用户自定义全局符号
下一执行的数值状态	数值不传递至下一执行中。	数值传递至下一执行中。	数值传递至下一执行中。	数值不传递至下一执行中。
显示	示于实例左侧。	示于实例右侧。	不显示	不显示
许可数	每个功能块最多 64 位数 （不包括 EN）	每个功能块最多 64 位数 （不包括 EN）	无限制	无限制
AT 设置	无	无	支持	无
数组设置	无	无	支持	无
保持设置	无	支持	支持	无
默认创建的变量	EN （Enable）：接收输入条件	ENO （Enable Output）：输出功能块执行状态	无	注册于 CX-Programmer 作为变量的预定义符号。例如，条件标志和一些辅助区域位。

注　　有关外部变量的详细情况请参阅附录 C 外部变量。

■输入变量

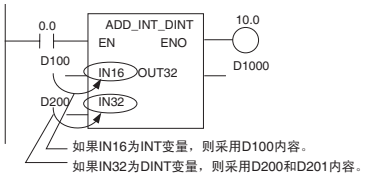
输入变量将外部操作数传递至实例中。输入变量示于实例左侧。

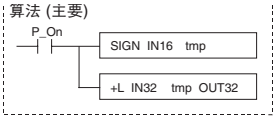
输入源值 （在调用实例之前包含在指定参数中的数据）传递至输入变量中。



规定为输入（值D0）的参数值传递至实例输入变量（PV）中。

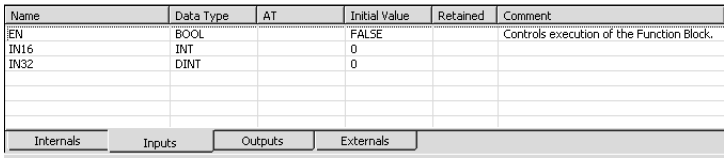
举例说明





变量一览表

用法	名称	类型
内部	tmp	DINT
输入	EN	BOOL
输入	IN16	INT
输入	IN32	DINT
输出	ENO	BOOL
输出	OUT32	DINT



注 1. 相同的名称不能同时分配于输入变量和输出变量。如果输入变量和输出变量为相同变量，则用不同名称注册变量并通过指令（例如 MOV）将输入变量值传送至功能块中的输出变量。

2. 执行实例时，在处理运算之前，输入值从参数传至输入变量。因此，在运算中，不能从参数中读取数值作为输入变量。如果在运算执行周期内必须读取数值，则不得从参数中传输数值。赋值给内部变量。使用 AT 设置（规定的地址）。或者，将全局符号作为外部变量。

初始值

设输入变量初始值但要启动输入参数值（输入变量 EN 的输入参数值处于 ON，执行实例）。

注 使用 CX-Programmer 时，不能省略输入参数设定值。

EN (Enable) 变量

创建输入变量时，默认输入变量为 EN 变量。

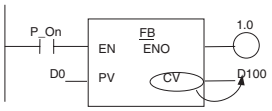
当输入变量 EN 参数处于 ON 时，执行实例。

■输出变量

输出变量将实例返回值传输至外部应用。

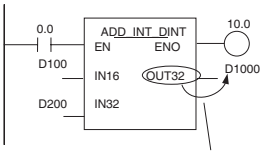
输出变量示于实例右侧。

执行实例后，输出变量值传至指定的参数中。

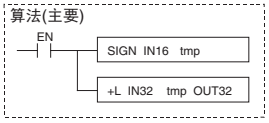


输出变量值（CV）传至指定为输出目的地的参数中。
在本例中，为D100。

举例说明

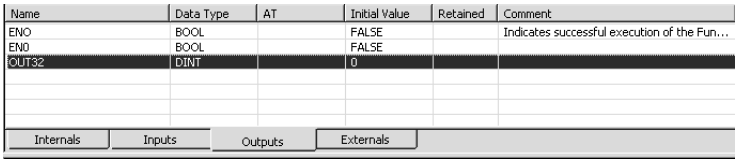


OUT32为DINT变量。因此，变量值传至D1000和D1001。



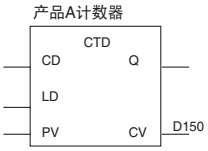
变量一览表

用法	名称	数据类型
内部	tmp	DINT
输入	EN	BOOL
输入	IN16	INT
输入	IN32	DINT
输出	ENO	BOOL
输出	OUT32	DINT



如内部变量一样，保持输出变量值直到下次执行实例（例如，EN 处于 OFF 时，保持输出变量值）。

举例说明：
在下例中，输出变量值 CV 一直保持直到下次执行实例。



注 1. 相同的名称不能同时分配于输入变量和输出变量。
如果输入变量和输出变量为相同变量，则用不同名称注册变量并通过指令（例如 MOV）将输入变量值传送至功能块中的输出变量。

2. 执行实例时，在处理运算之后，输入变量传至对应的参数中。因此，在运算中，不能将输出变量中的数值写入参数中。如果在运算执行周期内必须写入数值，则不得将数值写入参数中。赋值给内部变量。使用 AT 设置（规定的地址）。

初始值

设置未保持的输出变量初始值。例如，不选 Retain Option。如果选择 Retain Option，就不能设置输出变量初始值。

如果 IOM 保持位 （A50012）处于 ON，初始值不能写入输出变量中。

辅助区域控制位		初始值
IOM 保持位 （A50012）	ON	不设置初始值。

ENO （Enable Output）变量

ENO 变量创建为默认输出变量。当调用实例时，ENO 输出变量打开处于 ON 状态。用户可以修改该值。ENO 输出变量可以用作标记来检查实例执行是否正常完成。

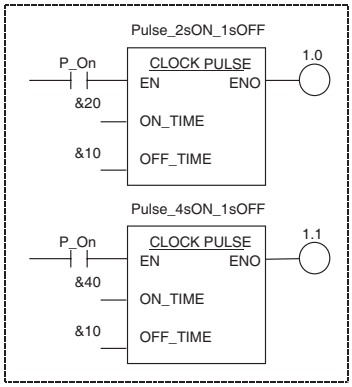
■内部变量

在实例内使用内部变量。对每个实例而言，这些变量为内部变量。不能从实例外将这些变量作为参考标准而且这些变量不显示于实例中。

内部变量一直保持直到下次执行实例 （例如， EN 处于 OFF 状态时，内部变量值保持不变）。因此，即使以相同 I/O 参数执行相同功能块定义实例，其结果也不一定一样。

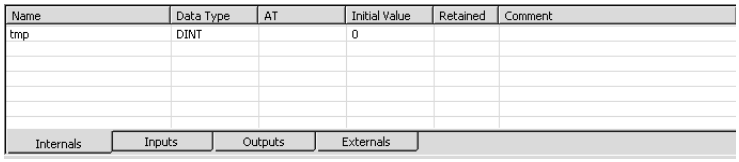
举例说明：

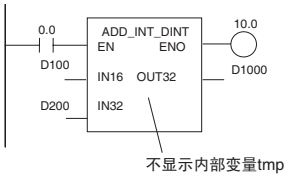
内部变量 tim_a in instance Pulse_2sON_1sOFF 与内部变量 tim_a in instance Pulse_4sON_1sOFF 不一样。因此，实例不能作为参考而且不影响相互的 tim_a 值。

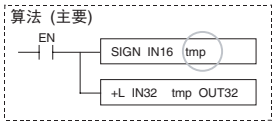


变量一览表

用法	名称	数据类型
内部	tim_a	TIMER
内部	tim_b	TIMER
输入	ON_TIME	INT
输入	OFF_TIME	INT







变量一览表

	名称	数据类型
内部	tmp	DINT
输入	EN	BOOL
输入	IN16	INT
输入	IN32	DINT
输出	ENO	BOOL
输出	OUT32	DINT

通过断电和开始操作保持数据

内部变量保持最后调用实例的值。此外，当断电或开始操作（PROGRAM 模式切换至 RUN 或 MONITOR 模式）时，可以选择 Retain Option 以便内部变量也保持其值不变。

当选择 Retain Option 时，一旦断电或开始操作时变量值即被保持不变—除非 CPU 单元无备用电池。如果 CPU 单元电池质量不佳，则变量值不稳定。

变量	条件	状态
变量设在 Retain	开始操作	保持
	通电	保持

如果未选 Retain Option，则当断电或开始操作时变量值不能被保持。如下表所示，即使变量未设为保持，但在开始操作时使 IOM 保持位（A50012）处于 ON 状态也可保持变量值。在断电期间，设置 PLC Setup 保持变量值。

变量	条件	IOM 保持位（A50012）设置		
		OFF	ON	
			启动时选择的 IOM 保持位（PLC Setup）状态	启动时未选的 IOM 保持位（PLC Setup）状态
未设在 Retain 的变量	开始操作	未保持	保持	保持
	通电	未保持	保持	未保持

注 IOM 保持位 （A50012）支持与以前型号兼容。

但要使变量值保持在功能块中不变，则使用 Retain Option 而非 IOM 保持位。

初始值

设置未保持的内部变量初始值。例如，不选 Retain Option。如果选择 Retain Option，就不能设置内部变量初始值。

不保持的内部变量初始化为 0。

如果 IOM 保持位 （A50012）处于 ON，初始值不能写入输出变量中。

辅助区域控制位		初始值
IOM 保持位 （A50012）	ON	不设置初始值。
	OFF	设置初始值。

■外部变量

外部变量是事先由 CX-Programmer 注册的系统定义变量或全局符号表中用户自定义变量的外部参考变量。

- 有关系统定义变量的详细情况，请参阅附录 C 外部变量。
- 如要将用户自定义变量作为全局符号表中参考变量，则必须采用与外部变量相同的变量名和数据类型将变量注册于全局符号表中。

变量属性

变量名

变量名用来识别功能块中的变量。变量名长度可达 30,000 个字符。相同的变量名可以用于其他功能块中。

注 即使变量带 AT 设置 （规定的地址），也必须输入变量名。

数据类型

可以使用以下类型的任何一个：

数据类型	内容	大小	输入	输出	内部
BOOL	位数据	1 位	可以	可以	可以
INT	整数	16 位	可以	可以	可以
UNIT	无符号整数	16 位	可以	可以	可以
DINT	双整数	32 位	可以	可以	可以
UDINT	无符号双整数	32 位	可以	可以	可以
LINT	长 （4 个字）整数	64 位	可以	可以	可以
ULINT	无符号长 （4 个字）	64 位	可以	可以	可以
WORD	16 位数据	16 位	可以	可以	可以
DWORD	32 位数据	32 位	可以	可以	可以
LWORD	64 位数据	64 位	可以	可以	可以
REAL	实数	32 位	可以	可以	可以
LREAL	长实数	64 位	可以	可以	可以
TIMER	定时器 （见注 1）	标记：1 位 PV：16 位	不支持	不支持	可以
COUNTER	计数器 （见注 2）	标记：1 位 PV：16 位	不支持	不支持	可以

注 (1) TIMER 数据类型用来将定时器号变量（0 到 4095）输入 TIMER 指令操作数（TIM 和 TIMH 等）。当该变量用于另一指令中时，对定时器完成标记（1 位）或定时器当前值（16 位）进行规定（取决于指令操作数）。TIMER 数据类型不能用于结构化文本功能块。

(2) COUNTER 数据类型用来将计数器号变量（0 到 4095）输入 COUNTER 指令操作数 （CNT 和 CNTR 等）。当该变量用于另一指令中时，对计数器完成标记 （1 位）或计数器当前值 （16 位）进行规定 （取决于指令操作数）。 COUNTER 数据类型不能用于结构化文本功能块。

AT 设置 （实际地址分配）

在内部变量中，将变量设为特殊 I/O 存储地址而不是系统自动分配地址。

在此属性中，用户可输入所需的 I/O 存储地址以规定特殊地址。

即使设置了具体的地址，变量名还是必须用于算法中。

注 (1) AT 属性仅为内部变量而设。

(2) AT 设置只能与 CIO （核心 I/O 区域）、 A （辅助区域）、D （数据存储区）、E（扩展存储区）和 H（保持中继区域）以及 W （内部中继区域）。不能在以下存储区中设置 AT 属性：

- 变址寄存器和数据寄存器区域 （直接 / 间接规定）
- 间接规定 DM/EM （：二进制模式； *：BCD 模式）

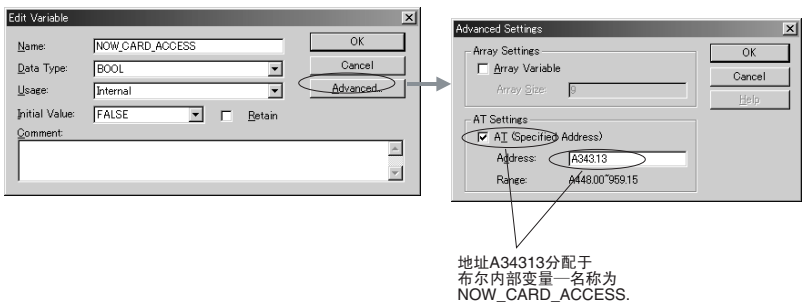
(3) AT 设置可用于以下分配：

- 基本 I/O 单元、 CPU 总线单元或特殊 I/O 单元的地址；
- 事先未注册为外部变量的辅助区域位；
- 网络中其他节点的 PLC 地址；
- 规定多字中的起始字 （或结束字）的指令操作数。

举例说明：

如果 READ DATA FILE 指令 （FREAD）正用于功能块定义而且必须检查文件存储操作标记 （A34313），则使用内部变量并指定 AT 设置中的标记地址。

注册内部变量，选择 AT 设置选项，指定 A34313 为内部变量地址。通过该内部变量检查文件存储器操作标记状态。



当使用 AT 设置时，功能块失去其灵活性。因此，只有在必要时才使用该功能。

数组设置

在内部变量中，其中一个变量定义为数组。

注 CX-Programmer 只支持一维数组。

在数组设置中，只需注册一个变量即可使用大量的具有相同属性的变量。

- 数组由 1 到 32,000 个数组元素组成。
- 仅对内部变量进行设置数组。
- 只要是内部变量，任何数据类型均可用于数组变量。
- 当将数组变量名输入于功能块定义算法中时，在变量号后面的方括号内输入数组变址号。可以采用以下三种方法来规定变址。（在这种情况下数组变量为 a[]。）
- 直接用数字 （用于梯级或 ST 语言编程）

  举例说明：a[2]
- 用变量 （用于梯级或 ST 语言编程）
  举例说明：a[n] ；其中 n 是变量。

  注 INT、DINT、LINT、UINT、UDINT 或 ULINT 可用作变量数据类型：
- 有方程式 （仅用于 ST 语言编程）
  举例说明：a[b+c]，其中， b 和 c 为变量。

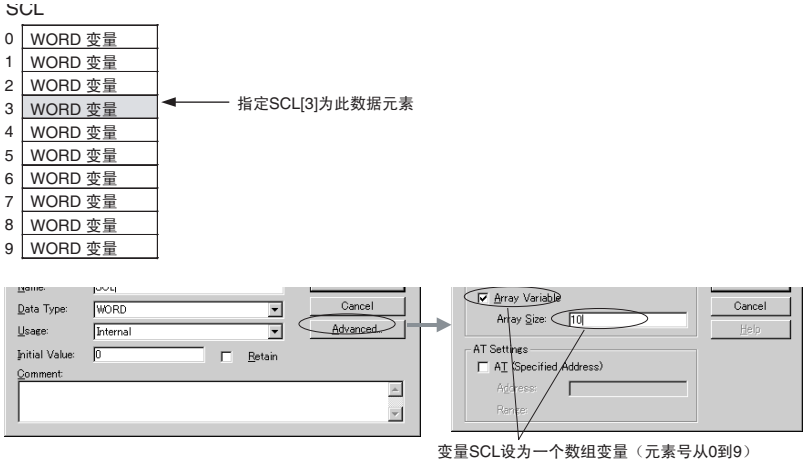
  注 方程式只包含算术运算符 （+、 .、 * 和 /）。

一个数组是由一组相同类型的数据项组成。每个数组元素具有相同的变量名和唯一的变址。（变址表示数组中元素的位置）。

一维数组是一个只有一个变址号的数组。

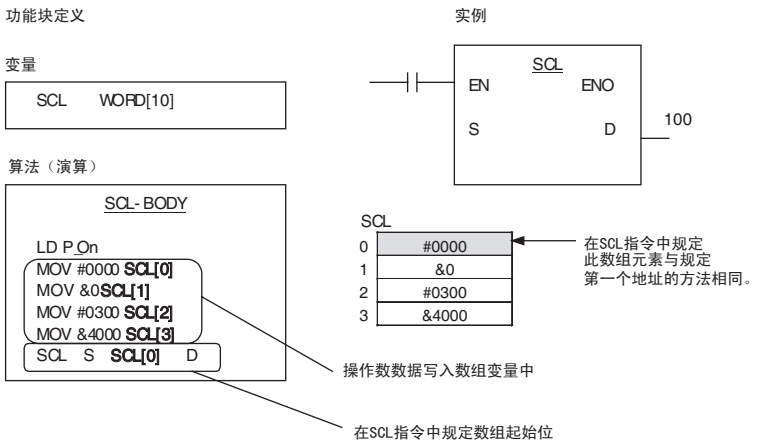
举例说明：当一个名为 SCL 的内部变量设为一个有 10 个元素的数组变量时，以下 10 个变量可以使用：

SCL[0]、SCL[1]、SCL[2]、SCL[3]、SCL[4]、SCL[5]、SCL[6]、SCL[7]、SCL[8] 和 SCL[9]。



注 如果有 AT 属性的内部变量不能设为操作数以及不能设置外部变量的话，在这种情况下，此时如果要在指令操作数中规定多字的第一或最后一个字以能再使用功能块，则使用内部数组变量。准备一个含所需大小的元素数的数组变量。在每个数组元素中设置了数据后，规定操作数数组变量的第一个或最后一个元素。

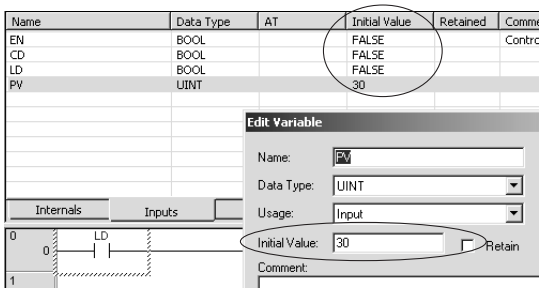
举例说明：



注 详细情况，请参阅 2-5 规定多字中第一个字或最后一个字的操作数指令注意事项。

初始值

当首次执行实例时，设置输入变量、内部变量和输出变量的初始值。详细情况请分别参阅前面所述的输入变量、内部变量和输出变量中的初有关初始值内容。



通过断电和开始操作保持数据

通过断电和开始操作保持数据。当选择 Retain Option 时，变量分配于存储区内。当断电和 PLC 操作开始时保持该存储区。

算法

用注册变量输入逻辑编程。

操作数输入限制

在功能块中，地址不能直接输入指令操作数中。直接输入的地址将作为变量名处理。

注 特例：直接或间接地将变址寄存器 IR0 到 IR15 指定地址输入指令操作数中；直接将数据寄存器 DR0 到 DR15 输入指令操作数中。

常数直接输入指令操作数。

- 梯级编程语言。在 # 后面输入十六进制数值；在 & 后面输入十进制数。

- 结构化文本（ST 语言）：在 16# 后面输入十六进制数值；十进制数照原样不动。

注释

可以输入多达 30,000 长字符的注释。

2-2 实例规格

2-2-1 实例组成部分

下表列出了注册实例时用户必须设置的项目。

项目	说明
实例名	实例名
语言变量定义	编程和变量与功能块定义一样
功能块实例区	变量所用的地址范围
注释	输入每个实例的注释

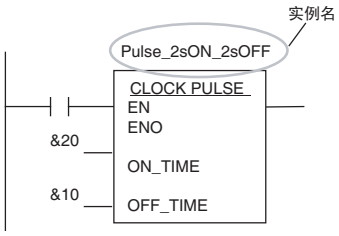
实例名

这是实例名。

- 实例名可达 30,000 个字符长。
- 实例名不能包含空格或以下字符中的任何一个：
  ! “ # $ % & ‘ （） = - ~ ^ \ | ‘ @ { [+ ; * :] } < ， > . ? /
- 实例名不能以数字 （0 到 9）开始。

无其他限制。

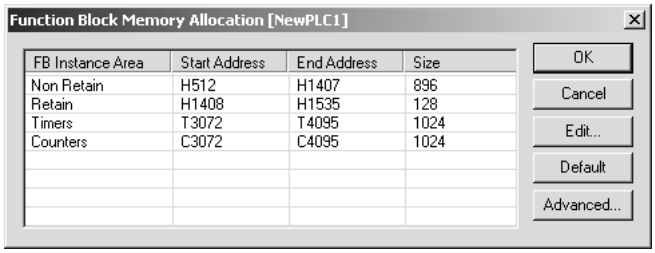
实例名示于实例上面。



功能块实例区域

系统需要存储器来存储实例内部变量和 I/O 变量从而可以使用功能块。这些区域作为功能块实例区域，用户必须定义第一个地址和大小。第一个地址和区域大小规定为 1 个字单位。

当 CX-Programmer 编辑功能时，如果有任何用户程序存取这些区域中的字会输出错误。



CS/CJ- 系列 CPU 单元 3.0 版或以后版本和 NSJ 控制器

FB 实例区域	默认值			适用存储器区域
	起始地址	终端地址	大小	
非保持	H512	H1407	896	CIO, WR, HR, DM, EM
保持	H1408	H1535	128	HR, DM, EM
定时器	T3072	T4095	1024	TIM
计数器	C3072	C4095	1024	CNT

FQM1 柔性运动控制器

FB 实例区域	默认值			适用存储器区域
	起始地址	终端地址	大小	
非保持	5000	5999	1000	CIO, WR, DM
保持	无			
定时器	T206	T255	50	TIM
计数器	C206	C255	50	CNT

功能块实例区域类型

在功能块实例区域中进行以下设置：

CS/CJ- 系列 CPU 单元 3.0 版或以后版本

非保持区域

项目	内容
分配的变量	在断电和操作开始时保持属性的变量设为非保持变量。（见注 1）
应用区域	H （功能块特殊保持区域）、 I/O （CIO 区域）、 H （保持区域）、 W （内部中继区域）、 D （数据存储器区域）（见注 2）、 E （扩展数据存储器区域）（见注 2 和 3）。
设置单位	设为字
分配字 （默认）	H512 到 H1407

注 (1) 当数据类型设为 TIMER 或 COUNTER 时除外。

(2) 即使DM或EM区域规定为非保持区域或保持区域，但仍可以存取位数据。

(3) 如果 EM 区域规定为非保持区域或保持区域的话，则相同的存储体名不能规定为用户程序中当前存储体。

保持区域

项目	内容
分配的变量	在断电和操作开始时保持属性的变量设为非保持变量。（见注 1）
应用区域	H （功能块特殊保持区域）、 I/O （CIO 区域）、 H （保持区域）、 W （内部中继区域）、 D （数据存储器区域）（见注 2）、 E （扩展数据存储器区域）（见注 2 和 3）。
设置单位	设为字
分配字 （默认）	H1408 到 H1535

注 (1) 当数据类型设为 TIMER 或 COUNTER 时除外。

(2) 即使DM或EM区域规定为非保持区域或保持区域，但仍可以存取位数据。

(3) 如果 EM 区域规定为非保持区域或保持区域的话，则相同的存储体名不能规定为用户程序中当前存储体。

定时器区域

项目	内容
分配的变量	TIMER 设为数据类型的变量
应用区域	T （定时器区域）定时器标记 （1 位）或定时器 PV （16 位）
分配字 （默认）	T3072 到 T4095 定时器标记 （1 位）或定时器 PV （16 位）

Counter Area

项目	内容
分配的变量	COUNTER 设为数据类型的变量
应用区域	C （计数器区域）定时器标记 （1 位）或定时器 PV （16 位）
分配字 （默认）	C3072 到 C4095 定时器标记 （1 位）或定时器 PV （16 位）

功能块保持区域 （H512 到 H1535）

设为保持和非保持字的功能块保持区域字默认分配为 H512 到 H1535。这些字与用于程序的标准保持区域 （H000 到 H511）不同而且这些字只能用于功能块实例区域 （内部分配变量区域）。

- 在内部变量 AT 设置中不能指定这些字。
- 这些字不能规定为指令操作数。
  - 如果未创建功能块时输入这些字，则这些字显示为红色。
  - 当创建功能块时虽然可以输入这些字，但在检查程序时会出现错误。
- 如果该区域指定为非保持区域，则当开始操作时接通电源 / 断电或清除区域而此时数值不被保留。

注 为防止实例区域地址与程序地址重叠，非保持区域和保持区域的地址设在 H512 到 H1535 之间 （功能块保持区域字）。

FQM1 柔性运动控制器

FB 实例区域	默认值			适用存储器区域
	起始地址	终端地址	大小	
非保持	5000	5999	1000	CIO, WR, DM
保持	无			
定时器	T206	T255	50	TIM
计数器	C206	C255	50	CNT

非保持区域

项目	内容
分配的变量	在断电和操作开始时保持属性的变量设为非保持变量。（见注 1）
应用区域	I/O （CIO 区域）、 W （内部中继区域）和 D （数据存储器区域）（见注 2）
设置单位	设为字
分配字 （默认）	CIO5000 到 CIO5999

注 (1) 当数据类型设为 TIMER 或 COUNTER 时除外。

(2) 即使DM或EM区域规定为非保持区域或保持区域，但仍可以存取位数据。

保持区域

无

定时器区域

项目	内容
分配的变量	TIMER 设为数据类型的变量
应用区域	T （定时器区域）定时器标记 （1 位）或定时器 PV （16 位）
分配字 （默认）	T206 到 T255 定时器标记 （1 位）或定时器 PV （16 位）

计数器区域

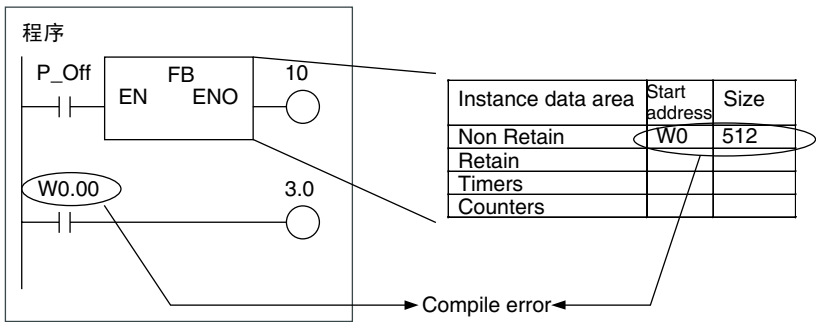
项目	内容
分配的变量	COUNTER 设为数据类型的变量
应用区域	C （计数器区域）定时器标记 （1 位）或定时器 PV （16 位）
分配字 （默认）	C206 到 C255 定时器标记 （1 位）或定时器 PV （16 位）

从用户程序中存取功能块实例区域

如果用户程序中含有指令可以存取功能块实例区域。则如果尝试以下操作时错误会示于 CX-Programmer 的 Output Window 的 Complie Tab 中。

- 试图在线编辑时写入 （不能写入）
- 执行程序检查（在程序菜单中选择Compile 或在PLC菜单中选择Compile All PLC Programs）

举例说明：如果 W0 到 W511 指定为功能块实例区域的非保持区域而 W0.00 用于梯级程序中，则在编辑时会发生错误并显示为“ ERROR: [omitted]...- Address -W0.00 is reserved for Function Block use]。



注 当增加或删除变量时，功能块实例区域中变量分配将自动再分配。单个实例按顺序获得地址。但如果不能按顺序获得地址，则所有变量分配到的地址均不相同。因此，会创建不用区域。如果出现这种情况，则执行优化操作从而有效使用分配区域并删除不用区域。

注释

可以输入多达 30,000 长字符的注释。

创建多个实例

调用相同实例

可从多个地方调用一个实例。在这种情况下，实例变量共享。

制作多个实例

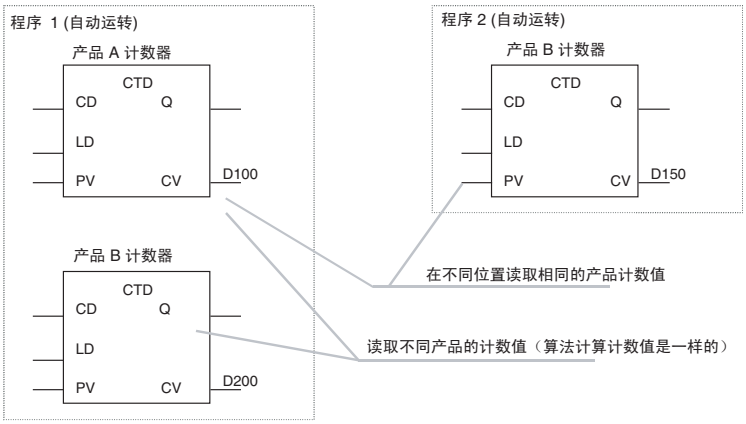
可从单个功能块定义中创建多个实例。在这种情况下，每个实例的内部变量值均不相同。

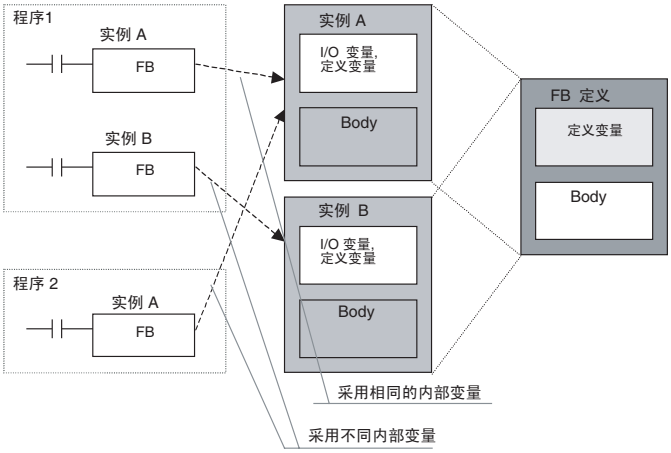
举例说明：计算产品 A 和产品 B

准备一块称作下计数器的功能块定义（CTD）安装产品 A 和产品 B 的计算、数器。有两种程序。一种程序是自动操作；另一种是手动操作。用户可以自己切换合适的操作模式。

在这种情况下，从单个功能块中创建多个实例。

必须从多个地方调用相同实例。





2-2-2 参数规格

由用户在输入参数和输出参数中设置的数据如下：

项目	应用区域
输入参数	数值 （见注 1）、地址和程序符号 （全局符号和局部符号）（见注 2） 注 从参数中传至输入变量中的数据为输入变量数据实际值。（即使在参数中设置了地址，但地址本身不传输）。 注 必须设置输入参数。如果有一个输入程序未设置，则会出现致命错误。输入参数不传输至实际的 PLC 中。
输出参数	地址和程序符号 （全局符号和局部符号）（见注 2）

注 (1) 数值输入于参数的方法示于下表。

输入变量数据类型	内容	大小	参数值输入方法	设置范围
BOOL	位数据	1 位	P_Off, P_On	0 (FALSE), 1 (TRUE)
INT	整数	16 位	正值：& 或 + 后面跟整数； 负值：- 后面跟整数；	-32,768 to 32,767
DINT	双整数	32 位		-2,147,483,648 to 2,147,483,647
LINT	长 （8 字节）	64 位		-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
UINT	无符号整数	16 位	正值：& 或 + 后面跟整数；	&0 to 65,535
UDINT	无符号双整数	32 位		&0 to 4,294,967,295
ULINT	无符号长 （8 字节）	64 位		&0 to 18,446,744,073,709,551,615
REAL	实数	32 位	正值：& 或 + 后面跟实数（有十进制小数）； 负值：- 后面跟整数 （有十进制小数）；	-3.402823×10^{38} ～ $-1.175494 \times 10^{-38}$, 0, 1.175494×10^{-38} ～ 3.402823×10^{38}
LREAL	长实数	64 位		$-1.79769313486232 \times 10^{308}$ ～ $-2.22507385850720 \times 10^{-308}$, 0, $2.22507385850720 \times 10^{-308}$, $1.79769313486232 \times 10^{308}$
WORD	16- 位数据	16 位	# 后面跟十六进制数 （最多 4 位数）； & 或 + 后面跟实数（有十进制小数）；	#0000 ～ FFFF 或 &0 ～ 65,535
DWORD	32- 位数据	32 位	# 后面跟十六进制数 （最多 8 位数）； & 或 + 后面跟实数（有十进制小数）；	#00000000 ～ FFFFFFFF 或 &0 ～ 4,294,967,295
LWORD	64- 位数据	64 位	# 后面跟十六进制数 （最多 16 位数）； & 或 + 后面跟实数（有十进制小数）；	#0000000000000000 ～ FFFFFFFFFFFFFFFF 或 &0 ～ 18,446,744,073,709,551,615

(2) 功能块输入变量和输出变量大小必须与程序符号（全局和局部符号）匹配一致。详见下表。

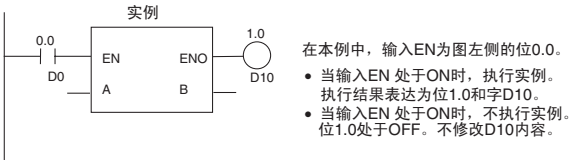
大小	功能块变量数据类型	程序符号 （全局和局部符号）数据类型
1 位	BOOL	BOOL
16 位	INT, UINT, WORD	INT, UINT, UINT BCD, WORD
32 位	DINT, UDINT, REAL, DWORD	DINT, UDINT, UDINT BCD, REAL, DWORD
64 位	LINT, ULINT, LREAL, LWORD	LINT, ULINT, ULINT BCD, LREAL, LWORD
1 位以上	非波尔	CHANNEL, NUMBER (见注释)

注 只有程序符号 NUMBER 可以设置于输入参数中。
输入值必须在功能块变量数据类型大小范围内。

2-2-3 操作规范

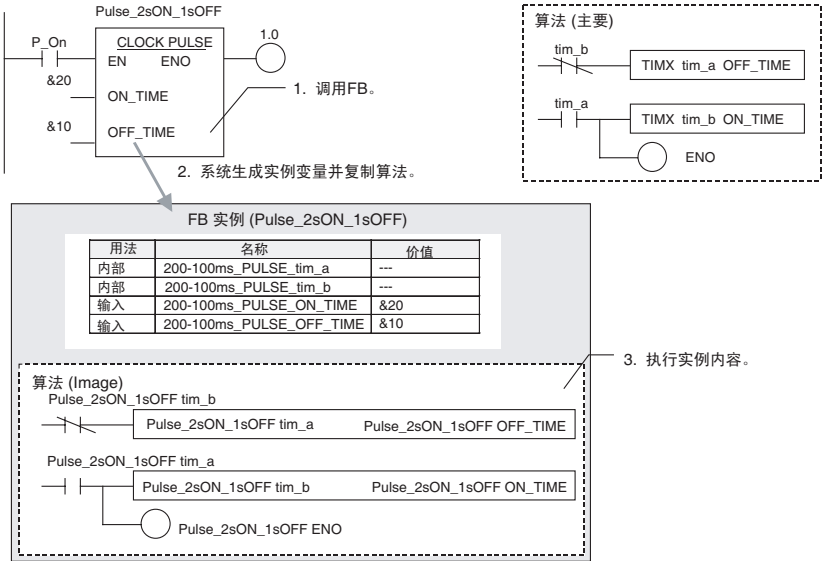
调用实例

用户可以从任何地方调用实例。当输入 EN 处于 ON 时，执行实例。



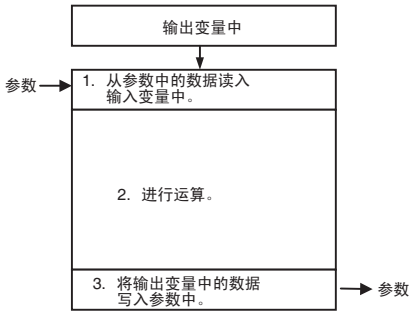
执行实例时的操作

当功能块 EN 输入变量的输入处于 ON 时，系统调用功能块。当调用功能块时，系统生成实例变量并复制注册功能块中的算法。然后，执行实例。



执行次序如下：

1. 从参数中的数据读入输入变量中。
2. 进行运算。
3. 将输出变量中的数据写入参数中。

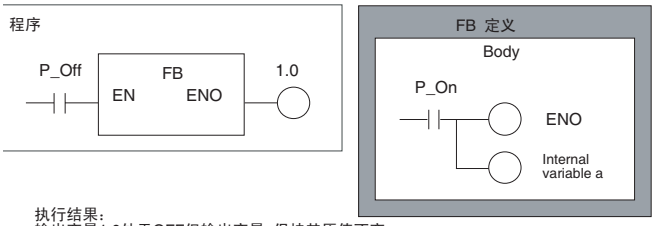


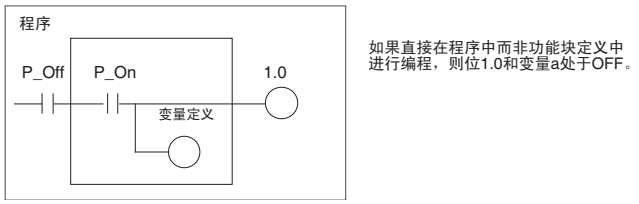
数据不能与算法中的参数交换。

此外，如果进行运算不修改输出变量，则输出参数仍保留其原值。

不执行实例时的操作

当功能块 EN 输入变量的输入处于 OFF 时，不调用功能块。因此，实例内部变量不发生变化 （数值保持不变）。同样，当 EN 处于 OFF 时，输出变量不发生变化 （数值保持不变）。



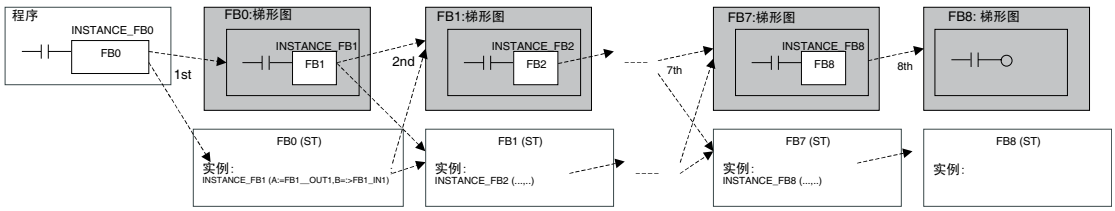


⚠ 注意 当实例的EN熟人变量处于OFF时不执行实例。因此，当EN处于OFF时，不初始化微分和定时器指令。如果微分和定时器指令正在使用，则使用 Always ON 标记 （P_On）用于 EN 输入条件以及包含在功能块定义内的指令输入条件。

嵌套

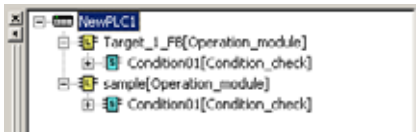
通过 CX-Programmer 6.0 版或以后版本可以从另一功能块中调用功能块。例如，支持嵌套。功能块可以嵌套 8 级 （包括从程序中调用的功能块）。

调用功能块和被调用功能块可以是 ST 语言和梯级语言或两者组合。



“INSTANCE_FB1,”和“INSTANCE_FB2,”等为功能块数据类型实例名。
注：任何梯级语言和结构化文本编程的组合的可以在被调用和调用功能块之间使用。

功能块嵌套级还可以目录树格式显示。同时还具有 FB 实例浏览器功能。



嵌套的功能块功能块定义包含在含调用功能块定义的功能块库文件（.cxf）中。

2-3 功能块限制

梯级编程限制

对用于梯级程序中的指令有一些限制。

功能块定义中禁止的指令

以下指令不能用于功能块定义中。如果使用其中任何一个指令，则会发生错误。

- 块编程指令 （包括 BPRG 和 BEND 的所有指令）
- 子程序指令 （SBS、 GSBS、 RET、 MCRO、 SBN、 GSBN 和 GRET）
- 跳转指令 （JMP、 CJP、 CJPN 和 JME）
- 梯级指令 （STEP 和 SNXT）
- 立即刷新指令 （!）
- I/O REFRESH 指令 （IORF）
- TMHH 和 TMHHX 指令
- CV 地址转换指令 （FRMCV 和 TOCV）
- 堆栈记录位置指令 （PUSH、 FIFO、 LIFO、 SETR 和 GETR）
- FAILURE POINT DETECTION 指令 （FPD）
- 定时器 / 计数器 PV 移至寄存器指令 （MOVRW）

AT 设置限制 （未支持数据区域）

以下区域中的地址不能用于 AT 设置。

- 变址寄存器 （既不支持间接编址也不支持直接编址）和数据寄存器

  注　　直接输入地址而不是 AT 设置。
- 间接编DM或EM地址（既不支持二进制模式也不支持BCD模式间接编址）

指令操作数中的 I/O 存储器直接编址

- 地址而非变量可以直接输入于变址寄存器 （间接和直接编址）和数据寄存器。
  以下数值可以输入于指令操作数中：
  直接编址：IR0 到 IR15 ；间接编址：,IR0 到 ,IR15; 常数偏移（举例说明）：+5,IR0 ； DR 偏移：DR0、IR0 ；自动递增量：,IR0++ ；自动递减量：--,IR0
- 指令操作数的直接编址不支持 I/O 存储器中的其他任何区域。

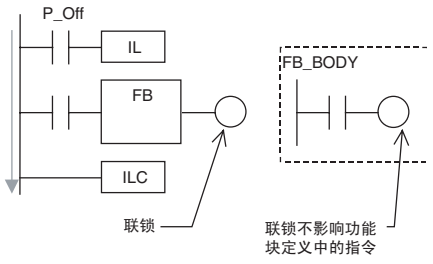
I/O 变量限制 （未支持数据区域）

以下数据区域中的地址不能用作输入和输出变量参数。

- 变址寄存器 （既不支持间接编址也不支持直接编址）和数据寄存器
- 间接编DM或EM地址（既不支持二进制模式也不支持BCD模式间接编址）

联锁限制

当从联锁的程序段中调用功能块时，不执行功能块定义内容。联锁的功能块作为联锁子程序。

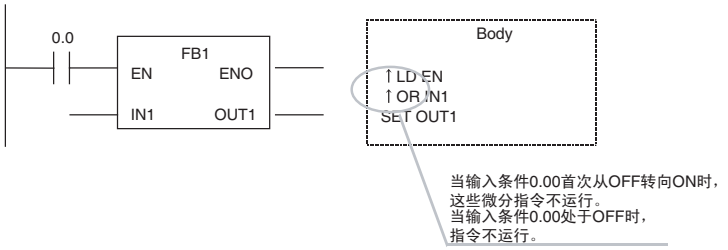


功能块定义中微分指令

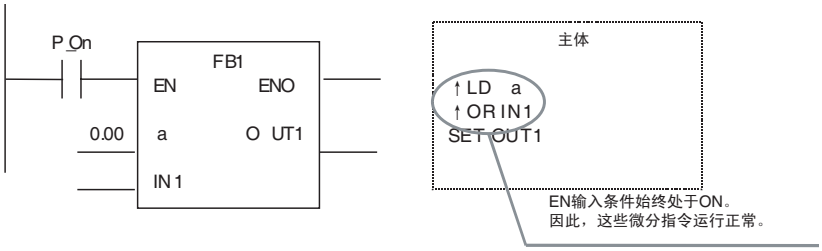
当 EN 输入变量处于 OFF 时，不执行实例。因此，当使用功能块定义中的微分指令时要遵守以下注意事项：（微分指令包括 DIFU、DIFD 和任何带 @ 或 % 前缀的指令）

- 只要实例EN输入变量处于OFF，执行条件就保持在其原有状态（EN输入变量处于 ON 时的最后状态）。微分指令不运行。
- 当实例 EN 输入变量将要处于 ON 时，当前的执行条件状态不与最后循环状态进行比较。当 EN 输入变量已处于 ON 时，当前的执行条件状态与最后循环状态进行比较。因此，微分指令运行不正常。（如果 EN 输入变量仍处于 ON 的话，则当下一前沿或后沿出现时微分指令正常运行）。

举例说明：



如果微分指令正在使用，则使用 Always ON 标记（P_On）用于 EN 输入条件以及包含在功能块定义内的指令输入条件。



- 当指定以下指令的第一个操作数时，在“ # ”后输入十进制数值：
  MILH(517), MILR(518), MILC(519), DIM(631), MSKS(690), MSKR(692), CLI(691), FAL(006), FALS(007), TKON(820), TKOF(821)

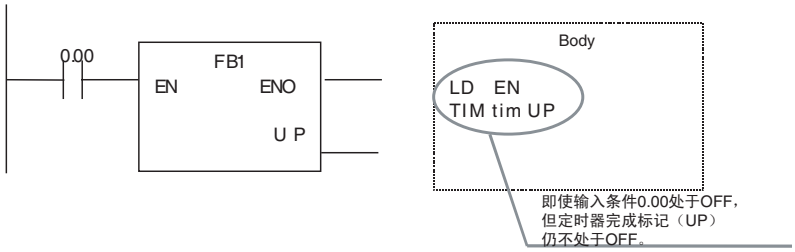
  注　　不支持“ & ”。

- CNR （545）和 CNRX （547）（RESET TIMER/COUNTER）指令不能用来同时复位功能块中的多个定时器和计数器。
  始终规定第一个操作数 （定时器 / 计数器编号 1）和第二个操作数 （定时器 / 计数器编号 2）变量为相同的变量。不能将不同变量赋予第一个操作数和第二个操作数的变量。

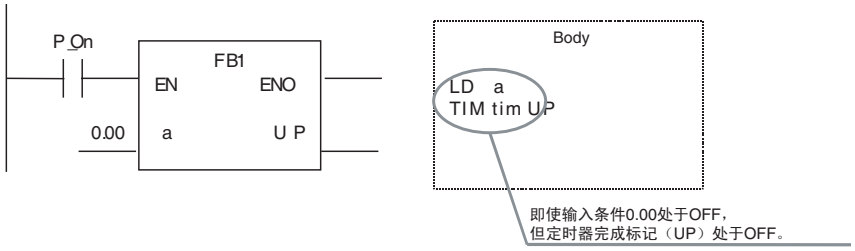
功能块定义中的定时器指令

当 EN 输入变量处于 OFF 时，不执行实例。因此，当使用功能块定义中的微分指令时要遵守以下注意事项：（微分指令包括 DIFU、DIFD 和任何带 @ 或 % 前缀的指令）

即使实例 EN 输入变量处于 OFF，定时器指令仍不初始化。因此，定时器已开始操作后如果 EN 输入变量处于 OFF 的话，则定时器完成标记处于 OFF。



如果定时器指令正在使用，则则使用 Always ON 标记（P_On）用于 EN 输入条件以及包含在功能块定义内的指令输入条件。



- 如果含一个定时器的相同实例同时用于多个地方的话，则复制定时器。

ST 编程限制

- 只支持以下的语句和运算符。
  - 赋值语句
  - 选择语句 （CASE 和 IF 语句）
  - 迭代语句 （FOR、 WHILE、 REPEAT 和 EXIT 语句）
  - RETURN 语句
  - 功能块调用语句
  - 算术运算符
  - 逻辑运算符
  - 比较运算符
  - 数函词
  - 算术函数
  - 注释
- 不能使用 TIMER 和 COUNTER 数据类型。

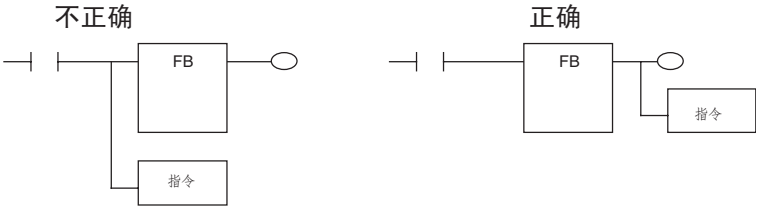
欲知详情，请参阅附录 B 结构化文本 （ST 语言）规格。

程序结构注意事项

程序结构注意事项

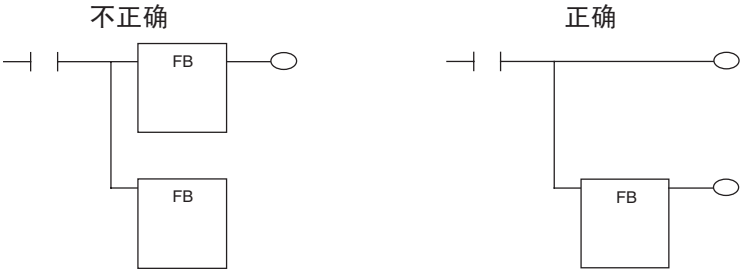
实例左面无分支

实例左侧不允许有分支。只允许在右侧有分支。



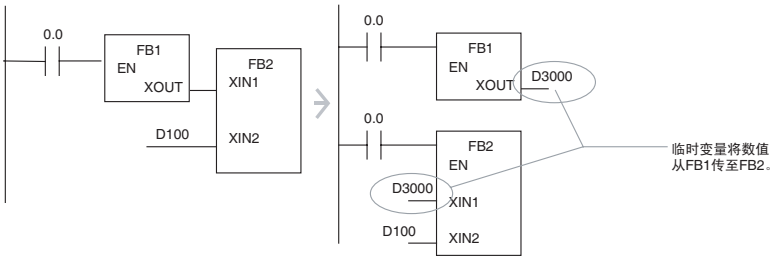
每级只有一个实例

程序级不能有一个以上的实例。



无功能块连接

功能块输入不能与另一功能块输出连接。在这种情况下，必须注册变量以便将第一个功能块的输出传输至第二个功能块输入中。



下载于任务单元

包含功能块的任务不能下载于任务单元中，但可以上载。

编程控制台显示

当用 CX-Programmer 创建的用户程序下载于 CPU 单元而且已被编程控制台读取时，所有实例以问号形式显示。（实例名不显示。）

在线编辑限制

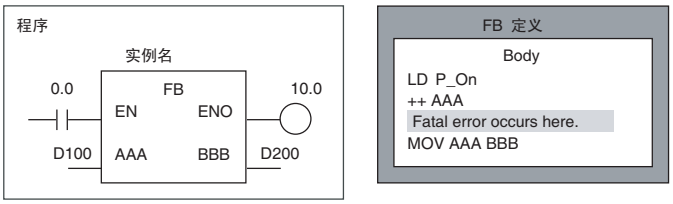
在 CPU 单元中，在用户程序上执行以下在线编辑运算符。

- 修改或删除功能块定义 （变量表或算法）
- 插入实例或修改实例名

注　　可以修改实例 I/O 参数，删除实例。修改实例外的指令。

与错误有关的限制

如果在执行功能块定义时CPU单元发生致命错误，在发生错误地方停止执行梯级程序。



在这种情况下，不执行 MOV AAA BBB 指令以及输出变量 D200 保持执行功能块之前的数值。

禁止进入 FB 实例区域

要使用功能块，则系统需要有存储器区来存储实例内部变量和 I/O 变量。

CS/CJ- 系列 CPU 单元 3.0 版或以后版本和 NSJ 控制器

功能块实例范围	启动地址的初始值	大小的初始值	允许的数据范围
非保持	H512	896	CIO, WR, HR, DM, EM
保持	H1408	128	HR, DM, EM
定时器	T3072	1,024	TIM
计数器	C3072	1,024	CNT

FQM1 柔性运动控制器

FB 实例区域	默认值			适用存储器区域
	起始地址	终端地址	大小	
非保持	5000	5999	1000	CIO, WR, DM
保持	无			
定时器	T206	T255	50	TIM
计数器	C206	C255	50	CNT

如果存取 FB 实例区域地址的用户程序有指令，则 CX-Programmer 在以下情况下输出错误。

- 用户在程序菜单中选择 Program -Compile 或在 PLC 菜单中选择 Compile All Programs 来检查程序时；
- 当试图通过在线编辑来写程序时 （可以写）。

2-4 功能块应用指南

本节提供了有 CX-Programmer 的功能块使用指南。

2-4-1 确定变量数据类型

整数数据类型
（1 字、 2 字或 4 字数据）

当处理 1 字、 2 字或 4 字数据中的单数时，使用以下数据类型：

- INT 和 UINT
- DINT 和 DINT
- LINT 和 ULINT

注 如果所用的数字在范围内，则使用带符号整数。

字数据类型
（1 字、 2 字或 4 字数据）

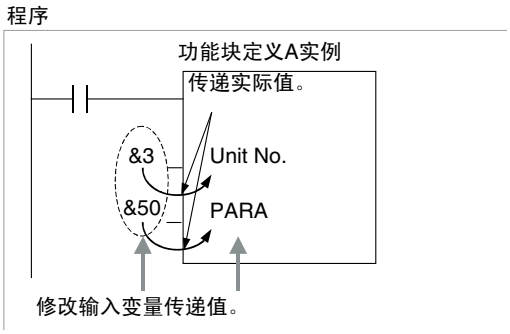
当处理 1 字、2 字或 4 字数据中的数据组（非数字数据）时，使用以下数据类型：

- WORD
- DWORD
- LWORD

2-4-2 确定变量类型（输入、输出、外部和内部）

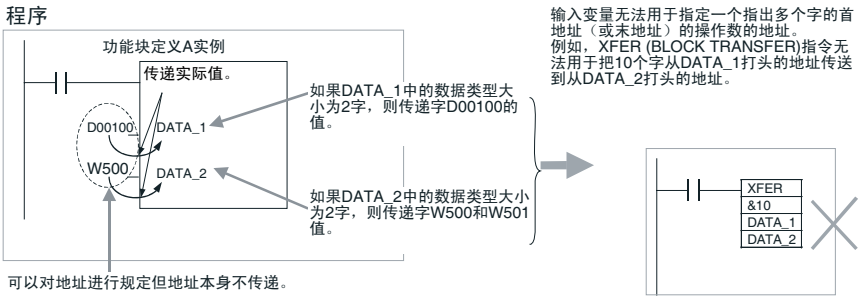
采用输入变量来修 改已传递的值

使用以下输入变量将功能块粘贴于程序中，然后修改传至每个实例功能块的数值 （不是地址）。



以下二种类型的限制可以采用。

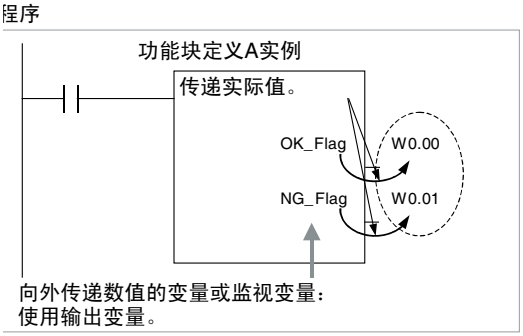
- 地址设于输入参数中但地址不能传至输入变量中 （即使地址设于输入参数中，输入变量的数据类型大小值传至功能块中）。因此，当在功能块内将多字的第一个字或最后一个字在操作数中加以规定时，输入变量不能用于操作数。规定使用带 AT 设置的内部变量；规定内部数组变量的第一个元素或最后一个元素或使用外部变量 （具体内容请参阅 2-4-4 内部变量数组设置）。



- 在进行运算之前，数值从输入参数中以批量方式传输至输入变量中 （不与执行算法中指令同时进行）。因此，当执行功能块算法中指令时，使用内部变量或外部变量而非输入变量将数值从参数传递至输入变量中。

从输出变量中传递 数值或监视输出变量

使用输出变量来粘贴至程序中，然后，从功能块中传输每个实例值 （程序）或监视数值。



采用以下限制。

- 进行运算后，数值从输出变量传输至少输出参数中。

外部变量：程序中的条件标记、时钟脉冲、辅助区域位

用于程序中的条件标记 （例如， Always ON 标记和 Equals 标记）、时钟脉冲（例如， 1.0 秒时钟脉冲位）、预先注册的辅助区域位 （例全局符号如，首次循环标记）和全局符号全部是由系统定义的外部变量。

内部变量：内部分配的变量和需要 AT 设置的变量

未规定位输入、输出或外部的变量均为内部变量。内部变量包括内部分配地址的变量、要求有 AT 设置地址的变量（例如，I/O 分配地址、特殊分配于特殊 I/O 单元的地址）或需要数组设置的变量。有关需要 AT 设置或数组设置条件的详细资料，请参阅 2-4-3 内部变量 AT 设置 和 2-4-4 内部变量数组设置。

2-4-3 内部变量 AT 设置

在以下条件下始终规定内部变量的 AT 设置。

- 当采用分配于基本I/O单元、特殊I/O单元或CPU总线单元的地址时，这些地址注册于不能规定为外部变量的全局符号中 （例如，为全局符号所设置的数据不稳定）。

  注 特殊 I/O 单元分配地址的变址寄存器规定方法需要 AT 设置 （为分配区域第一个地址而规定的）。（详情请参阅 2-4-5 规定分配于特殊 I/O 单元的地址。）

- 当使用未预先注册为外部变量的辅助区域位时，这些位注册为未规定为外部变量的全局符号。
- 当第一个目的地字在远程节点处设为 SEND （090）时以及第一个源字在本地节点处设为 RECV （098）。
- 当指令操作数规定了多字的第一个字或最后一个字以及内部数组变量不能规定为操作数时 （例如，不能规定数组元素数量）。

2-4-4 内部变量数组设置

采用数组变量来规定多字操作数中的第一个字或最后一个字

当在指令操作数中规定一组字的第一个字或最后一个字 （见注释）时，在 AT 规格或内部分配后根据地址进行指令操作。（因此，变量数据 类型和变量元素数量与指令操作无关）。始终规定有 AT 设置的变量 或有一组元素（与指令处理的数据量相匹配）的数组变量。

注 一些例子为 XFER(070) (BLOCK TRANSFER) 指令的第一个源字或第一个目的地字； SEND(090) 第一个源字或适用指令的控制数据。

详情请参阅 2-5 规定多字中第一个字或最后一个字的操作数指令注意事项。采用以下方法来规定数组变量。

1,2,3...

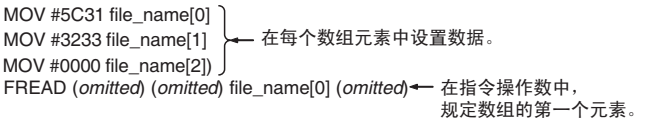
1. 配置有所需元素数的内部数组变量。
   注 确保指令处理的数据量与元素数一样。有关每个指令处理的数据量的详细内容请参阅 2-6 指令支持和操作数限制。
2. 采用功能块定义中的 MOV 指令来将数据设置于每个数组元素中。
3. 规定操作数的数组变量第一个 （或最后一个）元素。因此，第一个 （或最后一个）地址规格在字范围内。

下例可证。

处理多字中的单串数据

在本例中，数组包含 FREAD 指令的目录和文件名 （操作数 S2）。

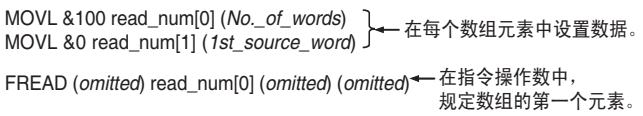
- 变量表
  内部变量；数据类型 = WORD ；有 10 个元素的数组设置；变量名 = 文件名 [0] 到文件名 [9]
- 梯级编程



处理多字中的控制数据

在本例中，数组包含 FREAD 指令的字数和第一个源字 （操作数 S1）。

- 变量表
  内部变量；数据类型 = DINT ；有 3 个元素的数组设置；变量名 = read_num[0] to read_num[9]
- 梯级编程



处理多字中的读取数据块

必须事先确定读取数据的许可量。必须配置数组。这样，可以处理最大数据量。在本例中，数组接收 FREAD 指令的读取数据 （操作数 D）。

- 变量表
  内部变量；数据类型 = WORD ；有 100 个元素的数组设置；变量名 = read_data[0] to read_data[99]
- 梯级编程

FREAD (*omitted*) (*omitted*) (*omitted*) read_data[0]

除法 -- 采用整数数组变量（仅指梯级编程）

两元素数组可以用来存储梯级程序结果。指令结果为 D （商数）和 D+1 （余数）。该方法可以用来获得梯级编程除法运算中获得余数。

注 当采用 ST 语言时，无需使用数组来接收除法运算结果。此外，不能在 ST 语言中直接计算余数。必须按如下方法计算余数

余数 = 被除数 . （除数×商数）

2-4-5 规定分配于特殊 I/O 单元的地址

使用变址寄存器 IR0 到 IR15 （间接规定的常数偏移）来规定分配于特殊 I/O 单元的地址。该地址是基于单元号值 （作为功能块定义内的输入参数）基础而设定的。具体见下例。

注 有关功能块中的变址寄存器使用详细内容请参阅 2-4-6 使用变址寄存器。

举例说明：

例 1：在功能块内指定 CIO 区域 （同 DM 区域）

特殊 I/O 单元

变量：单元号用作输入变量。规定第一个分配地址为内部变量 （AT 设在 CIO 2000）。

程序：采用以下程序：

1,2,3...

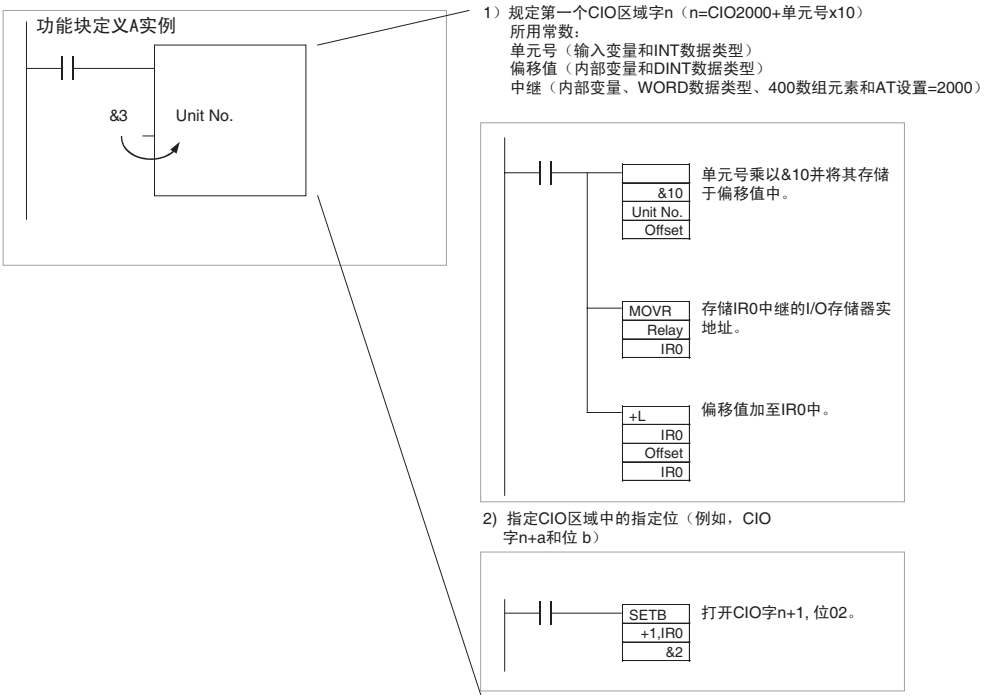
1. 单元号 （输入变量）乘以 &10，创建单元号偏移值 （内部变量和 DINT 数据类型）。
2. 使用 MOVR （560）（MOVE TO REGISTER）指令将第一个分配地址的 I/O 存储器实地址 （内部变量， AT =CIO 2000）存储于变址寄存器中 （例如， IR0）。
3. 在变址寄存器（例如，IR0）中，在 I/O 存储器实地址上加上单元号偏移值。

例 2：指定 CIO 区域中的指定位 （例如， CIO 字 n+a 和位 b）

程序：采用下面其中一个方法：

- 字地址：采用间接指令（例如，+a 和 IR0）来规定变址寄存器常数偏移值。
- 位地址：规定可以指定字内位地址的指令 （例如，当写入时 SETB 指令的第二个操作数中的 &b 以及读取时的 TST 指令）。

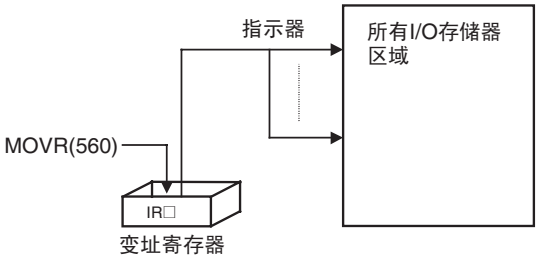
例子：特殊 I/O 单元



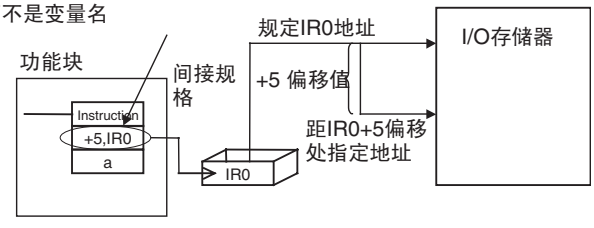
2-4-6 使用变址寄存器

变址寄存器 IR0 到 IR15 用作规定 I/O 存储器地址的指示器。这些变址寄存器可以在功能块中使用从而可以直接通过 IR0 到 IR15 来指定地址而不是变量名（变址寄存器直接规格：IR0 到 IR15 ；变址寄存器间接规格：,IR0 到 ,IR15）

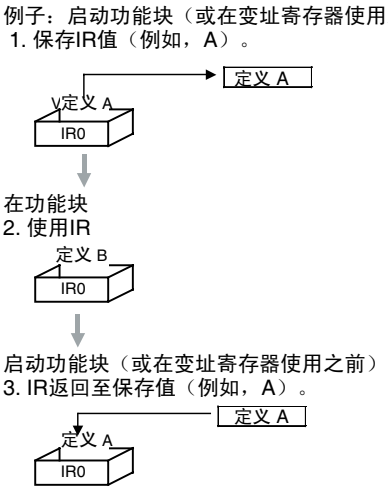
注 在使用 MOVR （560）指令将实际的 I/O 存储器地址存储于变址寄存器后，可采用通用指令来间接指定变址寄存器。这样，就可动态地规定所有 I/O 存储器区域。



例子：采用常数偏移值规格来规定+5, IR0而不是变量名



注 (1) 当在功能块中使用变址寄存器 IR0 到 IR15 时，使用另一功能块中相同的变址寄存器或功能块外程序中的变址寄存器将在两个实例之间形成竞争 . 程序不能正确执行。因此，当功能块开始运行（或变址寄存器使用之前）时以及功能块结束运行（或变址寄存器使用之后）时，始终保存变址寄存器值 （IR0 到 IR15）。进行程序处理以使变址寄存器返回至其保存值。

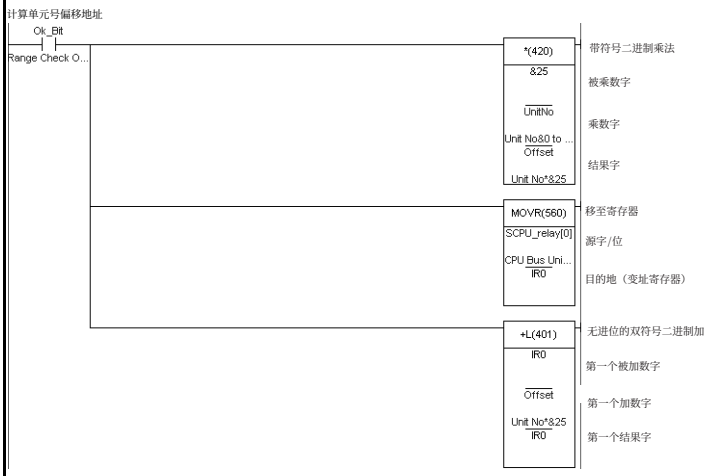


(2) 在变址寄存器使用之前，始终设置数值。但如果未设置数值情况下使用变址寄存器，则运算不稳定。

应用实例

下例仅涉及在功能块中使用变址寄存器 IR0 到 IR15。

范例	详细内容
在使用变址寄存器之前保存变址寄存器值。 	当变址寄存器用于此功能块时，当启动功能块 （或变址寄存器使用之前）以使数值在功能块结束运行后 （或变址寄存器使用之后）返回至其原变址寄存器值时进行处理以保存变址寄存器值。例子：将变址寄存器 IR0 保存于 SaveIR[0] （内部变量、数据类型 DINT 和 1 个数组元素）以保存变址寄存器 IR0 内容。

范例	详细内容
使用变址寄存器 1) 数值设于变址寄存器之中。（存储第一个 CIP 区域字 n 的 I/O 存储器实地址） 	例子：分配于 CPU 总线单元分配区域的 CIO1500+ 单元号 x25 的第一个字 I/O 存储器实地址存储于 IR0 中。CPU 总线单元分配区域是给基于功能块传输而来的 CPU 总线装置单元号 （&0~&15）基础上。 程序： 假设单元号 &0~&15 （从功能块外侧）已输入于 UnitNo （输入变量、 INT 数据类型）。 1. UnitNo 乘以 &25，存储于 Offset （内部变量、 DINT 数据类型）。 2. 存储 SCPU_Relay （内部变量、 INT 数据类型）I/O 存储器实地址 （内部变量、 WORD 数据类型、（如需，则可在变址寄存器 IR0 中规定数组为 400 个元素 （见注释）； AT 设置 =1500） 注 规定诸如 SCPU_relay [2] 的 SCPU_relay 数组。例如，规定地址 CIO 1500 + (UnitNo × &25) + 2。这也适用于下例 2。 3. 通过变量 Offset （变量 UnitNo × &25）来递增变址寄存器 IR0 中的 I/O 存储器实地址。
2) 指定变址寄存器常数偏移值 （规定 CIO n+0 到 n+24 之间的位） 	通过上述步骤 1 的处理后， CIO 1500+（UnitNo × &25）的 I/O 存储器实地址存储于变址寄存器 IR0 中。因此，使用 IR0 的常数偏移值来规定字地址。例如，指定 +2,IR0 将指定 CIO 1500 + (UnitNo × &25) + 2。 注 采用带 SCPU_relay 的数组设置来规定 SCPU_relay [2] 进而指定 CIO 1500 + (UnitNo × &25) + 2。 采用可以规定字内位地址的指令来规定位地址 （例如， TST(350/351)/SETB(532) 指令第二个操作数）。 例子：当 NetCheck_OK (内部变量、 BOOL 数据类型) 处于 ON 以及偏离 IR0+6 处的字的位 15 （CIO 1500 +UnitNo × &25 +6）处于 ON 时，变量 NodeSelf_OK 处于 ON。
变址寄存器返回至原值 	在功能块结束运行 （或变址寄存器已使用后）后，变址寄存器返回至原值例子：已保存的变量 SaveIR[0] 值存储于变址寄存器 IR0。功能启动 （或在变址寄存器使用之前）时数值返回至内容。

2-5 规定多字中第一个字或最后一个字的操作数指令注意事项

当使用梯级编程来创建指令操作数可以指定第一个字或最后一个字的功能块时，在规定操作数时应注意如下事项。

当操作数指定多字的第一个字或最后一个字时，根据 AT 设置的内部分配地址（或外部变量设置）来操作指令。因此，变量数据类型和数组元素数量与指令操作运行无关。规定有 AT 设置的变量或有一组元素 （与指令处理的数据量相匹配）的数组变量。

是采用 AT 设置 （或外部变量设置）来指定指令操作数中第一个字或最后一个字还是采用一组元素的数组设置来指定指令操作数中第一个字或最后一个字，具体情况请参阅 2-6 指令支持和操作数限制。

注 要在指令操作数中指定多字的第一个字或最后一个字，则始终规定有 AT 设置的变量（或外部变量）或与指令处理的数据量相同的变量。以下注意事项适用。

1,2,3...

1. 如果非数组变量规定时无 AT 设置以及数据量不匹配，则在编辑时 CX-Programmer 输出错误。
2. 当指定数组变量时，以下注意事项适用。

<u>指令操作数中处理的容量大小是固定的</u>

确保数组中的元素数量与指令处理的容量大小相同。否则，在编辑时 CX-Programmer 输出错误。

<u>指令操作数中处理的容量大小是不固定的</u>

确保数组中的元素数量等于或大于另一操作数规定的容量大小。

其他操作数规定大小：常数

编辑时 CX-Programmer 输出错误。

其他操作数规定大小：变量

编辑时 （显示警告信息）即使数组中的元素数量与另一操作数 （变量）中规定的容量大小不匹配， CX-Programmer 也不输出错误。

特别是数组中的元素数量小于另一操作数规定的容量大小时，（例如，指令处理量为 16 但实际注册于变量表中的元素数量为 10 时），在超出元素数量的区域中指令执行读 / 写处理。（在本例中，在元素数量注册于实际变量表中后，执行下一 6 个字的读 / 写处理。）如果另一指令正在使用相同的区域 （包括内部变量分配），则会发生意外操作从而会导致严重事故。

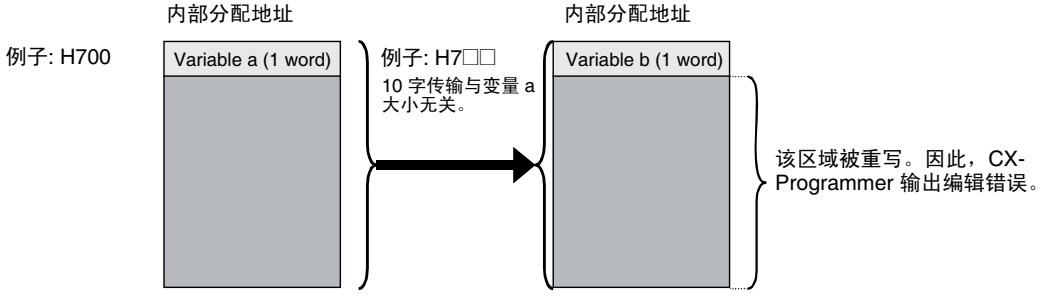
不得使用容量大小与指令处理的数据量不匹配的变量。指令在规定一组字第一个地址 （或最后一个地址）的操作数中。始终使用容量与指令所要求的数据量相同的非数组变量数据类型或元素数量与指令所要求的数据量相同的数组变量。否则，会出现以下错误。

数据量不匹配以及无 AT 设置的非数组变量

如果规定多字第一个地址 （或最后一个地址）的操作数采用容量大小与指令所要求的数据量不匹配以及未使用 AT 设置的非数组变量数据类型，则 CXProgrammer 不输出编辑错误。

例子：BLOCK TRANSFER(070) 指令：XFER W S D
（W：字数； S：第一个源字； D：第一个目的地字）
当 &10 指定于 W 中时，数据类型 WORD 的变量 a 指定于 S 中而数据类型 WORD 的变量 b 指定于 D 中：XFER &10 a b。
XFER （070）指令将自动分配地址于变量 a 中的 10 字数据传输至自动分配地址于变量 b 的 10 字中。因此， CX-Programmer 输出编辑错误。

例子: XFER &10 a b
(变量 a 和 b 为WORD数据类型)



数组变量

结果取决于以下条件。

指令处理的容量大小是固定的

如果指令处理的容量大小是固定的操作数而且该大小与数组元素数量不匹配，则 CX-Programmer 输出编辑错误。

例子：LINE TO COLUMN(064) 指令； COLM S D N

（S：位数； D：第一个目的地字； N：源字）

例如， COLM a b[0] c

当有 10 个数组元素的 WORD 数据类型数组用于 16 个数组元素时，如果其规定于 D 中，则在编辑时 CX-Programmer 输出错误。

指令处理的容量大小是不固定的

当指令处理的操作数容量大小是不固定的 （当容量大小由指令中的另一操作数指定时），确保数组元素数量等于或大于其他操作数中规定的容量大小 （例如，指令处理的容量大小）。

其他操作数规定大小：常数

编辑时 CX-Programmer 输出错误。

例子 : BLOCK TRANSFER: XFER W S D

（W：字数； S：第一个源字； D：第一个目的地字）

当 &20 规定于 W 中时，数据类型 WORD 和 10 个元素的数组变量 a 规定于 S 中而数据类型 WORD 和 10 个元素的数组变量 b 规定于 D 中。

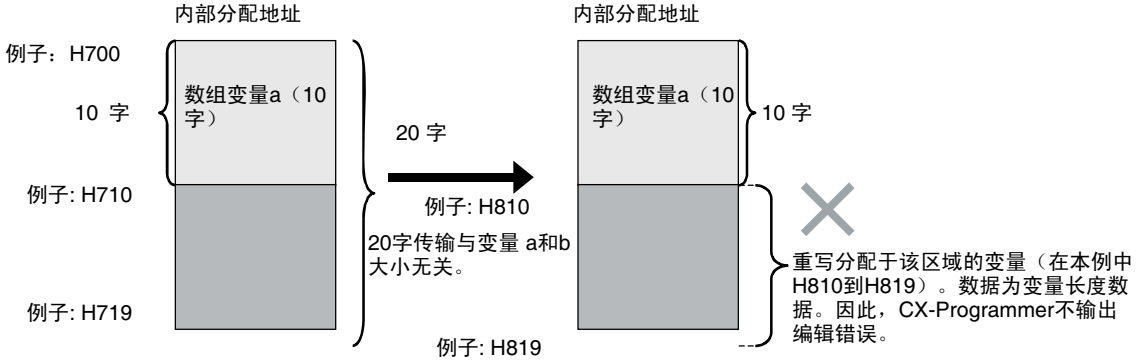
XFER &20 a[0] b[0]

即使数组变量 a[0] 和 b[0] 均为 10 字，XFER（070）指令将执行传输 20 字（规定于 W 中）的处理。因此，在已分配的数组元素数量后面 XFER （070）指令对 I/O 存储器区域进行读 / 写处理。详见下图。

因此，如果 a[10 elements] 内部分配字（例如，H700 ~ H709）和 b[10 elements] 内部分配字（例如，H800 ~ H809），XFER （070）将字 H700~ H719 中的数据传输至字 H800 ~ H819。

在此运算中，如果分配另一内部分配变量 （例如， c）为 H810 到 H819 中的分配字，则字被重写从而造成意外运算。要传输 20 字，确保元素数量规定为 20 个元素 （数组变量 a 和 b）。

XFER &20 a[0] b[0]
使用有10个元素的WORD 数据类型用于变量a 和b。
要传输20字，确保元素数量规定为20个元素（数组变量a 和b）。



其他操作数规定大小：变量

编辑时即使数组中的元素数量与另一操作数 （变量）中规定的容量大小不匹配， CX-Programmer 也不输出错误。

根据操作数所规定的容量大小执行指令。这与数组变量中的元素数量无关。

特别是数组中的元素数量小于另一操作数 （变量）规定的容量大小 （例如，指令处理的容量大小）时，其他变量将受到影响并会发生意外操作。

2-6 指令支持和操作数限制

本附录表中给出了哪个指令用于梯级编程语言创建的功能块和适用于这些功能块和操作数的限制 （包括变量是否需要数组变量和 AT 设置或外部变量规格和哪个数据类型可以使用）。

指令支持

不支持 CX-Programmer、CP- 系列 CPU 单元和 CS/CJ- 系列 CPU 单元（3.0 版）功能块定义的指令请参阅符号栏中不支持功能块 的内容。

操作数限制

- 需要数组变量 AT 设置或规格的操作数（规定多字第一个字或最后一个字）示于所需的 AT 设置或数组一栏中。
  是：必须规定操作数的 AT 设置 （或外部规格）或数组变量以规定多字第一个字或最后一个字。
  - 括号内的数值为指令用于读、写或其他处理的固定值。该容量大小显示数据类型大小或规定于字单元中的数组变量所需容量大小。至于数组变量，该容量大小必须与元素数量相同。否则， 在编辑时 CX-Programmer 输出错误。
  - 如果“ 不固定 ”示于括号中，则可以修改指令用于读、写或其他处理的容量大小。确保提供数组元素数量所需的最大容量。编辑时即使数组中的元素数量与另一操作数 （变量）中规定的容量大小不匹配， CX-Programmer 也不输出错误。根据操作数所规定的容量大小执行指令。这与数组变量中的元素数量无关。

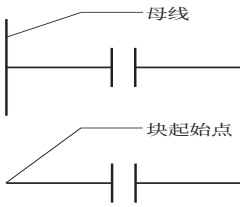
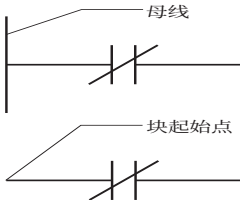
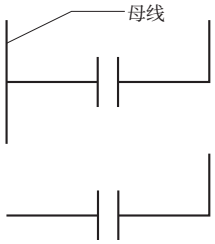
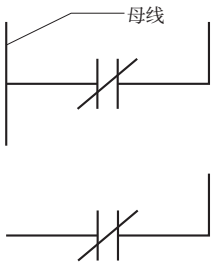
  ---: 不需要数组变量 AT 设置的操作数。

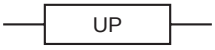
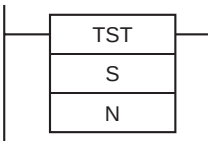
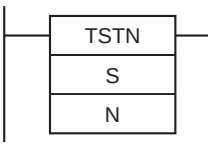
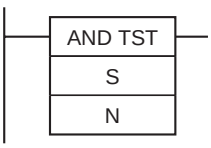
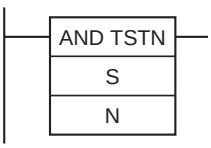
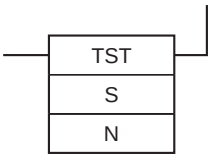
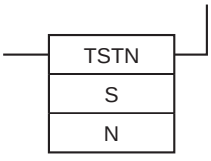
  注 当在指令操作数中规定多字第一个字或最后一个字时，输入参数不能用来传输数据至变量中或从变量中上传数据。必须配置 AT 设置或有所需元素数量的数组变量。将数组数据设置于功能块定义中后，必须规定操作数数组变量的第一个元素或最后一个元素。
- 必须在网络远程节点处指定 I/O 存储器地址 AT 设置的任何操作数示于所需的 AT 设置或数组栏中有 AT 设置的在远程节点处规定地址。

下表列出了所有 CS/CJ- 系列 CPU 单元、NSJ- 系列 NSJ 控制器和 FQM1 柔 性运动控制器 （3.0 版或以后版本）支持的指令。

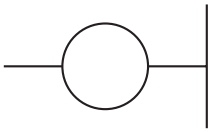
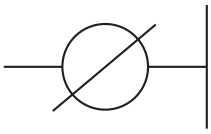
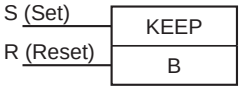
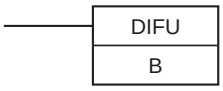
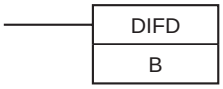
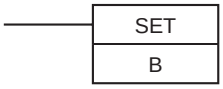
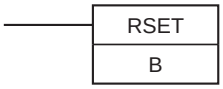
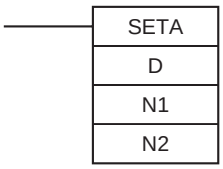
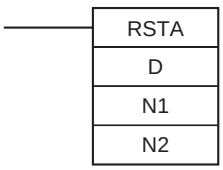
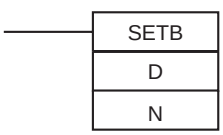
- 一些只有 FQM1 柔性运动控制器（3.0 版或以后版本）支持的指令。这些指令以“ FQM1 only ”（在助记符下）来表示。
- 只有 CS/CJ- 系列 CPU 单元（例如，不能被 FQM1 柔性运动控制器（3.0 版或以后版本）和 NSJ- 系列 NSJ 控制器支持的指令。详情请参阅 FQM1 指令参考手册 （目录号：O013）确认支持的指令。

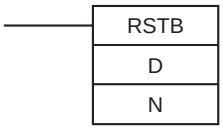
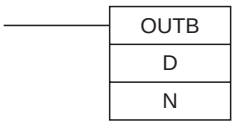
2-6-1 控制输入指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量 （所需的字数据量示于括号中）
LOAD	LD @LD %LD !LD !@LD !%LD			B: 位	BOOL	---
LOAD NOT	LD NOT !LD NOT @LD NOT %LD NOT !@LD NOT !%LD NOT			B: 位	BOOL	---
AND	AND @AND %AND !AND !@AND !%AND			B: 位	BOOL	---
AND NOT	AND NOT !AND NOT @AND NOT %AND NOT !@AND NOT !%AND NOT			---	BOOL	---
OR	OR @OR %OR !OR !@OR !%OR			---	BOOL	---
OR NOT	OR NOT !OR NOT @OR NOT %OR NOT !@OR NOT !%OR NOT			---	BOOL	---
AND LOAD	AND LD			---	---	---
OR LOAD	OR LD			---	---	---

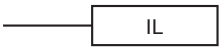
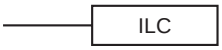
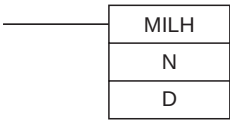
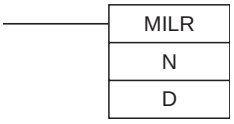
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量 （所需的字数据量示于括号中）
NOT	NOT	520		---	---	---
CONDITION ON	UP	521		---	---	---
CONDITION OFF	DOWN	522		---	---	---
BIT TEST	LD TST	350		S: 源字	WORD	---
				N: 位数	UINT	---
BIT TEST	LD TSTN	351		S: 源字	WORD	---
				N: 位数	UINT	---
BIT TEST	AND TST	350		S: 源字	WORD	---
				N: 位数	UINT	---
BIT TEST	AND TSTN	351		S: 源字	WORD	---
				N: 位数	UINT	---
BIT TEST	OR TST	350		S: 源字	WORD	---
				N: 位数	UINT	---
BIT TEST	OR TSTN	351		S: 源字	WORD	---
				N: 位数	UINT	---

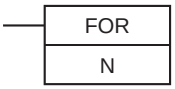
2-6-2 控制输出指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
OUTPUT	OUT !OUT			---	BOOL	---
OUTPUT NOT	OUT NOT !OUT NOT			---	BOOL	---
KEEP	KEEP !KEEP	011		---	BOOL	---
DIFFERENTIATE UP	DIFU !DIFU	013		---	BOOL	---
DIFFERENTIATE DOWN	DIFD !DIFD	014		---	BOOL	---
SET	SET @SET %SET !SET !@SET !%SET			B: 位	BOOL	---
RESET	RSET @RSET %RSET !RSET !@RSET !%RSET			B: 位	BOOL	---
MULTIPLE BIT SET	SETA @SETA	530		D: 起始字 N1: 起始位 N2: 位数	UINT UINT UINT	有 （不固定） --- ---
MULTIPLE BIT RESET	RSTA @RSTA	531		D: 起始字 N1: 起始位 N2: 位数	UINT UINT UINT	有 （不固定） --- ---
SINGLE BIT SET	SETB @SETB !SETB	532		D: 字地址 N: 位数	UINT UINT	--- ---

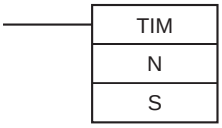
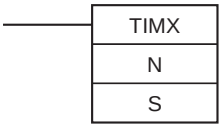
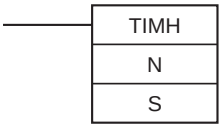
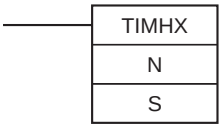
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
SINGLE BIT RESET	RSTB @RSTB !RSTB	533		D: 字地址	UINT	---
				N: 位数	UINT	---
SINGLE BIT OUTPUT	OUTB @OUTB !OUTB	534		D: 字地址	UINT	---
				N: 位数	UINT	---

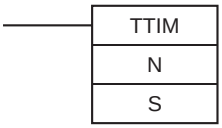
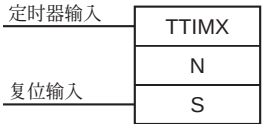
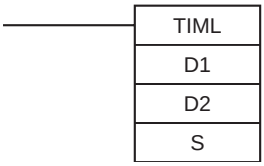
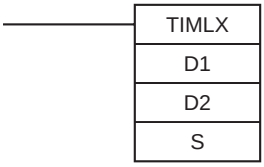
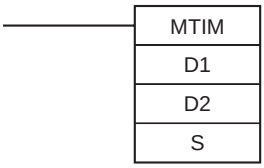
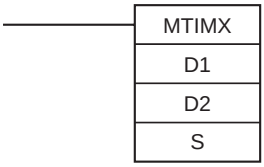
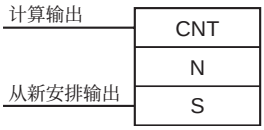
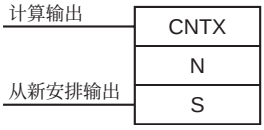
2-6-3 顺序控制指令

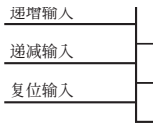
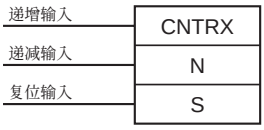
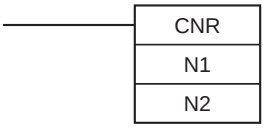
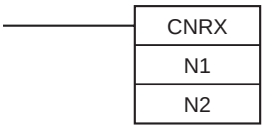
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
END	END	001		---	---	---
NO OPERATION	NOP	000	---	---	---	---
INTERLOCK	IL	002		---	---	---
INTERLOCK CLEAR	ILC	003		---	---	---
MULTI-INTERLOCK DIFFERENTIATION HOLD	MILH	517		N: 联锁	#+ 仅十进制	---
				D: 联锁状态位	BOOL	---
MULTI-INTERLOCK DIFFERENTIATION RELEASE	MILR	518		N: 联锁	#+ 仅十进制	---
				D: 联锁状态位	BOOL	---
MULTI-INTERLOCK CLEAR	MILC	519		N: 联锁	#+ 仅十进制	---
JUMP	JMP	004	功能块中不支持	N: 跳转数	---	---
JUMP END	JME	005	功能块中不支持	N: 跳转数	---	---
CONDITIONAL JUMP	CJP	510	功能块中不支持	N: 跳转数	---	---
CONDITIONAL JUMP	CJPN	511	功能块中不支持	N: 跳转数	---	---
MULTIPLE JUMP	JMP0	515		---	---	---
MULTIPLE JUMP END	JME0	516		---	---	---

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
FOR-NEXT LOOPS	FOR	512		N: 环数	UINT	---
BREAK LOOP	BREAK	514		---	---	---
FOR-NEXT LOOPS	NEXT	513		---	---	---

2-6-4 定时器和计数器指令

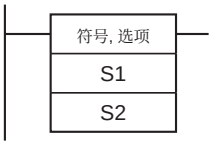
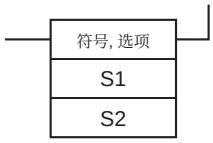
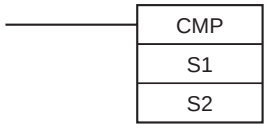
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
TIMER	TIM (BCD)			N: 定时器数	TIMER	---
				S: 设定值	WORD	---
	TIMX (BIN)	550		N: 定时器数	TIMER	---
				S: 设定值	UINT	---
HIGH-SPEED TIMER	TIMH (BCD)	015		N: 定时器数	TIMER	---
				S: 设定值	WORD	---
	TIMHX (BIN)	551		N: 定时器数	TIMER	---
				S: 设定值	UINT	---
ONE-MS TIMER	TMHH (BCD)	540	计数输入复位输入	N: 定时器数	TIMER	---
				S: 设定值	WORD	---
	TMHHX (BIN)	552	计数输入复位输入	N: 定时器数	TIMER	---
				S: 设定值	UINT	---

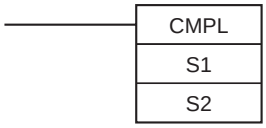
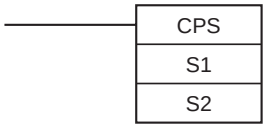
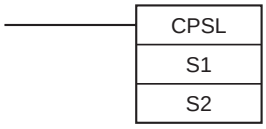
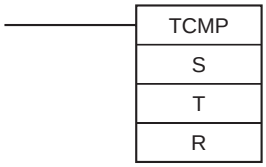
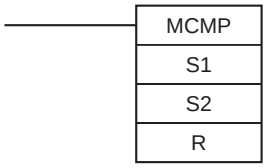
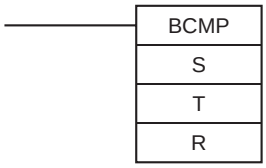
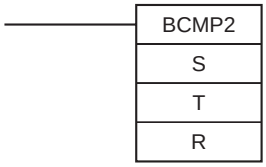
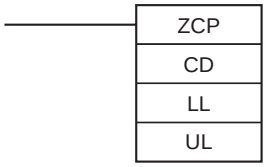
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
ACCUMULATIVE TIMER	TTIM (BCD)	087		N: 定时器数	TIMER	---
				S: 设定值	WORD	---
	TTIMX (BIN)	555		N: 定时器数	TIMER	---
				S: 设定值	UINT	---
LONG TIMER	TIML (BCD)	542		D1: 完成标记	WORD	---
				D2: PV 字	DWORD	---
				S: SV 字	DWORD	---
	TIMLX (BIN)	553		D1: 完成标记	UINT	---
				D2: PV 字	UDINT	---
				S: SV 字	UDINT	---
MULTI-OUTPUT TIMER	MTIM (BCD)	543		D1: 完成标记	UINT	---
				D2: PV 字	WORD	---
				S: SV 字	WORD	Yes (8)
	MTIMX (BIN)	554		D1: 完成标记	UINT	---
				D2: PV 字	UINT	---
				S: SV 字	WORD	Yes (8)
COUNTER	CNT (BCD)			N: 计数器数	COUNTER	---
				S: 设定值	WORD	---
	CNTX (BIN)	546		N: 计数器数	COUNTER	---
				S: 设定值	UINT	---

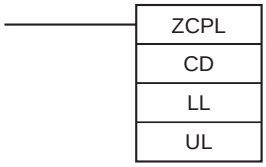
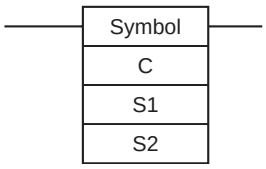
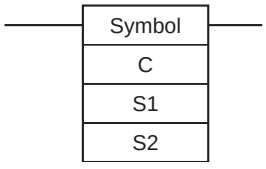
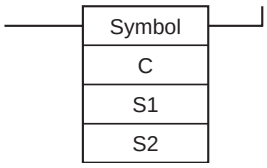
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
REVERSIBLE COUNTER	CNTR (BCD)	012		N: 计数器数	COUNTER	---
				S: 设定值	WORD	---
	CNTRX (BIN)	548		N: 计数器数	COUNTER	---
				S: 设定值	UINT	---
RESET TIMER/ COUNTER	CNR @CNR (BCD)	545		N1: 范围内第一个数字	TIMER or COUNTER（见注解）	---
				N2: 范围内最后一个数字	TIMER or COUNTER（见注解）	---
	CNRX @CNRX (BIN)	547		N1: 范围内第一个数字	TIMER or COUNTER（见注解）	---
				N2: 范围内最后一个数字	TIMER or COUNTER（见注解）	---

注　　当 N1 和 N2 规定的变量均相同时，启动。

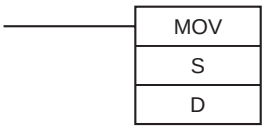
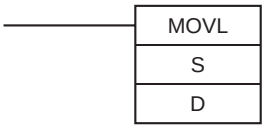
2-6-5 比较指令

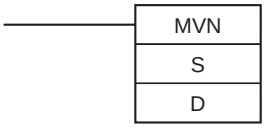
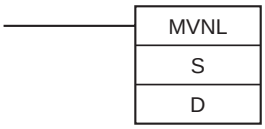
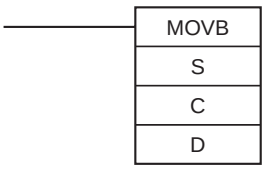
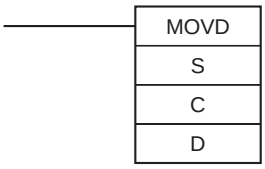
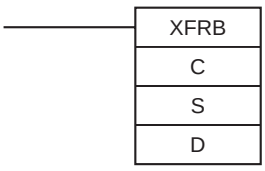
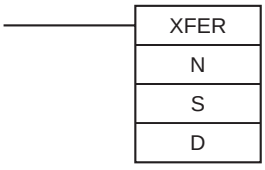
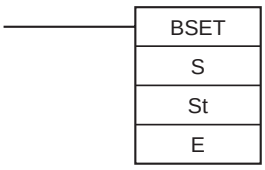
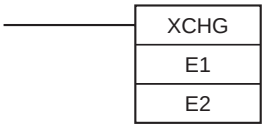
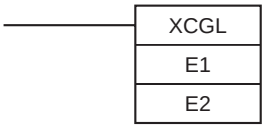
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
比较符号 (无符号)	LD,AND, OR + =, <>, <, <=, >, >=	300 (=) 305 (<>) 310 (<) 315 (<=) 320 (>) 325 (>=)	使用 LD:  使用 AND:  使用 OR:  	S1: 比较数据 1	UINT	---
				S2: 比较数据 2	UINT	---
比较符号 (双倍字，无符号)	LD,AND, OR + =, <>, <, <=, >, >= + L	301 (=) 306 (<>) 311 (<) 316 (<=) 321 (>) 326 (>=)	---	S1: 比较数据 1	UDINT	---
				S2: 比较数据 2	UDINT	---
比较符号 (有符号)	LD,AND, OR + =, <>, <, <=, >, >= + S	302 (=) 307 (<>) 312 (<) 317 (<=) 322 (>) 327 (>=)	---	S1: 比较数据 1	INT	---
				S2: 比较数据 2	INT	---
比较符号 (双倍字，有符号)	LD,AND, OR + =, <>, <, <=, >, >= + SL	303 (=) 308 (<>) 313 (<) 318 (<=) 323 (>) 328 (>=)	---	S1: 比较数据 1	DINT	---
				S2: 比较数据 2	DINT	---
UNSIGNED COMPARE	CMP !CMP	020	 	S1: 比较数据 1	UINT	---
				S2: 比较数据 2	UINT	---

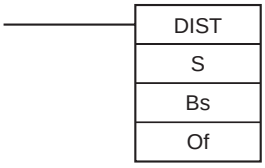
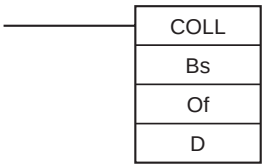
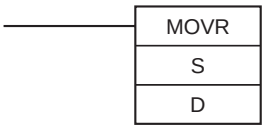
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
DOUBLE UNSIGNED COMPARE	CMPL	060		S1: 比较数据 1	UDINT	---
				S2: 比较数据 2	UDINT	---
SIGNED BINARY COMPARE	CPS !CPS	114		S1: 比较数据 1	INT	---
				S2: 比较数据 2	INT	---
DOUBLE SIGNED BINARY COMPARE	CPSL	115		S1: 比较数据 1	DINT	---
				S2: 比较数据 2	DINT	---
TABLE COMPARE	TCMP @TCMP	085		S: 源字	WORD	---
				T: 表中第一个字	WORD	Yes (16)
				R: 结果字	UINT	---
MULTIPLE COMPARE	MCMP @MCMP	019		S1: 比较数据 1	WORD	Yes (16)
				S2: 比较数据 2	WORD	Yes (16)
				R: 结果字	UINT	
UNSIGNED BLOCK COMPARE	BCMP @BCMP	068		S: 源字	WORD	---
				T: 表中第一个字	WORD	Yes (32)
				R: 结果字	UINT	---
EXPANDED BLOCK COMPARE	BCMP2 @BCMP2	502		S: 源字	WORD	---
				T: 表中第一个字	WORD	Yes (不固定)
				R: 结果字	WORD	---
AREA RANGE COMPARE	ZCP	088		CD: 比较数据 （1 字）	UINT	---
				LL: 范围下限	UINT	---
				UL: 范围上限	UINT	---

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
DOUBLE AREA RANGE COMPARE	ZCPL	116		CD: 比较数据 (2 字)	UDINT	---
				LL: 范围下限	UDINT	---
				UL: 范围上限	UDINT	---
时间比较	LD, AND, OR + =DT, <> DT, <DT, <=DT, >DT, >=DT	341 (=DT) 342 (<>DT) 343 (<DT) 344 (<=DT) 345 (>DT) 346 (>=DT)	LD (LOAD):  AND:  OR: 	C: 控制字	WORD	---
				S1: 现在时间的第一个字	WORD	Yes (3)
				S2: 比较时间的第一个字	WORD	Yes (3)

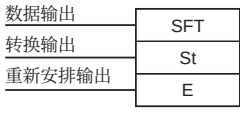
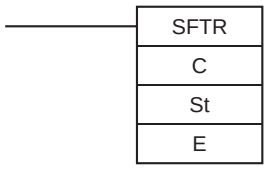
2-6-6 数据传送指令

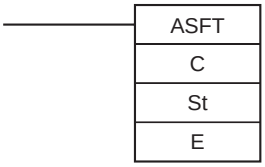
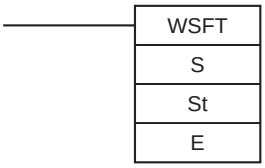
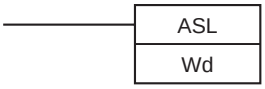
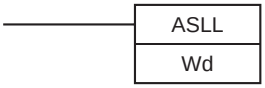
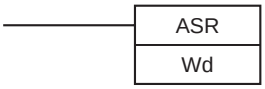
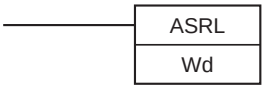
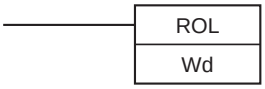
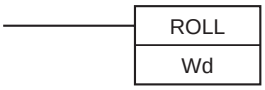
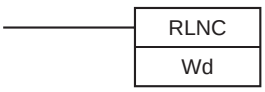
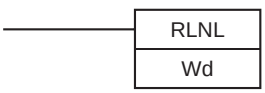
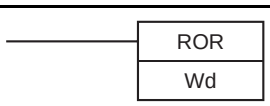
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
MOVE	MOV @MOV !MOV !@MOV	021		S: 源	WORD	---
				D: 目的地	WORD	---
DOUBLE MOVE	MOVL @MOVL	498		S: 第一个源字	DWORD	---
				D: 第一个目的地字	DWORD	---

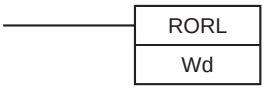
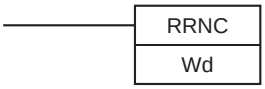
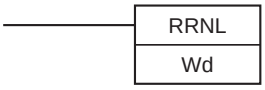
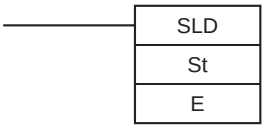
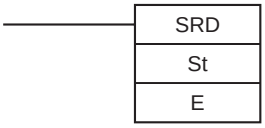
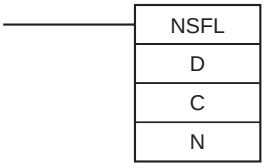
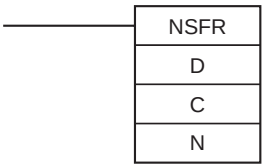
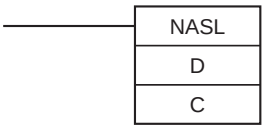
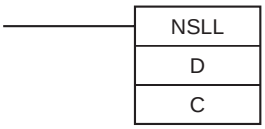
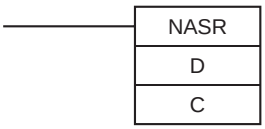
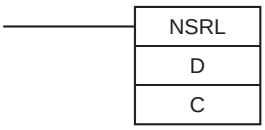
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
MOVE NOT	MVN @MVN	022		S: 源	WORD	---
				D: 目的地	WORD	---
DOUBLE MOVE NOT	MVNL @MVNL	499		S: 第一个源字	DWORD	---
				D: 第一个目的地字	DWORD	---
MOVE BIT	MOVB @MOVB	082		S: 源字或数据	WORD	---
				C: 控制字	UINT	---
				D: 目的地字	WORD	---
MOVE DIGIT	MOVD @MOVD	083		S: 源字或数据	WORD	---
				C: 控制字	UINT	---
				D: 目的地字	UINT	---
MULTIPLE BIT TRANSFER 亏	XFRB @XFRB	062		C: 控制字	UINT	---
				S: 第一个源字	WORD	有 （不固定）
				D: 第一个目的地字	WORD	有 （不固定）
BLOCK TRANSFER	XFER @XFER	070		N: 字数	UINT	---
				S: 第一个源字	WORD	有 （不固定）
				D: 第一个目的地字	WORD	有 （不固定）
BLOCK SET	BSET @BSET	071		S: 源字	WORD	---
				St: 起始字	WORD	有 （不固定）
				E: 结束字	WORD	有 （不固定）
DATA EXCHANGE	XCHG @XCHG	073		E1: 第一个交换字	WORD	---
				E2: 第二个交换字	WORD	---
DOUBLE DATA EXCHANGE	XCGL @XCGL	562		E1: 第一个交换字	DWORD	---
				E2: 第二个交换字	DWORD	---

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
SINGLE WORD DISTRIBUTE	DIST @DIST	080		S: 源字	WORD	---
				Bs: 目的地基地址	WORD	有 （不固定）
				Of: 偏移	UINT	---
DATA COLLECT	COLL @COLL	081		Bs: 目的地基地址	WORD	有 （不固定）
				Of: 偏移	WORD	---
				D: 目的地字	WORD	---
MOVE TO REGISTER	MOVR @MOVR	560		S: 源 （所需的字轨道）	BOOL	---
				D: 目的地 （变址寄存器）	WORD	---
MOVE TIMER/ COUNTER PV TO REGISTER	MOVRW @MOVRW	561	功能块中不支持	S: 源 （所需的 TC 数）	---	---
				D: 目的地 （变址寄存器）	---	---

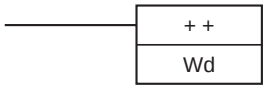
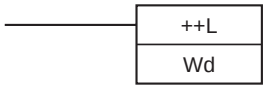
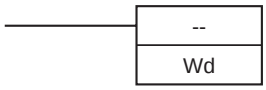
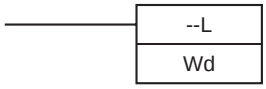
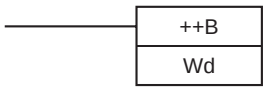
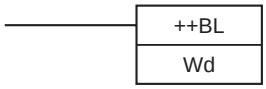
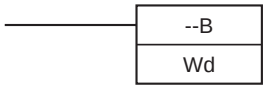
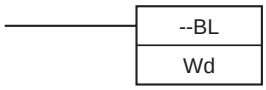
2-6-7 数据转换指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
SHIFT REGISTER	SFT	010		St: 起始字	UINT	有 （不固定）
				E: 结束字	UINT	有 （不固定） 当需要数组变量时， D1 和 D2 必须为相同的数组变量。
REVERSIBLE SHIFT REGISTER	SFTR @SFTR	084		C: 控制字	UINT	---
				St: 起始字	UINT	有 （不固定）
				E: 结束字	UINT	有 （不固定） 当需要数组变量时， D1 和 D2 必须为相同的数组变量。

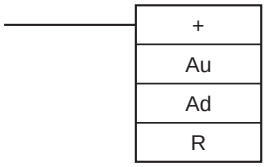
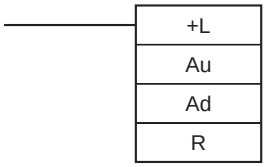
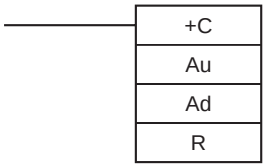
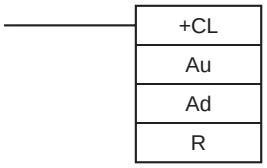
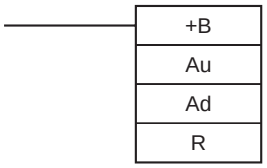
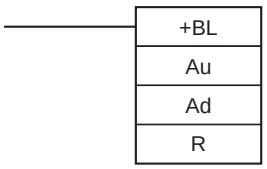
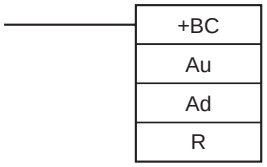
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
ASYNCHRONOUS SHIFT REGISTER	ASFT @ASFT	017		C: 控制字	UINT	---
				St: 起始字	UINT	有 （不固定）
				E: 结束字	UINT	有 （不固定） 当需要数组变量时， D1 和 D2 必须为相同的数组变量。
WORD SHIFT	WSFT @WSFT	016		S: 控制字	WORD	---
				St: 起始字	UINT	有 （不固定）
				E: 结束字	UINT	有 （不固定） 当需要数组变量时， D1 和 D2 必须为相同的数组变量。
ARITHMETIC SHIFT LEFT	ASL @ASL	025		Wd: 字	UINT	---
DOUBLE SHIFT LEFT	ASLL @ASLL	570		Wd: 字	UDINT	---
ARITHMETIC SHIFT RIGHT	ASR @ASR	026		Wd: 字	UINT	---
DOUBLE SHIFT RIGHT	ASRL @ASRL	571		Wd: 字	UDINT	---
ROTATE LEFT	ROL @ROL	027		Wd: 字	UINT	---
DOUBLE ROTATE LEFT	ROLL @ROLL	572		Wd: 字	UDINT	---
ROTATE LEFT WITHOUT CARRY	RLNC @RLNC	574		Wd: 字	UINT	---
DOUBLE ROTATE LEFT WITHOUT CARRY	RLNL @RLNL	576		Wd: 字	UDINT	---
ROTATE RIGHT	ROR @ROR	028		Wd: 字	UINT	---

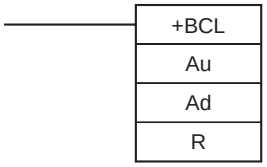
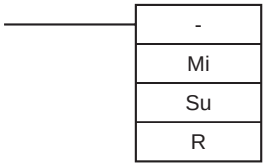
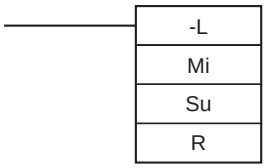
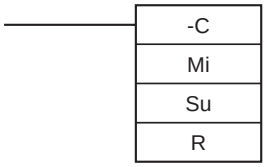
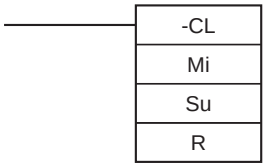
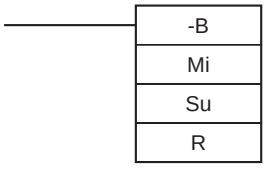
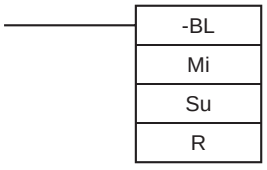
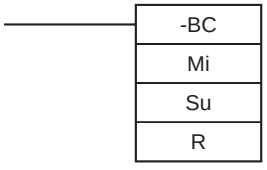
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
DOUBLE ROTATE RIGHT	RORL @RORL	573		Wd: 字	UDINT	---
ROTATE RIGHT WITHOUT CARRY	RRNC @RRNC	575		Wd: 字	UINT	---
DOUBLE ROTATE RIGHT WITHOUT CARRY	RRNL @RRNL	577		Wd: 字	UDINT	---
ONE DIGIT SHIFT LEFT	SLD @SLD	074		St: 起始字	UINT	有 （不固定）
				E: 结束字	UINT	有 （不固定）
ONE DIGIT SHIFT RIGHT	SRD @SRD	075		St: 起始字	UINT	有 （不固定）
				E: 结束字	UINT	有 （不固定）
SHIFT N-BIT DATA LEFT	NSFL @NSFL	578		D: 转换起始字	UINT	有 （不固定）
				C: 起始位	UINT	---
				N: 转换数据长度	UINT	---
SHIFT N-BIT DATA RIGHT	NSFR @NSFR	579		D: 转换起始字	UINT	有 （不固定）
				C: 起始位	UINT	---
				N: 转换数据长度	UINT	---
SHIFT N-BITS LEFT	NASL @NASL	580		D: 转换字	UINT	---
				C: 控制字	UINT	---
DOUBLE SHIFT N-BITS LEFT	NSLL @NSLL	582		D: 转换字	UDINT	---
				C: 控制字	UINT	---
SHIFT N-BITS RIGHT	NASR @NASR	581		D: 转换字	UINT	---
				C: 控制字	UINT	---
DOUBLE SHIFT N-BITS RIGHT	NSRL @NSRL	583		D: 转换字	UDINT	---
				C: 控制字	UINT	---

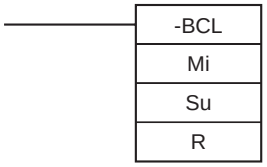
2-6-8 递增 / 递减指令

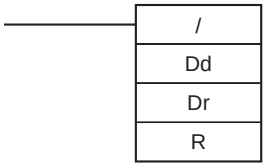
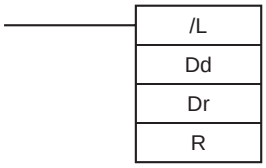
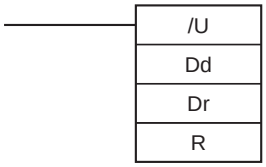
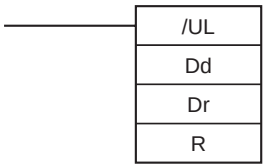
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量 （所需的字数据量示于括号中）
INCREMENT BINARY	++ @++	590		Wd: 字	UINT	---
DOUBLE INCREMENT BINARY	++L @++L	591		Wd: 字	UDINT	---
DECREMENT BINARY	-- @--	592		Wd: 字	UINT	---
DOUBLE DECREMENT BINARY	--L @--L	593		Wd: 第一个字	UDINT	---
INCREMENT BCD	++B @++B	594		Wd: 字	WORD	---
DOUBLE INCREMENT BCD	++BL @++BL	595		Wd: 第一个字	DWORD	---
DECREMENT BCD	--B @--B	596		Wd: 字	DWORD	---
DOUBLE DECREMENT BCD	--BL @--BL	597		Wd: 第一个字	WORD	---

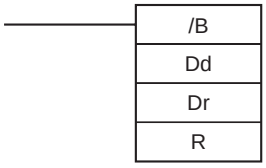
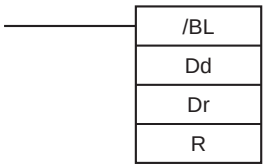
2-6-9 符号数学指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
SIGNED BINARY ADD WITHOUT CARRY	+ @+	400		Au: 被加数字	INT	---
				Ad: 加数字	INT	---
				R: 结果字	INT	---
DOUBLE SIGNED BINARY ADD WITHOUT CARRY	+L @+L	401		Au: 第一个被加数字	DINT	---
				Ad: 第一个被加数字	DINT	---
				R: 第一个结果字	DINT	---
SIGNED BINARY ADD WITH CARRY	+C @+C	402		Au: 被加数字	INT	---
				Ad: 加数字	INT	---
				R: 结果字	INT	---
DOUBLE SIGNED BINARY ADD WITH CARRY	+CL @+CL	403		Au: 第一个被加数字	DINT	---
				Ad: 第一个加数字	DINT	---
				R: 第一个结果字	DINT	---
BCD ADD WITHOUT CARRY	+B @+B	404		Au: 被加数字	WORD	---
				Ad: 加数字	WORD	---
				R: 结果字	WORD	---
DOUBLE BCD ADD WITHOUT CARRY	+BL @+BL	405		Au: 被加数字	DWORD	---
				Ad: 加数字	DWORD	---
				R: 结果字	DWORD	---
BCD ADD WITH CARRY	+BC @+BC	406		Au: 被加数字	WORD	---
				Ad: 加数字	WORD	---
				R: 结果字	WORD	---

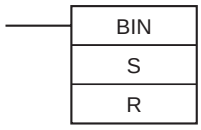
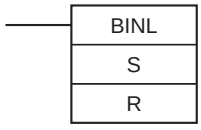
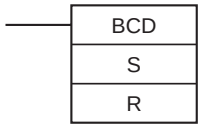
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
DOUBLE BCD ADD WITH CARRY	+BCL @+BCL	407		Au: 第一个被加数字	DWORD	---
				Ad: 第一个加数字	DWORD	---
				R: 第一个结果字	DWORD	---
SIGNED BINARY SUBTRACT WITHOUT CARRY	- @-	410		Mi: 被减数字	INT	---
				Su: 减数字	INT	---
				R: 结果字	INT	---
DOUBLE SIGNED BINARY SUBTRACT WITHOUT CARRY	-L @-L	411		Mi: 被减数字	DINT	---
				Su: 减数字	DINT	---
				R: 结果字	DINT	---
SIGNED BINARY SUBTRACT WITH CARRY	-C @-C	412		Mi: 被减数字	INT	---
				Su: 减数字	INT	---
				R: 结果字	INT	---
DOUBLE SIGNED BINARY WITH CARRY	-CL @-CL	413		Mi: 被减数字	DINT	---
				Su: 减数字	DINT	---
				R: 结果字	DINT	---
BCD SUBTRACT WITHOUT CARRY	-B @-B	414		Mi: 被减数字	WORD	---
				Su: 减数字	WORD	---
				R: 结果字	WORD	---
DOUBLE BCD SUBTRACT WITHOUT CARRY	-BL @-BL	415		Mi: 被减数字	DWORD	---
				Su: 减数字	DWORD	---
				R: 结果字	DWORD	---
BCD SUBTRACT WITH CARRY	-BC @-BC	416		Mi: 被减数字	WORD	---
				Su: 减数字	WORD	---
				R: 结果字	WORD	---

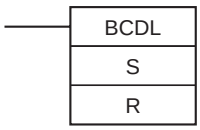
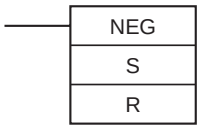
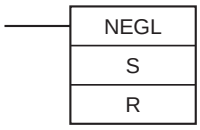
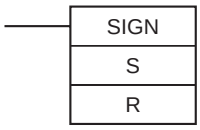
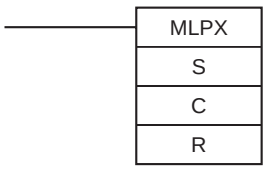
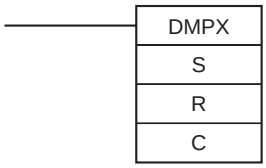
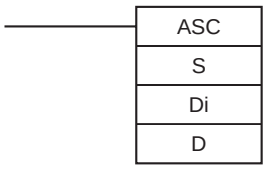
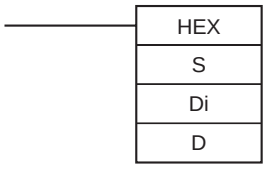
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量 (所需的字数据量示于括号中)
DOUBLE BCD SUBTRACT WITH CARRY	-BCL @-BCL	417		Mi: 第一个被减数字	DWORD	---
				Su: 第一个减数字	DWORD	---
				R: 结果字	DWORD	---
SIGNED BINARY MULTIPLY	* @*	420		Md: 被乘数字	INT	---
				Mr: 乘数字	INT	---
				R: 结果字	DINT	---
DOUBLE SIGNED BINARY MULTIPLY	*L @*L	421		Md: 第一个被乘数字	DINT	---
				Mr: 第一个乘数字	DINT	---
				R: 第一个结果字	LINT	---
UNSIGNED BINARY MULTIPLY	*U @*U	422		Md: 被乘数字	UINT	---
				Mr: 乘数字	UINT	---
				R: 结果字	UINT	---
DOUBLE UNSIGNED BINARY MULTIPLY	*UL @*UL	423		Md: 第一个被乘数字	UDINT	---
				Mr: 第一个乘数字	UDINT	---
				R: 第一个结果字	ULINT	---
BCD MULTIPLY	*B @*B	424		Md: 被乘数字	WORD	---
				Mr: 乘数字	WORD	---
				R: 结果字	DWORD	---
DOUBLE BCD MULTIPLY	*BL @*BL	425		Md: 第一个被乘数字	DWORD	---
				Mr: 第一个乘数字	DWORD	---
				R: 第一个结果字	LWORD	---

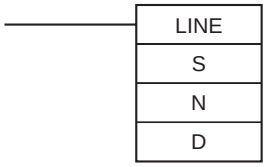
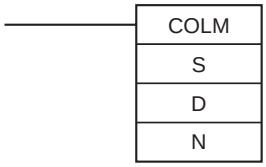
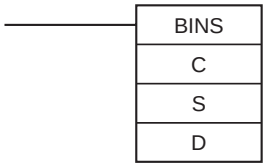
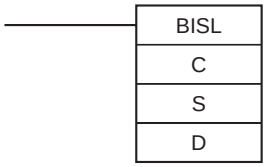
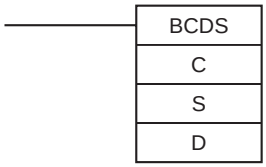
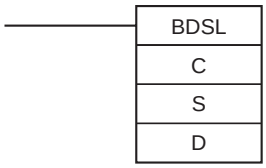
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
SIGNED BINARY DIVIDE	/ @/	430		Dd: 被除数字	INT	---
				Dr: 除数字	INT	---
				R: 结果字	DWORD	有 (2) 当需要数组变量时必须使用 INT。
DOUBLE SIGNED BINARY DIVIDE	/L @/L	431		Dd: 第一个被除数字	DINT	---
				Dr: 第一个除数字	DINT	---
				R: 第一个结果字	LWORD	有 (2) 当需要数组变量时必须使用 DINT。
UNSIGNED BINARY DIVIDE	/U @/U	432		Dd: 被除数字	UINT	---
				Dr: 除数字	UINT	---
				R: 结果字	DWORD	有 (2) 当需要数组变量时必须使用 UINT。
DOUBLE UNSIGNED BINARY DIVIDE	/UL @/UL	433		Dd: 第一个被除数字	UDINT	---
				Dr: 第一个除数字	UDINT	---
				R: 第一个结果字	LWORD	有 (2) 当需要数组变量时必须使用 UDINT。

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
BCD DIVIDE	/B @/B	434		Dd: 被除数字	WORD	---
				Dr: 除数字	WORD	---
				R: 结果字	DWORD	有 （2） 当需要数组变量时必须使用 WORD。
DOUBLE BCD DIVIDE	/BL @/BL	435		Dd: 第一个被除数字	DWORD	---
				Dr: 第一个除数字	DWORD	---
				R: 第一个结果字	LWORD	有 （2） 当需要数组变量时必须使用 DWORD。

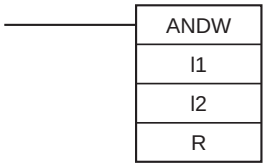
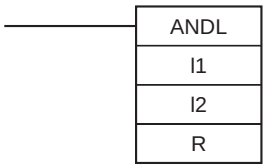
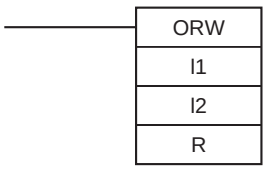
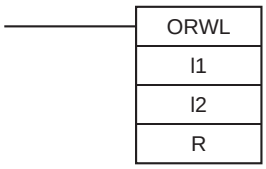
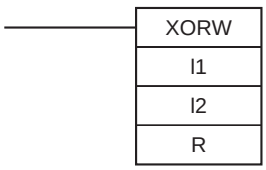
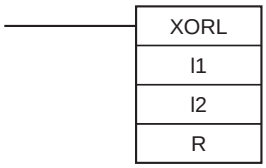
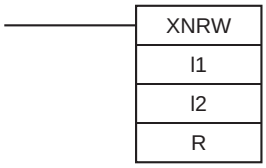
2-6-10 转换指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
BCD-TO-BINARY	BIN @BIN	023		S: 源字	WORD	---
				R: 结果字	UINT	---
DOUBLE BCD-TO-DOUBLE BINARY	BINL @BINL	058		S: 第一个源字	DWORD	---
				R: 第一个结果字	UDINT	---
BINARY-TO-BCD	BCD @BCD	024		S: 源字	UINT	---
				R: 结果字	WORD	---

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
DOUBLE BINARY-TO-DOUBLE BCD	BCDL @BCDL	059		S: 第一个源字	UDINT	---
				R: 第一个结果字	DWORD	---
2 謝 COMPLEMENT	NEG @NEG	160		S: 源字	WORD	---
				R: 结果字	UINT	---
DOUBLE 2 謝 COMPLEMENT	NEGL @NEGL	161		S: 第一个源字	DWORD	---
				R: 第一个结果字	UDINT	---
16-BIT TO 32-BIT SIGNED BINARY	SIGN @SIGN	600		S: 源字	WORD	---
				R: 第一个结果字	DINT	---
DATA DECODER	MLPX @MLPX	076		S: 源字	UINT	---
				C: 控制字	UINT	---
				R: 第一个结果字	UINT	有 （不固定）
DATA ENCODER	DMPX @DMPX	077		S: 第一个源字	UINT	有 （不固定）
				R: 结果字	UINT	---
				C: 控制字	UINT	---
ASCII CONVERT	ASC @ASC	086		S: 源字	UINT	---
				Di: 数字标志符	UINT	---
				D: 第一个目的地字	UINT	有 (3)
ASCII TO HEX	HEX @HEX	162		S: 源字	UINT	有 (2)
				Di: 数字标志符	UINT	---
				D: 目的地字	UINT	有 （不固定）

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量 (所需的字数据量示于括号中)
COLUMN TO LINE	LINE @LINE	063		S: 第一个源字	WORD	有 (16)
				N: 位数	UINT	---
				D: 目的地字	UINT	---
LINE TO COLUMN	COLM @COLM	064		S: 源字	WORD	---
				D: 第一个目的地字	WORD	是 (16)
				N: 位数	UINT	---
SIGNED BCD-TO-BINARY	BINS @BINS	470		C: 控制字	UINT	---
				S: 源字	WORD	---
				D: 目的地字	INT	---
DOUBLE SIGNED BCD-TO-BINARY	BISL @BISL	472		C: 控制字	UINT	---
				S: 第一个源字	DWORD	---
				D: 第一个目的地字	DINT	---
SIGNED BINARY-TO-BCD	BCDS @BCDS	471		C: 控制字	UINT	---
				S: 源字	INT	---
				D: 目的地字	WORD	---
DOUBLE SIGNED BINARY-TO-BCD	BDSL @BDSL	473		C: 控制字	UINT	---
				S: 第一个源字	DINT	---
				D: 第一个目的地字	DWORD	---

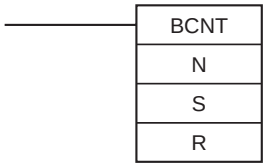
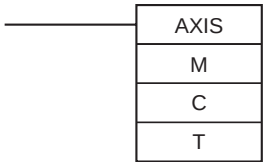
2-6-11 逻辑指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量 (所需的字数据量示于括号中)
LOGICAL AND	ANDW @ANDW	034		I1: 输入 1 I2: 输入 2 R: 结果字	WORD WORD WORD	--- --- ---
DOUBLE LOGICAL AND	ANDL @ANDL	610		I1: 输入 1 I2: 输入 2 R: 结果字	DWORD DWORD DWORD	--- --- ---
LOGICAL OR	ORW @ORW	035		I1: 输入 1 I2: 输入 2 R: 结果字	WORD WORD WORD	--- --- ---
DOUBLE LOGICAL OR	ORWL @ORWL	611		I1: 输入 1 I2: 输入 2 R: 结果字	DWORD DWORD DWORD	--- --- ---
EXCLUSIVE OR	XORW @XORW	036		I1: 输入 1 I2: 输入 2 R: 结果字	WORD WORD WORD	--- --- ---
DOUBLE EXCLUSIVE OR	XORL @XORL	612		I1: 输入 1 I2: 输入 2 R: 结果字	DWORD DWORD DWORD	--- --- ---
EXCLUSIVE NOR	XNRW @XNRW	037		I1: 输入 1 I2: 输入 2 R: 结果字	WORD WORD WORD	--- --- ---

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量 (所需的字数据量示于括号中)
DOUBLE EXCLUSIVE NOR	XNRL @XNRL	613	XNRL I1 I2 R	I1: 输入 1	DWORD	---
				I2: 输入 2	DWORD	---
				R: 结果字	DWORD	---
COMPLEMENT	COM @COM	029	COM Wd	Wd: 字	WORD	---
DOUBLE COMPLEMENT	COML @COML	614	COML Wd	Wd: 字	DWORD	---

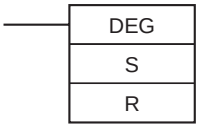
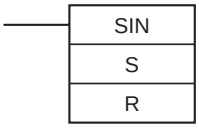
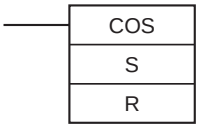
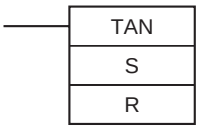
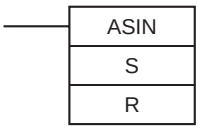
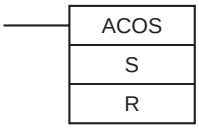
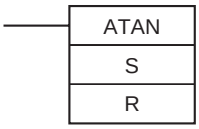
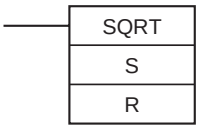
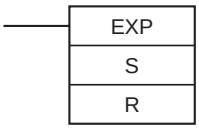
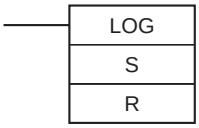
2-6-12 特殊数学指令

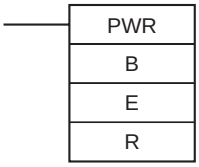
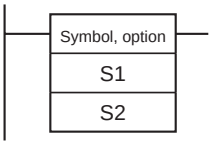
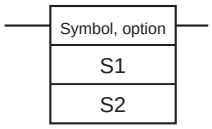
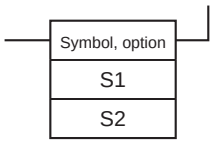
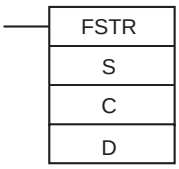
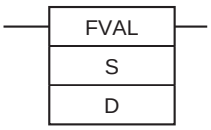
指令	助记符	功能码	符号 I	操作数	支持的变量数据类型	所需的 AT 设置或数组变量 (所需的字数据量示于括号中)
BINARY ROOT	ROTB @ROTB	620	ROTB S R	S: 第一个源字	UDINT	---
				R: 结果字	UINT	---
BCD SQUARE ROOT	ROOT @ROOT	072	ROOT S R	S: 第一个源字	DWORD	---
				R: 结果字	WORD	---
ARITHMETIC PROCESS	APR @APR	069	APR C S R	C: 控制字	UINT	Yes (not fixed)
				S: 源字	WORD	---
				R: 结果字	WORD	---
FLOATING POINT DIVIDE	FDIV @FDIV	079	FDIV Dd Dr R	Dd: 第一个被除数字	UDINT	---
				Dr: 第一个除数字	UDINT	---
				R: 第一个结果字	UDINT	---

指令	助记符	功能码	符号 I	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
BIT COUNTER	BCNT @BCNT	067		N: 字数	UINT	---
				S: 第一个源字	UINT	有 （不固定）
				R: 结果字	UINT	---
VIRTUAL AXIS	AXIS (FQM1 only)	981		M: 模式标志	WORD	---
				C: 处理周期	WORD	---
				T: 第一个设定值表字	WORD	有 (27)

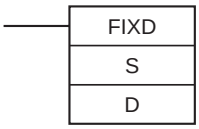
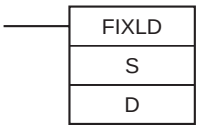
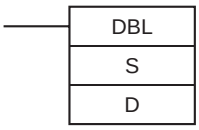
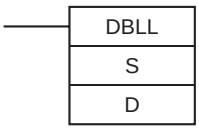
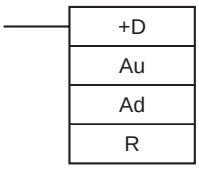
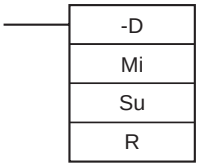
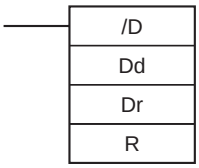
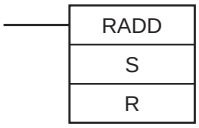
2-6-13 浮点数学指令

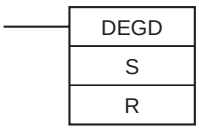
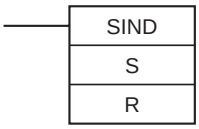
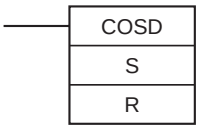
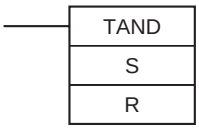
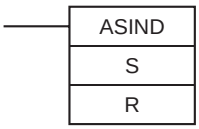
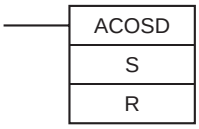
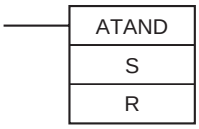
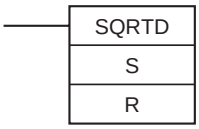
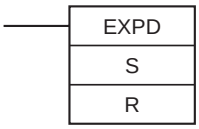
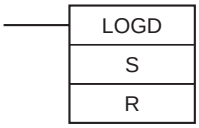
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
FLOATING TO 16-BIT	FIX @FIX	450	FIX S R	S: 第一个源字	REAL	---
				R: 结果字	INT	---
FLOATING TO 32-BIT	FIXL @FIXL	451	FIXL S R	S: 第一个源字	REAL	---
				R: 结果字	DINT	---
16-BIT TO FLOATING	FLT @FLT	452	FLT S R	S: 源字	INT	---
				R: 第一个结果字	REAL	---
32-BIT TO FLOATING	FLTL @FLTL	453	FLTL S R	S: 第一个源字	DINT	---
				R: 结果字	REAL	---
FLOATING-POINT ADD	+F @+F	454	+F Au Ad R	Au: 第一个被加数字	REAL	---
				Ad: 第一个加数字	REAL	---
				R: 第一个结果字	REAL	---
FLOATING-POINT SUBTRACT	-F @-F	455	-F Mi Su R	Mi: 第一个被减数字	REAL	---
				Su: 第一个减数字	REAL	---
				R: 第一个结果字	REAL	---
FLOATING- POINT MULTIPLY	*F @*F	456	* F Md Mr R	Md: 第一个被乘数字	REAL	---
				Mr: 第一个乘数字	REAL	---
				R: 第一个结果字	REAL	---
FLOATING- POINT DIVIDE	/F @/F	457	/F Dd Dr R	Dd: 第一个被除数字	REAL	---
				Dr: 第一个除数字	REAL	---
				R: 第一个结果字	REAL	---
DEGREES TO RADIANS	RAD @RAD	458	RAD S R	S: 第一个源字	REAL	---
				R: 第一个结果字	REAL	---

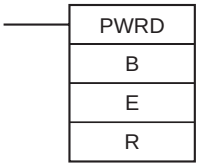
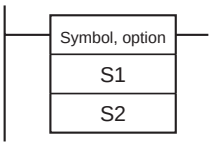
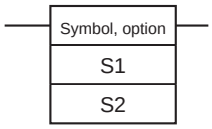
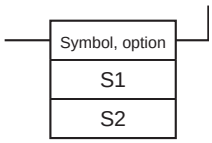
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
RADIANS TO DEGREES	DEG @DEG	459		S: 第一个源字 R: 第一个结果字	REAL REAL	--- ---
SINE	SIN @SIN	460		S: 第一个源字 R: 第一个结果字	REAL REAL	--- ---
COSINE	COS @COS	461		S: 第一个源字 R: 第一个结果字	REAL REAL	--- ---
TANGENT	TAN @TAN	462		S: 第一个源字 R: 第一个结果字	REAL REAL	--- ---
ARC SINE	ASIN @ASIN	463		S: 第一个源字 R: 第一个结果字	REAL REAL	--- ---
ARC COSINE	ACOS @ACOS	464		S: 第一个源字 R: 第一个结果字	REAL REAL	--- ---
ARC TANGENT	ATAN @ATAN	465		S: 第一个源字 R: 第一个结果字	REAL REAL	--- ---
SQUARE ROOT	SQRT @SQRT	466		S: 第一个源字 R: 第一个结果字	REAL REAL	--- ---
EXPONENT	EXP @EXP	467		S: 第一个源字 R: 第一个结果字	REAL REAL	--- ---
LOGARITHM	LOG @LOG	468		S: 第一个源字 R: 第一个结果字	REAL REAL	--- ---

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
EXPONENTIAL POWER	PWR @PWR	840		B: 第一个基字	REAL	---
				E: 第一个幂数字	REAL	---
				R: 第一个结果字	REAL	---
Floating Symbol Comparison	LD, AND, OR + =F, <>F, <F, <=F, >F, >=F	329 (=F) 330 (<>F) 331 (<F) 332 (<=F) 333 (>F) 334 (>=F)	Using LD:  Using AND:  Using OR: 	S1: 比较数据 1	REAL	---
				S2: 比较数据 2	REAL	---
FLOATING- POINT TO ASCII	FSTR @FSTR	448		S: 第一个源字	REAL	---
				C: 控制字	UINT	有 (3)
				D: 目的地字	UINT	有 （不固定）
ASCII TO FLOATING-POINT	FVAL @FVAL	449		S: 源字	UINT	有 （不固定）
				D: 第一个目的地字	REAL	---

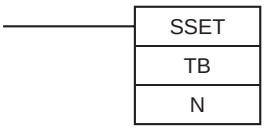
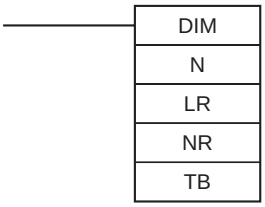
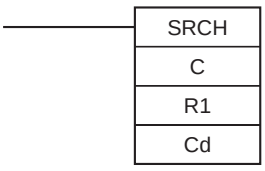
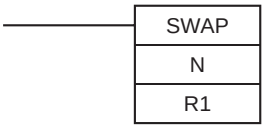
2-6-14 双倍精度浮点指令

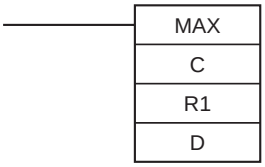
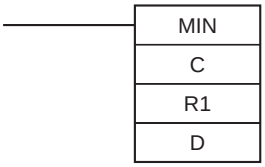
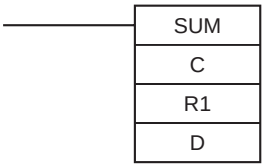
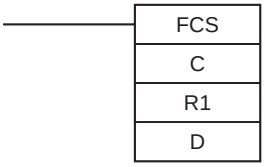
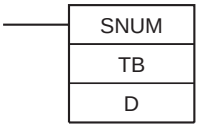
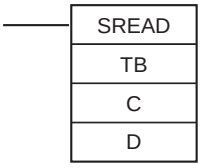
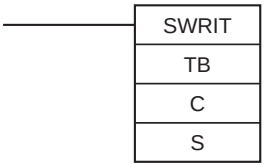
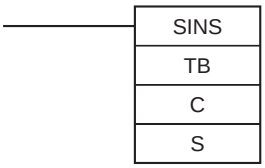
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
DOUBLE FLOATING TO 16-BIT BINARY	FIXD @FIXD	841		S: 第一个源字	LREAL	---
				D: 目的地字	INT	---
DOUBLE FLOATING TO 32-BIT BINARY	FIXLD @FIXLD	842		S: 第一个源字	LREAL	---
				D: 第一个目的地字	DINT	---
16-BIT BINARY TO DOUBLE FLOATING	DBL @DBL	843		S: 源字	INT	---
				D: 第一个目的地字	LREAL	---
32-BIT BINARY TO DOUBLE FLOATING	DBLL @DBLL	844		S: 第一个源字	DINT	---
				D: 第一个目的地字	DINT	---
DOUBLE FLOATING-POINT ADD	+D @+D	845		Au: 第一个被加数字	LREAL	---
				Ad: 第一个加数字	LREAL	---
				R: 第一个结果字	LREAL	---
DOUBLE FLOATING-POINT SUBTRACT	-D @-D	846		Mi: 第一个被减数字	LREAL	---
				Su: 第一个减数字	LREAL	---
				R: 第一个结果字	LREAL	---
DOUBLE FLOATING-POINT MULTIPLY	*D @*D	847		Md: 第一个被乘数字	LREAL	---
				Mr: 第一个乘数字	LREAL	---
				R: 第一个结果字	LREAL	---
DOUBLE FLOATING-POINT DIVIDE	/D @/D	848		Dd: 第一个被除数字	LREAL	---
				Dr: 第一个除数字	LREAL	---
				R: 第一个结果字	LREAL	---
DOUBLE DEGREES TO RADIANS	RADD @RADD	849		S: 第一个源字	LREAL	---
				R: 第一个结果字	LREAL	---

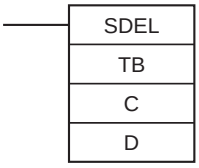
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
DOUBLE RADIANS TO DEGREES	DEGD @DEGD	850		S: 第一个源字 R: 第一个结果字	LREAL LREAL	--- ---
DOUBLE SINE	SIND @SIND	851		S: 第一个源字 R: 第一个结果字	LREAL LREAL	--- ---
DOUBLE COSINE	COSD @COSD	852		S: 第一个源字 R: 第一个结果字	LREAL LREAL	--- ---
DOUBLE TANGENT	TAND @TAND	853		S: 第一个源字 R: 第一个结果字	LREAL LREAL	--- ---
DOUBLE ARC SINE	ASIND @ASIND	854		S: 第一个源字 R: 第一个结果字	LREAL LREAL	--- ---
DOUBLE ARC COSINE	ACOSD @ACOSD	855		S: 第一个源字 R: 第一个结果字	LREAL LREAL	--- ---
DOUBLE ARC TANGENT	ATAND @ATAND	856		S: 第一个源字 R: 第一个结果字	LREAL LREAL	--- ---
DOUBLE SQUARE ROOT	SQRTD @SQRTD	857		S: 第一个源字 R: 第一个结果字	LREAL LREAL	--- ---
DOUBLE EXPONENT	EXPD @EXPD	858		S: 第一个源字 R: 第一个结果字	LREAL LREAL	--- ---
DOUBLE LOGARITHM	LOGD @LOGD	859		S: 第一个源字 R: 第一个结果字	LREAL LREAL	--- ---

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
DOUBLE EXPONENTIAL POWER	PWRD @PWRD	860		B: 第一个基字	LREAL	---
				E: 第一个幂数字	LREAL	---
				R: 第一个结果字	LREAL	---
DOUBLE SYMBOL COMPARISON	LD, AND, OR + =D, <>D, <D, <=D, >D, >=D	335 (=D) 336 (<>D) 337 (<D) 338 (<=D) 339 (>D) 340 (>=D)	使用 LD:  使用 AND:  使用 OR: 	S1: 比较数据 1	LREAL	---
				S2: 比较数据 2	LREAL	---

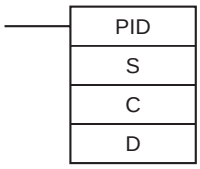
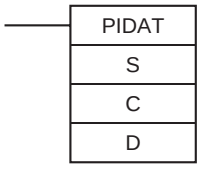
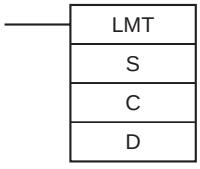
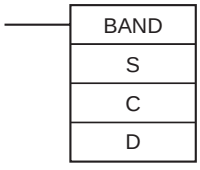
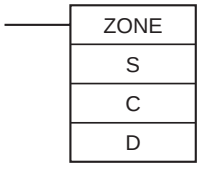
2-6-15 表数据处理指令

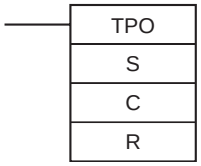
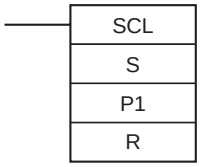
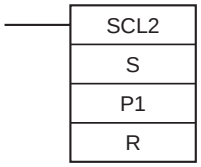
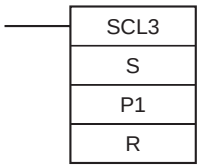
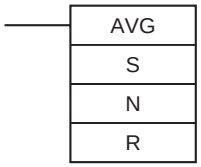
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
SET STACK	SSET @SSET	630		TB: 第一个栈地址	UINT	有 （不固定）
				N: 字数	UINT	---
PUSH ONTO STACK	PUSH @PUSH	632	在功能块中不支持	TB 第一个栈地址	---	---
				S: 源字	---	---
FIRST IN FIRST OUT	FIFO @FIFO	633	在功能块中不支持	TB: 第一个栈地址	---	---
				D: 目的地字	---	---
LAST IN FIRST OUT	LIFO @LIFO	634	在功能块中不支持	TB: 第一个栈地址	---	---
				D: 目的地字	---	---
DIMENSION RECORD TABLE	DIM @DIM	631		N: 表号	#+ 十进制小数 （仅）	---
				LR: 每个记录的长度	UINT	---
				NR: 记录数	UINT	---
				TB: 第一个表字	UINT	有 （不固定）
SET RECORD LOCATION	SETR @SETR	635	在功能块中不支持	N: 表号	---	---
				R: 记录数	---	---
				D: 目的地变址寄存器	---	---
GET RECORD NUMBER	GETR @GETR	636	在功能块中不支持	N: 表号	---	---
				IR: 变址寄存器	---	---
				D: 目的地字	---	---
DATA SEARCH	SRCH @SRCH	181		C: 第一个控制字	UDINT	---
				R1: 范围内的第一个字	UINT	有 （不固定）
				Cd: 比较数据	WORD	---
SWAP BYTES	SWAP @SWAP	637		N: 字数	UINT	---
				R1: 范围内的第一个字	UINT	有 （不固定）

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
FIND MAXIMUM	MAX @MAX	182		C: 第一个控制字	UDINT	---
				R1: 范围内的第一个字	UINT	有 （不固定）
				D: 目的地字	UINT	---
FIND MINIMUM	MIN @MIN	183		C: 第一个控制字	UDINT	---
				R1: 范围内的第一个字	UINT	有 （不固定）
				D: 目的地字	UINT	---
SUM	SUM @SUM	184		C: 第一个控制字	UDINT	---
				R1: 范围内的第一个字	UINT	有 （不固定）
				D: 目的地字	UDINT	---
FRAME CHECK SUM	FCS @FCS	180		C: 第一个控制字	UDINT	---
				R1: 范围内的第一个字	UINT	有 （不固定）
				D: 目的地字	UINT	---
STACK SIZE READ	SNUM @SNUM	638		TB: 第一个栈地址	UINT	有 （不固定）
				D: 目的地字	UINT	---
STACK DATA READ	SREAD @SREAD	639		TB: 第一个栈地址	UINT	有 （不固定）
				C: 偏离值	UINT	---
				D: 目的地字	UINT	---
STACK DATA OVERWRITE	SWRIT @SWRIT	640		TB: 第一个栈地址	UINT	有 （不固定）
				C: 偏离值	UINT	---
				S: 源数据	UINT	---
STACK DATA INSERT	SINS @SINS	641		TB: 第一个栈地址	UINT	有 （不固定）
				C: 偏离值	UINT	---
				S: 源数据	UINT	---

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
STACK DATA DELETE	SDEL @SDEL	642		TB: 第一个控制字	UINT	有 （不固定）
				C: 范围内的第一个字	UINT	---
				D: 目的地字	UINT	---

2-6-16 数据控制指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
PID CONTROL	PID	190		S: 输入字	UINT	---
				C: 第一个参数字	WORD	有 (39)
				D: 输出字	UINT	---
PID CONTROL WITH AUTO TUNING	PIDAT	191		S: 输入字	UINT	---
				C: 第一个参数字	WORD	有 (41)
				D: 输出字	UINT	---
LIMIT CONTROL	LMT @LMT	680		S: 输入字	INT	---
				C: 第一个参数字	DINT	有 (2)
				D: 输出字	INT	---
DEAD BAND CONTROL	BAND @BAND	681		S: 输入字	INT	---
				C: 第一个参数字	UINT	有 (2)
				D: 输出字	UINT	---
DEAD ZONE CONTROL	ZONE @ZONE	682		S: 输入字	INT	---
				C: 第一个参数字	UDINT	有 (2)
				D: 输出字	UINT	---

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
TIME-PROPORTIONAL OUTPUT	TPO	685		S: 输入字	UINT	---
				C: 第一个参数字	WORD	有 (7)
				R: 脉冲输出位	BOOL	---
SCALING	SCL @SCL	194		S: 输入字	UINT	---
				P1: 第一个参数字	LWORD	有 (4)
				R: 结果字	WORD	---
SCALING 2	SCL2 @SCL2	486		S: 输入字	INT	---
				P1: 第一个参数字	WORD	有 (3)
				R: 结果字	WORD	---
SCALING 3	SCL3 @SCL3	487		S: 输入字	WORD	---
				P1: 第一个参数字	WORD	有 (5)
				R: 结果字	INT	---
AVERAGE	AVG	195		S: 输入字	UINT	---
				N: 循环数	UINT	---
				R: 结果字	UINT	有 （不固定）

2-6-17 子程序指令

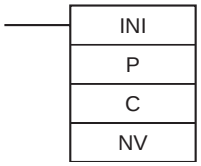
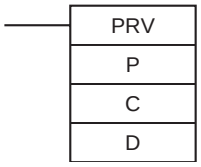
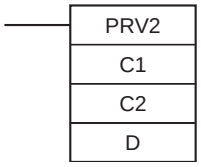
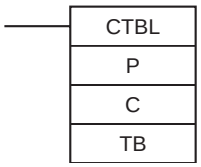
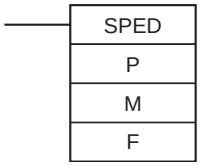
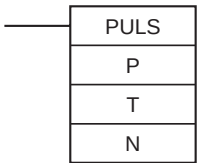
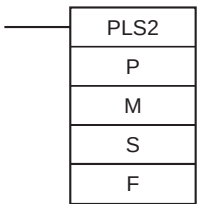
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
SUBROUTINE CALL	SBS @SBS	091	功能块中不支持	N：子程序号	---	---
SUBROUTINE ENTRY	SBN	092	功能块中不支持	N：子程序号	---	---
SUBROUTINE RETURN	RET	093	功能块中不支持		---	---

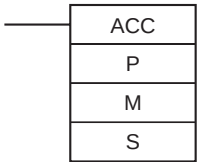
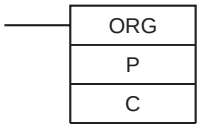
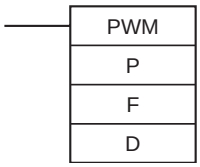
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
MACRO	MCRO @MCRO	099	功能块中不支持	N：子程序号	---	---
				S: 中断数据	---	---
				D: 第一个输出参数字	---	---
GLOBAL SUBROUTINE CALL	GSBS @GSBS	750	功能块中不支持	N：子程序号	---	---
GLOBAL SUBROUTINE ENTRY	GSBN	751	功能块中不支持	N：子程序号	---	---
GLOBAL SUBROUTINE RETURN	GRET	752	功能块中不支持		---	---
JUMP TO SUBROUTINE	JSB (FQM1 only)	982	功能块中不支持	---	---	---

2-6-18 中断控制指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
SET INTERRUPT MASK	MSKS @MSKS	690	MSKS N S	N: 中断识别符	# 十进制小数（仅）	---
				S: 中断数据	UINT	---
READ INTERRUPT MASK	MSKR @MSKR	692	MSKR N D	N: 中断识别符	# 十进制小数（仅）	---
				D: 目的地字	UINT	---
CLEAR INTERRUPT	CLI @CLI	691	CLI N S	N: 中断识别符	# 十进制小数（仅）	---
				S: 中断数据	UINT	---
DISABLE INTERRUPTS	DI @DI	693	DI	---	---	---
ENABLE INTERRUPTS	EI	694	EI	---	---	---
INTERVAL TIMER	STIM @STIM (FQM1 only)	980	STIM C1 C2 C3	C1: 控制数据 #1	WORD	---
				C2: 控制数据 #2	INT	---
				C3: 控制数据 #3	INT	---

2-6-19 高速计数器和脉冲输出指令 （仅 CJ1M-CPU21/22/23）

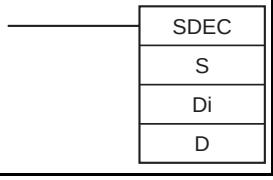
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
MODE CONTROL	INI @INI	880		P: 端口区分符	WORD	---
				C: 控制数据	UINT	---
				NV: 有新 PCV 的第一个字	DWORD	---
HIGH-SPEED COUNTER PV READ	PRV @PRV	881		P: 端口区分符	# 十进制小数（仅）	---
				C: 控制数据	# 十进制小数（仅）	---
				D: 第一个目的地字	WORD	有 （1 或 2）
COUNTER FREQUENCY CONVERT (CJ1M CPU Unit Ver. 2.0 or later only)	PRV2	883		C1: 控制数据	WORD	---
				C2: 脉冲 / 旋转	UINT	---
				D: 第一个目的地字	UDINT	---
COMPARISON TABLE LOAD	CTBL @CTBL	882		P: 端口区分符	# 十进制小数（仅）	---
				C: 控制数据	# 十进制小数（仅）	---
				TB: 第一个比较表字	LWORD	有 （不固定）
SPEED OUTPUT	SPED @SPED	885		P: 端口区分符	UINT	---
				M: 输出模式	WORD	---
				F: 第一个脉冲频率字	UDINT	---
SET PULSES	PULS @PULS	886		P: 端口区分符	UINT	---
				T: 脉冲类型	UINT	---
				N: 脉冲数	DINT	---
PULSE OUTPUT	PLS2 @PLS2	887		P: 端口区分符	# + decimal only	---
				M: 输出模式	# + decimal only	---
				S: 设置表第一个字	WORD	Yes (6)
				F: 起始频率第一个字	UDINT	---

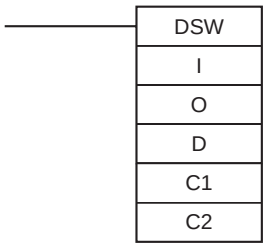
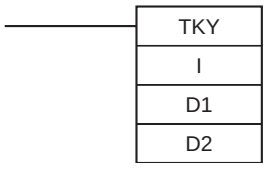
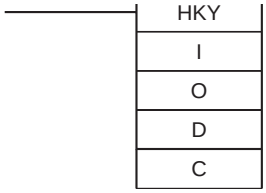
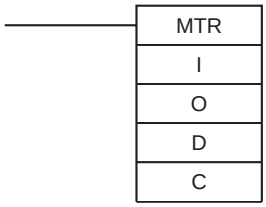
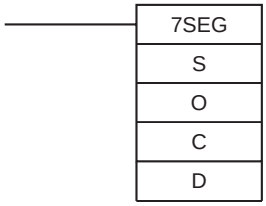
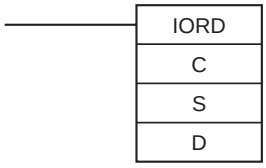
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
ACCELERATION CONTROL	ACC @ACC	888		P: 端口区分符	# + decimal only	---
				M: 输出模式	# + decimal only	---
				S: 设置表第一个字	WORD	Yes (3)
ORIGIN SEARCH	ORG @ORG	889		P: 端口区分符	# + decimal only	---
				C: 频率	# + decimal only	---
PULSE WITH VARIABLE DUTY FACTOR	PWM @PWM	891		P: 负载系数	# + decimal only	---
				F: 频率	# + decimal only	---
				D: 负载系数	# + decimal only	---

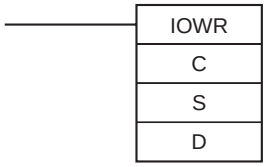
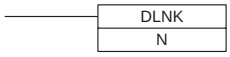
2-6-20 步进指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
STEP DEFINE	STEP	008	功能块中不支持	B: 位	---	---
STEP START	SNXT	009	功能块中不支持	B: 位	---	---

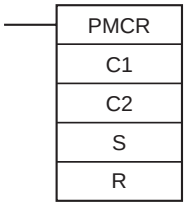
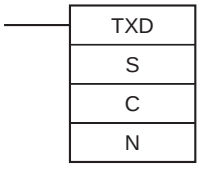
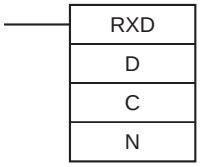
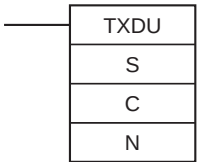
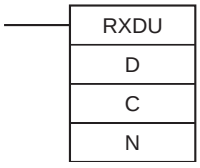
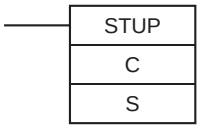
2-6-21 基本 I/O 单元指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
I/O REFRESH	IORF @IORF	097	功能块中不支持	St: 起始字	---	---
				E: 结束字	---	---
7-SEGMENT DECODER	SDEC @SDEC	078		S: 源字	UINT	---
				Di: 数字标志符	UINT	---
				D: 第一个目的地字	UINT	有 （不固定）

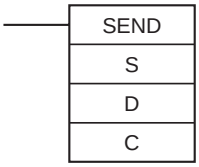
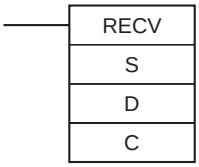
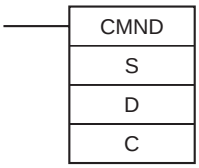
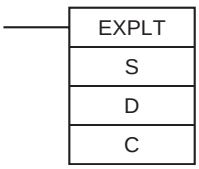
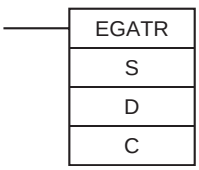
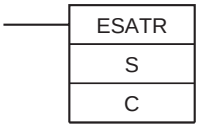
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
DIGITAL SWITCH INPUT	DSW	210		I: 数据输入字（D0~D3）	UINT	---
				O: 输出字	UINT	---
				D: 第一个结果字	WORD	---
				C1: 数字数量	UINT	---
				C2: 系统字	WORD	---
TEN KEY INPUT	TKY	211		I: 数据输入字	UINT	---
				D1: 第一个登录字	UDINT	---
				D2: 键输入字	UINT	---
HEXADECIMAL KEY INPUT	HKY	212		I: 数据输入字	UINT	---
				O: 输出字	UINT	---
				D: 第一个登录字	WORD	有 (3)
				C: 系统字	WORD	---
MATRIX INPUT	MTR	213		I: 数据输入字	UINT	---
				O: 输出字	UINT	---
				D: 第一个登录字	ULINT	有 (4)
				C: 系统字	WORD	---
7-SEGMENT DISPLAY OUTPUT	7SEG	214		S: 第一个源字	WORD	有 (2)
				O: 输出字	UINT	---
				C: 控制数据	# + decimal only	---
				D: 系统字	WORD	---
INTELLIGENT I/O READ	IORD @IORD	222		C: 控制数据	UINT	---
				S: 传送源字和字数	UDINT	有 （2）当需要数组变量时，必须使用 UNIT。
				D: 传送目的地字和字数	UINT	有 （不固定）

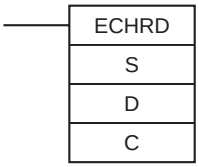
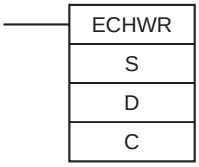
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
INTELLIGENT I/O WRITE	IOWR @IOWR	223		C: 控制数据	UINT	---
				S: 传送源字和字数	WORD	有 （不固定）
				D: 传送目的地字和字数	UINT	有 （2） 当需要数组变量时，必须使用 UNIT。
CPU BUS UNIT I/O REFRESH	DLNK @DLNK	226		N: 单元数	UINT	---

2-6-22 串行通信指令

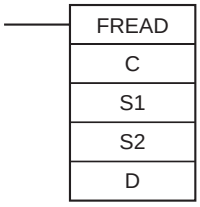
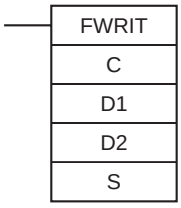
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
PROTOCOL MACRO	PMCR @PMCR	260		C1: 控制字 1	UINT	---
				C2: 控制字 2	UINT	---
				S: 第一个发送字	UINT	有 （不固定）
				R: 第一个接收字	UINT	有 （不固定）
TRANSMIT	TXD @TXD	236		S: 第一个源字	UINT	有 （不固定）
				C: 控制字	UINT	---
				N: 字节数 0000~0100 十六进制 （0 到 256 十进制）	UINT	---
RECEIVE	RXD @RXD	235		D: 第一个目的地字	UINT	有 （不固定）
				C: 控制字	UINT	---
				N: 字节数 0000~0100 十六进制 （0 到 256 十进制）	UINT	---
TRANSMIT VIA SERIAL COMMUNICATIONS UNIT	TXDU @TXDU	256		S: 第一个源字	UINT	有 （不固定）
				C: 第一个控制字	UDINT	---
				N: 发送字节数 （4 位数 BCD）	UINT	---
RECEIVE VIA SERIAL COMMUNICATIONS UNIT	RXDU @RXDU	255		D: 第一个目的地字	UINT	有 （不固定）
				C: 第一个控制字	UDINT	---
				N: 发送字节数 （4 位数 BCD）	UINT	---
CHANGE SERIAL PORT SETUP	STUP @STUP	237		C: 控制字 （端口）	UINT	---
				S: 第一个源字	UINT	有 （不固定）

2-6-23 网络指令

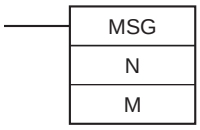
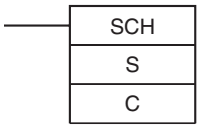
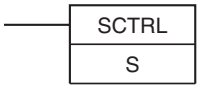
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
NETWORK SEND	SEND @SEND	090		S: 第一个源字	UINT	有 （不固定）
				D: 第一个目的地字	UINT	通过 AT 设置指定远程节点处的地址
				C: 第一个控制字	WORD	有 (5)
NETWORK RECEIVE	RECV @RECV	098		S: 第一个源字	UINT	通过 AT 设置指定远程节点处的地址
				D: 第一个目的地字	UINT	有 （不固定）
				C: 第一个控制字	WORD	有 (5)
DELIVER COMMAND	CMND @CMND	490		S: 第一个源字	UINT	有 （不固定）
				D: 第一个目的地字	UINT	有 （不固定）
				C: 第一个控制字	WORD	有 (6)
EXPLICIT MESSAGE SEND	EXPLT	720		S: 第一个源字	WORD	有 （不固定）
				D: 第一个目的地字	WORD	有 （不固定）
				C: 第一个控制字	LWORD	有 (4) 当需要数组变量时，必要使用 WORD。
EXPLICIT GET ATTRIBUTE	EGATR	721		S: 发送信息的第一个字	ULINT	有 (4) 当需要数组变量时，必要使用 WORD。
				D: 接收信息的第一个字	WORD	有 （不固定）
				C: 第一个控制字信息	LWORD	有 (4)
EXPLICIT SET ATTRIBUTE	ESATR	722		S: 发送信息的第一个字	WORD	有 （不固定）
				C: 第一个控制字	WORD	有 (3)

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
EXPLICIT WORD READ	ECHRD	723		S: 远程 CPU 单元中的第一个源字	UINT	当采用 AT 设置时，规定目的地地址。
				D: 当地 CPU 单元中的第一个目的地字	UINT	有 (2)
				C: 第一个控制字	WORD	有 (5)
EXPLICIT WORD WRITE	ECHWR	724		S: 远程 CPU 单元中的第一个源字	UINT	有 （不固定）
				D: 当地 CPU 单元中的第一个目的地字	UINT	当采用 AT 设置时，规定目的地地址。
				C: 第一个控制字	WORD	有 (5)

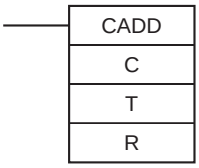
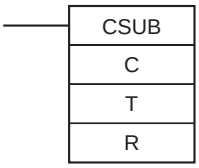
2-6-24 文件存储指令

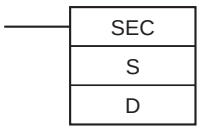
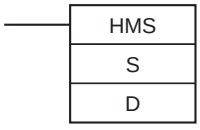
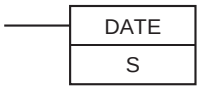
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
READ DATA FILE	FREAD @FREAD	700		C: 控制字	UINT	---
				S1: 第一个源字	LWORD	有 （2） 当需要数组变量时，必要使用 DWORD。
				S2: 文件名	UINT	有 (39)
				D: 第一个目的地字	UINT	有 （不固定）
WRITE DATA FILE	FWRIT @FWRIT	701		C: 控制字	UINT	---
				D1: 第一个目的地字	LWORD	有 （2） 当需要数组变量时，必要使用 DWORD。
				D2: 文件名	UINT	有 （39）
				S: 第一个源字	UINT	有 （不固定）

2-6-25 显示指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
DISPLAY MESSAGE	MSG @MSG	046		N: 信息数	UINT	---
				M: 第一个信息字	UINT	Yes (16)
SEVEN-SEGMENT LED WORD DATA DISPLAY	SCH @SCH	047		S: 信息数据	WORD	---
				C: 最左 / 最右 2 位数区分符	UINT	---
SEVEN-SEGMENT LED CONTROL	SCTRL @SCTRL	048		S: 控制数据	UINT	---

2-6-26 时钟指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
CALENDAR ADD	CADD @CADD	730		C: 第一个日历	WORD	有 (3)
				T: 第一个时间字	DWORD	有 （2）当需要数组变量时，必要使用 WORD。
				R: 第一个结果字	WORD	有 (3)
CALENDAR SUBTRACT	CSUB @CSUB	731		C: 第一个日历	WORD	有 (3)
				T: 第一个时间字	DWORD	有 （2）当需要数组变量时，必要使用 WORD。
				R: 第一个结果字	WORD	有 (3)

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
HOURS TO SECONDS	SEC @SEC	065		S: 第一个源字	DWORD	有 （2） 当需要数组变量时，必要使用 WORD。
				D: 第一个目的地字	DWORD	有 （2） 当需要数组变量时，必要使用 WORD。
SECONDS TO HOURS	HMS @HMS	066		S: 第一个源字	DWORD	有 （2） 当需要数组变量时，必要使用 WORD。
				D: 第一个目的地字	DWORD	有 （2） 当需要数组变量时，必要使用 WORD。
CLOCK ADJUSTMENT	DATE @DATE	735		S: 第一个源字	LWORD	有 （4） 当需要数组变量时，必要使用 WORD。

2-6-27 调试指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
TRACE MEMORY SAMPLING	TRSM	045	TRSM	---	---	---

2-6-28 故障诊断指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
FAILURE ALARM	FAL @FAL	006	FAL N M	N: FAL 数	#+ 仅十进制	---
				M：生成第一个信息字或错误代码（#0000~#FFFF）	WORD	---
SEVERE FAILURE ALARM	FALS	007	FALS N M	N：FALS 数	#+ 仅十进制	---
				M：生成第一个信息字或错误代码（#0000~#FFFF）	WORD	---
FAILURE POINT DETECTION	FPD	269	功能块中不支持	C：控制字	---	---
				T：监视时间	---	---
				R：第一个登录字	---	---

2-6-29 其他指令

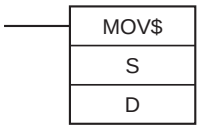
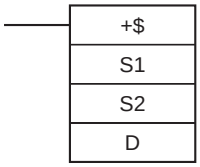
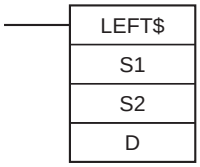
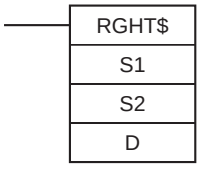
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
SET CARRY	STC @STC	040	STC	---	---	---
CLEAR CARRY	CLC @CLC	041	CLC	---	---	---
SELECT EM BANK	EMBC @EMBC	281	EMBC N	N:EM 组数	UINT	---
EXTEND MAXIMUM CYCLE TIME	WDT @WDT	094	WDT T	T: 定时器设置	仅指常数	---
SAVE Condition FlagS	CCS @CCS	282	CCS	---	---	---
LOAD Condition FlagS	CCL @CCL	283	CCL	---	---	---
CONVERT ADDRESS FROM CV	FRMCV @FRMCV	284	功能块中不支持	S: 含 CV- 系列存储器地址的字	---	---
				D: 目的地变址寄存器	---	---
CONVERT ADDRESS TO CV	TOCV @TOCV	285	功能块中不支持	S: 含 CS- 系列串行存储器地址的变址寄存器	---	---
				D: 目的地字	---	---
DISABLE PERIPHERAL SERVICING	IOSP @IOSP	287	IOSP	---	---	---
ENABLE PERIPHERAL SERVICING	IORS	288	IORS	---	---	---

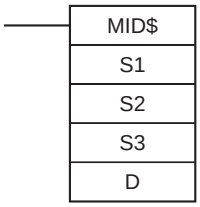
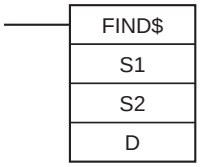
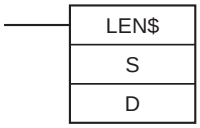
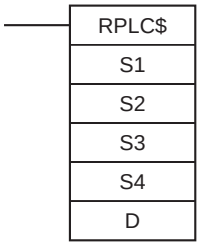
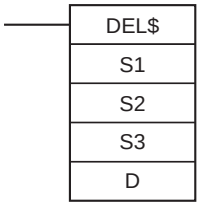
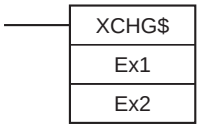
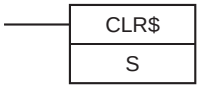
2-6-30 块编程指令

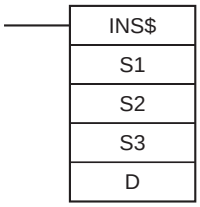
指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
BLOCK PROGRAM BEGIN	BPRG	096	功能块中不支持	N: 块程序号	---	---
BLOCK PROGRAM END	BEND	801	功能块中不支持	---	---	---
BLOCK PROGRAM PAUSE	BPPS	811	功能块中不支持	N: 块程序号	---	---
BLOCK PROGRAM RESTART	BPRS	812	功能块中不支持	N: 块程序号	---	---
CONDITIONAL BLOCK EXIT	CONDITION EXIT	806	功能块中不支持	---	---	---
CONDITIONAL BLOCK EXIT	EXIT Bit operand	806	功能块中不支持	B: 位操作数	---	---
CONDITIONAL BLOCK EXIT (NOT)	EXIT NOT Bit operand	806	功能块中不支持	B: 位操作数	---	---
CONDITIONAL BLOCK BRANCHING	CONDITION IF	802	功能块中不支持	---	---	---
CONDITIONAL BLOCK BRANCHING	IF Bit operand	802	功能块中不支持	B: 位操作数	---	---
CONDITIONAL BLOCK BRANCHING (NOT)	IF NOT Bit operand	802	功能块中不支持	B: 位操作数	---	---
CONDITIONAL BLOCK BRANCHING (ELSE)	ELSE	803	功能块中不支持	---	---	---
CONDITIONAL BLOCK BRANCHING END	IEND	804	功能块中不支持	---	---	---
ONE CYCLE AND WAIT	CONDITION WAIT	805	功能块中不支持	---	---	---
ONE CYCLE AND WAIT	WAIT Bit operand	805	功能块中不支持	B: 位操作数	---	---
ONE CYCLE AND WAIT (NOT)	WAIT NOT Bit operand	805	功能块中不支持	B: 位操作数	---	---
TIMER WAIT	TIMW (BCD)	813	功能块中不支持	N: 定时器数	---	---
				SV: 设定值	---	---
	TIMWX (BIN)	816	功能块中不支持	N: 定时器数	---	---
				SV: 设定值	---	---
COUNTER WAIT	CNTW (BCD)	814	功能块中不支持	N: 定时器数	---	---
				SV: 设定值	---	---
				I: 计数输入	---	---
	CNTWX (BIN)	817	功能块中不支持	N: 定时器数	---	---
				SV: 设定值	---	---
				I: 计数输入	---	---

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
HIGH-SPEED TIMER WAIT	TMHW (BCD)	815	功能块中不支持	N: 定时器数	---	---
				SV: 设定值	---	---
	TMHWX (BIN)	818	功能块中不支持	N: 定时器数	---	---
				SV: 设定值	---	---
LOOP	LOOP	809	功能块中不支持	---	---	---
LEND	LEND	810	功能块中不支持	---	---	---
LEND	LEND Bit operand	810	功能块中不支持	B: 位操作数	---	---
LEND NOT	LEND NOT Bit operand	810	功能块中不支持	B: 位操作数	---	---

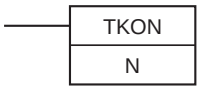
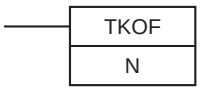
2-6-31 文本串处理指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
MOV STRING	MOV$ @MOV$	664		S: 第一个源字	UINT	有 （不固定）
				D: 第一个目的地字	UINT	有 （不固定）
CONCATENATE STRING	+$ @+$	656		S1: 文本串 1	INT	有 （不固定）
				S2: 文本串 2	INT	有 （不固定）
				D: 第一个目的地字	INT	有 （不固定）
GET STRING LEFT	LEFT$ @LEFT$	652		S1: 文本串第一个字	UINT	有 （不固定）
				S2: 字符数	UINT	---
				D: 第一个目的地字	UINT	有 （不固定）
GET STRING RIGHT	RGHT$ @RGHT$	653		S1: 文本串第一个字	UINT	有 （不固定）
				S2: 字符数	UINT	---
				D: 第一个目的地字	UINT	有 （不固定）

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
GET STRING MIDDLE	MID$ @MID$	654		S1: 文本串第一个字	UINT	有 （不固定）
				S2: 字符数	UINT	---
				S3: 起始位置	UINT	---
				D: 第一个目的地字	UINT	有 （不固定）
FIND IN STRING	FIND$ @FIND$	660		S1: 源文本串第一个字	UINT	有 （不固定）
				S2: 发现文本串第一个字	UINT	有 （不固定）
				D: 第一个目的地字	UINT	---
STRING LENGTH	LEN$ @LEN$	650		S: 文本串第一个字	UINT	有 （不固定）
				D: 第一个目的地字	UINT	---
REPLACE IN STRING	RPLC$ @RPLC$	661		S1: 源文本串第一个字	UINT	有 （不固定）
				S2: 替换文本串第一个字	UINT	有 （不固定）
				S3: 字符数	UINT	---
				S4: 起始位置	UINT	---
				D: 第一个目的地字	UINT	有 （不固定）
DELETE STRING	DEL$ @DEL$	658		S1: 第一个目的地字	UINT	有 （不固定）
				S2: 字符数	UINT	---
				S3: 起始位置	UINT	---
				D: 第一个目的地字	UINT	有 （不固定）
EXCHANGE STRING	XCHG$ @XCHG$	665		Ex1: 第一个交换字 1	UINT	有 （不固定）
				Ex2: 第一个交换字 2	UINT	有 （不固定）
CLEAR STRING	CLR$ @CLR$	666		S: 文本串第一个字	UINT	有 （不固定）

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
INSERT INTO STRING	INS$ @INS$	657		S1: 文本串第一个字	UINT	有 （不固定）
				S2: 字符数	UINT	有 （不固定）
				S3 起始位置	UINT	---
				D: 第一个目的地字	UINT	有 （不固定）
String Comparison	LD,AND, OR + =$,<>$,<$,<=$,>$,>=$	670 (=$) 671 (<>$) 672 (<$) 673 (<=$) 674 (>$) 675 (>=$)		S1: 文本串 1	UINT	有 （不固定）
				S2: 文本串 2	UINT	有 （不固定）

2-6-32 任务控制指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
TASK ON	TKON @TKON	820		N: 任务数	#+ 十进制数（仅）	---
TASK OFF	TKOF @TKOF	821		N: 任务数	#+ 十进制数（仅）	---

2-6-33 型号转换指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
BLOCK TRANSFER	XFERC @XFERC	565	XFERC W S D	W: 字数 （在 BCD 中）	WORD	---
				S: 第一个源字	WORD	有 （不固定）
				D: 第一个目的地字	WORD	有 （不固定）
SINGLE WORD DISTRIBUTE	DISTC @DISTC	566	DISTC S1 D S2	S1: 源字	WORD	---
				D: 目的地基地址	WORD	有 （不固定）
				S2: 偏移 （在 BCD 中）	WORD	---
DATA COLLECT	COLLC @COLLC	567	COLLC S1 S2 D	S1: 源基地址	WORD	有 （不固定）
				S2: 偏移 （在 BCD 中）	WORD	---
				D: 目的地基地址	WORD	---
MOVE BIT	MOVBC @MOVBC	568	MOVBC S C D	S: 源字或数据	WORD	---
				C: 控制字 （在 BCD 中）	WORD	---
				D: 目的地字	WORD	---
BIT COUNTER	BCNTC @BCNTC	621	BCNTC W S D	W: 字数 （在 BCD 中）	WORD	---
				S: 第一个源字	UINT	有 （不固定）
				D: 结果字	WORD	---

2-6-34 功能块特殊指令

指令	助记符	功能码	符号	操作数	支持的变量数据类型	所需的 AT 设置或数组变量（所需的字数据量示于括号中）
GET VARIABLE ID	GETID @GETIC	286	GETID S D1 D2	S: 源变量或地址	WORD	---
				D1: 变量类型 （I/O 存储器区） e	WORD	---
				D2: 字地址	WORD	---

2-7 CPU 单元功能块规格

下表所示的是用于 CS/CJ- 系列 CS1-H、 CJ1-H 和 CJ1M CPU 单元 （3.0 版或以后版本）的功能块规格。有关其他规格请参阅 CS/CJ 系列的其他操作手册。

2-7-1 规格

CS1-H CPU 单元

项目		规格								
型号		CS1H-CPU67H	CS1H-CPU66H	CS1H-CPU65H	CS1H-CPU64H	CS1H-CPU63H	CS1G-CPU45H	CS1G-CPU44H	CS1G-CPU43H	CS1G-CPU42H
I/O 点		5,120						1,280	960	
程序容量 （步骤）		250K	120K	60K	30K	20K	60K	30K	20K	10K
数据存储器		32K 字								
扩展数据存储器		32K 字 x13 组 E0_00000~EC_32767	32K 字 x7 组 E0_00000~E6_32767	32K 字 x3 组 E0_00000~E2_32767	32K 字 x1 组 E0_00000~E0_32767		32K 字 x3 组 E0_00000~E2_32767	32K 字 x1 组 E0_00000~EC_32767		
功能块	定义最大数	1,024	1,024	1,024	1,024	128	1,024	1,024	128	128
	最大实例数	2,048	2,048	2,048	2,048	256	2,048	2,048	256	256
闪存	功能块程序存储器（千字节）	1,664	1,664	1,024	512	512	1,024	512	512	512
	注释文件（千字节）	128	128	64	64	64	64	64	64	64

项目		规格								
闪存	程序索引文件 （千字节）	128	128	64	64	64	64	64	64	64
	变量表 （千字节）	128	128	128	64	64	128	64	64	64

CJ1-H CPU 单元

项目		规格						
型号		CJ1H-CPU67H	CJ1H-CPU66H	CJ1H-CPU65H	CJ1H-CPU64H	CJ1G-CPU44H	CJ1G-CPU43H	CJ1G-CPU42H
I/O 点		2,560				1,280	960	
程序容量 （步骤）		250K	120K	60K	30K	30K	20K	10K
数据存储器		32K 字						
扩展数据存储器		32K 字 x13 组 E0_00000~EC_32767	32K 字 x7 组 E0_00000~E6_32767	32K 字 x3 组 E0_00000~E2_32767	32K 字 x1 组 E0_00000~E2_32767	32K 字 x1 组 E0_00000~E0_32767		
功能块	定义最大数	1,024	1,024	1,024	1,024	1,024	128	128
	最大实例数	2,048	2,048	2,048	2,048	2,048	256	256
闪存	功能块程序存储器 （千字节）	1,664	1,664	1,024	1,024	512	512	512
	注释文件 （千字节）	128	128	64	64	64	64	64
闪存	程序索引文件 （千字节）	128	128	64	64	64	64	64
	变量表 （千字节）	128	128	128	128	64	64	64

CJ1-H CPU 单元

项目	规格					
	有内部 I/O 功能的单元			无内部 I/O 功能的单元		
型号	CJ1M-CPU23	CJ1M-CPU22	CJ1M-CPU21	CJ1M-CPU13	CJ1M-CPU12	CJ1M-CPU11
I/O 点	640	320	160	640	320	160
程序容量 （程序步）	20K	10K	5K	20K	10K	5K
扩展极数	最多一个	不支持扩展		最多一个	不支持扩展	
数据存储器	32K 字					
扩展数据存储器	无					

项目		规格			
		有内部 I/O 功能的单元		无内部 I/O 功能的单元	
脉冲开始时间		46ms （无加速 / 减速） 70ms （有加速 / 减速）	63ms （无加速 / 减速） 100ms （有加速 / 减速）	---	
计划中断数		2	1	2	1
PWM 输出		2	1	无	
子程序数最大值		1,024	256	1,024	256
JMP 指令中跳转数最大值		1,024	256	1,024	256
内部输入		10 点 · 4 个中断输入 （脉冲捕捉） · 2 个高速计数器输入 （50-kHz 相差或 100-kHz 单相）		---	
内部输出		6 点 · 2 个脉冲输出 （100-kHz） · 2 个 PWM 输出	6 点 · 2 个脉冲输出 （100-kHz） · 1 个 PWM 输出	---	
功能块	定义最大数	128			
	最大实例数	256			
闪存	功能块程序存储器 （千字节）	256			
	注释文件 （千字节）	64			
	程序索引文件 （千字节）	64			
	变量表 （千字节）	64			

CP1H CPU 单元

项目	X 型	XA 型	Y 型
型号	CP1H-X40DR-A CP1H-X40DT-D CP1H-X40DT1-D	CP1H-XA40DR-A CP1H-XA40DT-D CP1H-XA40DT1-D	CP1H-Y20DT-D
最大 I/O 点数	320 点 （40 个内置点 +40 点 / 扩展槽 x7 槽）		300 点 （20 个内置点 +40 点 / 扩展槽 x7 槽）
程序容量 （程序步）	20K		
数据存储器	32k 字		

项目		X 型	XA 型	Y 型
功能块	定义最大数	128		
	最大实例数	256		
闪存	功能块程序存储器 （千字节）	256		
	注释文件 （千字节）	64		
	程序索引文件 （千字节）	64		
	变量表 （千字节）	64		

NSJ 系列 NSJ 控制器

型号		NSJ5-TQ0 □ -G5D, NSJ5-SQ0 □ -G5D, NSJ8-TV0 □ -G5D, NSJ10-TV0 □ -G5D, NSJ12-TS0 □ -G5D,
最大 I/O 点数		1,280
程序容量 （程序步）		60K
数据存储器		32k 字
功能块	定义最大数	1024
	最大实例数	2048
闪存	功能块程序存储器 （千字节）	1024
	注释文件 （千字节）	64
	程序索引文件 （千字节）	64
	变量表 （千字节）	128

FQM1 柔性运动控制器

项目		协调指令	运动控制器指令	
型号		FQM1-CM002	FQM1-MMA22	FQM1-MMP22
最大 I/O 点数		344 点 （24 个内置点 + 基本 I/O 单元中的 320 点）	20 个内置点	
程序容量 （程序步）		10K		
数据存储器		32k 字		
功能块	定义最大数	128		
	最大实例数	256		
闪存	功能块程序存储器 （千字节）	256		
	注释文件 （千字节）	64		
	程序索引文件 （千字节）	64		
	变量表 （千字节）	64		

2-7-2 定时器指令操作

在 CP1H CPU 单元和 CS1H、CJ1-H 和 CJ1M CPU 单元（3.0 版或以后版本）的 PLC 属性中，有一种选项称作 Apply the same spec as TO-2047 to T2048-4095。该设置会影响本节所描述的定时器操作。

选择选项 （默认）

如果选择该选项，则无论定时器编号是多少所有定时器的操作均相同。详见下表。

定时器编号 T0000 ~ T4095 的定时器操作

刷新	描述
执行指令时	每当执行指令时刷新 PV。如果 PV 为 0，则完成标记处于 ON。如果 PV 不是 0，则完成标记处于 OFF。
所有任务执行完毕时	一个循环刷新所有 PV 一次。
每 80ms	如果循环时间超过 80ms，则每 80ms 刷新所有 PV 一次。

不选择选项

如果未选择此选项，则定时器编号 T0000 ~ T2047 的定时器指令刷新与下面给出的定时器编号 T2048~T4095 的定时器指令不一样。该工作与不支持功能块的 CPU 单元相同。（详情请参阅 CS/CJ 系列指令参考中的说明。）

定时器编号 T0000 ~ T2047 的定时器操作

刷新	描述
执行指令时	每当执行指令时刷新 PV。如果 PV 为 0，则完成标记处于 ON。如果 PV 不是 0，则完成标记处于 OFF。
所有任务执行完毕时	一个循环刷新所有 PV 一次。
每 80ms	如果循环时间超过 80ms，则每 80ms 刷新所有 PV 一次。

定时器编号 T2048~T4095 的定时器操作

刷新	描述
执行指令时	每当执行指令时刷新 PV。如果 PV 为 0，则完成标记处于 ON。如果 PV 不是 0，则完成标记处于 OFF。
所有任务执行完毕时	不更新 PV。
每 80ms	即使循环时间超过 80ms，也不更新 PV。

当采用默认分配于功能块变量的定时器编号 （T3072 ~ T4095）时，选择 Apply the same spec as TO-2047 to T2048-4095 选项来确保操作一致。

2-8 功能块程序步骤数和实例执行时间

2-8-1 功能块程序步骤数

本节只适用于 CP- 系列 CPU 单元 1.0 版或以后的版本和 CS/CJ- 系列 CPU 单元 3.0 版或以后的版本以及 NSJ 控制器和 FQM1 柔性运动控制器。

当已创建功能块定义并将实例复制于 CPU 单元的用户程序中时，采用以下公式来计算程序步骤数。 .

步骤数 = 实例数× （调用部分大小 m + I/O 参数传输部分大小 n × 参数数） + 功能块定义 p 中的实例步骤数 （见注释）

注 当相同的功能块定义复制于多个地方 （例如，多个实例）时，在子序列实例中不会减少功能块定义 （p）中的实例步骤数。因此，在以上公式中，实例数不乘以功能块定义 （p）中的实例步骤数。

下表只适用于 CP- 系列 CPU 单元 1.0 版或以后的版本和 CS/CJ- 系列 CPU 单元 3.0 版或以后的版本以及 NSJ 控制器和 FQM1 柔性运动控制器。

内容			步骤数
m	调用部分		57 步
n	I/O 参数传输部分数据类型示于括号中	1 位 I/O 变量 （BOOL）	6 步
		1 字 I/O 变量	6 步
		2 字 I/O 变量	6 步
		4 字 I/O 变量	12 步
p	功能块定义中的势力步骤数	总的指令步骤数 （与标准用户程序相同） +27 步	

例子：

有 1 字数据类型 （INT）的输入变量：5

有 1 字数据类型 （INT）的输出变量：5

功能块定义段：100 步

1 个实例的步数 =57+ （5+5） x 6 步 +100 步 +27 步 =244 步

2-8-2 功能块实例执行时间

本节只适用于 CP- 系列 CPU 单元 1.0 版或以后的版本和 CS/CJ- 系列 CPU 单元 3.0 版或以后的版本以及 NSJ 控制器和 FQM1 柔性运动控制器。

当已创建功能块定义并将实例复制于 CPU 单元的用户程序中时，采用以下公式来计算实例执行对循环时间的影响。

实例执行对循环时间的影响

= 启动时间 （A）
+ I/O 参数传输处理时间 （B）
+ 功能块定义中实例执行时间 （C）

下表所示的是 A、 B 和 C 的时间长度。

操作			CPU 单元型号		
			CS1H-CPU6 □ H CJ1H-CPU6 □ H	CS1G-CPU4 □ H CJ1G-CPU4 □ H NSJ	CJ1M-CPU □□ CP1H-X □□□ - □ CP1H-A □□□ - □
A	启动时间	启动时间未包含在 I/O 参数传输中	6.8 μs	8.8 μs	15.0 μs
B	I/O 参数传输处理时间数据类型示于括号中 1 位 I/O 变量 （BOOL）	1 位 I/O 变量 （BOOL）	0.4 μs	0.7 μs	1.0 μs
		1 字 I/O 变量	0.3 μs	0.6 μs	0.8 μs
		2 字 I/O 变量	0.5 μs	0.8 μs	1.1 μs
		4 字 I/O 变量	1.0 μs	1.6 μs	2.2 μs
C	功能块定义中的势力步骤数	总的指令步骤数 （与标准用户程序相同）			

例子：CS1H-CPU63H
有 1 字数据类型 （INT）的输入变量：3
有 1 字数据类型 （INT）的输出变量：2
功能块定义段指令总处理时间：10 μs

1 个实例执行时间 = 6.8 μs + (3 + 2) × 0.3 μs + 10 μ s = 18.3 μs

注 当相同功能块定义已复制于多个地方时，根据多个实例数来提高执行时间。

第 3 章
创建功能块

本章主要描述了 CX-Programmer 创建功能块程序

3-1 程序流

以下程序用来创建功能块、将功能块保存于文件中、将功能块传输至 CPU 单元中以及监视调试功能块。

创建功能块

创建项目

详情请参阅 3-2-1 创建项目。

■创建新项目

1,2,3...

1. 启动 CX-Programmer，在文件菜单中选择 New。
2. 选择 设备类型：。。。或 CP1H、 NSJ 或 FQM1-CM （MMA/MMP）。

■反复使用现有的 CX-Programmer 项目

1,2,3...

1. 启动 CX-Programmer，在文件菜单中选择文件来读取用 CX-Programmer 4.0 版或以前版本所创建的现有项目文件 （.cxp）。
2. 选择 设备类型：。。。或 CP1H、 NSJ 或 FQM1-CM （MMA/MMP）。

创建功能块定义

详情请参阅 3-2-2 创建新功能块定义。

1,2,3...

1. 在项目工作区内选择 Function Blocks，右键单击。
2. 在弹出式菜单中选择 Insert Function Block - Ladder 或 Insert Function Blocks -Structured Text。

定义功能块

详情请参阅 3-2-3 定义用户创建的功能块。

■在输入梯级程序或 ST 程序之前注册变量

1,2,3...

1. 将变量住院于变量表中。
2. 创建梯级程序或 ST 程序。

■输入梯级程序或 ST 程序同时根据需要注册变量

1,2,3...

1. 创建梯级程序或 ST 程序。
2. 只要需要，即可将变量住院于变量表中。

从功能块定义中创建实例

详情请参阅 3-2-4 从功能块定义中创建实例。

■将实例嵌入梯级程序窗口中然后输入实例名

1,2,3...

1. 将光标置于要创建功能块实例处 （例如，复制），然后按 F 键。
2. 输入实例名。
3. 选择要复制的功能块定义。

■将实例名注册于全局符号表中。然后嵌入时选择实例名

1,2,3...

1. 选择 Function Block 作为全局符号表中的变量数据类型。
2. 在梯级段窗口中，按 F 键。

3. 从 FB 实例 区的下拉菜单中选择已注册的实例名。

外部 I/O 分配于功能块中

详情请参阅 3-2-5 使用输入键设置功能块参数。

1,2,3...

1. 将光标置于输入或输出变量处，然后，按 P 键。
2. 输入输入变量的源地址或输出变量的目的地地址。

设置功能块存储器分配

详情请参阅 3-2-6 设置 FB 实例区。

1,2,3...

1. 选择实例。在 PLC 菜单中，选择 Function Block Memory - Function Block Memory Allocation。
2. 设置功能块存储器分配。

打印、保存及再使用功能块文件

编辑功能块定义并将功能块定义保存为库文件

详情请参阅 3-2-12 编辑功能块定义（检查程序）和 3-2-15 保存和再使用功能块定义文件。

1,2,3...

1. 编辑已保存的功能块。
2. 打印功能块。
3. 将功能块保存为功能块定义文件 （.cxf）。
4. 文件读入另一 PLC 项目中。

程序传输至 PLC 中

详情请参阅 3-2-16 下载 / 上载程序至实际 CPU 单元。

功能块监视调试

详情请参阅 3-2-17 功能块监视调试。

3-2 程序

3-2-1 创建项目

用 CX-Programmer 创建新项目

1,2,3... 1. 启动 CX-Programmer，从文件菜单中选择 New。

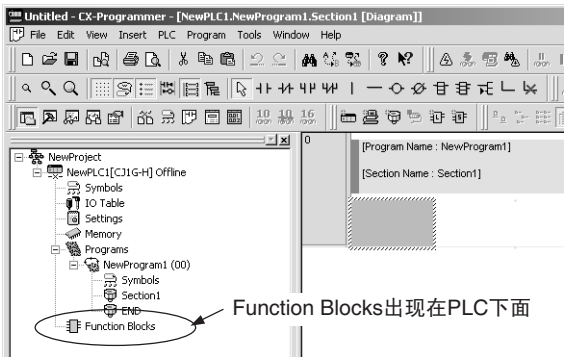
2. 在修改 PLC 窗口中，选择功能块数据类型。这些内容列于下表。

单元	CPU
CS1G-H	CPU42H/43H/44H/45H
CS1H-H	CPU63H/64H/65H/66H/67H
CJ1G-H	CPU42H/43H/44H/45H
CJ1H-H	CPU65H/66H/67H
CJ1M	CPU11/12/13/21/22/23
CP1H	XA/X
NSJ	G5D (用于 NSJ5-TQ0 □ -G5D, NSJ5-SQ0 □ -G5D, NSJ8-TV0 □ -G5D, NSJ10-TV0 □ -G5D, and NSJ12-TS0 □ -G5D)
FQM1-CM	FQM1-CM002
FQM1-MMA	FQM1-MMA22
FQM1-MMP	FQM1-MMP22

按设置按钮，选择 CPU 类型。有关其他设置的详细内容，请参阅 CX-Programmer 5.0 版操作手册 （W414）。

3-2-2 创建新功能块定义

1,2,3... 1. 创建项目时，Function Blocks 图标出现在项目工作区中。详见下图。



2. 将功能块嵌在功能块图标后，创建功能块定义。

创建功能块定义

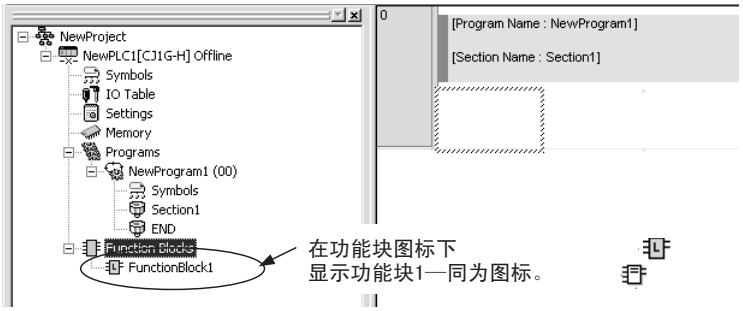
用户可采用梯级编程或结构化文本来定义功能块。

用梯级编程创建 （嵌入）功能块定义

1. 在项目工作区内选择 Function Blocks，右键单击从弹出式菜单中选择 Insert Function Blocks - Ladder。（或从插入菜单中选择 Function Block - Ladder。）

用结构化文本创建 （嵌入）功能块定义

1. 在项目工作区内选择 Function Blocks，右键单击从弹出式菜单中选择 Insert Function Blocks - Structured Text。（或从插入菜单中选择 Function Block - Structured Text。）

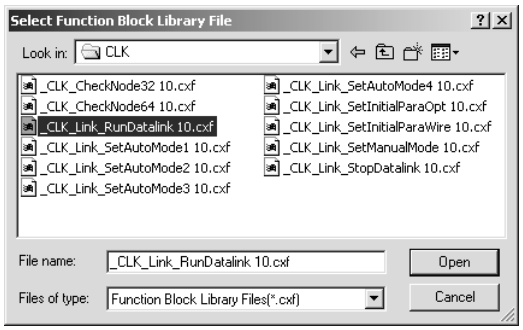


2. 在采用梯级编程语言 （默认）或 ST 语言后，调用功能块 1 的功能块自动嵌入。该图表包含新创建 （嵌入）的功能块定义。
3. 每当创建功能块定义，自动分配功能块 @ 名。其中，@ 为序列号。可以修改这些功能块名。所有功能块名称字节不得超过 64 个。

使用 OMRON FB 库文件

采用以下程序插入 OMRON FB 库文件 （.cxf）。

1. 在项目工作区内选择 Function Blocks，右键单击从弹出式菜单中选择 Insert Function Blocks - Library File。（或从插入菜单中选择Function Block - Library File。）
2. 显示以下选择功能块库文件对话框。



注 如要在功能块库文件对话框中指定默认文件夹 （文件位置），选择 Tools － Options，点击 General 条。然后，在 OMRON FB library storage

location 区域内选择默认文件。

3. 指定 OMRON FB 库文件所位于的文件夹，选择库文件，点击 Open 按钮。

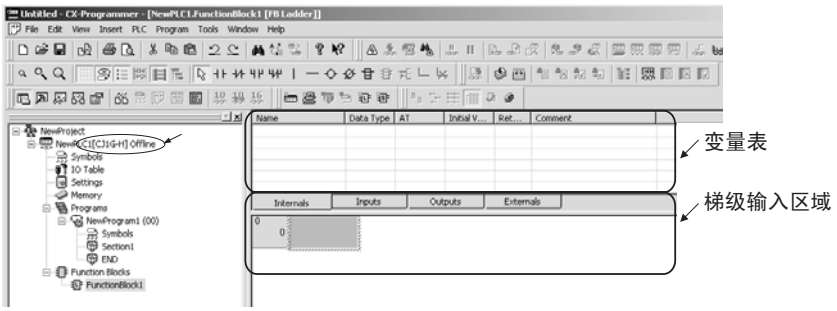
在 之后，插入库文件作为功能块定义。 .

功能块定义

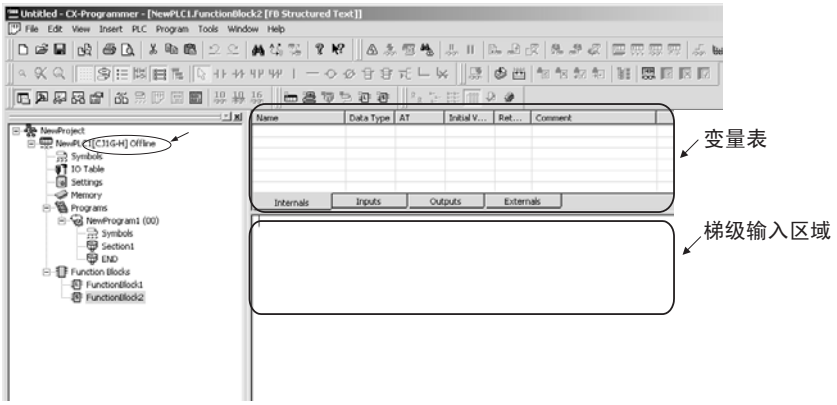
创建功能块定义

当双击新创建功能块 1 图标 （右键单击并从弹出式菜单中选择 Open 时，以下窗口中的一个窗口会显示在显示屏上。用于功能块变量的变量表示于顶部。梯级程序或结构化文本输入区域示于底部。

梯级程序



结构化文本



如图所示，功能块定义由用作接口的变量表和用作算法的梯级程序或结构化文本组成。

用作接口的变量表

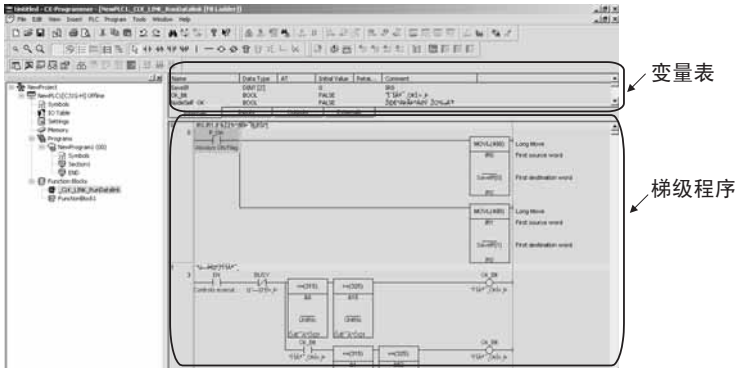
在此处，由于无变量分配于 PLC 中的 I/O 存储器地址，因此，变量表是空的。

用作算法的梯级程序或结构化文本

- 除了某些特殊情况之外，功能块梯级程序可以包括用于正常程序中的所有指令。可用指令限制请参阅 2-3 功能块限制。
- 根据 IEC61131-3 所规定的 ST 语言输入结构化文本。

使用 OMRON FB 库文件

双击插入的功能块库 （右键单击并从弹出式菜单中选择 Open），显示已创建于右上窗口中的变量表和创建于右下窗口中的梯级程序。两个窗口均显示为灰色而不能进行编辑。



注 OMRON FB 库文件 （.cxf）默认设置中不显示功能块定义。在功能块属性中选择 Display the inside of FB 选项来显示定义。（在项目工作区中选择 OMRON FB 库文件，右键单击，选择 Properties。在 General Tab 中选择 Display the inside of FB 选项。）

3-2-3 定义用户创建的功能块

注册变量并创建算法来定义功能块。

有二种方法来定义功能块。

- 先注册变量，然后输入梯级程序或结构化文本。
- 输入梯级程序或结构化文本时，如需要注册变量。

先注册变量

变量注册于变量表中

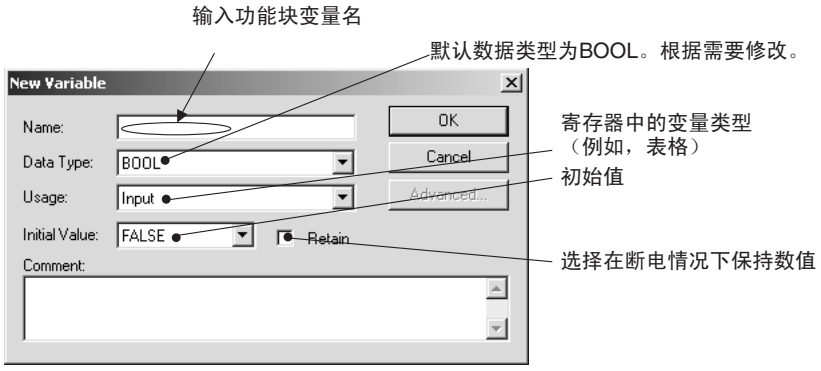
变量根据类型分成 4 张表格 （在变量表中）：内部、输入、输出和外部。注册或显示变量时必须打开这些变量表格。

1,2,3...

1. 为变量表中有效注册的变量类型做表格。（见注释。）将光标置于表格中，进行以下操作 （任一操作）：
   - 如要将变量添加至最后一行，则从弹出式菜单中选择 Insert Variable。
   - 如要在列表中将变量添加到一行上面或下面，则从弹出式菜单中选择 Insert Variable - Above or Below。

   注 当设置用法从而插入变量时，变量所注册的表格也可以切换。（N：内部； I：输入； O：输出； E：外部；）

   如下所示的是新变量对话框。
   - 名称：输入变量名；
   - 数据类型：选择数据类型；
   - 用法：选择变量类型；
   - 初始值：选择开始运算时的变量初始值。
   - 保持:断电时或当运行模式从PROGRAM 或MONITOR 模式切换到RUN 模式时，规定是否要保持变量值。如果不选 Retain ，则当出现上述情况时变量值被清除。



注 (a) 如是用户自定义外部变量，则将相同的变量名注册于全局符号表中来浏览全局符号表。

(b) 系统定义的外部变量事先注册于外部变量表中。

2. 例如，输入“ aaa ”作为变量名，点击 OK 按钮。如下所示，叫作 aaa 的 BOOL 变量创建于变量表的输入表格中。

注 (1) 添加了变量后，选择变量以反白显示变量。然后，拖曳移至另一行。要选择变量拖曳，则在任何一栏中 （除了 Name 区域之外）选择变量。

(2) 输入变量后，双击并切换至 Usage 区域中设置使变量移至注册表中。（N：内部； I：输入； O：输出； E：外部；）变量也可在内部、外部、输入和输出表格之间复制或移动。选择变量，右键单击并从弹出式菜单中选择 Copy 或 Cut 。然后，选择 Paste。

(3) 还必须输入由 AT 设置 （分配实地址）指定的变量名。

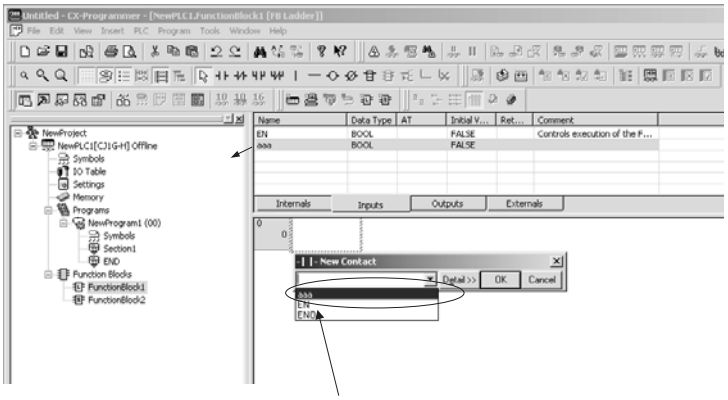
(4) 以下文本用来显示 PLC 中的 I/O 存储器地址。因而，这些文本不能作为功能块变量表中的变量名输入。

- A、W、H、HR、D、DM、E、EM、T、TIM、C 或 CNT 后面跟数值。

创建算法

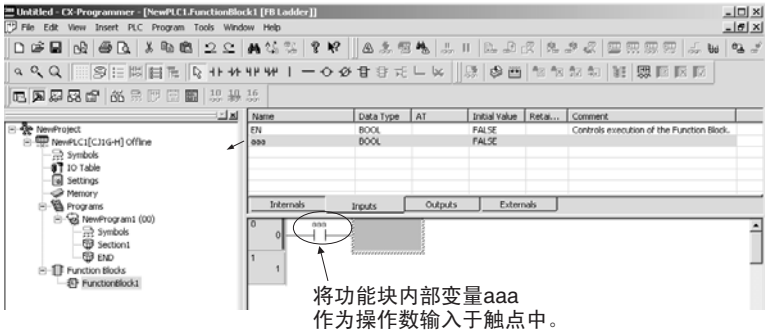
采用梯级程序

1,2,3... 1. 按 C 键，在新触点对话框的下拉菜单中选择前面已注册的 aaa。



按 C 键，在新触点对话框的下拉菜单中选择前面已注册的aaa。

2. 点击 OK 按钮。将功能块内部变量 aaa 作为操作数 （变量类型：内部）输入于触点中。



将功能块内部变量aaa作为操作数输入于触点中。

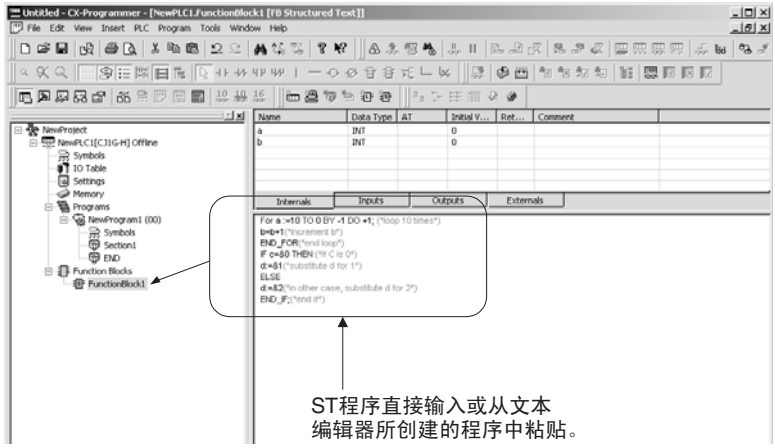
与标准程序相同的方法用 CX-Programmer 输入梯级程序剩余部分。

注 在功能块中，不能直接输入指令操作数地址。只有变址寄存器 （IR）和数据寄存器（DR）可以直接输入（不作为变量）：地址 DR0~DR5，直接规格 IR0~IR15

采用结构化文本

ST 语言程序 （见注解）可以直接输入于 ST 输入区域或先复制输入于通用文本编辑器中的程序，然后通过编辑菜单中的 Paste 命令将程序粘贴至 ST 输入中。

注 ST 语言符合 IEC61131-3 要求。详情请参阅附录 B 结构化文本（ST 语言）规格。



注 (1) 输入 Tab 或空格以生成缩进。这不影响算法。

(2) 按住 Ctrl 键不放同时转动带滚轮鼠标上的滚轮来更改显示大小。

(3) 当 ST 语言程序输入于或粘贴于 ST 输入区域中时，语法关键字保留字自动显示为蓝色、注释显示为绿色、错误显示为红色而其他内容显示为黑色。

(4) 要修改字体大小或颜色，则从工具菜单中选择 Options 然后点击外观标记页上的 ST Font 按钮。可以修改字体名称和大小（默认值为 8 点）以及颜色。

(5) 有关结构化文本规格的详细内容，请参阅附录 B 结构化文本 （ST 语言）规格。

根据需要注册变量

先输入梯级程序或结构化文本。然后，根据需要注册变量。

采用梯级程序

当采用梯级程序时，每当输入以前未曾注册的变量名时会显示对话框以便注册变量。此时，注册变量。

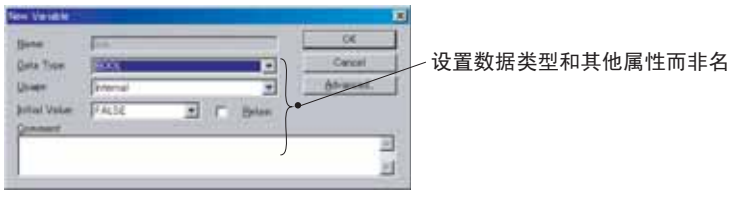
采用以下程序：

1,2,3...

1. 按 C 键将未曾输入的变量名 （例如， aaa）输入于新触点对话框中。

   注 在功能块中，不能直接输入指令操作数地址。只有变址寄存器（IR）和数据寄存器（DR）可以直接输入（不作为变量）：地址 DR0~DR5，直接规格 IR0~IR15 和间接规格 ,IR0 ~ ,IR15。

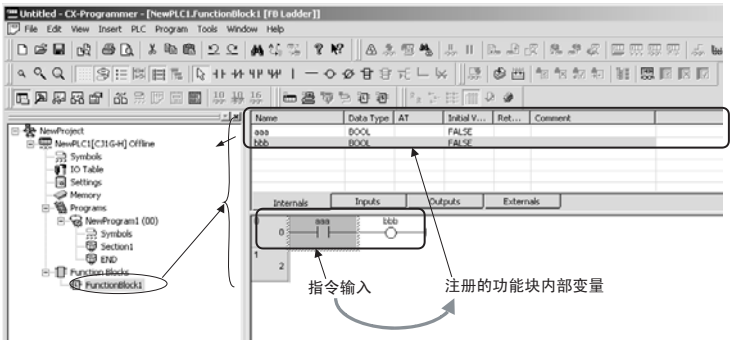
2. 点击 OK 按钮，显示新变量对话框。如有特殊指令，则显示指令中每个操作数的新变量对话框。



所有输入变量属性初始显示如下：

- 用法：内部
- 数据类型：BOOL—用于触点； WORD—用于通道 （字）
- 初始值：数据类型默认值。
- 保持：不选择。

3. 进行任何所需的修改。点击 OK 按钮。
4. 如下所示，已注册的变量示于程序上的变量表。



5. 如果已输入的变量类型或属性不正确，则双击变量表中的变量。然后，进行必需的修正。

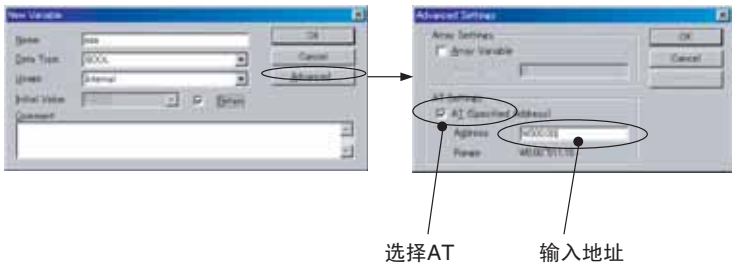
■参考信息

AT 设置 （指定地址）

在变量属性中进行 AT 设置以指定基本 I/O 单元、特殊 I/O 单元或 CPU 总线单元的分配地址或未用 CX-Programmer 注册的辅助区域地址。需要采用变量表来实现此目的。采用以下程序来规定地址。

1,2,3...

1. 在将变量名输入于新变量对话框中。点击 Advanced 按钮。显示高级设置对话框。
2. 在 AT 设置下选择 AT (指定地址)，输入所需地址。



即使变量中有地址指定用于 AT 设置（与无指定地址的变量相同），变量名仍可用来将变量输入于功能块定义算法中。

例如，如果名为Restart的变量有一个地址A50100指定用于AT 设置，Restart 可指定为指令操作数。

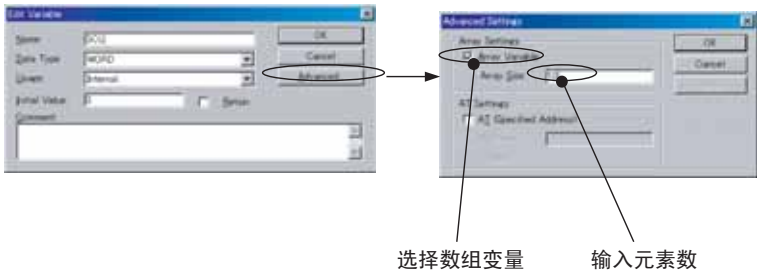
数组设置

指定数组以便一个以上的变量可以使用相同的数据属性并且将变量作为一个组来管理。

采用以下程序来设置数组。

1,2,3... 1. 变量名输入于新变量对话框后，点击 Advanced 按钮。显示高级设置对话框。

2. 在数组设置中选择数组变量。最大元素数输入于数组中。



当数组变量名输入于功能块定义算法中时，与变址相对应的方括号出现在数组名后。

例如，如果创建的变量名为 PV 且其元素最多只允许 3 个即 PV[0]、 PV[1] 和 PV[2] ，则 PV[0]、 PV[1] 和 PV[2] 可以指定为指令操作数。

有三种方法来指定指数：

- 直接带数字；例如，上例中的 PV[1]（用于梯级程序或 ST 语言编程）。
- 带变量；例如，上例中的 PV[a] 。其中，“ a ”是 INT 数据类型的变量（用于梯级程序或 ST 语言编程）。
- 方程式；例如，上例中的 PV[a+b] 或 PV[a+1] ；其中，“ a ”和“ b ”对 INT 数据类型的变量是一样的 （仅用于 ST 语言编程）。

使用结构化文本

当使用结构化文本时，每当输入未注册的变量名时不显示对话框注册变量。请保证在您需要时或完成运算后始终将标准文本编程中所用的变量注册于变量表中。（将光标置于注册变量的标记页中，右键单击并从弹出式菜单中选择 Insert Variable。

注 有关结构化文本规格的详细内容请参阅附录 B 结构化文本 （ST 语言）规格。

复制用户程序回路并粘贴于功能块定义的梯级编程中

用户程序中的单回路或多回路复制粘贴于功能块顶的梯级编程中。但该操作受到以下限制：

源指令操作数：仅是地址

地址不注册于功能块变量表中。粘贴后，地址示于操作数中。颜色为红色。双击指令并将变量名输入于操作数中。

注 但变址寄存器 （IR）和数据寄存器 （DR）不需要在粘贴后进行修改。操作数中的功能同样如此。

源指令操作数：地址和 I/O 注释

在工具菜单选项下的符号标记中选中自动生成符号名

用户程序符号名 （仅在全局符号表中）自动生成 AutoGen_ + Address （如果撤消选定的选项，则符号名被删除。）

例 1：地址 100.01—符号名显示为 AutoGen_100_01。

例 2：地址 D0-- 符号名显示为 AutoGen_D0。

如果用户程序回路复制粘贴于功能块定义程序中，则符号自动注册于功能块定义符号表中 （与回路复制同时进行）。符号名为 AutoGen_Address 而 I/O 注释则为 Comment。该功能使编程的回路作为地址和 I/O 注释便于在功能块中再使用。

注 前缀 AutoGen_ 不添加至变址寄存器 （IR）和全局数据寄存器 （DR）中。它们不能注册于原全局符号表中。

在工具菜单选项下的符号标记中未选自动生成符号名

地址和 I/O 注释未注册于功能块定义变量表中。地址示于操作数中。颜色为红色。 I/O 注释丢失。双击指令并将符号名输入于操作数中。

变址寄存器 （IR）和数据寄存器 （DR）不需要在粘贴后进行修改。操作数中的功能同样如此。

源指令操作数：符号

用户程序符号自动注册于功能块定义符号表的内部变量中。但该操作受到以下限制：

地址

符号地址未注册。使用 AT 设置来指定相同地址。

符号数据类型

如下表所示，当从用户程序粘贴于功能块定义中时，转换符号数据类型。

用户程序中符号数据类型	→	粘贴于功能块程序中后的变量数据类型
CHANNEL	→	WORD
NUMBER	→	变数值 （数字）直接粘贴于操作数中作为常数。
UINT BCD	→	WORD
UDINT BCD	→	DWORD
ULINT BCD	→	LWORD

但不能从符号表 （不是程序）中复制符号数据类型 CHANNEL、 NUMBER、 UINT BCD、 UDINT BCD 或 ULINT BCD 并将其粘贴于功能块定义变量表中。

注　不能从符号表中复制符号名自动生成的符号 （AutoGen_ + Address）并将其粘贴于功能块定义符号表中。

3-2-4　从功能块定义中创建实例

如果功能块定义注册于全局符号表中，则可采用以下方法中的一个来创建实例。

方法 1：选择功能块定义，插入程序中，输入新实例名。实例自动注册于全局符号表中。

方法 2：在全局符号表中将数据类型设为“ FUNCTION BLOCK, ”，指定功能块定义使用，输入实例名进行注册。

注　当使用 ST 语言时，选择“ FUNCTION BLOCK ”作为变量数据类型、使用所要的实例名以及输入功能块调用语句来调用功能块。

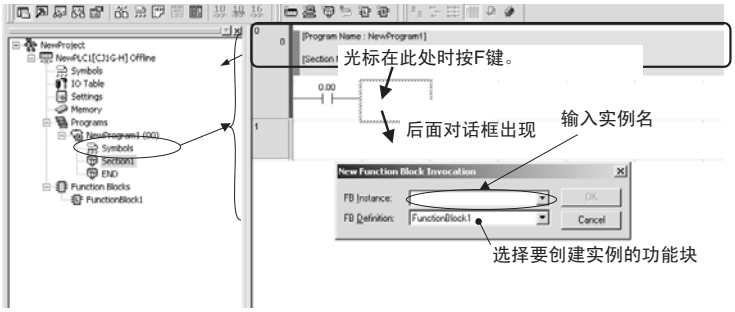
■方法 1：使用梯级段窗口中的 F 键并输入实例名

1,2,3...

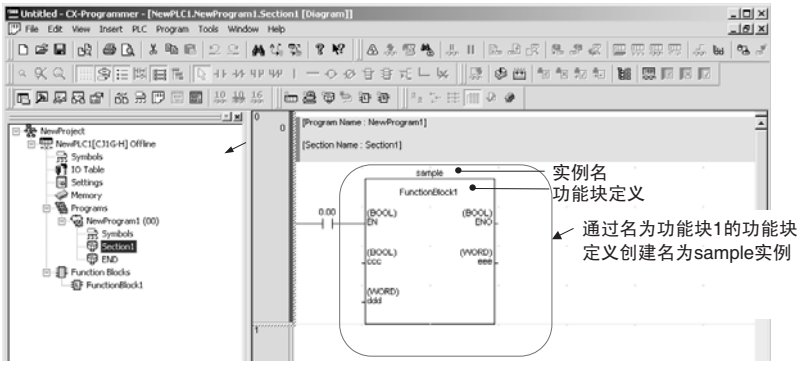
1. 在梯级段窗口中，将光标置于要插入实例的程序中，按 F 键。（或从插入菜单中选择 Function Block Invocation。）显示新功能块调用对话框。

   当使用 ST 语言时，选择“ FUNCTION BLOCK ”作为变量数据类型、使用所要的实例名以及输入功能块调用语句来调用功能块。在实例名后的括号中规定变元 （输入变量值从调用功能块中传输至被调用功能块的输入变量中）并规定返回值（调用功能块输出变量接收被调用功能块的输出变量）。实例名可设为任何有“ FUNCTION BLOCK ”数据类型的内部变量。

2. 输入实例名，选择创建实例的功能块，点击 OK 按钮。



3. 举例说明：FB 实例区域中的实例名设为 sample， FB 定义区域中的实例名设为 FunctionBlock1，点击 OK 按钮。如下所示，用实例名 sample 创建一份名为功能块 1 的功能块定义。



实例自动注册于全局符号表中。其中实例名为 sample 而数据类型为 FUNCTION BLOCK。

■实例名注册于全局符号表中然后选择实例名

如果实例名事先注册于全局符号表中，则从全局符号表中选择实例名来创建其他实例。

1,2,3...

1. 如是梯级程序，则在全局符号表中选择数据类型 Function block，输入实例名并注册实例。

   如是 ST，则选择数据类型 Function block，使用实例名，使用以下功能块调用语句来调用功能块：

   输入实例名 （带功能块数据类型的任何内部变量） -- 后面括号跟变元 （例如，指定调用功能块输入变量值来传递被调用功能块输入变量）。此外，还包括返回值 （例如，指定被调用功能块输出变量值来传回至调用功能块输出变量）。
2. 在梯级段窗口中，按 F 键。功能块调用对话框出现。
3. 选择以前已注册的实例名。从 FB 实例区域的下拉菜单注册实例名。创建实例。

限制

创建实例时遵守以下限制。详情请参阅 2-3 功能块限制。

- 每个程序回路中只可创建一个功能块。
- 不能在实例左面分级。
- 实例不能直接与左面总线相连。例如，必须插入 EN。

注 如果对功能块定义变量表中的 I/O 变量进行更改，则至所有实例 （已通过功能块定义创建）左面的总线显示红色以表示发生错误。发生这种情况时，选择功能块，右键单击，选择 Update Invocation。

如功能块定义已发生变化，则立即更新实例。然后，表示有错误的红色的总线清除。

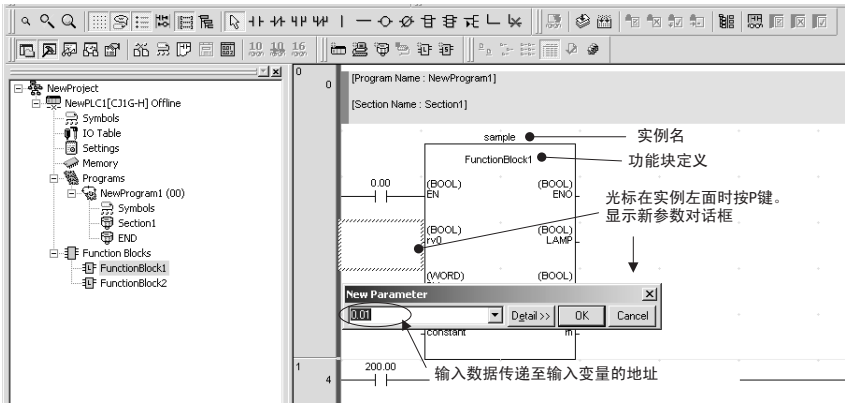
3-2-5 按 Enter 键设置功能块参数

创建功能块实例后，必须设置输入变量的输入参数和输出变量的输出参数以激活外部 I/O。

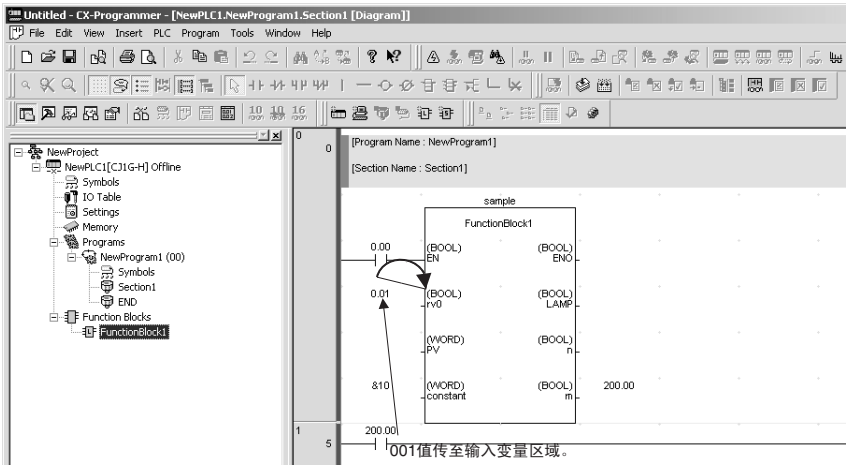
- 数值、地址和程序符号 （全局符号和局部符号）可以设置于输入参数中。（见注 a）。
- 地址和程序符号 （全局符号和局部符号）可以设置于输出参数中。（见注 b）。

注 (a) 功能块输入变量大小和程序符号数据大小必须匹配。
(b) 功能块输出变量大小和程序符号数据大小必须匹配。

1,2,3... 1. 输入位于实例左面而输出位于实例右面。将光标置于要设置的参数处，按 Enter 键。（或从插入菜单中选择 Function Block Parameter ）。新参数对话框如下所示。



2. 设置传至输入变量的地址数据源地址。同时，设置输出变量地址数据上传的目的地地址。



注 设置所有输入参数中的数据。即使只有一个输入参数处于空白，实例做总线仍显示红色表示有错误。如发生这种情况，则程序不能传输至 CPU 单元。

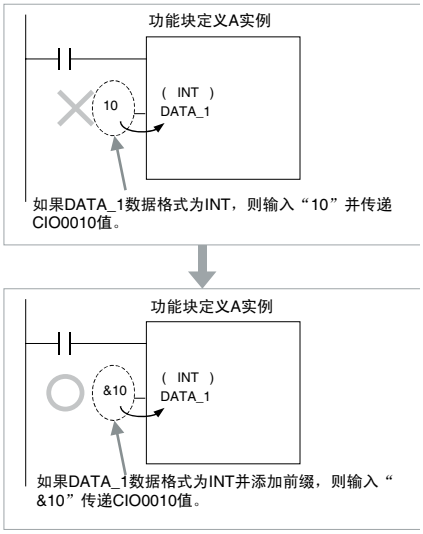
参数中输入变量

下表列出了输入参数值的方法。

输入变量数据类型	内容	大小	输入方法	设置范围
BOOL	位数据	1 位	P_Off, P_On	0 (FALSE), 1 (TRUE)
INT	整数	16 位	正值：& 或 + 后面跟整数； 负值：- 后面跟整数；	-32768 ～ +32767
DINT	双整数	32 位		-2147483648 ～ +2147483647
LINT	长 （4 字节）整数	64 位		-9223372036854775808 ～ +9223372036854775807
UINT	无符号整数	16 位	正值：& 或 + 后面跟整数；	&0 ～ 65535
UDINT	无符号双整数	32 位		&0 ～ 4294967295
ULINT	无符号长 （4 字节）整数	64 位		&0 ～ 18446744073709551615
REAL	实数	32 位	正值：& 或 + 后面跟实数（有十进制小数）； 负值：- 后面跟整数 （有十进制小数）；	-3.402823×10^{38} ～ $-1.175494 \times 10^{-38}$, 0, $+1.175494 \times 10^{-38}$ ～ $+3.402823 \times 10^{38}$
LREAL	长实数	64 位		$-1.79769313486232 \times 10^{308}$ ～ $-2.22507385850720 \times 10^{-308}$, 0, $+2.22507385850720 \times 10^{-308}$ ～ $+1.79769313486232 \times 10^{308}$
WORD	16 位数据	16 位	# 后面跟十六进制数 （最多 4 位数）； & 或 + 后面跟实数（有十进制小数）；	#0000 ～ FFFF 或 &0 ～ 65535
DWORD	32- 位数据	32 位	# 后面跟十六进制数 （最多 8 位数）； & 或 + 后面跟实数（有十进制小数）；	#00000000 ～ FFFFFFFF 或 &0 ～ 4294967295
LWORD	64- 位数据	64 位	# 后面跟十六进制数 （最多 16 位数）； & 或 + 后面跟实数（有十进制小数）；	#0000000000000000 ～ FFFFFFFFFFFFFFFF 或 & ～ 18446744073709551615

注 如果非布尔数据类型用于输入变量而且只输入一个数值 （例如， 20），则传递 CIO 区域地址值 （例如， CIO 0020）而不是数值。要设置数值，则在输入数值之前先插入 &、 #、 + 或 . 前缀。

程序举例说明：



如果输入变量数据类型为布尔而且只有一个数值（例如，0 或 1）输入于参数中，传递 CIO 000000 （0.00）或 CIO 000001 （0.01）值。输入 P_Off – 用于 0 （OFF）；输入 P_On – 用于 1 （ON）。

3-2-6 设置 FB 实例区域

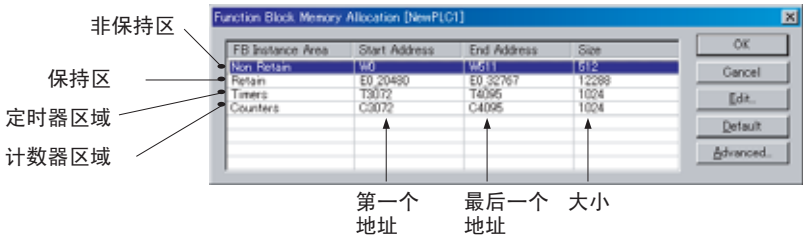
设置分配于功能块变量的地址区域。这些区域称作功能块实例区域。

1,2,3...

1. 在梯级段窗口中或全局符号表中选择实例，然后，从 PLC 菜单中选择 Function Block Memory - Function Block Memory Allocation。

   功能块存储器分配对话框如下所示。

2. 设置 FB 实例区。



非保持区和保持区设为字。采用时间和计数器数字来设置定时器和计数器区域。

默认值如下：

CS/CJ- 系列 CPU 单元 3.0 版或以后版本和 NSJ 控制器

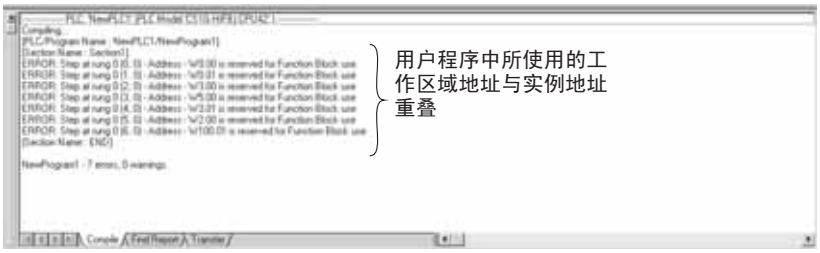
FB 实例区域	默认值			适用存储器区域
	起始地址	终端地址	大小	
非保持 （见注 1 和 3）	H512 （见注 2）	H1407 （见注 2）	896	CIO, WR, HR, DM, EM
保持 （见注 1）	H1408 （见注 2）	H1535 （见注 2）	128	HR, DM, EM
定时器	T3072	T4095	1024	TIM
计数器	C3072	C4095	1024	CNT

注 (1) 即使DM或EM区域规定为非保持区域或保持区域，但仍可以存取位数据。

(2) 功能块保持区域分配在 H512 和 H1535 之间。这些字不能指定于用户程序操作数中。同时，也不能指定于内部变量 AT 设置中。

(3) 字 H512 到 H1535 包括在保持区域中但断电 / 再通电时或开始操作时设为非保持的地址将被清除。

(4) 为防止实例区域地址和程序所用地址重叠，非保持区域和保持区域设在 H512 到 H1535 （功能块保持区域字）之间。如果设置另一区域，则地址与用户程序中所用的地址重叠。如果功能块实例区域地址与用户程序中所使用的用任何地址重叠，编　辑时会出错。当用户下载在线编辑以及检查程序时，该错误也会发生。



如果地址被复制将会出现错误，要么修改功能块实例区域要么修改用户程序中所使用的地址。

FQM1 柔性运动控制器

FB 实例区域	默认值			适用存储器区域
	起始地址	终端地址	大小	
非保持	5000	5999	1000	CIO, WR, DM
保持	无			
定时器	T206	T255	50	TIM
计数器	C206	C255	50	CNT

注 即使 DM 区域规定为非保持区域，但仍可以存取位数据。

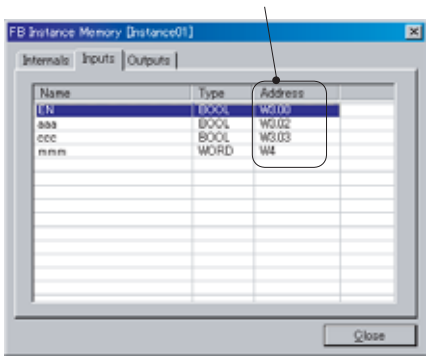
3-2-7 检查变量内部地址分配

以下程序可用来检查内部分配于变量的 I/O 存储器地址。

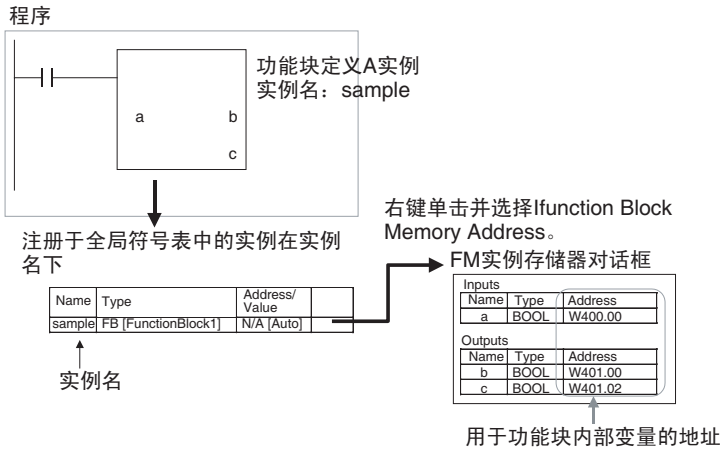
1,2,3...

1. 选择 View - Symbols － Global。
2. 在全局符号表中选择实例。右键单击并从弹出式菜单中选择 Function Block Memory Address。（或从 PLC 菜单中选择 Function Block Memory - Function Block Memory Address 。）

3. 显示 FB 接口存储器对话框。在此处，检查内部分配于变量的 I/O 存储器地址。



用于检查内部分配于变量的地址方法

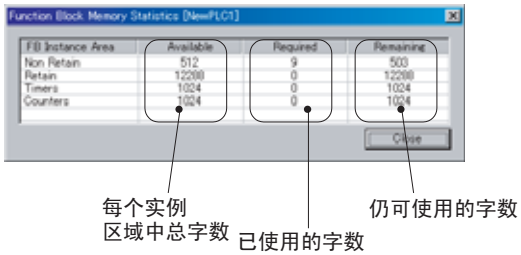


检查内部分配于 变量的地址状态

以下程序用来检查内部分配于变量的地址数。地址数对功能块实例区域分配也适用。

1,2,3...

1. 在梯级段窗口中选择实例，右键单击，从 PLC 菜单中选择 Function Block Memory - Function Block Memory Statistics。
2. 功能块存储器统计对话框如下所示。在此处，检查地址使用情况。



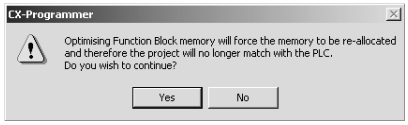
优化功能存储器

当添加或删除变量时，地址自动再分配于变量实例区域中。每个实例要求地址相邻。因此，如果源地址块不能对变量进行更改，则所有变量分配于其他地址块中。这将导致有些地址块空置不用。以下程序用来消除存储器中空置不用区域从而能够更有效地利用存储器。

1,2,3...

1. 在梯级段窗口中选择实例，右键单击，从 PLC 菜单中选择 Function Block Memory - Optimize Function Memory。

   显示以下对话框。



2. 点击 OK 按钮。优化功能块实例区域的分配。

3-2-8 复制编辑功能块定义

采用以下操作来复制编辑已创建的功能块定义。

1. 选择功能块进行复制，右键单击，从弹出式菜单中选择 Copy。
2. 将光标置于功能块项上。功能块项在项目目录中的 PLC 下。右键单击，从弹出式菜单中选择 Paste。
3. 复制功能块定义 （在功能块定义名在复制源处之前显示“ copy ”）。
4. 要修改功能块名，则左键单击，从弹出式菜单中选择 Rename。
5. 双击功能块定义进行编辑。

3-2-9 从实例检查源功能块定义

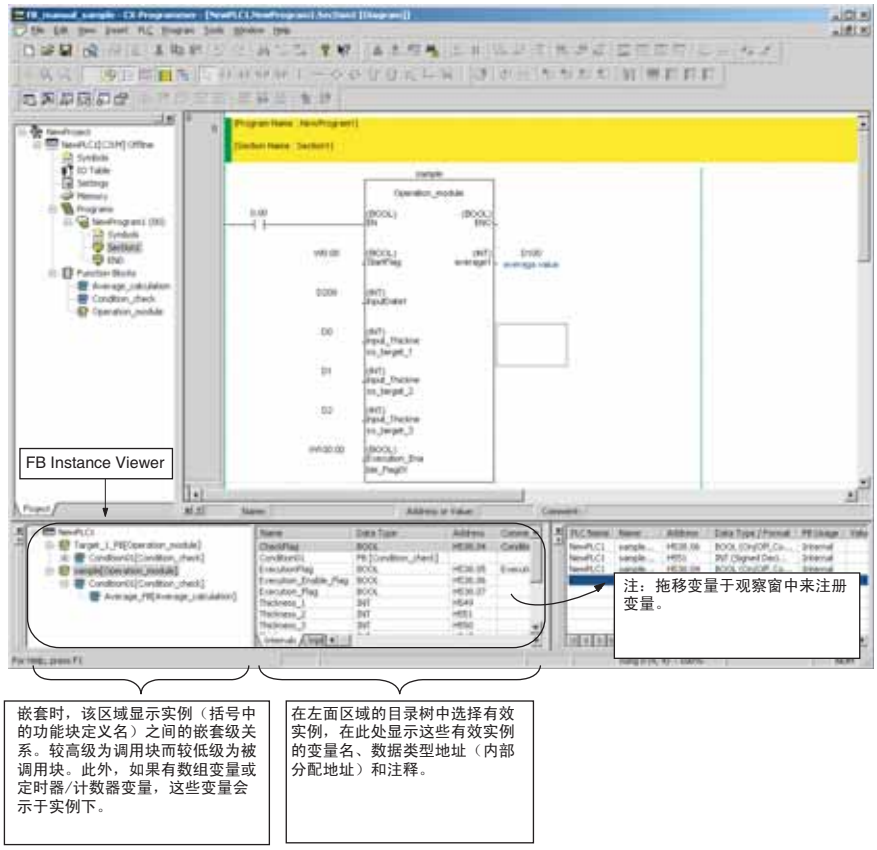
以下程序用来检查已创建实例的功能块定义。

或双击或右键单击实例并从弹出式菜单中选择 Go To -Function Block Definition。显示功能块定义。

3-2-10 检查诸如嵌套级等实例信息

当实例嵌套于已创建的程序中时，从视图菜单中选择 Windows - FB Instance Viewer 来检查嵌套级结构。功能块关系以目录树格式表示。调用功能块在较高级而被调用功能块在较低级。

FB 实例浏览器窗口提供其他信息。例如，正在使用的数组变量和下表所示的分配于变量的内部地址。只需从实例变量表中拖动变量并移至观察窗中即可将变量注册于观察窗中。

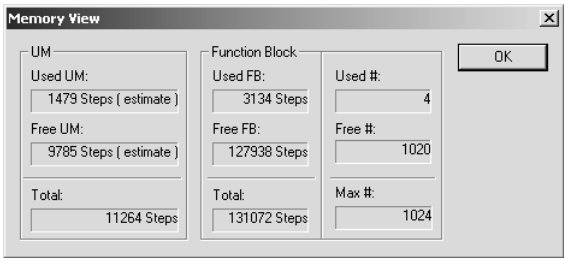


3-2-11 检查功能块定义大小

CX-Programmer 可用来检查创建的功能块定义大小。检查方法与检查程序容量的方法相似。具体程序如下：

1. 从视图菜单中选择 Memory View 。

2. 功能块定义大小和功能块定义数示于存储器视图对话框中。对话框如下所示。



- 功能块下所使用的 #、自由 # 和最大 # 字段请参阅功能块定义数。

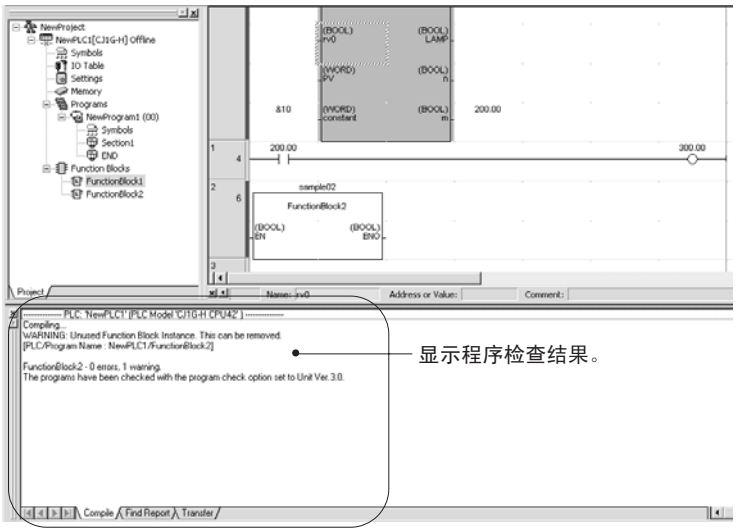
3-2-12 编辑功能块定义 （检查程序）

编辑功能块定义来执行程序检查。

采用以下程序：

1,2,3... 选择功能块定义，右键单击，从弹出式菜单中选择 Compile。（或按 Ctrl + F7 键。）

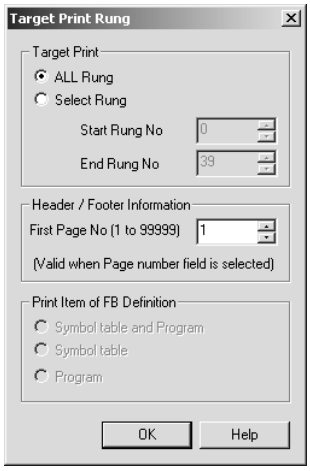
编辑功能块，程序检查结果自动显示于输出窗的编辑表页中。



3-2-13 打印功能块定义

以下程序用来打印功能块定义。

1,2,3... 1. 双击要打印的功能块定义，同时显示变量表和算法。从文件菜单中选择 Print。以下目标梯级对话框显示在屏幕上。



2. 选择 All Rung 或 Select Rung 选项。当选择选择梯级选项时，指定起始梯级和最终梯级号。当在文件 – 页码设置 页眉或页脚区域中已指定了页码时，指定第一页页码。
3. 至于功能块打印范围，则可从以下选项中选择一个选项：
   - 符号表和程序 （默认）
   - 符号表
   - 程序
4. 点击 OK 按钮，显示打印对话框。设置打印机和要打印的项目号以及纸张设置后，点击 OK 按钮。
5. 打印算法 （例如，梯级编程语言）后面的变量表 （列在下面）。

注 有关打印设置的详细内容请参阅 CX-Programmer 5.0 版操作手册 （W437）打印一章内容。

3-2-14 功能块定义密码保护

概述

设置密码限制存取以保护项目中的功能块定义。根据应用情况以下两级密码保护可以设置。

写入和读取密码保护 该密码保护级限制写入 （修改）和显示功能块定义内容。

要设置读 / 写保护，则在功能块属性中选择禁止写和显示作为保护类型。该级保护防止无意程序更改 / 修改并且还可防止程序材料滥用。

密码保护—仅对写

该密码保护等级限制写 （修改）功能块定义内容。

要设置写保护，则在功能块属性中选择禁止写作为保护类型。该级保护防止无意程序更改 / 修改。

设置密码保护

该操作只能在离线状态下进行。

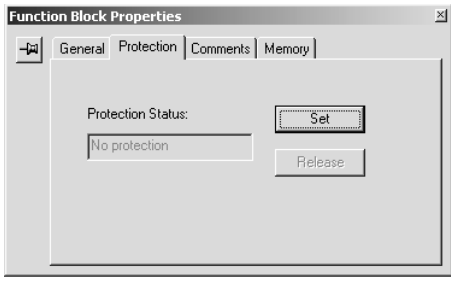
密码保护适用于单个功能块定义或多个功能块定义。

保护单个功能块定义

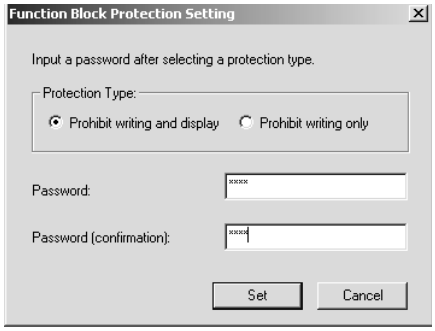
以下程序用来设置单个功能块定义密码保护。

1,2,3...

1. 在项目工作区内，选择功能块定义，右键单击并从弹出式菜单中选择 Properties。（或从视图菜单中选择 Properties。）
2. 显示功能块属性对话框。点击 Protection 条，点击 Set 按钮。



3. 显示功能块保护设置对话框。在保护类型区域中选择保护级。



下表所示的是每个保护等级所限制的功能。

功能	保护类型	
	禁止写和显示	禁止写
显示功能块内容	禁止	允许
打印功能块内容		
编辑功能块内容		禁止
保存 / 载入功能块库文件	禁止	允许

4. 密码输入于功能块项目设置对话框中的密码区域。再次输入相同的密码于确认区域中以验证密码。点击 Set 按钮。

   密码可达 8 个字符长但只有一个字母数字字符可以使用。

5. 当功能块定义受到密码保护时，功能块定义图标发生变化以示其受到保护。如下所示，图标也可显示保护等级。

   : 禁止写和显示 （梯级和 ST 均相同）

   : 禁止写 （梯级）

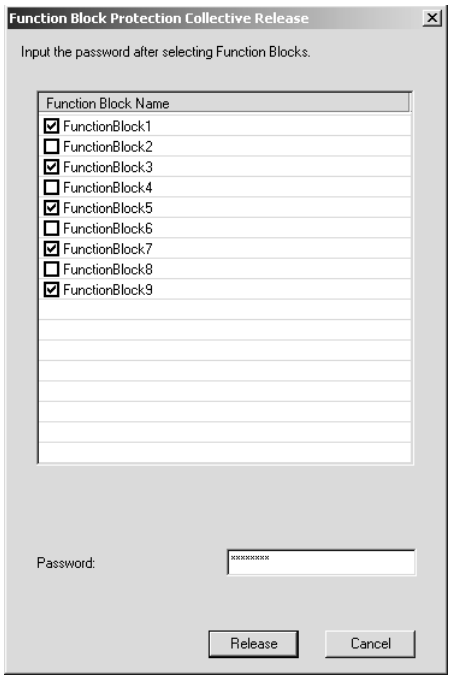
   : 禁止写 （ST）

保护多个功能块定义

以下程序用来同时设置两个或两个以上功能块定义的密码保护。

1,2,3...

1. 在项目工作区中选择 Function Blocks，右键单击并从弹出式菜单中选择 Function Block Protection – Set。
2. 显示功能块保护收集设置对话框。选择您想保护的功能块名。选择保护类型 （保护等级），输入密码，点击 Set 按钮。



3. 选定的功能块定义受到密码保护。

解除密码保护

该操作只能在离线状态下进行。
可以解除单个功能块定义或多个功能块定义密码保护。

解除单个功能块定义密码保护

以下程序用来解除单个功能块定义密码保护。

1,2,3...

1. 在项目工作区中选择功能块定义，右键单击并从弹出式菜单中选择 Properties。（或从视图菜单中选择 Properties。）
2. 显示功能块属性对话框。点击 Protection 条，点击 Release 按钮。
3. 显示功能块保护解除毒化框。密码输入于密码区域中，点击 Release 按钮。
4. 如果密码正确，则解除保护。在项目工作区中，功能块定义图标变为正常图标。

解除多个功能块 定义密码保护

以下程序用来同时解除两个或两个以上功能块定义的密码保护。

1,2,3...

1. 在项目工作区中 ，选择 Function Blocks，右键单击并从弹出式菜单中选择 Function Block Protection － Release。
2. 显示功能块保护收集解除对话框。选择您想去保护的功能块名，输入密码，点击 Release 按钮。
3. 如果输入的密码与选定的功能块密码匹配，则所有功能块定义保护立即被解除。

3-2-15 保存再使用功能块定义文件

分别保存已创建的功能块定义并作为功能块库文件 （*.cxf）以便在另一项目中使用。

注

(1) 在保存于文件之前之前或用于另一项目中之前，编辑功能块定义并进行程序检查。
(2) 当功能块在嵌套时，被调用（嵌套）功能块的功能块定义包含并保存于功能块库文件中。

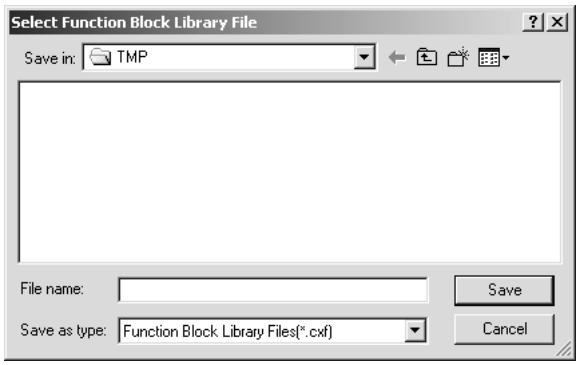
保存功能块库文件

以下程序用来将功能块定义保存于功能块库文件中。

1,2,3...

1. 选择功能块定义，右键单击并从弹出式菜单中选择 Save Function Block to File 。（或从文件菜单中选择 Function Block- Save Function Block to File。）

2. 显示以下对话框。输入文件名。选择功能块库文件（*.cxf）作为文件类型。

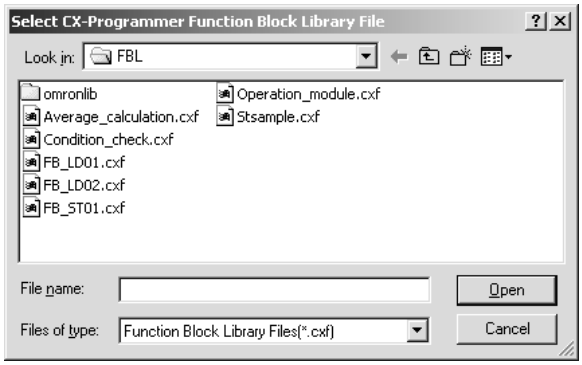


功能块库文件读入其他项目中

以下程序用来将功能块库文件 （*.cxf）读入项目中。

1,2,3...

1. 在项目工作区的 PLC 目录下选择功能块定义，右键单击并从弹出式菜单中选择 Insert Function Block － From File （或选择 File - Function Block - Load Function Block from File）。
2. 显示以下对话框。选择功能块库文件 （*.cxf），点击 Open 按钮。



3. 在功能块图标后插入名为功能块 1 的功能块。该图标包含了功能块定义。
4. 双击 FunctionBlock1 图标。显示变量表和算法。

3-2-16 下载 / 上载程序至实际 CPU 单元中

创建包含功能块的程序后，程序可以从 CX-Programmer 下载至在线连接的实际 CPU 单元中。也可以从实际 CPU 单元中上载程序。还可以检查 CX-Programmer （个人计算机）上的程序与实际 CPU 单元中的程序是否相同。但包含功能块的程序不能下载于任务单元中 （可以上载）。

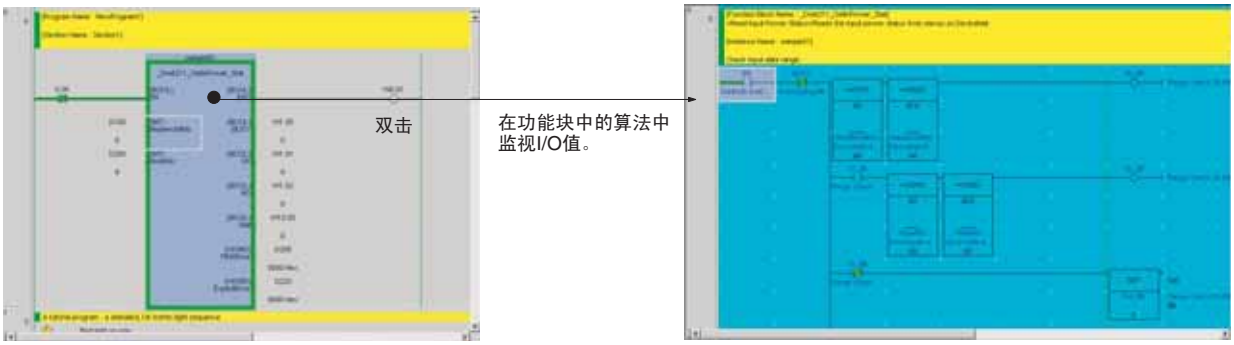
3-2-17 功能块监视调试

以下程序用来监视含功能块的程序。

监视实例中的梯级程序 I/O

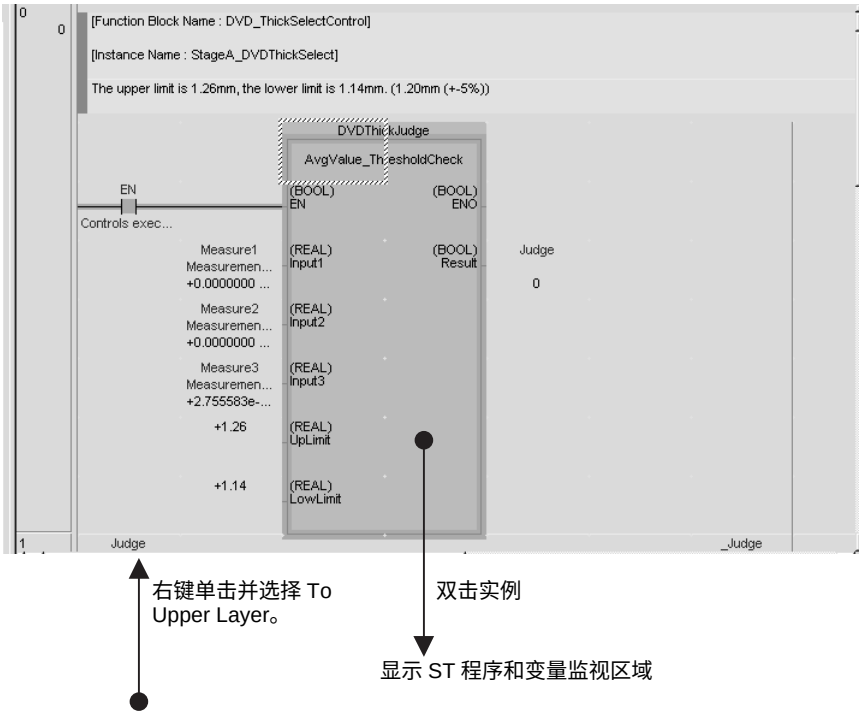
采用 CX-Programmer 6.0 版和以后版本，当监视程序时监视实例中梯级程序位和字内容状态。要监视 I/O 位和字 （I/O 位监视器），双击实例或右键单击实例并从弹出式菜单中选择 Monitor FB Ladder Instance。此时，可以监视位和字、修改 PV、位强制设置 / 复位及监视微分执行情况。

注 不能进行在线编辑或更改定时器 / 计数器 SV。



监视实例中 ST 程序变量

采用 CX-Programmer 6.1 版和以后版本，当监视程序时监视实例中 ST 程序。要监视 I/O 位和字（I/O 位监视器），双击实例或右键单击实例并从弹出式菜单中选择 Monitor FB Ladder Instance。要返回至原实例，则右键单击 ST 程序监视窗口并从弹出式菜单中选择 To Upper Layer。



ST 程序示于窗口 （称作 ST 程序监视窗）左侧。用于 ST 程序的变量值示于窗口 （称作 ST 变量监视窗）右侧。

此时，可以监视变量值、修改 PV、位强制设置 / 复位及复制 / 粘贴变量于观察窗中。（这些操作见如下描述。）

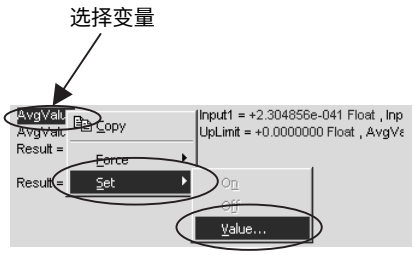
注 ST 程序不能在线编辑。

监视变量

变量值以蓝色示于 ST 变量监视窗中。

修改 PV

在 ST 变量监视窗中选择所需的变量（选中时反白显示），右键单击并从弹出式菜单中选择 Set -Value 来修改 PV。



显示设置新值对话框。新值输入于数值区域中。

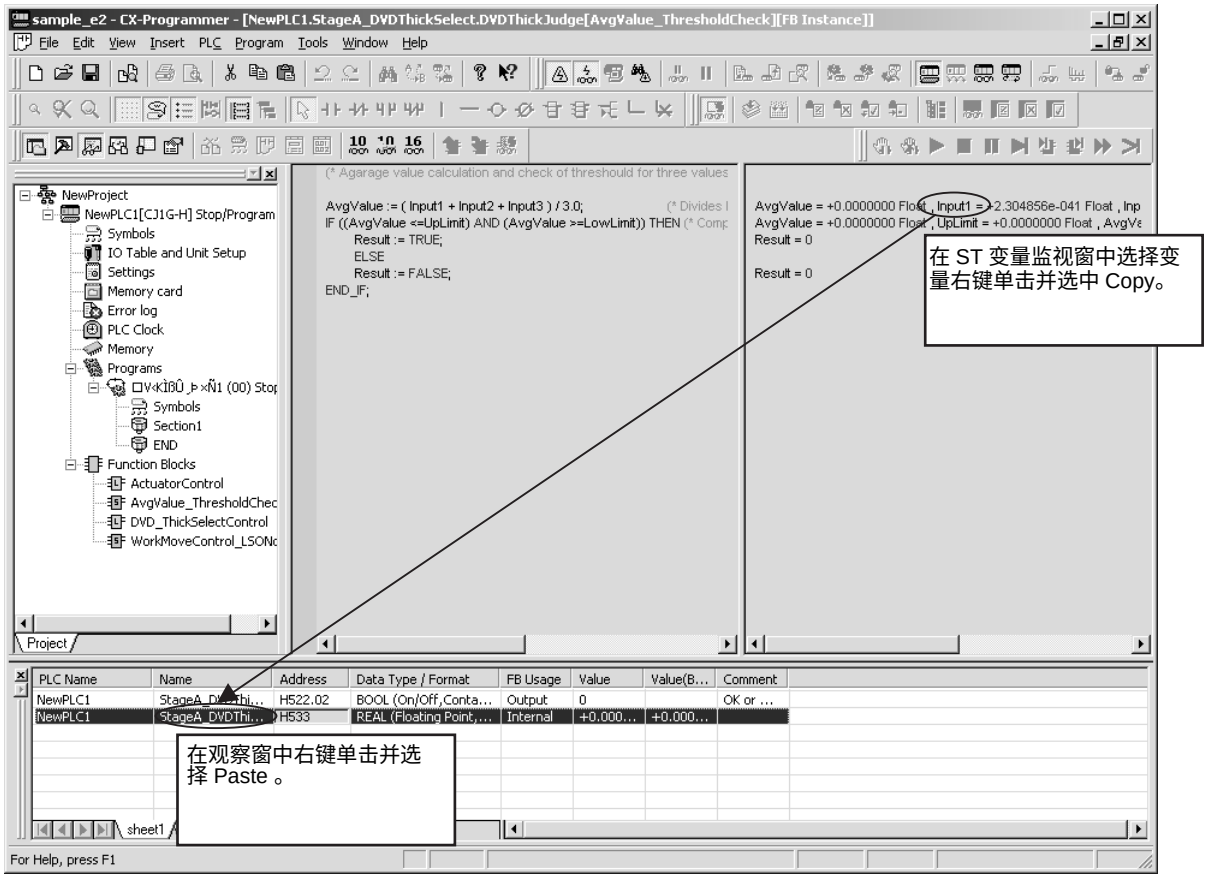
位强制设置和强制复位

在 ST 变量监视窗中选择所需的变量（选中时反白显示），右键单击并从弹出式菜单中选择 Force - On, Force - Off, Force - Cancel 或 Force -Cancel All Force 来强制设置和强制复位或清除强制状态。

复制粘贴于观察窗中

1,2,3...

1. 在 ST 变量监视窗中选择所需的变量（选中时反白显示），右键单击并从弹出式菜单中选择 Copy 将变量复制于观察窗中。
2. 在观察窗中右键单击并从弹出式菜单中选择 Paste 。



检查功能块定义中程序

以下程序用来在监视期间检查势力功能块定义中程序。

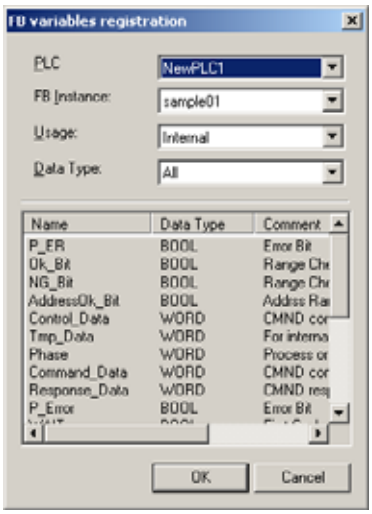
1,2,3... 右键单击实例并从弹出式菜单中选择 To Lower Layer。
显示功能块定义。

监视观察窗中实例变量

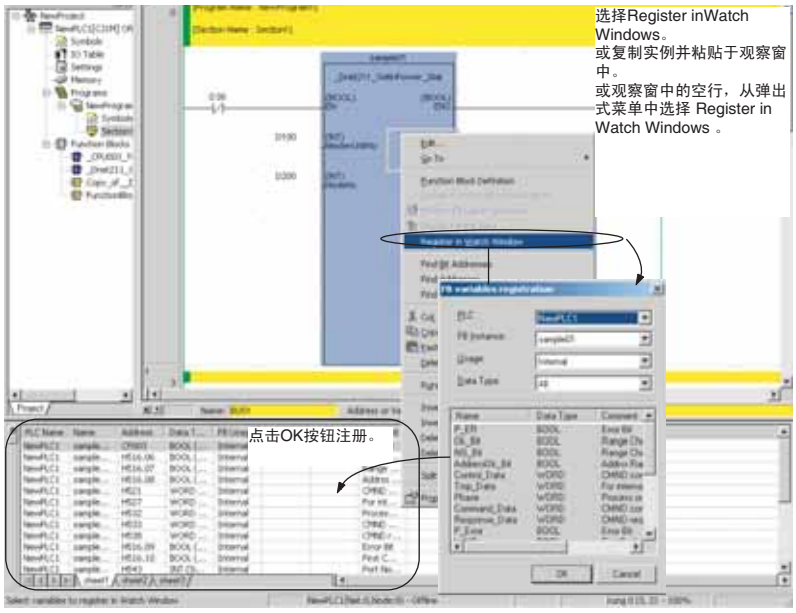
以下程序用来监视实例变量。

1,2,3... 1. 选择 View - Window – Watch。显示观察窗。

2. 采用以下三种方法中的其中一个来显示 FB 变量注册对话框。
   a. 右键单击实例并从弹出式菜单中选择 Register in Watch Windows 。
   b. 复制实例并将其粘贴于观察窗中。
   c. 右键单击观察窗中的空行，从弹出式菜单中选择 Register in Watch Windows 。

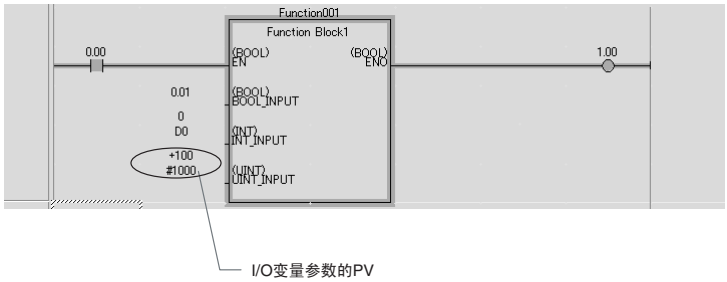


3. S 选择 Usage - Data Type。也可选择 FB Instance 设置。
   默认用法为N: Internal 以及其他变量选择为I: Input, O: Output和E: External。
   默认数据类型为 A: All。也可选择特殊数据类型 BOOL 和 INT。
4. 点击 OK 按钮。选中的变量注册于观察窗中。数值如下所示。



监视实例 I/O 变量

I/O 变量参数当前值示于参数下面。



在线编辑功能块定义程序

在线编辑使用功能块的程序。也可修改实例。

· 更改实例参数、删除实例。更改指令而不是实例中其他内容。

· 不能添加实例。不能更改实例名。功能块定义中的算法和变量表不能更改。

模拟实例中梯级程序 /ST 程序

可以模拟功能块实例中梯级程序或 ST 程序运算。支持分步执行和断点操作。要返回原实例，右键单击 ST 程序监视窗并从弹出式菜单中选择 To Upper Layer。

■启动模拟功能

以下程序用来启动模拟功能。

1,2,3...

1. 打开包含要调试实例的程序。
2. 选择 View － Toolbars，在 Toolbars 条中选择模拟器调试选项。
3. 从 CX-Programmer 的 PLC 菜单中选择 Work Online Simulator 并将程序传输至计算机中的 CX- 模拟器。

注　　以相反次序执行步骤 2 和 3。

■分步执行 （步骤运行）

逐步（指令）递进执行程序。当实例停止时，该功能块移至实例梯级程序或 ST 程序的第一个步骤 （指令）。

采用步骤运行或连续步骤运行方法 （见注释）来执行实例中程序。

注　　选择 CX-Programmer 的 Tools - Options 命令并将连续步骤间距设于 PLC 标记页上来设置连续步骤运行操作的持续时间

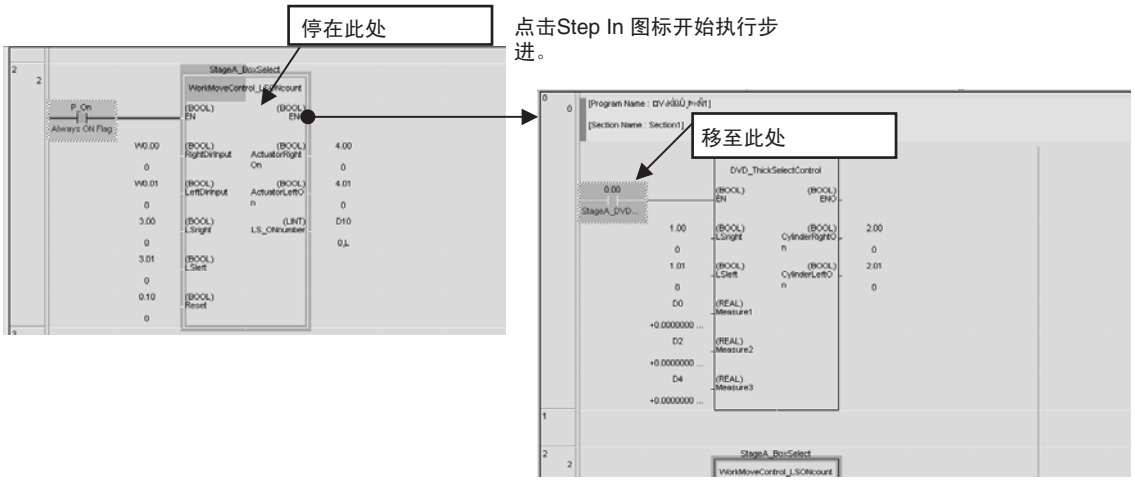
步进

以下程序用来启动实例中梯级程序 /ST 程序分步执行。（称作步骤运行操作）。

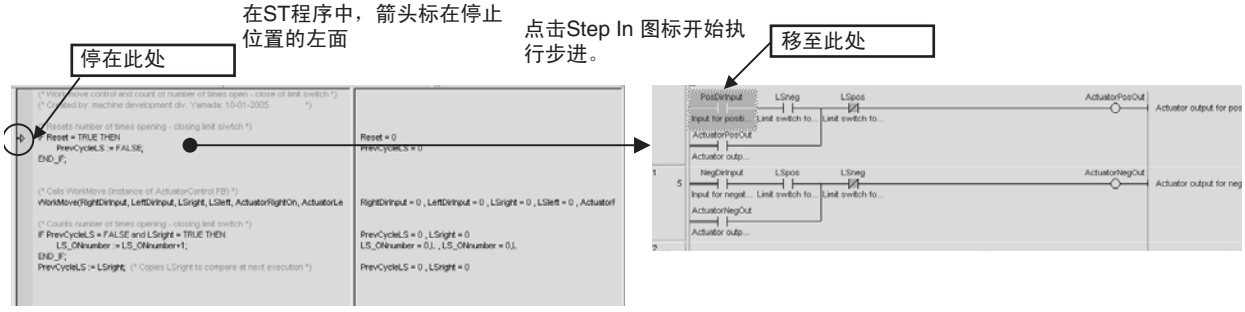
1,2,3...

1. 暂停实例执行 （见注释）。
2. 点击 Step In 图标或选择 Tools - Simulation - Mode - Step In。

举例说明：从实例步进至内部梯级程序



举例说明：从 ST 程序步进至内部梯级程序



注　　当在功能块实例外执行程序时，处理与正常步骤运行操作相同。

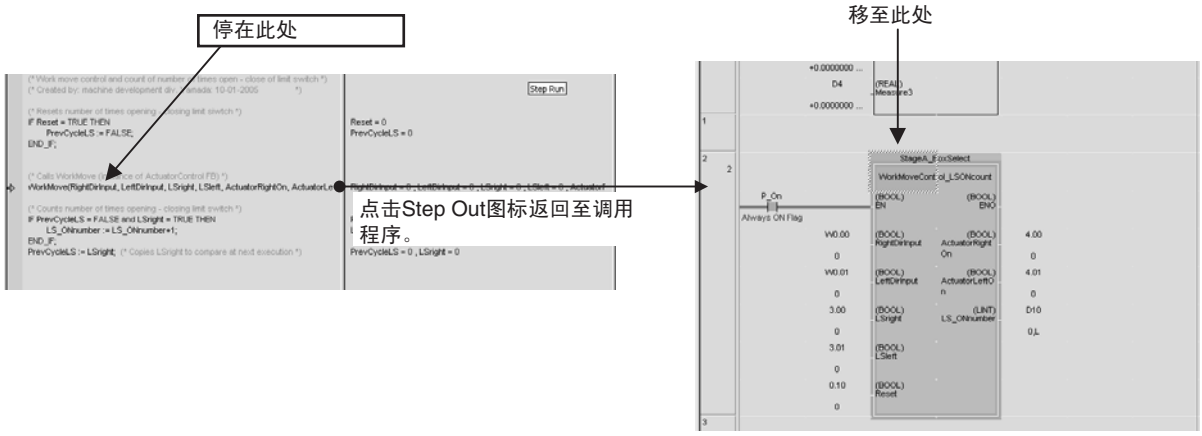
步出

采用以下程序来暂停实例中梯级程序 /ST 程序的分步执行 （步骤运行操作）。返回至比程序更高一级中 （程序或实例是调用源）。

1,2,3... 1. 在步骤运行操作期间，将光标移至实例中任何停止点处。

2. 点击 Step Out 图标或选择 Tools - Simulation - Mode - Step Out。

举例说明：从 ST 程序中返回至调用程序或实例



注 只能在实例的梯级程序 /ST 程序中执行 Step Out 命令。

■当操作被模拟功能暂停时显示

光标颜色 （或 ST 程序中箭头）表示在模拟功能窗口中操作已暂停以及哪个操作已暂停。

调试操作	颜色 （默认）	程序执行状态	详细说明
步骤运行或连续步骤运行	粉红色	模拟器暂停状态	被步骤运行操作或 Pause 按钮暂停
	常规颜色	由于互锁或有其他功能不执行	由于有指令 （例如 IL、MILR/MILH、 JMP0 或 FOR/BREAK。）存在，因此不执行步骤。
断点	蓝色	模拟器指令断开	被断点暂停 （断开状态）。

注 (1) 当已选择 Tools - Simulation - Always Display Current Execution Point 时，模拟器自动滚动显示内容来显示执行步骤运行或连续步骤运行操作时实例中的暂停点。

(2) 当模拟功能窗中的操作暂停时光标颜色 （或 ST 程序中的箭头）可以从默认颜色更改为其他颜色。选择 Tools - Options 并点击 Appearance 条来更改颜色。选择 Pause Simulator、 Simulator Instruction Break 或 Simulator

■IO Break，更改颜色。

在实例预先设定的断点处，自动暂停执行。（在这种情况下，不能使用步进操作。）

注 注 当实例设置断点时，仅该实例断点有效。（从相同功能块定义中创建的其他实例，该断点无效。）

附录 A
数据类型

基本数据类型

数据类型	内容	大小	数值范围
BOOL	位数据	1	0 (FALSE), 1 (TRUE)
INT	整数	16	-32,768 ～ +32,767
DINT	双整数	32	-2,147,483,648 ～ +2,147,483,647
LINT	长 （8 个字节）整数	64	-9,223,372,036,854,775,808 ～ +9,223,372,036,854,775,807
UINT	无符号整数	16	&0 ～ 65,535
UDINT	无符号双整数	32	&0 ～ 4,294,967,295
ULINT	无符号长 （8 个字节）整数	64	&0 ～ 18,446,744,073,709,551,615
REAL	实数	32	-3.402823×10^{38} ～ $-1.175494 \times 10^{-38}$, 0, $+1.175494 \times 10^{-38}$ ～ $+3.402823 \times 10^{38}$
LREAL	长实数	64	$-1.79769313486232 \times 10^{308}$ ～ $-2.22507385850720 \times 10^{-308}$, 0, $2.22507385850720 \times 10^{-308}$ ～ $1.79769313486232 \times 10^{308}$
WORD	16 位数据	16	#0000 ～ FFFF 或 &0 ～ 65,535
DWORD	32 位数据	32	#00000000 ～ FFFFFFFF or &0 ～ 4,294,967,295
LWORD	64 位数据	64	#0000000000000000 ～ FFFFFFFFFFFFFFFF 或 &0 ～ 18,446,744,073,709,551,615
TIMER （见注释）	定时器 （见注 1）	标记：1 位 PV：16 位	定时器数：0 ～ 4095 完成标记：0 或 1 定时器 PV：0 ～ 9999 （BCD）； 0 ～ 65535 （二进制）
COUNTER （见注释）	计数器 （见注 2）	标记：1 位 PV：16 位	定时器数：0 ～ 4095 完成标记：0 或 1 定时器 PV：0 ～ 9999 （BCD）； 0 ～ 65535 （二进制）
FUNCTION BLOCK	功能块实例	---	---

注　　TIMER 和 COUNTER 数据类型不能用于结构化文本功能块中。

导数数据类型

数组	1 维数组；最多 32000 元素

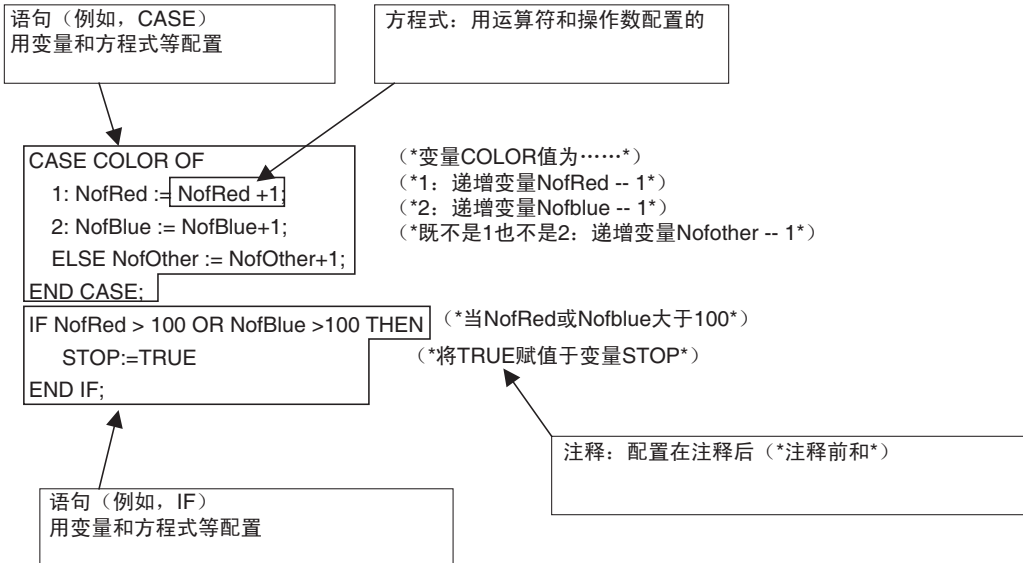
附录 B
结构化文本 （ST 语言）

结构化文本

结构化文本 （也称作 ST 语言）是一个与 PASCAL 相类似的高级编程语言。 PASCAL 使用诸如选择语句和迭代语句的语言结构。使用语句来配置用结构化文本写的程序。用变量和方程式来配置语句。

- 方程式是包含运算符和操作数 （变量或常数）的程序。运算符包括算术运算符、比较运算符和逻辑运算符。
- 语句既可是赋值语句也可是控制语句。赋值语句将方程式中的计算结果保存于变量中。控制语句包括选择语句和迭代语句。

结构化文本范例



限制

语句分隔符

- 语句 （赋值语句和控制语句）必须以分号 （；）结束。按回车不能完成语句。
- 不得将分号 （；）用作带保留字、数值或方程式的语句分隔符。除了在语句终端之外，分隔符插入语句中会导致句法发生错误。

注释

- 注释封闭在扩号和星号内。例如（* 注释 *）。除了扩号和星号之外的任何字符均可用于注释中。不支持注释嵌套。

记号	范例
(* 注释 *)	(* 这是注释 *)

注：不可以注释嵌套。例如， (*(* 不支持这种嵌套 *)*)

空格、回车和制表符

- 任何数量的空格、回车和制表符或其组合均可用于语句中任何地方。因此，在保留字和方程式之间使用空格、回车和制表符以便读取。
- 在以下标识 （用于编辑的最小意义单位）之间不能使用空格、回车和制表符 。在这种情况下，它们作为标识分隔符。

标识：保留字、变量名、特殊字符、常数 （数值）

保留字 （大写或小写： AND, CASE, DO, ELSE, FOR, IT, NOT, OF, OR, REPEAT, THEN, TO, UNTIL, WHILE, XOR, TRUE, FALSE, ELSIF, BY, EXIT, RETURN

变量名：不是保留字的任何文本均识别为变量名。

特殊字符：< =, > =, <> , :=, .., &, (*, *)

常数 （数值）：仅用于十进制数的数值

16# 跟十六进制数的数值
2# 跟二进制数的数值
8# 跟八进制数的数值

如果在上述标识（用于编辑的最小意义单位）之间使用空格、回车和制表符，任一侧处的标识部分作为分隔标识来处理。因此，确保空格、回车和制表符不用于单个标识中。

- 在保留字和变量名之间始终使用空格、回车和制表符或其他标识分隔符。

在下例中，框 （□）表示要求有空格、回车和制表符或其他标识分隔符的地方。

```
IF □ A > 0THEN □ X=10;
ELSE □
  X:=0;
END_IF;
```

大写和小写

- 保留字和变量名不分大小写 （两者均可使用）。

变量名禁止字符

- 以下方扩号内的字符不能用于变量名。

[!], [[#], [$], [%], [&], [注 , [(],, [)], [-], [=], [^], [~], [\], [|], [@], [注 , [[], [{], [;], [+], [:], [*], []], [}], [,], [<], [.], [>], [/], [?]

- 数字 0 到 9 不能用作变量名的第一个字符。
- 在变量名中，下划线不能紧跟着另一根下划线
- 空格不能用于变量名中。

如果这些字符中任何一个在使用，则会出现错误信息。

输入常数 （数值）

- 如下例所示，数值可以通过十进制、十六进制、八进制或二进制来表示。

	表示方法	范例 （用于十进制值 12）
十进制：	仅数值12	
十六进制：	16# 后面跟数值	16#C
八进制：	8# 后面跟数值	8#14
二进制：	2# 后面跟数值	2#1100

运算符优先度

- 考虑到结构化文本语法中的运算符优先度或扩号内需要优先度的运算，

例：AND 比 OR 优先。因此，在 X OR Y AND Z 范例中， Y AND Z 具有优先度。

CX-Programmer 的 ST 输入屏显示

文本显示颜色

当输入或粘贴于 ST 输入屏中时， CX-Programmer 自动以以下颜色显示文本。

- 文本关键字 （反白显示）：蓝色
- 注释：绿色
- 错误：红色
- 其他：黑色

更改字体

选择 Tools - Options，点击 Appearance 条。然后，点击 ST Font 按钮来修改字体大小或显示颜色。字体名称、字体大小 （默认为 8 点）以及颜色也可以修改。

语句

语句	功能	范例
语句结束	结束语句	;
赋值	（* 和 *）之间的所有文本作为注释来处理	(* 注释 *)
分配	替换表达式结果、变量或左面变量的右面值	A:=B;

IF, THEN, ELSIF, ELSE, END_IF	当条件正确时，评估表达式。	IF (条件 _1) THEN
		  (表达 1);
		ELSIF (条件 _2) THEN
		  (表达 2);
		ELSE
		  (表达 3);
		END_IF;

CASE, ELSE, END_CASE	在变量值基础上评估表达式。	CASE (变量) OF
		  1: (表达 1);
		  2: (表达 2);
		  3: (表达 3);
		ELSE
		  (表达 4);
		END_CASE;

FOR, TO, BY, DO, END_FOR	根据初始值、最终值和增量来反复评估表达式。	FOR (标识) := (初始值) TO (最终值) BY (递增) DO (表达); END_FOR;
WHILE, DO, END_WHILE	一旦条件正确，则反复评估表达式。	WHILE (条件) DO (表达); END_WHILE;
REPEAT, UNTIL, END_REPEAT	反复评估表达式直到条件正确。	REPEAT (表达); UNTIL (条件) END_REPEAT;
EXIT	停止重复处理。	EXIT;

语句	功能	范例
RETURN	返回至调用功能块程序中	RETURN;
功能块实例调用	调用另一功能块定义	带 FUNCTION BLOCK 数据类型的变量名 （被调用功能块定义输入变量名：= 调用功能块定义变量名或常数，。。。被调用功能块定义输出变量名 = > 调用功能块定义输出变量名 ...）；

运算符

运算	符号	运算符支持的数据类型	优先度 1：最低 11：最高
括弧和括号	（表达式），数组（变址）		1
功能评估	标识符	取决于功能块 （参阅 2-6 指令支持和操作数限制）	2
指数	**	REAL, LREAL	3
补码	NOT	BOOL, WORD, DWORD, LWORD	4
乘法	*	INT, DINT, UINT, UDINT, ULINT, REAL, LREAL	5
除法	/	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	5
加法	+	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	6
减法	-	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	6
比较	< , > , < =, > =	BOOL, INT, DINT, LINT, UINT, UDINT, ULINT, WORD, DWORD, LWORD, REAL, LREAL	7
等式	=	BOOL, INT, DINT, LINT, UINT, UDINT, ULINT, WORD, DWORD, LWORD, REAL, LREAL	8
不等式	<>	BOOL, INT, DINT, LINT, UINT, UDINT, ULINT, WORD, DWORD, LWORD, REAL, LREAL	8
布尔 AND	&	BOOL, WORD, DWORD, LWORD	9
布尔 AND	AND	BOOL, WORD, DWORD, LWORD	9
布尔专有 OR	XOR	BOOL, WORD, DWORD, LWORD	10
布尔 OR	OR	BOOL, WORD, DWORD, LWORD	11

注 根据数据类型运算。
因此， INT 数据加法结果必须是使用 INT 数据类型的变量。当整数型变量运算时出现进位或借位，需特别小心。例如，使用整数型变量 A=3 和 B= 2 时，如果执行运算（A/B）*2，则 A/B 结果为 1（1.5 – 小数点后面的数值忽略不计），因此 （A/B） *2 = 2。

函数

功能	语法
数函词	绝对值、三角函数等
算术函数	指数 (EXPT)
数据类型转换函数	源数据类型 _TO_ 新数据类型 (变量名)

数函词

以下数函词可以用于结构化文本中。

数函词	自变量数据类型	返回值数据类型	内容	范例
ABS （自变量）	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	INT, DINT, LINT, UINT, UDINT, ULINT, REAL, LREAL	绝对值 （自变量）	a: = ABS (b) （* 存储于变量 a 中的变量 b 绝对值 *）
SQRT （自变量）	REAL, LREAL	REAL, LREAL	平方根：$\sqrt{}$ 自变量	a: = SQRT (b) （* 存储于变量 a 中的变量 b 平方根 *）
LN （自变量）	REAL, LREAL	REAL, LREAL	自然对数：LOG_e 自变量	a: = LN (b) （* 存储于变量 a 中的变量 b 自然对数 *）
LOG （自变量）	REAL, LREAL	REAL, LREAL	常用对数：LOG_{10} 自变量	a: = LOG (b) （* 存储于变量 a 中的变量 b 常用对数 *）
EXP （自变量）	REAL, LREAL	REAL, LREAL	自然指数：$e^{自变量}$	a: = EXP (b) （* 存储于变量 a 中的变量 b 自然指数 *）
SIN （自变量）	REAL, LREAL	REAL, LREAL	正弦：自变量	a: = SIN (b) （* 存储于变量 a 中的变量 b 正弦 *）
COS （自变量）	REAL, LREAL	REAL, LREAL	余弦：自变量	a: = COS (b) （* 存储于变量 a 中的变量 b 余弦 *）
TAN （自变量）	REAL, LREAL	REAL, LREAL	正切：自变量	a: = TAN (b) （* 存储于变量 a 中的变量 b 正切 *）
ASIN （自变量）	REAL, LREAL	REAL, LREAL	反正弦：自变量	a: = ASIN (b) （* 存储于变量 a 中的变量 b 反正弦 *）
ACOS （自变量）	REAL, LREAL	REAL, LREAL	反余弦：自变量	a: = ACOS (b) （* 存储于变量 a 中的变量 b 反余弦 *）
ATAN （自变量）	REAL, LREAL	REAL, LREAL	反正切：自变量	a: = ATAN (b) （* 存储于变量 a 中的变量 b 反正切 *）

注 数函词返回的数据类型与指数中所用的数据类型一样。因此，功能返回抵替换的变量数据类型必须与指数数据类型一致。

算术函数

以下一般指数函数可以用于结构化文本中。

指数函数	自变量数据类型	返回值数据类型	内容	范例
EXPT （底数，指数）	底数：REAL, LREAL 指数：INT, DINT, LINT, UINT, UDINT, ULINT	REAL, LREAL	指数：$底数^{指数}$	（* 作为基数的变量 b 和作为指数的变量 c 的指数存储于变量 a 中 *）

注 数函词返回的数据类型与指数中所用的数据类型一样。因此，功能返回抵替换的变量数据类型必须与指数数据类型一致。

数据类型转换函数

以下数据类型转换函数可以用于结构化文本中。

语法

源数据类型 _TO_ 新数据类型 (变量名)

例子：REAL_TO_INT (C)

在本例中，变量 C 的数据类型可以从 REAL 修改为 INT。

数据类型组合

下表给出了可以转换的数据类型组合。

（是 = 可以转换；否 = 不能转）

从	至											
	BOOL	INT	DINT	LINT	UINT	UDINT	ULINT	WORD	DWORD	LWORD	REAL	LREAL
BOOL	否	否	否	否	否	否	否	否	否	否	否	否
INT	否	否	是	是	是	是	是	是	是	是	是	是
DINT	否	是	否	是	是	是	是	是	是	是	是	是
LINT	否	是	是	否	是	是	是	是	是	是	是	是
UINT	否	是	是	是	否	是	是	是	是	是	是	是
UDINT	否	是	是	是	是	否	是	是	是	是	是	是
ULINT	否	是	是	是	是	是	否	是	是	是	是	是
WORD	否	是	是	是	是	是	是	否	是	是	否	否
DWORD	否	是	是	是	是	是	是	是	否	是	否	否
LWORD	否	是	是	是	是	是	是	是	是	否	否	否
REAL	否	是	是	是	是	是	是	否	否	否	否	是
LREAL	否	是	是	是	是	是	是	否	否	否	是	否

语句详细说明

赋值

概述

用语句 （方程式、变量或常数）右侧来替代语句 （变量）左侧。

保留字

:=

冒号 (:) 和等号 (=) 组合。

语句语法

变量 : = 方程式、变量或常数；

用法

赋值语句用于变量输入值。在控制语句之前或之内这是基本语句。该语句可以用来设置初始值、保存计算结果和递增或递减变量。

描述

替换 （保存）变量的方程式、变量或常数。

举例说明

例 1：方程式 X+1 结果替代变量 A。

```
A:=X+1;
```

例 2：变量 B 数值替代变量 A。

```
A:=B;
```

例 3：用常数 10 替代变量 A。

```
A:=10;
```

注意事项
要赋值的方程式、变量或常数的数据类型必须与要替代的变量数据类型一致。否则，语法会出现错误。

控制语句

IF 语句 （单一条件）

概述
当符合指定条件时，该语句用来执行表达式。如果不符合条件，则执行其他表达式。

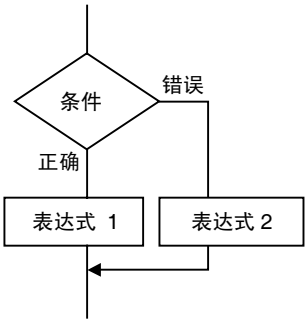
保留字
IF, THEN, (ELSE), END_IF

注 ELSE 不能省略。

语句语法

```
IF ＜条件＞ THEN
  ＜表达式 _1 ＞;
ELSE
  ＜表达式 _2 ＞;
END_IF;
```

流程图



用法
使用 IF 语句来进行其他运算。这将取决于是否满足单一条件 （条件方程式）。

描述
条件 = 如果成立 , 执行表达式 _1

条件 = 如果不成立 , 执行表达式 _2

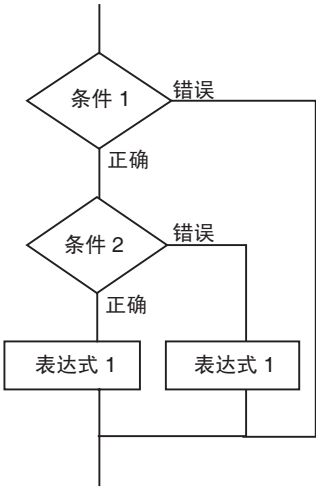
注意事项

- IF 必须与 END_IF 一起使用。
- 条件必须包括赋值结果的正确或错误方程式。
  例子：IF(A ＞ 10)
  条件还可指定为布尔变量而不是方程式。因此，变量值为 1 (ON) = 结果成立； 0 (OFF) = 结果不成立。
- 用于表达式 _1 和表达式 _2 的语句为赋值语句、 IF、 CASE、 FOR、 WHILE 或 REPEAT。

例子：

```
IF ＜条件 _1 ＞ THEN
 IF ＜条件 _2 ＞ THEN
  ＜表达式 _1 ＞;
 ELSE
  ＜表达式 _2 ＞:
  END_IF;
END_IF;
```

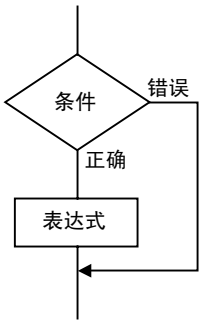
流程图如下：



如上图所示，在此之前， ELSE 与 THEN 立即对应。

- 在表达式 _1 和表达式 _2 内，可以执行多个语句。在表达式 中，多个语句之间使用分号 (;) 分隔符。
- ELSE 语句可以省略。当省略 ELSE 时，如果条件方程式结果不正确，则无运算执行。

流程图



举例说明

例 1：如果变量 A ＞ 0 是正确的，则数值 10 替代变量 X。如果 A ＞ 0 是错误的，则数值 0 替代变量 X。

```
IF A ＞ 0 THEN
  X:=10;
ELSE
  X:=0;
END_IF;
```

例 2：如果变量 A ＞ 0 和变量 B ＞ 1 均正确，则数值 10 替代变量 X 而数值 20 替代变量 Y。如果变量 A ＞ 0 和变量 B ＞ 1 均错误，则变量 X 和变量 Y 均被数值 0 替代。

```
IF A ＞ 0 AND B ＞ 1 THEN
  X:=10; Y:=20;
ELSE
  X:=0; Y:=0;
END_IF;
```

例 3：布尔（BOOL 数据类型）变量 A=1(ON)，用数值 10 替代变量 X。如果变量 A=0(OFF)，用数值 0 替代变量 X。

```
IF A THEN X:=10;
ELSE X:=0;
END_IF;
```

IF 语句 （多个条件）

概述

当符合指定条件时，该语句用来执行表达式。如果不符合第一个条件但另一个条件符合的话，则执行相应表达式。如果无一条件符合，则执行其他表达式。

保留字

IF, THEN, ELSIF, (ELSE)

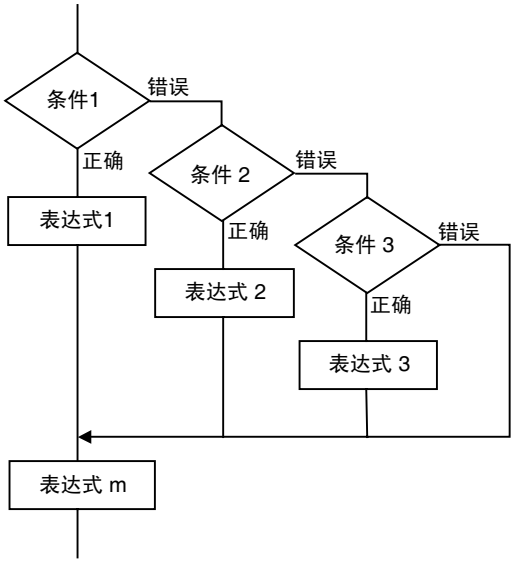
注 ELSE 不能省略。

语句语法

```
IF ＜条件 _1 ＞ THEN ＜表达式 _1 ＞;
   ELSIF ＜条件 _2 ＞ THEN ＜表达式 _2 ＞;
   ELSIF ＜条件 _3 ＞ THEN ＜表达式 _3 ＞;

            ...
   ELSIF ＜条件 _n ＞ THEN ＜表达式 _n ＞;
ELSE ＜表达式 _m ＞;
END_IF;
```

流程图



用法

根据多个条件中哪个条件是符合的来使用 IF 语句执行不同运算 （条件 方程式）

描述
条件 1 = 如果成立，执行表达式 1

条件 1 = 如果不成立，

条件 2 = 如果成立，执行表达式 2

条件 2 = 如果不成立，

条件 3 = 如果成立，执行表达式 3

等

条件 n = 如果成立，执行表达式 n

如无一条件符合，则执行条件 m

注意事项

- IF 必须与 END_IF 一起使用。
- 条件 _ □ 包含方程式正确或错误结果 （例如， IF(A ＞ 10)。
  只有布尔 （布尔数据类型）变量变还可指定为条件而不是方程式。
  至于布尔条件，则当变量值为 1 （ON）时结果正确而变量值为 0 （OFF）时结果错误。
- · 可以用于表达式 _ □ 的语句为赋值语句、 IF、 CASE、 FOR、 WHILE 或 REPEAT。
- · 表达式 _ □ 中可以执行多个语句。在表达式 中，多个语句之间使用分号 (;) 分隔符。
- · ELSE 语句可以省略。当省略 ELSE 时，如果任何条件方程式结果不正确，则无运算执行。

举例说明
例 1：如果变量 A ＞ 0 是正确的，则数值 10 替代变量 X。

如果 A ＞ 0 是错误的但变量 B=1，则数值 1 替代变量 X。

如果 A ＞ 0 是错误的但变量 B=2，则数值 2 替代变量 X。

如果符合这些条件中任何一个条件，则数值 0 替代变量 X。

```
IF A ＞ 0 THEN X:=10;
      ELSIF B=1 THEN X:=1;
      ELSIF B=2 THEN X:=2;
ELSE X:=0;
END_IF;
```

CASE 语句

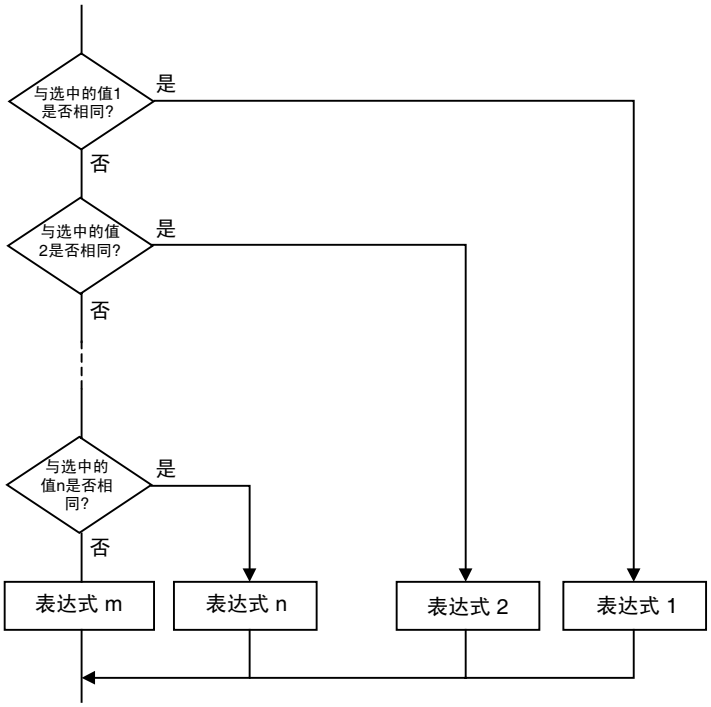
概述
该语句执行选中的与整数方程值匹配的整数。如果选中的整数值不一样，则既不能执行表达式也不能执行规定的表达式。

保留字
CASE

语句语法

```
CASE ＜整数方程式＞ OF
   ＜整数方程式值 _1＞:＜表达式 _1＞;
   ＜整数方程式值 _2＞:＜表达式 _2＞;
                  ...
   ＜整数方程式值 _n＞:＜表达式 _n＞;
ELSE ＜表达式 _m＞;
END_CASE;
```

流程图



用法

根据指定的整数值，使用 CASE 语句来执行不同运算。

描述

如果整数方程式与整数方程式值 _n 匹配，则执行表达式 _n。

如果整数方程式与任何整数方程式值 _n 不匹配，则执行表达式 _m。

注意事项

- CASE 必须与 END_CASE 一起使用。

  tinteger_equation 结果必须是整数格式 （INT、 DINT、 LINT、 UINT、 UDINT 或 ULINT）。
- 可以用于表达式 _ □ 的语句为赋值语句、 IF、 CASE、 FOR、 WHILE 或 REPEAT。
- 表达式 _ □ 中可以执行多个语句。在表达式 中，多个语句之间使用分号 (;) 分隔符。
- 整数格式（INT、DINT、LINT、UINT、UDINT 或 ULINT）中的变量或返回整数值的方程式可以在整数方程式中进行规定。
- 当OR逻辑用于整数方程式值_n中的多个整数时，用逗号分隔符来隔开数值。在第一个整数和最后一个整数之间用两个句点 (..) 作为分隔符来规定整数顺序。

举例说明

例 1：如果变量 A 是 1，数值 1 替代变量 X。如果变量 A 是 2，数值 2 替代变量 X。如果变量 A 是 3，数值 3 替代变量 X。如果无一匹配，则数值 9 替代变量 Y。

```
CASE A OF
    1:X:=1;
    2:X:=2;
    3:X:=3;
ELSE Y:=0;
END_CASE;
```

例 2：如果变量 A 是 1，数值 1 替代变量 X。如果变量 A 为 2 或 5，则数值 2 替代变量 X。如果变量 A 为 6 到 10 之间的数值，则数值 3 替代变量 X。如果变量 A 为 11、 12 或 15 到 20 之间的数值，则数值 4 替代变量 X。如果无一匹配，则数值 9 替代变量 Y。

```
CASE A OF
1:X:=1;
2,5:X:=2;
6..10:X:=3;
11,12,15..20:X:=4;
ELSE Y:=0;
END_CASE;
```

FOR 语句

概述

该语句用来反复执行指定的方程式直到变量 （此处作为迭代变量）到达规定值。

保留字

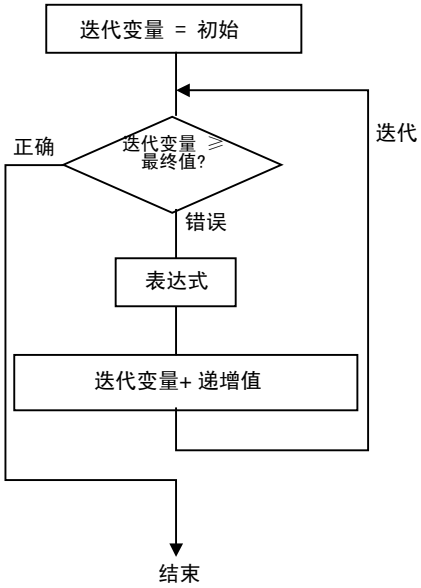
FOR, TO, (BY), DO, END_FOR

注 BY 可以省略。

语句语法

```
FOR ＜循环变量＞ := ＜初始值＞ TO ＜最终值表达式＞ BY ＜递增值表达式＞
DO
 ＜表达式＞;
END_FOR;
```

流程图



用法

当事先已确定迭代数时，使用FOR 语句。当根据指定的迭代变量值在数组变量中转换元素数时，FOR特别有用。

描述

当循环变量为初始值时，执行表达式。执行后，从递增方程式获得的数值添加至循环变量中。如果循环变量＜最终值表达式 （见注 1），则执行 表达式。执行后，从递增方程式获得的数值添加至循环变量中。如果循环变量＜最终值方程式的值 （见注 1），执行表达式 。重复该过程。如果循环变量≥最终值方程式 （见注 2），处理结束完成。

注 (1) 如果递增方程式值是负值，则条件是循环变量＞最终值方程式值。

(2) 如果递增方程式值是负值，则条件是循环变量≤最终值方程式值。

注意事项

- 可以在递增方程式中规定负值。
- FOR 必须与 END_FOR 一起使用。
- 初始值、最终值方程式和最终值方程式必须是整数数据类型（INT、DINT、LINT、UINT、UDINT 或 ULINT）。
- 最终值处理结束后，迭代值递增至最终值 + 1。迭代处理结束。
  例子：在以下结构化文本中，“ a ”值是 TRUE。

```
FOR i:=0 TO 100 DO
  array[i]:=0;
END_FOR;


IF i=101 THEN
   a:=TRUE;
ELSE
   a:=FALSE;
END_IF;
```

- 不得直接在更改迭代变量中使用 FOR 语句。这样做会导致意外运算。
  例子：

```
FOR i:=0 TO 100 BY 1 DO
  array[i]:=0;
  i:=i+5;
END_FOR;
```

- 可以用于表达式 _ □ 的语句为赋值语句、 IF、 CASE、 FOR、 WHILE 或 REPEAT。
- 表达式 _ □ 中可以执行多个语句。在表达式 中，多个语句之间使用分号 (;) 分隔符。
- BY 递增方程式可以省略。省略时， BY 取为 1。
- 在初始值、最终值方程式和递增方程式中可以规定带整数数据类型 （INT、 DINT、 LINT、 UINT、 UDINT 或 ULINT）或可返回整数值方程式的变量。

例 1：当迭代变量 n = 0 ~ 50 （以 5 递增）和数组变量 SP[n] 被 100 替代时，执行迭代。

```
FOR n:=0    TO 50  BY 5 DO
  SP[n]:=100;
END_FOR;
```

例 2：计算数组变量 DATA[n] 的元素 DATA[1] 到 DATA[50] 总值并替代变量 SUM。

```
FOR n:=0    TO 50  BY 1 DO
 SUM:=SUM+DATA[n];
END_FOR;
```

例 3：检测数组变量 DATA[n] 的元素 DATA[1] 到 DATA[50] 的最大值和最小值。最大值替代变量 MAX 而最小值替代变量 MIN。 DATA[n] 数值在 0 到 1000 之间。

```
MAX:=0;
MIN:=1000;
FOR n:=1    TO 50  BY 1 DO
 IF DATA[n] ＞ MAX THEN
 MAX:=DATA[n];
 END IF;
 IF DATA[n] ＜ MIN THEN
 MIN:=DATA[n];
 END IF;
END_FOR;
```

WHILE 语句

概述

只要指定的条件正确，该语句就一直反复执行指定表达式。

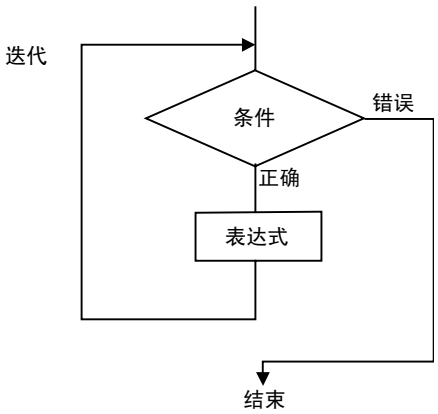
保留字

WHILE, DO, END_WHILE

语句语法

```
WHILE ＜条件＞ DO
 ＜表达式＞;
END_WHILE;
```

流程图



用法

当事先未确定迭代数时，使用 WHILE 语句（根据符合的条件）在符合条件的期间重复规定的处理。只有当条件方程式处于正确 （预测试回路）时才能执行该语句。

描述

在执行表达式 之前，评估条件。

如果条件 正确，执行表达式 I。然后，再次评估条件。重复该过程。如果条件 错误，则不执行表达式，条件 评估结束。

注意事项

- WHILE 必须与 END_WHILE 一起使用。
- 在执行表达式之前，如果条件 方程式是错误的，则不执行表达式情况下处理结束。
- 可以用于表达式 的语句为赋值语句、 IF、 CASE、 FOR、 WHILE 或 REPEAT。
- 表达式 中可以执行多个语句。在表达式 中，多个语句之间使用分号 (;) 分隔符。
- 条件还可规定为布尔变量 （布尔数据类型）而不是方程式。

举例说明

例 1：计算大于 1000 的数值 （以 7 递增）并替代变量 A。

```
A:=0;
WHILE A ＞ =1000 DO
 A:=A+7;
END_WHILE;
```

例 2：当 X ＜ 3000 时， X 值翻倍而且数值替代数组变量。

DATA[1]：然后， X 值再乘以 2。数值替代数组变量。

DATA[2]：重复该过程。

```
n:=1’
WHILE X ＜ 3000 DO
 X:=X*2;
 DATA[n]:=X;
 n:=n+1;
END_WHIE;
```

REPEAT 语句

概述

该语句用来反复执行表达式直到指定的条件正确。

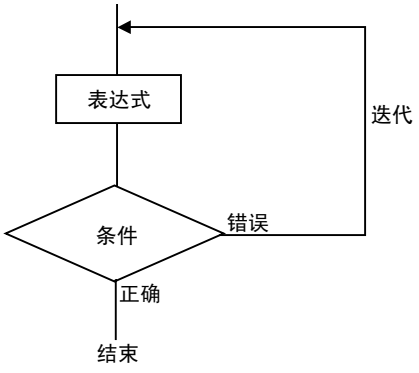
保留字

REPEAT, UNTIL, END_REPEAT

语句语法

```
REPEAT
 ＜表达式＞;
UNTIL ＜条件＞
END_REPEAT
```

流程图



用法
如果事先未确定迭代数时，在规定处理后一旦条件符合即采用 REPEAT 语句来重复处理 （取决于条件是否符合。）。该语句可用来确定是否要根据规定处理执行结果进行反复处理 （处理后回路）。

描述
第一次，无条件执行表达式。然后，评估条件 方程式。

如果条件 是错误的，再次执行表达式。如果条件 是正确的，则不执行表达式情况下处理结束。

注意事项

- REPEAT 必须与 END_REPEAT 一起使用。
- 即使在执行表达式 之前条件 方程式是正确的，表达式仍执行。
- 可以用于表达式的语句为赋值语句、 IF、 CASE、 FOR、 WHILE 或 REPEAT。
- 表达式中可以执行多个语句。在表达式 中，多个语句之间使用分号 (;) 分隔符。
- 条件 还可规定为布尔变量 （布尔数据类型）而不是方程式。

举例说明
例 1：递增从 1 到 10 的数值。总和替代变量 TOTAL。

```
A:=1;
TOTAL:=0;
REPEAT
  TOTAL:=TOTAL+A;
  A:=A+1;
UNTIL A ＞ 10
END_REPEAT;
```

EXIT 语句

概述
仅在迭代语句 （FOR、 WHILE、 REPEAT）中使用该语句从而强制迭代语句结束。当规定的条件符合时，该语句也可用于 IF 语句从而强制迭代语句结束。

保留字
EXIT

语句语法 （例子：在 IF 语句中使用）

```
FOR (WHILE, REPEAT) 表达式
                    ...
IF ＜条件＞ THEN EXIT;
END_IF;

                    ...
END_FOR (WHILE, REPEAT);
```

用法
在结束条件符合之前，使用 EXIT 语句来强制迭代处理结束。

描述 （例子：在 IF 语句中使用）
当条件 方程式是正确的时，迭代语句 （FOR、 WHILE、 REPEAT）强制结束。不执行 EXIT 后的任何语句。

注 (1) 条件 还可规定为布尔变量 （布尔数据类型）而不是方程式。

(2) 即使在执行表达式 之前条件 方程式是正确的，表达式仍执行。

举例说明
当变量 n = 1 到 50 （以 1 递增）且 n 添加至数组变量 DATA[n] 中时，重复处理。

但如果 DATA[n] 大于 100，则处理结束。

```
FOR n:=1; TO 50 BY 1 DO
  DATA[n]:=DATA[n]+n;
  IF DATA[n] ＞ 100 THEN EXIT;
  END_IF;
END_FOR;
```

RETURN 语句

概述
在结构化文本中功能块完成之前必须强制其结束时，该语句用来执行调用程序中功能块处后的下一指令。

保留字
RETURN

语句语法

```
RETURN;
```

用法
当功能块已强制结束时，使用 RETURN 语句。

功能块调用语句

概述
该语句用来调用另一功能块定义。

保留字
无

语句语法
在规定了实例名 （见注释）后，在括号内规定自变量 （调用功能块输入变量传至被调用功能块输入变量）和返回值 （被调用功能块输出变量传至调用功能块输出变量）。

注 实例名可以是任何具有功能块数据类型的内部变量。

以下两种方法中的任一一个均可用来输入功能调用语句。

1. 规格方法 A：规定被调用和调用功能块变量名

实例名 （被调用功能块定义输入变量名 = 调用功能块定义变量名或常数 , ..., 被调用功能块定义输出变量名或常数 = ＞ 调用功能块定义输出变量名 , ...）；

注 所有输入变量规格（被调用功能块定义输入变量名：= 调用功能块定义变量名或常数）必须用逗号分开。
只有所需的输出变量规格 （被调用功能块定义输出变量名或常数 = ＞ 调用功能块定义输出变量名）必须用逗号分开。

2. 规格方法 B：只规定调用功能块变量名 （或常数），省略被调用功能块变量名；

实例名 （被调用功能块定义输入变量名 = 调用功能块定义变量名或常数 , ..., 被调用功能块定义输出变量名或常数 = ＞ 调用功能块定义输出变量名 , ...）；

注 如上所示，当被调用功能块定义输入和输出变量名被省略时，调用功能块输入变量（或常数）值按注册于变量表中的变量先后次序自动传输至被调用功能块输入变量。同样，被调用功能块输出变量按注册于变量表中的变量先后次序自动返回至调用功能块输出变量。

用法

采用功能块调用语句从 ST 语言程序中调用功能块定义 （ST 或梯级程序）。

描述

1. 以下实例注册于变量表中的内部变量中。

内部变量元素	内容	范例
名称	任何实例名	Calcu_execute
数据类型	FUNCTION BLOCK	FUNCTION BLOCK
FB 定义	选择被调用功能块定义	计算

2. 如下例所示，在注册的实例名 （本例中的 Calcu_execute）后的扩号内，指定在变量之间传输的数值。语句结束用分号表示。

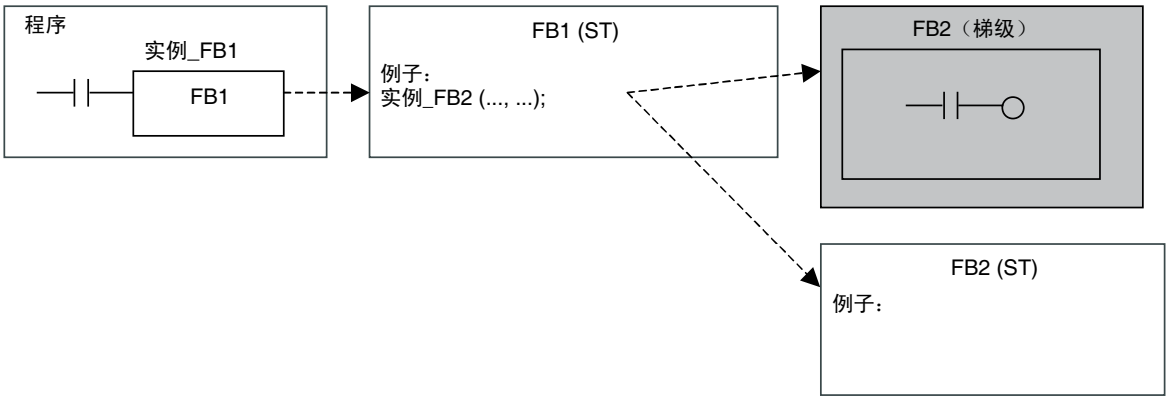
Calcu_execute (A:=B,C= ＞ D) ;

输入变量 B （在调用功能块中）数值传输至输入变量 A （在被调用功能块中）。象输出变量 C （在被调用功能块中）数值那样，该值返回至输出变量 D （在调用功能块中）。

举例说明

在下例中，从功能块 1 （FB1）中调用功能块。

- 用 ST 语言写功能块 1。
- 既可用梯级语言也可用 ST 语言写功能块 2。



INSTANCE _FB1是数据类型FUNCTION BLOCK的实例名。

- 下表所示的是功能块 1 中的变量和接收 / 传输数据的功能块 2 中对应的变量。

变量类型	变量名	传输至 FB2/ 从 FB2 中上传
输入变量	FB1_IN1	传输至 FB2_IN1
	FB1_IN2	传输至 FB2_IN2
	FB1_IN3	传输至 FB2_IN3

变量类型	变量名	传输至 FB2/ 从 FB2 中上传
输入变量	FB1_OUT1	从 FB2_OUT1 中接收
	FB1_OUT2	从 FB2_OUT2 中接收
	FB1_OUT3	从 FB2_OUT3 中接收
内部变量	A 注：数据类型 = 布尔	传输至 EN
	B 注：数据类型 = 布尔	传输至 ENO
内部变量 （实例）	实例 _FB2 注：数据类型 =FUNCTION BLOCK	被调用功能块定义：功能块 2

- 下表所示的是功能块 2 中的变量和接收 / 传输数据的功能块 1 中对应的变量。

变量类型	变量名	传输至 FB2/ 从 FB2 中上传
输入变量	FB2_IN1	从 FB1_IN1 中接收
	FB2_IN2	从 FB1_IN2 中接收
	FB2_IN3	从 FB1_IN3 中接收
输出变量	FB2_OUT1	传输至 FB1_OUT1
	FB2_OUT2	传输至 FB1_OUT2
	FB2_OUT3	传输至 FB1_OUT3

例 1：规格方法 A （指定 FB1 和 FB2 变量）

Instance_FB2(EN:=A,FB2_IN1:=FB1_IN1,FB2_IN2:=FB1_IN2,FB2_IN3:=FB1_IN3,FB2_OUT1= ＞ FB1_OUT1,FB2_OUT2= ＞ FB1_OUT2,FB2_OUT3= ＞ FB1_OUT3,ENO= ＞ B)

- 自变量和返回值以不规则次序排列，这没有关系。
- 输入变量的自变量必须以列表开始或如果列出 EN 变量时列在 EN 变量后面。
- 规格方法 B 不能与规格方法一起用于相同的功能块调用语句中。

其他语句变化：

- 无 EN 的语句

Instance_FB2(FB2_IN1:=FB1_IN1,FB2_IN2:=FB1_IN2,FB2_IN3:=FB1_IN3,FB2_OUT1= ＞ FB1_OUT1,FB2_OUT2= ＞ FB1_OUT2,FB2_OUT3= ＞ FB1_OUT3,ENO= ＞ B)

- 无 EN 和 ENO 的语句

Instance_FB2(FB2_IN1:=FB1_IN1,FB2_IN2:=FB1_IN2,FB2_IN3:=FB1_IN3,FB2_OUT1= ＞ FB1_OUT1,FB2_OUT2= ＞ FB1_OUT2,FB2_OUT3= ＞ FB1_OUT3)

- 无 ENO 语句

Instance_FB2(EN:=A,FB2_IN1:=FB1_IN1,FB2_IN2:=FB1_IN2,FB2_IN3:=FB1_IN3,FB2_OUT1= ＞ FB1_OUT1,FB2_OUT2= ＞ FB1_OUT2,FB2_OUT3= ＞ FB1_OUT3)

- 无 FB2_OUT2 （不需要 FB2_OUT2 数据。）的语句

Instance_FB2(EN:=A,FB2_IN1:=FB1_IN1,FB2_IN2:=FB1_IN2,FB2_IN3:=FB1_IN3,FB2_OUT1= ＞ FB1_OUT1,FB2_OUT3= ＞ FB1_OUT3,ENO= ＞ B)

- 无 FB2_OUT2 （不需要 FB2_OUT2 数据。）的语句

Instance_FB2(EN:=A,FB2_IN1:=FB1_IN1,FB2_IN2:=FB1_IN2,FB2_IN3:=FB1_IN3,FB2_OUT1= ＞ FB1_OUT1,FB2_OUT3= ＞ FB1_OUT3,ENO= ＞ B)

- 不规则次序的语句

Instance_FB2(EN:=A,FB2_IN1:=FB1_IN1,FB2_OUT1= ＞ FB1_OUT1,FB2_IN2:=FB1_IN2,FB2_OUT2= ＞ FB1_OUT2,FB2_IN3:=FB1_IN3,FB2_OUT3= ＞ FB1_OUT3,ENO= ＞ B)

例 2：规格方法 B （仅规定 FB1 变量）

Instance_FB2(FB1_IN1,FB1_IN2,FB1_IN3,FB1_OUT1,FB1_OUT2,FB1_OUT3)

Instance_FB2(FB1_IN1,FB1_IN2,FB1_IN3,FB1_OUT1)

- 必须定序列出笔变量和返回值。

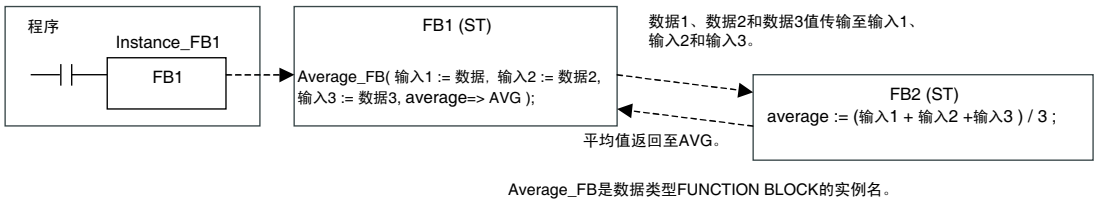
输入变量 1, 输入变量 2, ..., 输出变量 1, 输出变量 2, ...

- 输入变量的自变量必须以列表开始或如果列出 EN 变量时列在 EN 变量后面。
- 如果实际正在使用的数据和输出变量不在输出变量列表中间，则输出变量可以省略。

例如：Instance_FB2(FB1_IN1,FB1_IN2,FB1_IN3,FB1_OUT1,B1_OUT3)
在本例中，列表末端处的 FB1_OUT3 返回 FB2_OUT2 值。

- EN 和 ENO 数据不能作为自变量或返回值输入。
- 规格方法 A 不能与规格方法 B 一起用于功能块调用语句。

例 3：平均功能块
在下例中，功能块 1 调用计算平均值的功能块 2。



功能块 1

- 变量表

变量类型	变量名	数据类型	传输至 FB2/ 从 FB2 中上传
输入变量	EN	BOOL	---
输入变量	data1	INT	传输至输入 1
输入变量	data2	INT	传输至输入 2
输入变量	data3	INT	传输至输入 3
输入变量	bCheck	BOOL	---
输出变量	ENO	BOOL	---
输出变量	AVG	INT	从平均值中接收
内部变量	Average_FB	FUNCTION BLOCK 被调用功能块 定义：功能块 2	---

- ST- 语言算法

如果 b 检查正确，则调用功能块 2 来计算平均值。 3 个数值 -- 数据 1、数据 2 和数据 3 分别传输至功能块 2 的输入变量—输入 1、输入 2 和输入 3。计算结果 （平均）返回至 AVG。

注 下表所示的是采用规格方法 A 调用的 Average_FB 功能块 （列出的功能块变量）。

```
IF bCheck = TRUE THEN
Average(input1:=data1,input2:=data2,input3:=data3,average= > AVG);
ELSE
RETURN;
END_IF;
```

功能块 2

• 变量表

变量类型	变量名	数据类型	传输至 FB2/ 从 FB2 中上传
输入变量	EN	BOOL	---
输入变量	input1	INT	从数据 1 接收
输入变量	input2	INT	从数据 2 接收
输入变量	input3	INT	从数据 3 接收
输出变量	ENO	BOOL	---
输出变量	average	INT	传输至 AVG

tST- 语言算法

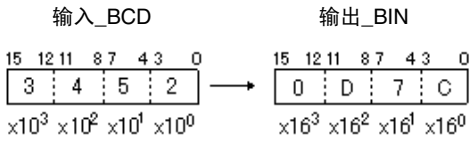
计算输入 1、输入 2 和输入 3 的平均值并将结果保存于 average 中。

```
average:=(input1+input2+input3)/3;
```

结构化文本编程范例

例 1：BCD 数据 （#0000-#9999）转换至 BIN 数据

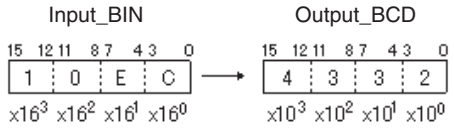
```
（* 检查输入参数“ Input_BCD ”（BCD 数据） *）
IF (Input_BCD ＞ =0 & Input_BCD ＜ =16#9999) THEN
ENO:=true;
ELSE
ENO:=false;
RETURN;
END_IF;
（*BCD 数据 4 次除以 16 以获得 BIN 数据每个数字 -- 从 BCD 数据中转换而来 *）
DIV_1:=Input_BCD/16;
DIV_2:=DIV_1/16;
DIV_3:=DIV_2/16;
DIV_4:=DIV_3/16;
（* 计算 BIN 数据每个数字 -- 从 BCD 数据中转换而来 *）
BIN_1:=Input_BCD-16*DIV_1;  (*a number of 16^0  数 *)
BIN_2:=DIV_1-16*DIV_2;  (*a number of 16^1数 *)
BIN_3:=DIV_2-16*DIV_3;  (*a number of 16^2数 *)
BIN_4:=DIV_3-16*DIV_4;  (*a number of 16^3 数 *)
（* 计算 BIN 数据“ Output_BIN ”（输出参数） *）
Output_BIN:=BIN_1+BIN_2*10+BIN_3*10*10+BIN_4*10*10*10;
```



例 2：BIN 数据 （#0000-#FFFF）转换至 BCD 数据

```
（* 检查输入参数 “ Input_BIN ”(BIN 数据 ) *）
IF (Input_BIN ＞ =0 & Input_BIN ＜ =16#FFFF) THEN
   ENO:=true;
ELSE
   ENO:=false;
   RETURN;
END_IF;
(*BIN 数据 4 次除以 10 以获得 BCD 数据每个数字 -- 从 BIN 数据中转换而来 *)
DIV_1:=Input_BIN/10;
DIV_2:=DIV_1/10;
DIV_3:=DIV_2/10;
DIV_4:=DIV_3/10;
```

```
(* 计算 BCD 数据每个数字 -- 从 BIN 数据中转换而来 *)
BCD_1:=Input_BIN-10*DIV_1;  (*a number of 10^0 数 *)
BCD_2:=DIV_1-10*DIV_2;  (*a number of 10^1 数 *)
BCD_3:=DIV_2-10*DIV_3;  (*a number of 10^2 数 *)
BCD_4:=DIV_3-10*DIV_4;  (*a number of 10^3 数 *)
(* 计算 BCD 数据 “ Output_BCD ”( 输出参数 ) *)
Output_BCD:=BCD_1+BCD_2*16+BCD_3*16*16+BCD_4*16*16*16;
```



限制

嵌套

用于 IF、 CASE、 FOR、 WHILE 或 REPEAT 语句的嵌套数没有限制。

数据类型限制

- 整数只能分配于数据类型 WORD、DWORD、INT、DINT、UINT、UDINT 或 ULINT 的变量。例如，如果 A 是 INT 数据类型，则 A:=1 ；如果数值不是整数数据类型，则出现语法错误。例如，如果 A 是 INT 数据类型，则 A:=2.5; 时会出现语法错误。
- 如果实数（浮点十进制数据）只可分配于数据类型 REAL 和 UREAL 的变量。例如，A 是 REAL 数据类型，则 A:=1.5;。如果数值不是实际数据类型，则会出现语法错误。例如，如果 A 是 REAL 数据类型，则 A:=2; 时会出现语法错误。使用 A:=2.0;。
- 位 （TRUE 和 FALSE）只可分配于 BOOL 数据类型的变量。例如，如果 A 为 BOOL 数据类型，则 A:=FALSE;。如果不采用 BOOL 数据类型，则会出现语法错误。。例如，如果 A 为 INT 数据类型，则 A:=FALSE; 时会出现语法错误。
- 数据类型在结构化文本中一致。例如，如果 A、B 和 C 为 INT 数据类型，则 A:=B+C;。但如果 A 和 B 是 INT 数据类型而 C 是 REAL 数据类型或 LINT 数据类型，则 A:=B+C; 时会出现语法错误。

结构化文本错误

错误信息

错误信息	错误原因	范例
数值不能赋值于 %s' 输入变量	数值已替代了变量。	
%s' 数据类型不支持 %s' 运算符	采用运算符不支持的数值或数据类型变量。	A:=B+1; （*A 和 B 为 WORD 类型变量 *）
%s' 变量已有只读存储器 AT 地址而且赋值。	数值已替代了分配于只读存储器地址的变量 （只读辅助区域地址或条件标记）	
数组变址超出范围	数组变址大于数组大小。	数组 [100]:=10; （*Array 是一个数组大小 100 的数组变量 *）

错误信息	错误原因	范例
不能从 %s 转换到 %s。	运算结果的数据类型数字方程式与替代目的地处的变量不匹配。	替代了与数据类型不一致的变量Y:=ABS(X); （*X 是 INT 类型变量； Y 是 UINT 类型变量 *）
除 0	数字表达式中有除数除以 0。	
未找到注释结束处	注释中对应于打开的括号和星号 “ (* ” 的另一括号和星号 “ *) ” 未封闭。	(* 注释
文字格式 '%s' 无效	数字格式不合法。	X:=123_; （* 下划线后无数字 *） X:=1__23; （* 下划线后立即跟另一根下划线 *） X:=2#301; Y:=8#90; （* 使用了不能用于二进制或八进制值的数字 *） 注 下划线插在数字之间以便读取。将 2#、 8# 和 16# 置于数字起始处以分别表示数字是二进制、八进制和十六进制值。
无效文字值	数字值不合法	X:=1e2; （* 变址已用于不是 REAL 数据类型的数值 *） 注 “ e ” 表示指数是 10。
无效数组变址	有非整数类型运算或非整数变量的数字方程式已指定于数组变址中。	Array[Index]:=10; （*Index 是 WORD 类型变量 *）
无效常数	有非整数类型运算或非整数变量的数字方程式已指定于 CASE 语句的整数方程式。	CASE A OF （*A 是 REAL 类型变量 *） 1: X:=1; 2: X:=2; END_CASE;
无效表达式	数字方程式不合法。例如，整数方程式或条件方程式不合法或已指定于语法 （IF、 WHILE、 REPEA、FOR、 CASE）。	WHILE DO （* WHILE 语句不包含条件方程式 *） X:=X+1; END_WHILE;
FOR 回路说明中有无效参数	数据类型不是 IF、 WHILE、 REPEA、FOR、 CASE 的变量已用于 FOR 语句中的变量。	FOR I:=1 TO 100 DO （*I 为 WORD 类型变量 *） X:=X+1; END_FOR;
无效语句	语句不合法。例如，语句 （IF、 WHILE、 REPEA、FOR、 CASE）在语法中不包含 IF、 WHILE、 REPEA、FOR、 CASE。	X:=X+1; （* 语法中无 REPEAT*） UNTIL X ＞ 10 END_REPEAT;

错误信息	错误原因	范例
功能输出无效变量	功能输出的指定变量不合法（非布尔 （BOOL）变量或数字已规定为 ENO 转送目的地）	Y:=SIN(X1, ENO= ＞ 1);
无 （	数据格式转换指令或功能调用不包含“ (“（开口括号）。	Y:=INT_TO_DINT X);
无）	运算符括号或数据格式转换指令或功能调用不包含“) “（封闭括号）与“ (“（开口括号）对应。	Y:=(X1+X2/2
无：	CASE 语句的整数方程式中无“ : ”（冒号）。	CASE A OF 1 X:=1; END_CASE;
无 =	“ = ”不在赋值方程式中。	
无；	语句中无“ ; ”（分号）。	
无 DO	FOR 或 WHILE 语句中无“ DO ”。	
无 END_CASE	CASE 语句结束处无“ END_CASE ”。	
无 END_FOR	FOR 语句结束处无“ END_FOR ”。	
无 END_IF	IF 语句结束处无“ END_IF ”。	
无 END_REPEAT	REPEAT 语句结束处无“ END_REPEAT ”。	
无 END_WHILE	WHILE 语句结束处无“ END_WHILE ”。	
无输入参数。必须设置所有输入变量。	未规定功能自变量或功能自变量不足。	Y:=EXPT(X);
无 OF	CASE 语句中无“ OF ”。	
无 THEN	IF 语句中无“ THEN ”。	
无 TO	FOR 语句中无“ TO ”。	
无 UNTIL	REPEAT 语句中无“ UNTIL ”。	
无 [	未指定数组变量的数组变址。	X:=Array ; （*Array 是数组变量 *）
无]	未指定数组变量的数组变址。	X:=Array [2; （*Array 是数组变量 *）

错误信息	错误原因	范例
无常数	CASE 语句的整数方程式不含常数。	CASE A OF 2..: X:=1; 2,: X:=2; END_CASE;
文字数中不支持 NOT 运算	NOT 运算符用于数值。	Result:=NOT 1;
%s 数据类型不支持负值	在数据类型不支持负值（UINT、UDINT、ULINT）的变量之前已使用了减符号。	Y:=-X; (*X 是一个 UINT 型变量， Y 是一个 INT 型变量 *)
必须有一行有效码 （不包括注释）	无一行有效码 （不包括注释）	
有太多变量指定于函数中	设置了太多函数参数。	Y:=SIN(X1,X2);
未定义的识别符 '%s'	使用了未在变量表中定义的变量。	
意外语法 '%s'	已非法使用关键字 （保留字）或变量。	FOR I:=1 TO 100 DO BY -1 (*DO 位置非法 *) X:=X+1; END_FOR;
用法与函数变量不匹配	函数参数非法使用。	Y:=SIN(X1,EN= ＞ BOOL1); (* 输入参数 EN 已用作输出参数 *)
变量超出范围	在变量中，已用超出变量数据类型范围的值替代。	X:=32768; (*X 是 INT 类型变量 *)
变量 '%s' 不是函数参数 '%s'	在参数中已规定了不能指定于函数参数中的变量。	(*X 和 Y 是 REAL 类型变量而 Z 不是 SIN 函数参数 *)

警告信息

警告信息	警告原因	范例
关键字 '%s' 冗余	关键字已用于无效地方。例如，在回路语法范围外使用 EXIT 语句。	
从 '%s' 转换到 '%s' 期间，可能数据丢失	由于在大数据量的数据类型转换到小数据量的数据类型期间，数据可能被丢失。	Y:=DINT_TO_INT(X); （*X 是一个 DINT 类型变量； Y 是一个 INT 类型变量 *）

经常会被提及的问题

问题：如何表示十六进制值 ?

回答：在数值之前添加 “ 16# ”。例如， 16#123F。

也可添加前缀 8# 和 2# 分别用来表示八进制和二进制数。无这些前缀的数作为十进制数。

问题：FOR 语句可以使用多少次 ?

回答：在下例中， FOR 语句内容执行 101 次。当 “ i ” 值等于 101 时，回路处理结束。

```
FOR i:=0 TO 100 BY 1 DO
  a:=a+1;
END_FOR;
```

问题：当超出数组下标时回发生什么情况？

回答：如果数组变量 INT[10] 有 10 个元素，则以下类型的语句不能检测到错误。

当执行语句时，运算不稳定。

```
i:=15;
INT[i]:=10;
```

问题：结构化文本编辑器中的变量是否会自动注册于变量表？

回答：不会。在使用变量之前，将变量注册于变量表中。

问题：是否可以直接调用梯级编程指令？

回答：不可以。

附录 C
外部变量

分类	名称	CX-Programmer 中的外部变量	数据类型	地址
条件标记	大于或等于 （GE）标记	P_GE	BOOL	CF00
	不等于 （NE）标记	P_NE	BOOL	CF001
	小于或等于 （LE）标记	P_LE	BOOL	CF002
	指令执行错误 （ER）标记	P_ER	BOOL	CF003
	进位 （CY）标记	P_CY	BOOL	CF004
	大于 （GT）标记	P_GT	BOOL	CF005
	等于 （EQ）标记	P_EQ	BOOL	CF006
	小于 （LT）标记	P_LT	BOOL	CF007
	负数 （N）标记	P_N	BOOL	CF008
	溢出 （OF）标记	P_OF	BOOL	CF009
	下溢 （UF）标记	P_UF	BOOL	CF010
	存取错误标记	P_AER	BOOL	CF011
	始终 OFF 标记	P_Off	BOOL	CF114
	始终 ON 标记	P_On	BOOL	CF113
时钟脉冲	0.02 秒时钟脉冲位	P_0_02s	BOOL	CF103
	0.1 秒时钟脉冲位	P_0_1s	BOOL	CF100
	0.2 秒时钟脉冲位	P_0_2s	BOOL	CF101
	1 分钟时钟脉冲位	P_1mim	BOOL	CF104
	1.0 秒时钟脉冲位	P_1s	BOOL	CF102
辅助区域标记 / 位	第一个循环标记	P_First_Cycle	BOOL	A200.11
	步骤标记	P_Step	BOOL	A200.12
	第一个任务执行标记	P_First_Cycle_Task	BOOL	A200.15
	最大循环标记	P_Max_Cycle_Time	UDINT	A262
	当前扫描时间	P_Cycle_Time_Value	UDINT	A264
	循环时间错误标记	P_Cycle_Time_Error	BOOL	A401.08
	电池电量低标记	P_Low_Battery	BOOL	A402.04
	I/O 验证错误标记	P_IO_Verify_Error	BOOL	A402.09
	输出 OFF 位	P_Output_Off_Bit	BOOL	A500.15
OMRON FB 库字（见注释）	CIO 区域规格	P_CIO	WORD	A450
	HR 区域规格	P_HR	WORD	A452
	WR 区域规格	P_WR	WORD	A451
	DM 区域规格	P_DM	WORD	A460
	EM0 到 C 区域规格	P_EM0 to P_EMC	WORD	A461 to A473

注　　这些字是 OMRON FB 库的外部变量。不得用这些字创建功能块。