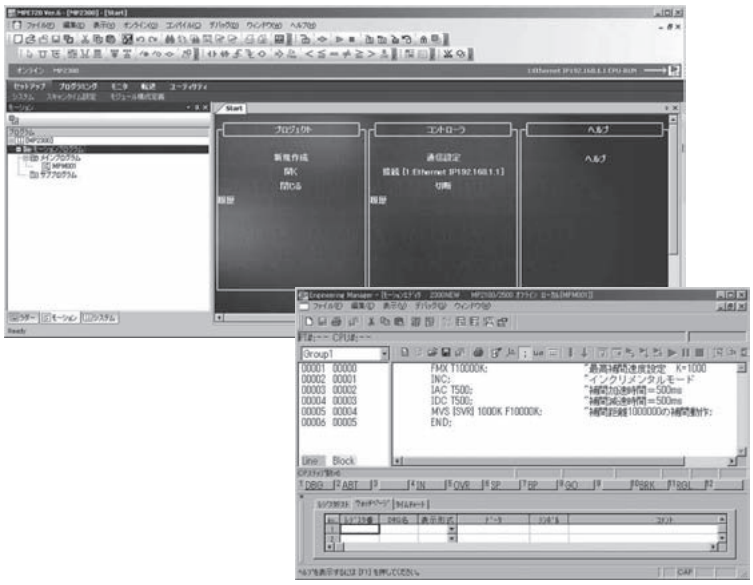
YASKAWA

机器控制器 MP2000系列

用户手册
运动程序篇



资料编号 SICP C880700 38A

使用手册概要

■ 本使用手册汇总了机器控制器 MP2000 系列的运动语言命令相关的信息，并主要根据以下项目进行说明。

- 运动程序的概要
- 规格
- 程序的开发流程
- 运动程序及顺序程序
- 变量
- 编程
- 相关命令
- 软件工具 （MPE720）
- 故障检修等

■ 为了能正确使用 MP2000 系列，请您仔细阅读本手册。同时，请妥善保管好本手册，以便需要时能够随时查阅。

本手册的使用方法

■ 本手册的读者对象

本手册主要面向以下两种读者。

- MP2000 系列的系统设计人员
- MP2000 系列的运动程序及顺序程序的编程人员

■ 本使用手册使用的软件工具

本手册使用 MPE720 Ver.6 的画面进行说明。

如果您使用的是 MPE720 Ver.5，阅读时请结合 MPE720 Ver.5 的实际画面理解其中的内容。

■ 缩略语

本手册中使用了以下缩略语。

- MP2000：MP2100、MP2100M、MP2200、MP2300、MP2300S、MP2310、MP2400 以及 MP2500/MP2500M/MP2500D/MP2500MD 的总称

本手册的资料体系

■ MP2000 系列包括 MP2100、MP2100M、MP2200、MP2300、MP2300S、MP2310、MP2400 以及 MP2500/MP2500M/MP2500D/MP2500MD。

本手册根据上述产品系列对相关资料进行了汇总。相关手册的内容，请参阅下页的列表。

相关资料

■ 请根据使用目的，阅读下表所示的相关资料。

■ 请在充分了解产品的规格、使用限制条件等内容后使用本产品。

资料名称	资料编号	内容
Machine Controller MP210□/MP210□M USER'S MANUAL Design and Maintenance （英文版）	SIEP C880700 01	对机器控制器 MP210□/MP210□M 的功能、使用方法以及安装步骤等内容进行详细说明。
机器控制器 MP2200 用户手册	SICP C880700 14	对机器控制器 MP2200 的功能、使用方法以及安装步骤等内容进行详细说明。
机器控制器 MP2300 基本模块 用户手册	SICP C880700 03	对机器控制器 MP2300 的功能、使用方法以及安装步骤等内容进行详细说明。
机器控制器 MP2300S 基本模块 用户手册	SICP C880732 00	对机器控制器 MP2300S 的功能、使用方法以及安装步骤等内容进行详细说明。
Machine Controller MP2310 Basic Module USER'S MANUAL （英文版）	SIEP C880732 01	对机器控制器 MP2310 的功能、使用方法以及安装步骤等内容进行详细说明。
机器控制器 MP2400 用户手册	SICP C880742 00	对机器控制器 MP2400 的功能、使用方法以及安装步骤等内容进行详细说明。
Machine Controller MP2500/MP2500M/ MP2500D/MP2500MD USER'S MANUAL （英文版）	SIEP C880752 00	对机器控制器 MP2500/MP2500M/MP2500D/MP2500MD 的使用方法进行详细说明。
Machine Controller MP2000 Series Built-in SVB/SVB-01 Motion Module USER'S MANUAL （英文版）	SIEP C880700 33	对 MP2000 系列机器控制器的运动模块 （内置 SVB、SVB-01）功能、规格以及使用方法进行详细说明。
Machine Controller MP2000 Series SVA-01 Motion Module USER'S MANUAL （英文版）	SIEP C880700 32	对用于 MP2000 系列机器控制器的运动模块 SVA-01 的相关内容进行详细说明。
Machine Controller MP2000 Series Pulse Output Motion Module PO-01 USER'S MANUAL （英文版）	SIEP C880700 28	对可与 MP2000 系列机器控制器连接的脉冲输出运动模块 PO-01 的相关内容进行详细说明。
机器控制器 MP2300 系列 通信模块 用户手册	SICP C880700 04	对可与 MP2300 系列机器控制器连接的通信模块的相关内容进行详细说明。
Engineering Tool for MP2000 Series Machine Controller MPE720 Ver.6 USER'S MANUAL （英文版）	SIEP C880700 30	对支持 MP2000 系列机器控制器的软件工具 MPE720 Ver.6 的安装以及操作方法进行详细说明。
Machine Controller MP900/MP2000 Series MPE720 Software for Programming Device USER'S MANUAL （英文版）	SIEP C880700 05	对支持 MP900/MP2000 系列机器控制器的软件工具 MPE720 的安装以及操作方法进行详细说明。
机器控制器 MP2000 系列 用户手册 梯形图程序篇	SICP C880712 02	对 MP2000 系列机器控制器梯形图程序中使用的相关程序语言进行详细说明。
Machine Controller MP900/MP2000 Series New Ladder Editior PROGRAMMING MANUAL （英文版）	SIEZ-C887-13.1	对 MP 系列机器控制器新型梯形图编辑软件的程序命令进行详细说明。
Machine Controller MP900/MP2000 Series New Ladder Editior USER'S MANUAL （英文版）	SIEZ-C887-13.2	对 MP900/MP2000 系列机器控制器设计、维护的新型梯形图编辑软件的操作方法进行详细说明。

图标的说明

为了让说明内容的分类更加清晰，在必要时采用了以下图标。

- 表示说明中特别重要的事项。
  也表示发生警报等但还不至于造成装置损坏的轻度注意事项。

- 表示程序、操作等示例。

- 表示补充事项以及了解后可为工作提供便利的功能。

- 对比较难懂的术语及出现的新术语进行说明。

安全标识的说明

本手册根据与安全有关的内容，使用了下列标识。
有关安全标识的说明，均为重要内容，请务必遵守。

表示错误使用时，将会引发危险情况，有可能导致人身伤亡。

表示错误使用时，将会引发发危险情况，导致轻度或中度人身伤害，损坏设备。

另外，即使是 

中的说明事项，根据具体情况，有时也可能导致重大事故。

表示禁止（绝对不能做）。例如严禁烟火时，表示为  。

表示强制（必须做）。例如接地时，表示为  。

安全注意事项

本节将对正确使用机器时的安全注意事项进行说明。在安装、运行、保养 / 检查之前，请务必仔细阅读本手册及附带资料，以便正确使用。请在充分了解机器的相关知识安全信息和注意事项后使用。

■ 安全注意事项

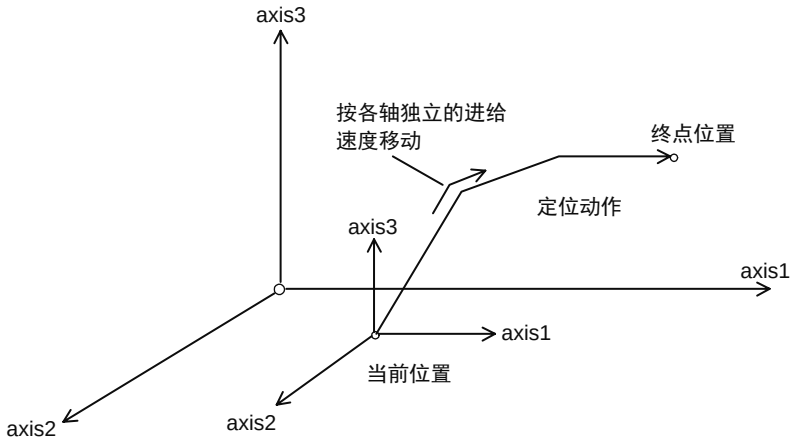
⚠注意

- 在进行下列轴移动命令编程时，请务必进行轨迹检查，以免设备和工件之间的碰撞。

需要检查的轴移动命令

- “ 定位 （MOV）” 命令
- “ 直线插补 （MVS）” 命令
- “ 圆弧插补 (MCC,MCW)” 命令
- “ 螺旋插补 (MCC,MCW)” 命令
- “ 时间指定定位 （MVT）” 命令
- “ 带跳过功能的直线插补 （SKP）” 命令
- “ 原点复归 （ZRN）” 命令
- “ 外部定位 （EXM）” 命令

例



“ 定位 (MOV)” 命令的基本轨迹示例

如果疏于上述检查，则有可能造成设备损坏，甚至有可能导致人身伤害事故。

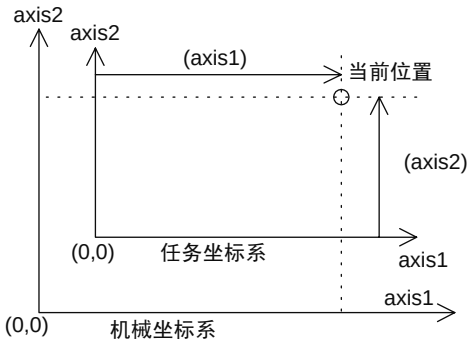
⚠注意

· 如果误发出下列坐标命令，则之后的移动动作会完全不同。在运行前，请务必检查是否正确指定了上述命令。

需要检查的坐标命令

- “ 绝对值模式 (ABS)” 命令
- “ 增量模式 （INC）” 命令
- “ 更改当前值 （POS）” 命令
- “ 机械坐标指令 (MVM)” 命令

例



“ 更改当前值 （POS）” 命令创建的新任务坐标系的示例

如果疏于上述检查，则有可能造成设备损坏，甚至有可能导致人身伤害事故。

■ 一般注意事项

请在使用时予以注意。

- MP2000 系列不是为了用于系统或者在性命攸关的状况下所使用的器械而设计制造的。
  需要将本手册所记载的产品使用于载人移动体、医疗、航空航天、海底中转通信用器械或者系统等特殊用途时，请向本公司代理店或销售负责人垂询。
- MP2000 系列是在严格的质量管理下生产的，但是用于因本产品故障会造成重大事故或损失的设备时，请配置安全装置。
- 本手册中使用的插图仅为代表性图例，可能会与交付的产品有所不同。
- 因产品改良、规格变更或为了提高手册的使用便利性，本手册可能会随时变更。
- 变更后，本手册的资料编号将进行更新，并作为改订版发行。关于改订版的版数请参见封底的记载。
- 因破损或遗失而需索取本手册时，请于本公司代理店或封底记载的最近的分公司联系。联系时请告知资料编号。
- 安装在产品上的铭牌出现磨损或破损时，请向公司代理店或手册封底记载的最近的销售网点订购。

关于保证

(1) 保证内容

■ 保证期限

您购买的产品 （以下简称交付产品）的保修期为向指定地点交货后 1 年或自本公司工厂出厂后 18 个月，以先到者为准。

■ 保证范围

在上述保修期内，因本公司的责任而引起故障时，本公司将提供替代品或提供免费修理。
因交付产品使用寿命导致的故障、消耗件及过期零件的更换不在保修范围内。

另外，故障原因属于以下情形时，本公司将不予保证。

1. 在产品样本或说明书、另行签署的规格书规定外的、不适当条件或环境下安装、使用时引发的故障
2. 因交付产品以外的原因而引发的故障
3. 因本公司以外的改造或修理而引发的故障
4. 因将产品用于原本用途以外时引发的故障
5. 因产品出厂时的科学、技术水平所无法预见的原因而引发的故障
6. 其他天灾人祸等非本公司责任的原因导致的故障

(2) 免责事项

1. 因交付产品的故障而造成的损失及给客户带来的不便，本公司将不承担任何责任。
2. 对于本公司可编程的产品，由本公司以外的人员进行的编程 （含各种参数设定）以及因此造成的后果，本公司概不负责。
3. 产品产品目录或手册中记载的信息，旨在帮助客户购买符合用途的适当产品，并不保证或承诺不会因该信息的使用而侵犯本公司及第三方的知识产权及其他权利。
4. 因使用产品产品目录或手册中记载的信息而对第三方的知识产权及其他权利造成侵害时，本公司概不负责。

(3) 确认正确的用途及使用条件

1. 将本公司的产品与其他产品配套使用时，请客户自行确认适用的标准、应遵守的法规或条例。
2. 请客户自行确认本公司产品针对客户使用的系统、机械、装置的适用性。
3. 将产品用于以下用途时，请在与本公司协商的基础上决定使用与否。如果决定使用，请选择额定值及性能有盈余的产品，并采取安全措施，以便将故障时的危险降低到最小限度。
   - 在室外使用、在有潜在化学污染或电气干扰的条件下使用、或在产品目录或手册中规定外的条件及环境下使用
   - 原子能控制设备、焚烧设备、铁路 / 航空 / 车辆设备、医用器械、娱乐设备及遵守行政机关、个别行业条例的设备
   - 可能危及生命及财产安全的系统、机械、设备
   - 燃气、给排水、供电系统及 24 小时连续运行的系统等要求具有高度可靠性的系统
   - 其他需具备与上述各项要求相当的高度安全性的系统
4. 将本公司产品用于可能会会对人身或财产安全带来重大危险的用途时，请务必通过危险警告或冗余设计，事先确认以确保必要的安全性，以及是否已对本公司产品进行了正确的配电和设置。
5. 产品目录或手册中记载的回路事例及其它应用事例仅供参考。请确认所使用机器、装置的功能和安全性后再正确使用。
6. 请在彻底理解所有使用禁止事项及使用注意事项的基础上，正确使用本公司产品，以免给第三方带来意外损失。

(4) 规格的变更

因产品改良或其他原因，产品目录或手册中记载的产品名称、规格、外观、附件等若有变更，恕不另行通知。变更后，产品目录或手册的资料编号将进行更新，并作为修订版发行。您在考虑或订购所记载产品时，请事先向销售窗口确认。

目录

3章　程序的开发流程

4章　运动程序

5章　顺序程序

6章　变量（寄存器）

7章　编程

8章　相关命令

9 章　软件工具（MPE720）

10 章　故障诊断

附录

1章

概要

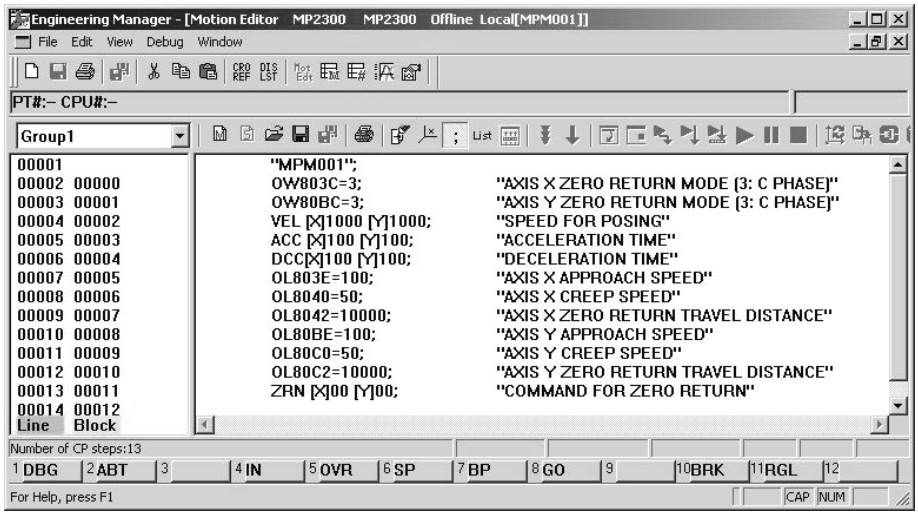
本章主要对首次使用运动程序的读者就程序的概要及特征进行说明。

1.1 运动程序的概要

运动程序是使用 （株）安川電機独有的文本格式语言 — 运动语言编写的程序。它可通过梯形图程序来编写 MSEE 命令，或登录到 M-EXECUTOR 模块（注）1 的登录画面来执行。

（注）1. M-EXECUTOR 模块是 MP2100、MP2100M、MP2300S、MP2310 以及 MP2400 可使用的模块。
在其它的 MP2000 系列机型中不能使用。

运动程序与梯形图程序不同，最多可创建 256 个该运动程序。
运动程序的实例如下图所示。

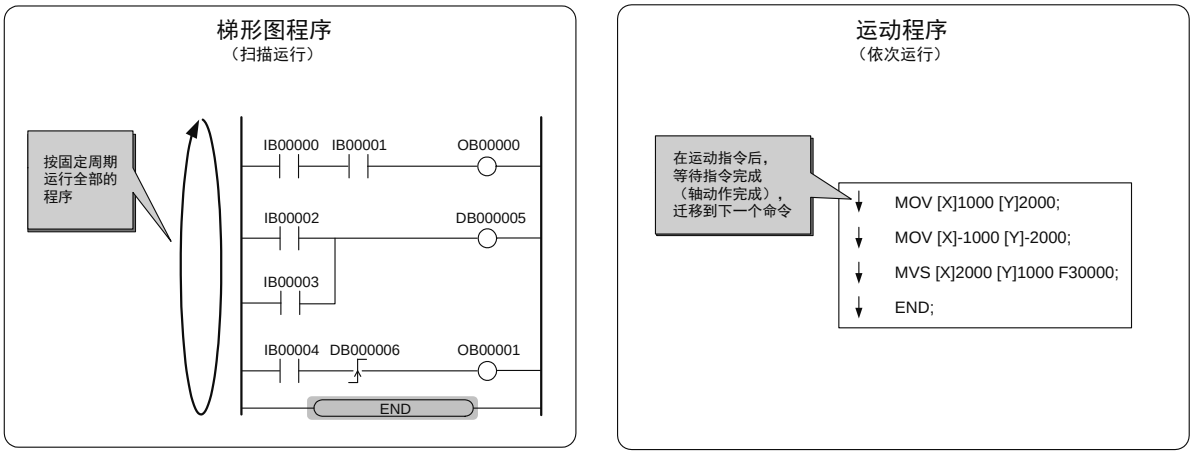


下面本手册开始对运动程序的特点进行说明。

1.2 运动程序的特点

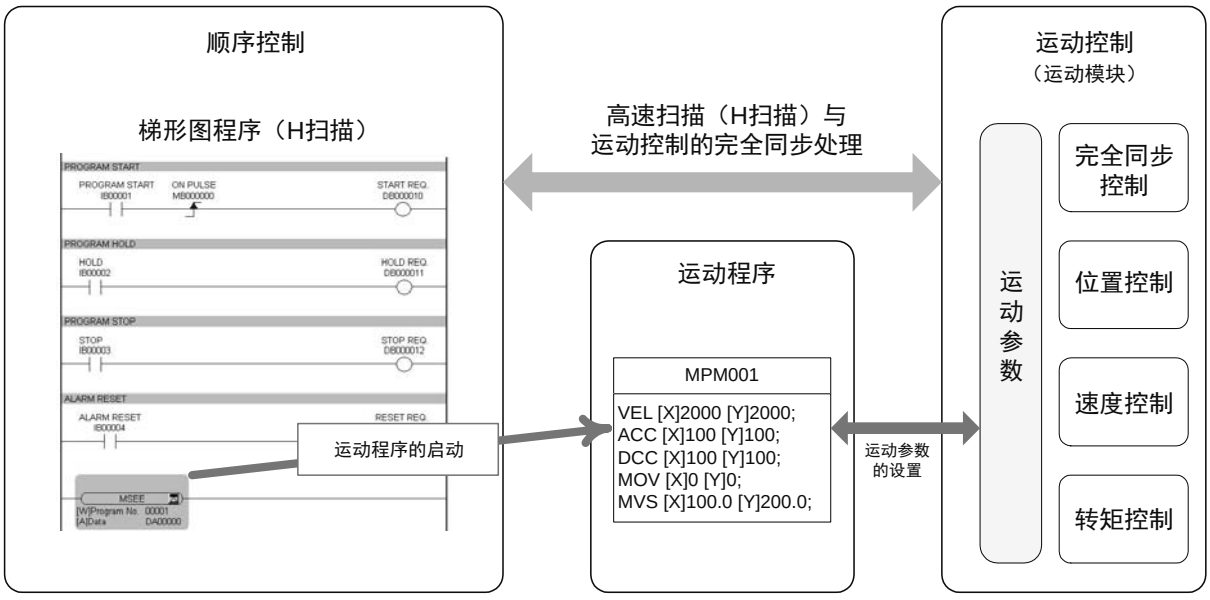
1.2.1 运动程序的运行方式

运动程序采用了不同于梯形图程序的运行方式。
梯形图程序从程序的起始到 END 命令会在 1 次扫描内完成处理。
而运动程序即使在一个命令中也会执行多次扫描处理。然后，在一个命令完成后会进入下一个命令，运行顺序处理。
本手册将上述运行方式分别表述为“ 扫描运行 ”、“ 依次运行 ”，并加以区别。



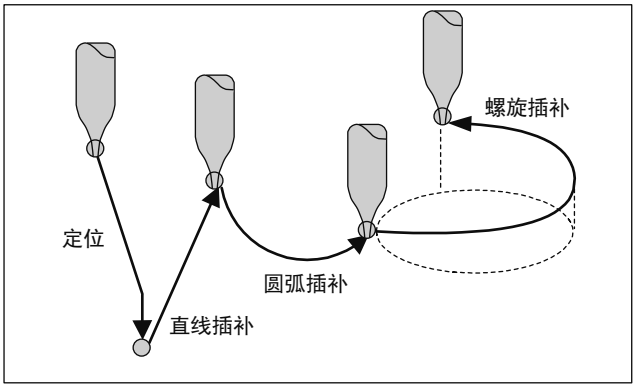
1.2.2 顺序控制与运动控制的完全同步处理

运动程序所写的指令和 MP2000 系列高速扫描 （H 扫描）做完全同步处理，运动程序启动没有延迟，可以在一个扫描周期内完成梯形图的起始信号到轴动作为止的系列指令。



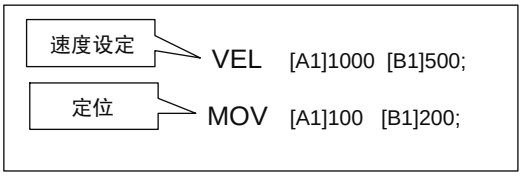
1. 2. 3 简单实现高级运动控制

除基本的运动控制外，即使是要求复杂动作的运动，只要使用运动程序，也能够简单实现。



1. 2. 4 采用简明易懂的命令符

如下面例子所示，运动程序采用了直观性强、简明易懂的命令符，将 “ 速度设定 ” 命令记述为 “VEL”，“ 定位 ” 命令记述为 “MOV” 等。



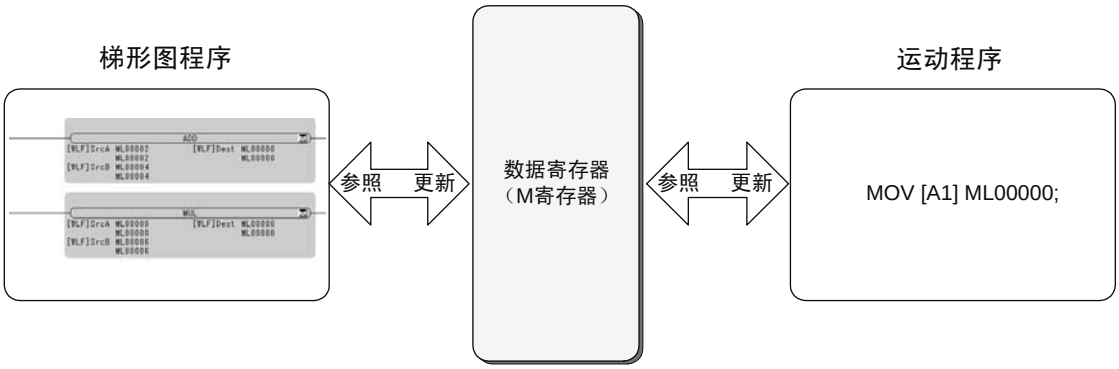
1. 2. 5 梯形图程序内部可进行数值运算

运动语言提供四则运算及逻辑运算的命令。
利用这些运算能够在运动程序内部计算目标位置等数值。

```
DL00000 = DL00002 + DW00004;
DL00000 = DW00002 * DL00004;
MW00000 = MW00000 & 00FFH;
MF00000 = SIN(30.0);
```

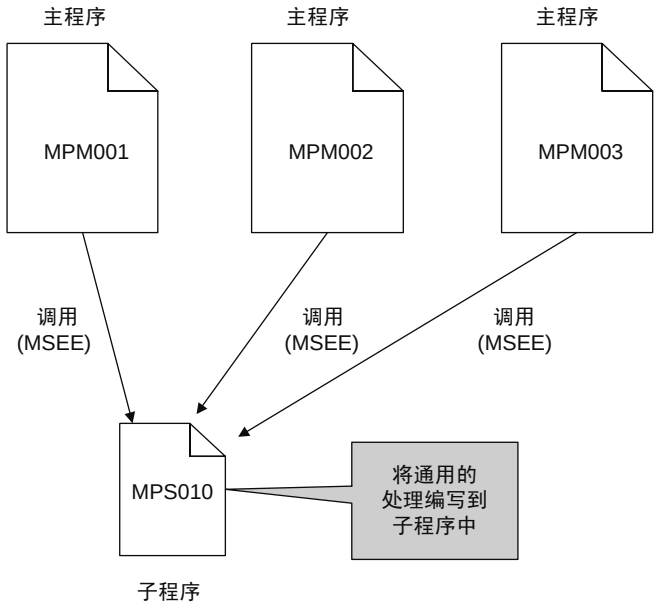
1.2.6 可从梯形图交换数据

梯形图程序与运动程序之间可通过数据寄存器 （M 寄存器）实现数据交换。
通过梯形图程序内部更新的数据能够在运动程序中使用。同样，通过运动程序更新的数据也可以在梯形图程序中使用。



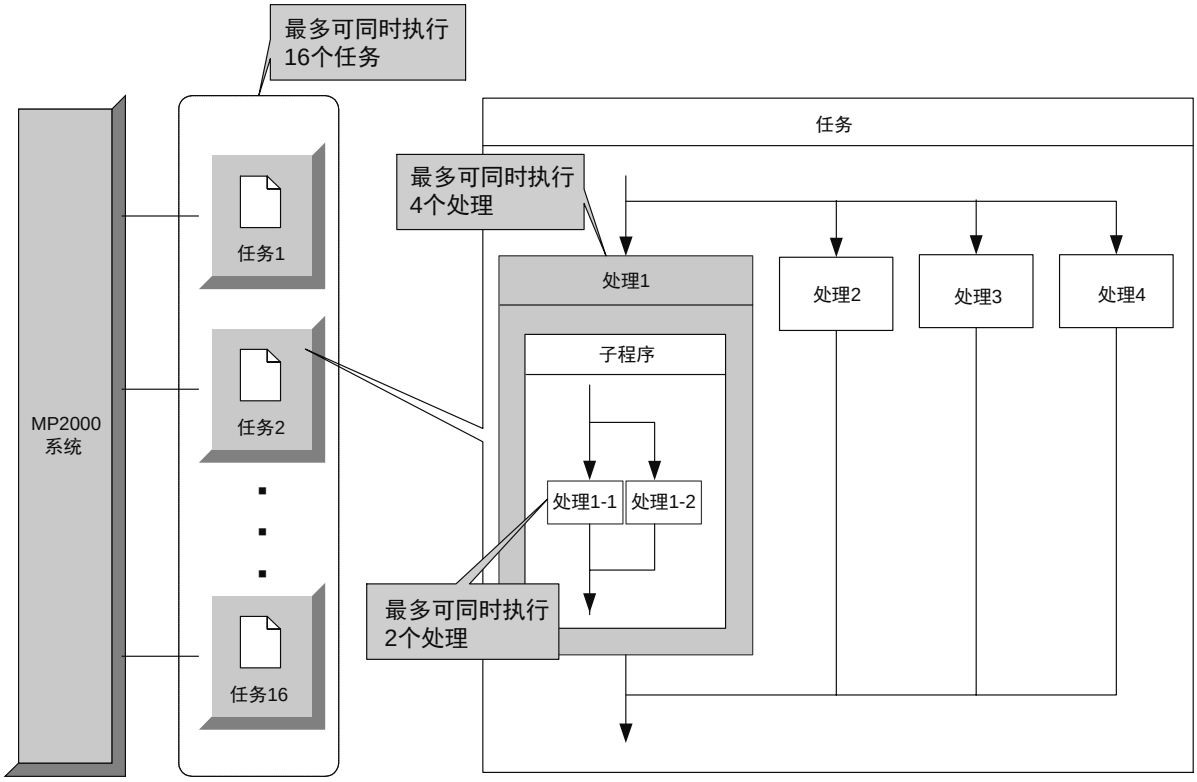
1.2.7 灵活运用子程序最大程度优化内存的利用率

可通过运动程序创建子程序。
通用动作子程序化 （通用化），能够将程序步数控制到最少，高效利用内存。



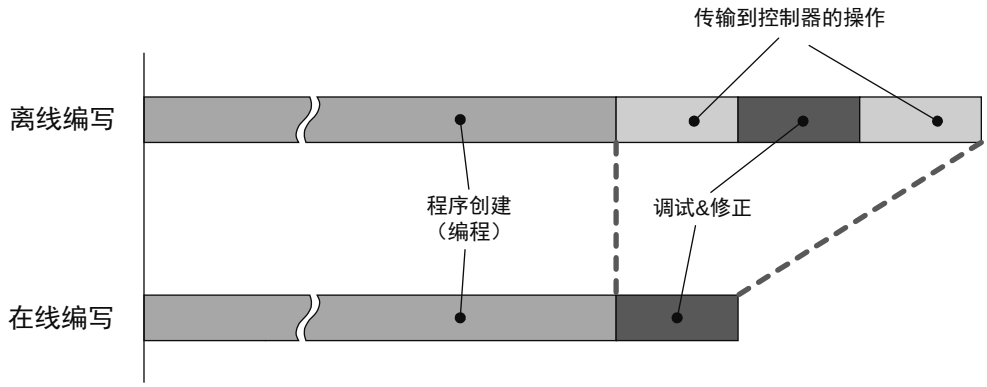
1.2.8 程序的同时执行

一台 MP2000 系列运行运动程序时，最多可同时执行 16 个任务。各运动程序通过主程序最多可同时执行 4 个处理。此外，如果通过主程序来调用子程序，则子程序最多可同时执行 2 个处理。使用该功能可同时控制多个不同的动作。



1.2.9 程序的在线编写

与梯形图程序相同，运动程序可进行在线编写。
在线编写是指在登录到控制器的状态下编写程序。在线编写过程中，仅在保存操作时才会执行向控制器的传输动作，因此，无需单独进行向控制器的传输操作，从而提高了程序开发效率。



补充　运动程序运行中，无法使用在线编写。

1.2.10 多种类型的简单程序功能 （MPE720 Ver.6.04 以上）

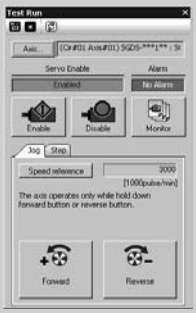
MP2000 系列的软件工具 “MPE720 Ver.6” 具备以下简单程序功能。

● 命令输入辅助功能

只需在画面中选择命令、设定数据，
就能够将命令插入到编辑器中。

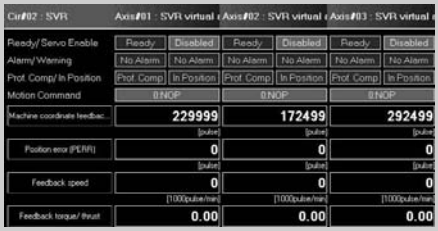
● 测试运行功能

可通过画面执行轴控制。



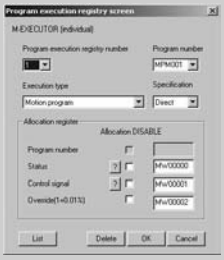
● 轴运行监视功能

各轴的运行情况一目了然。



● 程序运行登录功能

能够非常简单地将要运行程序登录到系统。

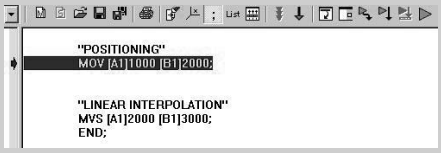


● 运行控制面板功能

运动程序可通过运动编辑器画面启动。

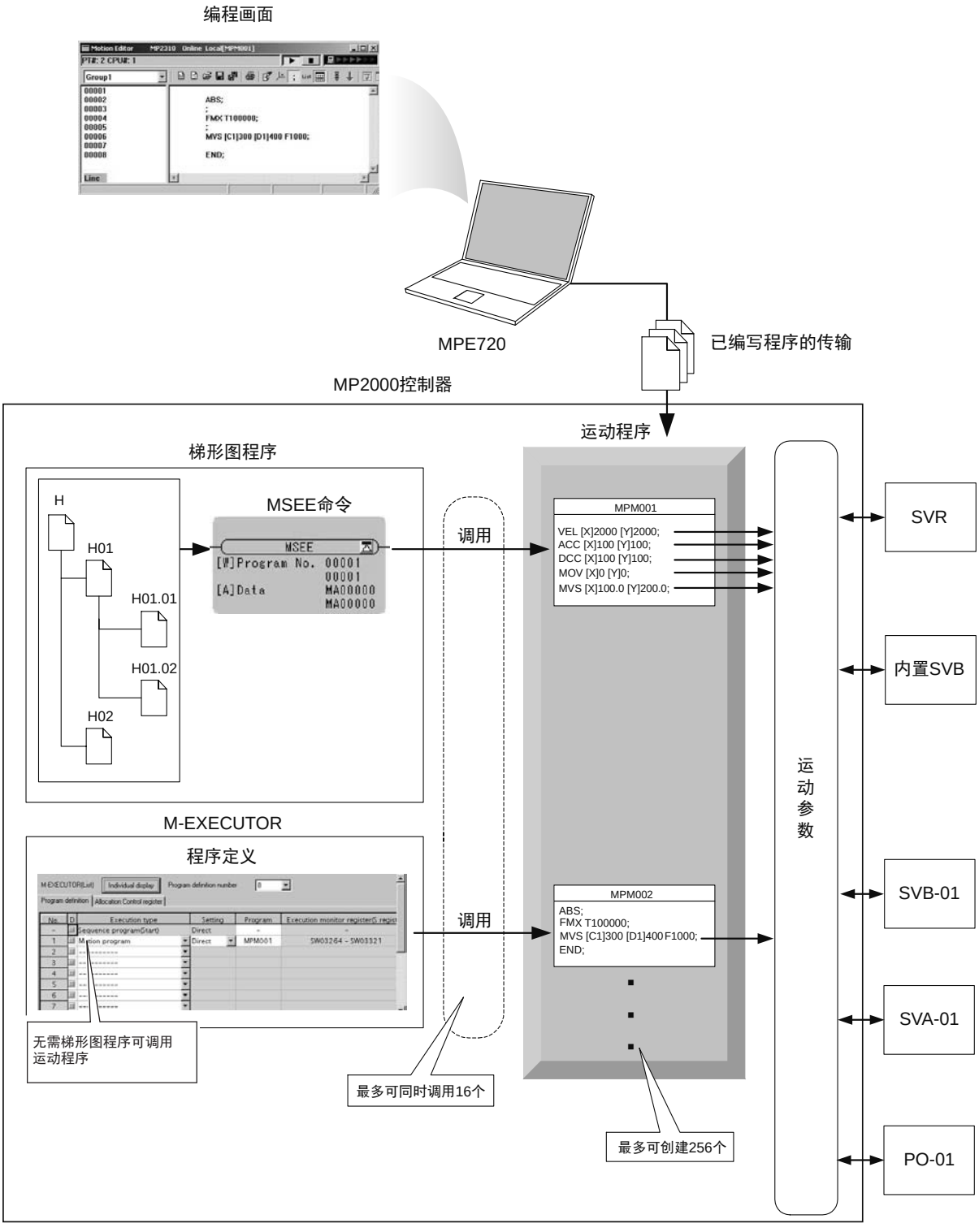
● 调试运行功能

是调试运动程序的功能。
可基于分步运行、断点设定等命令来调试运动程序。



1.3 运动程序的系统构成

运动程序通过 MPE720 的运动编辑器编写后，传输到 MP2000 控制器。MP2000 控制器可通过用梯形图程序编写的 MSEE 命令、或 M-EXECUTOR 模块的登录画面调用运动程序，用以编写运动命令。运动命令通过运动参数向运动模块发出指令，使得轴动作。
运动程序的系统构成见下图。

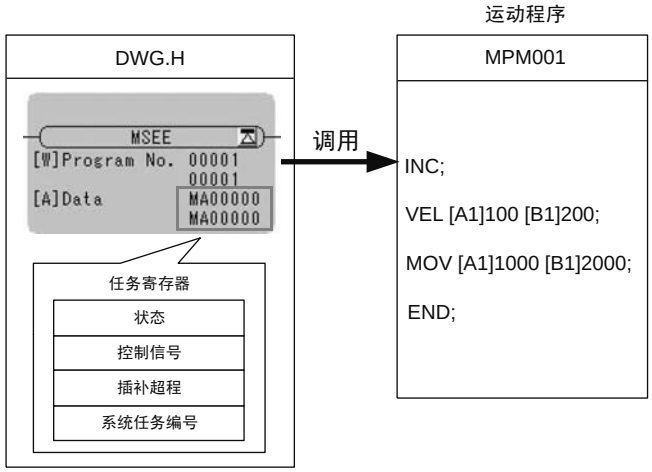


1.4 运动程序的登录

登录运动程序有以下 2 种方法。

■ 使用梯形图程序调用运动程序

通过 DWG.H 编写 MSEE 命令，调用要运行的运动程序。通过 MSEE 任务寄存器，执行调用的运动程序的启动、停止请求等操作。如果是 DWG.H，也可通过总图、子图、孙图中的一个图来调用运动程序。

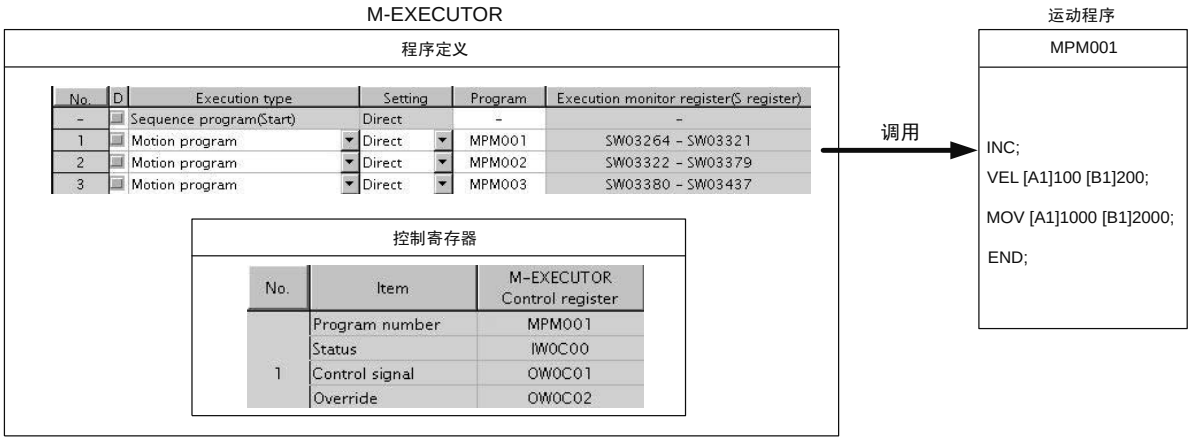


补充

本手册将梯形图程序的高速处理图表述为 DWG.H。

■ 将运动程序登录到 M-EXECUTOR

在 M-EXECUTOR 的程序定义中登录运动程序。同时，通过 M-EXECUTOR 控制寄存器 （I/O 寄存器），执行已登录运动程序的开始、停止请求等操作。

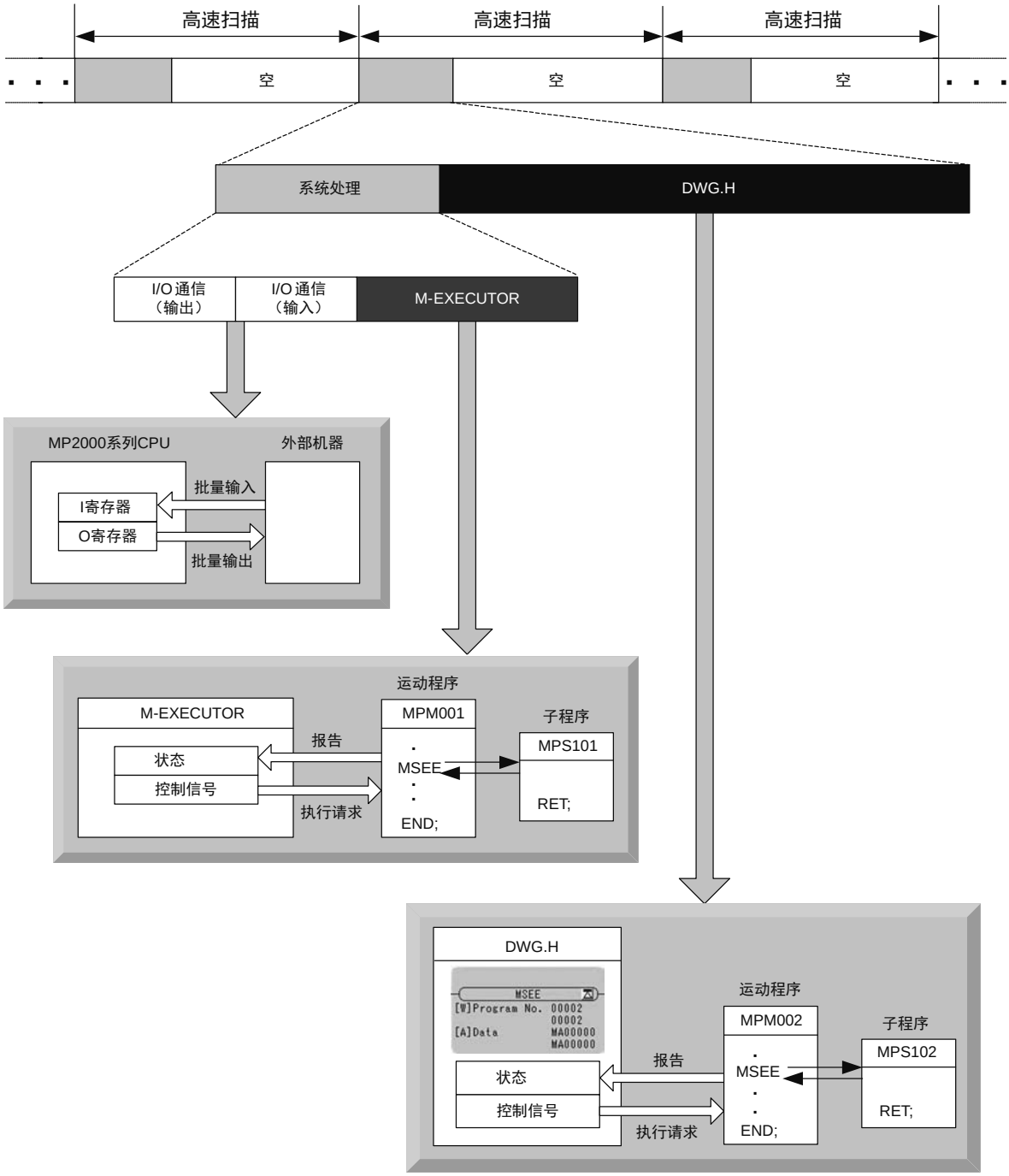


补充

M-EXECUTOR 是指运行运动程序及顺序程序的软件模块。

1.5 运动程序的运行时刻

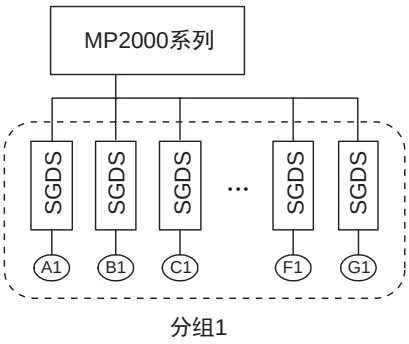
运动程序可实现与 MP2000 系列的高速扫描完全同步的处理。高速扫描首先会执行 I/O 通信，接着运行已登录到 M-EXECUTOR 的运动程序，然后利用 DWG.H 中编写 MSEE 命令的运行时刻运行运动程序。
运动程序的运行时刻如下图所示。



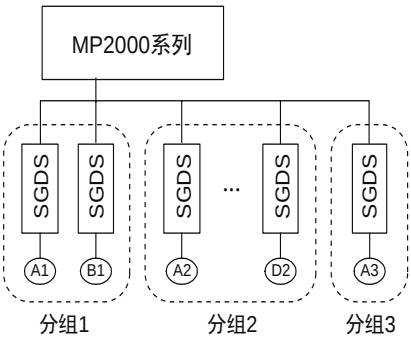
1.6 分组

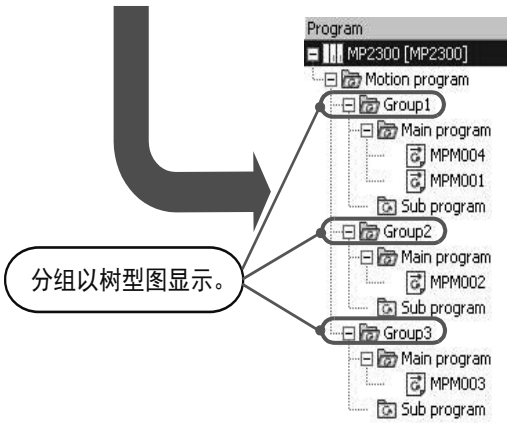
将动作相关联的轴群归为 1 个组，并以组为单位编写运动程序。因此，仅通过 1 台 MP2000 控制器就能够独立控制多台机器。分组运行分为单组运行和多组运行。可以在 “ 分组定义 ” 中对所要分组的轴进行定义。

■ 单组运行



■ 多组运行





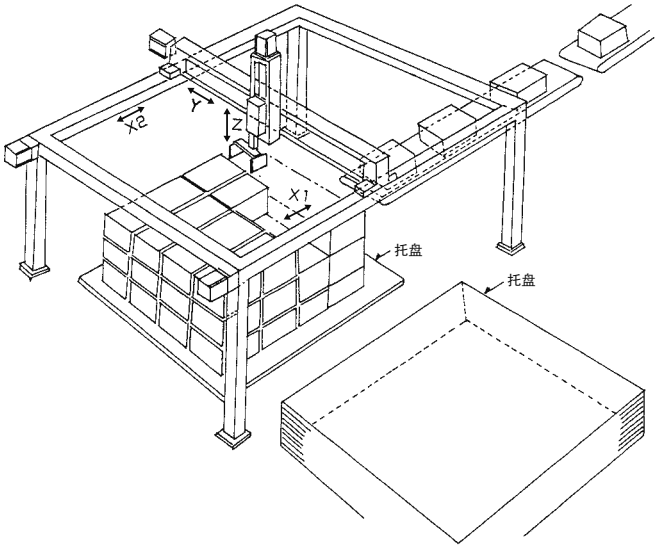
1.7 程序的应用实例

运动程序能够应用于各种装置。以下介绍 4 种该程序的应用实例。

1.7.1 实例 1：搬运装置

概要

- 按照规定个数码放瓦楞箱，并传送到下一工序的装置。
- 除用于码垛的 3 轴运动控制外，可用于自动供应托盘的顺序操作。



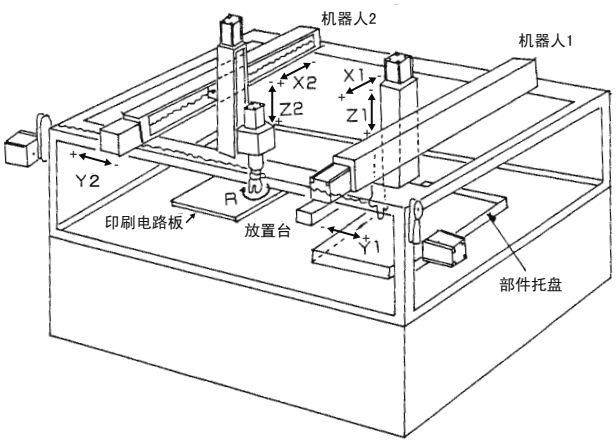
控制要点

- 利用虚拟轴使 X1、X2 轴同步运行。
- 利用插补指令实现平滑动作。
- 根据预先设定的条件 （箱尺寸、横列数、纵列数、堆积层数），通过运动程序计算位置数据，执行码垛堆积。

1.7.2 实例 2：零部件插入机

概要

- 是指在印刷电路板中插入端口等零部件的装置。
- 利用搬运机器人从托盘中取出零部件，搬运到放置台上，并根据插入机器人指定的位置 / 角度插入到电路板中。



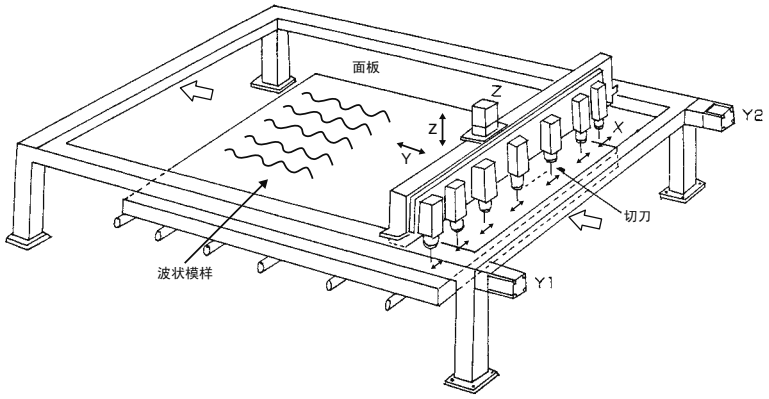
控制要点

- 使用分组运行分别编写 2 个机器人控制程序。
- 利用 2 轴或 3 轴直线插补动作缩短产距。

1.7.3 实例 3：面板加工机

概要

- 在建材用平板中添加图案的装置。
- 在 X 轴上并排配置大于 10 轴的切刀，可自动更改图案的宽度。



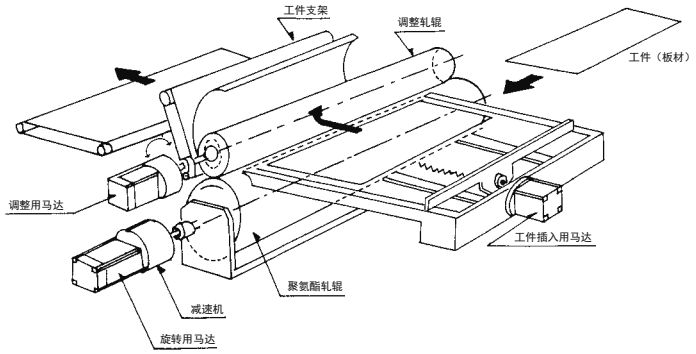
控制要点

- 对 X 轴、Y 轴执行圆弧插补指令，刻划出波形图案。
- 使用虚拟轴同步运行 Y1 轴、Y2 轴。

1.7.4 实例 4：板材成形装置

概要

- 将板材弯曲成型的装置。
- 在通过旋转轴传送板材的同时，改变调整轴，能够将板材加工成任意形状。



控制要点

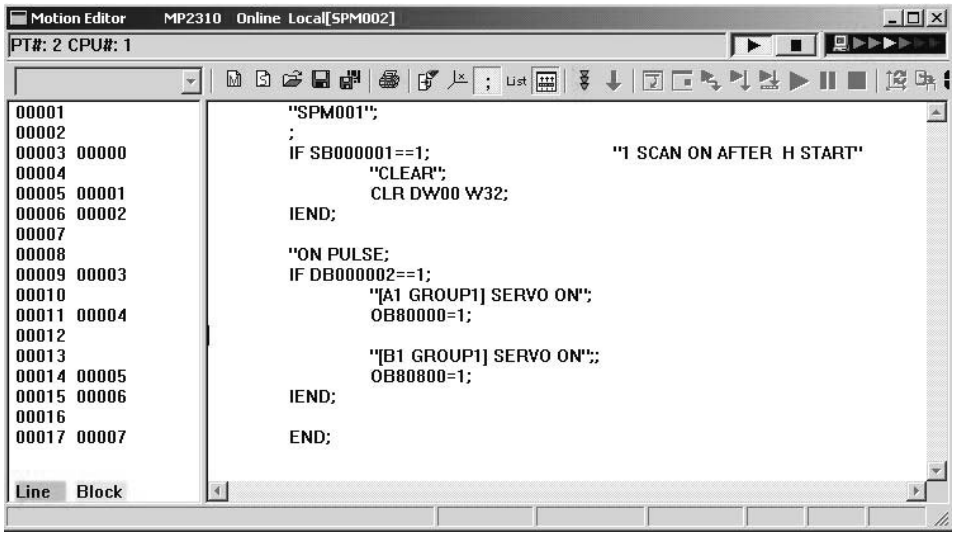
- 针对直线轴和旋转轴 2 个轴执行直线插补控制。
- 根据加工情况切换要调用的运动程序。

1.8 顺序程序

顺序程序是使用与运动程序通用的语言编写的扫描运行型程序。使用顺序程序可构建联锁检测等按固定周期检测状态类的应用程序。
该程序可通过 M-EXECUTOR 模块（注）的运行登录画面进行调用。

（注）M-EXECUTOR 模块是 MP2100，MP2100M，MP2300S，MP2310，MP2400 可使用的模块。在其它的 MP2000 系列机型中不能使用。

顺序程序与运动程序总共最多可创建 256 个。
以下是顺序程序的编写示例。

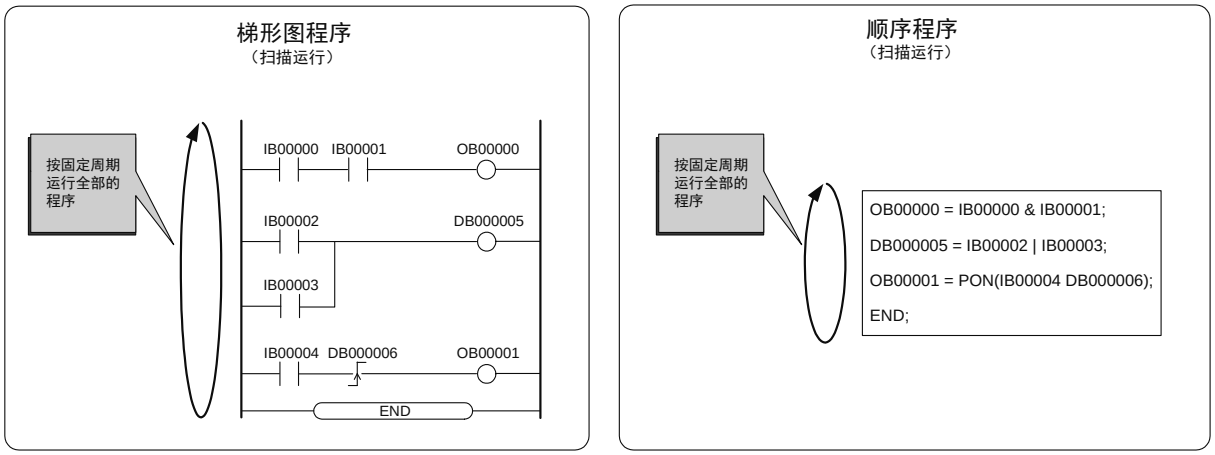


下面对顺序程序的特点进行介绍。

1.9 顺序程序的特点

1.9.1 顺序程序的运行方式

顺序程序采用了与梯形图程序相同的运行方式，顺序程序是按固定周期运行，从程序的起始到 END 命令，在扫描内结束处理，即 “ 扫描运行 ” 型程序。顺序程序可在登录到 M-EXECUTOR 模块的登录画面后使用。

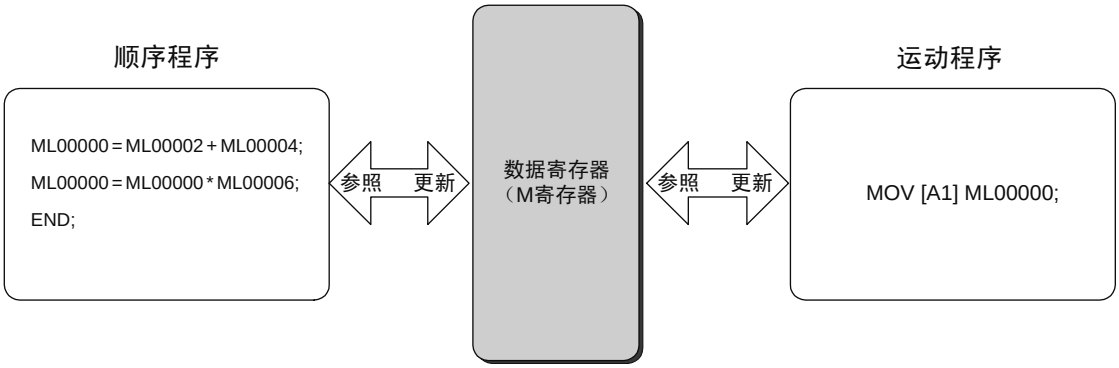


1.9.2 采用了与运动程序相同的运动语言

顺序程序采用了与运动程序相同的运动语言。
顺序程序的运动语言命令中，仅可以使用运算命令等顺序控制命令，不可以使用轴移动命令等运动控制命令。
使用顺序程序，则无需使用梯形图程序也能够创建用于顺序控制的系统。

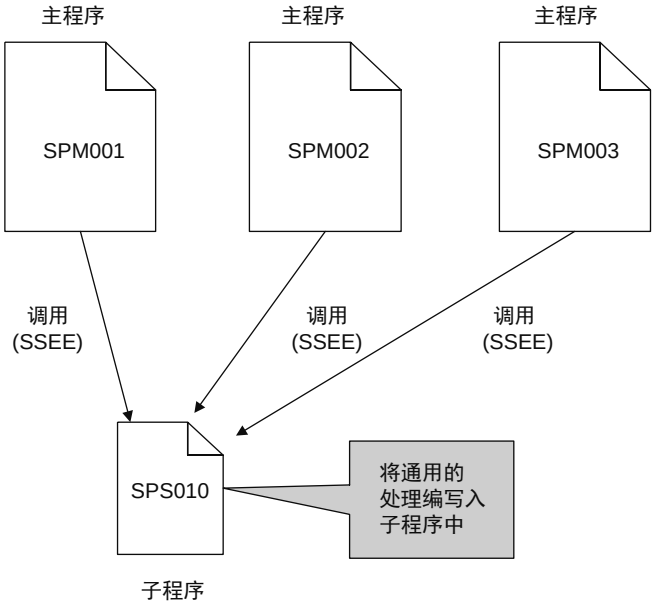
1.9.3 可将数据交换到运动程序

顺序程序与运动程序之间可通过数据寄存器 （M 寄存器）实现数据交换。
通过顺序程序内部更新后的数据能够在运动程序中使用。同样，通过运动程序更新后的数据也可以在顺序程序中使用。



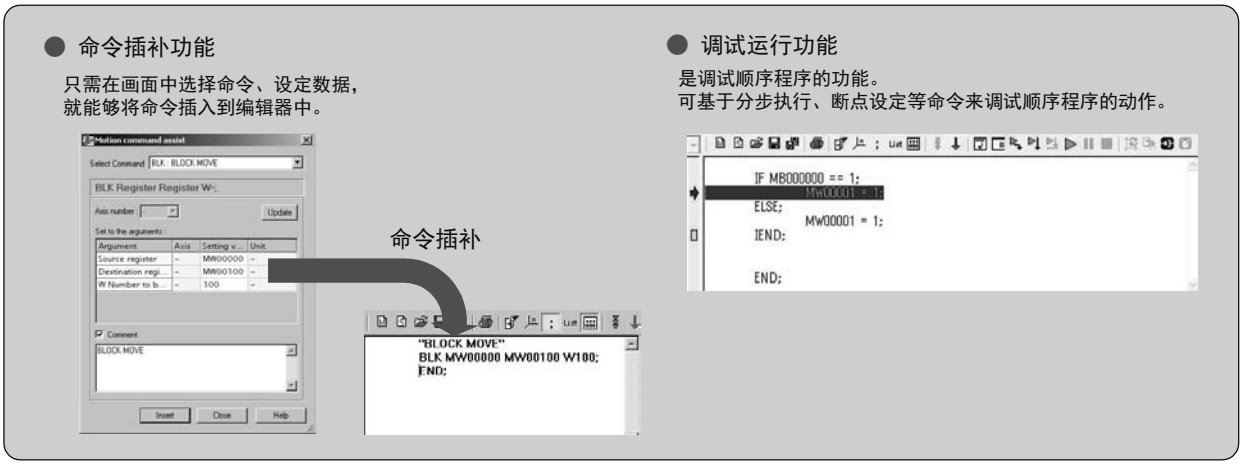
1.9.4 灵活运用子程序最大程度优化内存的利用率

可通过顺序程序创建子程序。
通过将通用动作子程序化 （通用化），能够使程序步数控制到最少，从而提高内存的利用率。



1.9.5 可进行简单编辑 （MPE720 Ver.6.04 以上）

顺序程序也可实现如下所示的简单编程功能。



2 章

规格

本章对与运动程序相关的控制器规格及软件工具（MPE720）规格进行说明。

2.1 MP2000 系列控制器的规格

2.1.1 对应控制器

下面的 MP2000 系列控制器可使用运动程序。

- MP2100
- MP2100M
- MP2200/CPU-01
- MP2200/CPU-02
- MP2300
- MP2300S
- MP2310
- MP2400
- MP2500
- MP2500D
- MP2500M
- MP2500MD

MP2100，MP2100M，MP2300S，MP2310，MP2400 可使用运动程序和顺序程序。

MP2100，MP2100M 可使用 M-EXECUTOR 模块或顺序程序，要用到以下版本的软件工具。

MP2000 系列控制器	支持版本
MP2100 MP2100M	Ver 2.66 或更高

MPE720	支持版本
MPE720 Ver 5	MPE720 Ver 5.44 或更高
MPE720 Ver 6	MPE720 Ver 6.10 或更高 MPE720 Ver 6.10 Lite 或更高

2.1.2 支持的运动模块

可通过支持以下运动模块的运动程序，对连接至模块的轴进行轴控制。

- ◆内置 SVB （MP2100/MP2100M、MP2300、MP2300S、MP2310、MP2400、MP2500/MP2500D/MP2500M/MP2500MD 的标准配置）
- ◆ SVR （MP2000 系列控制器全部机型的标准配置）
- ◆ SVA-01
- ◆ SVB-01
- ◆ PO-01

2.1.3 控制器规格一览

		MP2100、MP2100M	MP2300	MP2300S	MP2400	MP2200/CPU-01	MP2310	MP2200/CPU-02	备注
程序容量		5.5MB				7.5MB		11.5MB	包括梯形图程序、运动程序、顺序程序在内的整个用户程序容量。
梯形图程序		○			×	○			
	起动处理	最多 64 图							
	中断处理	最多 64 图							
	高速扫描处理	最多 200 图							
	低速扫描处理	最多 500 图							
	用户函数	最多 500 图							
运动程序		○							
	程序数	最多 256 个							顺序程序 / 运动程序总共最多可设定 256 个。
	群组数	8 组							1 个群组最多可设定 16 轴。
	任务数	16 个							同时可运行的运动程序数。
	同时处理程序数 （1 个任务）	8 个							主程序 4 个 （同时）× 子程序 2 个 （同时）
	登录	• 通过梯形图程序使用 MSEE 命令 • 使用 M-EXECUTOR 模块（仅限 MP2100，MP2100M，MP2300S，MP2310，MP2400）							
	启动方法	控制信号 Bit0 通过 “ 程序运动开始请求 ” 的启动检测来启动程序							
	超程	可在 0.01% ～ 327.67% 的范围进行设定							
	动作模式	ABS/INC 模式							使用专用命令 （ABS/INC） 来切换模式。
	指令单位	• 内置 SVB/SVB-01/SVR pulse、mm、deg、inch、 μm • SVA-01/PO-01 pulse、mm、deg、inch							
	最小指令单位	• pulse 1 • mm、deg、inch、 μm 1、0.1、0.01、0.001、0.0001、0.00001							
	指令范围	-2147483648 ～ +2147483647 （带 32Bit 符号）							
	同时控制轴数 （1 个任务）	最多 16 轴							

		MP2100、MP2100M	MP2300	MP2300S	MP2400	MP2200/CPU-01	MP2310	MP2200/CPU-02	备注
		○	×	○		×	○	×	
顺序程序	程序数	最多 256 个 （可在启动处理、高速扫描处理、低速扫描处理中选择运行时刻）							顺序程序 / 运动程序总共最多可设定 256 个。
	任务数	最多 16 任务							同时可运行的顺序程序数
	同时处理程序数 （1 个任务）	无							
	登录	使用 M-EXECUTOR 模块							
	启动方法	通过系统进行启动							将程序登录到 M-EXECUTOR 模块，利用系统进行启动。
可访问的寄存器	M 寄存器	○ （65535 字）							电池备份内存
	S 寄存器	○ （8192 字）							电池备份内存
	I 寄存器	○ （32768 字 ＋ 运动监控器参数）							
	O 寄存器	○ （32768 字 ＋ 运动设定参数）							
	C 寄存器	○ （16384 字）							
	D 寄存器	○ （可在 0 ～ 16384 字内进行指定）							各 DWG 特有的内部寄存器。只能使用相关 DWG。
	# 寄存器	△ （可在 0 ～ 16384 字内进行指定）							可通过梯形图程序访问。不可以通过运动程序、顺序程序访问。

2.2 软件工具的规格

2.2.1 对应的软件工具

通过下面的软件工具可管理运动程序或顺序程序。

- ◆ MPE720 Ver.5 （MP2400 除外的 MP2000 系列全部机型）
- ◆ MPE720 Ver.6 （MP2000 系列全部机型）
- ◆ MPE720 Ver.6 Lite （仅限 MP2400）

补充

使用以上软件工具，需首先将其安装到 1 台计算机中。

2.2.2 软件工具的规格一览

		MPE720 Ver.5 (CPMC-MPE720)	MPE720 Ver.6 (CPMC-MPE770)	MPE720 Ver.6 Lite (CPMC-MPE770L)	备注
对应控制器	MP2100	○		×	
	MP2100M	○		×	
	MP2200/CPU-01	○		×	
	MP2200/CPU-02	○		×	
	MP2300	○		×	
	MP2300S	○		×	
	MP2310	○		×	
	MP2400	×	○		
程序	梯形图程序	○		×	
	运动程序	○			
	顺序程序	○			
命令输入辅助		×	○ （Ver.6.04 以上）		
程序登录功能		○ （Ver.5.38 以上）	○ （Ver.6.04 以上）		仅适用于 MP2100、MP2100M、MP2300S、MP2310、MP2400 机型
调试运行		○			
运动任务管理器		○			
运行控制面板		○ （Ver.5.38 以上）	○ （Ver.6.04 以上）		仅适用于 MP2100、MP2100M、MP2300S、MP2310、MP2400 机型
测试运行		×	○ （Ver.6.04 以上）		
轴运行监视器 / 警报监视器		×	○ （Ver.6.04 以上）		

2.3 运动语言命令一览

类型	命令	功能	类型	命令	功能
轴设定命令	ABS	绝对值模式	数值运算	=	代入
	INC	增量模式		+	加法
	ACC	加速时间变更		-	减法
	DCC	减速时间变更		*	乘法
	SCC	S 字时间常数变更		/	除法
	VEL	进给速度变更		MOD	除余
	FMX	插补进给最高速度设定	逻辑运算	|	OR(逻辑或)
	IFP	插补进给速度比率设定		&	AND(逻辑与)
	IAC	插补加速时间变更		^	XOR(异或)
	IDC	插补减速时间变更		!	NOT(逻辑非)
轴移动命令	MOV	定位	数值比较	==	一致
	MVS	直线插补		<>	不一致
	MCW	圆弧插补、螺旋插补 (顺时针)		>	较大
	MCC	圆弧插补、螺旋插补 (逆时针)		<	较小
	ZRN	原点复归		>=	以上
	SKP	带跳过功能的直线插补		<=	以下
	MVT	时间指定定位	数据操作	SFR	位右移
	EXM	外部定位		SFL	位左移
控制命令	POS	变更当前值		BLK	段传输
	MVM	机械坐标系指令		CLR	清除
	PLN	坐标平面指定		ASCII	ASCII 转换 1
	PLD	程序当前位置更新	基本函数	SIN	正弦
	PFN	到位 (In position) 检查		COS	余弦
	INP	到位检查宽度设定		TAN	正切
程序控制命令	IF、ELSE、IEND	分支		ASN	反正弦
	WHILE、WEND	重复		ACS	反余弦
	PFORK、JOINTO、PJOINT	同时执行		ATN	反正切
				SQT	平方根
				BIN	BCD → BIN
	SFORK、JOINTO、SJOINT	选择执行		BCD	BIN → BCD
				S{ }	指定位 ON
				R{ }	指定位 OFF
	MSEE	运动子程序调用		PON	上升脉冲
	SSEE	顺序子程序调用		NON	下降脉冲
	UFC	调用运动程序的用户函数		TON	接通延时定时器
				TOF	断开延时定时器
	FUNC	调用顺序程序的用户函数	C 语言控制命令	CTSK	C 语言任务控制
	END	程序退出			
	RET	子程序退出			
	TIM	时间等待		CFUNC	C 语言函数调用
	IOW	输入 / 输出变量等待			
	EOX	1 扫描 WAIT			
	SNGD/SNGE	单段无效 / 有效			

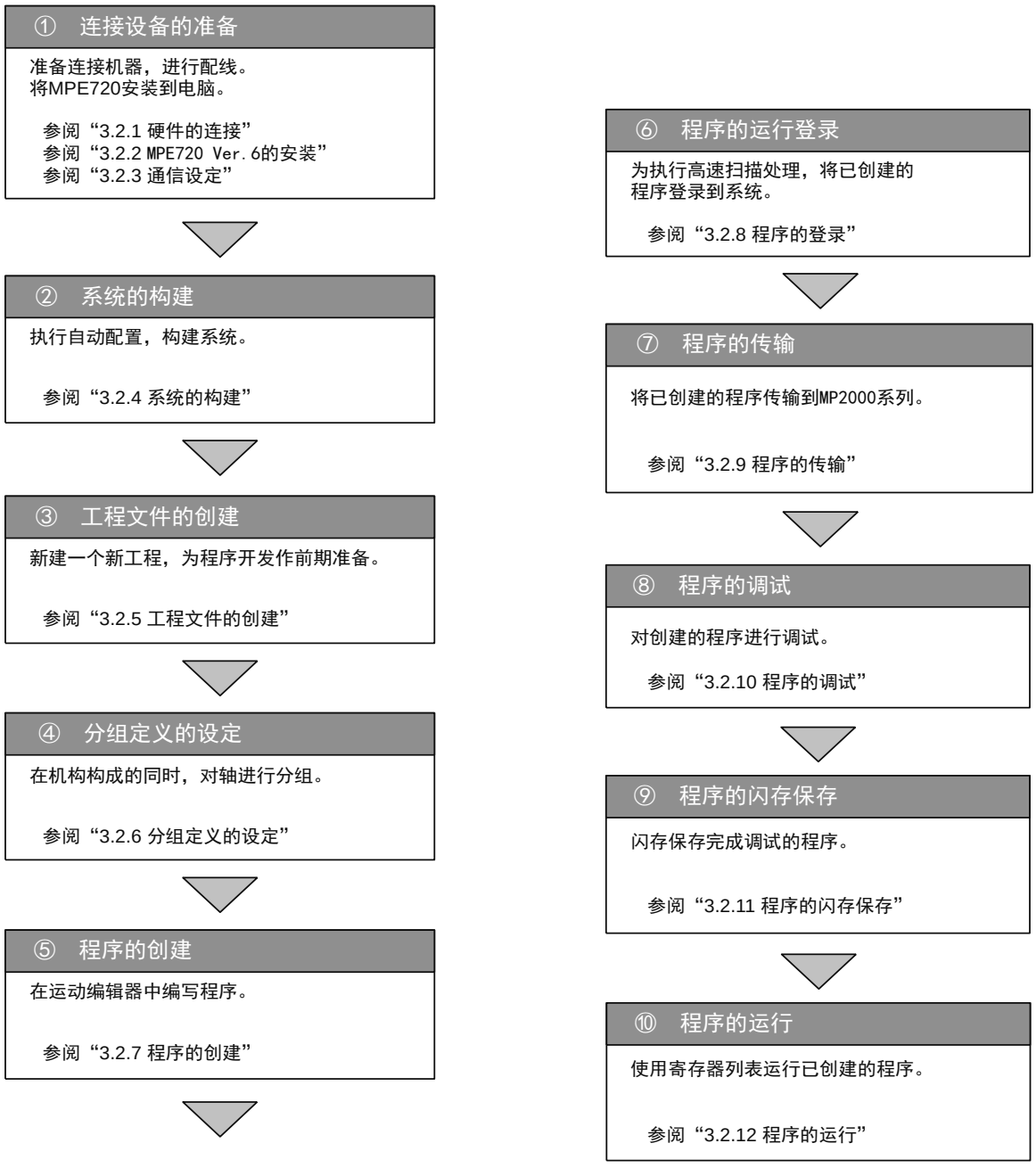
3 章

程序的开发流程

本章介绍使用软件工具 MPE720 Ver.6 时，从系统启动到实机运行的程序开发流程。

3.1 程序的开发流程

运动程序开发流程如下所示。



（注） 1. 运动程序与顺序程序的开发步骤基本相同。
在此省略顺序程序开发流程的介绍。
2. 上述流程是程序开发的示例。将程序应用到实际装置时，请设置外部机器。

■ 要创建的运动程序

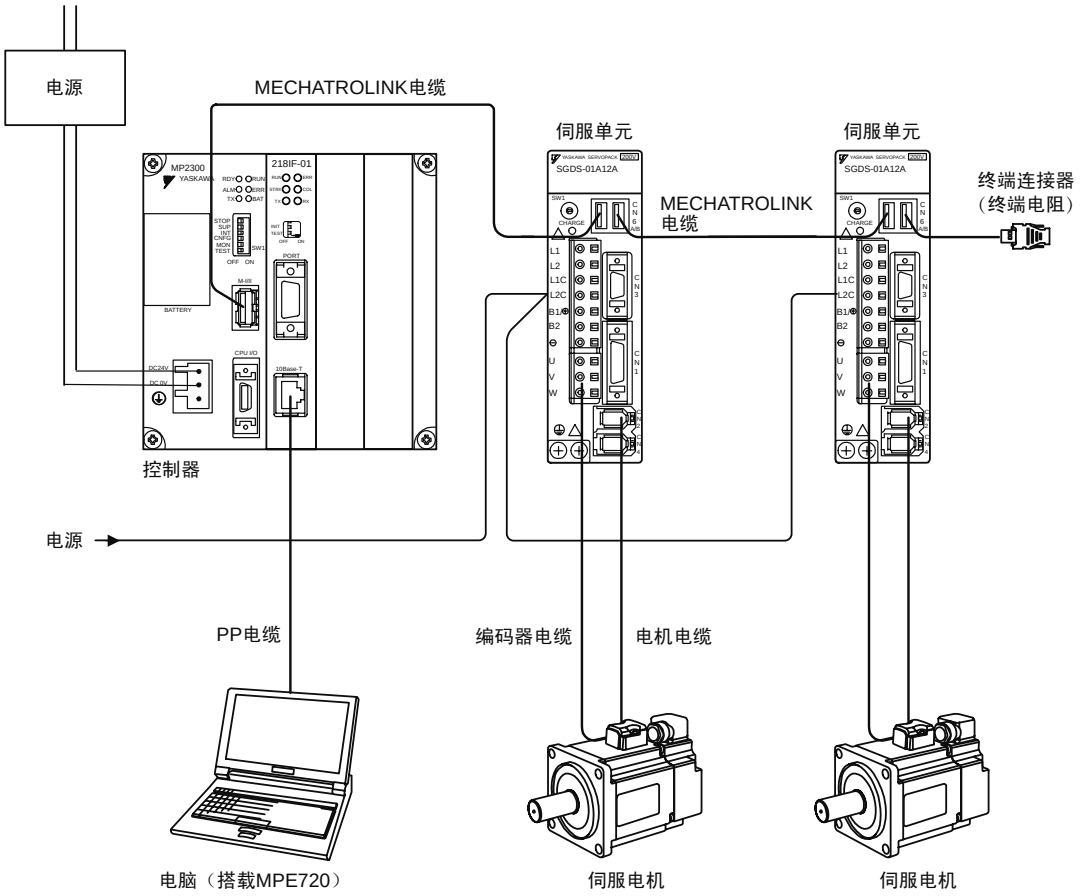
本章以下面的运动程序为例说明运动程序的开发流程。要创建的运动程序仅 3 行，内容非常简单，即从当前位置移动 150000 脉冲后停止。

```
INC;                                    “ 指定增量模式 ”
MOV [A1]150000 [B1]150000;              “2 轴 150000 脉冲定位 ”
END;
```

3.2 程序的开发步骤

3.2.1 硬件的连接

在本章中的程序开发流程中，使用以下系统构成模型进行说明。



（注） 请在上述示例中将伺服单元的站编号分别设为 1 和 2。

3.2.2 MPE720 Ver.6的安装

将MPE720 Ver.6安装到电脑。

关于安装步骤，请参阅《MP2000系列MPE720 Ver.6.0用户手册》（资料编号：SIJP C880700 30）（日文）。

3.2.3 通信设定

设定已安装有MPE720 Ver.6的电脑与MP2000系列之间通信的条件。

关于通信设定的步骤，请参阅《MP2000系列MPE720 Ver.6.0用户手册》（资料编号：SIJP C880700 30）（日文）。

3.2.4 系统的构建

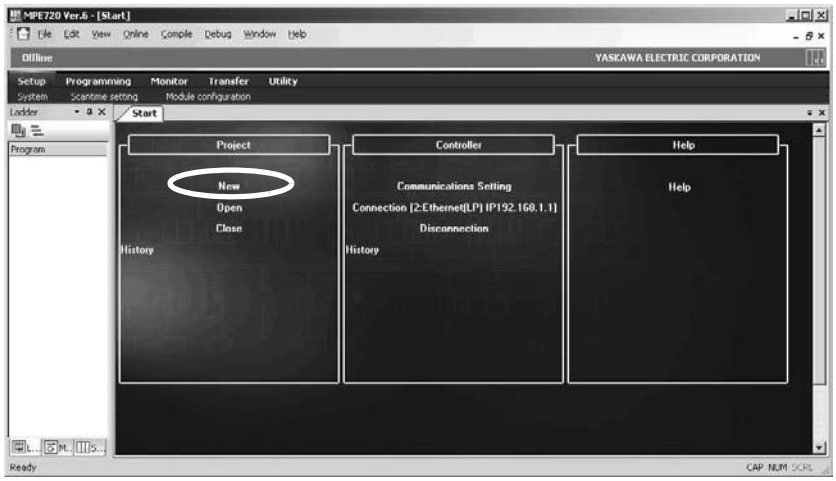
使用自动配置功能来构建系统。
自动配置，是指自动识别MP2000系列控制器中安装的模块以及连接到MECHATROLINK端口上的机器的功能。这样就能够非常简单地在短时间内完成系统构建。

自动配置可通过设备控制器接通电源时或MPE720执行。关于自动配置的步骤，请参阅相应的机器控制器的用户手册。

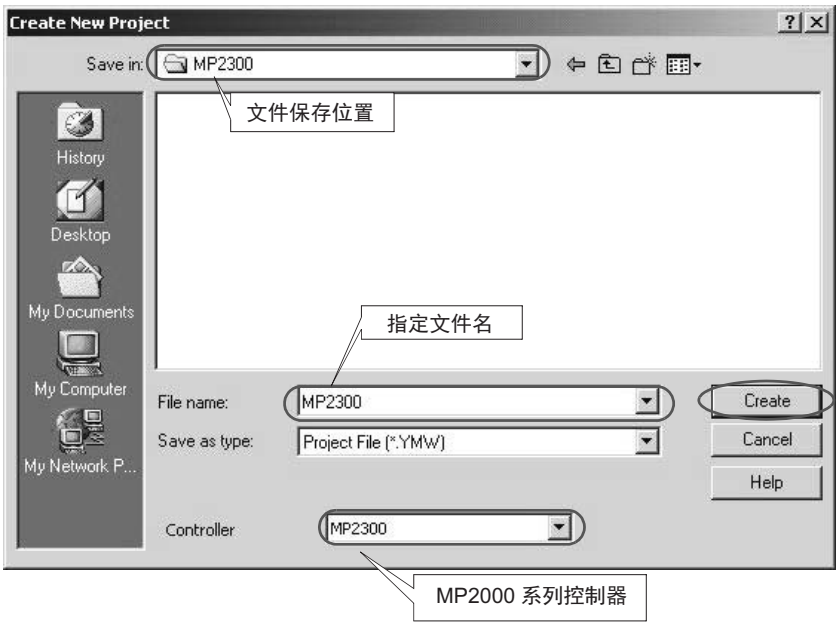
3.2.5 工程文件的创建

1. 双击电脑桌面上的图标，启动 MPE720 Ver.6。

2. MPE720 Ver.6 启动后，点击 ***New***。



3. 指定文件名、文件保存位置、控制器机型后，点击 ***Create*** 按钮。

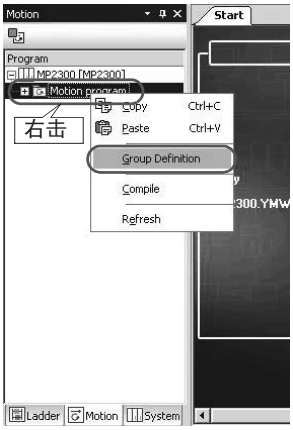


3. 2. 6 分组定义的设定

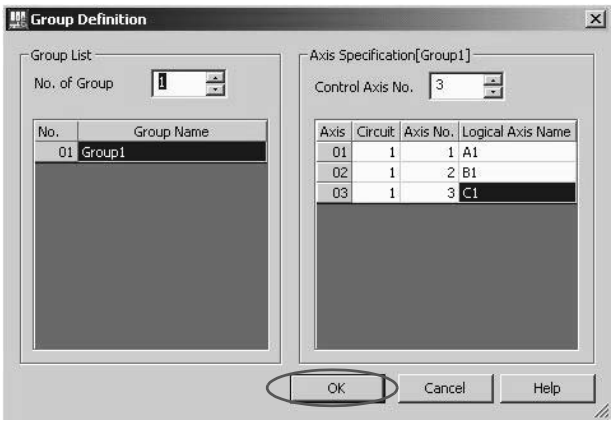
创建运动程序之前，在构成机构的同时，对轴进行分组。

1. 点击子窗口中的 ***Motion*** 选项卡，在子窗口中弹出 ***Motion program*** （运动程序）窗口。

2. 鼠标右键点击子窗口中的 ***Motion program***，从弹出菜单中选择 ***Group Definition*** （分组定义）。



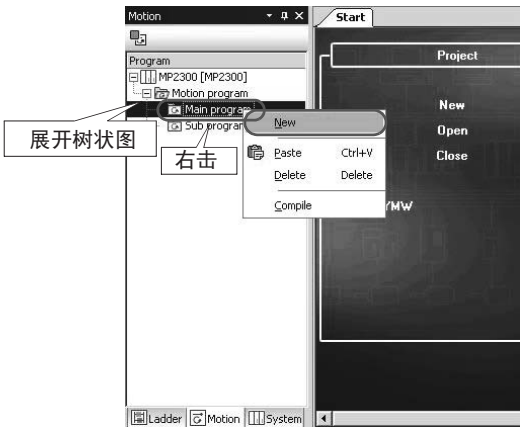
3. 单击 ***OK*** 按钮。分组定义的详细内容，请参阅 “7. 3 分组定义 ” 。



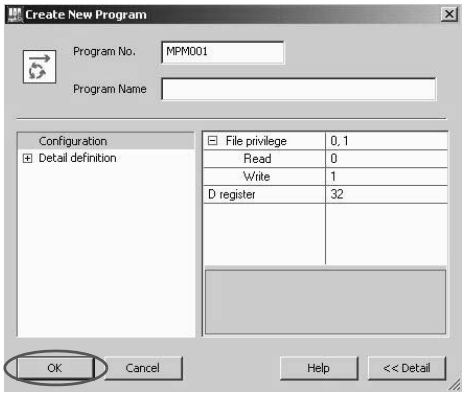
3. 2. 7 程序的创建

启动运动编辑器，编写运动程序。

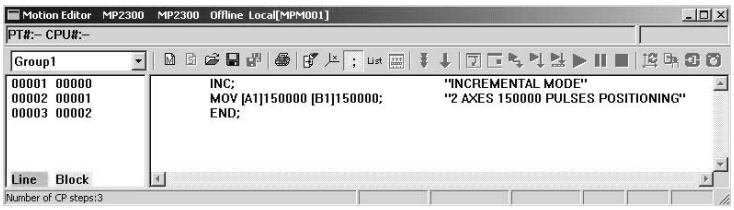
1. 展开子窗口中的运动程序树状图。鼠标右键点击 ***Main program***，在弹出的菜单中点击 ***New***。



2. 单击 ***OK*** 按钮。



3. 编写本章开始部分记载的运动程序示例。



4. 在运动编辑器中点击 “  ” 图标，执行编辑。编译完成后，自动保存运动程序。

重要

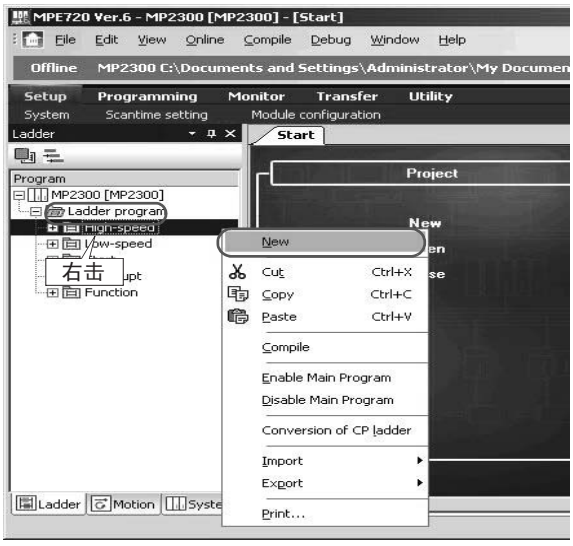
编译时如果显示 ***Error List***（错误列表）对话框，则不会保存运动程序。

3.2.8 程序的登录

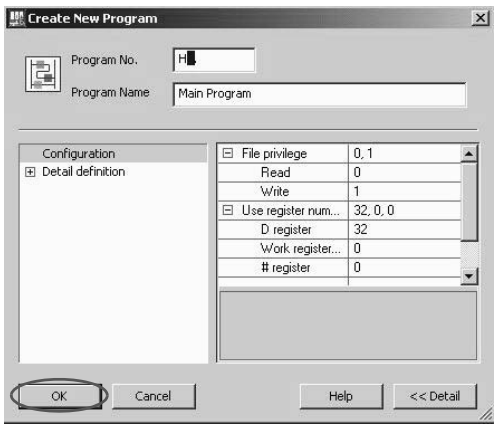
使用 MSEE 命令从 DWG.H 中调用已创建的运动程序。详细信息请参阅 “4.3.2 程序的登录 ”。

1. 单击子窗口中的 ***Ladder*** （梯形图）选项卡。子窗口中会显示 ***Ladder program***。

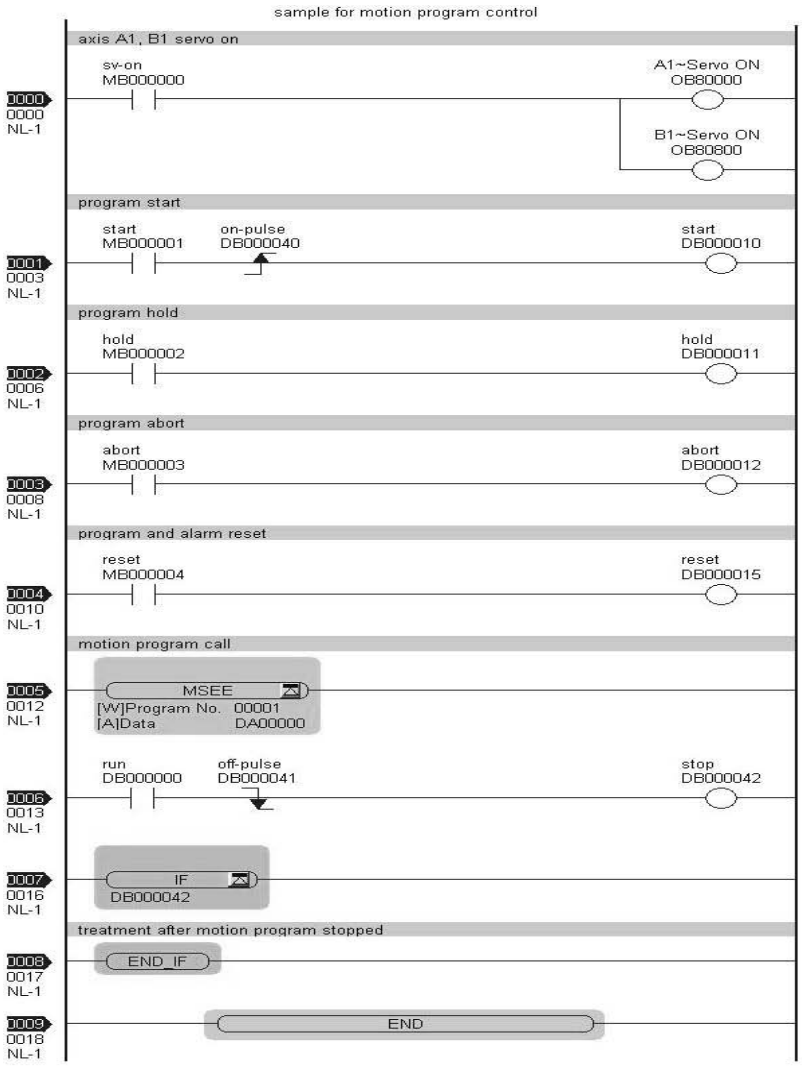
2. 鼠标右键点击子窗口中的 ***High-speed***，在弹出的菜单中点击 ***New***。



3. 单击 ***OK*** 按钮。



4. 创建以下所示的梯形图程序。编写后，执行编译 （F8 键或 “ ” 图标）。



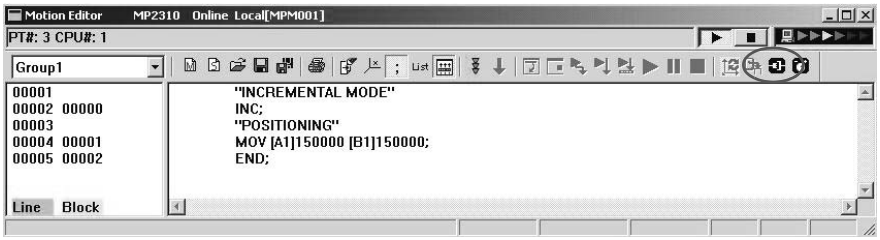
补充

- 请先确认运动监控器参数 IWxx00 Bit0 （机器控制器运行准备完毕）已启动，再启用伺服 ON 指令 “MB000000”。
- 如果机器控制器运行准备完毕为 OFF，则无法接收伺服 ON 指令。

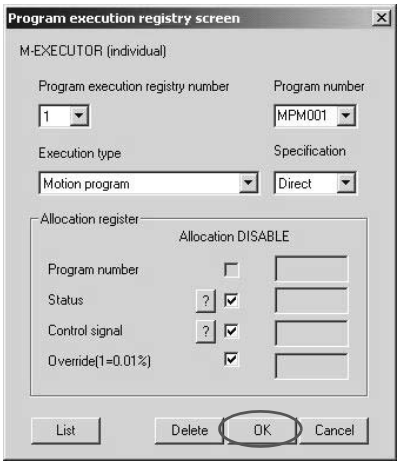
补充

也可以不创建上一页中的梯形图程序，而是将程序登录到 M-EXECUTOR 程序定义，也能够完成登录。
以下为登录到 M-EXECUTOR 程序定义的步骤。但在进行以下操作前，请先进行下一页的 “3.2.9 程序的传输 ” 操作。

1. 在已完成的程序创建的运动编辑器上单击登录图标 “  ”。



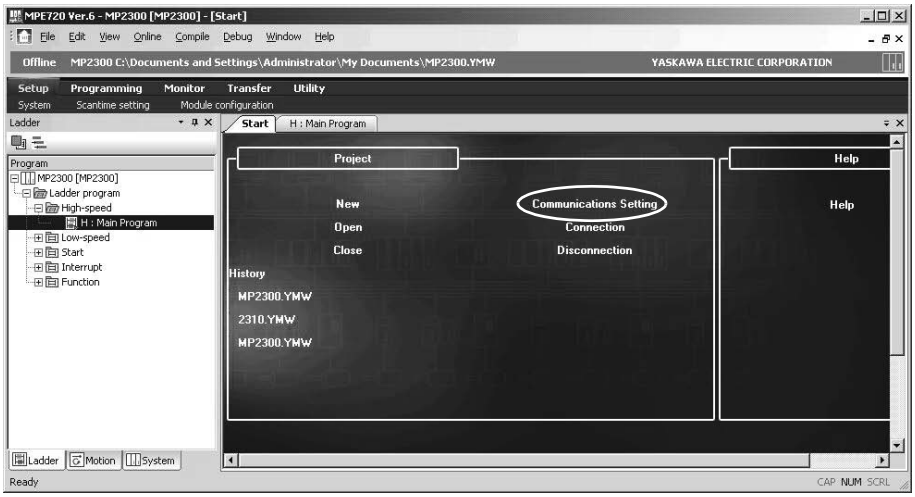
2. 显示程序登录画面。按下 ***OK*** 按钮，登录程序。



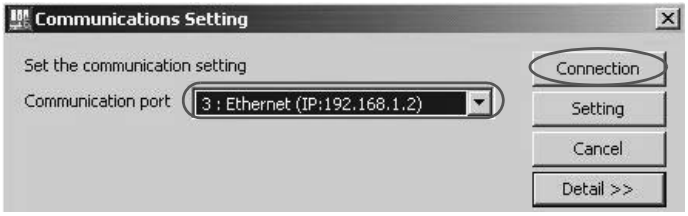
3.2.9 程序的传输

将运动程序传输到 MP2000 系列。在线创建运动程序时，不需要该步骤。

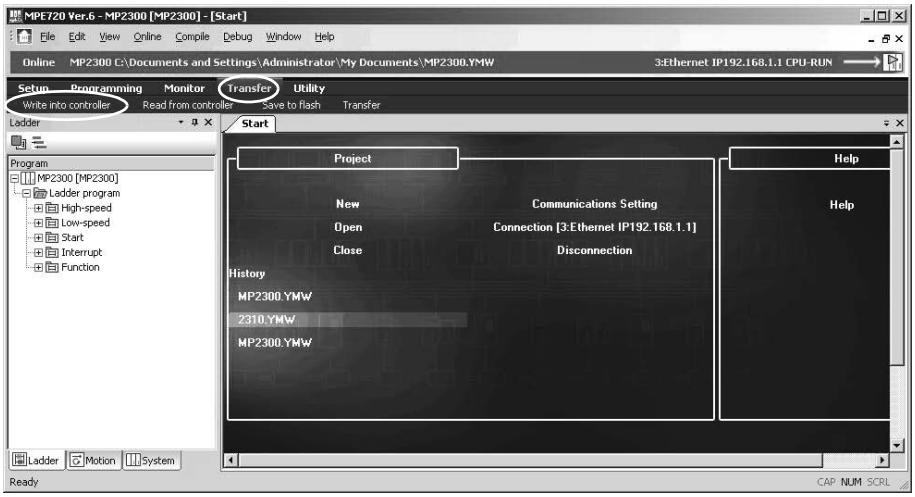
1. 在以下画面中单击 ***Communications Setting*** （通信设定）。



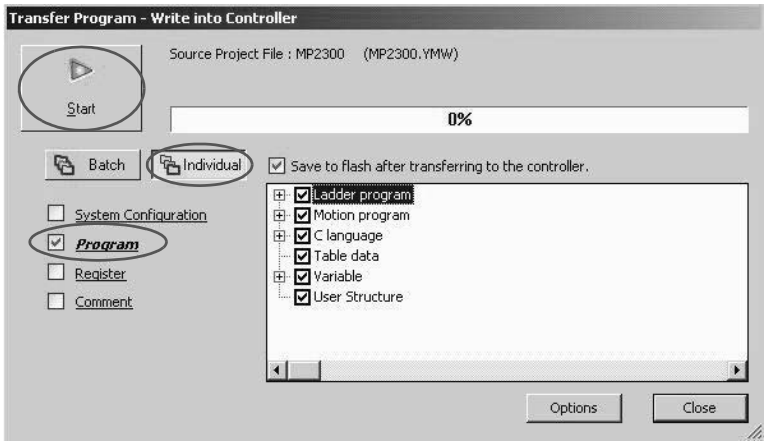
2. 选择通过 “3.2.3 通信设定 ” 设定的通信端口，单击 ***Connection*** 按钮。



3. 联机后，单击 ***Transfer*** （传输）—***Write into controller*** （写入控制器）。



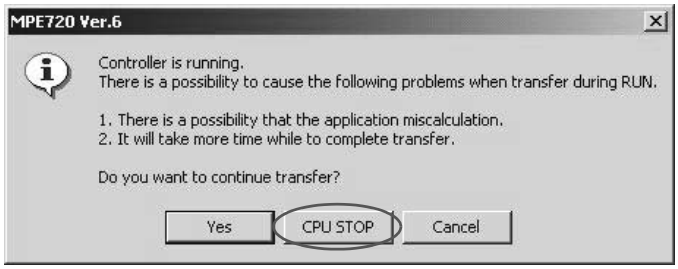
4. 单击 ***Individual*** （个别）按钮，只勾选 ***Program*** （程序）。单击 ***Start*** （开始）按钮。



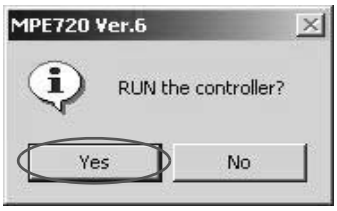
补充

- 个别传输时，控制器中的同名文件会被所选的项目文件数据覆盖。
- 批量传输时，在传输之前会清除 MP2000 系列的 RAM，批量写入项目文件数据。

5. 单击 ***CPU STOP*** 按钮。开始传输。



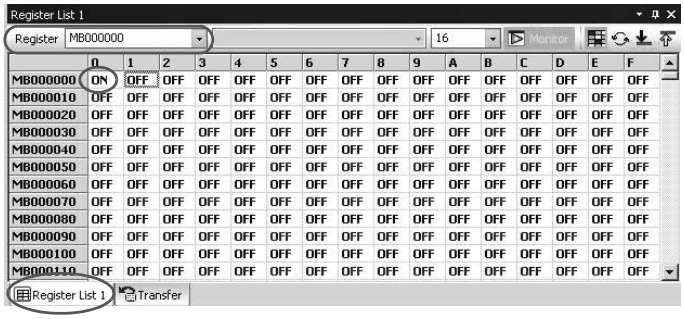
6. 在以下画面中选择 ***Yes***，运行控制器。



3.2.10 程序的调试

对创建的程序进行调试。调试的详细情况，请参阅 “9.4 调试运行 ”。

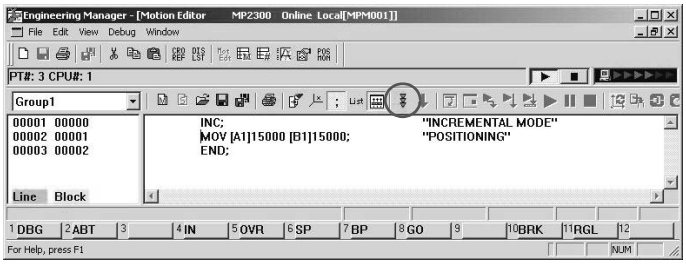
1. 单击 ***Register List1***（寄存器列表 1）选项卡，显示寄存器列表。寄存器指定 MB000000。如下图所示，在 MB000000 中编写 ***ON***，伺服 ON。



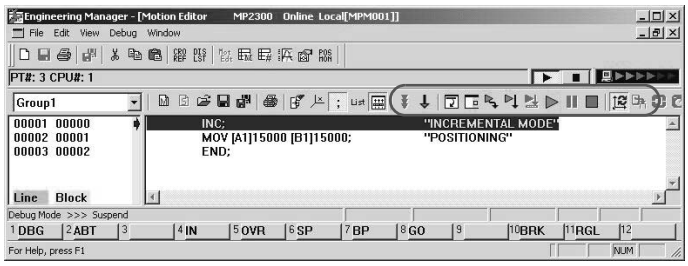
补充

“3.2.8 程序的登录 ” 中使用了 M-EXECUTOR 时，操作直接运动设定参数，执行伺服 ON。

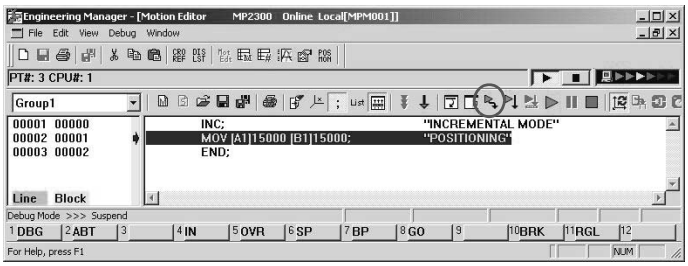
2. 单击 ***Debug Mode***（调试模式）图标 “ ”。



3. 切换到调试模式。



4. 单击 ***Step In*** 图标 “ ”，逐行运行程序，确认程序的动作。关于调试运行的内容，请参阅 “9.4 调试运行 ”。

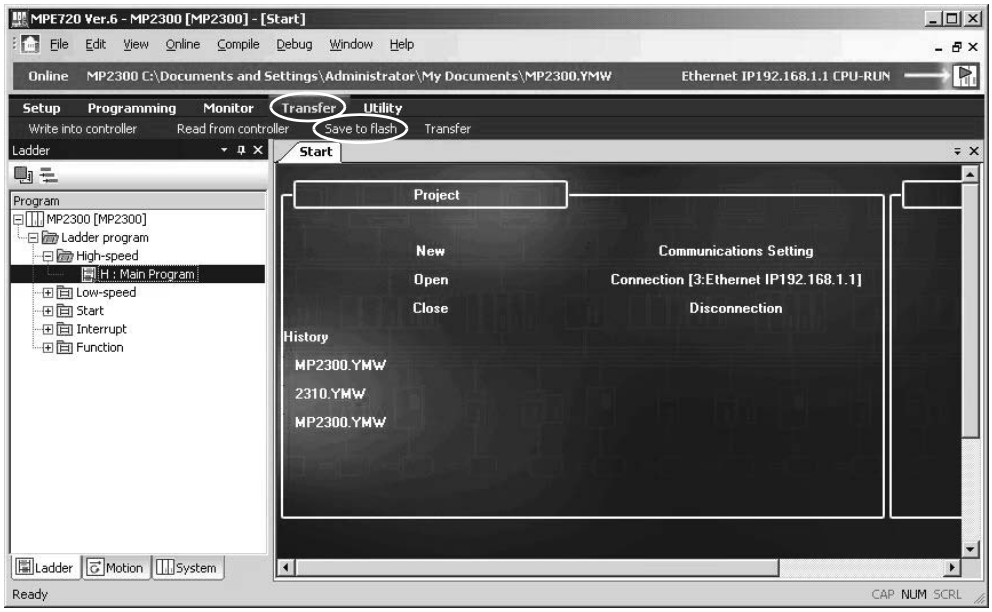


5. 一直执行到 END 命令，调试运行结束后，伺服 OFF。

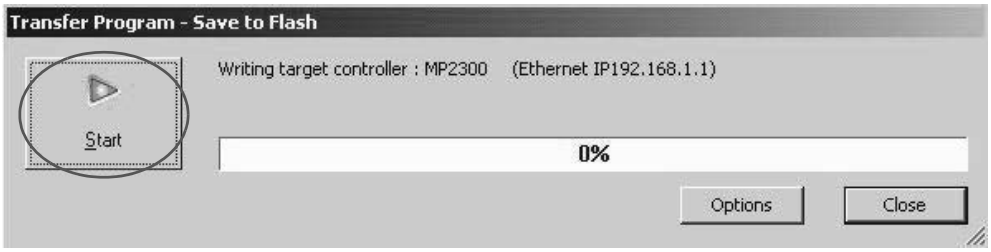
3. 2. 11 程序的闪存保存

闪存保存 MP2000 系列的 RAM 中的数据。

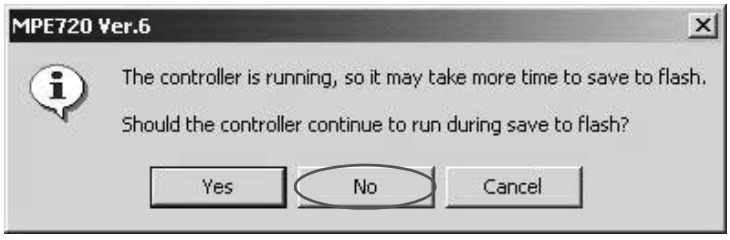
1. 在以下画面中单击 ***Transfer*** （传输）-***Save to flash*** （闪存保存）。



2. 单击 ***Start*** 按钮。



3. 单击 ***CPU STOP*** 按钮。开始传输。

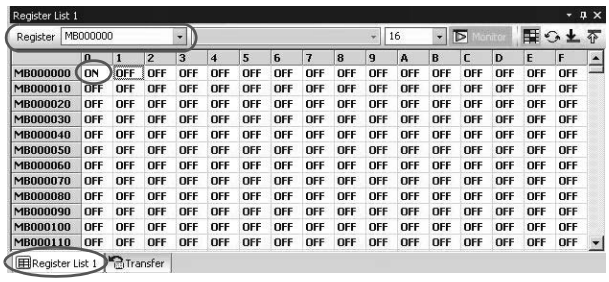


4. 在以下画面中选择 ***Yes***，运行控制器。

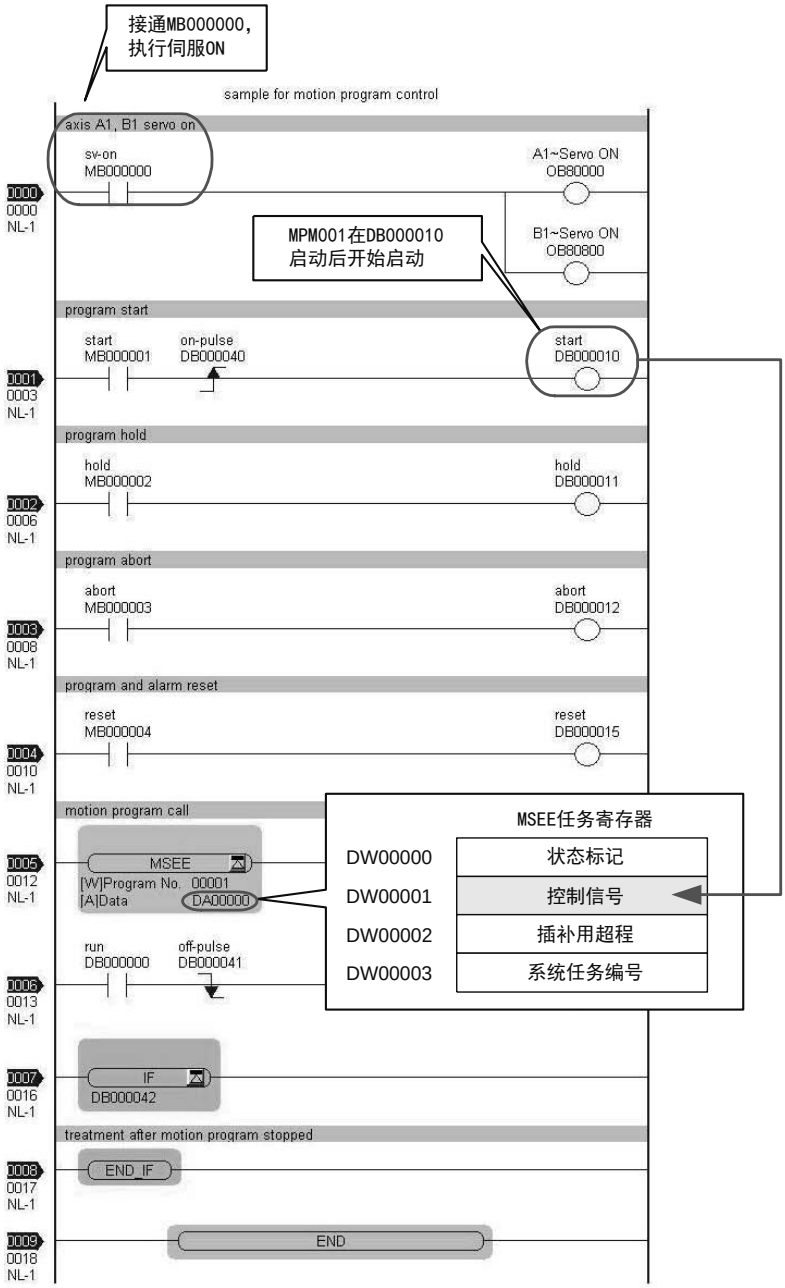
3.2.12 程序的运行

通过控制信号启动程序运行开始请求，用来运行已创建的运动程序，从而对机器进行控制。

1. 单击 ***Register List1*** （寄存器列表 1）选项卡，显示寄存器列表。寄存器指定 MB000000。如下图所示，在 MB000000 中编写 ***ON***，伺服 ON。



2. 在寄存器列表的 MB000001 中编写 ***ON***，执行运动程序 MPM001。



4 章

运动程序

本章对运动程序的种类及运行方法进行说明。

4.1 运动程序的种类

如下表中所示，运动程序分为 2 种。

类别	指定方法	特征	程序数量
主程序	MPM □□□ （□□□ =1 ～ 256）	• 从 M-EXECUTOR 程序定义中调用 • 从 DWG.H 调用	以下程序合计最多可创建 256 个 • 运动主程序 • 运动程序 • 顺序主程序 • 顺序程序
子程序	MPS □□□ （□□□ =1 ～ 256）	• 从主程序调用	

补充

■ 运动程序与顺序程序的程序编号统一进行管理。
程序编号请使用互不相同的编号。
- 运动程序 MPM □□□、MPS □□□的程序编号
- 顺序程序 SPM □□□、SPS □□□的程序编号

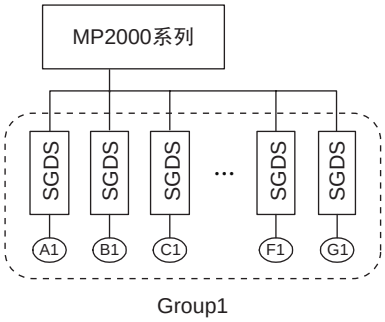
■ MP2000 系列可同时运行的运动程序的数量为 16 个。
同时运行 17 个以上的程序时，会发生警报 （无系统任务错误）。
- 无系统任务错误：运动程序的状态标记的 BitE

4.2 运动程序的分组

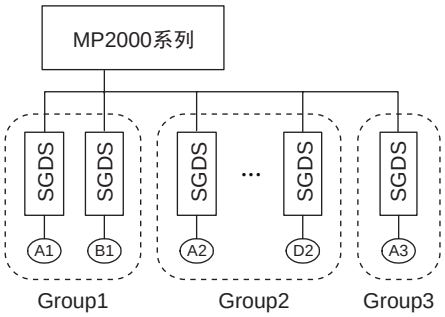
在运动程序中，可以将动作相关的轴群归纳为 1 个组，并按组进行编程。因此，利用 1 台 MP2000 控制器就能够独立控制多台机器。分组运行可分为单组运行和多组运行 2 类。

可以在 “ 分组定义 ” 中对所要归纳为组的轴进行定义。关于分组定义的画面设定，请参阅 “7.3 分组定义 ”。

■单组运行



■多组运行



4.3 运动程序的运行

4.3.1 运动程序的运行方式

要运行运动程序，需将已创建的程序登录到系统中。登录到系统之后，运动程序就可以按照 H 扫描周期进行引用。

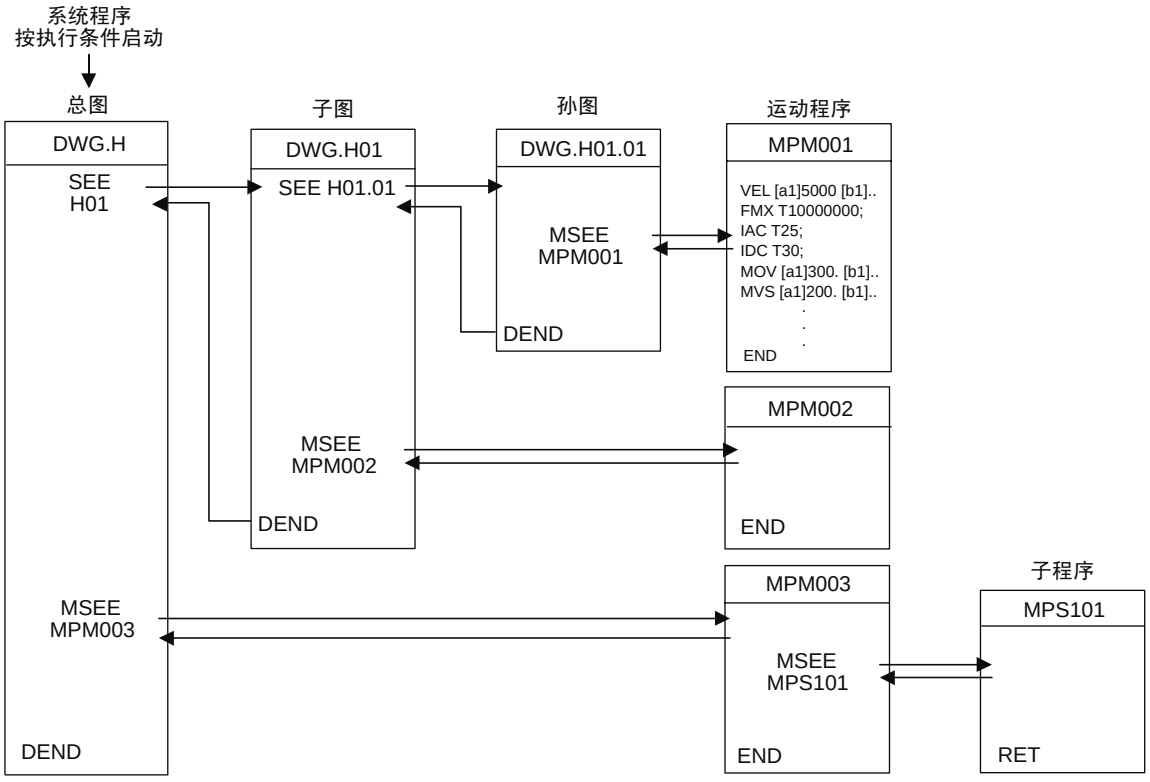
运动程序登录到系统的方式有以下 2 种。

- 利用 MSEE 命令从梯形图程序进行调用
- 登录到 M-EXECUTOR 程序定义

下页分别对这 2 种运行方式进行介绍。

■ 利用 MSEE 命令从梯形图程序进行调用的方式

运动程序创建后，将 MSEE 命令 （运动程序调用命令）加入 DWG.H。如果是 H 图，也可通过总图、子图、孙图三者中的任何一个图运行。下图为运行实例。



H 图的梯形图指令将在各高速扫描周期内按总图 - 子图 - 孙图的层次构成顺序来执行。
加入 MSEE 命令，通过控制信号启动程序运行开始请求后，运动程序开始运行。
运动程序会按照扫描周期运行，但无法像梯形图程序那样一个扫描周期内运行全部程序。因此，可通过专门的系统运动管理功能运行和管理运动程序。

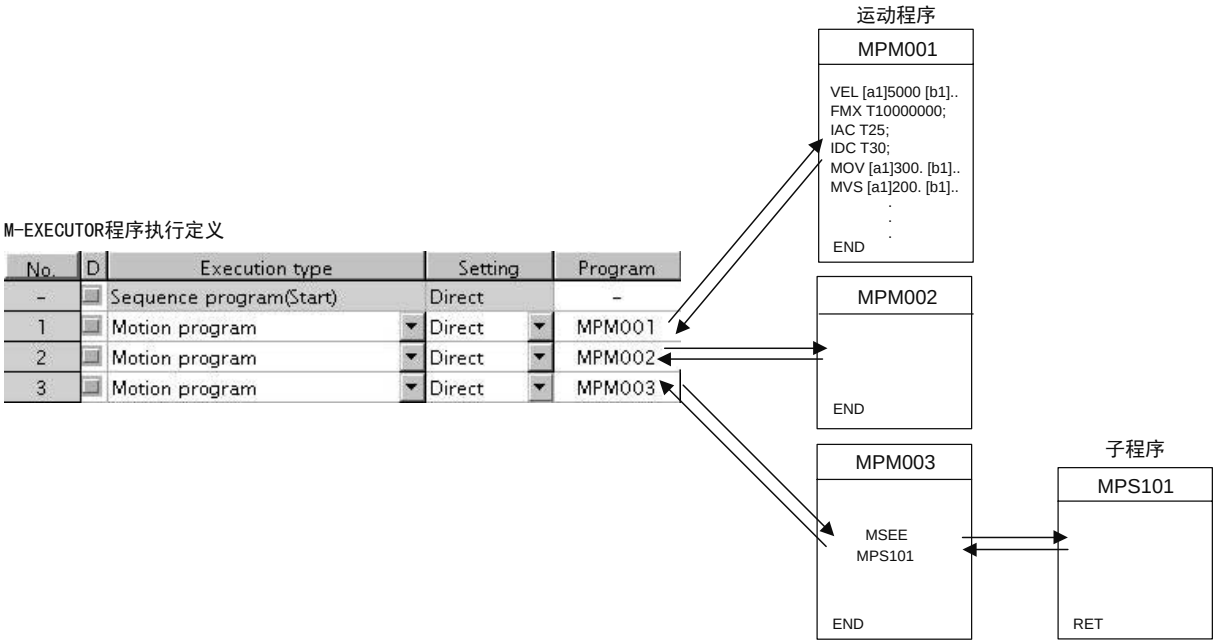
重要

运行运动程序时，请注意以下几点。

- 无法通过 MSEE 命令运行已登录到 M-EXECUTOR 的运动程序。
- 无法利用 MSEE 指令来运行多个相同编号的运动程序。
- 无法通过梯形图的 MSEE 命令来运行子程序 （MPS □□□）。
  只能从运动程序 （MPM □□□，MPS □□□）来引用。
- 无法通过梯形图的 MSEE 命令来运行顺序程序 （SPM □□□，SPS □□□）。

■ 登录到 M-EXECUTOR 程序定义的方法

创建运动程序后，登录到 M-EXECUTOR 程序定义画面。
登录到 M-EXECUTOR 程序定义画面中的程序会按照编号顺序依次运行。下图为运行实例。



登录到 M-EXECUTOR 程序定义，通过控制信号启动程序运行开始请求后，运动程序开始运行。
已登录到 M-EXECUTOR 的运动程序会按扫描周期运行，但无法像梯形图那样 1 次扫描运行全部程序。
因此，可通过专门的系统运动管理功能运行和管理运动程序。

重要

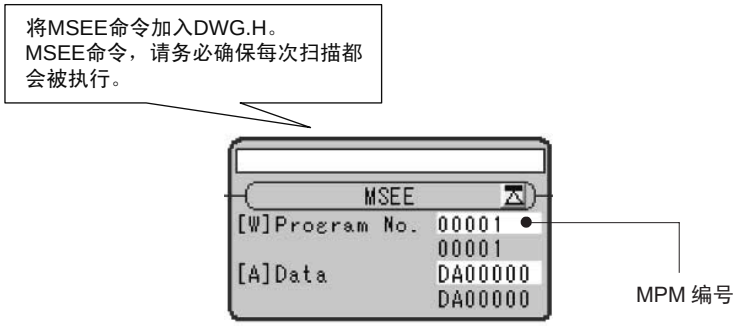
将运动程序登录到 M-EXECUTOR 时，请注意以下几点。

- 无法登录多个相同编号的运动程序。
- 无法使用间接指定引用多个相同编号的运动程序。

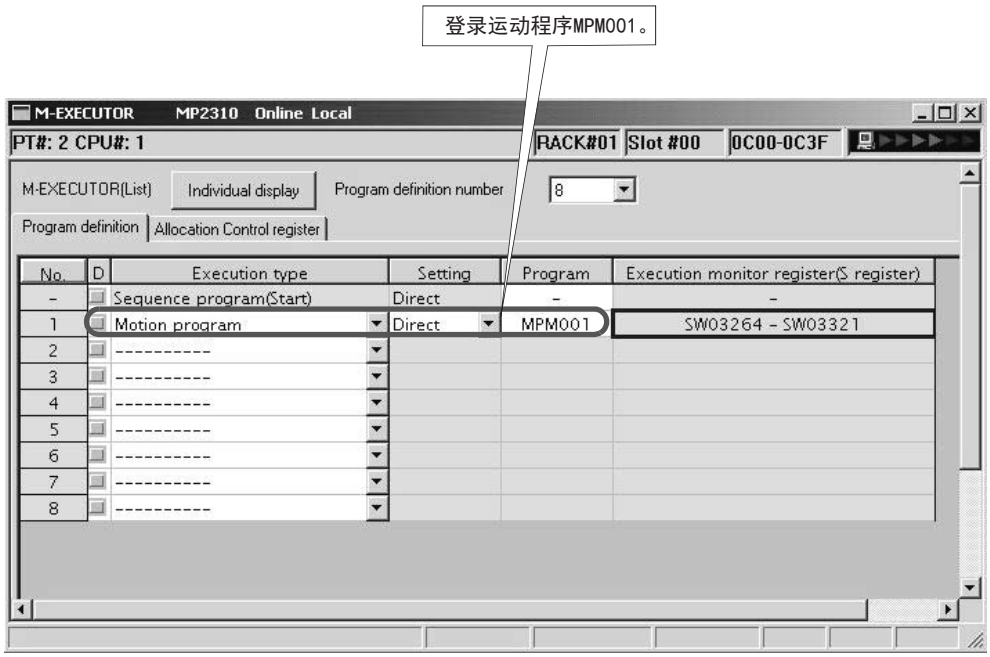
4.3.2 程序的登录

程序的登录有以下 2 种方式。
下面以登录运动程序 MPM001 为例进行说明。

■ 将 MSEE 命令加入梯形图程序



■ 登录到 M-EXECUTOR

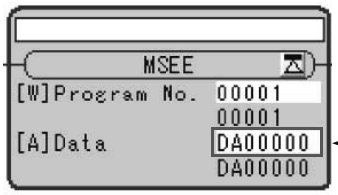


4.3.3 任务寄存器

通过 “4.3.2 程序的登录 ” 来分配运动程序运行控制 / 运行监视的任务寄存器。任务寄存器用于从运动程序控制用程序向运动程序发出指令或获取运动程序的状态。

■ 利用 MSEE 命令从梯形图程序调用的运动程序时

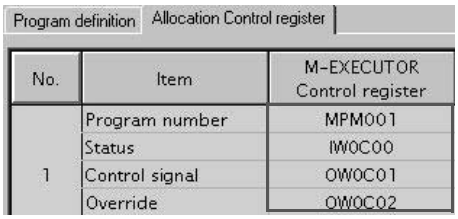
从 MSEE 命令的 Data 中指定的寄存器 （MAxxxx 或 DAxxxx）开始的 4 个字部分，被用作任务寄存器。



任务寄存器	左例	内容	输入输出
第1字	DW00000	状态标记	OUT
第2字	DW00001	控制信号	IN
第3字	DW00002	插补用超程	IN
第4字	DW00003	系统任务编号	IN

■ 已登录到 M-EXECUTOR 程序定义的运动程序时

M-EXECUTOR 控制寄存器被用作任务寄存器。
M-EXECUTOR 控制寄存器通过系统进行自动定义。



任务寄存器（M-EXECUTOR 控制寄存器）	左例	内容	输入输出
状态	IW0C00	状态标记	OUT
控制信号	OW0C01	控制信号	IN
超程	OW0C02	插补用超程	IN

下页开始对任务寄存器的详细情况进行介绍。

(1) 状态标记

Bit No		名称	内容
0～3	Bit 0	程序正在运行	运动程序启动后，该位的值将变为 1。 0：运动程序停止 1：运动程序正在运行
	Bit 1	程序暂停	响应程序暂停请求，运动程序暂停时，该位的值将变为 1。 0：非响应暂停请求的停止状态 1：响应暂停请求的停止状态
	Bit 2	程序停止请求停止状态	响应程序暂停请求，运动程序暂停时，该位的值将变为 1。 0：非响应停止请求的停止状态 1：响应停止请求的停止状态
	Bit 3	（系统预约）	
4～7	Bit 4	程序单段运行停止过程中	在调试运动中，单段停止时，该位的值将变为 1。 0：非单段停止状态 1：单段停止状态
	Bit 5	（系统预约）	
	Bit 6	（系统预约）	
	Bit 7	（系统预约）	
8～B	Bit 8	程序警报发生	发生程序警报时，该位的值将变为 1。 该位的值为 1 时，警报的详细信息会反应到错误信息画面及 S 寄存器。 0：未发生程序警报 1：发生程序警报
	Bit 9	因断点停止中	在调试运行中，因断点而停止时，该位的值将变为 1。 0：非断点停止状态 1：断点停止状态
	Bit A	（系统预约）	
	Bit B	调试模式	在通过调试运行程序时，该位的值将变为 1。 0：非调试模式运行状态（正常运行） 1：调试模式运行状态
C～F	Bit C	程序类别	报告正在运行的程序是运动程序还是顺序程序。 0：运动程序 1：顺序程序
	Bit D	开始请求历史记录	程序运行开始请求为 ON 时，该位的值将变为 1。 0：程序运行开始请求 OFF 1：程序运行开始请求 ON
	Bit E	无系统任务错误、扫描异常	无法获取运行运动程序所需的系统任务时，以及将 MSEE 命令加入 DWG.H 以外时，该位的值将变为 1。 0：未发生 “ 无系统任务错误、扫描异常 ” 1：发生 “ 无系统任务错误、扫描异常 ”
	Bit F	主程序编号溢出错误	指定的运动程序编号超出范围时，该位的值将变为 1。 运动程序编号范围：1 ～ 256 0：无运动程序编号指定异常 1：运动程序编号指定异常

(2) 控制信号

A接点 ：该标记的信号表示在系统接收到信号之前需一直保持 ON 状态。

微分 ：该标记的信号表示只要 1 次高速扫描 ON 即可。

Bit No		名称	内容
0～3	Bit 0 A接点 微分	程序运行开始请求	是启动运动程序的请求。该位发生 OFF → ON 变化时，运动程序启动。 但如果发生运动程序警报，则该位操作无效。 0：程序运行开始请求 OFF， 1：程序运行开始请求 ON
	Bit 1 A接点	程序暂停请求	使运行中的运动程序暂停的请求。 暂停后，如果解除暂停，则会从停止的程序开始重新运行。 0：程序暂停请求 OFF （暂停解除） 1：程序暂停请求 ON
	Bit 2 A接点	程序停止请求	使运行中的运动程序停止的请求。 如果在轴动作过程中启动了该位，则会发生运动程序警报。 0：程序停止请求 OFF，　　1：程序停止请求 ON
	Bit 3 A接点	程序单段模式选择	选择程序单段模式的请求。 可用于取代调试运行操作。 0：程序单段模式选择 OFF 1：程序单段模式选择 ON
4～7	Bit 4 A接点 微分	程序单段开始请求	该位发生 OFF → ON 变化时，程序会进行单段运行 （STEP 运行）。 该位仅在启用控制信号 “Bit3” （程序单段模式选择）时有效。 0：程序单段开始请求 OFF 1：程序单段开始请求 ON
	Bit 5 A接点	程序复位及警报复位请求	将运动程序及警报复位。 0：程序复位及警报复位请求 OFF 1：程序复位及警报复位请求 ON
	Bit 6 A接点 微分	程序继续运行开始请求	继续运行因程序停止请求而停止的程序的请求。 0：程序继续运行开始请求 OFF 1：程序继续运行开始请求 ON
	Bit 7	（系统预约）	
8～B	Bit 8 A接点	跳过 1 信息	在响应 SKP 命令 （向跳过输入信号选择发出 “SS1” 指令时）的轴移动过程中，如果接通该位，则正在移动的轴会减速停止，取消剩下的移动量指令。 0：跳过 1 信号 OFF，　　1：跳过 1 信号 ON
	Bit 9 A接点	跳过 2 信息	在响应 SKP 命令 （向跳过输入信号选择发出 “SS2” 指令时）的轴移动过程中，如果接通该位，则正在移动的轴会减速停止，取消剩下的移动量指令。 0：跳过 2 信号 OFF，　　1：跳过 2 信号 ON
	Bit A	（系统预约）	
	Bit B	（系统预约）	
C～F	Bit C	（系统预约）	
	Bit D A接点	系统任务编号设定	指定系统任务编号时，会接通该位。 0：不指定系统任务编号，　　1：指定系统任务编号
	Bit E A接点	插补用超程设定	指定插补超程时，会接通该位。 0：不指定插补用超程，　　1：指定插补用超程
	Bit F	（系统预约）	

(3) 插补用超程

内容
设定执行插补类命令 （MVS，MCW，MCC，SKP）时的超程值。 将针对响应插补类命令的轴移动的速度指令来更改输出比例称为插补用超程。 插补用超程仅在将控制信号 “BitE” （插补用超程设定）设为 ON 时有效。 • 插补超程值的指令范围：0 ～ 32767 • 单位：1=0.01%

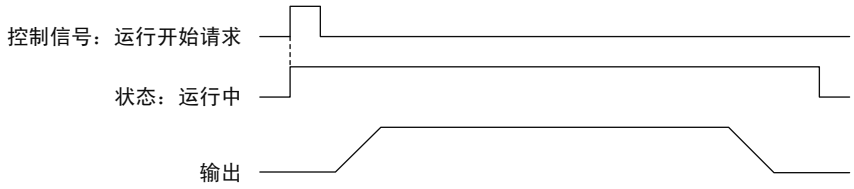
(4) 系统任务编号

内容
使用通过 MSEE 命令调用运动程序的方式时，设定调用运动程序的系统任务编号。该系统任务编号仅在控制信号 “BitD” （系统任务编号设定）设为 ON 时有效。 设定范围：1 ～ 16 （注）1. 使用 M-EXECUTOR 时，无法设定系统任务编号。使用与定义 No 一致的系统任务编号。 2. 同时使用梯形图 MSEE 命令和 M-EXECUTOR 时，请不要使用 MSEE 命令来指定 M-EXECUTOR 用的系统任务编号。否则会发生 “ 无系统任务错误 ”。 M-EXECUTOR 用的系统任务编号： 0 ～ 程序定义个数的设定值

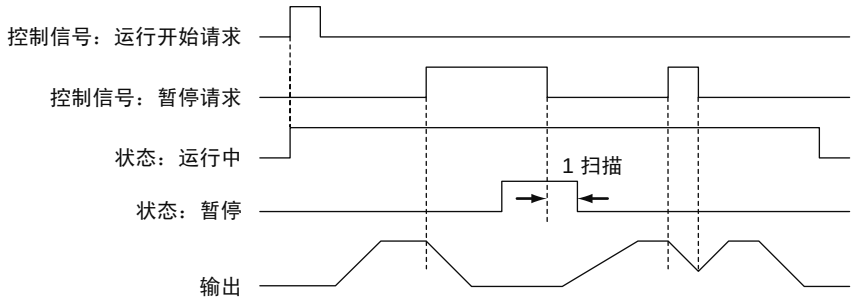
■ 运动程序控制信号的时序图

以下为运动程序控制信号的时序图的示例。

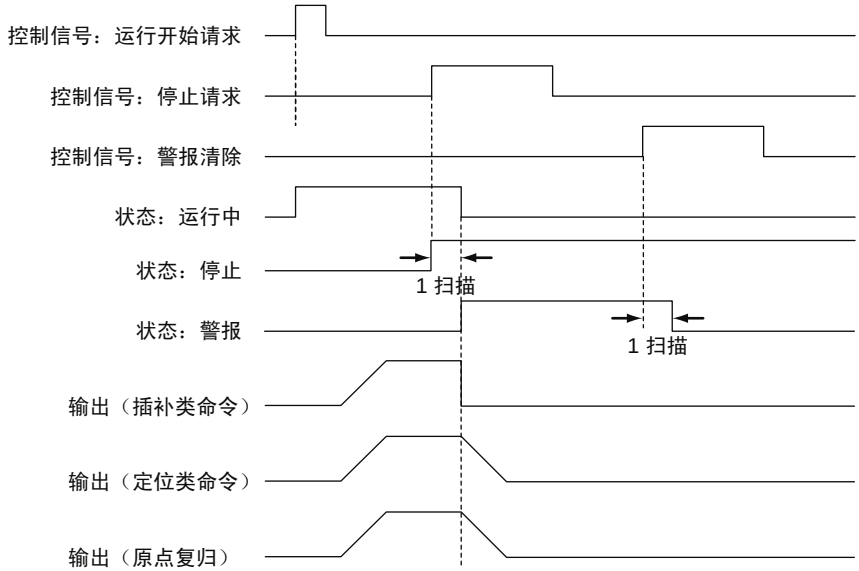
・程序运行开始请求时



・暂停请求时



・暂停请求时



重要

- 在轴移动过程中使用运动指令运行停止请求时，会发生警报。
- 在轴移动过程中使用插补类指令运行停止请求时，轴将停止动作。
  请使用暂停请求令轴减速停止。
- 执行原点复归 (ZRN) 时，暂停请求无效。
  请使用暂停请求令其停止。

关于运动控制程序例子，请参阅 “ 附录 B.1 运动程序控制用程序 ”。

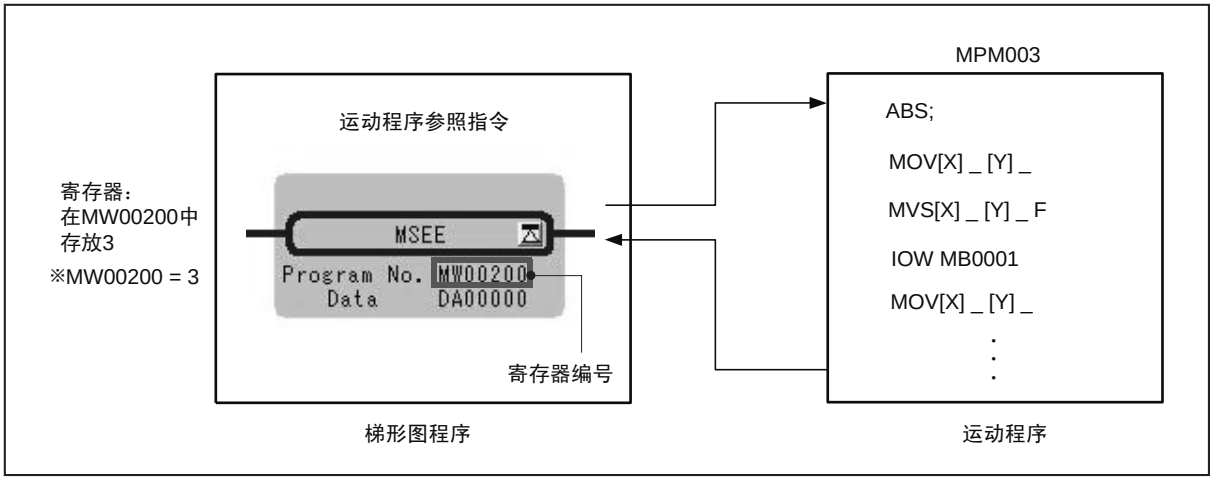
4.4 高级使用方法

4.4.1 使用寄存器间接指定程序编号

使用该方法会调用与寄存器中存储的值一致的程序 （MPM □□□）。

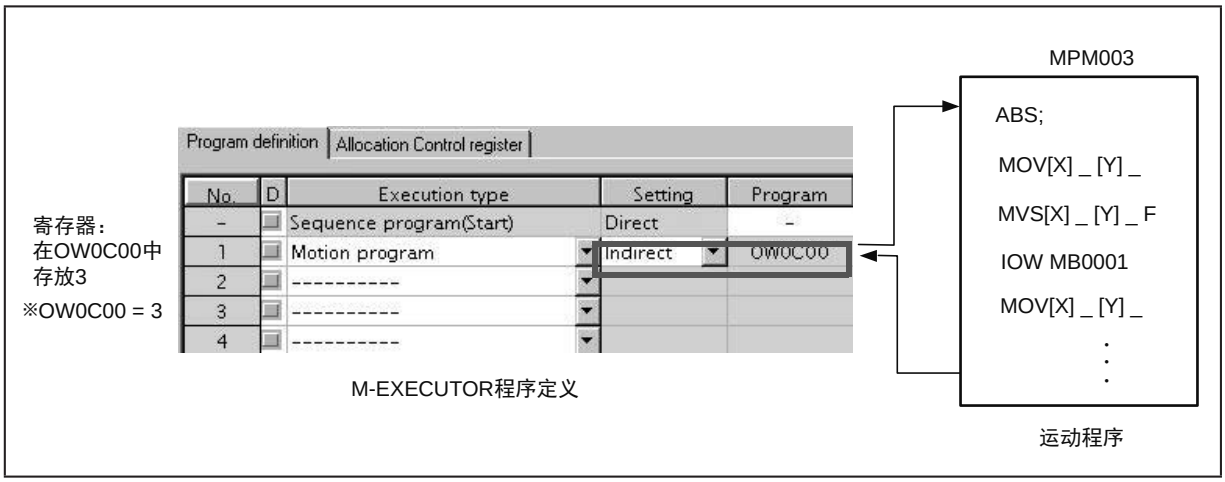
■ 利用 MSEE 命令从梯形图程序调用的运动程序时

在 MSEE 命令的 Program No. （程序编号）中指定间接指定使用的任意寄存器 （M 寄存器或 D 寄存器）。



■ 已登录到 M-EXECUTOR 程序定义的运动程序时

指定方法选择 “ 间接指定 ”。间接指定用寄存器由系统自动分配。



4.4.2 通过外部机器直接控制运动程序

M-EXECUTOR 模块能够将 M-EXECUTOR 的控制寄存器分配到任意寄存器。

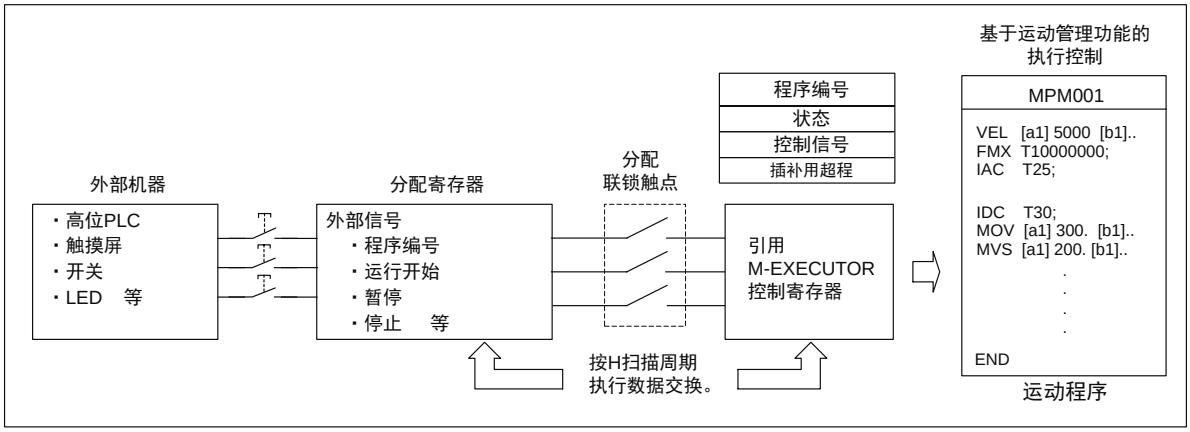
使用该功能能够在 M-EXECUTOR 的控制寄存器与连接到外部机器上的 I/O 寄存器之间自动进行数据交换。这样，外部机器就能够直接控制运动程序。

下图是使用本方法的设定实例。

M-EXECUTOR寄存器分配画面

No.	Item	M-EXECUTOR Control register	Allocation Disable	Direction	Allocation register	Allocation Contact interlock
1	Program number	MPM001				
	Status	IW0C00	☐	->	OW0000	IB00020
	Control signal	OW0C01	☐	<-	IW0000	IB00020
	Override	OW0C02	☐	<-	IW0001	IB00020

在分配寄存器、分配接口触点上
设定任意寄存器。



补充

■ 分配联锁触点将作为运动程序动作的联锁使用。在设定分配寄存器时，请务必对分配联锁触点进行设定。

根据分配联锁触点的 ON/OFF 状态，可执行以下处理。

- 当分配联锁触点为 ON 时，分配寄存器和 M-EXECUTOR 控制寄存器会按 H 扫描周期进行数据交换，此时，运动程序处于可运行状态。
- 当分配联锁触点为 OFF 时，分配寄存器和 M-EXECUTOR 控制寄存器不进行数据交换，此时，运动程序处于不可运行状态。
- 在运动程序运行过程中，当分配联锁触点进行 ON → OFF 切换时，正在运行的运动程序会停止，正在动作的轴会停止。此时会变成运动程序警报 “1Bh：正在执行紧急停止指令 ” 状态，同时，状态 “Bit8：程序警报发生 ” 变成 ON。

重新运行运动程序时，请按以下步骤操作。

①将联锁触点从 OFF 设定成 ON。
②接通控制信号 “Bit5：程序复位及警报复位请求 ”。
③确认状态 “Bit8：程序警报发生 ” 变成 OFF。
④关闭控制信号 “Bit5：程序复位及警报复位请求 ”。
⑤接通控制信号 “Bit0：程序运行开始请求 ”。

4.4.3 利用 S 寄存器监视运动程序运行

可以利用 S 寄存器（SW03200 ～ SW04191）来监视运动程序的运行信息。监视运行信息的方法，使用 MSEE 命令调用的运动程序和已登录到 M-EXECUTOR 程序定义中的运动程序会有所不同。
下面是不同情况时的监视方法。

(1) 利用 MSEE 命令从梯形图程序调用的运动程序

利用 MSEE 命令从梯形图程序调用的运动程序时，根据运动程序控制信号 “BitD” （系统任务编号设定）设定有以下不同。

- **运动程序控制信号 “BitD 系统任务编号设定 ” =ON 时**

  在 “ 任务 n 使用程序信息 ” 寄存器 （SW03264 ～ SW04191）中会报告运行信息。
  例如：当 “ 系统任务编号 ” =1 时，可利用 SW03264 ～ SW03321 “ 任务 1 使用程序信息 ” 来监视运动程序运行信息。

- **运动程序控制信号 “BitD 系统任务编号设定 ” =OFF 时**

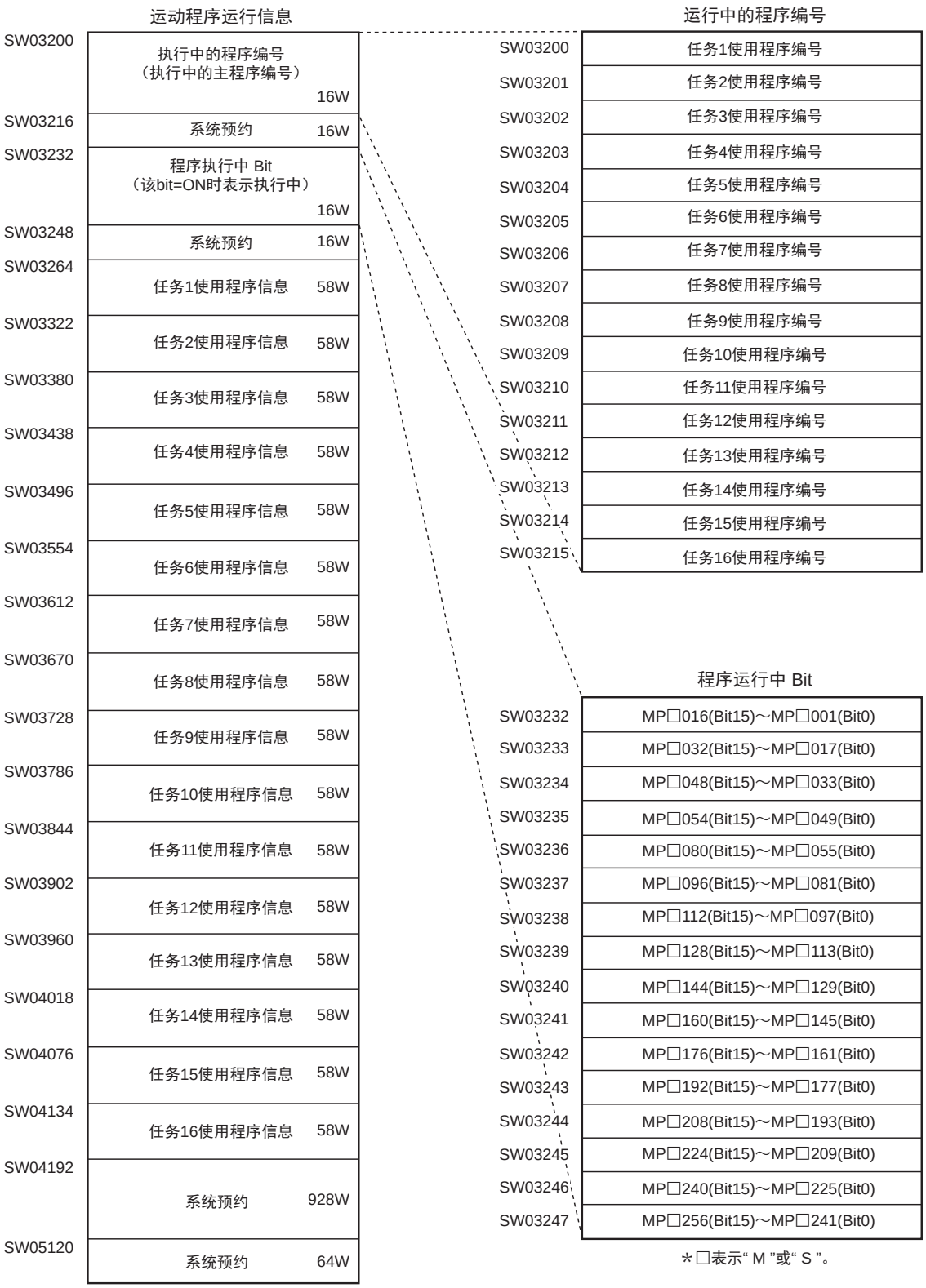
  所使用的系统任务将由系统自动决定。因此，可参照 “ 运行中程序编号 ” （=SW03200 ～ SW03215）来确认当前正在使用哪个任务。
  例如，要监视的运动程序为 MPM001，SW03202 为 1 时，使用的 “ 系统任务 ” =3。此时，可通过 “ 任务 3 使用程序信息 ”(=SW03380 ～ SW03437) 来监视运动程序的运行信息。

(2) 已登录到 M-EXECUTOR 程序定义的运动程序

已登录到 M-EXECUTOR 程序定义的运动程序时，使用的系统任务编号与已登录到 M-EXECUTOR 模块的程序登录 No 相同。
例如，将运动程序登录为 “ 程序登录 No” =3 时，使用的系统任务编号变成 “ 系统任务 ” =3。此时，可通过 “ 任务 3 使用程序信息 ”(=SW03380 ～ SW03437) 来监视运动程序的运行信息。

运动程序的运行信息中寄存器区域的详细信息，请参阅随后的内容。

・ 运动程序运行信息的寄存器区域



＊□表示“ M ”或“ S ”。

· 任务 n 使用程序信息的详细信息

任务n使用程序信息

+0	程序状态	
+1	程序控制信号	
+2	同时执行0信息	3W
+5	同时执行1信息	3W
+8	同时执行2信息	3W
+11	同时执行3信息	3W
+14	同时执行4信息	3W
+17	同时执行5信息	3W
+20	同时执行6信息	3W
+23	同时执行7信息	3W
+26	逻辑轴#1程序当前位置	2W
+28	逻辑轴#2程序当前位置	2W
+30	逻辑轴#3程序当前位置	2W
+32	逻辑轴#4程序当前位置	2W
+34	逻辑轴#5程序当前位置	2W
+36	逻辑轴#6程序当前位置	2W
+38	逻辑轴#7程序当前位置	2W
+40	逻辑轴#8程序当前位置	2W
+42	逻辑轴#9程序当前位置	2W
+44	逻辑轴#10程序当前位置	2W
+46	逻辑轴#11程序当前位置	2W
+48	逻辑轴#12程序当前位置	2W
+50	逻辑轴#13程序当前位置	2W
+52	逻辑轴#14程序当前位置	2W
+54	逻辑轴#15程序当前位置	2W
+56	逻辑轴#16程序当前位置	2W

运行中的程序编号
运行中的程序段编号
警报代码

补充 关于 S 寄存器一览，请参阅 “10.2.3 警报代码确认方法 (2) S 寄存器 ”。

5 章

顺序程序

本章对顺序程序的种类及运行方法进行说明。

5.1 顺序程序的种类

如下表中所示，顺序程序分为 2 种。

类别	指定方法	特征	程序数量
主程序	SPM □□□ （□□□ =1 ～ 256）	• 从 M-EXECUTOR 程序运行定义中调用	以下程序合计最多可创建 256 个 • 运动主程序 • 运动程序 • 顺序主程序 • 顺序程序
子程序	SPS □□□ （□□□ =1 ～ 256）	• 从主程序调用	

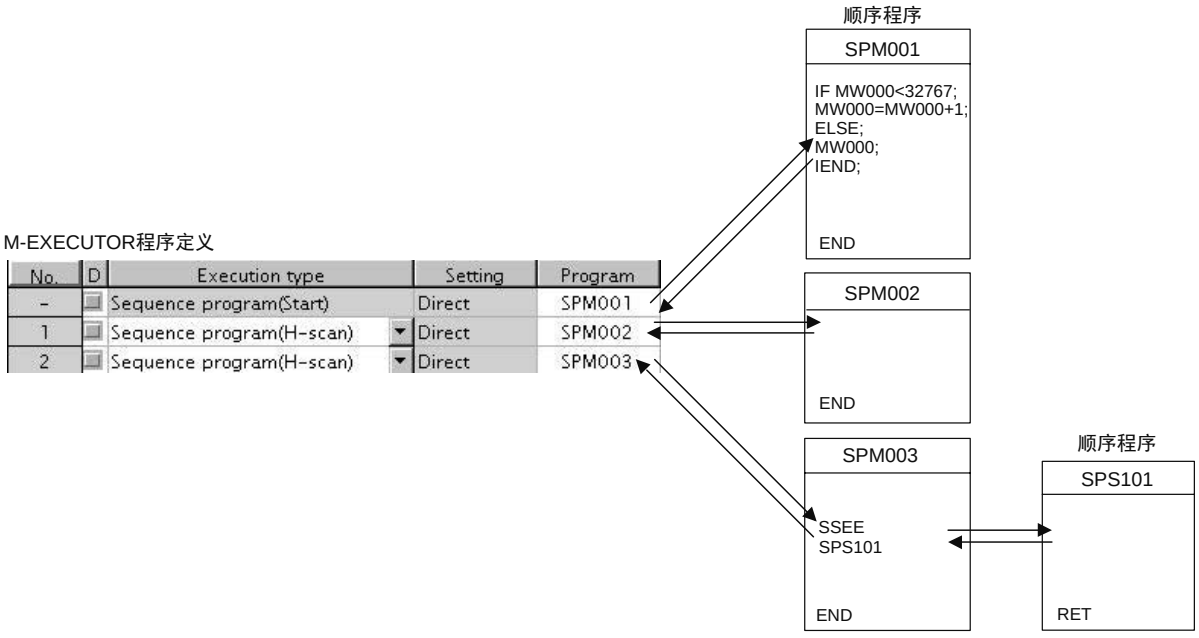
补充

■ 顺序程序与运动程序的程序编号统一进行管理。
程序编号请使用互不相同的编号。
- 运动程序 MPM □□□、MPS □□□的程序编号
- 顺序程序 SPM □□□、SPS □□□的程序编号

5.2 顺序程序的运行

5.2.1 顺序程序的运行方式

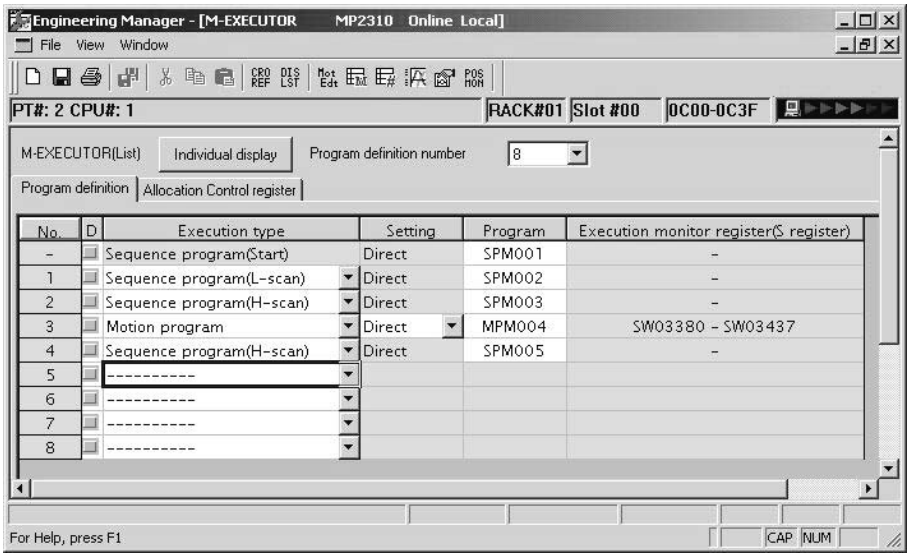
顺序程序通过登录到 M-EXECUTOR 程序定义运行。



运行类型设定成 “ 顺序程序 （H 扫描）” 或 “ 顺序程序 （L 扫描）时，在保存定义的时间点运行程序。运行类型设定成 “ 顺序程序 （启动）” 时，会在下次重新接通电源的时刻运行程序。

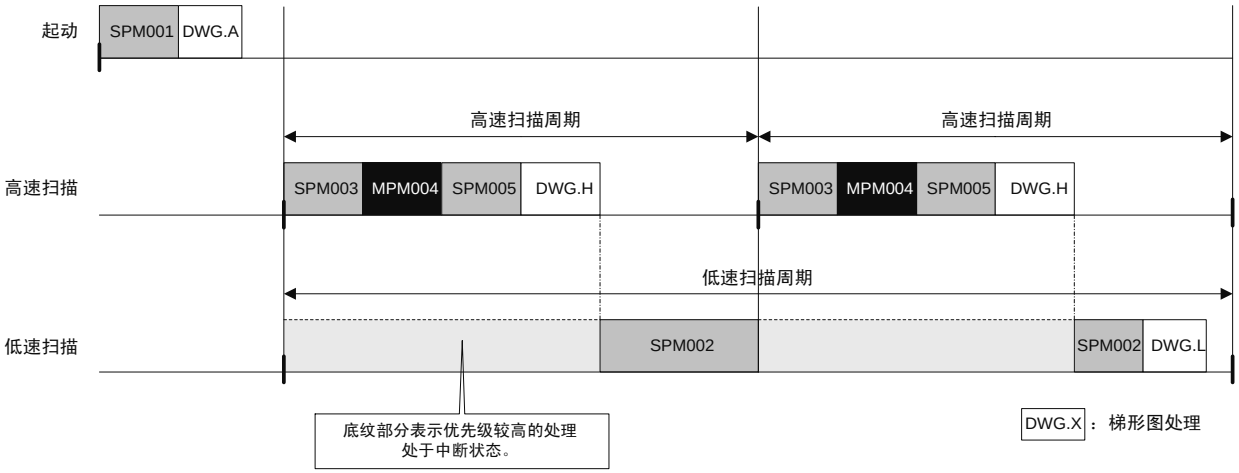
下面是顺序程序的运行示例。

■ M-EXECUTOR 程序定义



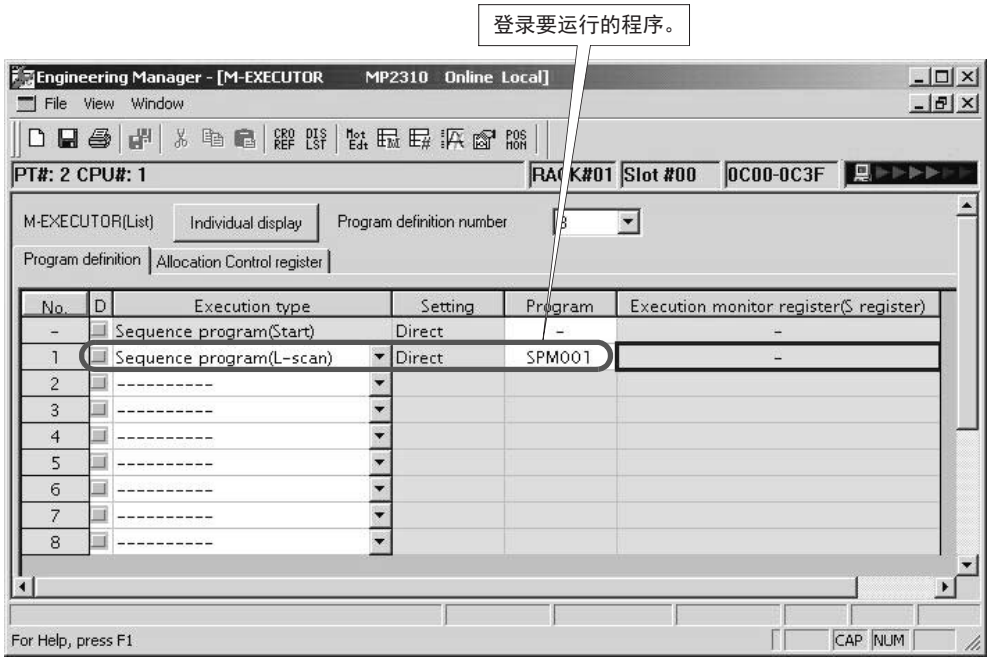
■ 定时运行

表示上述设定的运行时刻。
如下图所示，按程序定义的登录顺序依次运行。



5. 2. 2 程序的登录

登录程序的方法如下所示。以下是将 SPM001 登录到 H 扫描处理的示例。



补充

顺序程序的指定方法仅限直接指定。间接指定无法使用。

5.2.3 任务寄存器

通过 “5.2.2 程序的登录 ” 分配运行顺序程序状态监视的状态标记。顺序程序的状态标记可通过下面的公式计算。

IWxxxx + 4 ×（程序执行登录No. - 1）

└ M-EXECUTOR用输入输出起始寄存器编号

・输入输出起始寄存器编号可通过模块构成定义画面进行确认。

(1) 状态标记

Bit No		名称	内容
0 ～ 3	Bit 0	程序正在运行	顺序程序启动后，该位会 ON。 0：顺序程序停止，　1：顺序程序正在运行
	Bit 1	（系统预约）	
	Bit 2	（系统预约）	
	Bit 3	（系统预约）	
4 ～ 7	Bit 4	（系统预约）	
	Bit 5	（系统预约）	
	Bit 6	（系统预约）	
	Bit 7	（系统预约）	
8 ～ B	Bit 8	程序警报发生	引用顺序子程序 （执行 SSEE 命令时），如果发生了下面的异常，则该位会 ON。解除异常后，该位会 OFF。 • 调用目标程序未登录 • 调用程序非顺序程序 • 调用目标程序非子程序 （调用了主程序） • 调用目标程序编号溢出错误 • 嵌套溢出错误 0：未发生程序警报，　1：正在发生程序警报
	Bit 9	因断点而处于停止状态	在调试运行中，因断点而停止时，该位会 ON。 0：非断点停止状态，　1：断点停止状态
	Bit A	（系统预约）	
	Bit B	调试模式	在通过调试运行程序时，该位会 ON。 0：非调试模式运行状态 （正常运行），　1：调试模式运行状态
C ～ F	Bit C	程序类别	报告正在运行的程序是运动程序还是顺序程序。 0：运动程序，　1：顺序程序
	Bit D	开始请求历史记录	顺序程序启动后，该位会 ON。 0：顺序程序停止，　1：顺序程序正在运行
	Bit E	（系统预约）	
	Bit F	（系统预约）	

6章

变量（寄存器）

本章对运动程序及顺序程序中能够使用的“变量”的详细情况进行介绍。

6.1 变量

下面对变量的整体情况进行说明。

6.1.1 变量的概要

在运动程序及顺序程序中，可以不直接赋与 “ 数值 ”，而是使用 “ 变量 ” 进行预编程。实际使用 “ 变量 ” 时，应先提取出 “ 变量区域 ” 中存放的 “ 数值 ” 再使用。

(1) 变量 （寄存器）的类型

在运动程序及顺序程序中，可以将下表列出的 6 种寄存器作为变量使用。这些寄存器中，S、M、I、O、C 寄存器是所有程序都能够使用的通用全局变量。此外，各程序分别拥有独自的 D 寄存器，内存有其他程序无法使用的特有变量。

表 6.1 变量的类型

类型	名称	指定方法	范围	内容	特性
S	系统寄存器	SB、SW、SL、SFnnnnn	SW00000 ～ SW08191	系统中备有的寄存器。寄存器编号 nnnnn 采用十进制。	程序通用
M	数据寄存器	MB、MW、ML、MFnnnnn	MW00000 ～ MW65534	各程序之间通用的寄存器。使用程序之间的 I/F 等。寄存器编号 nnnnn 采用十进制。	
I	输入寄存器	IB、IW、IL、IFhhhh	IW0000 ～ IW7FFF	用于输入数据的寄存器。 寄存器编号 hhhh 采用十进制。寄存器编号为 8000 的寄存器被用作运动监视器参数。	
O	输出寄存器	OB、OW、OL、OFhhhh	OW0000 ～ OW7FFF	用于输出数据的寄存器。 寄存器编号 hhhh 采用十六进制。寄存器编号为 8000 的寄存器被用作运动设定参数。	
C	常数寄存器	CB、CW、CL、CFnnnnn	CW00000 ～ CW16383	只能通过程序来引用的寄存器。寄存器编号 nnnnn 采用十进制。	
D	D 寄存器	DB、DW、DL、DFnnnnn	DW00000 ～ DW16383	各程序特有的内部寄存器。只能用于相应程序。 实际可利用的范围由用户通过 MPE720 来指定。 寄存器编号 nnnnn 采用十进制。	程序个别

重要

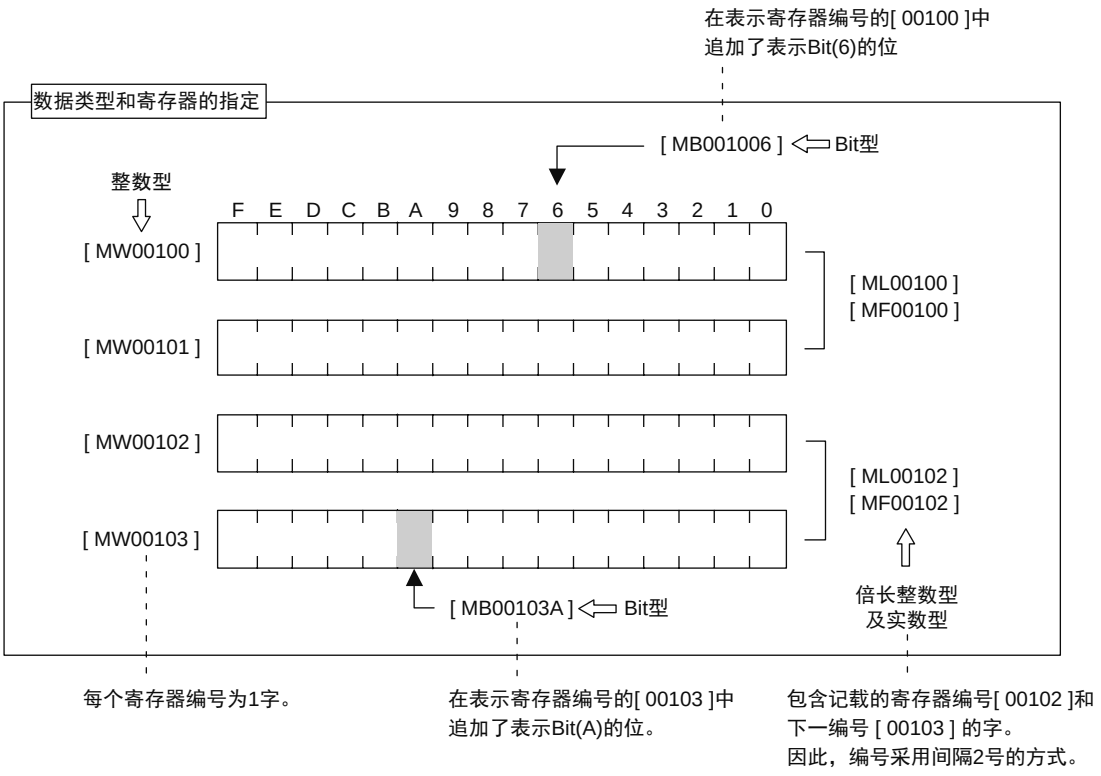
运动程序及顺序程序中，# 寄存器无法使用。否则，运行程序保存时会出现 “ 语法错误 ”。

(2) 数据类型

如下表所示，数据类型分为 Bit 型、整数型、倍长整数型、实数型，可根据目的分别使用。

表 6.2 数据类型

记号	数据类型	数值范围	备注
B	Bit	ON （1）、OFF （0）	用于继电器顺序及 ON/OFF 条件的判定
W	整数	-32768 ～ +32767 （8000H ～ 7FFFH）	用于数值运算。 （ ）内表示用于逻辑运算时的数值范围。
L	倍长整数	-2147483648 ～ +2147483647 （80000000H ～ 7FFFFFFFH）	用于数值运算。 （ ）内表示用于逻辑运算时的数值范围。
F	实数	± （1.175E-38 ～ 3.402E+38）	用于高级数值运算。



6.1.2 全局变量和局部变量

(1) 全局变量

全局变量是梯形图程序、用户函数、运动程序、顺序程序等各程序能够通用的变量。 即，某一梯形图程序运算的结果在其它的用户函数及运动程序、顺序程序中也能够使用。 全局变量的大小由系统按变量的不同进行分配 （参照下图）。

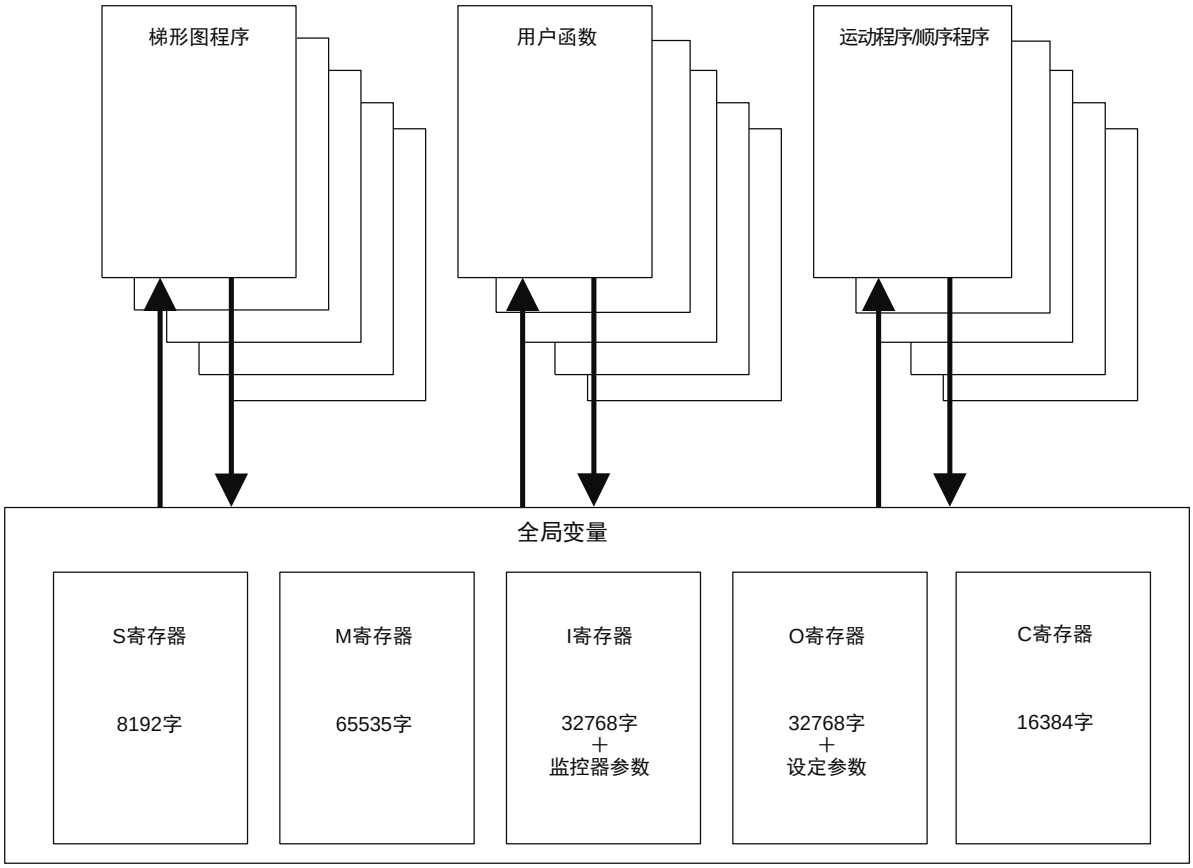
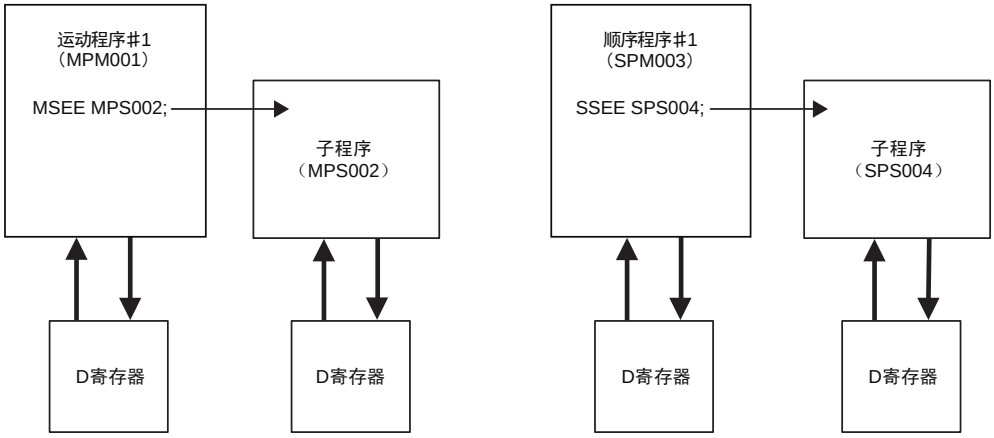


图 6.1 全局变量

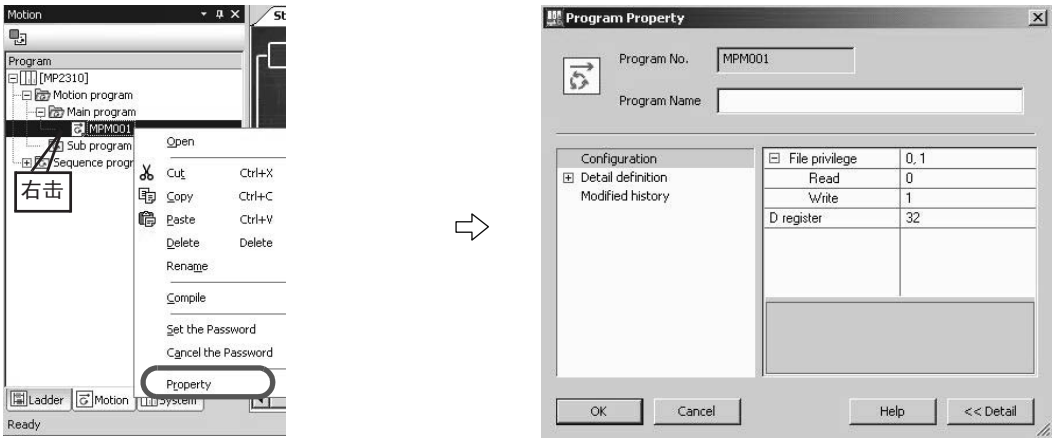
(2) 局部变量

局部变量是各个程序分别使用的变量。其它程序无法使用。局部变量由程序内存决定。

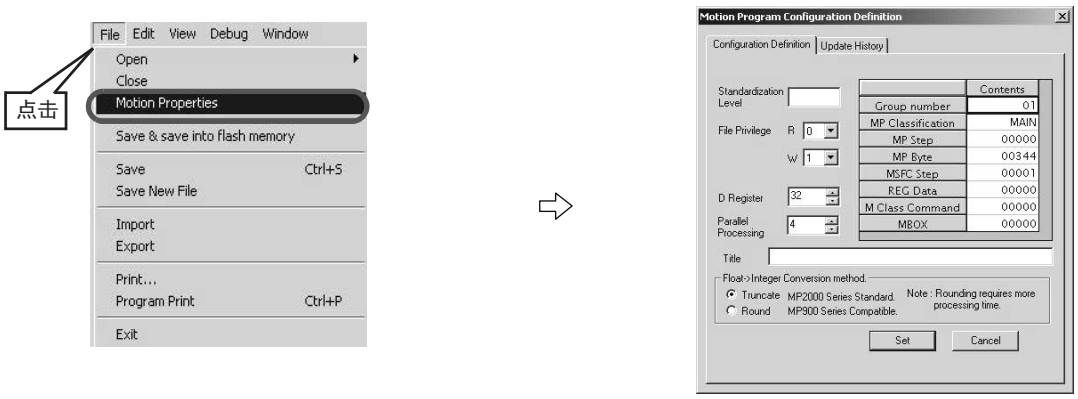


各程序使用的变量范围可通过 ***Program Property***（程序属性）窗口或 ***Motion Program Configuration Definition***（运动程序构成定义）画面进行指定。1 个图纸最多可使用 16384 字。

程序的属性画面



运动程序构成定义画面



补充

运动程序及顺序程序中，无法使用 # 寄存器。

重要

■ 变量运算时的注意事项

存放到数据类型不同的变量时，会得出以下结果。

格式	• 数值运算命令使用 =（代入）。 • 左边编写寄存器，右边编写运算公式。 MW00100 = MW00101 + MW00102;
变量运算	• 将实数型数据存放到整数型变量中 MW00100 = MF00200; 将实数值转换成整数，再进行存放。 （00001） （1.234） （注）将实数数据存放到整数型变量时，请注意舍入误差。 从实数舍入整数的方法（舍去 / 四舍五入）可通过 ***Program Property***（运动属性）设定。 MW00100 = MF00200 + MF00202; （0124） （123.48） （0.02） 要运算的变量的值不同，运算结果会有所不同。 （0123） （123.49） （0.01） • 将实数型数据存放到倍长整数型变量中 ML00100 = MF00200; 将实数值转换成整数，再进行存放。 （65432） （65432.1） • 将倍长整数型数据存放到整数型变量中 MW00100 = ML00200; 直接存放倍长整数型数据的低位 16 位。 （-00001） （65535） • 将整数型数据存放到倍长整数型变量中 ML00100 = MW00200; 将整数型数据转换成倍长整数型数据后进行存放。 （0001234） （1234）
发生语法错误的示例	• 将整数数据存放到比特型变量时 MB000100 = 123; => 语法错误 MB000100 = MW00100; => 语法错误

6.2 变量的使用方法

下面介绍各变量的使用方法。

6.2.1 系统变量 （S 寄存器）

(1) 概要

系统变量 （S 寄存器）是由机器控制器的系统提供的寄存器，可以读取系统的错误信息及运行状况等。
S 寄存器是运动程序及顺序程序通用的全局变量。详细情况，请参阅各机器控制器的用户手册。

(2) 详细说明

以下是 S 寄存器的指定方法。

SB000000	～	SB08191F
SW00000	～	SW08191
SL00000	～	SL08190
SF00000	～	SF08190

变量编号采用十进制进行指定。但是指定位时的 Bit 编号采用十六进制。

(3) 程序示例

例

• Bit 指定

```
OB00010 = SB000402 | SB000403;
```

• 整数指定

```
MW00100 = SW00041;
```

• 倍长整数指定

```
ML00100 = SL00062;
```

重要

系统寄存器 （S）专用于读取。写入时，无法保证系统正常工作。

6.2.2 数据变量 （M 寄存器）

(1) 概要

M 寄存器是在梯形图程序、用户函数以及各运动程序及顺序程序中能够通用的变量。同时也是可用于与运动程序、顺序程序及梯形图程序之间接口的全局变量。

(2) 详细说明

M 寄存器的指定方法如下所示。

MB000000	～	MB65534F
MW00000	～	MW65534
ML00000	～	ML65533
MF00000	～	MF65533

M 寄存器在各种运算中作为 “ 变量 ” 使用，可代入运算结果或使用 “ 变量 ” 指定定位的坐标值及速度。变量编号采用十进制进行指定。

(3) 程序示例

(a) 使用轴移动命令指定位置、使用变量指定速度时

例

```
• 参数  指令单位 ＝mm
        小数点位置＝3 时
  ML00100 ＝ 100000;          → 100.000 mm
  ML00102 ＝ 200000;          → 200.000 mm
  ML00104 ＝ 300000;          → 300.000 mm
  ML00106 ＝ 500000;          → 500.000 mm/min
  MVS [X]ML00100 [Y]ML00102 [Z]ML00104 FML00106;
```

(b) 运算中使用变量时

例

• 指定 Bit

```
MB001001 ＝ IB00100 & IB00201;
```

• 指定整数

```
MW00101 ＝ (MW00101 | MW00102) & FF0CH;
```

• 指定倍长整数

```
ML00200 ＝ ((MI00202 * ML00204) / ML00206) * 3;
```

• 指定实数

```
MF00200 ＝ ((MF00202 * MF00204) / MF00206) * 3.14;
```

重要

在以下运动命令中，使用变量指定移动量坐标值或速度时，务必使用倍长整数型。

MOV、MVS、MCW/MCC、ZRN、SKP、MVT、EXM、POS、
ACC、DCC、SCC、IAC、IDC、IFP、FMX、INP、VEL

6.2.3 输入变量 （I 寄存器）

(1) 概要

使用输入数据及运动监视器参数的变量。但是运动监视器参数仅用于读取。写入时，无法保证正常工作。

(2) 详细说明

I 寄存器的指定方法如下所示。

```
IW0000 ～ IW7FFF … 输入数据
IW8000 ～ IWFFFF … 运动监控器参数
```

(a) 输入数据的寄存器编号

基于使用模块构成定义设定的地址。

(b) 运动监视器参数寄存器编号

控制轴的数量取决于模块类型。不同模块的控制轴数及最大模块数如下表所示。

表 6.3 模块的控制轴数

运动模块	每个模块的控制轴数	最大模块数									
		MP2100	MP2100M	MP2200	MP2300	MP2300S	MP2310	MP2400	MP2500(D)	MP2500M	MP2500MD
MP2000 系列 内置 SVB	16 （最大）*1	1	1	—	1	1	1	1	1	1	1
MP2100M/MP2500M SVB-01	16 （最大）*1	—	1	—	—	—	—	—	—	1	1
选购模块 SVB-01	16 （最大）*1	—	14 *2 *3	16 *3	2	1	3	—	—	—	14 *2 *3
选购模块 SVA-01	2										
选购模块 PO-01	4										

*1. 内置 SVB/SVB-01 的每个模块的控制轴数取决于 MECHATROLINK 定义。
*2. MP2100M，MP2500MD 仅在构建扩展搁架构成时才能够使用选购模块 (SVB-01，SVA-01，PO-01)。
*3. MP2100M，MP2200，MP2500MD 中选购模块 (SVB-01，SVA-01，PO-01) 的最大模块数为构建扩展搁架构成时的模块数。

表 6.4 运动参数寄存器编号

轴编号 线路编号	1 轴	2 轴	3 轴	4 轴	5 轴	・・	16 轴
1	8000 ～ 807F	8080 ～ 80FF	8100 ～ 817F	8180 ～ 81FF	8200 ～ 827F	・・	8780 ～ 87FF
2	8800 ～ 887F	8880 ～ 88FF	8900 ～ 897F	8980 ～ 89FF	8A00 ～ 8A7F	・・	8F80 ～ 8FFF
3	9000 ～ 907F	9080 ～ 90FF	9100 ～ 917F	9180 ～ 91FF	9200 ～ 9A7F	・・	9780 ～ 97FF
4	9800 ～ 987F	9880 ～ 98FF	9900 ～ 997F	9980 ～ 99FF	9A00 ～ 997F	・・	9F80 ～ 9FFF
5	A000 ～ A07F	A080 ～ A0FF	A100 ～ A17F	A180 ～ A1FF	A200 ～ A27F	・・	A780 ～ A7FF
6	A800 ～ A87F	A880 ～ A8FF	A800 ～ A87F	A980 ～ A9FF	AA00～AA7F	・・	AF80 ～ AFFF
7	B000 ～ B07F	B080 ～ B0FF	B100 ～ B17F	B180 ～ B1FF	B200 ～ B27F	・・	B780 ～ B7FF
8	B800 ～ B87F	B880 ～ B8FF	B900 ～ B97F	B980 ～ B9FF	BA00～BA7F	・・	BF80 ～ BFFF
9	C000 ～ C07F	C080 ～ C0FF	C100 ～ C17F	C180 ～ C1FF	C200 ～ C27F	・・	C780 ～ C7FF
10	C800 ～ C87F	C880 ～ C8FF	C900 ～ C97F	C980 ～ C9FF	CA00～CA7F	・・	CF80 ～ CFFF
11	D000 ～ D07F	D080 ～ D0FF	D100 ～ D17F	D180 ～ D1FF	D200 ～ D27F	・・	D780 ～ D7FF
12	D800 ～ D87F	D880 ～ D8FF	D900 ～ D97F	D980 ～ D9FF	DA00～DA7F	・・	DF80 ～ DFFF
13	E000 ～ E07F	E080 ～ E0FF	E100 ～ E17F	E180 ～ E1FF	E200 ～ E27F	・・	E780 ～ E7FF
14	E800 ～ E87F	E880 ～ E8FF	E900 ～ E97F	E980 ～ E9FF	EA00 ～ A97F	・・	EF80 ～ EFFF
15	F000 ～ F07F	F080 ～ F0FF	F100 ～ F17F	F180 ～ F1FF	F200 ～ F27F	・・	F780 ～ EFFF
16	F800 ～ F87F	F880 ～ F8FF	F900 ～ F97F	F980 ～ F9FF	F900 ～ F97F	・・	FF80 ～ FFFF

↑
模块编号偏置

补充

运动监视器参数的各轴寄存器编号可通过以下公式进行计算。

运动监视器参数寄存器起始编号 = IW8000 +（线路编号 -1）×800h +（轴编号 -1）×80h

(3) 程序示例

读取输入数据及运动监视器参数。

例

• Bit 指定

```
MB001000 = IB00010 & IB00105;
```

• 整数指定

```
MW00100 = IW8008;
```

• 倍长整数指定

```
ML00100 = IL8004;
```

6.2.4 输出变量 （0 寄存器）

(1) 概要

使用输出数据及运动设定参数的变量。

(2) 详细说明

0 寄存器的指定方法如下所示。

```
OW0000 ～ OW7FFF … 输出数据
OW8000 ～ OWFFFF … 运动设定参数
```

(a) 输出数据的寄存器编号

基于使用模块构成定义设定的地址。

(b) 运动设定参数寄存器编号

控制轴的数量取决于模块类型。不同模块的控制轴数及最大模块数如下表所示。

表 6.5 模块的控制轴数

运动模块	每个模块的控制轴数	最大模块数									
		MP2100	MP2100M	MP2200	MP2300	MP2300S	MP2310	MP2400	MP2500(D)	MP2500M	MP2500MD
MP2000 系列 内置 SVB	16 （最大）*1	1	1	—	1	1	1	1	1	1	1
MP2100M/MP2500M SVB-01	16 （最大）*1	—	1	—	—	—	—	—	—	1	1
选购模块 SVB-01	16 （最大）*1	—	14 *2 *3	16 *3	2	1	3	—	—	—	14 *2 *3
选购模块 SVA-01	2										
选购模块 PO-01	4										

*1. 内置 SVB/SVB-01 的每个模块的控制轴数取决于 MECHATROLINK 定义。
*2. MP2100M，MP2500MD 仅在构建扩展搁架构成时才能够使用选购模块 (SVB-01，SVA-01，PO-01)。
*3. MP2100M，MP2200，MP2500MD 中选购模块 (SVB-01，SVA-01，PO-01) 的最大模块数为构建扩展搁架构成时的模块数。

表 6.6 运动参数寄存器编号

轴编号 线路编号	1 轴	2 轴	3 轴	4 轴	5 轴	・・	16 轴
1	8000 ～ 807F	8080 ～ 80FF	8100 ～ 817F	8180 ～ 81FF	8200 ～ 827F	・・	8780 ～ 87FF
2	8800 ～ 887F	8880 ～ 88FF	8900 ～ 897F	8980 ～ 89FF	8A00 ～ 8A7F	・・	8F80 ～ 8FFF
3	9000 ～ 907F	9080 ～ 90FF	9100 ～ 917F	9180 ～ 91FF	9200 ～ 9A7F	・・	9780 ～ 97FF
4	9800 ～ 987F	9880 ～ 98FF	9900 ～ 997F	9980 ～ 99FF	9A00 ～ 997F	・・	9F80 ～ 9FFF
5	A000 ～ A07F	A080 ～ A0FF	A100 ～ A17F	A180 ～ A1FF	A200 ～ A27F	・・	A780 ～ A7FF
6	A800 ～ A87F	A880 ～ A8FF	A800 ～ A87F	A980 ～ A9FF	AA00～AA7F	・・	AF80 ～ AFFF
7	B000 ～ B07F	B080 ～ B0FF	B100 ～ B17F	B180 ～ B1FF	B200 ～ B27F	・・	B780 ～ B7FF
8	B800 ～ B87F	B880 ～ B8FF	B900 ～ B97F	B980 ～ B9FF	BA00～BA7F	・・	BF80 ～ BFFF
9	C000 ～ C07F	C080 ～ C0FF	C100 ～ C17F	C180 ～ C1FF	C200 ～ C27F	・・	C780 ～ C7FF
10	C800 ～ C87F	C880 ～ C8FF	C900 ～ C97F	C980 ～ C9FF	CA00～CA7F	・・	CF80 ～ CFFF
11	D000 ～ D07F	D080 ～ D0FF	D100 ～ D17F	D180 ～ D1FF	D200 ～ D27F	・・	D780 ～ D7FF
12	D800 ～ D87F	D880 ～ D8FF	D900 ～ D97F	D980 ～ D9FF	DA00～DA7F	・・	DF80 ～ DFFF
13	E000 ～ E07F	E080 ～ E0FF	E100 ～ E17F	E180 ～ E1FF	E200 ～ E27F	・・	E780 ～ E7FF
14	E800 ～ E87F	E880 ～ E8FF	E900 ～ E97F	E980 ～ E9FF	EA00 ～ A97F	・・	EF80 ～ EFFF
15	F000 ～ F07F	F080 ～ F0FF	F100 ～ F17F	F180 ～ F1FF	F200 ～ F27F	・・	F780 ～ EFFF
16	F800 ～ F87F	F880 ～ F8FF	F900 ～ F97F	F980 ～ F9FF	F900 ～ F97F	・・	FF80 ～ FFFF

↑
模块编号偏置

补充

运动设定参数的各轴寄存器编号可通过以下公式进行计算。

运动设定参数寄存器起始编号 =OW8000 +（线路编号 -1）×800h +（轴编号 -1）×80h

(3) 程序示例

写入输出数据及运动设定参数。

例

• Bit 指定

```
OB01000 = MB001000 & IB00100;
```

• 整数指定

```
OW8008 = MW00100;
```

• 倍长整数指定

```
OL8010 = ML00100 + ML00200;
```

6.2.5 C变量 （C寄存器）

(1) 概要

C寄存器是通过程序进行引用的变量。不能进行写入。

(2) 详细说明

C寄存器的指定方法如下所示。

CW00000 ～ CW16383

C寄存器不能通过程序写入。

(3) 程序示例

运算中使用变量时

例

- 指定Bit

```
MB001000 = CB001001;
```

- 指定整数

```
MW00100 = CW00100;
```

- 指定倍长整数

```
ML00100 = CL00100;
```

- 指定实数

```
MF00100 = CF00100;
```

6.2.6 D 变量 （D 寄存器）

(1) 概要

作为各运动程序及顺序程序特有的内部寄存器，是仅限相关程序才能使用的变量。

(2) 详细说明

D 寄存器的指定方法如下所示。

DW00000 ～ DW16383 （最大）

以上变量在各种运算中作为 “ 变量 ” 使用，可代入运算结果或使用 “ 变量 ” 指定定位的坐标值及速度。变量编号采用十进制进行指定。大小可通过程序构成定义 （运动属性）设定 （默认 32 字）。

(3) 程序示例

(a) 使用轴移动命令指定位置、使用变量指定速度时

例

```
• 参数  指令单位＝ mm
        小数点位置＝ 3 时
    DL00100 ＝ 100000;          → 100.000 mm
    DL00102 ＝ 200000;          → 200.000 mm
    DL00104 ＝ 300000;          → 300.000 mm
    DL00106 ＝ 500000;          → 500.000 mm/min
    MVS [A1]DL00100 [B1]DL00102 [C1]DL00104 FDL00106;
```

(b) 运算中使用变量时

例

• Bit 指定

```
DB001000 ＝ IB01001 & MB000101;
```

• 整数指定

```
DW00102 ＝ ( CW00103 | DW00104 ) & DW00105;
```

• 倍长整数指定

```
DL00106 ＝ ( DL00108 * ML00011 ) / ML00200;
```

• 实数指定

```
DF00200 ＝ ( MF00202 * DF00202 ) * 3.14;
```

重要

在以下运动命令中，使用变量指定移动量坐标值或速度时，务必使用倍长整数型。

MOV、MVS、MCW/MCC、ZRN、SKP、MVT、EXM、POS、
ACC、DCC、SCC、IAC、IDC、IFP、FMX、INP、VEL

6.3 附加字符 i、j 的使用方法

作为修饰继电器编号和寄存器编号的专用寄存器，备有 “i” 和 “j” 2 种可供选择。i 和 j 的功能完全相同。在希望将寄存器编号作为变量进行处理时，可予以使用。

下面就按寄存器的数据类型分别举例说明。

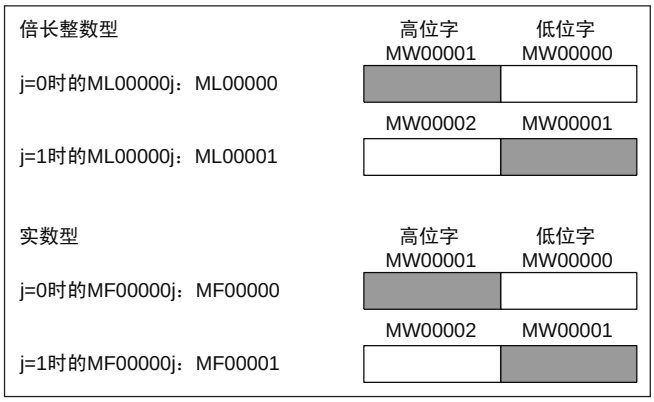
(1) 带有附加字符的 Bit 型数据

与在寄存器编号上加上 i 或 j 的值时相同。
例如，i=2 时的 MB000000i 与 MB000002 相同。此外，j=27 时的 MB000000j 与 MB00001B 相同。

(2) 带有附加字符的整数型数据

与在寄存器编号上加上 i 或 j 的值时相同。
例如：i=3 时的 MW00010i 与 MW00013 相同。另外，j=30 时的 MW00001j 与 MW00031 相同。

(3) 带有附加字符的倍长整数型或实数型数据



与在寄存器编号上加上 i 或 j 的值时相同。
例如：j=1 时的 ML00000j 与 ML00001 相同。此外，j=1 时的 MF00000j 与 MF00001 相同。
倍长整数型、实数型时，使用 （寄存器编号）的 1 字区域和 （寄存器编号 +1）的 1 字区域。倍长整数型或实数型数据带有附加字符时，请注意区域的重复情况。例如，j=0 时的 ML00000j 和 j=1 时的 ML00000j 中，MW00001 的 1 字区域重复。

以下是使用了附加字符的程序示例。

例

```
    :
    :
ML00200 = 0 ;
J = 0 ;
WHILE J < 100 ;
    ML00200 = ML00200 + ML00100j ;
    J = J + 2 ;
WEND ;
    :
    :
```

左边的程序是使用了附加字符 j，要求 ML00200 对从 ML00100 到 ML00198 为止的 50 个寄存器进行合计。

补充

- 使用附加字符 i、j 需使用以下版本的软件工具。

MP2000 系列控制器	支持版本
全部机型	Ver2.63 以上

MPE720	支持版本
MPE720 Ver.5	MPE720 Ver.5.42 以上
MPE720 Ver.6	MPE720 Ver.6.08 以上 MPE720 Ver.6.08 Lite 以上

- 附加字符 i，j 不区分大小写。

```
i = 0;
I = 0;
DW00000 = MW00000i ;
DW00000 = MW00000I ;
```

7 章

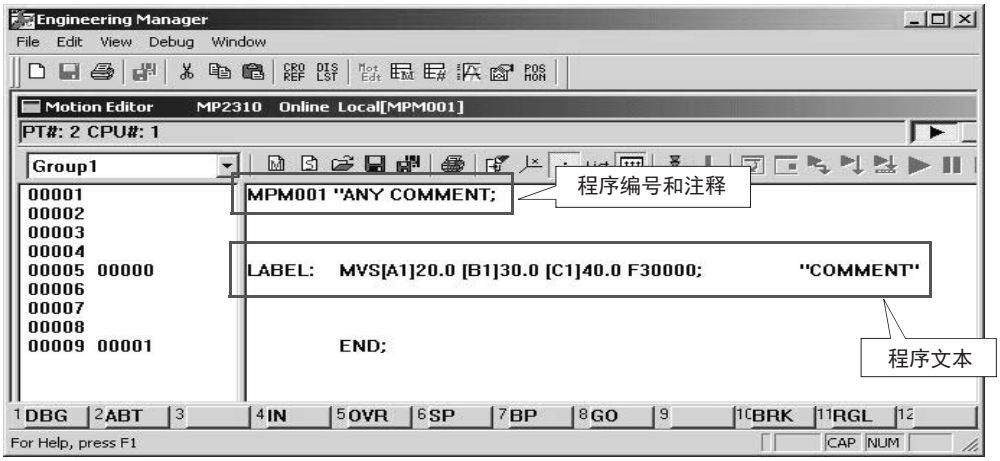
编程

本章对运动程序及顺序程序的编写规则进行说明。

7.1 运动程序的格式

7.1.1 运动程序的编写构成

运动程序由程序编号、任意注释、程序文本、END 命令构成。在程序文本中编写运动程序的处理。运动程序的编写构成图如下所示。

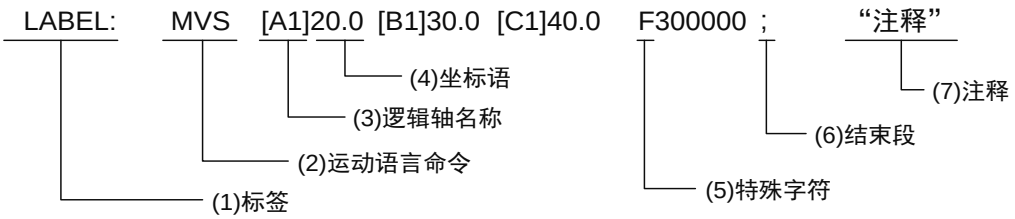


补充

“ 程序编号和注释 ” 行未必需要进行指定。

7.1.2 程序段的格式

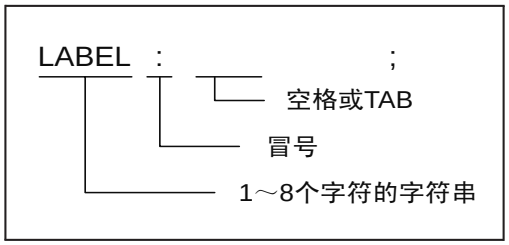
程序段是执行处理的单位。编写 1 个以上的程序段，构成程序文本。运动程序的程序段格式如下所示。各项目的详细信息，请参阅下页以后的内容。



No.	项目	含义
(1)	标签	表示 PFROK 命令、SFORK 命令的分支目标。
(2)	运动语言命令	指定运动程序的命令。
(3)	逻辑轴名称	指定群组定义设定的逻辑轴名称。
(4)	坐标语	指定轴的坐标值或轴的增量移动量等。
(5)	特殊字符	指定运动命令的辅助数据。
(6)	段结束	指定程序段的结束。
(7)	注释	编写编程的注释。

(1) 标签

标签由使用英文数字、符号的 1 ～ 8 个字符的字符串和冒号 “:”、空格或 TAB 构成。



类型	可用于标签的英文数字、记号
英文字母	A ～ Z、a ～ z
数字	0 ～ 9
记号	_（连字符）

（注）标签开关不能使用数字。

使用同时执行 （PFORK）、选择执行 （SFORK）时，需编写标签。
不使用同时执行 （PFORK）、选择执行 （SFORK）时，无需编写标签。

标签的编写示例

例

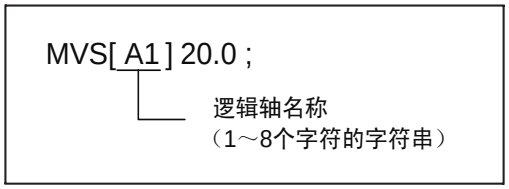
```
PFORK LAB1, LAB2;
LAB1: ZRN [A1]0 [B1]0 [C1]0;
JOINTO LAB3;
LAB2: MVS [D1]100.0 [E1]200.0 [F1]300.0;
JOINTO LAB3;
LAB3: PJOINT;
```

(2) 运动语言命令

编写运动语言命令。
关于运动语言命令，请参阅 “8 章 相关命令 ” 或 “ 附录 A 运动语言命令 ”。

(3) 逻辑轴名称

通过群组定义设定的逻辑轴名称编写时用括弧 （[ ]）括起来。



类型	逻辑轴名称中能够使用的字符
英文字母	A ～ Z、a ～ z
数字	0 ～ 9

(4) 坐标语

坐标语是接在轴名称后面的数值及变量。指定指令位置及速度、加减速度等参数。

■ 使用数字编写坐标语。

在轴名称的后面直接指定数值。

数值可指定为整数或实数，但指定为整数时需引起注意。

指令单位设定为 0.001mm 时，例如，在坐标语中输入指令位置 “1000” 时，在控制器内部会被解释为 1.000mm。使用实数输入 “1.000” 的话，则直接解释为 1.000mm。

例

```
MVS [A1]1000;        → 1.000 mm
或
MVS [A1]1.000;       → 1.000 mm
或
MVS [A1]1.;          → 1.000 mm
```

■ 使用变量编写坐标语。

在轴名称的后面使用倍长整数型变量进行间接指定。

在寄存器的间接指定中，指令单位设为 0.001mm，如果将寄存器值设定为 1000，即与上面例子中输入整数值相同，则在控制器内部会被译为 1.000mm。

例

```
ML00000 =1000;
MVS [A1]ML00000;     → 1.000 mm
```

补充

坐标语的单位因命令及运动模块的设定而异。关于坐标语的单位，请参阅 “7.2.2 指令单位 ”。

(5) 特殊字符

以下是各特殊字符的意思及使用示例一览。各特殊字符的详细信息，请查看参考页。

字符	含义	使用示例	参考页
F	插补进给速度	MVS [A1]1000 [B1]2000 ***F***3000000; MVS [A1]1000 [B1]2000 ***F***ML00000;	8.2.2 直线插补 （MVS）
T	插补进给最高速度	FMX ***T***30000000; FMX ***T***ML00000;	8.1.7 插补进给最高速度设定 （FMX）
	时间设定值	TIM ***T***100; TIM ***T***MW00000; MVT [A1]1000 [B1]2000 ***T***100; MVT [A1]1000 [B1]2000 ***T***ML00000; IAC ***T***100; IAC ***T***ML00000; IDC ***T***100; IDC ***T***ML00000;	8.4.11 时间等待 （TIM） 8.2.9 指定时间定位 （MVT） 8.1.9 插补加速时间变更 （IAC） 8.1.10 插补减速时间变更 （IDC）
	圆弧转数	MCW [A1]1000 [B1]2000 U500 V500 ***T***2 F3000000; MCW [A1]1000 [B1]2000 U500 V500 ***T***ML00000 F3000000;	8.2.3 圆弧插补＜指定中心位置＞（MCW、MCC） 8.2.4 圆弧插补＜指定半径＞（MCW、MCC）
R	圆弧半径	MCW [A1]1000 [B1]2000 ***R***500 F3000000; MCW [A1]1000 [B1]2000 ***R***ML00000 F3000000;	8.2.3 圆弧插补＜指定中心位置＞（MCW、MCC） 8.2.4 圆弧插补＜指定半径＞（MCW、MCC）
U	圆弧中心坐标1（横轴）	MCW [A1]1000 [B1]2000 ***U***500 V500 T2 F3000000; MCW [A1]1000 [B1]2000 ***U***ML00000 V500 T2 F3000000;	8.2.3 圆弧插补＜指定中心位置＞（MCW、MCC） 8.2.4 圆弧插补＜指定半径＞（MCW、MCC）
V	圆弧中心坐标2（纵轴）	MCW [A1]1000 [B1]2000 U500 ***V***500 T2 F3000000; MCW [A1]1000 [B1]2000 U500 ***V***ML00000 T2 F3000000;	8.2.3 圆弧插补＜指定中心位置＞（MCW、MCC） 8.2.4 圆弧插补＜指定半径＞（MCW、MCC）
P	基于比率的插补进给速度	IFP ***P***50; IFP ***P***ML00000;	8.1.8 插补进给速度比率设定 （IFP）
SS	跳过信号的选择	SKP [A1]1000 [B1]2000 F3000000 ***SS***1; SKP [A1]1000 [B1]2000 F3000000 ***SS***2;	8.2.8 带跳过功能的直线插补 （SKP）
D	外部定位移动量	EXM [A1]1000 ***D***1000; EXM [A1]1000 ***D***ML00000;	8.2.10 外部定位 （EXM）
N	移动数	SFR MB001000 ***N***5 W10; SFR MB001000 ***N***MW00000 W10;	8.8.1 位右移 （SFR） 8.8.2 位左移 （SFL）
W	bit 宽度	BLK MW00100 DW00100 ***W***10; BLK MW00100 DW00100 ***W***MW00000;	8.8.1 位右移 （SFR） 8.8.2 位左移 （SFL） 8.8.3 段传输 （BLK） 8.8.4 清除 （CLR）
MPS	运动子程序编号	MSEE ***MPS***002;	8.4.5 运动子程序调用 （MSEE）
SPS	顺序子程序编号	SSEE ***SPS***002;	8.4.6 调用顺序子程序 （SSEE）

(6) 结束段

在程序段结束处编写分号 （;）。程序段可跨多行。编写结束段，指定程序段的结束。
请务必在编写结束的段后面换行。

结束段的编写示例

例

```
                                          换行字符
MOV   [A1]1000;          “轴A1移动”        ↵
            程序段的结束代码
                                          换行字符
MOV   [A1]1000           “轴A1移动”        ↵
      [B1]2000           “轴B1移动”        ↵
      [C1]3000;          “轴C1移动”        ↵
            程序段的结束代码
```

(7) 注释

程序段的格式有如下 2 种。

■ 在由 2 个双引号括起来的位置编写字符串。

由双引号括起来的字符串被称为注释。

```
“ 字符串 ”
```

注释的编写示例 1

例

```
ZRN [A1]0 [B1]0 [C1]0;                    “ 全轴的原点复归 ”
MVS [A1]100.0 [B1]200.0 [C1]300.0;        “3 轴直线插补 ”
```

■ 在 1 个双引号后面继续编写字符串 （注释）。

从一个双引号到后面换行 （Enter）之前的字符全部被称为注释。

```
                换行字符
“ 字符串        ↵
```

注释的编写示例 2

例

```
 “ 全轴的原点复归
ZRN [A1]0 [B1]0 [C1]0;

 “3 轴直线插补
MVS [A1]100.0 [B1]200.0 [C1]300.0;
```

补充 字符串 （注释）除使用半角英文数字外，还可使用包括汉字在内的全角字符。

7.1.3 常数与变量的表述

(1) 常数的表述

运动程序能够使用的常数如下所示。

类别	范围	编写示例
十进制整数	-2147483648 ～ 2147483647	823、-2493、123k、123K
十六进制整数	00000000H ～ FFFFFFFFH	FFFABCDEH、2345H、FH
实数	-214748.3648 ～ 214748.3647 根据小数点位置变化	763.0、824.2、234.56、-321.12345

补充

- -（负）符号不能省略，但 +（正）符号可以省略。

```
[A1]+123   ⇒  [A1]123
[A1]-123   ⇒  [A1]-123
```

- 在十进制整数后添加 K，则数值会发生 1000 倍变化。位置指令值等中 0 较多，不好识别时使用该方法会非常方便。

```
[A1]123k   ⇒  [A1]123000
[A1]123K   ⇒  [A1]123000
```

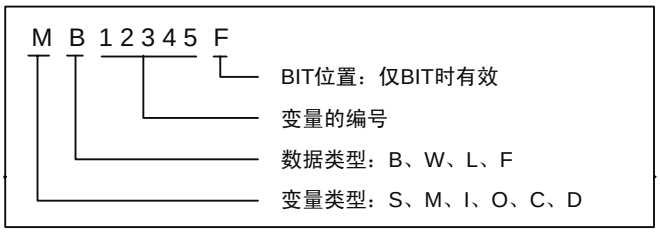
(2) 变量的表述

运动程序能够使用的变量如下所示。

类别	变量型	数据类型			
		BIT	WORD	LONG	FLOAT
全局变量	S 寄存器	SB	SW	SL	SF
	M 寄存器	MB	MW	ML	MF
	I 寄存器	IB	IW	IL	IF
	O 寄存器	OB	OW	OL	OF
	C 寄存器	CB	CW	CL	CF
局部变量	D 寄存器	DB	DW	DL	DF

变量的详细信息，请参阅 “6 章 变量 （寄存器）”。
以下是变量的编写示例。

例



补充

参数与变量的表述分为可省略零和不能省略零两种。

■ 可省略零的示例

```
[A1]00123      ⇒  [A1]123
[A1]MW00010    ⇒  [A1]MW10
[A1]100.000    ⇒  [A1]100.
```

■ 不能省略零的示例

```
MPM001;（在程序起始处编写的程序编号）
MSEE MPS002;
```

7.2 运动模块的参数

运动程序的运动控制动作取决于运动模块的参数设定。因此，运行运动程序之前，需根据机械情况预先设定运动模块的参数。本节将对运动程序控制最低所需的运动模块的参数设定进行说明。

7.2.1 轴类型的选择

位置控制方式分为，往复运动等在某一规定区间内执行的有限长度位置控制方式和只单方向旋转的无限长度位置控制方式。此外，无限长度位置控制方式分为像皮带传送机等旋转 1 圈后会将位置数据复位成零的方式和即使旋转 1 圈，也不会复位位置数据的单方向旋转的方式。轴类型的选择是对要使用以上哪种位置控制方式的选择。

参数类型	参数编号（寄存器编号）	名称	内容	初始值
运动固定参数	No.1 Bit0	功能选择标记 1 “轴型选择”	指定控制轴的位置控制方式。 0: 有限长轴 是使用有限长度位置控制方式或旋转 1 圈后位置数据不复位，并单方向旋转的无限度长度位置控制方式的轴 1: 无限长轴 是使用旋转后位置数据会复位的无限长位置控制方式的轴	0
	No.10	无限长轴的复位位置	“轴型选择”为“1”（无限长轴位置控制方式）时，使用指令单位设定位置数据的复位位置。	360000

7.2.2 指令单位

运动程序中输入的指令位置的单位包括 pulse，mm，deg，inch，μm，其统称为指令单位。指令单位通过运动固定参数 No.4“ 指令单位选择 ” 进行设定。此外，通过设定运动固定参数 No.5“ 小数点后位数 ” 确定能够发出指令的最小指令单位。

运动固定参数 No.5 “ 小数点后位数 ”	运动固定参数 No.4“ 指令单位选择 ”				
	0：pulse	1：mm	2：deg	3：inch	4：μm
0：小数点后 0 位数	1 pulse*1	1 mm	1 deg	25.40 mm	1 μm
1：小数点后 1 位数		0.1 mm	0.1 deg	2.54 mm	0.1 μm
2：小数点后 2 位数		0.01 mm	0.01 deg	0.25 mm	0.01 μm
3：小数点后 3 位数		0.001 mm	0.001 deg	0.025 mm	0.001 μm
4：小数点后 4 位数		0.0001 mm	0.0001 deg	0.0025 mm	0.0001 μm
5：小数点后 5 位数		0.00001 mm	0.00001 deg	0.00025 mm	0.00001 μm

最小指令单位

*1.“pulse” 时，运动固定参数 No.5“ 小数点后位数设定 ” 无效。

补充

轴移动命令的指令位置的范围见下表。

运动固定参数 No.5 “ 小数点后位数 ”	运动固定参数 No.4 “ 指令单位选择 ”	
	0：pulse	1：mm, 2：deg, 3：inch, 4：μm
0：小数点后 0 位	-2147483648 ～ 2147483647	-2147483648 ～ 2147483647
1：小数点后 1 位	-2147483648 ～ 2147483647	-214748364.8 ～ 214748364.7
2：小数点后 2 位	-2147483648 ～ 2147483647	-21474836.48 ～ 21474836.47
3：小数点后 3 位	-2147483648 ～ 2147483647	-2147483.648 ～ 2147483.647
4：小数点后 4 位	-2147483648 ～ 2147483647	-214748.3648 ～ 214748.3647
5：小数点后 5 位	-2147483648 ～ 2147483647	-21474.83648 ～ 21474.83647

7.2.3 电子齿轮

将机械系在输入 1 个指令单位后的变化量 （移动量）称为 “ 输出单位 ”。电子齿轮，是指输入位置或速度的 1 个指令单位时，不通过减速机等的输入装置来调节机械系变化量大小的功能。
电机侧的轴旋转 m 次，负载侧的轴会旋转 n 次的机械构成时，使用电子齿轮功能可以设定成 “ 指令单位 ”=“ 输出单位 ”。
电子齿轮的功能通过

- 运动固定参数 No.6 “ 机械每旋转 1 次的移动量 ”
- 运动固定参数 No.8 “ 电机侧齿轮比 ”
- 运动固定参数 No.9 “ 机械侧齿轮比 ”

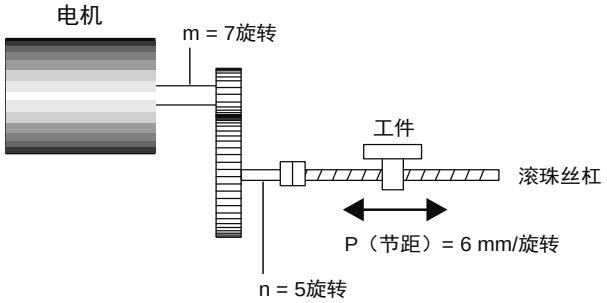
的设定来实现。

（注）“ 指令单位选择 ” 为 pulse 时，电子齿轮的功能无效。

以下是使用滚珠丝杠及圆台的工件时的设定示例。

(1) 滚珠丝杠时参数设定示例

- 机械规格：电机轴旋转 7 次时，滚珠丝杠侧的轴旋转 5 次 （参考下图）。
- 指令单位：0.001mm



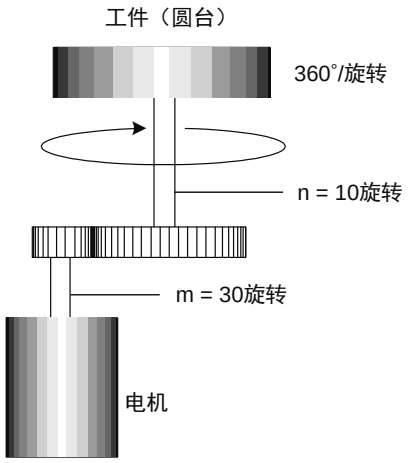
在上述条件下，想通过输入 “1 指令单位 ”，工件会移动 “0.001mm” 时，即，想设定成 “1 指令单位 ” = “1 输出单位 ” 时，应如下设定运动固定参数 No.6 • No.8 • No.9。

- No.6：机械旋转 1 次的移动量 = 6mm/0.001mm = 6000 （指令单位）
- No.8：电机侧齿轮比 = m = 7
- No.9：机械侧齿轮比 = n = 5

（注）请将伺服单元的电子齿轮设定成 “1:1”。

(2) 圆台时参数设定示例

- 机械规格：电机轴旋转 30 次时，圆台侧的轴旋转 10 次 （参考下图）。
- 指令单位：0.1°



在上述条件下，想通过输入 “1 指令单位 ” 实现圆台旋转 “0.1° ” 时，（想设定成 “1 指令单位 ” = “1 输出单位 ” 时）应如下设定运动固定参数 No.6 • No.8 • No.9。

- No.6：机械旋转 1 次的移动量 = 360° /0.1° = 3600 （指令单位）
- No.8：电机侧齿轮比 = m = 30
- No.9：机械侧齿轮比 = n = 10

（注）1. 如果 No.8 • No.9 的比值 m/n 固定，则也可以设定为 “m = 3，n = 1”。
2. 如果是伺服单元的电子齿轮，请将 m/n 设定为 “1:1”。

7.2.4 速度指令

运动程序中输入的速度单位包括指令单位 /s、10^n 指令单位 /min、0.01% 指定、0.0001% 指定，通过运动设定参数 OWxx03 Bit0 ～ 3 “ 速度单位选择 ” 设定。

运动设定参数	速度单位	坐标语的输入方法
OWxx03 Bit0 ～ 3 “ 速度单位选择 ”	0：指令单位 /s	按 1 秒钟的移动量 [指令单位] 输入。
	1：10^n 指令单位 /min	按 1 分钟的移动量 [指令单位] 输入。 指令单位为 pulse 时： 指令单位为 pulse 时，输入值发生 1000 倍变化。 例）指令单位＝ pulse VEL [A1]1000; 时，变成 000000[pulse/min]。 指令单位不为 pulse 时： 指令单位不为 pulse 时，会执行下面示例所示的处理。 例）指令单位＝ mm，小数点以下位数 n ＝ 3 VEL [A1]1000; 时，变成 1000*10^3[0.001mm/min]。
	2：0.01% 指定	使用额定转数的 0.01% 单位输入。
	3：0.0001% 指定	使用额定转数的 0.0001% 单位输入。

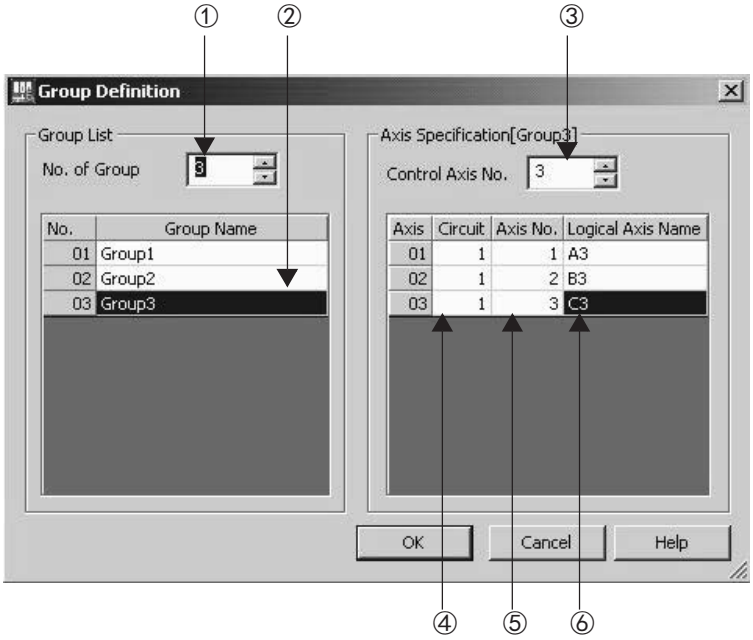
7.2.5 加减速设定

运动程序中输入的加减速度的单位包括指令单位 /s^2，ms，通过运动设定参数 OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” 进行设定。

运动设定参数	速度单位	坐标语的输入方法
OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ”	0：指令单位 /s^2	按 1 秒钟的加减速度 [指令单位 /s^2] 输入。
	1：ms	使用加减速时间 [ms] 输入。

7.3 分组定义

可以在 ***Group Definition*** （分组定义）画面中对所要归纳为群组的轴进行定义。
下面介绍分组定义画面。



①群组数 （No.of Group）

设定组运行的个数。
在单组运行中使用时，将其设定为 1。
在多组运行中使用时，设定为相应群组运行的个数。

②组名称 （Group Name）

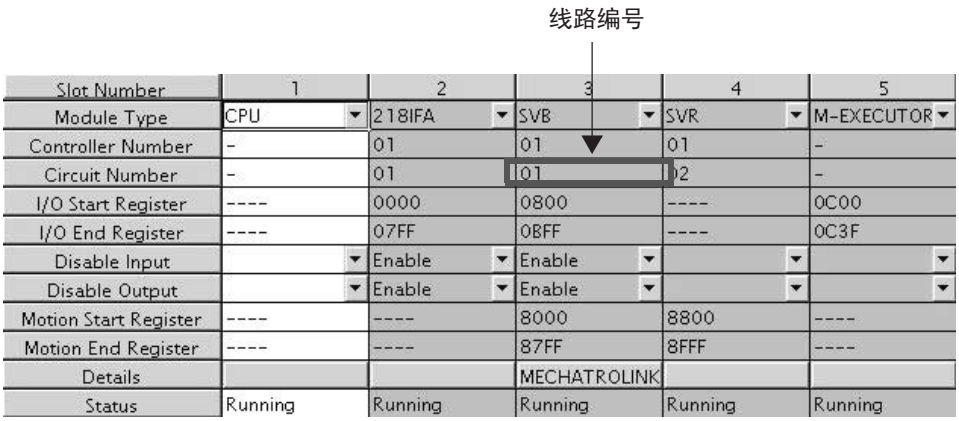
定义组的名称。

③控制轴数 （Control Axis No.）

设定以组进行控制的轴数。

④线路 （Circuit）

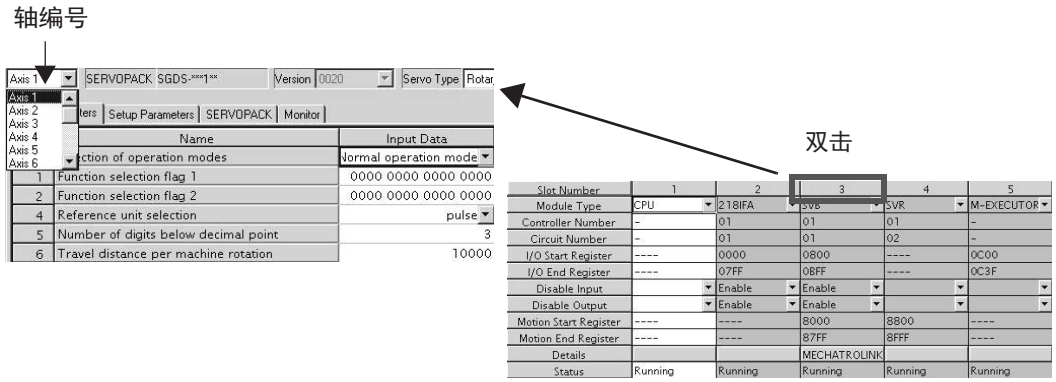
设定要使用的运动模块的线路编号。
线路编号可通过模块构成定义进行确认。



⑤轴编号

设定要使用的轴的轴编号。

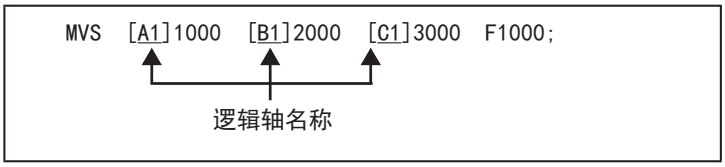
轴编号可通过要使用的运动模块的详细画面进行确认。



⑥逻辑轴名称

定义指定的周边号名称。

此时定义的名称可用于运动程序编程。



7.4 运算的优先顺序

运动语言的运算存在优先顺序。
进行 3 项以上的运算时，请使用 （）指定运算的优先顺序。
以下列出了运算的优先顺序。

高 ⟶ 低

优先顺序 / 运算符	1	2	3	4
括弧	()			
NOT		!		
AND			&	
OR				|
XOR				^
数值运算				+ − * /

■ 数值运算示例

例

・运算示例

```
MW00100 = 1 + 2
```

在该式中，运算 1 + 2 = 3，将运算结果代入 MW00100。

・3 项以上的运算示例

```
MW00100 = 1 + (2 * 3)
```

在该式中，将 2 * 3 = 6 的运算结果和 1 进行相加，将其运算结果代入 MW00100。
因此，得出 MW00100 = 7。

补充

■ 3 项以上运算时的注意事项

如下例所示，进行以下编写。

```
MW00100 = 1 + 2 * 3;
```

如上表所示，在该公式中，首先运算 1 + 2 = 3，然后运算 3 * 3 = 9。
将其运算结果代入 MW00100。因此，得出 MW00100 = 9。

■ 逻辑运算的示例

例

・运算示例

```
MW00100 = 0001H | 0002H
```

在该式中，进行 0001H 与 0002H 的逻辑或 (OR) 运算，并将结果代入 MW00100。

・3 项以上的运算示例

```
MW00100 = (1111H | 2222H) & 00FFH
```

在该式中，先进行 1111H 与 2222H 的逻辑或 (OR) 运算，然后令其结果再与 00FFH 进行逻辑与 (AND) 运算，最后将结果代入 MW00100。
因此，得出 MW00100 = 0033H。

补充

■ 3 项以上运算时的注意事项

如下例所示，进行以下编写。

```
MW00100 = 1111H | 2222H & 00FFH;
```

在该式中，先进行 2222H 和 00FFH 的逻辑与 (AND) 运算，然后令其结果再与 1111H 进行逻辑或 (OR) 运算，最后将结果代入 MW00100。
因此，得出 MW00100 = 1133H。

7.5 命令和执行扫描

7.5.1 命令类型

运动语言命令有 3 种类型。根据命令类型的不同，扫描的次数会有所不同。下面列出了命令类型的分类和相应的执行扫描的次数。

命令类型	命令	执行扫描的次数
S 型	运算命令	单次扫描
M 型	轴移动类命令	多次扫描
T 型	定时器类命令	

以下列对各命令类型进行详细说明。

■ S 型命令

S 型命令是包含运算命令，在一个扫描周期内完成。
用 S 型命令连续编译的程序会一个扫描周期内运行。

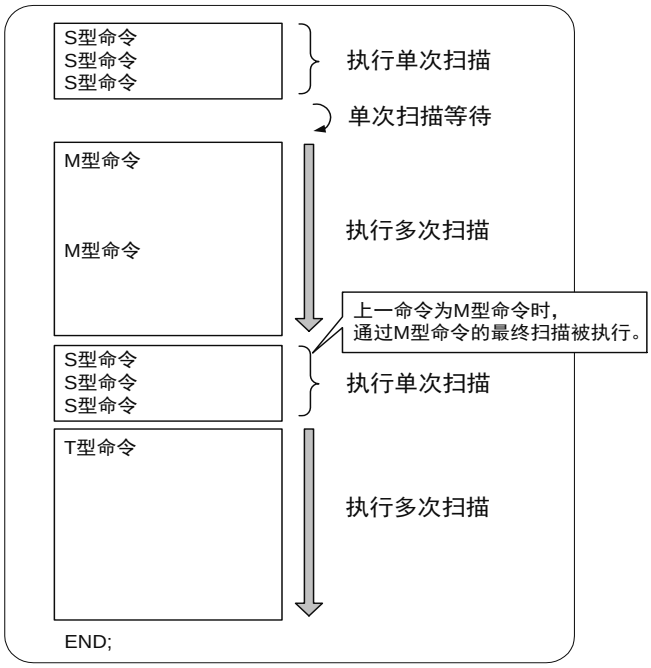
■ M 型命令

M 型命令包含轴移动命令，可执行多次扫描。
从 S 型命令切换到 M 型命令可通过单次扫描的请求。

■ T 型命令

T 型命令包含定时器命令，可执行多次扫描。

下面是各命令类型的工作示意图。



7.5.2 命令类型一览

命令类型一览如下表所示。

类别	命令	S 型	M 型	T 型	类别	命令	S 型	M 型	T 型
轴设定命令	ABS	○			数值运算	=	○		
	INC	○				+	○		
	ACC		○			−	○		
	DCC		○			*	○		
	SCC		○			/	○		
	VEL		○			MOD	○		
	FMX	○			逻辑运算	|	○		
	IFP	○				&	○		
	IAC	○				^	○		
	IDC	○				!	○		
轴移动命令	MOV		○		数值比较	==	○		
	MVS		○			<>	○		
	MCW		○			>	○		
	MCC		○			<	○		
	ZRN		○			>=	○		
	SKP		○			<=	○		
	MVT		○		数据操作	SFORK	○		
	EXM		○			SFL	○		
控制命令	POS	○				BLK	○		
	MVM	○				CLR	○		
	PLN	○				ASCII	○		
	PLD	○			基本函数	SIN	○		
	PFN		○			COS	○		
	INP		○			TAN	○		
程序控制命令	IF ELSE IEND	○				ASN	○		
	WHILE WEND	○				ACS	○		
	PFORK JOINTO PJOINT	○				ATN	○		
	SFORK JOINTO SJOINT	○				SQT	○		
	MSEE	○				BIN	○		
	SSEE	○				BCD	○		
	UFC		○			S{ }	○		
	FUNC	○				R{ }	○		
	END	○				PON	○		
	RET	○				NON	○		
	TIM			○		TON	○		
	IOW			○		TOF	○		
	EOX			○	C 语言控制命令	CTSK	○		
	SNGD/SNGE	○				CFUNC	○		

7.6 顺序程序的格式

顺序程序与运动程序的格式相同。
但是，以上条件仅限于能够使用运动命令语言命令的顺序程序。关于顺序程序能够使用的命令，请参阅“ 附录 A 运动语言命令 ”。

8 章

相关命令

本章对运动语言命令的相关命令进行说明。

8.1 轴设定命令

在本节中，将对轴设定命令进行说明。

8.1.1 绝对值模式 （ABS）

运动程序	顺序程序
○	×

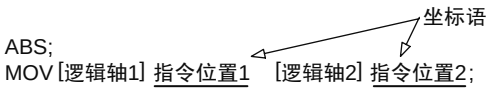
⚠注意

· 在绝对值模式及增量模式下发出相同坐标语指令时，其移动动作会完全不同。运行之前，请务必首先检查是否已正确发出 ABS / INC 命令的指令。

否则可能会因干扰造成工具损坏从而导致人身事故。

(1) 动作说明

绝对值模式 （ABS）是将执行轴移动的命令的坐标语作为 “ 最终目标位置 ” 进行处置的命令。
绝对值模式 （ABS）一旦发出指令，则在下一次发出增量模式 （INC）之前会一直保持 ABS 模式。程序运行开始时，变成 ABS 模式。



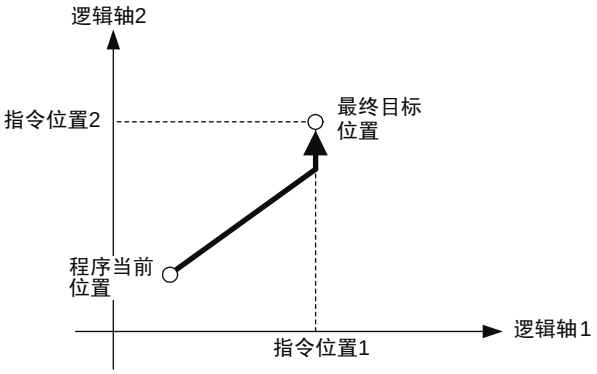


图 8.1 绝对值模式 （ABS）的移动模式

本手册中，将接在执行轴移动的命令的逻辑轴名称后面的坐标语表述为 “ 指令位置 ” 或 “ 位置指令值 ”。

术语?

■ 程序当前位置

所谓 “ 程序当前位置 ” 是指轴移动命令开始轴动作时任务坐标上的当前位置。
此外，将运动程序处理的坐标系称为任务坐标系。
关于任务坐标系，请参阅 “8.3.1 当前值变更 （POS）”。

(2) 格式

- 单独指定时
  ABS;
- 指定到与轴移动命令相同的程序段时
  ABS MOV [逻辑轴名称1] — [逻辑轴名称2] — ;

(3) 程序示例

例 以下是 ABS 命令的程序示例。

```
ABS;                          " 绝对值模式
MOV [A1]10000 [B1]40000;      " 定位
MOV [A1]50000 [B1]20000;      " 定位
END;
```

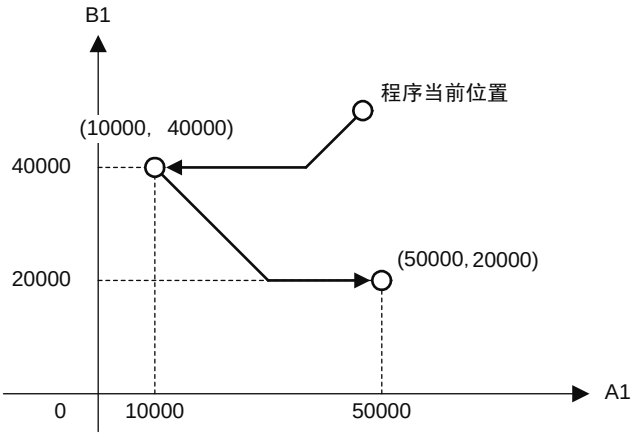


图 8.2 ABS 模式程序示例

(4)ABS 命令的追加事项

(a) 相关运动参数

ABS 命令与运动设定参数没有相关性。
轴移动命令的移动模式（ABS 模式 /INC 模式 ）为运动程序专用的控制数据，无法通过运动设定参数指定。

补充
- 轴移动命令的移动模式 （ABS 模式 / INC 模式）与运动设定参数 OWxx09 Bit5 " 位置指令型 " 的含义不同，请勿混淆。
- 运行运动程序时，与移动模式的设定无关，请务必设定成运动设定参数 OWxx09 Bit5 " 位置指令型 " ＝ 0（增量值叠加方式）。

(b) 有限长轴和无限长轴

有限长轴和无限长轴中坐标语的位置指令值的指定方法不同。
下面分别介绍有限长轴和无限长轴上位置指令值的处理情况。

轴型	轴移动命令的移动模式指定	位置指令值的指定方法
有限长轴	ABS 模式	在位置指令值中指定最终目标位置。
	INC 模式	在位置指令值中指定相对移动量。
无限长轴	ABS 模式	在 0 ～ POSMAX 的范围内，在位置指令值中指定最终目标位置。 使用位置指令值的正负指定移动方向。正值时，指定朝正方向移动，负值时朝负方向移动。
	INC 模式	在位置指令值中指定相对移动量。

补充

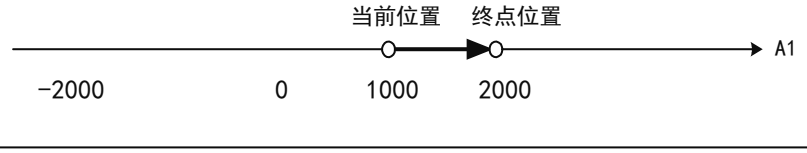
- 有限长轴 / 无限长轴通过运动固定参数 No.1 ***Function selection flag1*** （功能选择标记 1）Bit0 ***Axis selection*** （轴型选择）进行设定。
  有限长轴 / 无限长轴的设定根据机械的构成确定。关于与机械匹配的运动参数设定，请参阅相应运动模块的使用手册。
- POSMAX 通过运动固定参数 No.10 ***Inifinite length axis reset position*** （无限长轴的复位位置 (POSMAX)）设定。

下面列出了 ABS 模式时有限长轴 / 无限长轴的动作。
关于 INC 模式时的动作，请参阅 “8.1.2 增量模式 （INC）”。

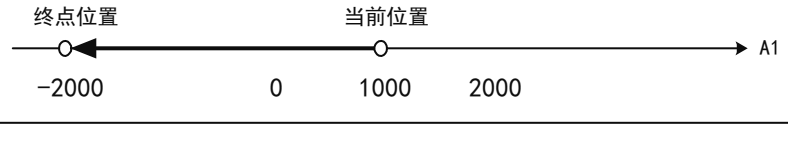
· 针对有限长轴指定了 ABS 模式时

在位置指令值中指定最终目标位置。
例如，指定了距当前位置 1000 的 2000，-2000 的位置指令时，分别会执行以下动作。

```
ABS;
MOV [A1]2000;
```

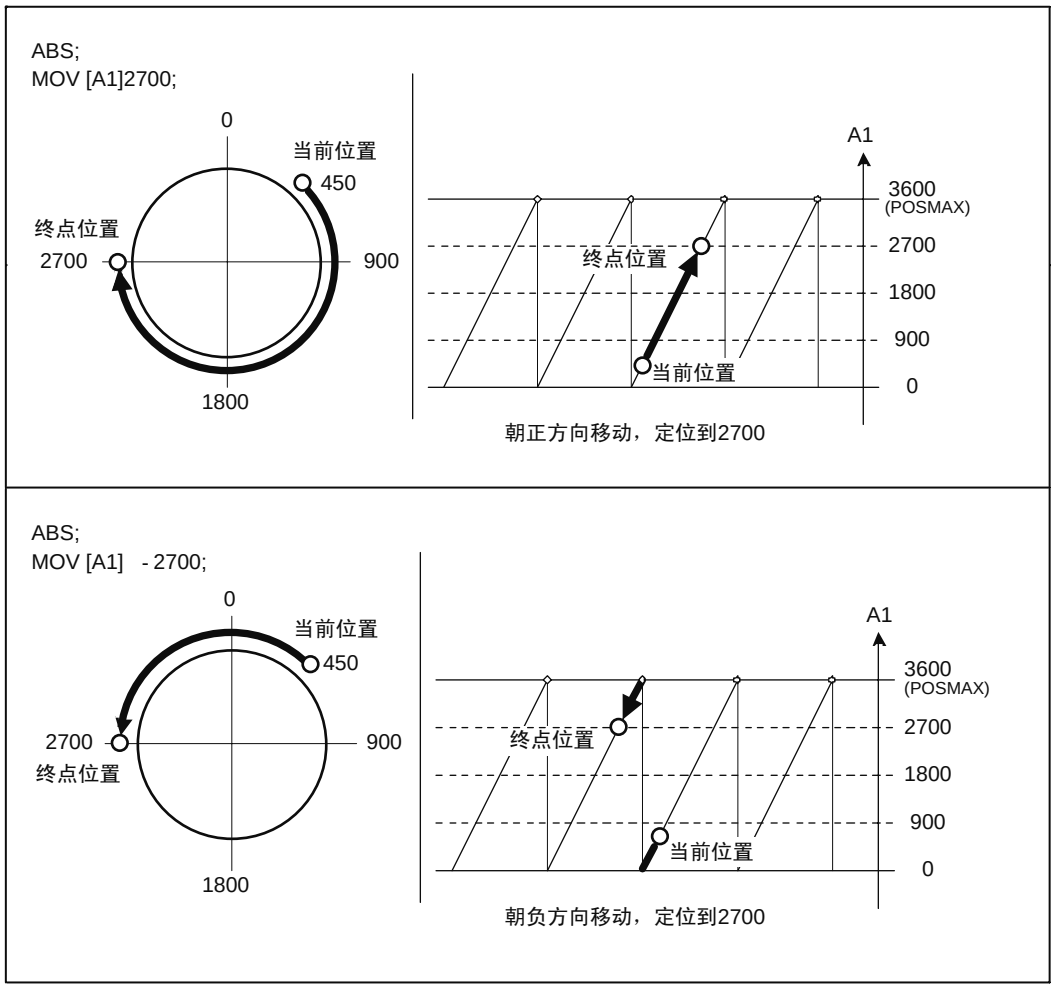



```
ABS;
MOV [A1] -2000;
```



· 针对无限长轴指定了 ABS 模式时

在 0 ～ POSMAX 的范围内，在位置指令值中指定最终目标位置。
使用位置指令值的正负指定移动方向。正值时，指定朝正方向移动，负值时朝负方向移动。
例如，针对 POSMAX ＝ 3600 的无限长轴指定了距当前位置＝ 450 的 2700，-2700 的位置指令时，分别会执行以下动作。



补充

- 通过无限长轴中 ABS 模式的指定向 0 位置移动时，即使指定的是 +0，也会向负方向移动。
  要需要向正方向移动时，请指定 POSMAX 的值。
- 通过无限长轴中 ABS 模式的指定，最终目标位置 （位置指令值的绝对值）超过 POSMAX 时，发生运动程序警报。

8.1.2 增量模式 （INC）

运动程序	顺序程序
○	×

⚠注意

・在绝对值模式及增量模式下发出相同坐标语指令时，其移动动作会完全不同。运行之前，请务必首先检查是否已正确发出 ABS / INC 命令的指令。
否则可能会因干扰造成工具损坏及由此导致人身事故。

(1) 动作说明

增量模式 （INC）是将执行轴移动的命令的坐标语作为 “ 相对移动量 ” 进行处置的命令。
增量模式 （INC）一旦发出指令，则在下一次发出绝对值模式 （ABS）之前会一直保持 INC 模式。程序运行开始时，变成 ABS 模式。

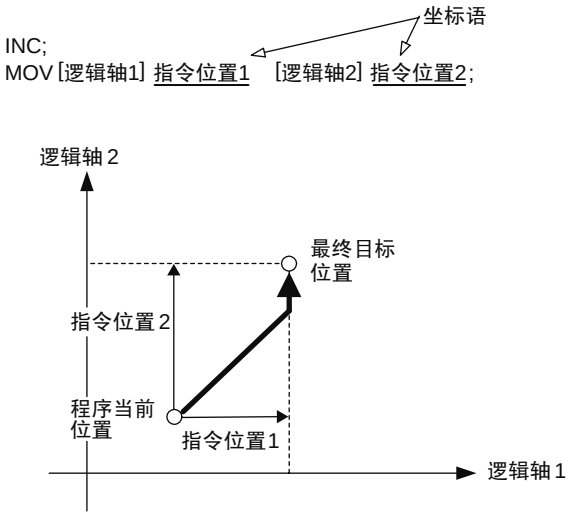


图 8.3 增量模式 （INC）的移动模式

本手册将接在执行轴移动的命令的逻辑轴名称后的坐标语表述为 “ 指令位置 ” 或 “ 位置指令值 ”。

补充

■ 程序当前位置

“ 程序当前位置 ” 是指轴移动命令开始轴动作时任务坐标上的当前位置。
此外，将运动程序处理的坐标系称为任务坐标系。关于任务坐标系，请参阅 “8.3.1 当前值变更 （POS）”。

(2) 格式

- 单独指定时
  INC;

- 指定到与轴移动命令相同的段时
  INC　MOV［逻辑轴名称1］一［逻辑轴名称2］一；

(3) 程序示例

例 以下是 INC 命令的程序示例。

```
INC;                          “增量模式
MOV [A1]20000 [B1]30000;      “定位
MOV [A1]20000 [B1]10000;      “定位
END;
```

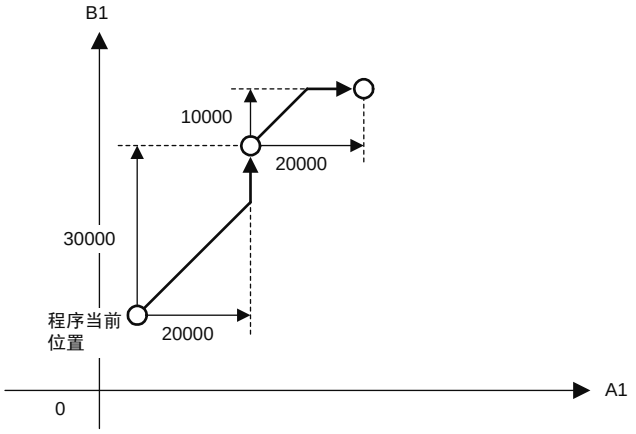


图 8.4 INC 模式程序示例

(4) INC 命令的追加事项

(a) 相关运动参数

INC 命令与运动设定参数无关。
轴移动命令的移动模式（ABS/INC 模式）为运动程序专用的控制数据，无法通过运动设定参数指定。

(b) 有限长轴和无限长轴

有限长轴和无限长轴处置坐标语的位置指令值的方法不同。
下面分别介绍有限长轴和无限长轴上位置指令值的处置情况。

轴型	轴移动命令的移动模式指定	位置指令值的指定方法
有限长轴	ABS 模式	在位置指令值中指定最终目标位置。
	INC 模式	在位置指令值中指定相对移动量。
无限长轴	ABS 模式	在 0 ～ POSMAX 的范围内，在位置指令值中指定最终目标位置。 使用位置指令值的正负指定移动方向。指定值为正值时，向正方向移动，负值时向负方向移动。
	INC 模式	在位置指令值中指定相对移动量。

补充

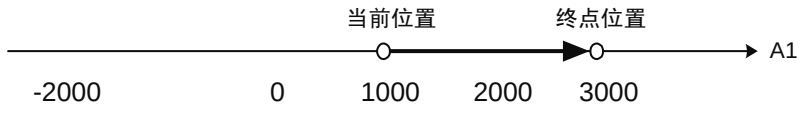
- 有限长轴 / 无限长轴通过运动固定参数 No.1 ***Function selection flag1*** （功能选择标记 1）的 Bit0 ***Axis selection*** （轴型选择）进行设定。
  根据机械的构成可选择有限长轴 / 无限长轴。关于与机械匹配的运动参数设定，请参阅相应运动模块的使用手册。
- POSMAX 通过运动固定参数 No.10 ***Infinite length axis reset position*** （无限长轴的复位位置 (POSMAX)）设定。

下面列出了 INC 模式时有限长轴 / 无限长轴的动作。
关于 ABS 模式时的动作，请参阅 “8.1.1 绝对值模式 （ABS）”。

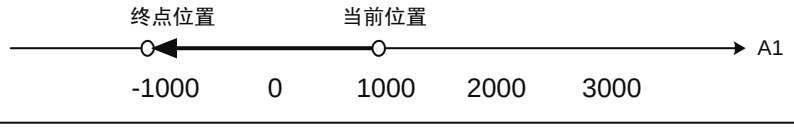
· 针对有限长轴指定了 INC 模式时

在位置指令值中指定相对移动量。
例如，指定了距当前位置为 1000 的 2000，-2000 的位置指令时，分别会执行以下动作。

```
INC;
MOV [A1]2000;
```

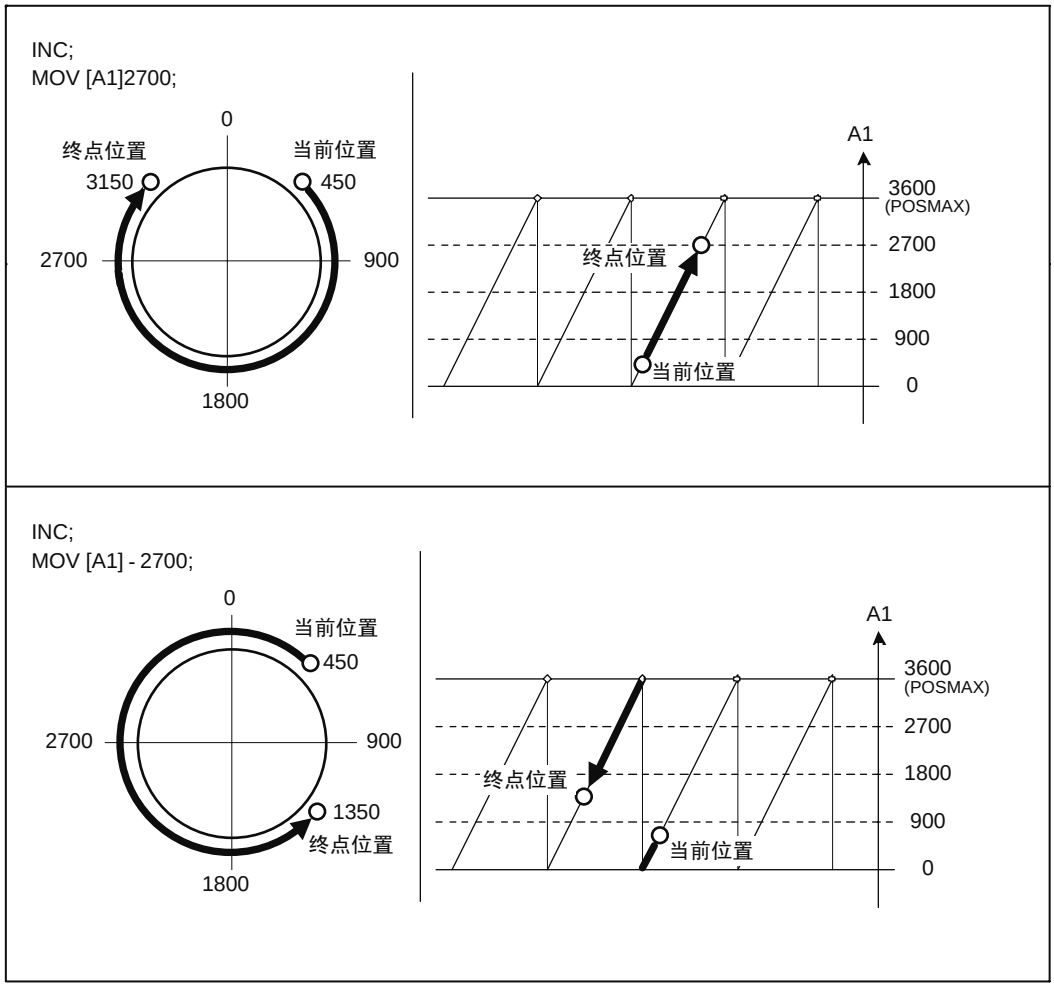



```
INC;
MOV [A1] - 2000;
```



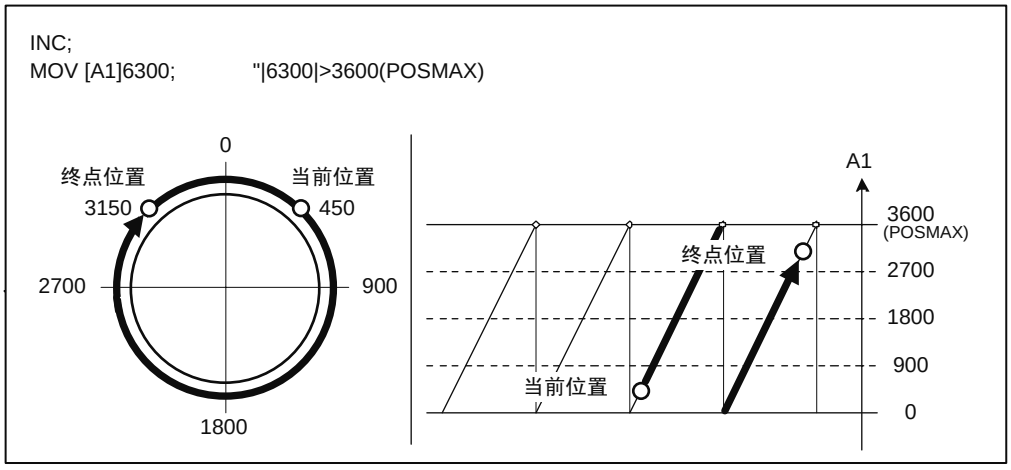
· 针对无限长轴指定了 INC 模式时

在位置指令值中指定相对移动量。
例如，针对 POSMAX ＝ 3600 的无限长轴指定了距当前位置为 450 的 2700，-2700 的位置指令时，分别会执行以下动作。



补充

· 位置指令值 （坐标语）的绝对值超过了 POSMAX 时，INC 模式下，位置指令值 （坐标语）会作为相对移动量执行轴动作。



8.1.3 加速时间变更 （ACC）

运动程序	顺序程序
○	×

(1) 动作说明

加速时间变更 （ACC）是针对以下轴移动命令更改各轴的加速时间或加速度的命令。

- 定位 （MOV）
- 指定时间定位 （MVT）
- 外部定位 （EXM）

同时最多可更改 16 轴。省略该指令的轴的加速时间不会更改。
已变更的轴的加速时间会一直保持，直至通过加速时间变更 （ACC）重新进行设定。

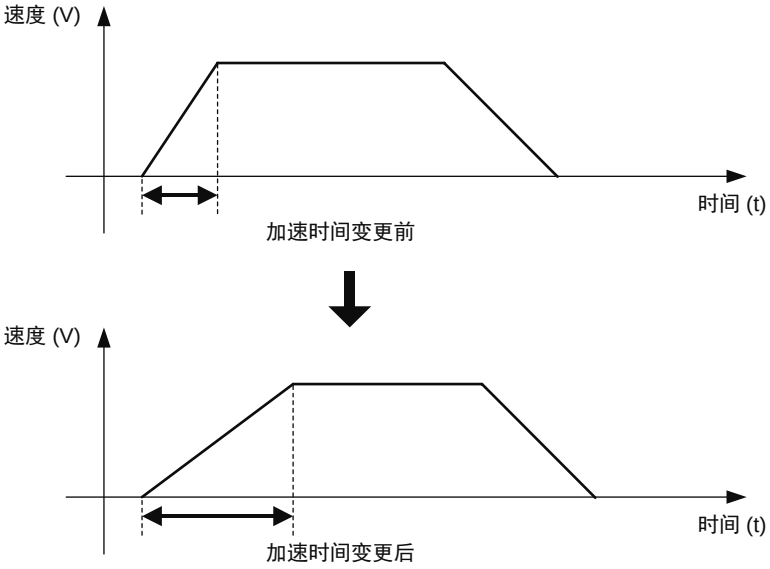


图 8.5 更改加速时间

补充

- 加速时间变更 (ACC) 是设定定位类命令 （MOV、EXM、MVT）加速时间的命令。插补类命令 （MVS、MCW、MCC、SKP）的加速时间通过 IAC 命令进行设定。
- ACC/DCC/SCC 命令适用于所有的运动模块。
  此外，与 PO-01 模块的组合支持以下版本。

MP2000 系列版本	PO-01 版本
MP2000 系列 Ver.2.46 或更高	PO-01 Ver.1.06 或更高

(2) 格式

ACC [逻辑轴名称1] 加速时间 [逻辑轴名称2] 加速时间 [逻辑轴名称3] 加速时间 ・・・;

项目	单位	可使用的数据
加速时间或加速度	ms、或 指令单位/s^2 ※ 单位通过运动设定参数 OWxx03 Bit4～7 “加减速度单位选择”进行设定	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

(3)ACC 命令的设定项目

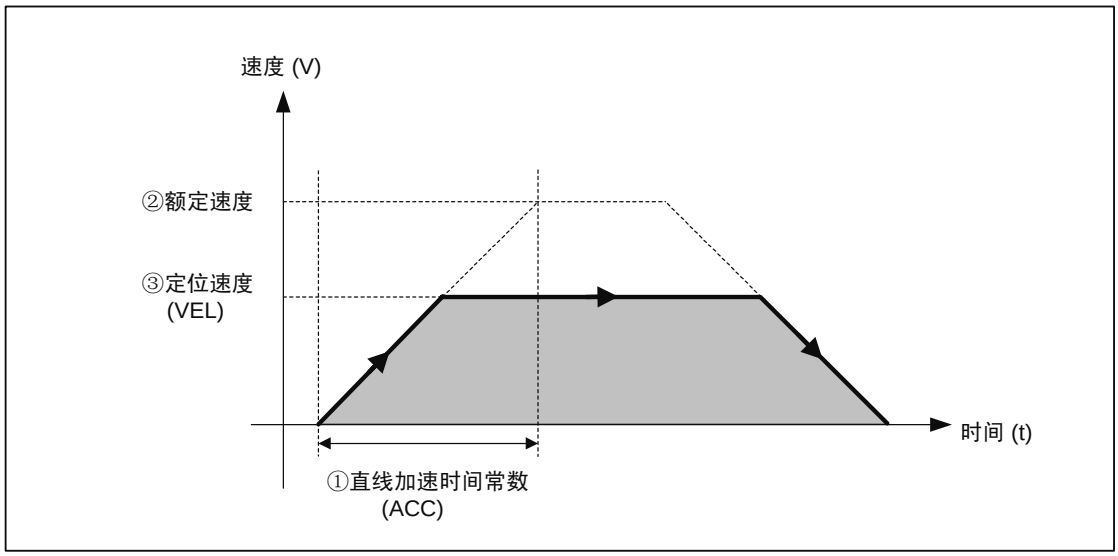
ACC 命令的设定值单位可从加速时间 [ms] 或加速度 [指令单位 /s²] 中选择。

运动设定参数 OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ”

参数名称	加减速度单位
功能设定 1 加减速度单位选择	0：指令单位 /s² 1：ms （默认）

- OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” 为 “1：ms” 时

动作示意图



①直线加速时间常数

ACC 命令的设定值变成 “ 直线加速时间常数 ” （从速度 0 到额定速度所需的时间）。
指令范围会发生如下变化。

1 ～ 32767[ms]

②额定速度

各轴的额定速度通过运动固定参数 No.34 “ 额定转数 ” 进行设定。
详细信息，请参阅对应运动模块的用户手册。

③定位速度

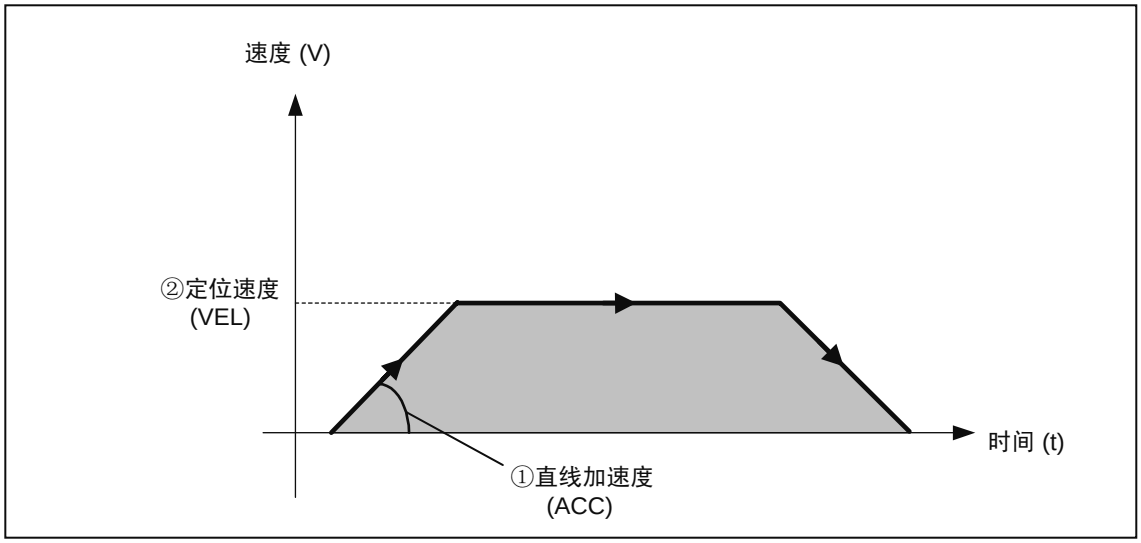
是定位类命令 (MOV、MVT、EXM) 的速度。
各轴的定位速度通过进给速度变更 （VEL）进行设定。

补充

指定时间定位 (MVT) 时，定位速度不是 VEL 命令的指令值。
指定时间定位 (MVT) 基于设定的定位时间和移动量更改定位速度。

・ OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” 为 “0：指令单位 /s²” 时

动作示意图



①直线加速度

ACC 命令的设定值变成 “ 直线加速度 ” 。
指令范围会发生如下变化。

1 ～ 2^{31}-1[指令单位 /s²]

②定位速度

是定位类命令 (MOV、MVT、EXM) 的速度。
各轴的定位速度通过进给速度变更 （VEL）进行设定。

补充

指定时间定位 (MVT) 时，定位速度不是 VEL 命令的指令值。
指定时间定位 (MVT) 基于设定的定位时间和移动量更改定位速度。

(4) 程序示例

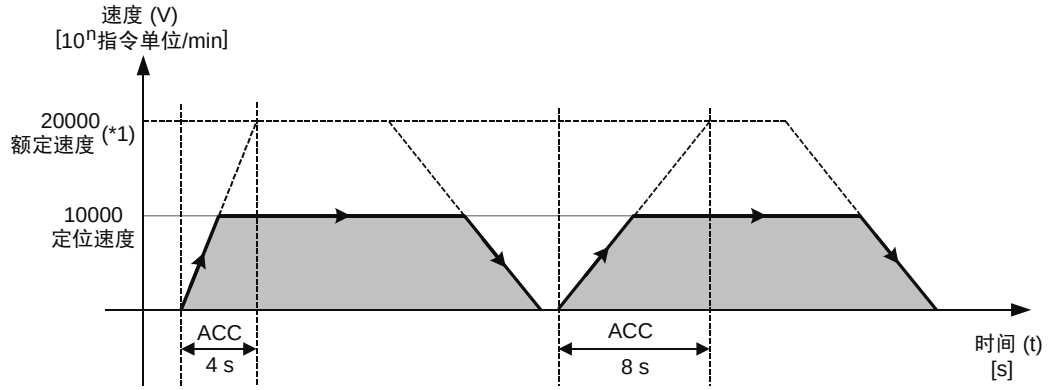
以下是 ACC 命令的程序示例。

例

・ OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” 为 “1：ms” 时

下面是执行以 4s 从速度 0 加速到额定速度的 MOV 命令和以 8s 进行加速的 MOV 命令的示例。

```
INC;                        “ 增量模式
VEL [A1]10000;              “ 进给速度变更 [10**n 指令单位 /min]
DCC [A1]8000;               “ 减速时间变更 [ms]
ACC [A1]4000;               “ 加速时间变更 [ms]
MOV [A1]5000000;            “ 定位
DL00000 = 8000;             “ 加速时间 [ms]
ACC [A1]DL00000;            “ 加速时间变更 [ms]
MOV [A1]5000000;            “ 定位
END;
```



*1. 额定速度的单位不是 [min^{-1}]，需转换成与定位速度相同的单位 [10^n 指令单位 /min]。

图 8.6“ 加速时间变更 ” 的程序示例 (1)
加减速度单位 “ms” 时

例

- OWxx03 Bit4 ～ 7“ 加减速度单位选择 ” 为 “0：指令单位 /s²” 时

下面是执行以加速度 60.000[mm/s²] 进行加速的 MOV 命令和以加速度 100.000[mm/s²] 进行加速的 MOV 命令的示例。但，1 指令单位取 0.001mm。

```
INC;                    “ 增量模式
VEL [A1]18000;          “ 进给速度变更 [10**n 指令单位 /min]
DCC [A1]100000;         “ 减速时间变更 [ 指令单位 /S*S]
ACC [A1]60000;          “ 加速度变更 [ 指令单位 /S*S]
MOV [A1]5000000;        “ 定位
DL00000 = 100000;       “ 加速度 [ 指令单位 /S*S]
ACC [A1] DL00000;       “ 加速度变更 [ 指令单位 /S*S]
MOV [A1]5000000;        “ 定位
END;
```

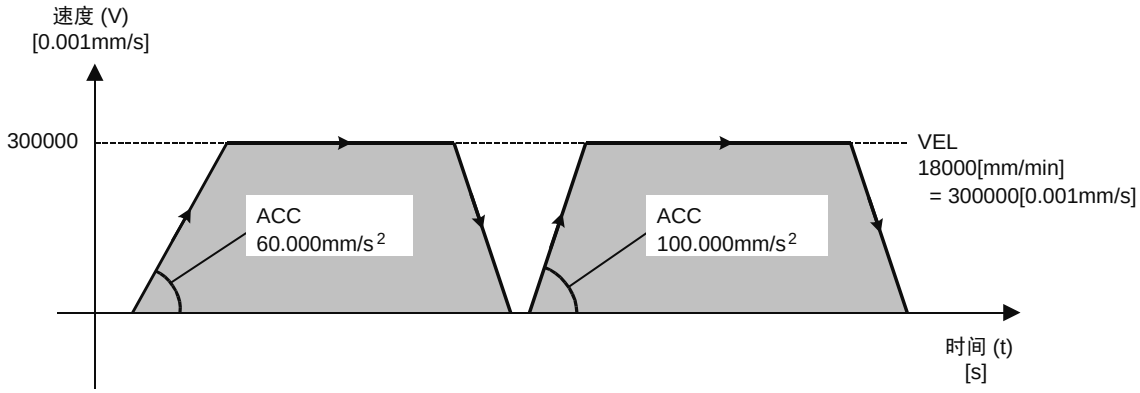


图 8.7“ 加速时间变更 ” 程序示例 (2)
加减速度单位 “ 指令单位 /s²” 时

(5) ACC 命令的追加事项

(a) 相关运动参数

ACC 命令会更改运动设定参数的加速时间。

参数名称	寄存器编号	说明
直线加速度 / 加速时间常数	OLxx36	设定直线加速度或直线加速时间常数。

不通过 ACC 命令，直接更改运动设定参数 OLxx36“ 直线加速度 / 加速时间常数 ” 也可以更改加速时间。直接更改加速时间的步骤，请参阅下表。

（参考）定位类命令加速时间的规格和设定步骤

运动模块	规格	设定步骤
SVA-01、PO-01、SVR	轴会按照运动设定参数 OLxx36“ 直线加速度 / 加速时间常数 ” 的加速时间执行动作。	将加速时间设定成运动设定参数 OLxx36“ 直线加速度 / 加速时间常数 ”。
SVB-01、内置 SVB	轴会按照伺服单元用户常数的加速度执行动作。	将加速时间设定成运动设定参数 OLxx36“ 直线加速度 / 加速时间常数 ”。之后，为了将加速时间应用到伺服单元，会通过运动设定参数 OWxx08“ 运动命令 ” 执行 “ 更改直线加速时间常数 ”(=10)。(*1)

*1. 内置 SVB/SVB-01 具有将运动设定参数 OLxx36“ 直线加速度 / 加速时间常数 ” 自动应用到伺服单元用户常数加速度的功能。启用了执行自动应用的功能时，不需要通过运动设定参数 OWxx08“ 运动命令 ” 执行 “ 更改直线加速 ”=10)。执行自动应用的功能时，请参阅 《机器控制器 MP2000 系列运动模块 SVB/SVB-01 用户手册》(资料编号：SIJP C880700 33) （日文） 中的 “11.6 自动反映的用户常数 / 参数 ”。

(b) 加速时间和减速时间

下表中的运动模块与伺服单元组合时，无法分别设定加速时间和减速时间。通过设定加速时间来设定减速时间。除以下伺服单元外，可使用 ACC 命令和 DCC 命令分别设定加速时间和减速时间。

运动模块	伺服单元	备注
SVB-01、内置 SVB	SGD-N	• 内置 SVB/SVB-01 时，轴会按照伺服单元用户常数的加减速度执行动作。 • 伺服单元 SGD-N、SGDB-N 的加速时间 / 减速时间的用户常数为通用。
	SGDB-N	

8.1.4 减速时间变更 （DCC）

运动程序	顺序程序
○	×

(1) 动作说明

减速时间变更 （DCC）是针对以下轴移动命令更改各轴的减速时间或减速度的命令。

- 定位 （MOV）
- 指定时间定位 （MVT）
- 外部定位 （EXM）

同时最多可更改 16 轴。省略该指令的轴的减速时间不会更改。
已更改轴的减速时间会一直保持，直至通过减速时间变更 （DCC）命令重新进行设定。

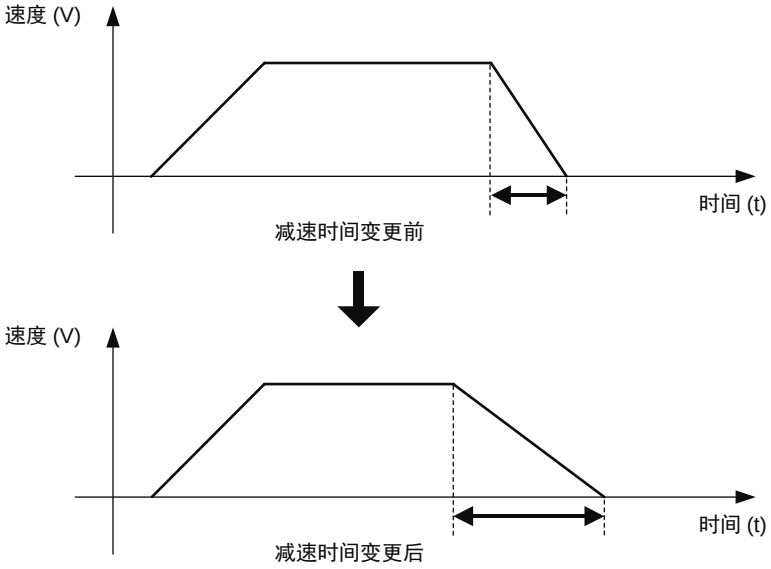


图 8.8 减速时间变更

补充

- 减速时间变更 (DCC) 是设定定位类命令 （MOV、EXM、MVT）减速时间的命令。插补类命令 （MVS、MCW、MCC、SKP）的减速时间使用 IDC 命令进行设定。
- ACC/DCC/SCC 命令支持所有的运动模块。
  此外，与 PO-01 模块的组合支持以下版本。

MP2000 系列版本	PO-01 版本
MP2000 系列 Ver.2.46 或更高	PO-01 Ver.1.06 或更高

(2) 格式

DCC [逻辑轴名称1] 减速时间 [逻辑轴名称2] 减速时间 [逻辑轴名称3] 减速时间 ・・・;

项目	单位	可使用的数据
减速时间或减速度	ms或指令单位/s^2 ※ 单位通过运动设定参数 OWxx03 Bit4～7 “加减速度单位选择”进行设定	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

(3)DCC 命令的设定项目

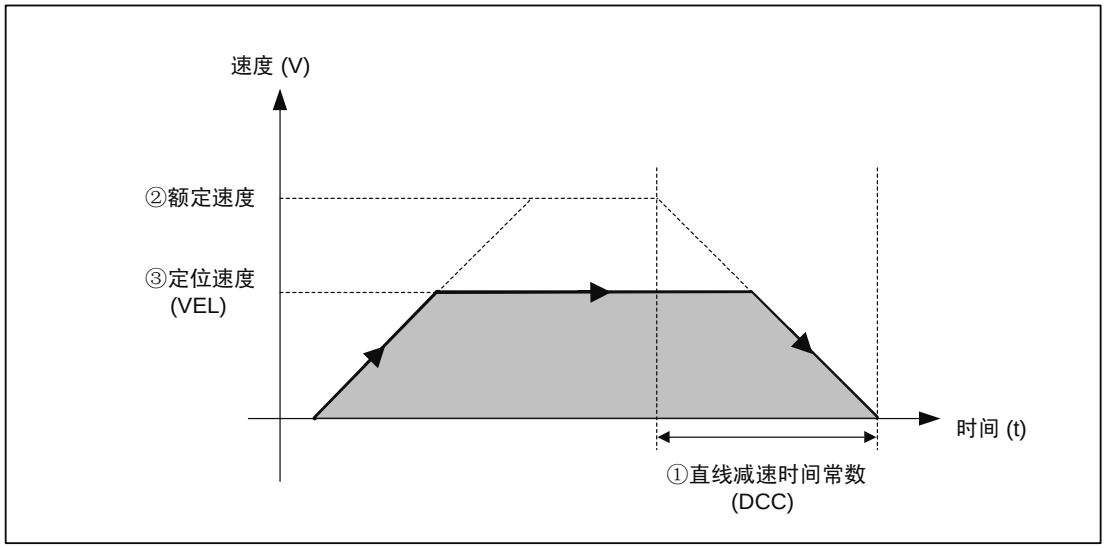
DCC 命令的设定值单位可从减速时间 [ms] 或减速度 [指令单位 /s²] 中选择。

运动设定参数 OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ”

参数名称	加减速度单位
功能设定 1 加减速度单位选择	0：指令单位 /s² 1：ms （默认）

- **OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” 为 “1：ms” 时**

动作示意图



①直线减速时间常数

DCC 命令的设定值变成 “ 直线减速时间常数 ” （从额定速度开始直线减速到速度 0 所需的时间）。
指令范围会发生如下变化。

1 ～ 32767[ms]

②额定速度

各轴的额定速度通过运动固定参数 No.34 “ 额定转数 ” 进行设定。
详细信息，请参阅对应运动模块的用户手册。

③定位速度

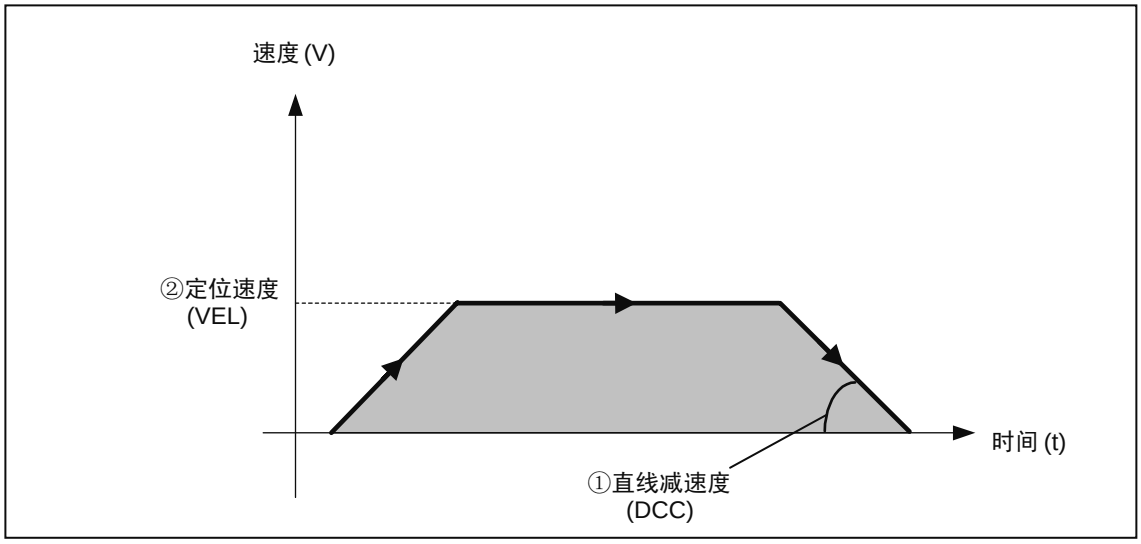
是定位类命令 (MOV、MVT、EXM) 的速度。
各轴的定位速度通过进给速度变更 （VEL）进行设定。

补充

指定时间定位 (MVT) 时，定位速度不是 VEL 命令的指令值。
指定时间定位 (MVT) 基于设定的定位时间和移动量更改定位速度。

・ OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” 为 “0：指令单位 /s²” 时

动作示意图



①直线减速度

DCC 命令的设定值变成 “ 直线减速度 ” 。
指令范围会发生如下变化。

1 ～ 2^{31}-1[指令单位 /s²]

②定位速度

是定位类命令 (MOV、MVT、EXM) 的速度。
各轴的定位速度通过进给速度变更 （VEL）进行设定。

补充

指定时间定位 (MVT) 时，定位速度不是 VEL 命令的指令值。
指定时间定位 (MVT) 基于设定的定位时间和移动量更改定位速度。

(4) 程序示例

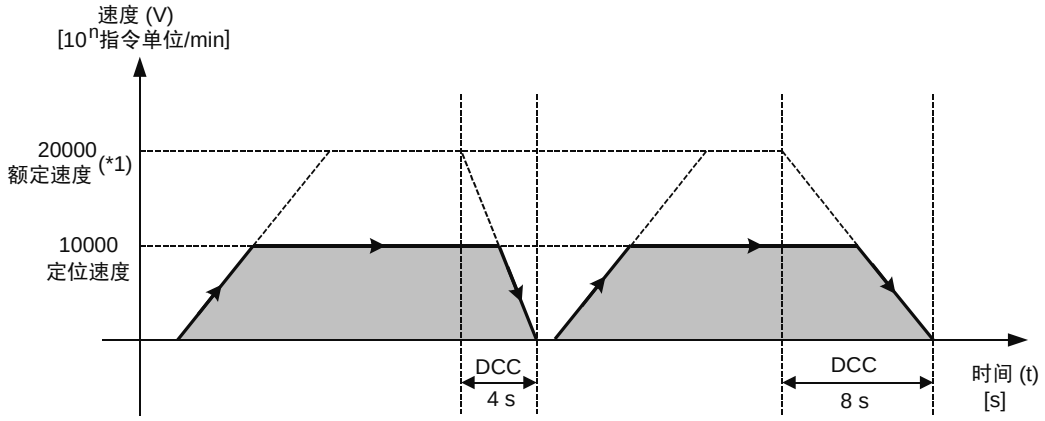
以下是 DCC 命令的程序示例。

例

・OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” 为 “1：ms” 时

下面是执行以 4s 从速度 0 加速到额定速度的 MOV 命令和以 8s 进行加速的 MOV 命令的示例。

```
INC;                  “ 增量模式
VEL [A1]10000;        “ 进给速度变更 [10**n 指令单位 /min]
ACC [A1]8000;         “ 加速时间变更 [ms]
DCC [A1]4000;         “ 减速时间变更 [ms]
MOV [A1]5000000;      “ 定位
DL00000 = 8000;       “ 减速时间 [ms]
DCC [A1]DL00000;      “ 减速时间变更 [ms]
MOV [A1]5000000;      “ 定位
END;
```



*1. 额定速度的单位不是 [min^{-1}]，需转换成与定位速度相同的单位 [10^{n} 指令单位 /min]。

图 8.9 “ 减速时间变更 ” 程序示例 (1)
加减速度单位 “ms” 时

例

- OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” 为 “0：指令单位 /s^2” 时

下面是执行以减速度 60.000[mm/s^2] 进行减速的 MOV 命令和以减速度 100.000[mm/s^2] 进行减速的 MOV 命令的示例。指令单位为 0.001mm。

```
INC;                    “ 增量模式
VEL [A1]18000;          “ 进给速度变更 [10**n 指令单位 /min]
ACC [A1]100000;         “ 加速度变更 [ 指令单位 /S*S]
DCC [A1]60000;          “ 减速度变更 [ 指令单位 /S*S]
MOV [A1]5000000;        “ 定位
DL00000 = 100000;       “ 减速度 [ 指令单位 /S*S]
DCC [A1] DL00000;       “ 减速度变更 [ 指令单位 /S*S]
MOV [A1]5000000;        “ 定位
END;
```

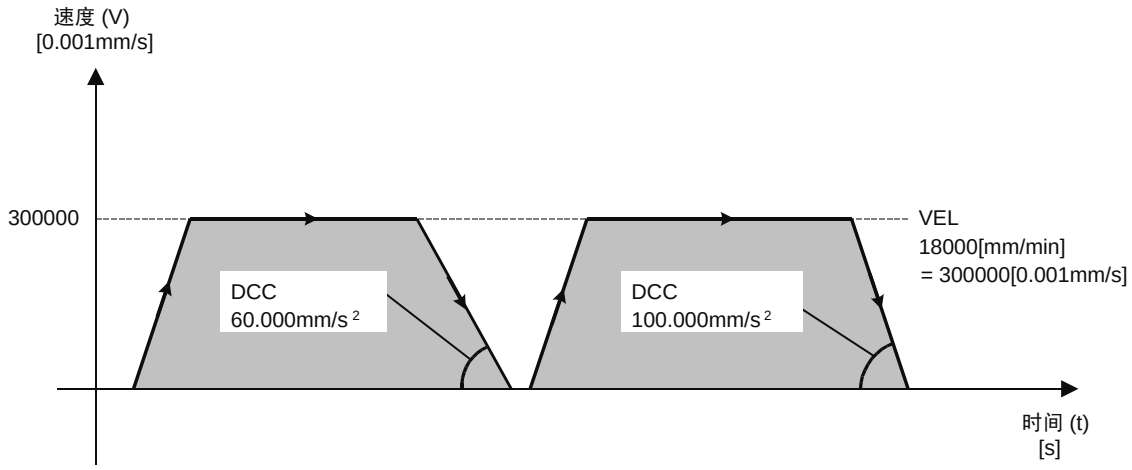


图 8.10 “ 减速时间变更 ” 程序示例 (2)
加减速度单位 “ 指令单位 /s^2” 时

(5)DCC 命令的追加事项

(a) 相关运动参数

DCC 命令会更改运动设定参数的减速时间。

参数名称	寄存器编号	说明
直线减速度 / 减速时间常数	OLxx38	设定直线减速度或直线减速时间常数。

不通过 DCC 命令，直接更改运动设定参数 OLxx38“ 直线减速度 / 减速时间常数 ” 也可以更改减速时间。直接更改减速时间的步骤，请参阅下表。

（参考）定位类命令减速时间的规格和设定步骤

运动模块	规格	设定步骤
SVA-01、PO-01、SVR	轴会按照运动设定参数 OLxx38“ 直线减速度 / 减速时间常数 ” 的减速时间执行动作。	将减速时间设定成运动设定参数 OLxx38“ 直线减速度 / 减速时间常数 ”。
SVB-01、内置 SVB	轴会按照伺服单元用户常数的减速度执行动作。	将减速时间设定成运动设定参数 OLxx38“ 直线减速度 / 减速时间常数 ”。之后，为了将减速时间应用到伺服单元，会通过运动设定参数 OWxx08“ 运动命令 ” 执行 “ 更改直线减速时间常数 ”(=11) 命令。(*1)

*1. 内置 SVB/SVB-01 具有将运动设定参数 OLxx38“ 直线减速度 / 减速时间常数 ” 自动应用到伺服单元用户常数减速度的功能。启用了执行自动应用的功能时，不需要通过运动设定参数 OWxx08“ 运动命令 ” 执行 “ 更改直线减速时 ”=11)。执行自动应用的功能时，请参阅《机器控制器 MP2000 系列运动模块 SVB/SVB-01 用户手册》(资料编号：SIJP C880700 33) （日文） 中的 “11.6 自动反映的用户常数 / 参数 ”。

(b) 加速时间和减速时间

下表中的运动模块与伺服单元组合时，无法分别设定加速时间和减速时间。通过设定加速时间来设定减速时间。除以下伺服单元外，可使用 ACC 命令和 DCC 命令分别设定加速时间和减速时间。

运动模块	伺服单元	备注
SVB-01、内置 SVB	SGD-N	• 内置 SVB/SVB-01 时，轴会按照伺服单元用户常数的加减速度执行动作。 • 伺服单元 SGD-N、SGDB-N 的加速时间 / 减速时间的用户常数为通用。
	SGDB-N	

8.1.5 S 字时间常数变更 （SCC）

运动程序	顺序程序
○	×

(1) 动作说明

S 字时间常数变更 （SCC）是更改轴移动命令的各轴的 S 字时间常数的命令。
S 字时间常数是指抑制加减速时机械系振动的 S 字加减速功能的参数。
同时最多可更改 16 轴。省略该指令的轴的 S 字时间常数不会更改。
已更改轴的 S 字时间常数会一直保持，直到通过 S 字时间常数变更 （SCC）重新进行设定。

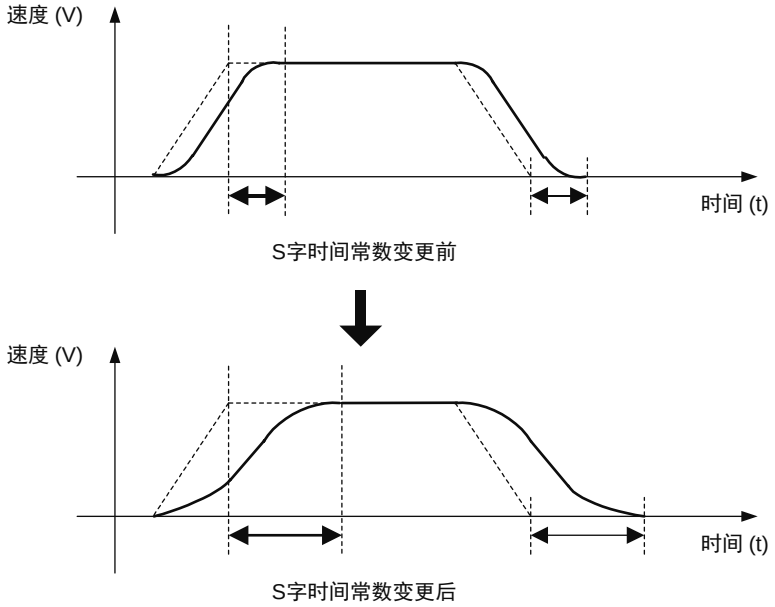


图 8.11 S 字时间常数变更

补充

ACC/DCC/SCC 命令支持所有的运动模块。
此外，与 PO-01 模块的组合支持以下版本。

MP2000 系列版本	PO-01 版本
MP2000 系列 Ver.2.46 或更高	PO-01 Ver.1.06 或更高

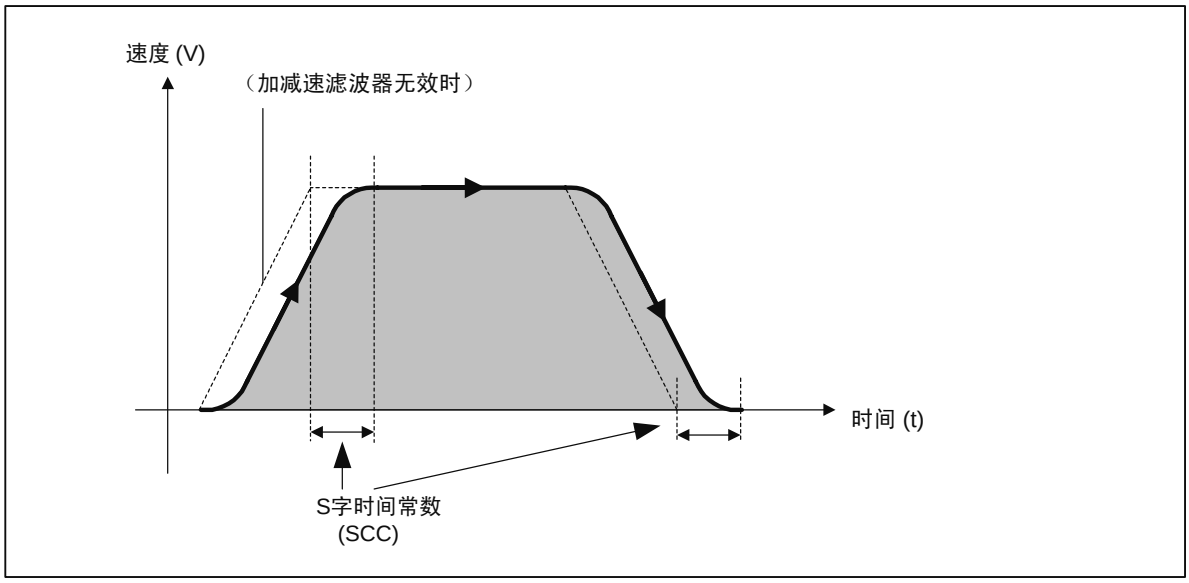
(2) 格式

SCC [逻辑轴名称1] S字时间常数 [逻辑轴名称2] S字时间常数・・・;

项目	单位	可使用的数据
S字 时间常数	ms	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

(3) SCC 命令的设定项目

动作示意图



各轴的 S 字时间常数在 SCC 命令中通过数值或寄存器发出指令。
S 字时间常数的指令范围取决于所使用的运动模块类型。

- SVA-01、PO-01 及 SVR 遵照运动设定参数 OWxx3A “ 滤波器时间常数 ” 的指令范围。
- 内置 SVB/SVB-01 遵照伺服单元用户常数 “ 移动平均时间 ” 的指令范围。

S 字时间常数的指令范围详细情况见下表。

运动模块	SCC 命令指令范围 [ms]	备注
SVA-01	0 ～ 6553	
SVB-01、内置 SVB	0 ～ 510	伺服单元为 SGD-N、SGDB-N、SGDH+NS110/NS115、SGDS、SGDX、SGDV 时
	–	伺服单元为 SGDJ 时，由于伺服单元用户常数中没有 “ 移动平均时间 ”，无法利用 S 字加减速
PO-01	0 ～ 6553	
SVR	0 ～ 6553	

补充

- 执行超过 6553 ms 的指令时，发生运动程序警报，与运动模块无关。
- 通过内置 SVB/SVB-01 模块执行超过上限值 (511 ～ 6553 ms) 指令时，变成运动监视器参数 ILxx02 Bit1 ＝ 1“ 设定参数设定故障 ”，请在伺服单元用户常数 “ 移动平均时间 ” 中设定上限值 (510 ms)。

(4) 程序示例

以下是 SCC 命令的程序示例。

例

表示执行 S 字时间常数 250 ms 的 MOV 命令和 S 字时间常数 500 ms 的 MOV 命令的示例。
运行该程序时，采用以下参数设定。

- 运动设定参数 OWxx03 bit0 ～ 3 “ 选择速度单位 ” ＝ 0 （指令单位 /s）
- 运动设定参数 OWxx03 bit4 ～ 7 “ 选择加减速度单位 ” ＝ 0 （指令单位 /s^2）

```
INC;                    “增量模式
VEL [A1]10000;          “进给速度变更 [ 指令单位 /S]
ACC [A1]20000;          “加速度变更 [ 指令单位 /S*S]
DCC [A1]20000;          “减速度变更 [ 指令单位 /S*S]
SCC [A1]250;            “S 字时间常数变更 [ms]
MOV [A1]20000;          “定位
DL00000 = 500;          “S 字时间常数 [ms]
SCC [A1]DL00000;        “S 字时间常数变更 [ms]
MOV [A1]20000;          “定位
END;
```

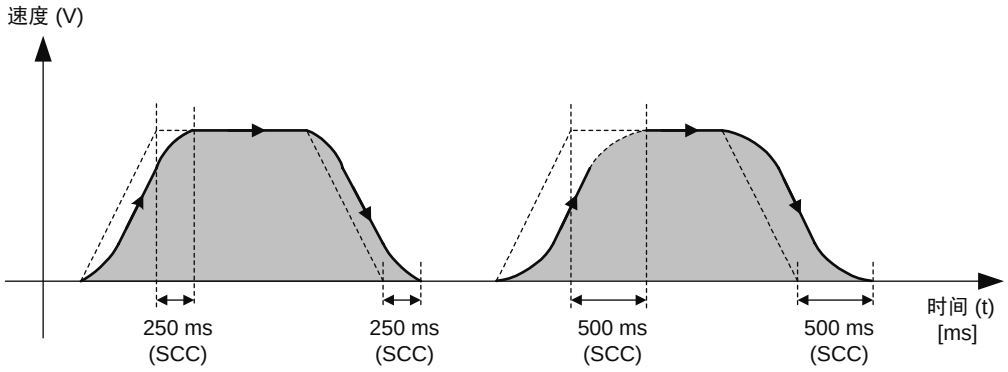


图 8.12“S 字时间常数变更 ” 程序示例

(5) SCC 命令的追加事项

(a) 相关运动参数

SCC 命令会更改运动设定参数的 S 字时间常数。

参数名称	寄存器编号	说明
滤波器时间常数	OWxx3A	设定加减速滤波器时间常数 (1=0.1ms)。 • 滤波器时间常数更改之前应先确认已变成 “ 输出完成状态 (IWxx0C Bit0=1) ”。 • 先通过 “ 选择滤波器类型 ” (OWxx03 Bit8 ～ B) 选择要使用的滤波器类型，再更改时间常数。

S 字时间常数无需通过 SCC 命令，可直接通过更改运动设定参数 OWxx3A“ 滤波器时间常数 ” 进行更改。有关直接更改 S 字时间常数的步骤，请参阅下表。

（参考）S 字时间常数的规格和设定步骤

运动模块	规格	设定步骤
SVA-01、PO-01、SVR	启用 S 字加减速度时，轴会按照运动设定参数 OWxx3A“ 滤波器时间常数 ” 中的 S 字时间常数执行动作。	将 S 字时间常数设定成运动设定参数 OWxx3A“ 滤波器时间常数 ”。
SVB-01、内置 SVB	启用 S 字加减速度时，轴会按照伺服单元用户常数中的移动平均滤波器时间常数执行动作。	将 S 字时间常数设定成运动设定参数 OWxx3A“ 滤波器时间常数 ”。之后，为了将 S 字时间常数应用到伺服单元，会通过运动设定参数 OWxx08“ 运动命令 ” 执行 “ 更改滤波器时间常数 ”(=12)。(*1)

*1. 内置 SVB/SVB-01 具有将运动设定参数 OWxx3A“ 滤波器时间常数 ” 自动应用到伺服单元用户常数中的移动平均滤波器时间常数的功能。启用了执行自动应用的功能时，不需要通过运动设定参数 OWxx08“ 运动命令 ” 执行 “ 更改滤波器时间常数 ” 命令 (=12)。执行自动应用的功能时，请参阅 《机器控制器 MP2000 系列运动模块 SVB/SVB-01 用户手册》(资料编号：SIJP C880700 33) （日文） 中的 “11.6 自动反映的用户常数 / 参数 ”

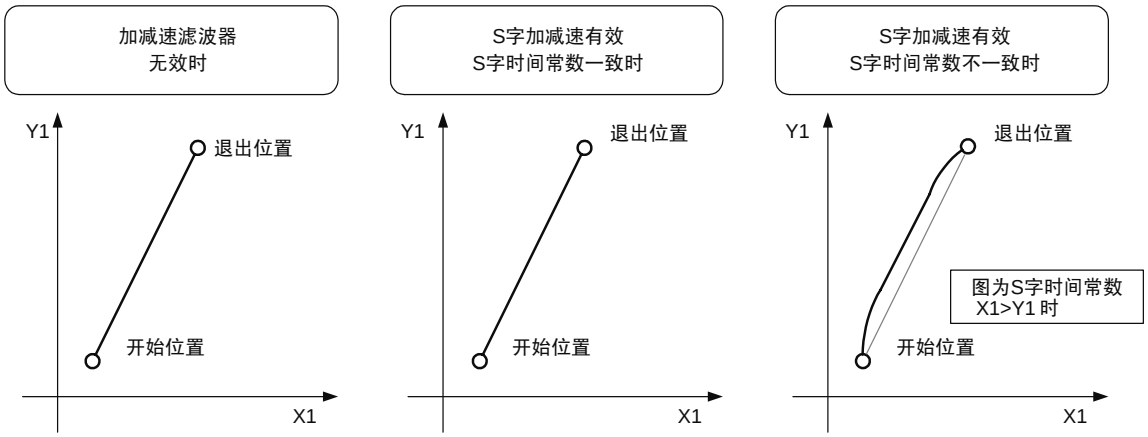
(b) 插补的移动轨迹和 S 字加减速

S 字加减速会影响到插补类命令（MVS、MCW、MCC、SKP）的移动轨迹。

- 如果想在直线插补时获得与 S 字加减速无效时相同的移动轨迹，应令插补对象轴的 S 字时间常数全部保持一致。
- 圆弧插补中如果启用了 S 字加减速，则不会变成与 S 字加减速无效时相同的移动轨迹。

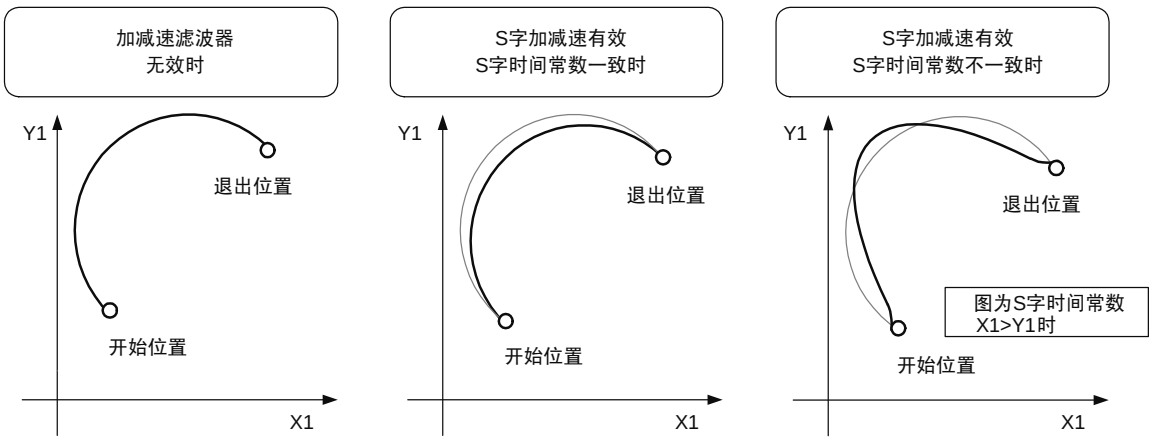
· 直线插补的移动轨迹

例



· 圆弧插补的移动轨迹

例



(c) 滤波器类型选择

启用了 S 字加减速时，请将各轴的滤波器类型预先通过运动设定参数 OWxx03 Bit8 ～ B“ 选择滤波器类型 ” 设定成 “2：移动平均滤波器 ”。

参数名称	寄存器编号	滤波器类型
功能设定 1 滤波器类型选择	OWxx03 Bit8 ～ B	0：无滤波器 （默认） 1：指数函数加减速滤波器 2：移动平均滤波器

使用内置 SVB/SVB-01 模块将 “ 自动写入 ” 伺服单元设定成无效时，请通过运动设定参数 OWxx08“ 运动命令 ” 执行 “ 更改滤波器类型 ”（ = 13），使其应用到伺服单元中。
下面是通过运动程序更改滤波器类型的程序示例。

（注）使用 SVA-01，PO-01，SVR 模块时，不需要以下程序。
此外，即使在使用内置 SVB/SVB-01 模块时，已将 “ 自动写入 ” 伺服单元设定成有效，仍然不需要以下程序。

```
“ 确认更新滤波器类型后，状态是否变好
IOW IW8008 == 0;                “ 等待无运动命令的状态
IOW IB800C0 == 1;               “ 等待输出完成状态

“ 滤波器类型选择移动平均滤波器
DW00000 = OW8003 & F0FFH;       “ 保持 “ 选择滤波器类型 ” 以外的信息
OW8003  = DW00000 | 0200H;      “ 滤波器类型＝移动平均滤波器

“ 将滤波器类型从内置 SVB/SVB-01 模块反映到伺服单元
OW8008 = 13;                    “ 请求 “ 更改滤波器类型 ”
IOW IW8008 == 13;               “ 等待变成 “ 更改滤波器类型 ” 处理
IOW IB80098 == 1;               “ 等待运动命令执行完成
OW8008 = 0 ;                    “ 清除请求
IOW IW8008 == 0;                “ 等待无运动命令的状态
```

补充

关于带有自动写入功能的内置 SVB/SVB-01 模块的伺服单元，请参阅 《机器控制器 MP2000 系列运动模块 SVB/SVB-01 用户手册》（资料编号：SIJP C880700 33） （日文）中的 “11.6 自动反映的用户常数 / 参数 ”

8.1.6 进给速度变更 （VEL）

运动程序	顺序程序
○	×

(1) 动作说明

进给速度变更 （VEL）是针对以下轴移动命令更改各轴的进给速度的命令。

- 定位 （MOV）
- 外部定位 （EXM）

本使用手册中，将上述轴移动命令和指定时间定位 （MVT）统称为 “ 定位类命令 ”，将其进给速度称为 “ 定位速度 ”。
同时最多可更改 16 轴。省略该指令的轴的定位速度不会更改。
已更改轴的定位速度会一直保持，直到通过进给速度变更 （VEL）重新进行设定，或通过指定时间定位 (MVT) 更改速度。

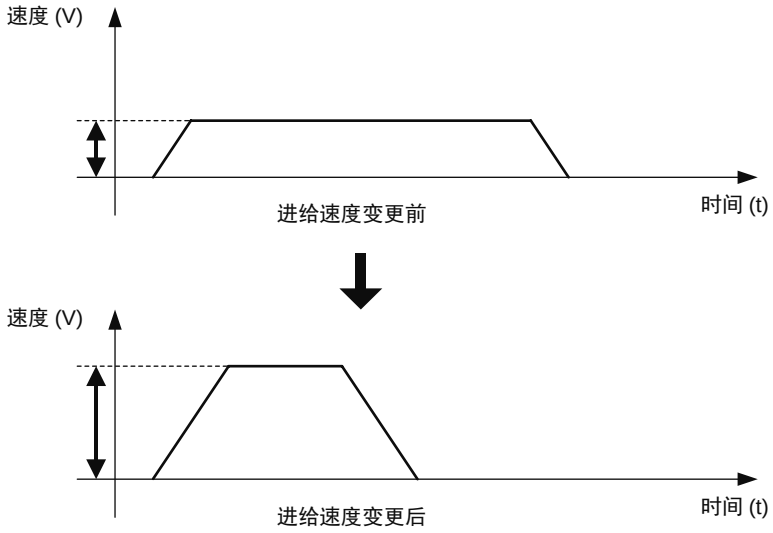


图 8.13 进给速度变更

补充

进给速度变更 (VEL) 是设定定位类命令 （MOV、EXM）定位速度的命令。插补类命令 （MVS、MCW、MCC、SKP）的进给速度通过 F 指令设定或 IFP 命令进行设定。

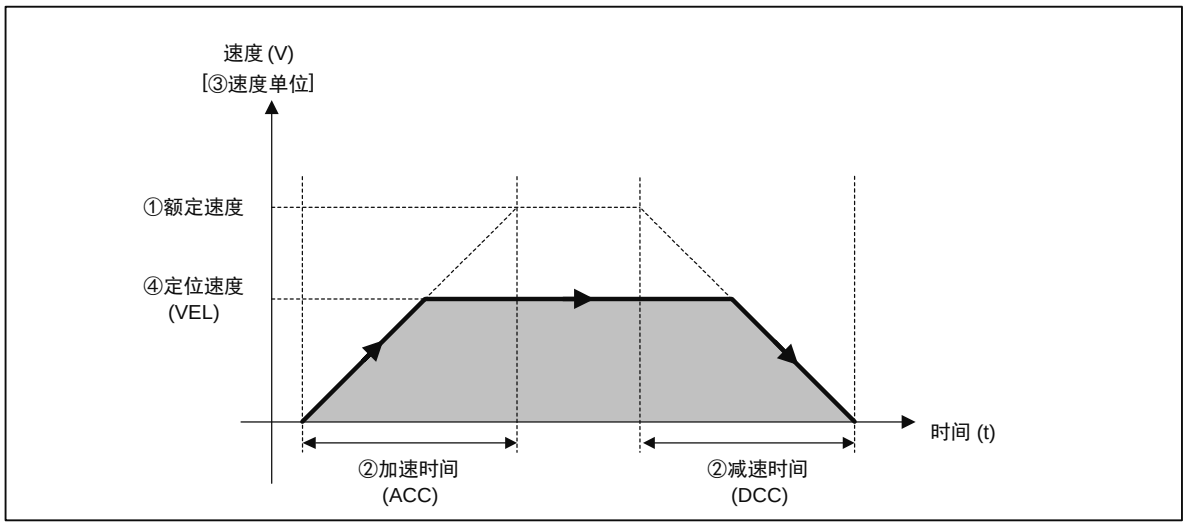
(2) 格式

VEL [逻辑轴名称1] 定位速度 [逻辑轴名称2] 定位速度 ・・・;

项目	单位	可使用的数据
定位速度	10^n 指令单位/min、 指令单位/s、 0.01%（指定额定速度的比率）、 0.0001%（指定额定速度的比率） ※ 单位通过运动设定参数OWxx03 Bit0～3 “选择速度单位”进行设定	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

(3) VEL 命令的设定项目

动作示意图



①额定速度

各轴的额定速度通过运动固定参数 No.34 “ 额定转数 ” 进行设定。
详细信息，请参阅对应运动模块的用户手册。

②加速时间 / 减速时间

各轴的加减速时间通过加速时间变更 (ACC) / 减速时间变更 (DCC) 进行设定。
通过 ACC 命令设定的时间变成达到额定速度前的加减速时间。

③速度单位

各轴的速度单位通过运动设定参数 OWxx03 Bit0 ～ 3 “ 选择速度单位 ” 进行设定。默认单位为 “10^n 指令单位 /min”。

参数名称	寄存器编号	速度单位	指令范围
功能设定 1 速度单位选择	OWxx03 Bit0 ～ 3	0：指令单位 /s	0 ～ 2^{31}-1[指令单位 /s]
		1：10^n 指令单位 /min	0 ～ 2^{31}-1[10^n 指令单位 /min]
		2：0.01% 指定	0 ～ 32767[0.01%]
		3：0.0001% 指定	0 ～ 3276700[0.0001%]

补充

选择 “10^n 指令单位 /min” 时 VEL 命令的设定单位取决于运动固定参数 No.4 “ 选择指令单位 ”。

运动固定参数 No.4 “ 指令单位选择 ”	速度单位 “10^n 指令单位 / min”	备注
pulse	1 = 1000 pulse/min	• “ 选择指令单位 ” = pulse 时 n = 3 • “ 选择指令单位 ” ≠ pulse 时 n = 运动固定参数 No.5 “ 小数点后位数 ”
mm	1 = 1 mm/min	
deg	1 = 1 deg/min	
inch	1 = 25.40 mm/min	
μm	1 = 1 μm/min	

④定位速度

各轴的定位速度在 VEL 命令通过数值或寄存器发出指令。

(4) 程序示例

以下是 VEL 命令的程序示例。

例

表示相对额定速度执行 40% 定位速度的 MOV 命令和相对额定速度执行 20% 定位速度的 MOV 命令的示例。

```
INC;                “ 增量模式
ACC [A1]5000;       “ 加速时间变更 [ms]
DCC [A1]5000;       “ 减速时间变更 [ms]
VEL [A1]4000;       “ 进给速度变更 [0.01%]
MOV [A1]3000000;    “ 定位
VEL [A1]2000;       “ 进给速度变更 [0.01%]
MOV [A1]3000000;    “ 定位
END;
```

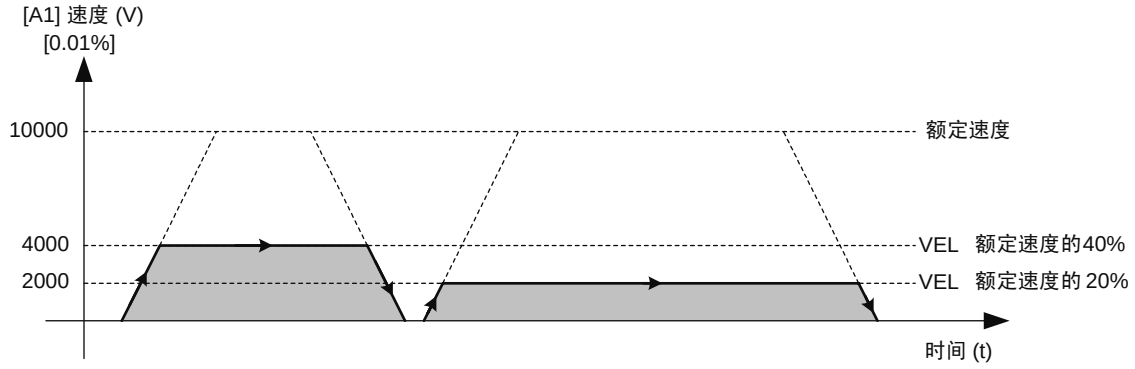


图 8.14 “ 进给速度变更 ” 程序示例
速度单位 “% 指定 （0.01% 单位）”

(5) VEL 命令的追加事项

(a) 相关运动参数

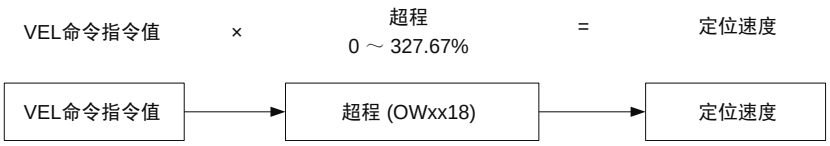
VEL 命令能够更改运动设定参数的定位速度。

参数名称	寄存器编号	说明
速度指令设定	OLxx10	设定速度指令值。

定位速度无需通过 VEL 命令，可直接通过更改运动设定参数 OLxx10 “ 设定速度指令 ” 进行更改。

(b) 关于超程

以 0.01% 为单位，可通过运动设定参数 OWxx18 “ 超程 ” 设定利用 VEL 命令发出指令的定位速度的 % （输出比率）。
运动设定参数 OWxx18 “ 超程 ” 的默认值为 10000(100.00%)。



术语？

■ 超程

英语中，“OVERRIDE” 有 “ 设定成无效 ” 的含义，但在这里，请解释为 “ 更改 ” 设定值。

运动设定参数 OWxx18“ 超程 ” 能够在轴移动过程中进行更改。

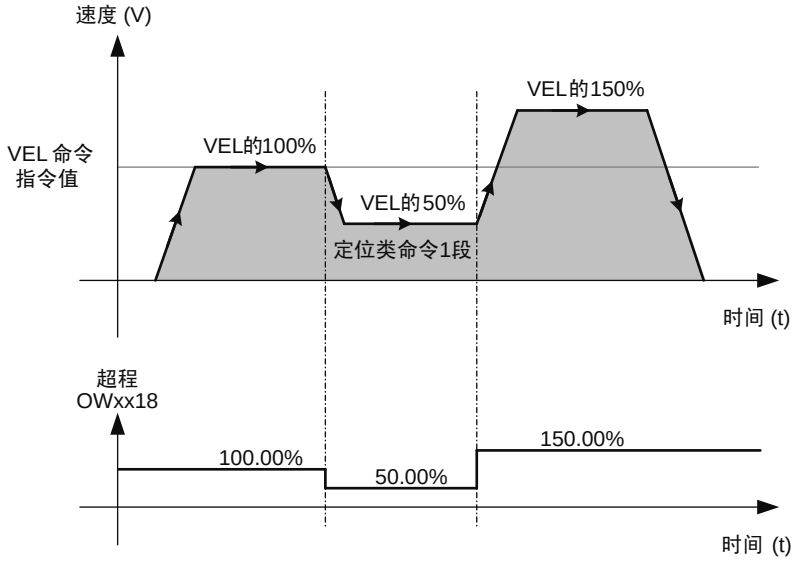


图 8.150Wxx18“ 超程 ” 和定位类命令

补充

- SVR 模块中没有运动设定参数 OWxx18“ 超程 ”。
- 指定时间定位 (MVT) 时，成为超程基准的定位速度非 VEL 命令的指令值。通过时间指定定位 (MVT) 更改的定位速度变成超程的基准速度。
- 指定时间定位 (MVT) 使用超程时，不会通过指定时间来完成定位。
  通过指定时间定位 (MVT)，MP2000 系统执行的定位速度的运算作为超程 100% 执行。
- 使用固定参数指定的额定速度与运动程序处理的定位速度 （VEL）其速度单位不同。

速度	速度单位
运动固定参数 No.34“ 额定转数 ”	转 /min
定位速度 (VEL)	指令单位 /s，10^n 指令单位 /min，0.01%，0.0001%

计算与定位速度 (VEL) 的速度单位相匹配的额定速度时，请参阅 “(c) 关于电机的速度规格 ”。

(c) 电机的速度规格

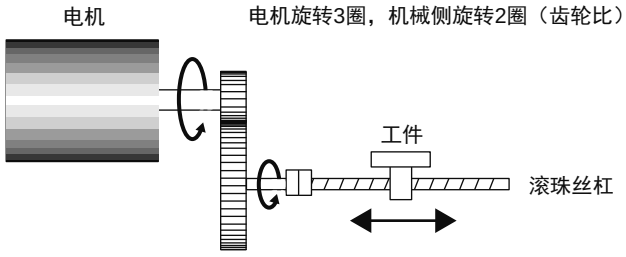
设定 VEL 命令的设定值时，除 VEL 命令的指令范围外，还需考虑电机的额定速度 / 最高速度。设定前请首先确认使用的电机的速度规格，速度不能过大。

补充

电机类型为旋转型时，电机的速度规格表述为每单位时间的转数。
速度单位 10^n 指令单位 /min 时的额定速度，通过运动固定参数进行如下计算。

· 参数设定示例（电子齿轮有效时）
（注）运动固定参数 No.4 “ 选择指令单位 ” 选择了 pulse 以外时，电子齿轮有效。

运动固定参数
No.4 ： 选择指令单位 = mm
No.5 ： 小数点后位数 = 3 位
No.6 ： 机械每旋转 1 次的移动量 = 10000 指令单位
No.8 ： 电机侧齿轮比 = 3
No.9 ： 机械侧齿轮比 = 2
No.34：额定转数 = 3000 转 /min



旋转1圈，平台移动10mm
（机械每旋转1圈的移动量）

电子齿轮有效时，速度单位 [10^n 指令单位 /min] 的 “n” 表示小数点后面的位数，因此，速度单位
[10^n 指令单位 /min] = [10^3 · 0.001mm/min] = [mm/min]

电机按额定转数动作时机械侧的转速为
额定转数 [转数 /min] × 齿轮比
= 3000×(2/3) = 2000 [转 /min]

如果将机械侧的转数转换成指令单位 (0.001mm)，则
机械每旋转 1 次的移动量 [0.001mm/ 转]×2000 [转 /min]
= 10000×2000 = 20000000[0.001mm/min]

假设速度单位为 [mm/min]，则
20000000[0.001mm/min] = 20000[mm/min]

· 参数设定示例（电子齿轮无效－ SVA-01 时）

（注）运动固定参数 No.4 “ 选择指令单位 ” 选择了 pulse 时，电子齿轮无效。

运动固定参数
No.4 ： 选择指令单位 = pulse
No.22：脉冲计数方式＝ A/B×4 （→ 4 倍频）
No.34：额定转数 = 3000 转 /min
No.36：电机旋转 1 次的脉冲数（倍频前）= 16384 pulse / 转

电子齿轮无效时，速度单位 [10^n 指令单位 /min] 的 “n” 是 3，因此，速度单位

[10^n 指令单位 /min] = [10^3 pulse/min] = [1000pulse/min]

如果将电机的额定转数转换成 pulse 单位，则

额定转数 [转数 /min] ×（电机旋转 1 次的脉冲数 [pulse/ 旋转]× 倍频）
＝ 3000×(16384×4) ＝ 196608000[pulse/min]

假设速度单位为 [1000pulse/min]，则

196608000[pulse/min] ＝ 196608[1000pulse/min]

· 参数设定示例（电子齿轮无效－内置 SVB、SVB-01、PO-01、SVR 时）

（注）运动固定参数 No.4 “ 选择指令单位 ” 选择了 pulse 时，电子齿轮无效。

运动固定参数
No.4 ： 选择指令单位 = pulse
No.34：额定转数 = 3000 转 /min
No.36：电机旋转 1 次的脉冲数 = 65536 pulse/ 转

电子齿轮无效时，速度单位 [10^n 指令单位 /min] 的 “n” 是 3，因此，速度单位

[10^n 指令单位 /min] = [10^3 pulse/min] = [1000pulse/min]

如果将电机的额定转数转换成 pulse 单位，则

额定转数 [转数 /min] × 电机旋转 1 次的脉冲数 [pulse/ 转]
＝ 3000×65536 ＝ 196608000[pulse/min]

假设速度单位为 [1000pulse/min]，则

196608000[pulse/min] ＝ 196608[1000pulse/min]

为了实现轴正常工作，需对参数设定示例中未列出的运动固定参数进行正确设定。
关于各参数的详细信息及与机械匹配的参数设定，请参阅相应运动模块的使用手册。

8.1.7 插补进给最高速度设定 （FMX）

运动程序	顺序程序
○	×

(1) 动作说明

设定插补进给最高速度 （FMX）是设定插补类命令 （MVS、MCW、MCC、SKP）最高速度的命令。已设定的插补进给最高速度会一直保持到通过设定插补进给最高速度 （FMX）进行重新设定。
程序运行开始时，插补进给最高速度处于未设定状态。FMX 命令在执行以下与插补相关的各种命令之前需进行预执行。

- 直线插补 （MVS）
- 圆弧插补 （MCW、MCC）
- 螺旋插补 （MCW、MCC）
- 带跳过功能的直线插补 （SKP）
- 插补进给速度比率设定 （IFP）
- 更改插补加速时间 （IAC）
- 更改插补减速时间 （IDC）

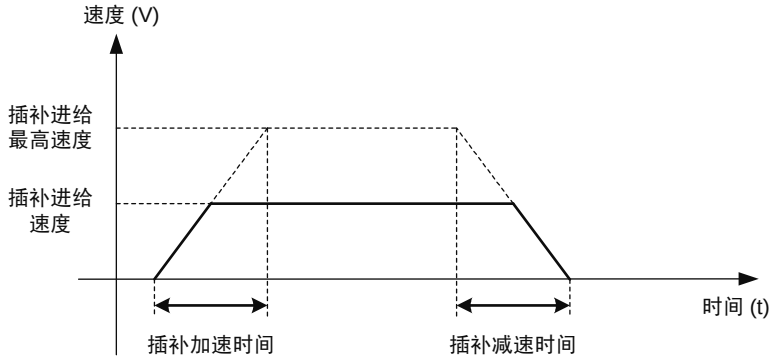


图 8.16 插补进给最高速度设定

补充

- 与插补相关的各种命令执行处理的前提是事先设定了插补进给最高速度。例如，更改插补加速时间 (IAC) 及更改插补减速时间 (IDC) 变成指定从速度 0 达到插补进给最高速度的时间的命令。
- 不发出 FMX 命令的指令，执行了与插补相关的各种命令 (MVS、MCW、MCC、SKP、IFP、IAC、IDC) 时，发生运动程序警报。

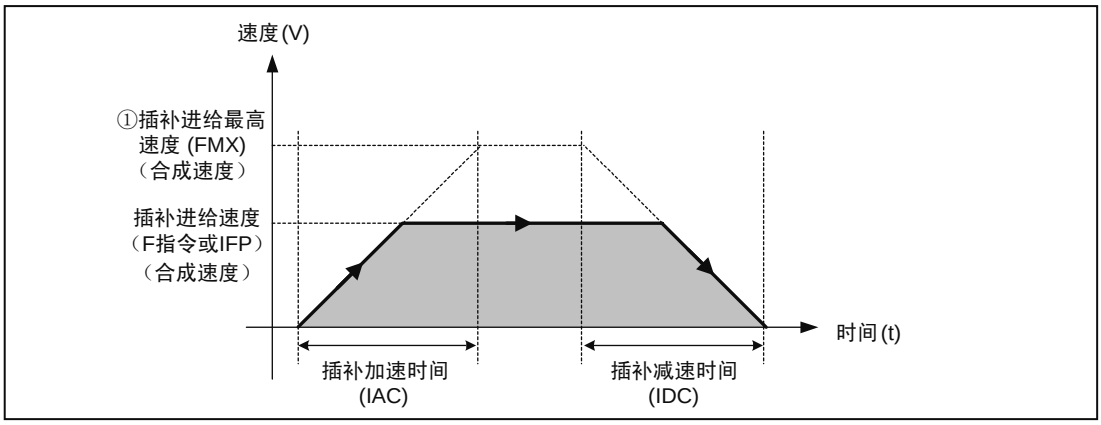
(2) 格式

FMX T插补进给最高速度;

项目	单位	可使用的数据
插补进给 最高速度	指令单位/min	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

(3)FMX 命令的设定项目

动作示意图



①插补进给最高速度

插补进给最高速度使用 FMX 命令，通过字符 “T” 后面的数值或寄存器发出指令。插补进给最高速度的指令范围会发生如下变化。

1 ～ 2^{31}-1 [指令单位 /min]

插补进给最高速度是与插补相关的所有命令可以利用的控制数据。因此，使用插补类命令 （MVS、MCW、MCC、SKP）时，需在运动程序的起始位置发出 FMX 命令的指令。

(4) 程序示例

例 以下是 FMX 命令的程序示例。

```
INC;                                  “ 增量模式
FMX T300000;                          “ 设定插补进给最高速度
IAC T4000;                            “ 更改插补加速时间 [ms]
IDC T4000;                            “ 更改插补减速时间 [ms]
IFP P75;                              “ 设定插补进给速度比率 [%]
MVS [A1]30000 [B1]30000;              “ 直线插补
MVS [A1]30000 [B1]30000 F150000;      “ 直线插补 (F 指令 )
END;
```

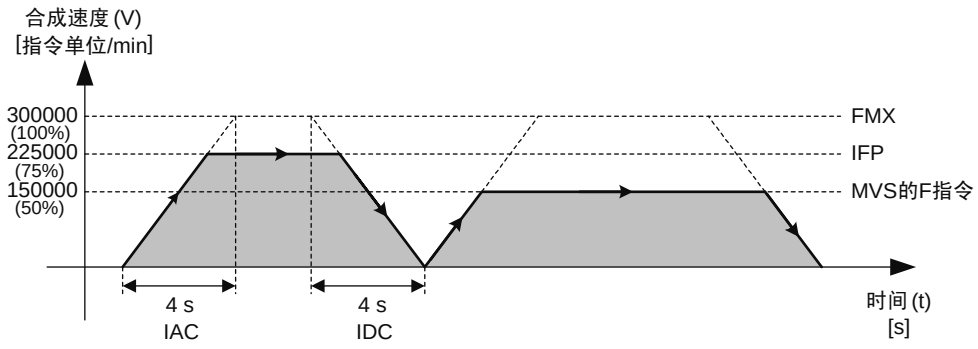


图 8.17 “ 设定插补进给最高速度 ” 程序示例

(5)FMX 命令的追加事项

(a) 相关运动参数

FMX 命令与运动设定参数没有相关性。
通过 FMX 命令指定的插补进给最高速度为运动程序专用的控制数据，无法通过运动设定参数指定。

8.1.8 插补进给速度比率设定 （IFP）

运动程序	顺序程序
○	×

(1) 动作说明

插补进给速度比率设定 （IFP）是设定以下轴移动命令进给速度的命令。以相对于插补进给最高速度的比率指定进给速度。

- 直线插补 （MVS）
- 圆弧插补 （MCW、MCC）
- 螺旋插补 （MCW、MCC）
- 带跳过功能的直线插补 （SKP）

本使用手册中，将上述轴移动命令统称为 “ 插补类命令 ”，将其进给速度称为 “ 插补进给速度 ”。已设定的插补进给速度会一直保持，直到重新设定插补进给速度比率或发出基于插补类命令的 F 指令。程序运行开始时，插补进给速度处于未设定状态。执行插补类命令时，请先执行设定插补进给速度比率（IFP）或 F 指令，再设定插补进给速度。

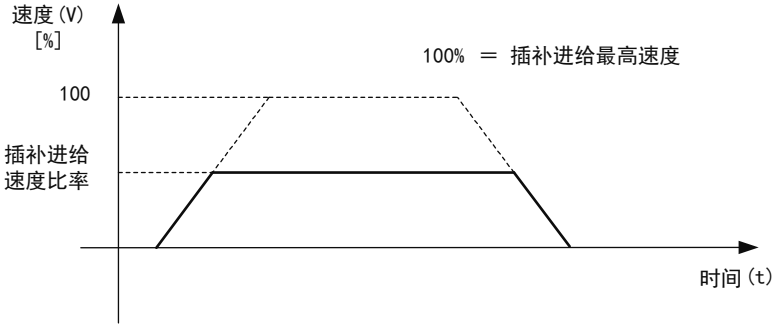


图 8.18 插补进给速度比率设定

补充

- 请不要预先指定插补进给最高速度 （FMX）。没有 FMX 命令指定时，IFP 命令会导致发生运动程序警报。
- F 指令是使用插补类命令指定字符 “F” 后面的数值或使用寄存器指定插补进给速度的方法。使用 “ 指令单位 /min” 指定插补进给速度。
- 如果在 F 指令后执行 IFP 命令，则使用 F 指令指定的插补进给速度会被取消。
  此外，如果相反在 IFP 命令后执行 F 指令，则使用 IFP 命令指定的插补进给速度会被取消。
- 在未指定插补进给速度的情况下执行插补类命令时，会发生运动程序警报。
- 插补进给速度比率设定 （IFP）是设定插补类命令 （MVS、MCW、MCC、SKP）进给速度的命令。定位类命令（MOV、EXM）的进给速度通过 VEL 命令进行设定。

(2) 格式

IFP　P插补进给速度比率;

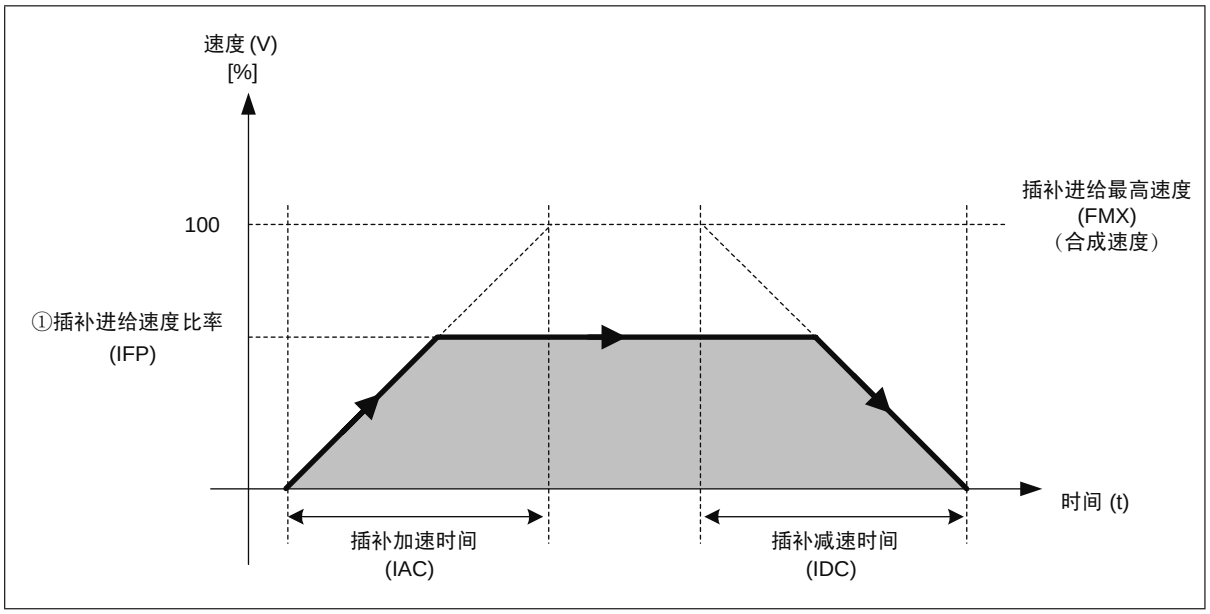
项目	单位	可使用的数据
插补进给速度比率	%	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

补充

无法在与插补类命令 (MVS、MCW、MCC、SKP) 相同的程序段中指定 IFP 命令。

(3) IFP 命令的设定项目

动作示意图



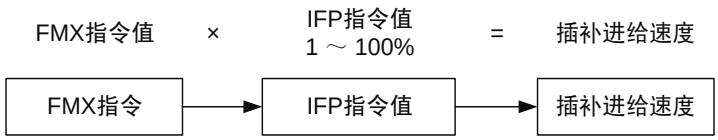
①插补进给速度比率

插补进给速度比率使用 IFP 命令，通过字符 “P” 后面的数值或寄存器发出指令。
使用 IFP 命令设定的时间变成相对于插补进给最高速率的插补进给速度的比率。
插补进给速度是使用插补类命令（MVS、MCW、MCC、SKP）发出指令的所有轴的合成速度。
插补进给速率比率的指令范围会发生如下变化。

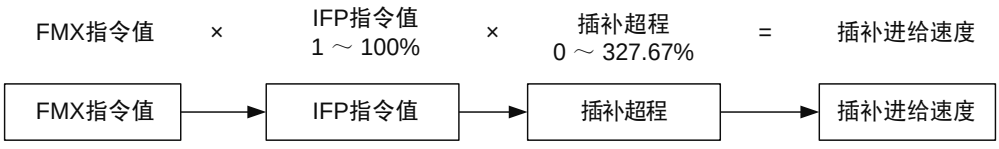
1 ～ 100[%]

还可以选择是否在插补进给速度上施加插补超程。
插补超程的使用方法，请参阅 “4.3.3 任务寄存器 ”。

・ 不使用插补超程时



・ 使用插补超程时



补充

- 插补进给速度的指定可以通过 IFP 指令或者 F 指令的指定实现。
  详细内容，请参阅 “8.2.2 直线插补 （MVS）(3) MVS 命令的设定项目② 插补进给速度 ”。
- 在 IFP 指令值 [%] 中指定了超过 100[%] 的值时，发生运动程序警报。
- 利用插补超程发出超过 FMX 指令值的插补进给速度时，插补进给速度的输出值变成 FMX 指令值。

(4) 程序示例

例 以下是 IFP 命令的程序示例。

```
INC;                          “ 增量模式
FMX T300000;                  “ 设定插补进给最高速度 [ 指令单位 /min]
IAC T4000;                    “ 插补加速时间变更 [ms]
IDC T4000;                    “ 插补减速时间变更 [ms]
IFP P75;                      “ 插补进给速度比率设定 [%]
MVS [A1]30000 [B1]30000;      “ 直线插补
DL00000 = 50;                 “ 插补进给速度比率 [%]
IFP PDL00000;                 “ 插补进给速度比率设定 [%]
MVS [A1]30000 [B1]30000;      “ 直线插补
END;
```

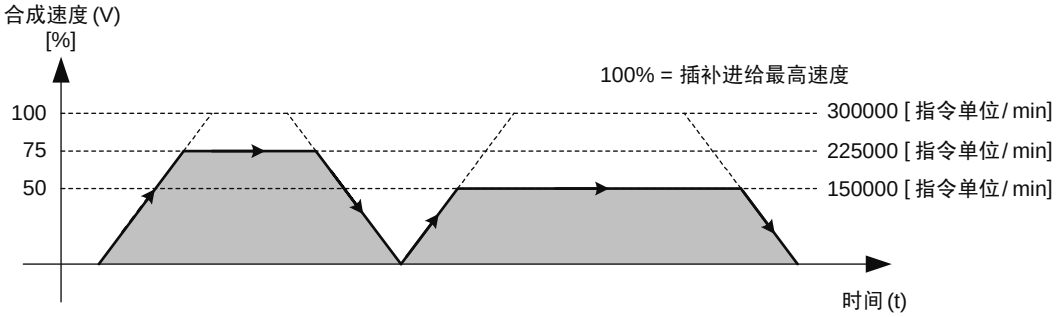


图 8.19 “ 插补进给速度比率设定 ” 程序示例

(5) IFP 命令的追加事项

(a) 相关运动参数

IFP 命令与运动设定参数没有相关性。
通过 IFP 命令指定的插补进给速度比率为运动程序专用的控制数据，无法通过运动设定参数指定。

8.1.9 插补加速时间变更 （IAC）

运动程序	顺序程序
○	×

(1) 动作说明

插补加速时间变更 （IAC）是更改以下轴移动命令加速时间的命令。

- 直线插补 （MVS）
- 圆弧插补 （MCW、MCC）
- 螺旋插补 （MCW、MCC）
- 带跳过功能的直线插补 （SKP）

本手册将以上轴移动命令统称为 “ 插补类命令 ”。
执行 IAC 命令之前，需预先执行设定插补进给最高速度 （FMX）命令。更改后的加速时间会一直保持，直到通过插补加速时间变更 （IAC）指令重新进行设定。
程序运行开始时，插补加速时间为 0 ms。

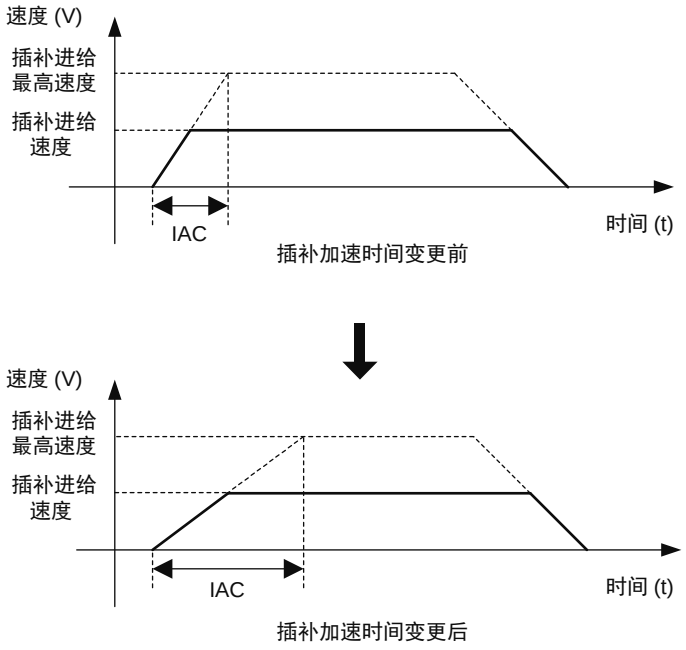


图 8.20 插补加速时间变更

补充

插补加速时间变更 (IAC) 是设定插补类命令 （MVS、MCW、MCC、SKP）加速时间的命令。
定位类命令 （MOV、EXM、MVT）的加速时间通过 ACC 命令进行设定。

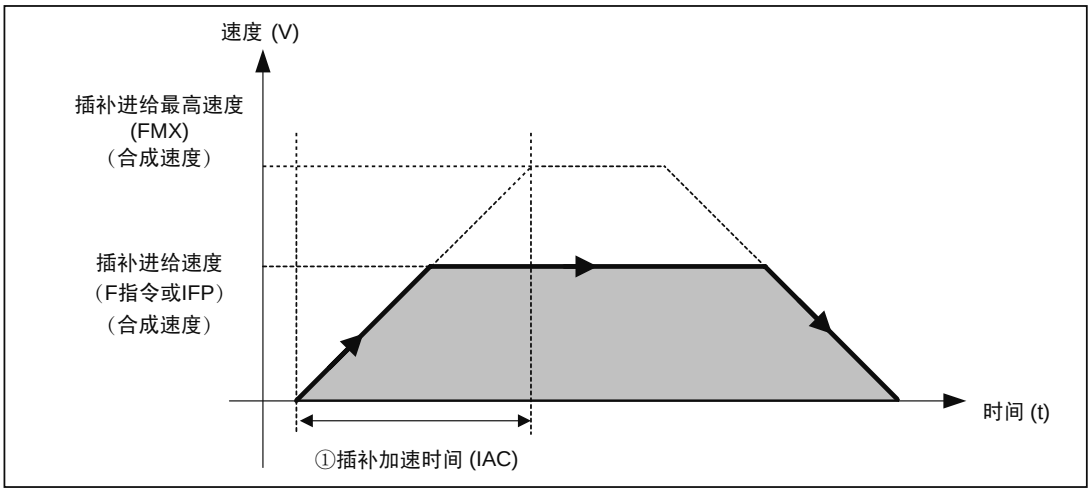
(2) 格式

IAC T插补加速时间 ;

项目	单位	可使用的数据
插补加速时间	ms	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

(3) IAC 命令的设定项目

动作示意图



①插补加速时间

插补加速时间使用 IAC 命令，通过字符 “T” 后面的数值或寄存器发出指令。
通过 IAC 命令设定的时间变成从速度 0 到插补进给最高速度所需的加速时间。
插补加速时间的指令范围会发生如下变化。

0 ～ 32767[ms]

(4) 程序示例

例 以下是 IAC 命令的程序示例。

```
INC;                                  “ 增量模式
FMX T300000;                          “ 设定插补进给最高速度 [ 指令单位 /min]
IDC T4000;                            “ 插补减速时间变更 [ms]
IAC T2000;                            “ 插补加速时间变更 [ms]
MVS [A1]30000 [B1]30000 F150000;      “ 直线插补
DL00000 = 4000;                       “ 插补加速时间 [ms]
IAC TDL00000;                         “ 插补加速时间变更 [ms]
MVS [A1]30000 [B1]30000;              “ 直线插补
END;
```

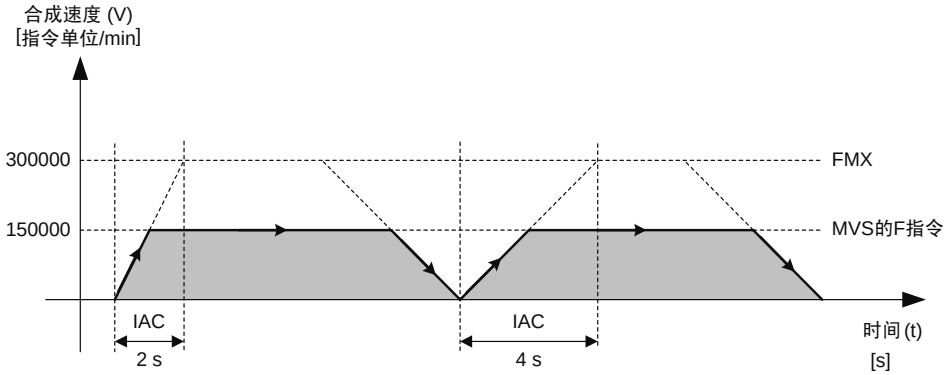


图 8.21 “ 插补加速时间变更 ” 程序示例

(5) IAC 命令的追加事项

(a) 相关运动参数

IAC 命令与运动设定参数无关。
通过 IAC 命令指定的插补加速时间为运动程序专用的控制数据，无法通过运动设定参数指定。

8.1.10 插补减速时间变更 （IDC）

运动程序	顺序程序
○	×

(1) 动作说明

插补减速时间变更 （IDC）是更改以下轴移动命令减速时间的命令。

- 直线插补 （MVS）
- 圆弧插补 （MCW、MCC）
- 螺旋插补 （MCW、MCC）
- 带跳过功能的直线插补 （SKP）

本手册将以上轴移动命令统称为 “ 插补类命令 ”。
执行 IDC 命令之前，需预先执行设定插补进给最高速度 （FMX）。更改后的减速时间会一直保持，直到通过插补减速时间变更 （IDC）重新进行设定。
程序运行开始时，插补减速时间为 0 ms。

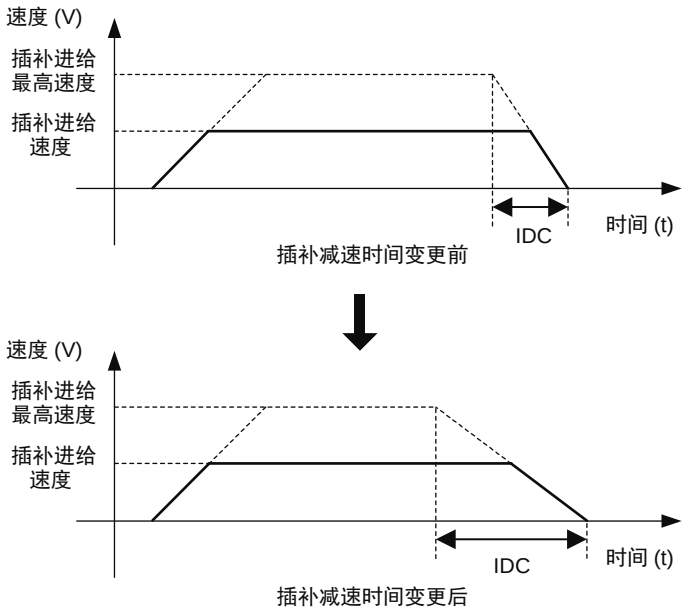


图 8.22 插补减速时间更改

补充

插补减速时间变更 (IDC) 是设定插补类命令 （MVS、MCW、MCC、SKP）减速时间的命令。
定位类命令 （MOV、EXM、MVT）的减速时间通过 DCC 命令进行设定。

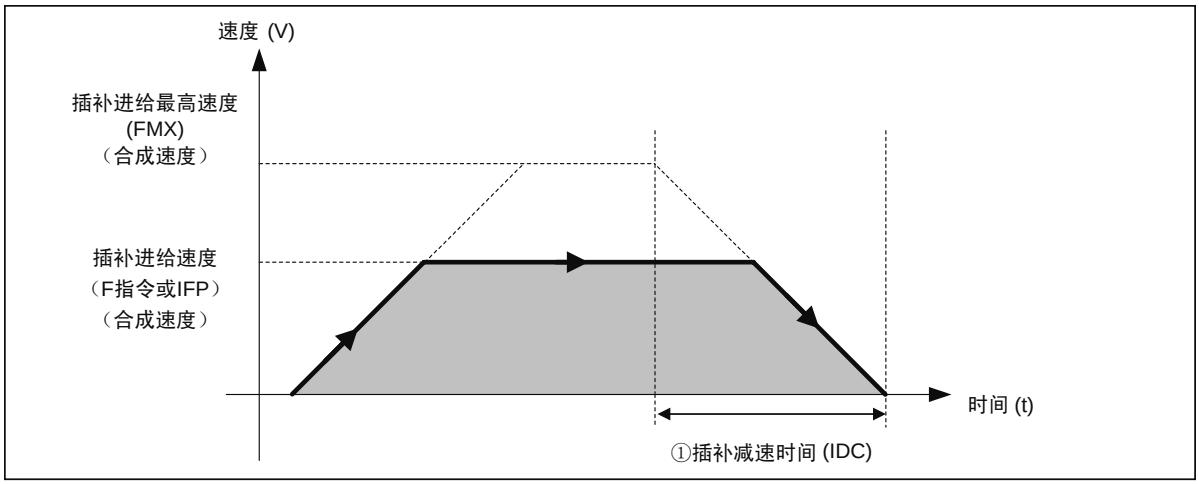
(2) 格式

IDC　T插补减速时间；

项目	单位	可使用的数据
插补减速时间	ms	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

(3) IDC 命令的设定项目

动作示意图



①插补减速时间

插补减速时间使用 IDC 命令，通过字符 “T” 后面的数值或寄存器发出指令。
通过 IDC 命令设定的时间变成从插补最高速度到速度 0 所需的减速时间。
插补减速时间的指令范围会发生如下变化。

0 ～ 32767[ms]

(4) 程序示例

例 以下是 IDC 命令的程序示例。

```
INC;                                  “ 增量模式
FMX T300000;                          “ 设定插补进给最高速度 [ 指令单位 /min]
IAC T4000;                            “ 插补加速时间变更 [ms]
IDC T2000;                            “ 插补减速时间变更 [ms]
MVS [A1]30000 [B1]30000 F150000;      “ 直线插补
DL00000 = 4000;                       “ 插补减速时间 [ms]
IDC TDL00000;                         “ 插补减速时间变更 [ms]
MVS [A1]30000 [B1]30000;              “ 直线插补
END;
```

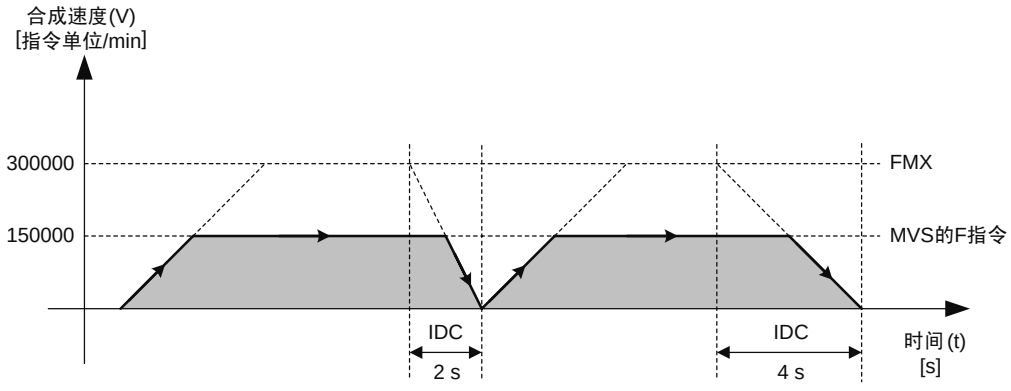


图 8.23 “ 插补减速时间变更 ” 程序示例

(5) IDC 命令的追加事项

(a) 相关运动参数

IDC 命令与运动设定参数无关。
通过 IDC 命令指定的插补减速时间为运动程序专用的控制数据，无法通过运动设定参数指定。

8.2 轴移动命令

在本节中，将对轴移动命令进行说明。

8.2.1 定位 （MOV）

运动程序	顺序程序
○	×

(1) 动作说明

定位 （MOV）是将各轴进行独立，以定位速度使其从程序当前位置移动到终点位置的命令。
同时最多可移动 16 轴。省略该指令的轴不移动。
基于 MOV 命令的移动轨迹不会变成与直线插补相同的直线移动。

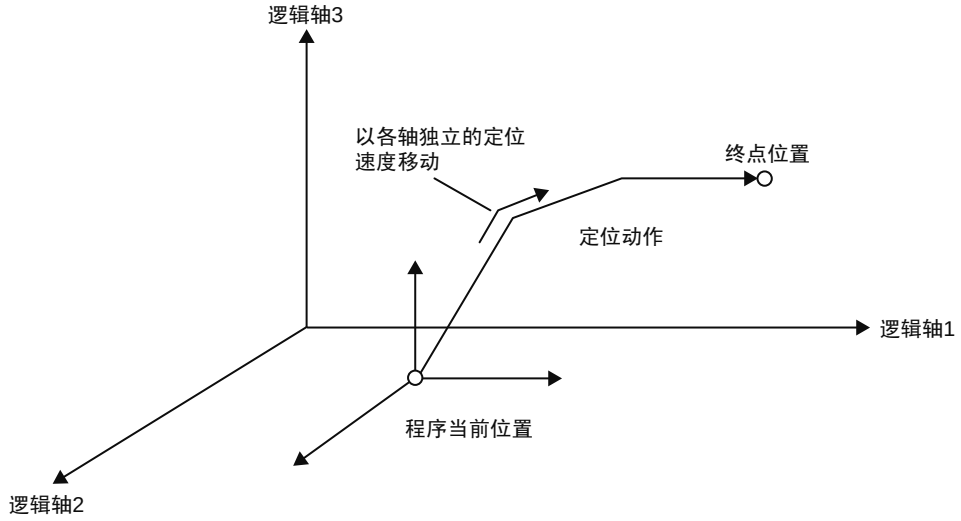


图 8.24 基于 MOV 命令的移动轨迹

⚠注意

・定位 （MOV）的移动轨迹不会变成直线。编程时，请务必检查轨迹，避免工具会干扰到工件等。
否则可能会因干扰造成工具损坏及由此导致人身事故。

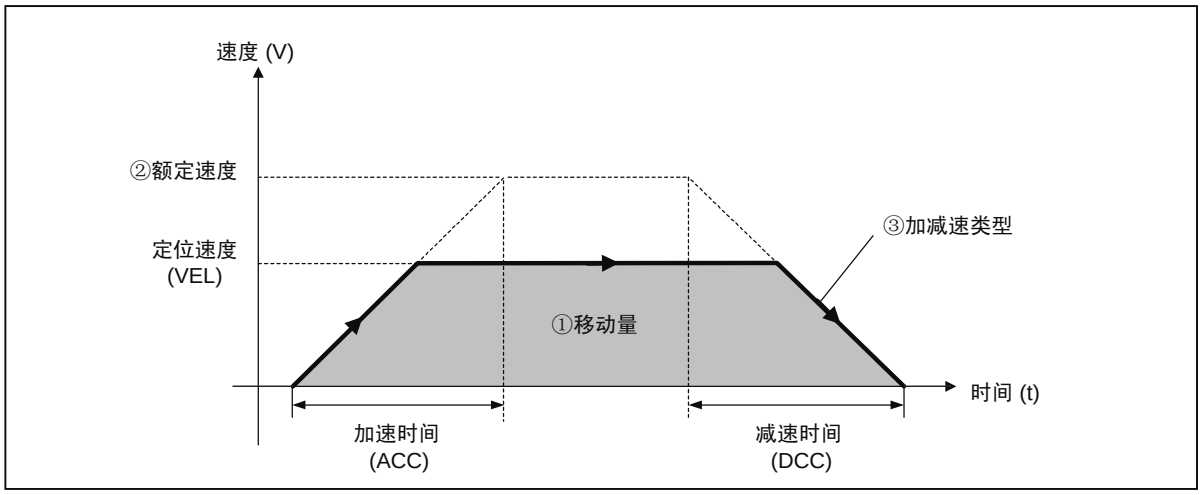
(2) 格式

MOV [逻辑轴名称1] 指令位置 [逻辑轴名称2] 指令位置 [逻辑轴名称3] 指令位置・・・;

项目	单位	可使用的数据
指令位置	指令单位	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

(3)MOV 命令的设定项目

动作示意图



①移动量

各轴的移动量取决于 ABS 或 INC 模式。

- ABS 模式的移动量

  ABS 模式下时，将从程序当前位置到指令位置的差作为移动量进行处置。

- INC 模式的移动量

  INC 模式下时，指令位置被直接视为移动量。

补充

关于移动量的单位，请参阅 “7.2.2 指令单位 ”。

②额定速度

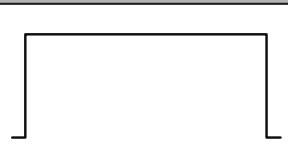
各轴的额定速度通过运动固定参数 No.34 “ 额定转数 ” 进行设定。
详细信息，请参阅对应运动模块的用户手册。

③加减速类型

MOV 命令的加减速类型可从以下 3 种类型中选择。
加减速类型通过 ACC/DCC/SCC 命令和运动设定参数 OWxx03 Bit4 ～ 7 “ 选择加减速单位 ” 及 OWxx03 Bit8 ～ B “ 选择滤波器类型 ” 的组合进行设定。

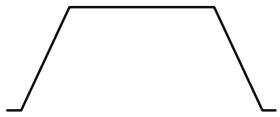
(a) 无加减速

加速时间、减速时间均是 0 的运动方式。

设定方法	动作示意图
• 设定成 OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” = 1(ms) • 设定成 OWxx03 Bit8 ～ B “ 选择滤波器类型 ” = 0(无滤波器) • 将 ACC 命令设定成 0 • 将 DCC 命令设定成 0	

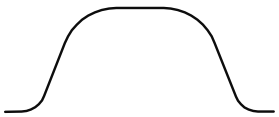
(b) 1 段直线加减速

以固定的加速度、减速度运动的方式。

设定方法	动作示意图
• 设定成 OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” = 1(ms) • 设定成 OWxx03 Bit8 ～ B “ 选择滤波器类型 ” = 0(无滤波器) • 将 ACC 命令设定成 0 以外的数值 • 将 DCC 命令设定成 0 以外的数值	

(c) S 字加减速

以 S 字的加速度、减速度运动的方式。

设定方法	动作示意图
• 设定成 OWxx03 Bit4 ～ 7 “ 加减速度单位选择 ” = 1(ms) • 设定成 OWxx03 Bit8 ～ B “ 选择滤波器类型 ” = 2(移动平均滤波器) • 将 ACC 命令设定成 0 以外的数值 • 将 DCC 命令设定成 0 以外的数值 • 将 SCC 命令设定成 0 以外的数值	

补充

针对基于 MOV 命令的轴移动，执行检查已进入定位完成范围的到位检查。到位检查后，执行下一个移动指令的程序段。
下图表示到位检查的动作。

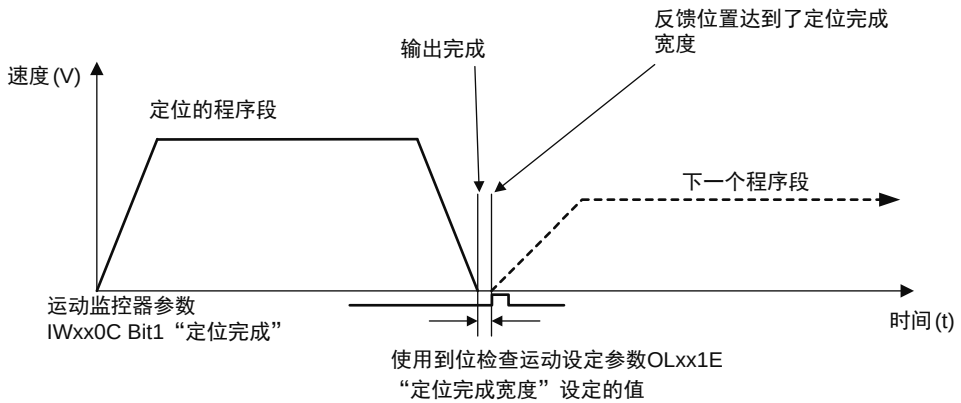


图 8.25 到位检查的动作

(4) 程序示例

例 以下是 ABS 模式下 MOV 命令的程序示例。

```
ABS;
ACC [A1]1000 [B1]1000 [C1]1000;
DCC [A1]1000 [B1]1000 [C1]1000;
VEL [A1]2000 [B1]2000 [C1]2000;
MOV [A1]4000 [B1]3000 [C1]2000;
END;
```

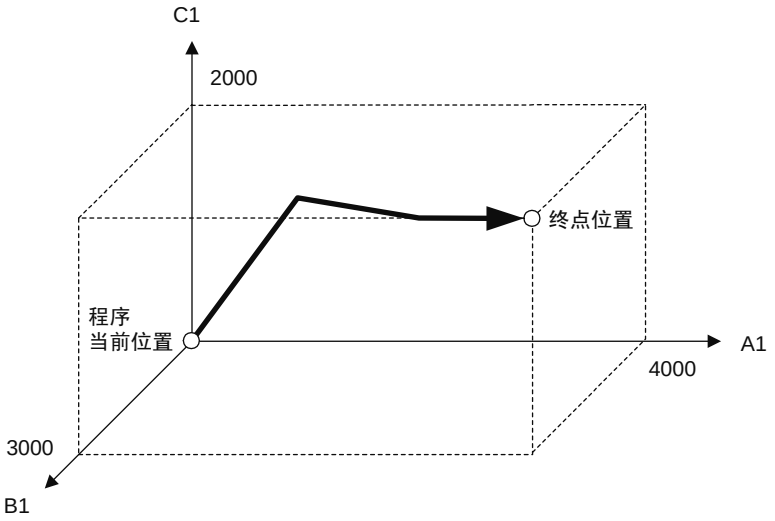


图 8.26 MOV 命令的程序示例

8.2.2 直线插补 （MVS）

运动程序	顺序程序
○	×

(1) 动作说明

直线插补 （MVS）是以插补进给速度使各轴以直线的方式从程序当前位置移动到终点位置的命令。同时最多可移动 16 轴。省略该指令的轴不移动。

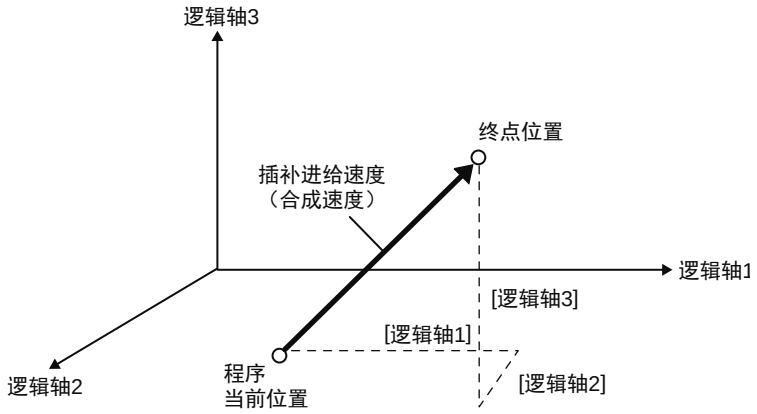


图 8.27 基于 MVS 命令的移动轨迹

⚠注意

・执行直线插补 （MVS）的轴可以是直线轴，也可以是旋转轴。包含旋转轴时，直线插补的轨迹不会变成“直线”。编程时，请务必检查轨迹，避免工具会干扰到工件等。
否则可能会因干扰造成工具损坏及由此导致人身事故。

补充

针对基于 MVS 命令的轴移动，不会检查是否已进入定位完成范围。需到位检查时，请使用 PEN 命令。

(2) 格式

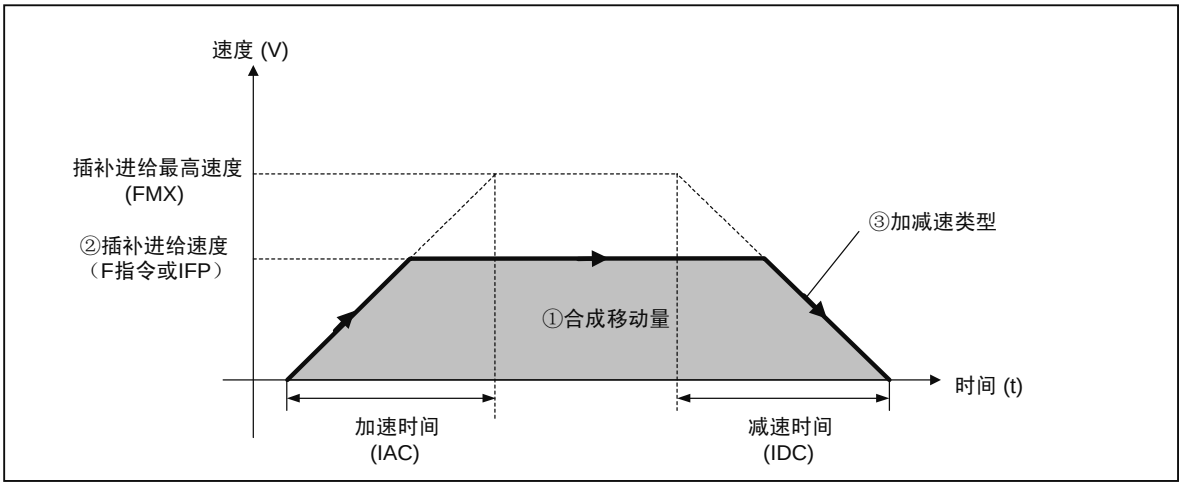
MVS [逻辑轴名称1] 指令位置 [逻辑轴名称1] 指令位置 [逻辑轴名称1] 指令位置 ・・・ F插补进给速度 ;

项目	单位	可使用的数据
指令位置	指令单位	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定
插补进给速度	指令单位/min	

※插补进给速度可以省略。

(3)MVS 命令的设定项目

动作示意图



①合成移动量

合成移动量会因 ABS 模式或 INC 模式而异。

- ABS 模式的合成移动量

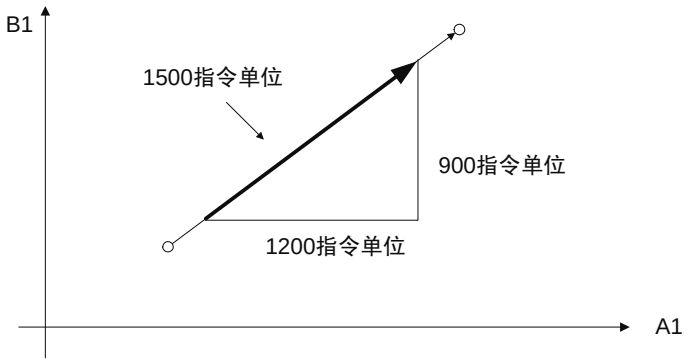
  ABS 模式下时，将从程序当前位置到指令位置的差作为合成移动量进行处置。

- INC 模式的合成移动量

  INC 模式下时，指令位置被直接视为合成移动量。

INC MVS[A1]1200 [B1]900; 时

合成移动量 = $\sqrt{1200^2 + 900^2}$ = 1500 [指令单位]



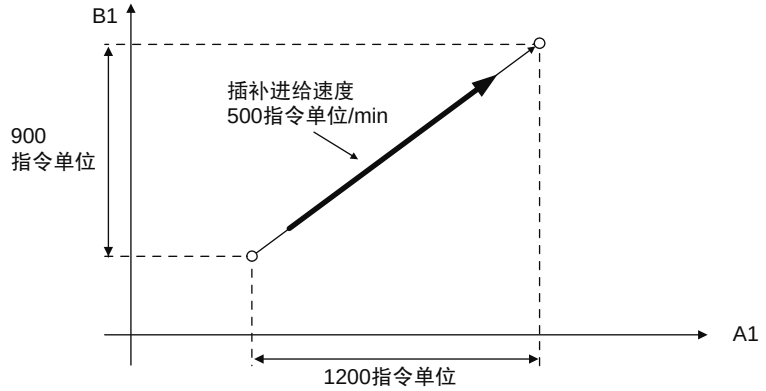
补充

关于移动量的单位，请参阅 “7.2.2 指令单位 ”。

②插补进给速度 （F 指令或 IFP）

插补进给速度使用 MVS 命令，通过字符 “F” 后面的数值或寄存器发出指令。
（F 指令）。
插补进给速度对所有轴的合成速度发出指令。
指令范围为 “1 ～ 插补进给最高速度 (FMX)”［ 指令单位 /min］。

INC MVS[A1]1200 [B1]900 F500; 时



补充

各轴的进给速度可通过下面的计算公式计算。

$$\text{各轴的进给速度［指令单位/min］} = \frac{\text{各轴的移动量［指令单位］}}{\text{合成移动量［指令单位］}} \text{插补进给速度［指令单位/min］}$$

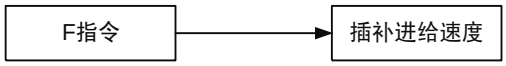
以上示例中，各轴的进给速度如下所示。
插补进给速度 （F 指令值）= 500［ 指令单位 /min］
合成移动量 $=\sqrt{1200^2+900^2}$ = 1500［ 指令单位 ］

- A1 的进给速度 $=\frac{1200}{1500}$ ×500 = 400［ 指令单位 /min］
- B1 的进给速度 $=\frac{900}{1500}$ ×500 = 300［ 指令单位 /min］

还可以选择是否在 F 指令上施加插补超程。
插补超程的使用方法，请参阅 “4.3.3 任务寄存器 ”。

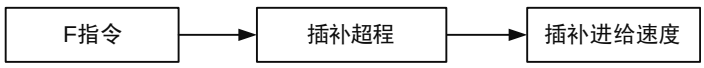
- 不使用插补超程时

F指令 = 插补进给速度



- 使用插补超程时

F指令 × 插补超程 0 ～ 327.67% = 插补进给速度



还可以按相对于插补进给速度 (FMX) 的比率来指令插补进给速度。
关于基于比率的插补进给速度的指令，请参考 IFP 命令。

补充

- F 指令［ 指令单位 /min］中指定超过 FMX 指令值［ 指令单位 /min］时，发生运动程序警报。
- 利用插补超程发出超过 FMX 指令值的插补进给速度时，插补进给速度的输出值变成 FMX 指令值。
- 省略了插补进给速度指定时，按最后指定的插补进给速度动作。

插补超程能够在轴移动过程中进行更改。

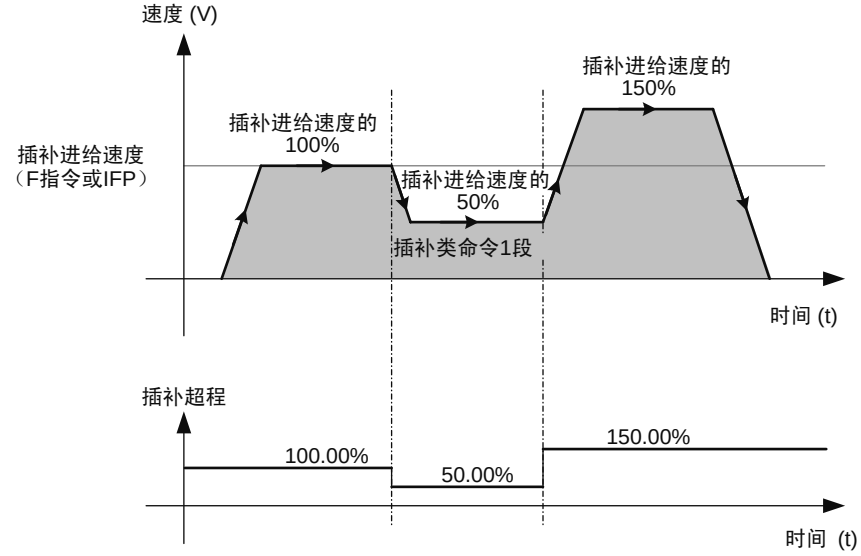


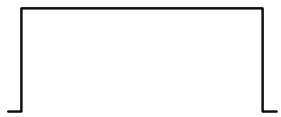
图 8.28 “ 插补超程 ” 和插补类命令

③加减速类型

MVS 命令的加减速类型可从以下 3 种类型中选择。加减速类型通过 IAC/IDC/SCC 命令与运动设定参数 OWxx03 Bit8 ～ B “ 选择滤波器类型 ” 的组合进行设定。

(a) 无加减速

加速时间、减速时间均是 0 的运动方式。

设定方法	动作示意图
• 设定成 OWxx03 Bit8 ～ B “ 选择滤波器类型 ” = 0（无滤波器） • 将 IAC 命令设定成 0 • 将 IDC 命令设定成 0	

(b) 1 段直线加减速

以固定的加速度、减速度运动的方式。

设定方法	动作示意图
• 设定成 OWxx03 Bit8 ～ B “ 选择滤波器类型 ” = 0（无滤波器） • 将 IAC 命令设定成 0 以外的数值 • 将 IDC 命令设定成 0 以外的数值	

(c) S 字加减速

以 S 字的加速度、减速度运动的方式。

设定方法	动作示意图
• 设定成 OWxx03 Bit8 ～ B “ 选择滤波器类型 ” = 2（移动平均滤波器） • 将 IAC 命令设定成 0 以外的数值 • 将 IDC 命令设定成 0 以外的数值 • 将 SCC 命令设定成 0 以外的数值	

补充

- 请务必在运动程序的起始处指定插补进给最高速度 （FMX）。
  否则，MVS 命令会发生运动程序警报。
- 没有加减速时间指令时，按加减速时间的默认值 (0 ms) 动作。
- 针对基于 MVS 命令的轴移动，不会检查是否已进入定位完成范围。需到位检查时，请使用 PEN 命令。

(4) 程序示例

例 以下是 ABS 模式下 MVS 命令的程序示例。

```
FMX T30000000;
ABS;
IAC T1000;
IDC T1000;
MVS [A1]4000 [B1]3000 [C1]2000 F50000;
END;
```

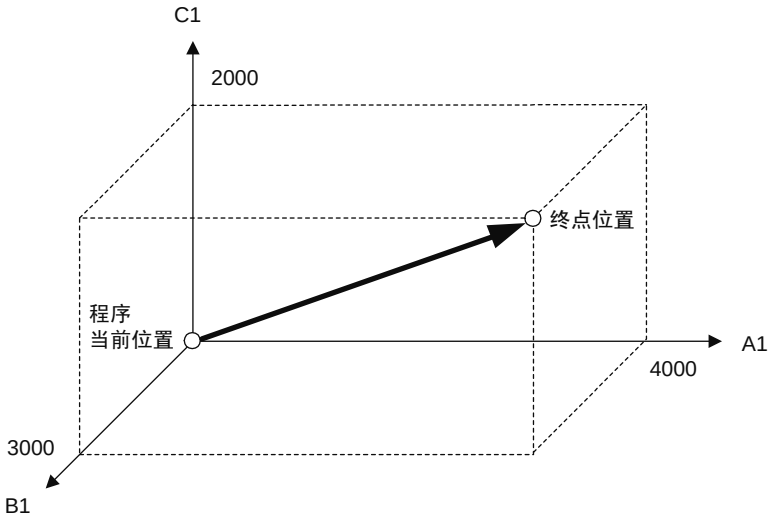


图 8.29 MVS 命令的程序示例

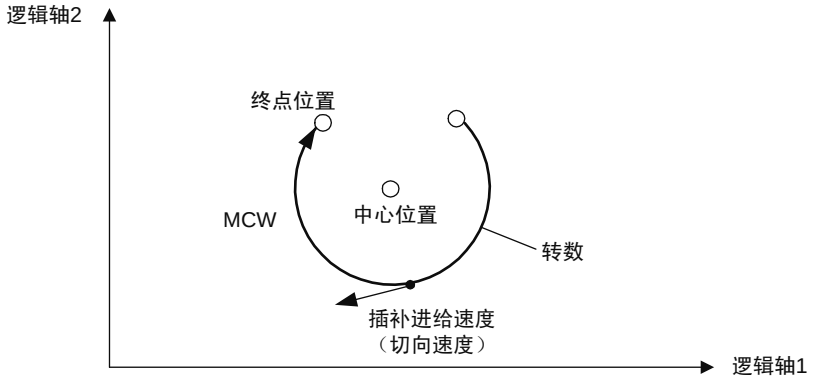
8.2.3 圆弧插补＜指定中心位置＞（MCW、MCC）

运动程序	顺序程序
○	×

(1) 动作说明

圆弧插补＜指定中心位置＞（MCW、MCC）是以插补进给速度沿中心位置所决定的圆弧将指定平面上的同时 2 轴从程序当前位置移动到终点位置的命令。

- MCW：顺时针 （CW）
- MCC：逆时针 （CCW）



重要

- 请务必在圆弧插补命令的前面以坐标平面指定 （PLN）指定圆弧插补的平面。
  没有 PLN 命令指定时，圆弧插补 （MCW、MCC）会发生运动程序警报。
- 终点位置、圆弧中心，请按 PLN 命令指定的横轴名称、纵轴名称对应的顺序指定。
- 请务必在程序的起始处指定插补进给最高速度 （FMX）。
  否则，MCW、MCC 命令会发生运动程序警报。
- 没有加减速时间指令时，按加减速时间的默认值（0ms）动作。

补充

针对基 MCW、MCC 命令的轴移动，不会检查是否已进入定位完成范围。需到位检查时，请使用 PEN 命令。

(2) 格式

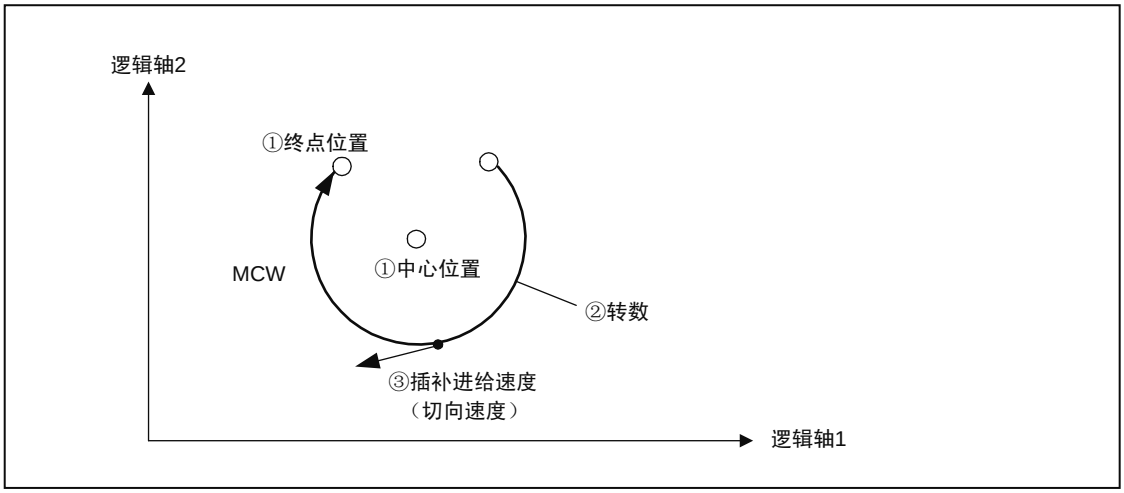
MCW [逻辑轴名称1] 终点位置 [逻辑轴名称2] 终点位置 U中心位置 V中心位置 T转数 F插补进给速度；

项目	单位	可使用的数据
终点位置	指令单位	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定
中心位置	指令单位	
转数	次数	
插补进给速度	指令单位/min	

※转数、插补进给速度可以省略。

(3) 圆弧插补＜指定中心位置＞（MCW、MCC）的设定项目

动作示意图

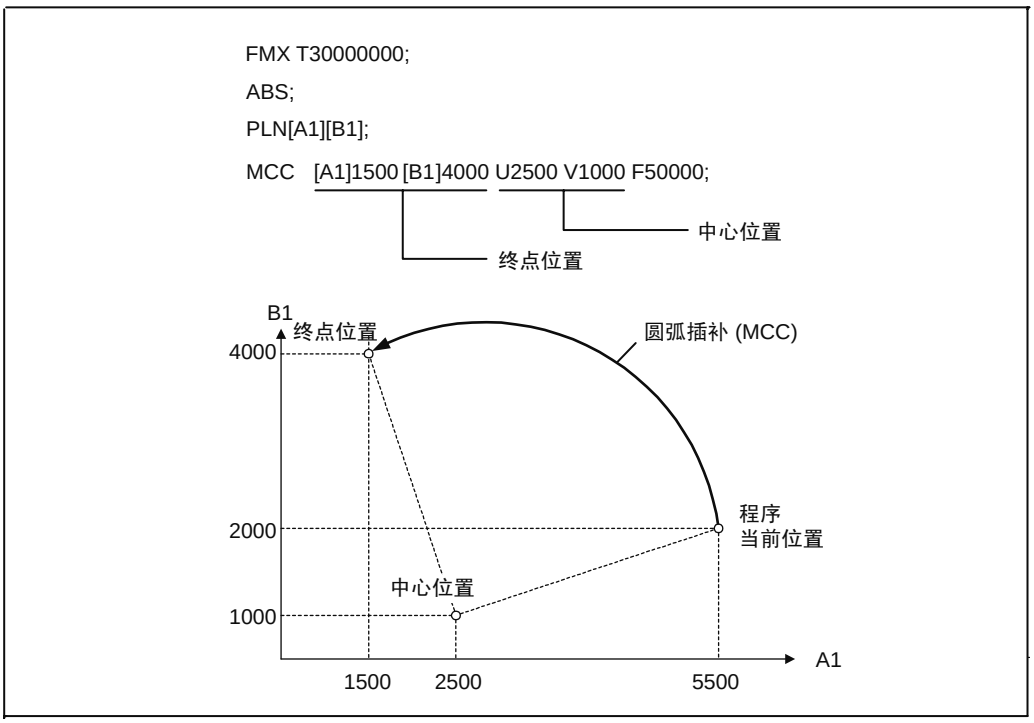


①终点位置及中心位置

终点位置通过逻辑轴名称后面的数值或寄存器进行指定。
中心位置通过 MCW、MCC 命令中的字符 “U”、“V” 后面的数值或寄存器发出指令。
针对指令位置，实际的终点位置以及中心位置由 ABS 或 INC 模式决定。

- ABS 模式

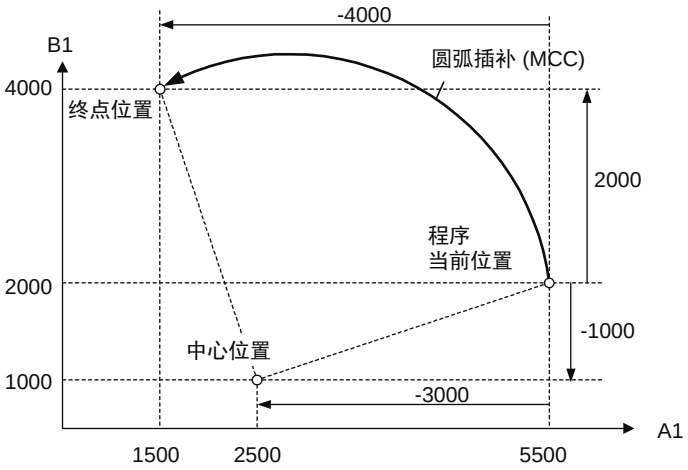
ABS 模式下时，中心位置及终点位置被作为绝对位置进行处理。



• INC 模式

INC 模式下时，中心位置及终点位置被作为距程序当前位置的相对位置进行处理。

```
FMX T30000000;
INC;
PLN[A1][B1];
MCC [A1]-4000 [B1]2000 U-3000 V-1000 F50000;
```



重要

始点半径 ≠ 终点半径时，圆弧插补的轨迹如下，请注意。

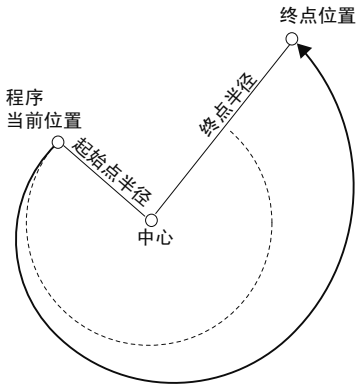


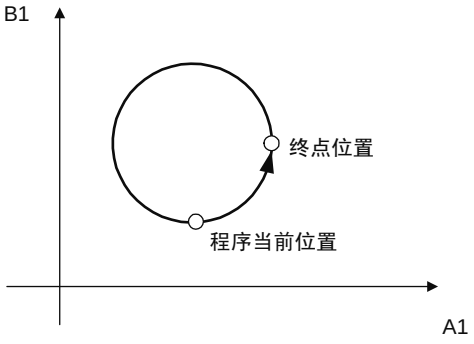
图 8.30 始点半径 ≠ 终点半径的圆弧插补的轨迹

②转数

转数使用 MCW、MCC 命令，通过字符 “T” 后面的数值或寄存器发出指令。
通过指定转数执行多周圆动作。如果在转数中指定了负值，则会发生运动程序警报。基于程序当前位置和终点位置的关系，转数指定与多周圆动作的次数如下所示。

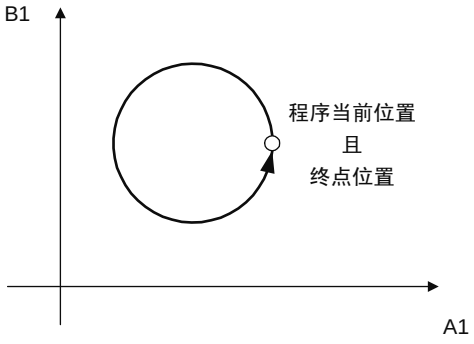
例）转数设定为 2 次时

程序当前位置 ≠ 终点位置时



变成2周 + 1/4周的圆弧。

程序当前位置 = 终点位置时



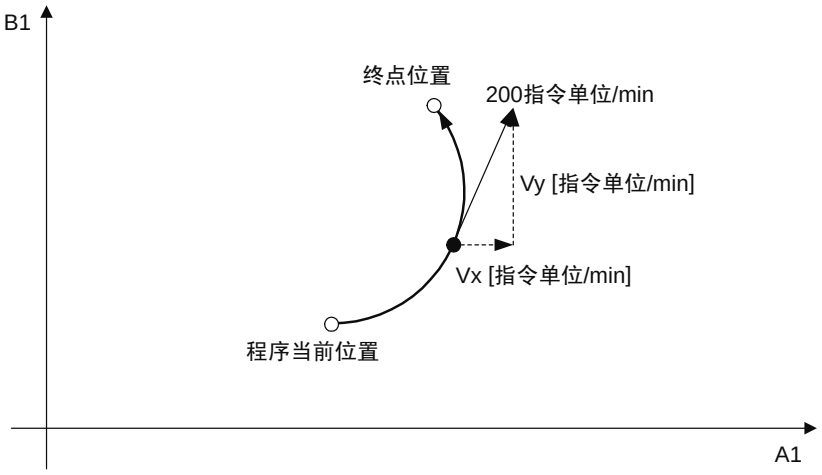
变成3周的圆弧。

③插补进给速度

圆弧插补 （MCW、MCC）时，插补进给速度变成发出切线方向速度的指令。
指令范围为 “1 ～ 插补进给最高速度 (FMX)”[指令单位 /min]。

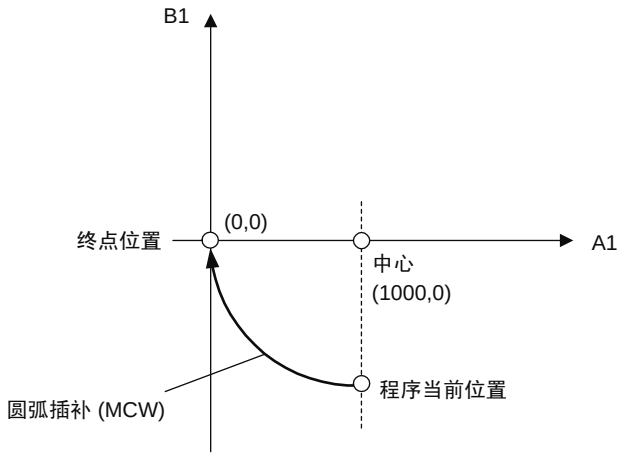
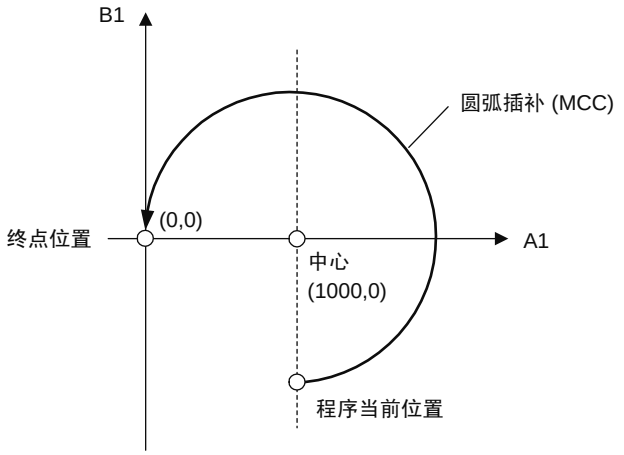
MCC[A1]- [B1]- F200;　时

$F = 200 = \sqrt{Vx^2 + Vy^2}$ [指令单位/min]



(4) 程序示例

例 以下是 ABS 模式下的圆弧插补 （MCW、MCC）的程序示例。
旋转方向指定为 MCW （顺时针）或 MCC （逆时针）。

旋转方向	程序示例
顺时针（MCW）	ABS; FMX T30000000; PLN [A1][B1]; MCW [A1]0 [B1]0 U1000 V0 F2000; “MCW（顺时针）” END;  图 8.31 指定中心位置的程序示例（MCW）
逆时针（MCC）	ABS; FMX T30000000; PLN [A1][B1]; MCC [A1]0 [B1]0 U1000 V0 F2000; “MCC（逆时针）” END;  图 8.32 指定中心位置的程序示例（MCC）

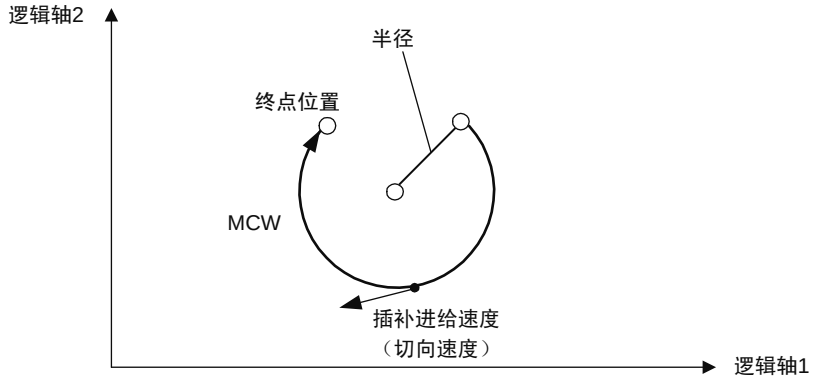
8.2.4 圆弧插补＜指定半径＞（MCW、MCC）

运动程序	顺序程序
○	×

(1) 动作说明

圆弧插补＜指定半径＞（MCW、MCC）是以插补进给速度沿半径所决定的圆弧，同时将指定平面上的 2 轴从程序当前位置移动到终点位置的命令。

- MCW：顺时针（CW）
- MCC：逆时针（CCW）



重要

- 请务必在圆弧插补命令的前面以坐标平面指定（PLN）指定圆弧插补的平面。
  没有 PLN 命令指定时，圆弧插补（MCW、MCC）会发生运动程序警报。
- 终点位置、圆弧中心，请按 PLN 命令指定的横轴名称、纵轴名称对应的顺序指定。
- 请务必在程序的起始处指定插补进给最高速度（FMX）。
  否则，MCW、MCC 命令会发生运动程序警报。
- 没有加减速时间指令时，按加减速时间的默认值（0ms）动作。

补充

针对基 MCW、MCC 命令的轴移动，不会检查是否已进入定位完成范围。需到位检查时，请使用 PEN 命令。

(2) 格式

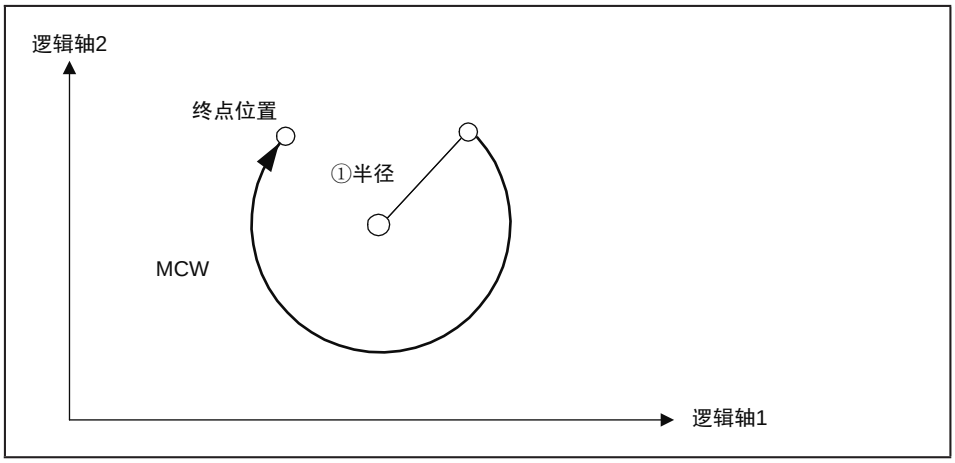
MCW [逻辑轴名称1] 终点位置 [逻辑轴名称2] 终点位置 R半径 F插补进给速度 ;

项目	单位	可使用的数据
终点位置	指令单位	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定
半径	指令单位	
插补进给速度	指令单位/min	

※插补进给速度可以省略。
※半径指定时无法指定转数。

(3) 圆弧插补＜指定半径＞（MCW、MCC）的设定项目

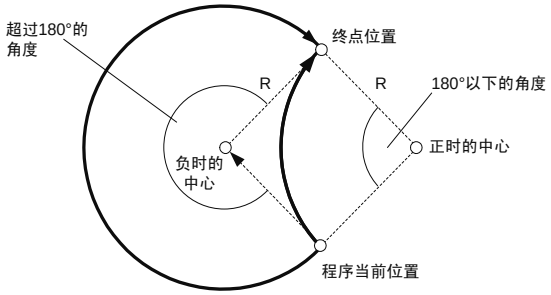
动作示意图



①半径

半径使用 MCW、MCC 命令中的字符 “R” 后面的数值或寄存器发出指令。
根据半径指令值的符号，圆弧插补的轨迹如下。

MCW [A1] — [B1] — R — ; 的指令中，
R ＞ 0 时：圆弧角 180° 以下的圆弧插补
R ＜ 0 时：超过圆弧角 180° 的圆弧插补
R = 0 时：发生运动程序警报。



补充

指定半径时，无法发出一周圆及多周圆的指令。

(4) 程序示例

例 以下是 ABS 模式下的圆弧插补 （MCW、MCC）的程序示例。
旋转方向指定为 MCW （顺时针）或 MCC （逆时针），圆弧角由使用半径指令值的符号进行指定。

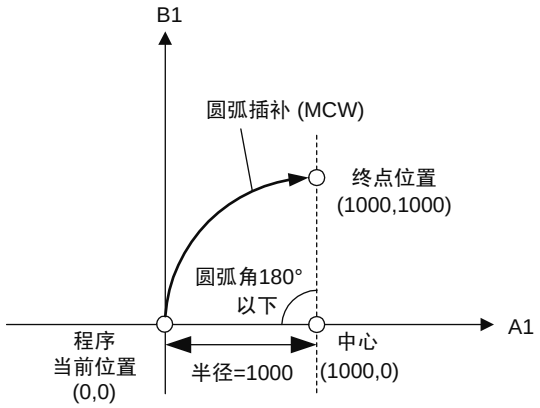
<table>
<tr><th>旋转方向</th><th>圆弧角</th><th>程序示例</th></tr>
<tr><td rowspan="2">顺时针
（MCW）</td><td>180° 以下
（半径指令值＞ 0）</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCW [A1]1000 [B1]1000 R1000 F2000; “MCW（顺时针）”
END;
 

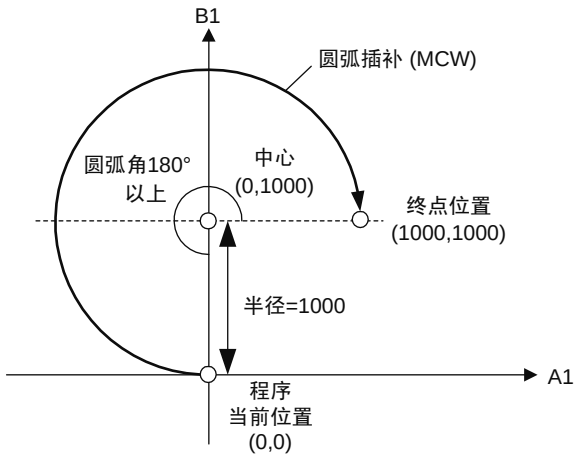
图 8.33 指定半径的程序示例（MCW）</td></tr>
<tr><td>180° 以上
（半径指令值＜ 0）</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCW [A1]1000 [B1]1000 R-1000 F2000; “MCW（顺时针）”
END;
 

图 8.34 指定半径的程序示例（MCW）</td></tr>
</table>

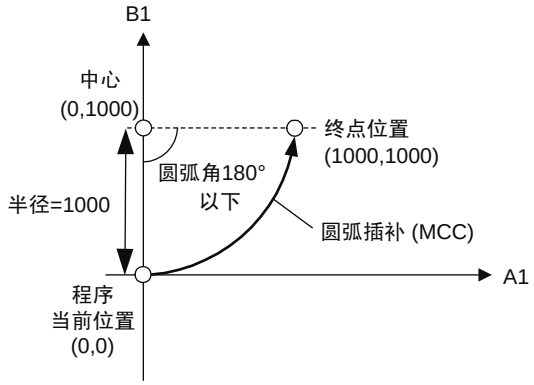
<table>
<tr><th>旋转方向</th><th>圆弧角</th><th>程序示例</th></tr>
<tr><td rowspan="2">逆时针
（MCC）</td><td>180° 以下
（半径指令值＞ 0）</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]1000 R1000 F2000;“MCC（逆时针）”
END;
 

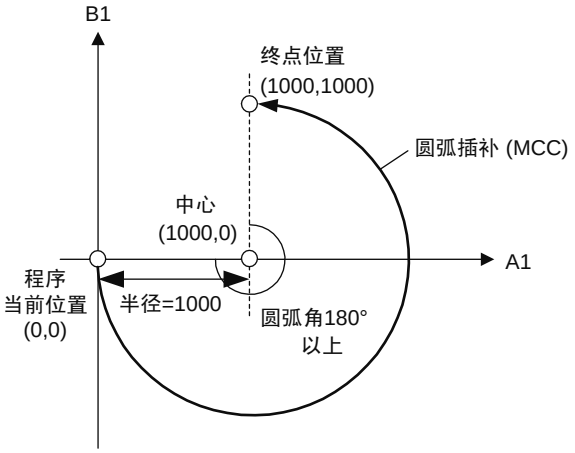
图 8.35 指定半径的程序示例（MCC）</td></tr>
<tr><td>180° 以上
（半径指令值＜ 0）</td><td>ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]1000 R-1000 F2000;“MCC（逆时针）”
END;
 

图 8.36 指定半径的程序示例（MCC）</td></tr>
</table>

8.2.5 螺旋插补＜指定中心位置＞（MCW、MCC）

运动程序	顺序程序
○	×

(1) 动作说明

螺旋插补＜指定中心位置＞（MCW、MCC）是在沿指定中心位置确定的圆弧移动 （圆弧插补）的同时，同步执行直线插补移动的命令。
插补进给速度变成圆弧插补的切线速度与直线插补的合成速度。

- MCW：顺时针 （CW）
- MCC：逆时针 （CCW）

⚠注意

· 螺旋插补 （MCW、MCC）中，执行直线插补的轴可以是直线轴，也可以是旋转轴。但是直线插补部分的轴安装方法不同，螺旋插补的轨迹可能不会变成 “ 螺旋形状 ” 。编程时，请务必检查轨迹，避免工具会干扰到工件等。
否则可能会因干扰造成工具损坏及由此导致人身事故。

重要

- 请务必在螺旋插补命令的前面以坐标平面指定 （PLN）指定圆弧插补的平面。
  通过 [逻辑轴 1]、[逻辑轴 2] 发出指定平面的横轴、纵轴的终点位置和圆弧中心的指令。
- 终点位置、圆弧中心，请按 PLN 命令指定的横轴名称、纵轴名称对应的顺序指定。
- 向直线插补的轴字符中，可发出指定或未指定平面的 1 轴指令。与插补平面的夹角不一定是直角。

补充

针对基于螺旋插补 （MCW、MCC）的轴移动，不会检查是否已进入定位完成范围。
需到位检查时，请使用 PEN 命令。

(2) 格式

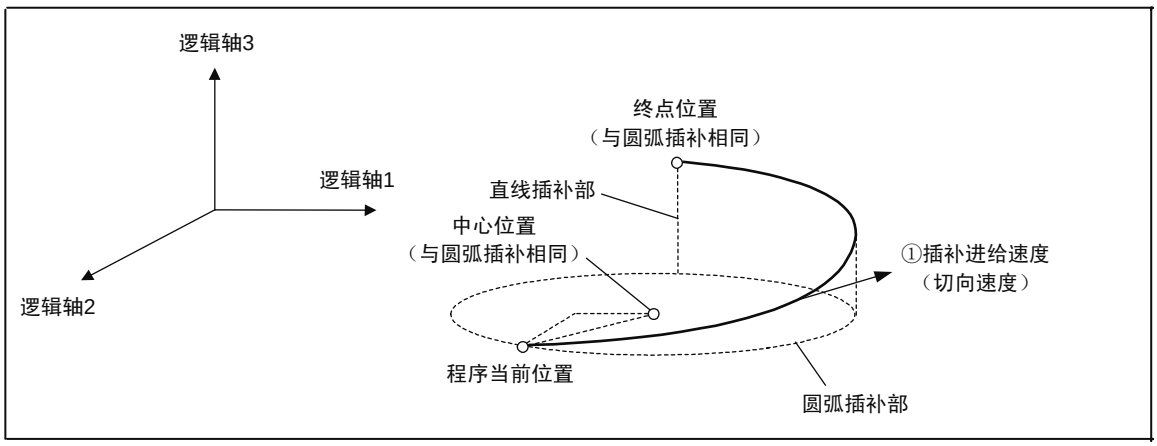
MCW [逻辑轴名称1] 终点位置 [逻辑轴名称2] 终点位置 U中心位置 V中心位置
[逻辑轴名称3] 直线插补的终点位置 T转数 F插补进给速度；

项目	单位	可使用的数据
终点位置	指令单位	· 基于直接数值的直接指定 · 基于倍长整数型寄存器的间接指定
中心位置	指令单位	
转数	次数	
插补进给速度	指令单位/min	

※转数、插补进给速度可以省略。

(3) 螺旋插补＜指定中心位置＞（MCW、MCC）的设定项目

动作示意图

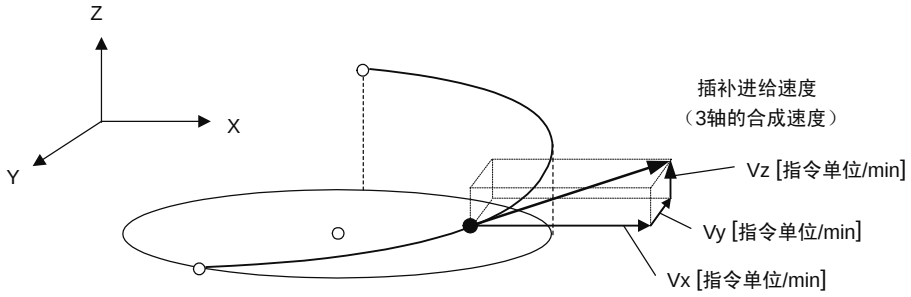


①插补进给速度

螺旋插补 （MCW、MCC）时，插补进给速度变成发出圆弧插补的切线方向和直线插补轴方向之间合成速度。

MCC[X]- [Y]- U- V- [Z]- F300; 时

$F = 300 = \sqrt{Vx^2 + Vy^2 + Vz^2}$ [指令单位/min]



(4) 程序示例

例 以下是 ABS 模式下的螺旋插补 （MCC）的程序示例。

```
ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]0 U0 V0 [C1]500 F2000;
END;
```

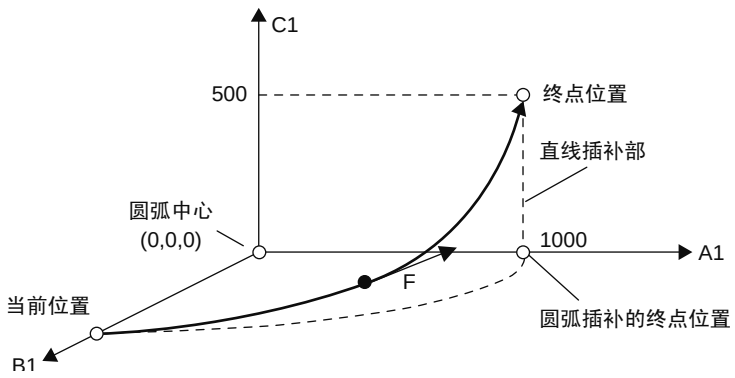


图 8.37 螺旋插补＜指定中心位置＞（MCC）的程序示例

8.2.6 螺旋插补＜指定半径＞（MCW、MCC）

运动程序	顺序程序
○	×

(1) 动作说明

螺旋插补＜指定半径＞（MCW、MCC）是在沿指定半径值确定的圆弧移动 （圆弧插补）的同时，同步执行直线插补移动的命令。
插补进给速度变成圆弧插补的切线速度与直线插补的合成速度。

- MCW：顺时针 （CW）
- MCC：逆时针 （CCW）

⚠注意

· 螺旋插补 （MCW、MCC）中，执行直线插补的轴可以是直线轴，也可以是旋转轴。但是直线插补部分的轴安装方法不同，螺旋插补的轨迹可能不会变成 “ 螺旋形状 ” 。编程时，请务必检查轨迹，避免工具会干扰到工件等。
否则可能会因干扰造成工具损坏及由此导致人身事故。

重要

- 请务必在螺旋插补命令的前面以坐标平面指定 （PLN）指定圆弧插补的平面。
  通过 [逻辑轴 1]、[逻辑轴 2] 发出指定平面的横轴、纵轴的终点位置和圆弧中心的指令。
- 终点位置、圆弧中心，请按 PLN 命令指定的横轴名称、纵轴名称对应的顺序指定。
- 向直线插补的轴字符中，可发出指定或未指定平面的 1 轴指令。与插补平面的夹角未必就是直角。

补充

针对基于螺旋插补 （MCW、MCC）的轴移动，不会检查是否已进入定位完成范围的到位检查。
需到位检查时，请使用 PEN 命令。

(2) 格式

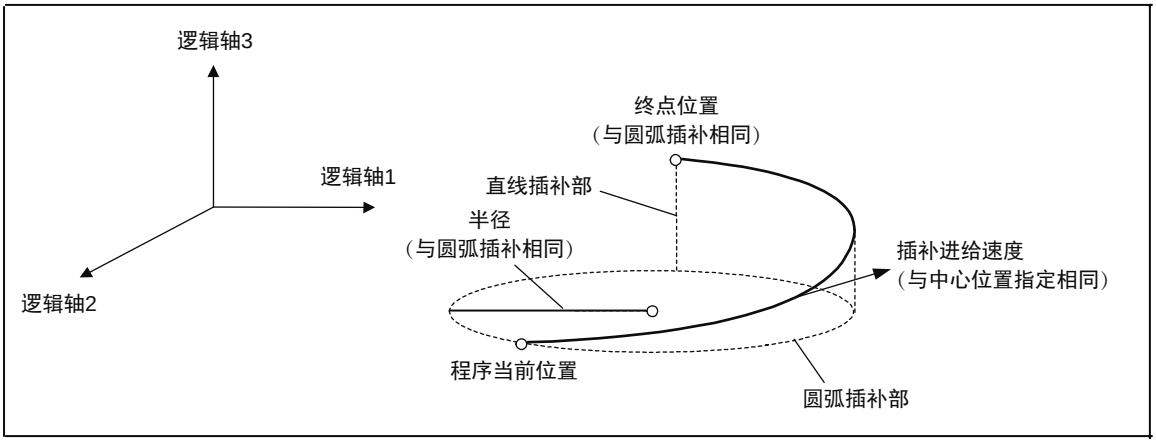
MCW [逻辑轴名称1] 终点位置 [逻辑轴名称2] 终点位置 R半径
[逻辑轴名称3] 直线插补的终点位置 F插补进给速度 ;

项目	单位	可使用的数据
终点位置	指令单位	· 基于直接数值的直接指定 · 基于倍长整数型寄存器的间接指定
中心位置	指令单位	
半径	指令单位	
插补进给速度	指令单位/min	

※插补进给速度可以省略。
※半径指定时无法指定转数。

(3) 螺旋插补＜指定半径＞（MCW、MCC）的设定项目

动作示意图



补充

螺旋插补＜指定半径＞命令的半径、终点位置的指定方法与圆弧插补＜指定半径＞时相同。
此外，插补进给速度的指定方法与螺旋圆弧插补＜指定中心位置＞时相同。

(4) 程序示例

例 以下是 ABS 模式下的螺旋插补 （MCC）的程序示例。

```
ABS;
FMX T30000000;
PLN [A1][B1];
MCC [A1]1000 [B1]0 R1000 [C1]500 F2000;
END;
```

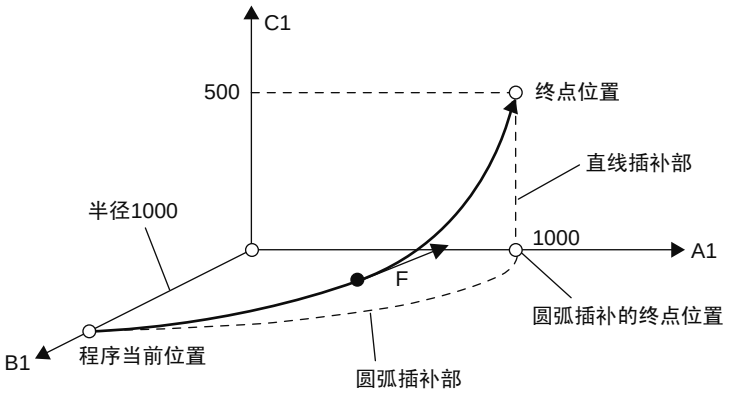


图 8.38 螺旋插补＜指定半径＞（MCC）的程序示例

8.2.7 原点复归 （ZRN）

运动程序	顺序程序
○	×

(1) 动作说明

原点复归 （ZRN）是执行原点复归动作的命令。
同时最多可移动 16 轴。省略该指令的轴不移动。指令中所有轴的原点复归动作完成后，进入下一程序段。

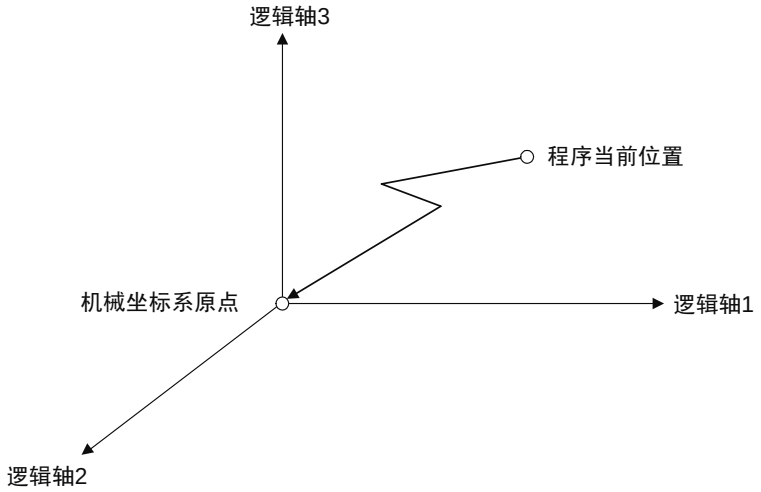


图 8.39 基于 ZRN 命令的移动轨迹

执行 ZRN 命令后，会将复归后的位置设定为机械坐标系的原点。同时，通过更改当前值 （POS）取消之前设定的任务坐标系。
执行 ZRN 命令后，机械坐标系与任务坐标系一致。接着，在执行更改当前值 （POS）之前，机械坐标指令 （MVM）即使发出指令也视为无效。
机械坐标系与任务坐标系的详细情况，请参阅 “8.3.1 当前值变更 （POS）”。

重要

程序暂停请求在 ZRN 命令中无效。
中途停止时，请执行程序停止请求。
关于程序暂停请求、程序停止请求，请参阅 “4.3.3 任务寄存器 ”。

(2) 格式

ZRN [逻辑轴名称1] 0 [逻辑轴名称2] 0 [逻辑轴名称3] 0・・・;

※逻辑轴名称后面的数据0，请务必指定。

(3) ZRN 命令的设定项目

(a) 原点复归方式

各轴的原点复归方式通过运动运动设定参数 OWxx3C “ 原点复归方式 ” 进行设定。
以下表示可使用的原点复归方式。
有关各方式的详细说明，请参阅所使用运动单元的使用手册。

名称	原点复归方式设定 (OWxx3C)	SVA-01	SVB-01/ 内置 SVB	PO-01
DEC1+C 相脉冲	0	○	○	×
ZERO 信号	1	○	○	×
DEC1+ZERO 信号	2	○	○	○
C 相脉冲	3	○	○	×
DEC2+ZERO 信号	4	○	×	○
DEC1+LMT+ZERO 信号	5	○	×	○
DEC2+C 相信号	6	○	×	×
DEC1+LMT+C 相信号	7	○	×	×
单 C 脉冲	11	○	○	×
P-OT&C 相脉冲	12	○	○	×
P-OT	13	○	○	×
HOME LS&C 相脉冲	14	○	○	×
HOME LS	15	○	○	×
N-OT&C 相脉冲	16	○	○	×
N-OT	17	○	○	×
INPUT&C 相脉冲	18	○	○	×
INPUT	19	○	○	×

○：可使用，×：不能使用

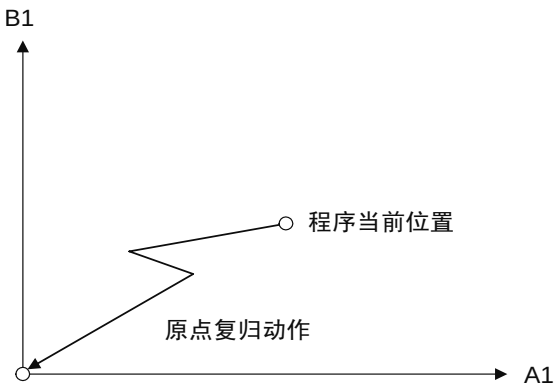
(b) 原点复归速度

原点复归速度会由使用的原点复归方式决定。
详细说明，请参阅所使用运动单元的使用手册。

(4) 程序示例

例 以下是 ABS 模式下 ZRB 命令的程序示例。

```
ZRN [A1]0 [B1]0;
END;
```



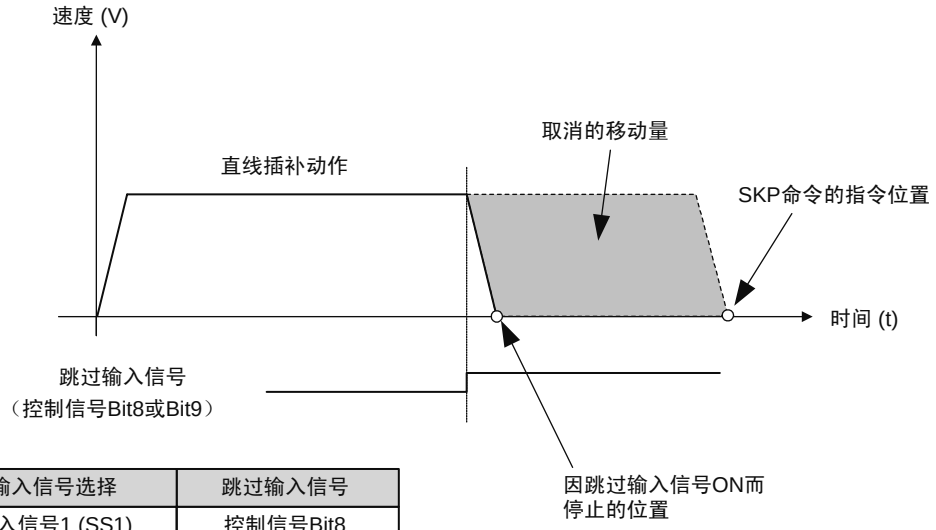
将停止位置设为机械坐标系原点 (0, 0)。

8.2.8 带跳过功能的直线插补 （SKP）

运动程序	顺序程序
○	×

(1) 动作说明

带跳过功能的直线插补 （SKP）是扩展直线插补 （MVS）的命令。SKP 命令中，在轴移动过程中跳过输入信号变成 ON 时，移动中的轴会减速停止，取消余下的移动量指令。
使用 SKP 命令，根据外部情况运行运动控制程序，并且可跳过输入信号输入到 MSEE 命令或 M-EXECUTOR 控制寄存器的控制信号中。



跳过输入信号选择	跳过输入信号
跳过输入信号1 (SS1)	控制信号Bit8
跳过输入信号2 (SS2)	控制信号Bit9

图 8.40 基于 SKP 命令的动作示意图

重要

跳过输入信号变成 ON 时，轴会停止减速，但在定位完成信号 ON 之前会一直执行 SKP 命令。

(2) 格式

SKP [逻辑轴名称1] 指令位置 [逻辑轴名称2] 指令位置 [逻辑轴名称3] 指令位置・・・
F插补进给速度 SS跳过输入信号选择;

项目	单位	可使用的数据
指令位置	指令单位	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定
插补进给速度	指令单位/min	
跳过输入信号选择	—	・基于直接数值的直接指定（1或2） ・基于倍长整数型寄存器的间接指定

※插补进给速度可以省略。

(3) 程序示例

例 以下是 ABS 模式下 SKP 命令的程序示例。

```
FMX T30000000;
ABS;
IAC T1000;
IDC T1000;
SKP [A1]4000 [B1]3000 [C1]2000 F50000 SS1;
END;
```

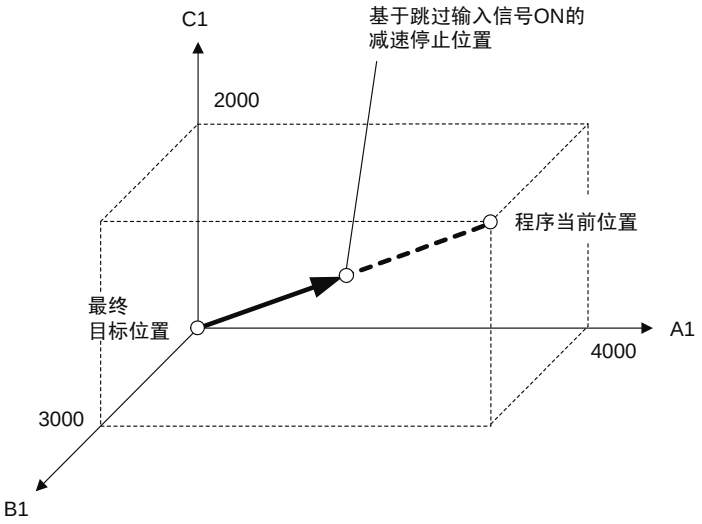


图 8.41 SKP 命令的程序示例

8.2.9 指定时间定位 （MVT）

运动程序	顺序程序
○	×

(1) 动作说明

指定时间定位 （MVT）是扩展定位 （MOV）的命令。
同时最多可移动 16 轴。省略该指令的轴不移动。
指定时间定位 （MVT）为了在指定时间内完成定位，而调节各轴的进给速度，执行定位。该命令非插补动作，因此，并非指令轴全部同时定位完成。
时间制定设定不同，会存在时间偏差。

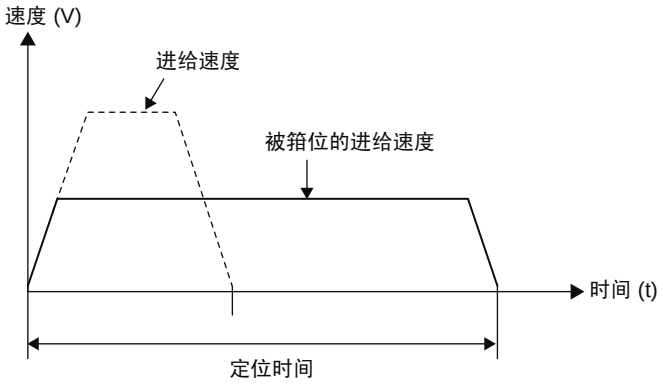


图 8.42 基于 MVT 命令的动作示意图

使用超程时，无法在指定时间内完成定位。
使用滤波器时，定位时间仅会延迟相当于滤波器时间常数的量。

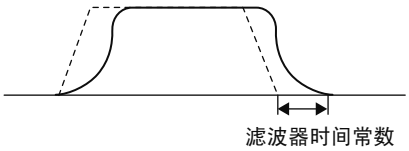


图 8.43 使用滤波器时定位时间延迟

重要

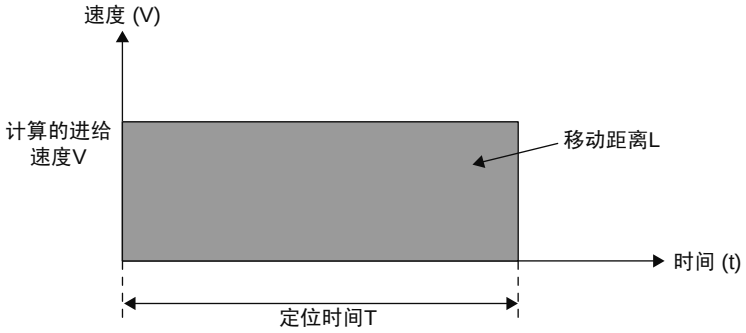
- 将定位时间设定成 0，则会发生运动程序警报。
- 如果存在移动量为 0 的轴，则会发生运动程序警报。

(2) 格式

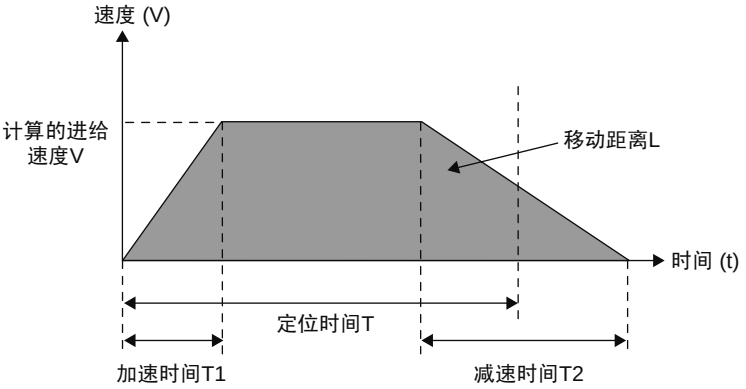
MVT [逻辑轴名称1] 指令位置 [逻辑轴名称2] 指令位置 [逻辑轴名称3] 指令位置・・・
T定位时间；

项目	单位	可使用的数据
指令位置	指令单位	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定
定位时间	ms	

定位时间的指令范围为 “1 ～ 2147483647[ms]”。
MVT 命令时的进给速度会根据定位时间和移动量通过 MP2000 控制器内部进行计算。
如下所示，该计算中无加速度，即加速度为 0。



实际动作如下所示 （加速时间 T1 〈 减速时间 T2 时）。



进给速度可通过 MVT 命令进行更改。因此，请在执行 MVT 命令后，通过 VEL 命令重新设定进给速度。

补充

针对基于 MVT 命令的轴移动，与 MOV 命令一样，检查是否已进入定位完成范围。

(3) 程序示例

例

以下是 ABS 模式下 MVT 命令的程序示例。

```
ABS;
ACC [A1]1000;
DCC [A1]1000;
MVT [A1]4000 T1000;
END;
```

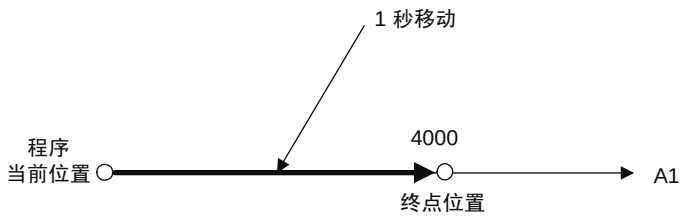


图 8.44 MVT 命令的程序示例

8.2.10 外部定位 （EXM）

运动程序	顺序程序
○	×

(1) 动作说明

外部定位 （EXM）是扩展定位 （MOV）的命令。
外部定位 （EXM）在外部定位信号 ON 后会执行只会移动指定量 （增量值）的定位。外部定位信号没有接通时，轴会完成 EXM 命令中指定的位置定位。
外部定位 （EXM）只能指定 1 轴。

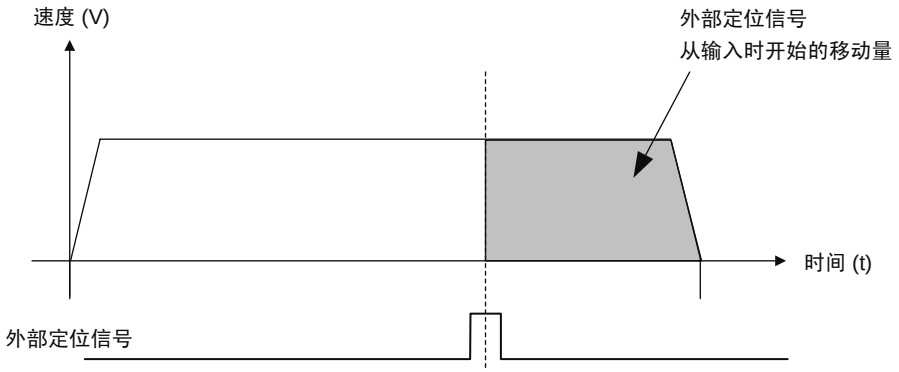


图 8.45 基于 EXM 命令的动作示意图

如果在移动量中指定了负值，则会在减速停止后朝负的方向移动。

重要

- 关于外部定位信号，请参阅对应运动模块的用户手册。
- PO-01 模块无法使用外部定位 （EXM）。
  发生运动程序警报。
- 使用时请注意，外部闩锁输入信号有时会在原点复归中使用。

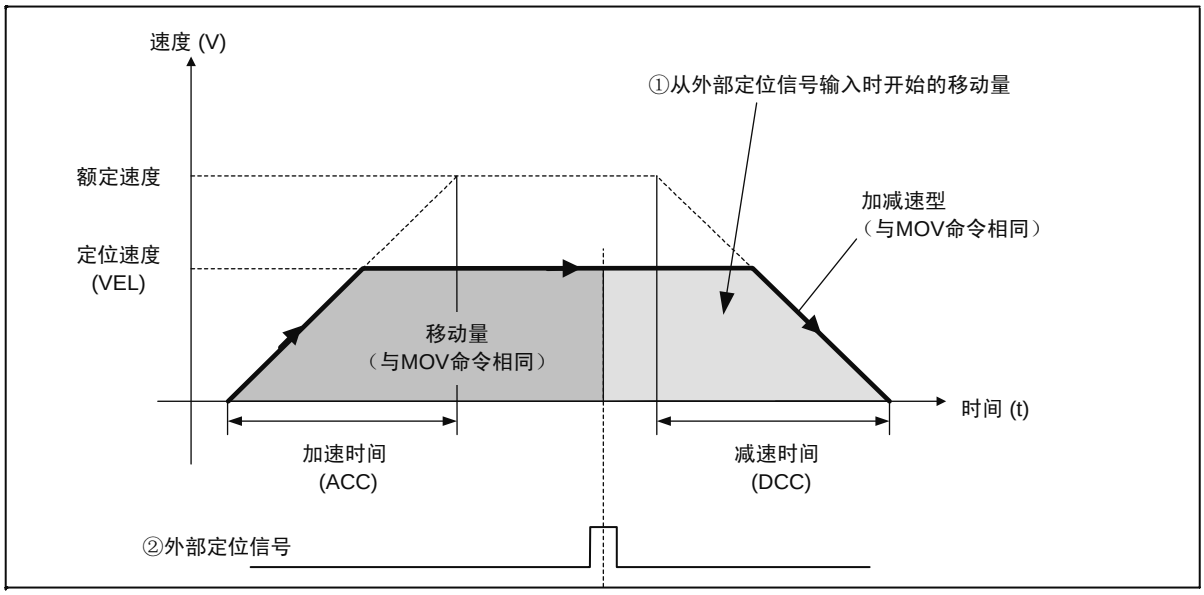
(2) 格式

EXM [逻辑轴名称1] 指令位置 从外部定位信号输入时开始的移动量 ;

项目	单位	可使用的数据
指令位置	指令单位	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定
从外部定位信号输入时开始的移动量	指令单位	

(3)EXM 命令的设定项目

动作示意图



①从外部定位信号输入时开始的移动量

使用增量值设定外部定位信号 ON 后的移动量。
指令范围为 “-2147483648 ～ 2147483647[指令单位]”。

②外部定位信号

外部定位信号通过运动设定参数 OWxx04 Bit4 ～ 7 “ 功能设定 2” 进行设定。
详细信息，请参阅对应运动模块的用户手册。

补充 PO-01 模块没有外部定位功能。

(4) 程序示例

例 以下是 ABS 模式下 EXM 命令的程序示例。

```
ABS;
ACC [A1]1000;
DCC [A1]1000;
VEL [A1]2000;
DL00000 = 1000;
EXM [A1]4000 DDL00000;
END;
```

8.3 轴控制命令

在本节中，将对轴控制命令进行说明。

8.3.1 当前值变更 （POS）

运动程序	顺序程序
○	×

⚠注意

- 请注意当前值变更 （POS）。
  当前值变更 （POS）创建新的 “ 任务坐标值 ”。因此，如果错误地发出 POS 命令，则后面的移动动作将完全不同。运行之前，请务必要求确认是否已发出了正确的 “ 任务坐标系 ” 指令。
  否则可能会因干扰造成工具损坏及由此导致人身事故。

(1) 动作说明

当前值变更 （POS）是将当前位置更改成 “ 所需坐标值 ”，创建新坐标系的命令。本使用手册中，将这个新设定的坐标系称为 “ 任务坐标系 ”，将机械原来就有的坐标称为 “ 机械坐标系 ”。
POS 命令后面发出的移动指令在任务坐标系上动作。

坐标系	说明	备注
机械坐标系	机械原有坐标	原点复归时的位置为原点 (0)
任务坐标系	由用户设定任意位置的坐标	使用 POS 命令来设定新的坐标。

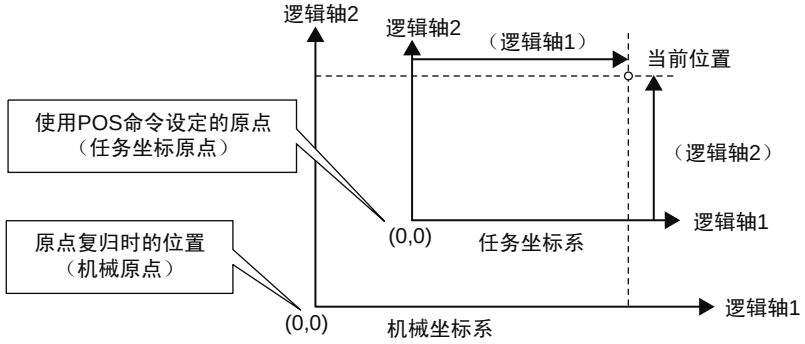


图 8.46 基于 POS 命令的任务坐标系设定

可通过 POS 命令频繁切换任务坐标系与机械坐标系。请事先设定好机械坐标系。另外，POS 命令的机械坐标系没有影响。
POS 命令最多可同时发出 16 轴的指令。省略该指令的轴不会更新任务坐标系。
任务坐标系上的移动指令在通过机械坐标系进行换算时，无法运行最大指令值。

下表表示机械坐标系与任务坐标系设定的状态。

表 8.1 坐标系设定的定时

坐标系设定的定时	运动固定参数 No.30 “ 选择编码器 ”	
	增量型编码器 / 绝对值编码器（用作增量型）	绝对值编码器
接通电源的操作后	机械坐标系：临时设定 *1 任务坐标系：取消 *3	机械坐标系：有 *2 任务坐标系：取消
执行原点复归 （ZRN）后	机械坐标系：设定 任务坐标系：取消	任务坐标系：取消
执行 POS 命令后	任务坐标系：设定	任务坐标系：设定
原点设定的操作后	机械坐标系：设定	机械坐标系：设定

*1. 临时设定：设定将接通电源时的当前位置作为原点的机械坐标系。
之后，如果不执行原点复归，则软件 LS 功能仍为无效。
*2. 有 ：利用来自绝对值检测编码器的位置信息创建坐标原点。
*3. 取消 ：取消以前设定的任务坐标系，与机械坐标系相等。

重要

- 按照无限长设定的轴，只可在 0 ～ POSMAX 的范围内进行设定。
如果设定该范围以外的值，则会发生运动程序警报。
- 通不适用 ZRN 命令执行原点复归时，如同通过梯形图程序执行原点复归一样，任务坐标系不会被取消。

(2) 格式

POS [逻辑轴名称1] 坐标轴 [逻辑轴名称2] 坐标轴・・・;

项目	单位	可使用的数据
坐标轴	指令单位	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

(3) 程序示例

例 以下是 POS 命令的程序示例。

```
ABS;                              “ 绝对值模式

MOV [A1]1000 [B1]2000;            “ 定位

POS [A1]0 [B1]0;                  “ 更新任务坐标系
MOV [A1]3000 [B1]4000;            “ 定位

DL00000 = IL8010;                 “ 获取 A1 轴的机械坐标系计算位置 (CPOS)
DL00002 = IL8090;                 “ 获取 B1 轴的机械坐标系计算位置 (CPOS)
POS [A1]DL00000 [B1]DL00002;      “ 取消任务坐标系

END;
```

8.3.2 机械坐标指令 （MVM）

运动程序	顺序程序
○	×

⚠注意

- 机械坐标指令 （MVM）默认暂时执行定位到 “ 机械坐标系 ” 上的位置动作。因此，如果不确认 “ 机械坐标系 ” 的原点位置就发出指令，则会导致移动动作异常。请在运行之前务必确认是否已发出了 “ 机械坐标系 ” 上的正确位置的指令。

否则可能会因干扰造成工具损坏及由此导致人身事故。

(1) 动作说明

机械坐标指令 （MVM）是在利用更改当前值 （POS）设定了与机械坐标系不同的任务坐标系后，可暂时在机械坐标系上移动时使用的命令。
如果轴移动命令指定了机械坐标指令 （MVM），会暂时将轴移动到机械坐标系的绝对坐标位置的动作。此时，轴的移动与 ABS/INC 模式的设定状态无关，务必在 ABS 模式下动作。
MVM 命令仅在指定程序段有效。例如，下一个程序段以后的直线插补 （MVS）中，仅有任务坐标系上的移动。

(2) 格式

```
MVM  MOV ……;
或
MVM  MVS ……;
```

(3) 程序示例

例 以下是 MVM 命令的程序示例。

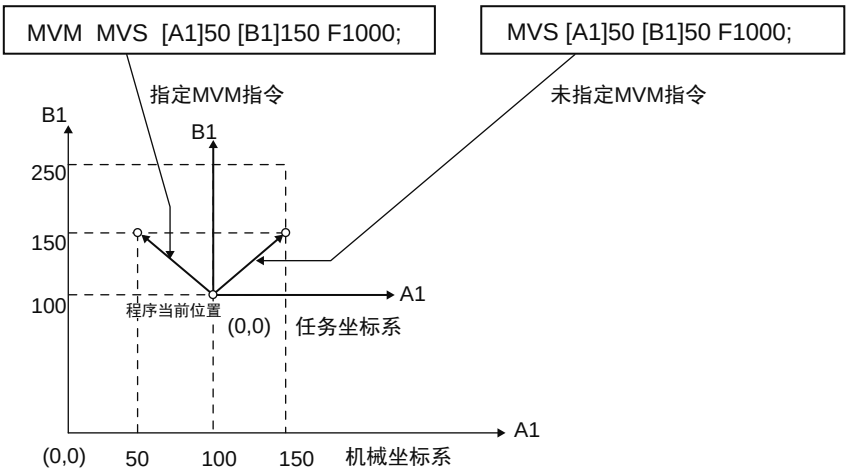


图 8.47 机械坐标指令 （MVM）的程序示例

8.3.3 更新程序当前位置 （PLD）

运动程序	顺序程序
○	×

(1) 动作说明

更新程序当前位置 （PLD）是通过手动介入等方式来更新程序当前位置的命令。
在运动程序运行中，通过非运动程序 （通过 JOG 动作、STEP 动作及用户函数驱动轴等时）驱动轴时，不会更新 “ 程序当前位置 ”。在此状态下如果继续运行运动程序，则会按照手动介入移动到相当于移动量的位置。为了解决这个问题，可使用更新 “ 程序当前位置 ” 的 PLD 命令。

(2) 格式

```
PLD [ 逻辑轴名称 1 ] [ 逻辑轴名称 2 ] [ 逻辑轴名称 3 ]…;
```

(3) 程序示例

以下是 PLD 命令的程序示例。

例 (a) 在运动程序运行中手动介入时

```
MOV [A1]1000;
                “ 在此期间通过 JOG 驱动了 [A1] 轴
PLD [A1];       “ 更新 “ 程序当前位置 ”
MOV [A1]2000;
```

例 (b) 通过运动程序的用户函数驱动轴时

```
MOV [A1]1000;
UFC  FNC10 MB000000 IW0100 MB000020; “ 利用用户函数驱动 [A1] 轴
PLD [A1];                            “ 更新 “ 程序当前位置 ”
MOV [A1]2000;
```

补充 PLD 命令在某些用途时通过用户激活来执行。在运动程序运行中，即使是手动介入的应用程序，有时也不会使用 PLD 命令。

8.3.4 到位检查 （PFN）

运动程序	顺序程序
○	×

(1) 动作说明

到位检查 （PFN）是在插补移动时检查已进入定位点附近的轴的命令。
针对插补类命令 (MVS、MCW、MCC、SKP) 的轴移动，通常不会对已进入定位完成范围的轴执行到位检查。检查到轴已进入定位完成范围时，使用到位检查 （PFN）。

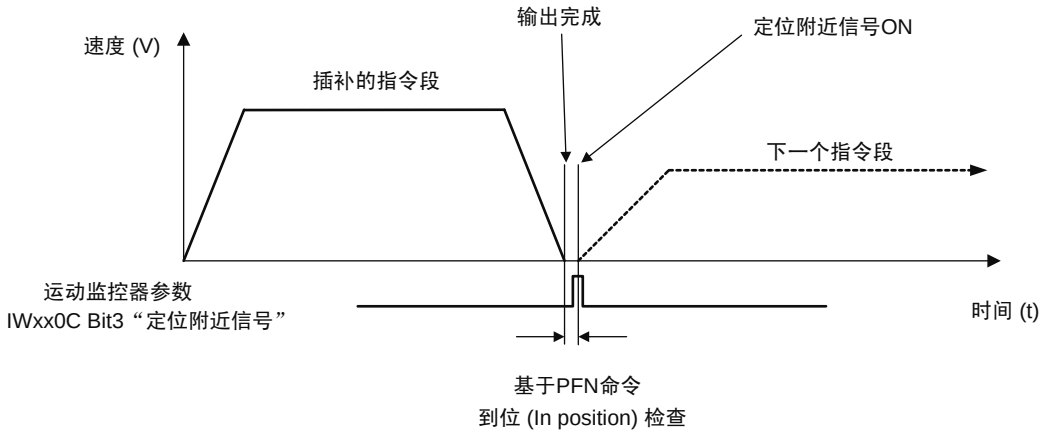


图 8.48 PFN 命令的动作

运动监控器参数 IWxx0C Bit3“ 定位附近信号 ” 在满足以下条件后会变成 ON。
定位附近检测宽度使用 INP 命令进行设定。

| MPOS － APOS | ≦ 定位附近检测宽度

MPOS： 运动监控器参数 ILxx12“ 机械坐标系指令位置 ”
APOS： 运动监控器参数 ILxx16“ 机械坐标系反馈位置 ”

补充

定位附近检测宽度为 0 时，包括滤波器在内的输出完成后变成 ON。

(2) 格式

・ 指定到与插补类命令相同的程序段时
MVS [逻辑轴名称 1] － [逻辑轴名称 2] － [逻辑轴名称 3]… **PFN**;

・ 单独指定时
PFN [逻辑轴名称 1] [逻辑轴名称 2] [逻辑轴名称 3]…;

(3) 程序示例

以下是 PFN 命令的程序示例。

例 (a) 指定到与插补类命令相同的程序段时

```
MVS [A1]1000 F20000 PFN;
MOV [A1]3000;
END;
```

例 (b) 单独指定时

```
MVS [A1]1000 F20000;
PFN [A1];
MOV [A1]3000;
END;
```

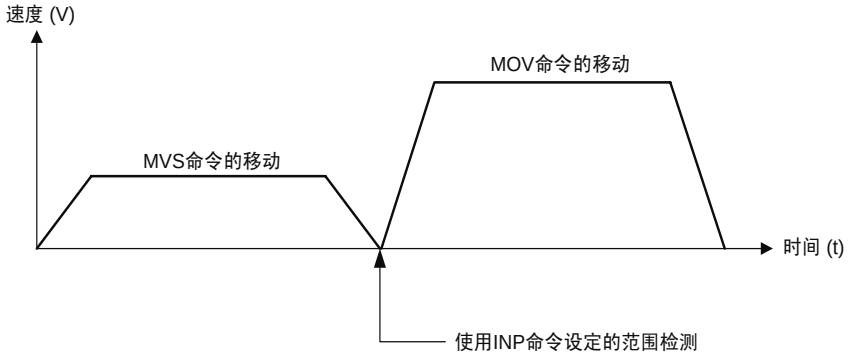


图 8.49 PFN 命令的程序示例

8.3.5 到位检查宽度设定 （INP）

运动程序	顺序程序
○	×

(1) 动作说明

到位检测宽度设定 （INP）是设定定位附近宽度 （到位检查宽度）的命令。
最多可同时设定 16 轴。指令中的轴会更新各轴的运动设定参数 OLxx20“ 定位附近检测宽度 ”。
指令范围为 “1 ～ 65535[指令单位]”。

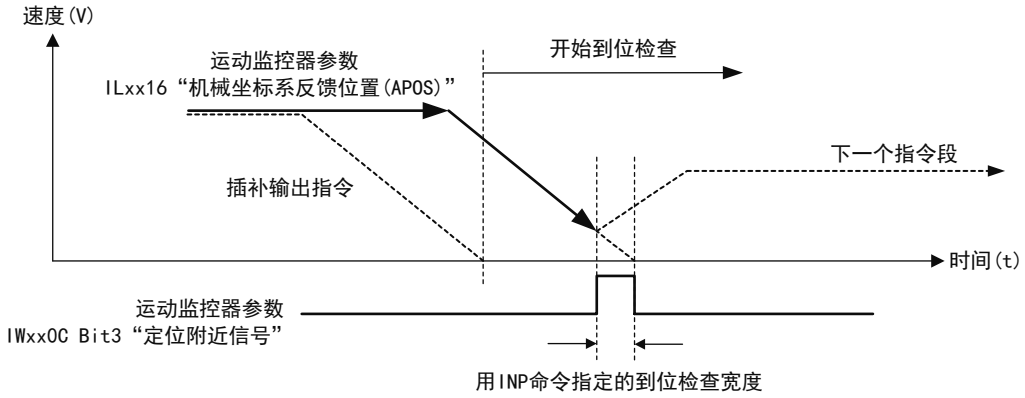


图 8.50 INP 命令的指令方法

补充

SVR 模块中没有运动设定参数 OLxx20“ 定位附近检测宽度 ”。
SVR 模块按照 0 来处理检测宽度。

(2) 格式

INP [逻辑轴名称1] 定位附近检测宽度 [逻辑轴名称2] 定位附近检测宽度 ・・・;

项目	单位	可使用的数据
定位附近检测宽度	指令单位	・基于直接数值的直接指定 ・基于倍长整数型寄存器的间接指定

(3) 程序示例

例 以下是 INP 命令的程序示例。

```
ABS;
MOV [A1]0 [B1]0;        “向原点定位
INP [A1]100 [B1]200;    “设定到位检查宽度
MVS [A1]1000 PFN;
MVS [B1]1000 PFN;
MVS [A1]-1000 ;
END;
```

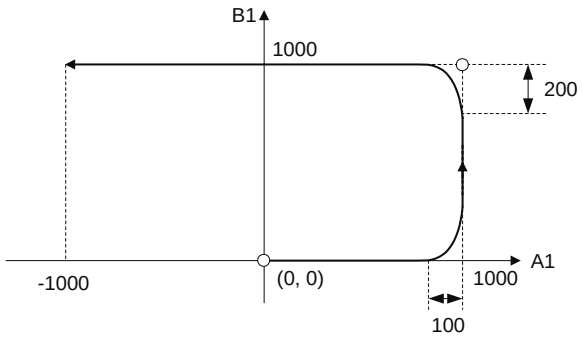


图 8.51 INP 命令的程序示例

8.3.6 指定坐标平面 （PLN）

运动程序	顺序程序
○	×

(1) 动作说明

指定坐标平面 （PLN）是指定 2 个通过参数设定的逻辑轴，定义为坐标平面的命令。在执行圆弧插补（MCW、MCC）、螺旋插补 （MCW、MCC）之前，请务必执行该命令。
坐标平面一旦指定，会一直有效，直到重新定义或执行程序的 END。

(2) 格式

```
                横轴名称          纵轴名称
PLN  [逻辑轴名称 1 ][逻辑轴名称 2 ];
       指定坐标平面的 2 轴
```

(3) 程序示例

例 以下是 PLN 命令的程序示例。

```
PLN[A1][B1];                    “ 指定由 A1、B1 轴构成的平面。
MCW  [A1]50 [B1]50 R50 F1000;
```

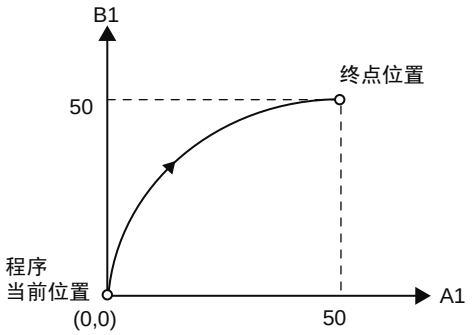
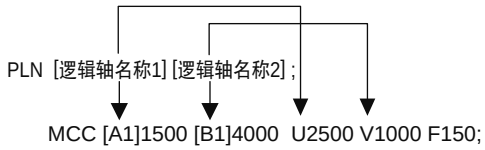


图 8.52 PLN 命令的程序示例

补充 指定圆弧插补及螺旋插补的终点位置、中心位置，请按使用 PLN 命令指定的横轴名称、纵轴名称对应的顺序进行指定。



8.4 程序控制

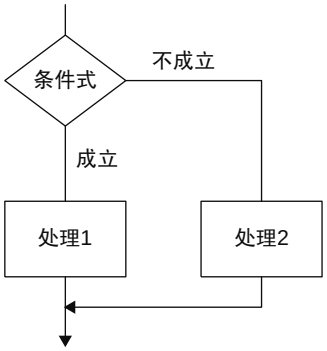
在本节中，将对循环等程序控制命令进行说明。

8.4.1 分支 （IF ELSE IEND）

运动程序	顺序程序
○	○

(1) 动作说明

分支 （IF ELSE IEND）是指在满足条件式时执行 “IF ～ ELSE” 间的程序段，不满足时执行 “ELSE ～ IEND” 间的程序段的命令。
“ELSE” 可省略。此时，如果不满足条件式，则会从 “IEND” 的下一程序段开始继续执行。



重要

分支 （IF ELSE IEND）最多可嵌套 8 层。

(2) 格式

```
IF （条件式）;
   … （处理 1）
ELSE;
   … （处理 2）
IEND;
```

分支中能够使用的条件式如下所示。

(a)Bit 型数据比较

格式	• 数值比较命令使用 == （一致）。 • 左边为寄存器，右边为 0 或 1。 IF MB000000 == 0;“MB000000 = 0 IF MB000000 == 1;“MB000000 = 1
按条件式的运算	• &，|，！可编写 （逻辑与、逻辑或、逻辑非）的逻辑运算。 IF （MB000000 & MB000001） == 1;“MB000000=1 且 MB000001=1 IF （MB000000 & !MB000001） == 1;“MB000000=1 且 MB000001=0 IF （MB000000	MB000001） == 1;“MB000000=1 或 MB000001=1 IF （MB000000	!MB000001） == 1;“MB000000=1 或 MB000001=0
发生语法错误的示例	• 在数值比较命令中编写了 <> （不一致）时 IF MB000000 <> 0; => 语法错误 • 左边为数值或右边为寄存器时 IF 1 == MB000000; => 语法错误 IF MB000000 == MB000001; => 语法错误 • 没有数值比较命令时 IF MB000000; => 语法错误 IF (0); => 语法错误 • 编写了多个数值比较命令时 IF （MB000000 == 0） & （MB000001 == 1）; => 语法错误

(b) 比较整数 / 倍长整数 / 实数型数据

格式	• 可使用全部 （==，<>，>，<，>=，<=）数值比较命令。 • 在左边或右边编写寄存器。 IF MW00000 == 3; “MW00000 = 3 IF ML00000 <> ML00002;“ML00000 ≠ ML00002 IF 1.23456 >= MF00000;“1.23456 ≧ MF00000
按条件式的运算	• 可编写数值运算、逻辑运算。 IF MW00000 == （MW00001/3）; “MW00000 = （MW00001÷3） IF （ML00000 & F0000000H） <> ML00002;“（ML00000 ∧ F0000000H） ≠ ML00002 IF 1.23456 >= （MF00000 * MF00002）; “1.23456 ≧ （MF00000×MF00002）
发生语法错误的示例	• 两边都是常数时 IF 0 == 3; => 语法错误 IF （3.14*2*1000） > 9000.0; => 语法错误 • 没有数值比较命令时 IF MW000000; => 语法错误 IF （-1）; => 语法错误 • 编写了多个数值比较命令时 IF （MW00000 < 0） & （MW000001 > 0）; => 语法错误

(3) 程序示例

例 以下是分支 （IF ELSE IEND）的程序示例。

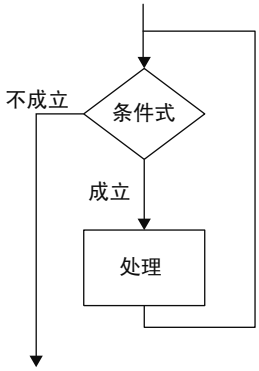
```
IF MB000000== 1;
   MOV [A1] 10000; “MB000000 为 ON 时执行 A1 的定位。
ELSE;
   MOV [B1] 10000; “MB000000 为 OFF 时，执行 B1 的定位。
IEND;
```

8.4.2 循环 （WHILE WEND）

运动程序	顺序程序
○	○

(1) 动作说明

循环 （WHILE　WEND）是指在满足条件式期间重复运行 “WHILE ～ WEND” 程序段，＜条件式＞不成立时，跳到 “WEND” 下一程序段的命令。



重要

- 循环 （WHILE WEND）最多嵌套 8 层。
- 只使用通过 1 次扫描运行的命令创建循环处理的话，有时会在扫描处理中施加负载，发生扫描超时或看门狗超时。
  只使用 1 次扫描运行的命令时，请务必插入 EOX （1 扫描等待命令）或 TIM （等待时间）。
  关于 1 次扫描运行的命令，请参阅 “7.5 命令和执行扫描 ”。

(2) 格式

```
WHILE （条件式）;
…;
（处理）;
…;
WEND;      “ 循环命令的结束
```

循环中能够使用的条件式如下所示。

(a) Bit 型数据比较

格式	• 数值比较命令使用 == （一致）。 • 左边为寄存器，右边为 0 或 1。 WHILE MB000000 == 0; “MB000000 = 0 WHILE MB000000 == 1; “MB000000 = 1

按条件式的运算	• &，|，！可编写 （逻辑与、逻辑或、逻辑非）的逻辑运算。
	WHILE (MB000000 & MB000001) == 1; “MB000000=1 且 MB000001=1
	WHILE (MB000000 & !MB000001) == 1; “MB000000=1 且 MB000001=0
	WHILE (MB000000 | MB000001) == 1; “MB000000=1 或 MB000001=1
	WHILE (MB000000 | !MB000001) == 1; “MB000000=1 或 MB000001=0

发生语法错误的示例	• 在数值比较命令中编写了 <> （不一致）时
	WHILE MB000000 <> 0; ⇒ 语法错误
	• 左边为数值或右边为寄存器时
	WHILE 1 == MB000000; ⇒ 语法错误
	WHILE MB000000 == MB000001; ⇒ 语法错误
	• 没有数值比较命令时
	WHILE MB000000; ⇒ 语法错误
	WHILE (0); ⇒ 语法错误
	• 编写了多个数值比较命令时
	WHILE (MB000000 == 0) & (MB000001 == 1); ⇒ 语法错误

(b) 比较整数 / 倍长整数 / 实数型数据

格式	• 可使用全部 （==，<>，>，<，>=，<=）数值比较命令。 • 在左边或右边编写寄存器。 WHILE MW00000 == 3; “MW00000 = 3 WHILE ML00000 <> ML00002; “ML00000 ≠ ML00002 WHILE 1.23456 >= MF00000; “1.23456 ≧ MF00000
按条件式的运算	• 可编写数值运算、逻辑运算。 WHILE MW00000 == (MW00001/3); “MW00000 = (MW00001÷3) WHILE (ML00000 & F0000000H) <> ML00002; “(ML00000 ∧ F0000000H) ≠ ML00002 WHILE 1.23456 >= (MF00000 * MF00002); “1.23456 ≧ (MF00000×MF00002)

发生语法错误的示例	• 两边都是常数时
	WHILE 0 == 3; ⇒ 语法错误
	WHILE (3.14*2*1000) > 9000.0; ⇒ 语法错误
	• 没有数值比较命令时
	WHILE MW000000; ⇒ 语法错误
	WHILE (-1); ⇒ 语法错误
	• 编写了多个数值比较命令时
	WHILE (MW00000 < 0) & (MW000001 > 0); ⇒ 语法错误

(3) 程序示例

以下是循环 （WHILE　WEND）的程序示例。

例 通过下面的程序画 10 个半径为 50 的圆。

```
MOV [A1] 0 [B1] 0;                          “ 定位
MW00100 = 1;                                “ 计数器预设
INC;                                        “ 指定增量模式
PLN [A1] [B1];                              “ 指定坐标平面
WHILE MW00100 <= 10 ;                       “ 循环命令
    MCW [A1]0 [B1]0 U50. V50. F8000 ;       “ 圆弧插补
    MOV [A1]50. [B1]50.;                    “ 定位
    MW00100 = MW00100 + 1;                  “ 计数器增量
WEND ;                                      “ 循环命令的结束
```

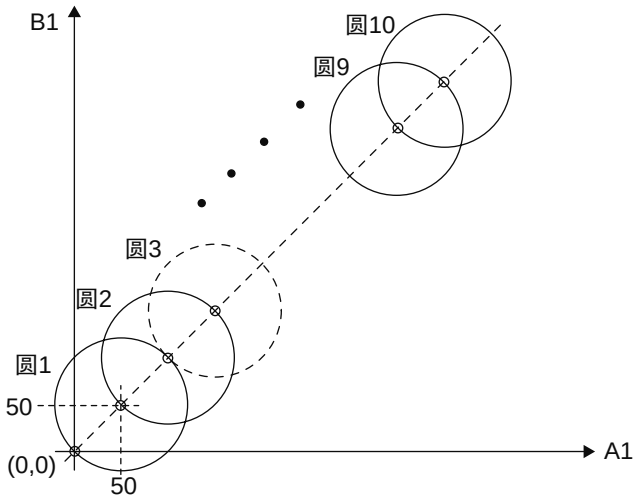


图 8.53 循环 （WHILE WEND）的程序示例

8.4.3 同时执行 （PFORK、JOINTO、PJOINT）

运动程序	顺序程序
○	×

(1) 动作说明

同时执行 （PFORK）是同时执行指定标签的段的命令。各处理同时执行后，会合并到使用 JOINTO 命令指定的标签。同时处理最多可指定 4 轴。关于标签，请参阅 “7.1.2 程序段的格式 (1) 标签 ”。

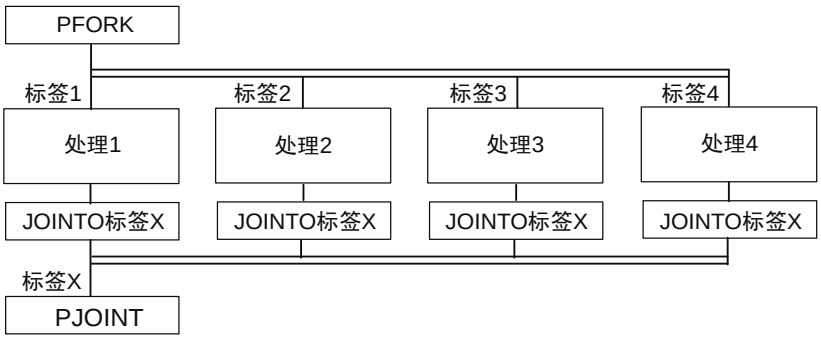


图 8.54 同时执行 （PFORK、JOINTO、PJOINT）的指令方法

基于以上指令使用 PFORK 命令同时执行指定标签的程序段 （处理 1、处理 2、处理 3、…）。各处理同时执行后，会合并到使用 JOINTO 命令指定的标签。

基于 PFORK 命令，可任意指定轴移动命令，以及顺序同时执行轴移动。

(a) 在 PFORK 命令前面指定的命令

在 PFORK 命令前面指定的命令：FMX、ABS/INC、F 指令、IFP、PLN、IAC/IDC… 的值会保留到通过同时执行命令进行处理。此外，同时执行命令也可用于在每个同时处理中设定上述不同的命令值。该处理在合并后，保留最左侧处理的值。

(b) 子程序中的同时执行命令

子程序中的同时执行命令存在以下限制。

- 子程序中的可同时执行命令数最多 2 个。
- MSEE 命令只能编写到使用起始标签指定的程序段中。

```
      PFORK 0001 0002;
0001:MVS [A1]100.[C1]100.;
      JOINTO 0003;
0002:IOW MW10000==1;
      JOINTO 0003;
0003:PJOINT;
```

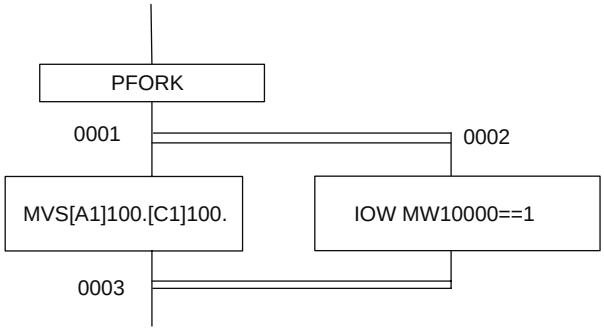


图 8.55 子程序中的同时执行命令

重要

- 程序中有多个相同标签时，发生错误 （“ 标签进行了双重定义 ”）。
- PFORK 的分支数和标签的数不同时，发生错误。

(2) 格式

```
PFORK  标签 1  标签 2  标签 3・・・・・

  标签 1：处理 1
        JOINTO  标签 X；
  标签 2：处理 2
        JOINTO  标签 X；
  标签 3：处理 3
        JOINTO  标签 X；
  标签 X：PJOINT
```

(3) 程序示例

例 以下是同时执行 （PFORK、JOINTO、PJOINT）的程序示例。

```
   MOV [A1]100. [B1]150.;
   MVS [A1]200. [B1]250. F1000;
   PFORK 0001 0002 0003;
0001:MVS [A1]300. [B1]100.
   JOINTO 0004;
0002:MW12345=MW10000+MW10002;
   IOW MB120001==1;
   JOINTO 0004;
0003:MVS [C1]100. [D1]100. F3000;
   JOINTO 0004;
0004:PJOINT;
   MOV [A1]500. [B1]500. [C1]500.;
        ・
        ・
```

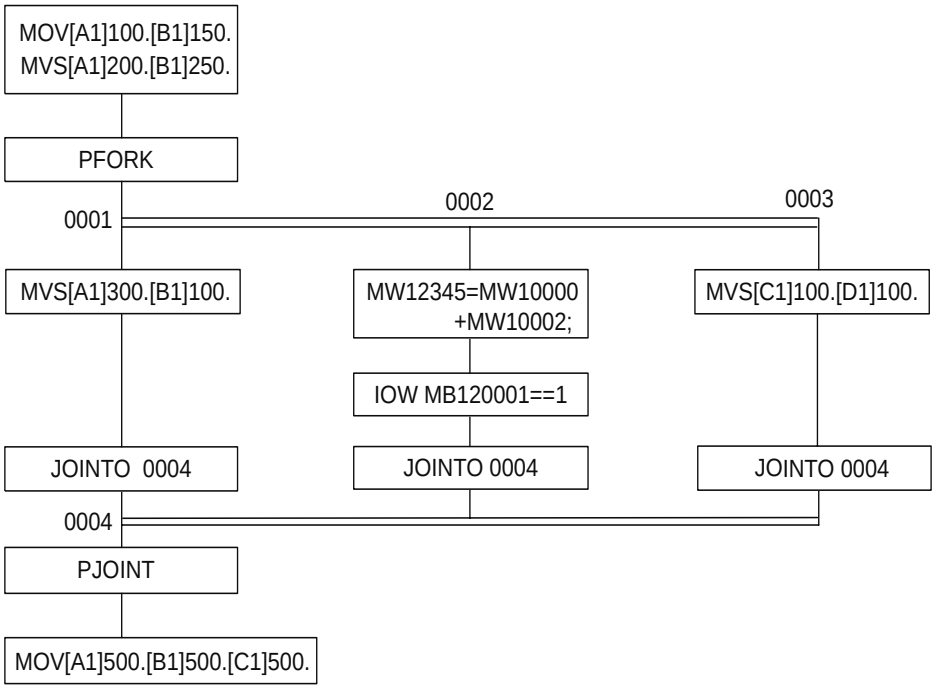


图 8.56 同时执行 （PFORK、JOINTO、PJOINT）的程序示例

8.4.4 选择执行 （SFORK、JOINTO、SJOINT）

运动程序	顺序程序
○	○

(1) 动作说明

选择执行 （SFORK、JOINTO、SJOINT）是在满足指定的条件式时执行 “?” 后面标签程序段的命令。执行各处理后，会合并到使用 JOINTO 命令指定的标签程序段。条件式指定最多为 16 个（含 DEFAULT）。
如果不能满足所有条件式，则执行 “DEFAULT?” 后面的标签程序段。
DEFAULT 的指定只能指定到最后的条件式。
运动程序中可以省略 DEFAULT 指定，但顺序程序中不能省略。

补充

- 在顺序程序中使用 SFORK 命令需 MP2000 系列控制器的系统版本为 Ver2.66 或更高版本。运动程序中，可在所有系统版本下使用。
- 指定 DEFAULT 需使用以下版本的编程工具。

MP2000 系列控制器	支持版本
全部机型	所有版本

MPE720	支持版本
MPE720 Ver.5	MPE720 Ver.5.41 或更高
MPE720 Ver.6	MPE720 Ver.6.06 或更高 MPE720 Ver.6.06 Lite 或更高

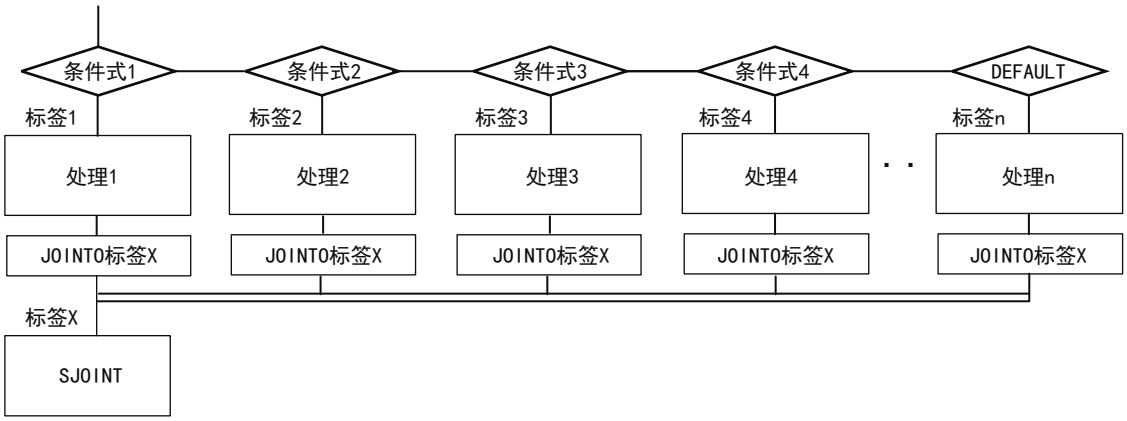


图 8.57 选择执行 （SFORK、JOINTO、SJOINT）的方法

补充

- 条件式从开头的 “ 条件式 1” 开始依次进行判断。 因此，即使有多个条件式成立时，仍会按照第一个条件式成立的标签执行处理。
- 在运动程序中使用 SFORK 时，请务必编写条件式成立的条件。 如果条件不成立，该处理将在 SFORK 命令段中保持等待状态，直到满足成立条件为止。

(2) 格式

```
SFORK  条件式 1? 标签 1，条件式 2? 标签 2，条件式 3? 标签 3，条件式 4? 标签 4，・・，
DEFAULT? 标签 n；

   标签 1：处理 1
           JOINTO  标签 X
   标签 2：处理 2
           JOINTO  标签 X
   标签 3：处理 3
           JOINTO  标签 X
   标签 4：处理 4
           JOINTO  标签 X
            ・
            ・
   标签 n：处理 n
           JOINTO  标签 X
   标签 X：SJOINT
```

选择执行 （SFORK）中能够使用的条件式如下所示。

(a) Bit 型数据比较

格式	• 数值比较命令使用 == （一致）。 • 左边为寄存器，右边为 0 或 1。 MB000000 == 0? 标签 “MB000000 = 0 MB000000 == 1? 标签 “MB000000 = 1
按条件式的运算	• &，|，！可编写 （逻辑与、逻辑或、逻辑非）的逻辑运算。 (MB000000 & MB000001) == 1? 标签 “MB000000=1 且 MB000001=1 (MB000000 & !MB000001) == 1? 标签 “MB000000=1 且 MB000001=0 (MB000000 | MB000001) == 1? 标签 “MB000000=1 或 MB000001=1 (MB000000 | !MB000001) == 1? 标签 “MB000000=1 或 MB000001=0
发生语法错误的示例	• 在数值比较命令中编写了 <> （不一致）时 MB000000 <> 0? 标签 ⇒ 语法错误 • 左边为数值或右边为寄存器时 1 == MB000000? 标签 ⇒ 语法错误 MB000000 == MB000001? 标签 ⇒ 语法错误 • 没有数值比较命令时 MB000000? 标签 ⇒ 语法错误 (0)? 标签 ⇒ 语法错误 • 编写了多个数值比较命令时 (MB000000 == 0) & (MB000001 == 1)? 标签 ⇒ 语法错误

(b) 比较整数 / 倍长整数 / 实数型数据

格式	• 可使用全部 （==，<>，>，<，>=，<=）数值比较命令。 • 在左边或右边编写寄存器。 MW00000 == 3? 标签 “MW00000 ＝ 3 ML00000 <> ML00002? 标签 “ML00000 ≠ ML00002 1.23456 >= MF00000? 标签 “1.23456 ≧ MF00000
按条件式的运算	• 可编写数值运算、逻辑运算。 MW00000 == (MW00001/3)? 标签 “MW00000 ＝ (MW00001÷3) (ML00000 & F0000000H) <> ML00002? 标签 “(ML00000 ∧ F0000000H) ≠ ML00002 1.23456 >= (MF00000 * MF00002)? 标签 “1.23456 ≧ (MF00000×MF00002)
发生语法错误的示例	• 两边都是常数时 0 == 3? 标签 ⇒ 语法错误 (3.14*2*1000) > 9000.0? 标签 ⇒ 语法错误 • 没有数值比较命令时 MW000000? 标签 ⇒ 语法错误 (-1)? 标签 ⇒ 语法错误 • 编写了多个数值比较命令时 (MW00000 < 0) & (MW000001 > 0)? 标签 ⇒ 语法错误

(3) 程序示例

例 以下是选择执行 （SFORK、JOINTO、SJOINT）的程序示例。

```
    MOV [A1]100.[B1]150.;
    MVS [A1]200.[B1]250.F1000;
    SFORK  MW00100==1 ? 0001,MW00100==2 ? 0002,MW00100==3 ? 0003,DEFAULT ? 0004;
0001:MVS [A1]300.[B1]100.F3000;
    JOINTO 0005
0002:MVS [A1]300.[C1]100.F3000;
    JOINTO 0005
0003:MVS [C1]300.[S]100.F3000;
    JOINTO 0005
0004:JOINTO 0005;
0005:SJOINT;
MOV[A1]500.[B1]500.[C1]500.
```

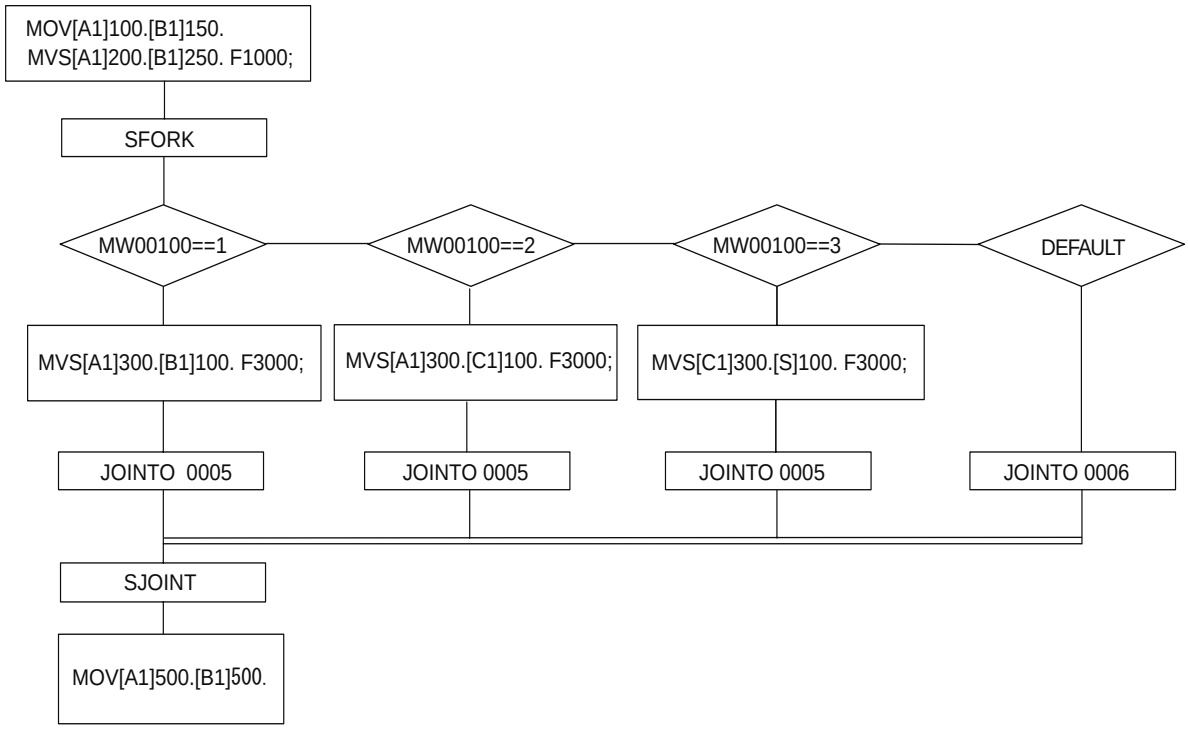


图 8.58 选择执行 （SFORK、JOINTO、SJOINT）的程序示例

8.4.5 运动子程序调用 （MSEE）

运动程序	顺序程序
○	×

(1) 动作说明

运动程序调用 （MSEE）是从运动程序中调用预先记录在运动程序内存中的 “ 子程序 ” 的命令。
子程序调用最多嵌套 8 层。

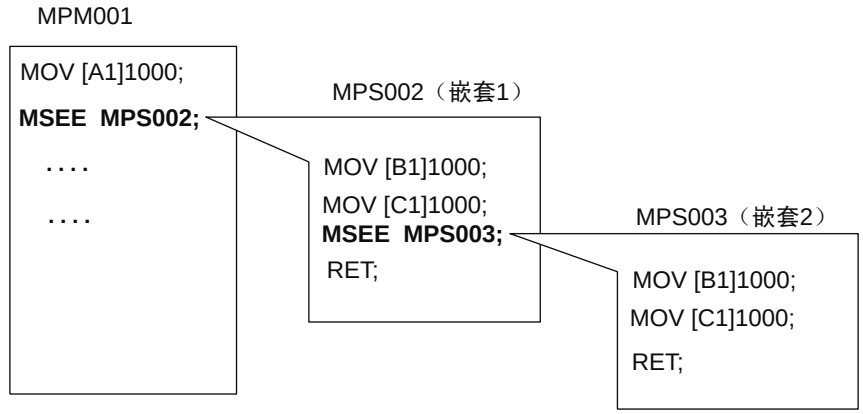


图 8.59 “ 子程序 ”

务必在 “ 子程序 ” 的最后发出子程序结束 （RET）的指令。

重要

■ 子程序中的限制条件

子程序中编写的运动程序有以下限制。

- PFORK 命令中最多可同时运行 2 个。
- 使用 MSEE 命令调用主程序时，主程序不运行。

(2) 格式

MSEE MPS 子程序编号 ;

项目	单位	可使用的数据
子程序编号	—	使用 001 ～ 256 的数值指定

(3) 程序示例

例

以下是 MSEE 命令的程序示例 （调用运动子程序 MPS101 时）。

```
MSEE MPS101;
     指定子程序编号
```

8.4.6 调用顺序子程序 （SSEE）

运动程序	顺序程序
×	○

(1) 动作说明

顺序程序调用 （SSEE）是从顺序程序中调用预先记录在顺序程序内存中的 “ 子程序 ” 执行的命令。
子程序调用最多嵌套 8 层。

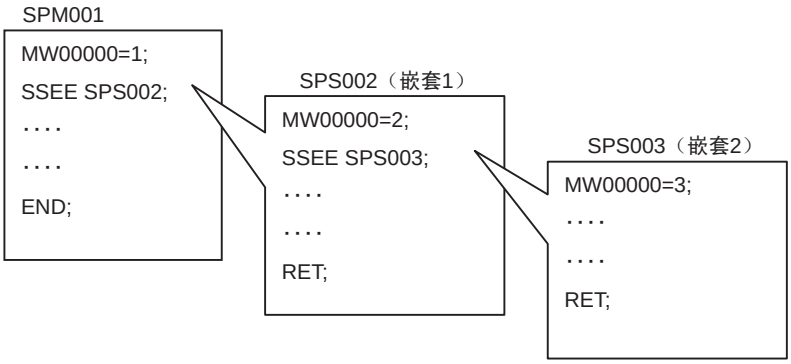


图 8.60 子程序

务必在 “ 子程序 ” 的最后发出子程序结束 （RET）的指令。

重要

■ 子程序中的限制条件

子程序中编写的顺序程序有以下限制。

- 使用 SSEE 命令调用主程序时，主程序不运行。

(2) 格式

SSEE　SPS 子程序编号 ;

项目	单位	可使用的数据
子程序编号	—	使用 001 ～ 256 的数值指定

(3) 程序示例

例

以下是 SSEE 命令的程序示例 （调用顺序子程序 SPS101 时）。

```
SSEE SPS101;
     指定子程序编号
```

8.4.7 从运动程序调用用户函数 （UFC）

运动程序	顺序程序
○	×

(1) 动作说明

从运动程序调用用户函数 （UFC）是通过运动程序调用用户函数 （梯形图程序）的命令。
用户函数执行完成后，进入 UFC 命令的下一程序段。

重要

通过运动程序调用的用户函数中，输出位 YB000000 被用作用户函数的结束判定 (Complete Bit)。

- YB000000 = OFF，用户函数结束时
  用户函数被处理为未完成，通过下一扫描重新调用用户函数。
- YB000000 = ON，用户函数结束时
  用户函数的处理完成后，进入 UFC 命令的下一程序段。

(2) 格式

UFC 函数名称　输入数据、输入地址、输出数据 ;

项目	单位	可使用的数据
函数名称	—	ASCII 8 字节
输入数据	—	最多 16 个数据 （至少 1 个数据）
输入地址	—	最多 1 个地址
输出数据 *1	—	最多 16 个数据 （至少 1 个数据）

*1. 输入地址可省略。[输入数据、输出数据] 表示无输入地址。输入数据、输出数据均最低需 1 个。

(3) 程序示例

例 以下是 UFC 命令的程序示例。

```
UFC KANSUU  MB000000 IW0010 MB000002,  MA00100 ,
   函数名称          输入数据           输入地址
MB000001 MW00200 ML00201;
         输出数据
```

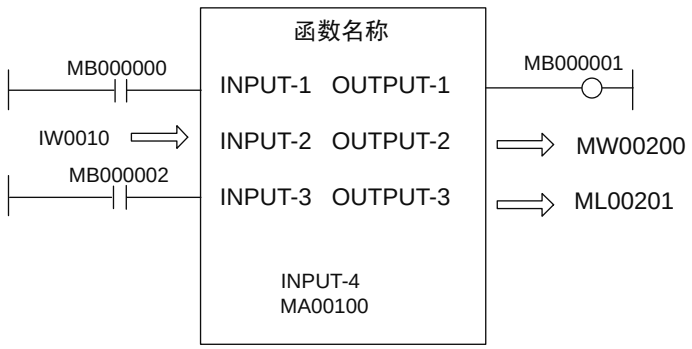
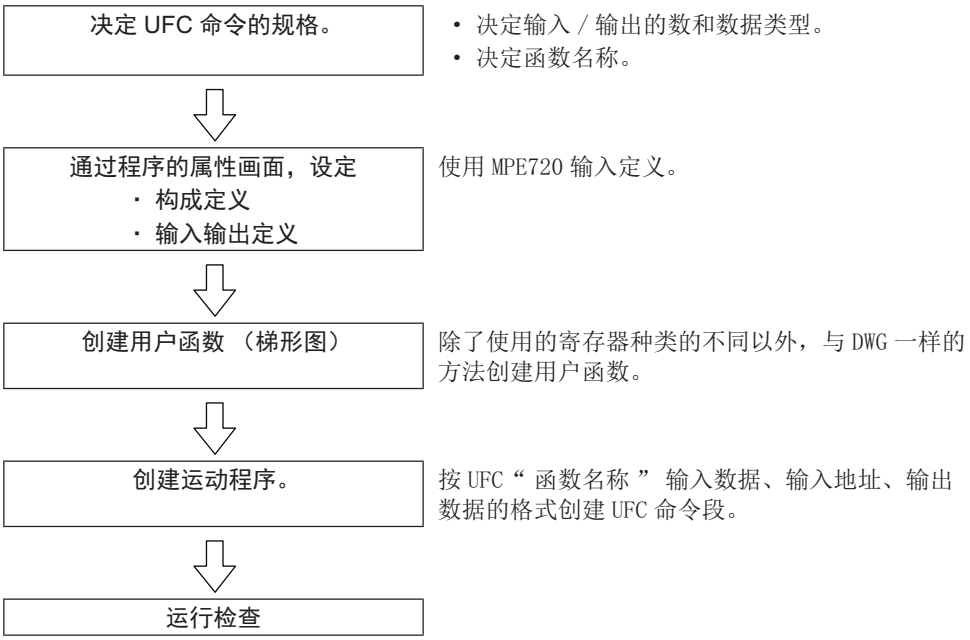


图 8.61 调用用户函数 （UFC）的程序示例

(4)UFC 命令的创建步骤

以下是 UFC 命令的创建步骤。



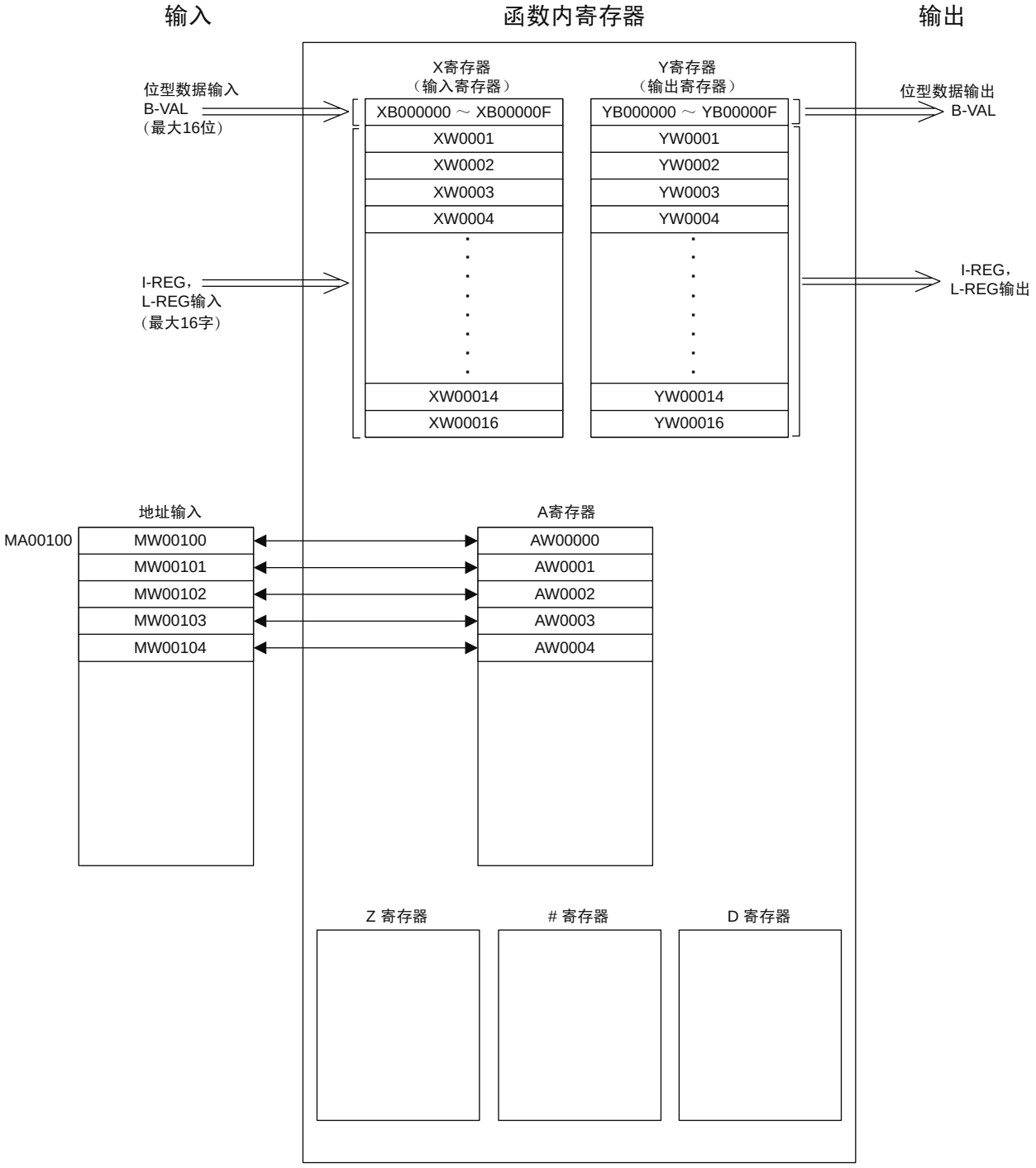
(5) 用户函数中使用的寄存器类型

数据类型如下所示。

类型	型
BIT	bit 型
WORD	整数型
LONG	倍长整数型
FLOAT	实数型

(6) 输入输出寄存器与函数中寄存器的关系

以下是使用 UFC 命令指定的输入输出寄存器与函数寄存器的对应关系。



在各函数中，可以使用下表列出的 11 种寄存器。

表 8.2 函数寄存器

类型	名称	指定方法	内容	特性
X	函数输入寄存器	XB、XW、XL、XFnnnnn	输入至函数 位输入：XB000000 ～ XB0000F 整数输入 ：YW00001 ～ XW00016 倍长整数输入：XL00001 ～ XL00015 寄存器编号 nnnnn 采用十进制。	函数个别
Y	函数输出寄存器	YB、YW、YL、YFnnnnn	输入至函数 位输入：YB000000 ～ YB0000F 整数输入 ：YW00001 ～ YW00016 倍长整数输入：YL00001 ～ YL00015 寄存器编号 nnnnn 采用十进制。	
Z	函数内部寄存器	ZB、ZW、ZL、ZFnnnnn	各函数特有的内部寄存器。 可用作函数的内部处理。 寄存器编号 nnnnn 采用十进制。	
A	函数外部寄存器	AB、AW、AL、AFnnnnn	将地址输入值作为基本地址的外部寄存器。 用于与 （S,M,I,O,#,DAnnnnn）的链接。 寄存器编号 nnnnn 采用十进制。	
#	# 寄存器	#B、#W、#L、#Fnnnnn （#Annnnn）	只能通过程序来引用的寄存器。 只能引用相关 DWG。 实际可利用的范围由用户通过 MPE720 来指定。 寄存器编号 nnnnn 采用十进制。	
D	D 寄存器	DB、DW、DL、DFnnnnn （DAnnnnn）	各 DWG 特有的寄存器。 只能引用相关 DWG。 实际可利用的范围由用户通过 MPE720 来指定。 寄存器编号 nnnnn 采用十进制。	
S	系统寄存器	SB、SW、SL、SFnnnnn （SAnnnnn）	与 DWG 寄存器一样 该寄存器为 DWG/ 函数通用，因此，从优先级不同的 DWG 引用同一函数时，请注意使用方法。	DWG 通用
M	数据寄存器	MB、MW、ML、MFnnnnn （MAnnnnn）		
I	输入寄存器	IB、IW、IL、IFhhhh （IAhhhh）		
O	输出寄存器	OB、OW、OL、OFhhhh （OAhhhh）		
C	常数寄存器	CB、CW、CL、CFnnnnn （CAnnnnn）		

（注）也可以在函数内部使用 SA、MA、IA、OA、DA、#A、CA。

以下是输入输出寄存器交换的例子。

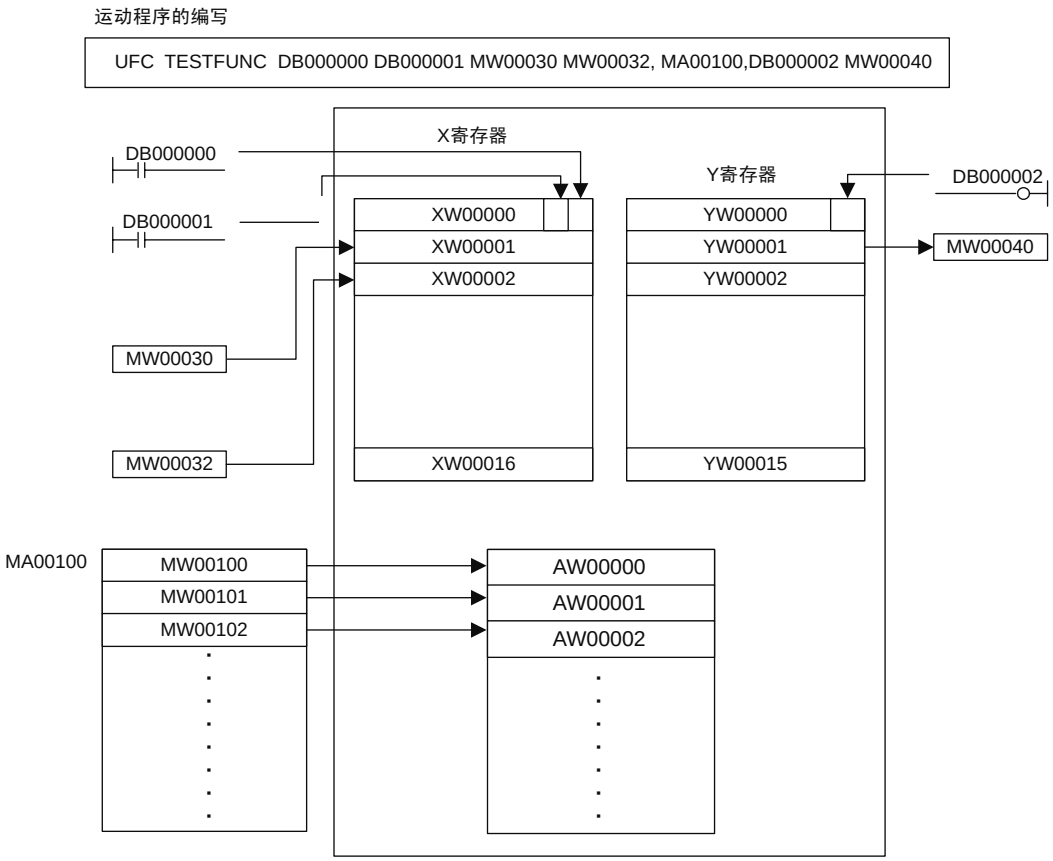


图 8.62 运动程序的编写

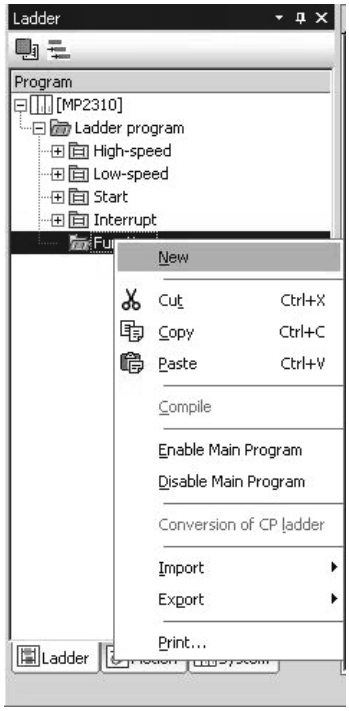
(7) 创建用户函数

以如下规格的用户函数为例对创建用户函数的步骤进行说明。

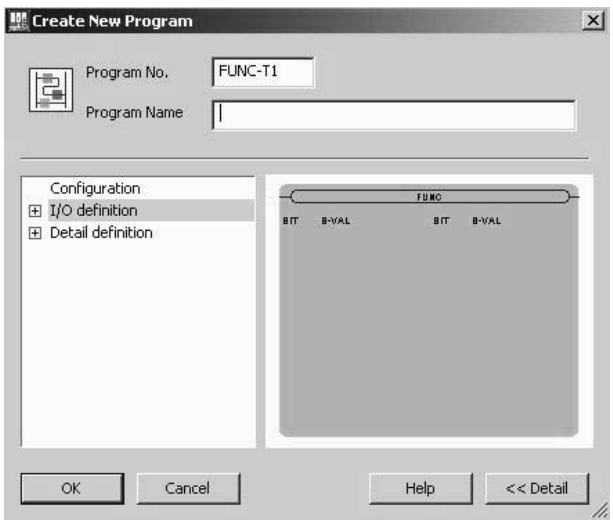
规格	运动程序
指定伺服轴 No. 与速度数据，设定成运动设定参数 “快进速度 OLxx10”。	MW00030 ＝伺服轴 No. （1 或 2） ML00032 ＝快进速度 UFC FUNC-T1 MW00030 ML00032,,DB000001;

创建用户函数的步骤如下所示。

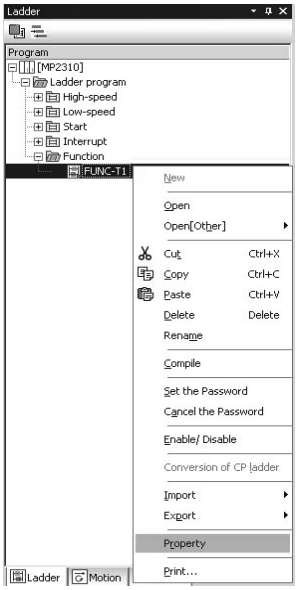
1. 打开 ***Ladder***（梯形图）子窗口，鼠标右键菜单点击 ***Ladder Program*** 下的 ***Function***，并在下拉菜单中选择 ***New***。



2. 通过 ***Create New Program*** 画面，在 ***Program No.*** 中输入 ***FUNC-T1***，点击 ***OK***。

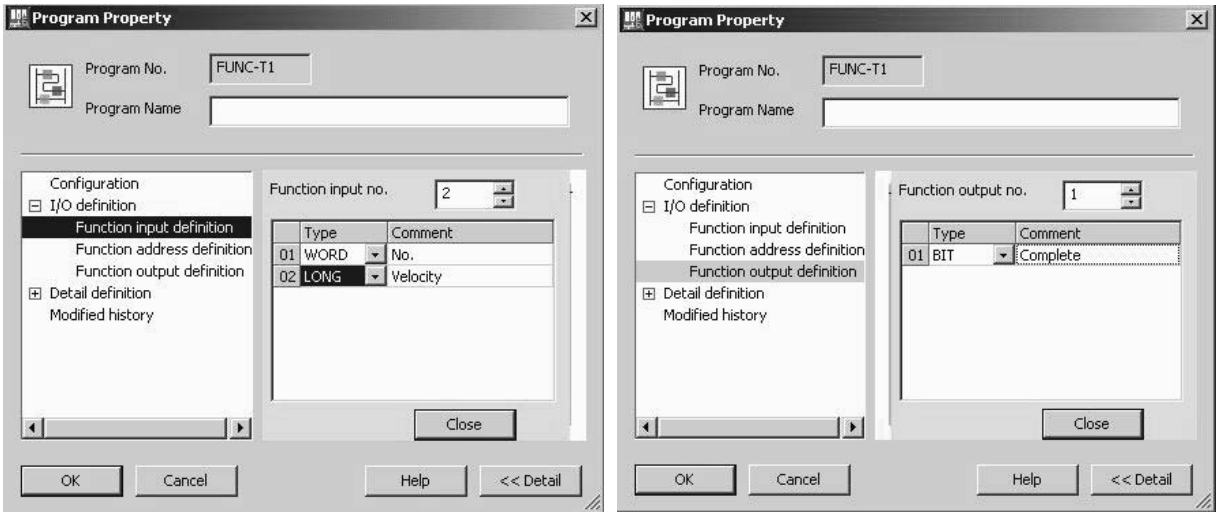


3. ***Ladder***子窗口中将出现一个空白的***Ladder Program***项，鼠标右键点击***FUNC-T1***，并在下拉菜单中选择***Property***（属性）。

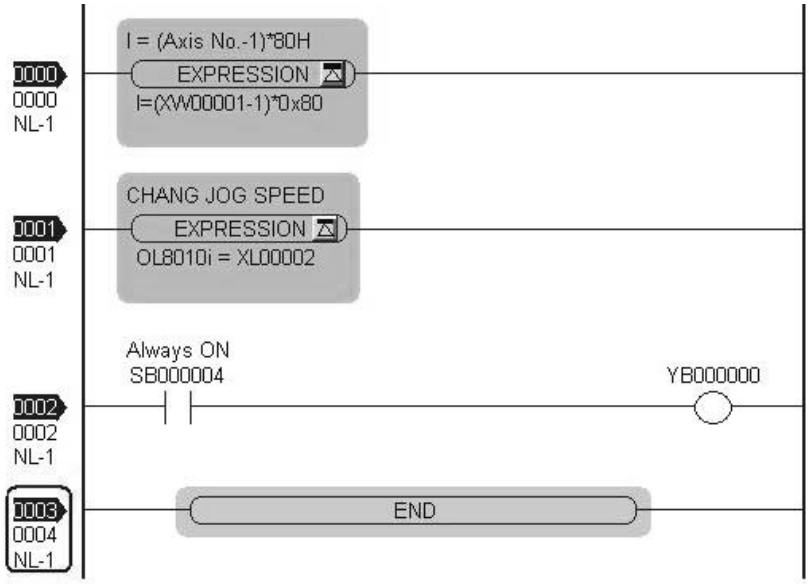


4. 在***Program Property***窗口中，点击***I/O definition***下的***Function input definition***和***Function output definition***选项，分别设定函数的输入和输出的个数以及数据类型。

（例）UFC FUNC-T1 MW00030 ML00032,,DB000001;时，进行如下设定。



5. 关闭 ***DWG Configuration Definition*** 画面，通过梯形图编辑画面编辑用户函数程序。



6. 在梯形图菜单的 ***Compile*** （编译）中选择 ***Compile***。

7. 通过 ***Motion Editor*** 窗口创建调用用户函数的程序。

```
MW00030 = 1;
ML00032 = 500;
UFC FUNC-T1 MW00030 ML00032, , DB000001;
END;
```

至此，从运动程序中调用的用户函数创建完毕。
请运行运动程序并确认动作。

8.4.8 从顺序程序调用用户函数 （FUNC）

运动程序	顺序程序
×	○

(1) 动作说明

从顺序程序调用用户函数 （FUNC）是通过顺序程序调用用户函数 （梯形图程序）的命令。

(2) 格式

```
UFC  函数名称  输入数据 1  输入数据 2  输入数据 3・・・,  输入地址,
               输出数据 1  输出数据 2  输出数据 3・・・ ;
```

项目	单位	可使用的数据
函数名称	—	ASCII 8 字节
输入数据	—	最多 16 个数据 （至少 1 个数据）
输入地址	—	最多 1 个地址
输出数据	—	最多 16 个数据 （至少 1 个数据）

（注） 1. 以上输入数据、输出数据分别可编写多个。
（最少需 1 个）。输入地址可省略。省略输入地址时，只编写 “, ”。
2. 通过上述指令调用用户函数。FUNC 命令会进入 “FUNC” 的下一程序段，与用户函数的执行是否结束无关。

(3) 程序示例

例 以下是 FUNC 命令的程序示例。
本例中采用 3 个输入数据、1 个输入地址、3 个输出数据。

```
FUNC KANSUU  MB000000 IW0010 MB000020,  MA00100,
   函数名称          输入数据            输入地址
MB000001 MW00201 ML00202;
     输出数据
```

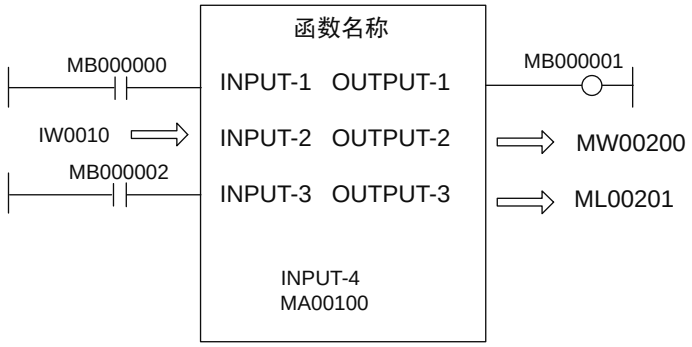


图 8.63 调用用户函数 （FUFC）的程序示例

8.4.9 程序退出 （END）

运动程序	顺序程序
○	○

(1) 动作说明

程序退出 （END）是退出程序运行的命令。不能在该程序段中重复其它命令。
程序在执行程序退出命令后结束运行。上一程序段为移动指令时，程序在完成到位检查后结束运行。

(2) 格式

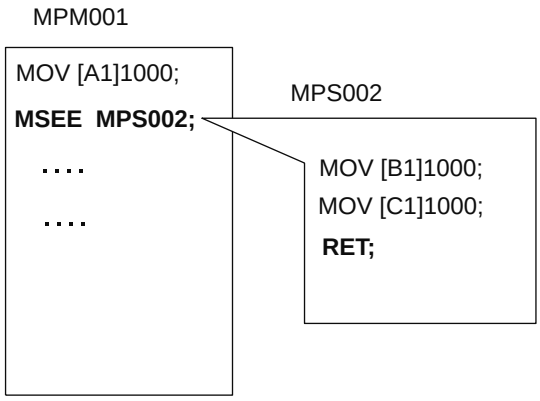
END;
程序退出

8.4.10 子程序退出 （RET）

运动程序	顺序程序
○	○

(1) 动作说明

子程序退出 （RET）是退出子程序的命令。
使用 RET 命令结束调用子程序的操作后，程序的运行进入调用该子程序的程序 （主或子程序）的运动子程序调用 （MESS）或顺序子程序调用 （SSEE）的下一程序段。



(2) 格式

RET;
子程序退出

8.4.11 时间等待 （TIM）

运动程序	顺序程序
○	×

(1) 动作说明

时间等待 （TIM）是在进入下一程序段前等待指定时间的命令。
能够指定的时间范围为 0.00 ～ 600.00s。

(2) 格式

TIM　T等待时间 ;

项目	单位	可使用的数据
等待时间	0.01s	・基于直接数值的直接指定 ・基于整数型寄存器的间接指定

(3) 程序示例

例 以下是 TIM 命令的程序示例。

```
MOV [A1]100;
TIM T250;
    2.5s
```

定位完成后，执行 TIM 命令。

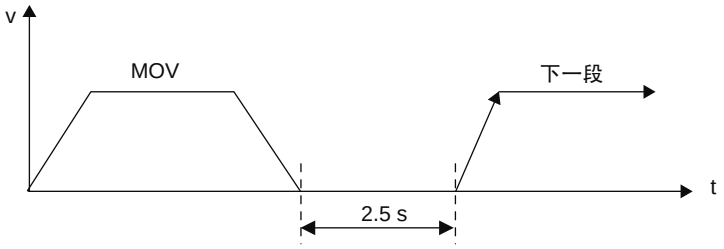


图 8.64 时间等待 （TIM）的程序示例

8.4.12 输入输出变量等待 （IOW）

运动程序	顺序程序
○	×

(1) 动作说明

输入输出变量等待 （IOW），是指通过条件式变成指定状态等待的命令。当条件成立时，进入下一程序段。

(2) 格式

```
IOW     IB00001&IB00002 == 1;
            A
```

内容	用途	可使用的数据
A	条件式	· 所有的整数型，倍长整数型，实数型寄存器 （#，C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数

IOW 中能够使用的条件式如下所示。

(a) Bit 型数据比较

格式	• 数值比较命令使用 == （一致）。 • 左边为寄存器，右边为 0 或 1。 IOW MB000000 == 0; “MB000000 = 0 IOW MB000000 == 1; “MB000000 = 1
按条件式的运算	• &，|，！可编写 （逻辑与、逻辑或、逻辑非）的逻辑运算。 IOW (MB000000 & MB000001) == 1; “MB000000=1 且 MB000001=1 IOW (MB000000 & !MB000001) == 1; “MB000000=1 且 MB000001=0 IOW (MB000000 | MB000001) == 1; “MB000000=1 或 MB000001=1 IOW (MB000000 | !MB000001) == 1; “MB000000=1 或 MB000001=0
发生语法错误的示例	• 在数值比较命令中编写了 <> （不一致）时 IOW MB000000 <> 0; => 语法错误 • 左边为数值或右边为寄存器时 IOW 1 == MB000000; => 语法错误 IOW MB000000 == MB000001; => 语法错误 • 没有数值比较命令时 IOW MB000000; => 语法错误 IOW (0); => 语法错误 • 编写了多个数值比较命令时 IOW (MB000000 == 0) & (MB000001 == 1); => 语法错误

② 比较整数 / 倍长整数 / 实数型数据

格式	• 可使用全部 （==，<>，>，<，>=，<=）数值比较命令。 • 在左边或右边编写寄存器。 IOW MW00000 == 3; “MW00000 ＝ 3 IOW ML00000 <> ML00002; “ML00000 ≠ ML00002 IOW 1.23456 >= MF00000; “1.23456 ≧ MF00000
按条件式的运算	• 可编写数值运算、逻辑运算。 IOW MW00000 == (MW00001/3); “MW00000 ＝ (MW00001÷3) IOW (ML00000 & F0000000H) <> ML00002; “(ML00000 ∧ F0000000H) ≠ ML00002 IOW 1.23456 >= (MF00000 * MF00002); “1.23456 ≧ (MF00000×MF00002)
发生语法错误的示例	• 两边都是常数时 IOW 0 == 3; ⇒ 语法错误 IOW (3.14*2*1000) > 9000.0; ⇒ 语法错误 • 没有数值比较命令时 IOW MW000000; ⇒ 语法错误 IOW (-1); ⇒ 语法错误 • 编写了多个数值比较命令时 IOW (MW00000 < 0) & (MW000001 > 0); ⇒ 语法错误

(3) 程序示例

例 以下是 IOW 命令的程序示例。

```
IOW (MB001001&MB001002)== 1;
MOV [A1]1000;
```

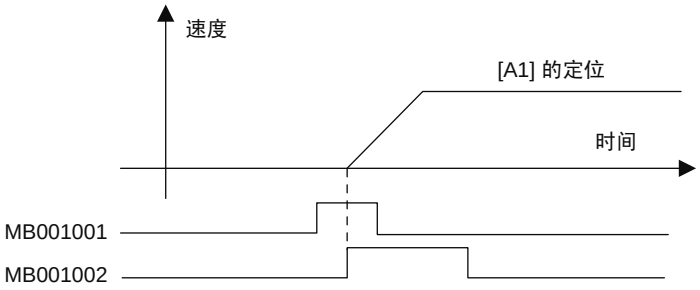


图 8.65 输入输出变量等待 （IOW）的程序示例

8.4.13 1 次扫描等待 （EOX）

运动程序	顺序程序
○	×

(1) 动作说明

1 次扫描等待 （EOX）是使得程序执行 1 次扫描时等待的命令。
EOX 命令后面的程序段在下一次扫描时执行。

(2) 格式

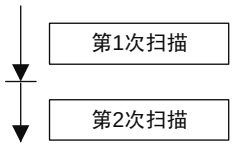
```
EOX;
```

(3) 程序示例

以下是 EOX 命令的程序示例。

例 (a) 与顺序命令组合使用时

```
MW00000=100;
OB00010=1;
EOX;
OB00011=0;
```



例 (b) WHILE 命令的使用示例

```
WHILE OB00010==1;
EOX;
WEND;
```

8.4.14 单段无效 （SNGD）/ 单段有效 （SNGE）

运动程序	顺序程序
○	×

(1) 动作说明

单段无效 （SNGD）/ 单段有效 （SNGE）是指定是否调试运行中单步动作的命令。
即使是在 “ 单段运行模式 ” 下，对于 SNGD 和 SNGE 括起来的程序段，仍不执行 “ 单段停止 ”，反而连续运行。

■ 单段有效运行模式

在单段运行模式下，分段执行单段停止。

(2) 格式

```
SNGD;
需连续执行的部分
SNGE;
```

(3) 程序示例

以下是 SNGD/SNGE 命令的程序示例。

```
MVS [A1]0 [B1]0;
SNGD;
MVS [A1]100 [B1]200;  “①”
MB000101 = 1;         “②”
MB000102 = 1;         “③”
SNGE;
MB000103 = 1;
```

上述示例中，即使是在 “ 单段运行模式 ” 下，对于 SNGD 和 SNGE 括起来的程序段①～③，仍不执行 “ 单段停止 ”，反而连续运行。

8.5 数值运算

本节对数值运算命令进行说明。
数值运算的优先顺序，请参照 “7.4 运算的优先顺序 ”。

8.5.1 代入 （=）

运动程序	顺序程序
○	○

(1) 动作说明

代入 （=）将右边的运算结果代入公式左边的寄存器。

(2) 格式

结果 = 运算式 ;

A B

内容	用途	可使用的寄存器
A	结果	· 所有位型、整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） · 附带字符的上述寄存器 · 附加字符寄存器
B	运算式	· 所有位型、整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） · 附带字符的上述寄存器 · 附加字符寄存器 · 常数

(3) 程序示例

以下是代入 （=）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	MB001000=1;	
W	MW00100=12345;	
L	ML00100=1234567;	
F	MF00100=1.2345;	

8.5.2 加法 （+）

运动程序	顺序程序
○	○

(1) 动作说明

加法 （+）对右边的整数或实数执行加法，将结果存放到左边的寄存器中。右边的加法还可不使用寄存器，使用常数。整数或实数混合时，其数据类型也与左侧数据相同。

(2) 格式

MW00101 = MW00100 + 12345;
A B C

内容	用途	可使用的寄存器
A	数据输出	· 所有的整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	数据输入	· 所有的整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数
C	加法数据	

(3) 程序示例

以下是加法 （+）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	MW00101=MW00100+12345;	
L	ML00106=ML00102+ML00104;	
F	MF00202=MF00200+1.23456;	

重要

数据类型不同的变量运算时，结果会因左边的数据类型而异。详细信息请参阅“6.1.2 全局变量和局部变量 ”。

8.5.3 减法 （-）

运动程序	顺序程序
○	○

(1) 动作说明

减法 （-）对右边的整数或实数执行减法，将结果存放到左边的寄存器中。右边的加法还可不使用寄存器，而使用常数。 整数或实数混合时，其数据类型也与左侧数据相同。

(2) 格式

MW00101 ＝ MW00100 － 12345;
A B C

内容	用途	可使用的寄存器
A	数据输出	・ 所有的整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器
B	数据输入	・ 所有的整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器 ・ 常数
C	减法数据	

(3) 程序示例

以下是减法 （-）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	－	－
W	MW00101=MW00100-12345;	SUB [WLF]SrcA MW00100 [WLF]Dest MW00101 [WLF]SrcB 12345
L	ML00106=ML00102-ML00104;	SUB [WLF]SrcA ML00102 [WLF]Dest ML00106 [WLF]SrcB ML00104
F	MF00202=MF00200-1.23456;	SUB [WLF]SrcA MF00200 [WLF]Dest MF00202 [WLF]SrcB 1.234560E+000

8.5.4 乘法 （*）

运动程序	顺序程序
○	○

(1) 动作说明

乘法 （*）对右边的整数或实数执行乘法，将结果存放到左边的寄存器中。右边的乘法还可不使用寄存器，而使用常数。 整数或实数混合时，其数据类型也与左侧数据相同。

(2) 格式

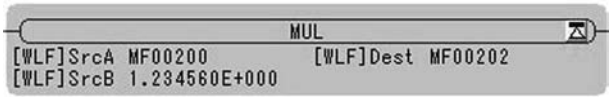
MW00101 = MW00100 * 12345;
A B C

内容	用途	可使用的寄存器
A	数据输出	· 所有的整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	数据输入	· 所有的整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数
C	乘法数据	

(3) 程序示例

以下是乘法 （*）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	MW00102=MW00100*MW00101	
L	ML00106=ML00102*ML00104;	
F	MF00202=MF00200*1.23456;	

8.5.5 除法 （/）

运动程序	顺序程序
○	○

(1) 动作说明

除法 （/）对右边的整数或实数执行除法，将结果存放到左边的寄存器中。右边的除法还可不使用寄存器，而使用常数。整数或实数混合时，其数据类型也与左侧数据相同。

(2) 格式

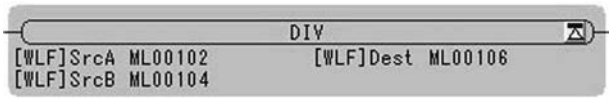
MW00101 ＝ MW00100 / 12345;
A B C

内容	用途	可使用的寄存器
A	数据输出	· 所有的整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	数据输入	· 所有的整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数
C	除法数据	

(3) 程序示例

以下是除法 （/）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	MW00102=MW00100/MW00101;	
L	ML00106=ML00102/ML00104;	
F	MF00202=MF00200/1.23456;	

8.5.6 除余 （MOD）

运动程序	顺序程序
○	○

(1) 动作说明

除余 （MOD）如果通过除法命令的下一程序段进行指定，则 MOD 指令会在指定变量中存放余数。
余数的数据类型与左侧数据相同。

(2) 格式

```
MW00001 = 1000 / 999;
MW00002 = MOD;
```

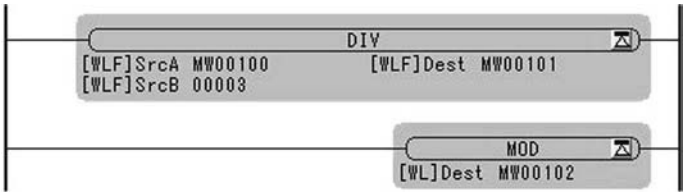
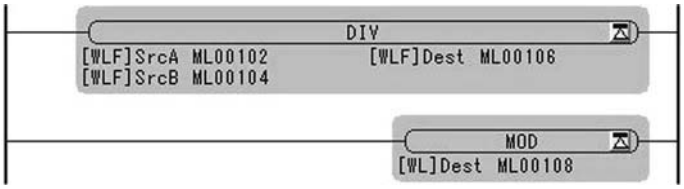
A

内容	用途	可使用的寄存器
A	数据输出	・ 所有的整数型、倍长整数型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器

(3) 程序示例

以下是 MOD 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	－	－
W	MW00101=MW00100/3; MW00102=MOD;	
L	ML00106=ML00102/ML00104; ML00108=MOD;	
F	－	－

（例）倍长整数

```
ML00106=ML00100*ML00102/ML00104;
(173575)(100000)(60000)(34567)
ML00108=MOD;
(32975)
```

重要

MOD 命令，请务必通过除法命令的下一程序段进行指定。不在除法命令的程序段执行时，无法保证运算结果。

8.6 逻辑运算

在本节中，将对执行位或整数的逻辑运算的命令进行说明。还可以进行与数值运算混合的公式的运算，但无法执行实数运算。
有关运算的优先顺序，请参阅 “7.4 运算的优先顺序 ”。

8.6.1 逻辑或 （|）

运动程序	顺序程序
○	○

(1) 动作说明

逻辑或 （|）执行指定为前一运算结果的寄存器的逻辑或，保留运算结果。实数寄存器无法使用。

表 8.3 逻辑或的直值表 （A=B|C）

B	C	A
0	0	0
0	1	1
1	0	1
1	1	1

(2) 格式

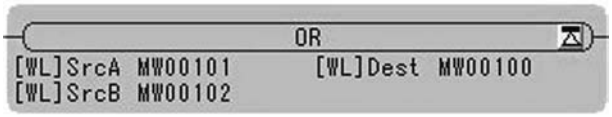
MW00100 = DW00102 | AAAAH;
A B C

内容	用途	可使用的寄存器
A	数据输出	· 所有位型、整数型、倍长整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B、C	数据输入	· 所有位型、整数型、倍长整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数

(3) 程序示例

以下是逻辑或 （|）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	MB001000=MB001010|MB001011;	
W	MW00100=MW00101|MW00102	
L	ML00106=ML00102|ML00104;	
F	–	–

8.6.2 逻辑与 （&）

运动程序	顺序程序
○	○

(1) 动作说明

逻辑与 （&）执行指定为前一运算结果的寄存器的逻辑与，保留运算结果。实数寄存器无法使用。

表 8.4 逻辑与的直值表 （A=B&C）

B	C	A
0	0	0
0	1	0
1	0	0
1	1	1

(2) 格式

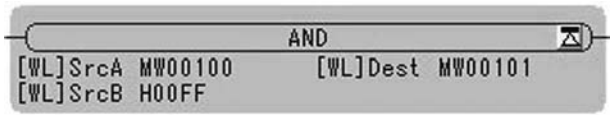
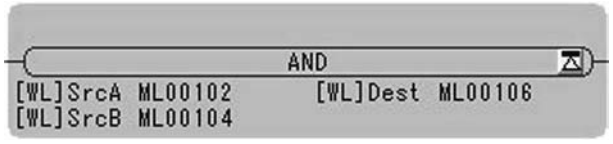
MW00100 ＝ DW00102 & AAAAH;
A B C

内容	用途	可使用的寄存器
A	数据输出	・ 所有位型、整数型、倍长整数型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器
B, C	数据输入	・ 所有位型、整数型、倍长整数型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器 ・ 常数

(3) 程序示例

以下是逻辑与 （&）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	MB001000=MB001010&MB001011;	
W	MW00101=MW00100&00FFH;	
L	ML00106=ML00102&ML00104;	
F	–	–

8.6.3 异或 （^）

运动程序	顺序程序
○	○

(1) 动作说明

异或 （^）执行指定为前一运算结果的寄存器的异或，保留运算结果。实数寄存器无法使用。

表 8.5 异或的真值表 （A= B ^ C）

B	C	A
0	0	0
0	1	1
1	0	1
1	1	0

(2) 格式

MW00100 = DW00102 ^ AAAAH;
A B C

内容	用途	可使用的寄存器
A	数据输出	· 所有的整数型、倍长整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B, C	数据输入	· 所有的整数型、倍长整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数

(3) 程序示例

以下是异或运算 （^）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	MW00101=MW00100 ^ 00FFH;	
L	ML00106=ML00102 ^ ML00104;	
F	–	–

8.6.4 逻辑非 （!）

运动程序	顺序程序
○	○

(1) 动作说明

逻辑非 （!）将指定寄存器的数据进行反转，保留运算结果。实数寄存器无法使用。

(2) 格式

MB001000 ＝ ! MB001010;
A B

内容	用途	可使用的寄存器
A	数据输出	· 所有位型、整数型、倍长整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	数据输入	· 所有位型、整数型、倍长整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数*

* 不能指定位型的常数。

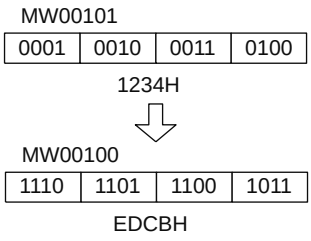
(3) 程序示例

以下是逻辑非 （!）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	MB001000=!MB001010;	
W	MW00100=!MW00101;	
L	ML00100=!ML00102	
F	–	–

（例）MW00100=!MW00101;



8.7 数值比较

本节将说明用于条件式判别用的数值比较命令。

8.7.1 数值比较 （==，<>，>，<，>=，<=）

运动程序	顺序程序
○	○

(1) 动作说明

数值比较 （==，<>，>，<，>=，<=）用于判别分支 （IF ELSE IEND），循环 （WHILE WEND），输入输出等待 （IOW）等 “ 条件式 ”。比较命令有以下 6 种。

比较命令	含义
= =	一致
< >	不一致
>	较大
<	较小
> =	以上
< =	以下

(2) 格式

```
IF MB001000 == 1;
   A
```

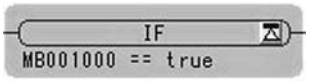
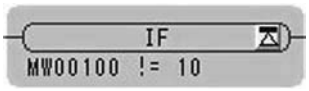
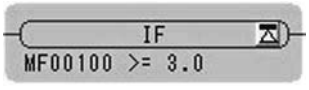
内容	用途	可使用的寄存器
A	条件式	· 所有位型 *1、整数型、倍长整数型、实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器

*1. 位型条件式中，仅一致 （= =）有效。

(3) 程序示例

以下是数值比较命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	IF MB001000==1;	
W	IF MW00100<>10;	
L	IF ML00100>10000;	
F	IF MF00100>=3.0;	

数值比较命令中能够使用的条件式如下所示。

(a) Bit 型数据比较

格式	• 数值比较命令使用 == （一致）。 • 左边为寄存器，右边为 0 或 1。 IF MB000000 == 0;“MB000000 = 0 IF MB000000 == 1;“MB000000 = 1

按条件式的运算	• &，|，！可编写 （逻辑与、逻辑或、逻辑非）的逻辑运算。
	IF (MB000000 & MB000001) == 1;“MB000000=1 且 MB000001=1
	IF (MB000000 & !MB000001) == 1;“MB000000=1 且 MB000001=0
	IF (MB000000 | MB000001) == 1;“MB000000=1 或 MB000001=1
	IF (MB000000 | !MB000001) == 1;“MB000000=1 或 MB000001=0

发生语法错误的示例	• 在数值比较命令中编写了 <> （不一致）时
	IF MB000000 <> 0; ⇒语法错误
	• 左边为数值或右边为寄存器时
	IF 1 == MB000000; ⇒语法错误
	IF MB000000 == MB000001; ⇒语法错误
	• 没有数值比较命令时
	IF MB000000; ⇒语法错误
	IF (0); ⇒语法错误
	• 编写了多个数值比较命令时
	IF (MB000000 == 0) & (MB000001 == 1); ⇒语法错误

(b) 比较整数 / 倍长整数 / 实数型数据

格式	• 可使用全部 （==，<>，>，<，>=，<=）数值比较命令。 • 在左边或右边编写寄存器。 IF MW00000 == 3; “MW00000 = 3 IF ML00000 <> ML00002;“ML00000 ≠ ML00002 IF 1.23456 >= MF00000;“1.23456 ≧ MF00000
按条件式的运算	• 可编写数值运算、逻辑运算。 IF MW00000 == (MW00001/3); “MW00000 = (MW00001÷3) IF (ML00000 & F0000000H) <> ML00002;“(ML00000 ∧ F0000000H) ≠ ML00002 IF 1.23456 >= (MF00000 * MF00002); “1.23456 ≧ (MF00000×MF00002)

发生语法错误的示例	• 两边都是常数时
	IF 0 == 3; ⇒语法错误
	IF (3.14*2*1000) > 9000.0; ⇒语法错误
	• 没有数值比较命令时
	IF MW000000; ⇒语法错误
	IF (-1); ⇒语法错误
	• 编写了多个数值比较命令时
	IF (MW00000 < 0) & (MW000001 > 0); ⇒语法错误

8.8 数据操作

本节将说明执行数据迁移、移动、初始化的数据操作命令。

8.8.1 位右移 （SFR）

运动程序	顺序程序
○	○

(1) 动作说明

位右移 （SFR）对于以起始位编号和位宽度指定的位列，向右移动要求的位数。

(2) 格式

SFR MB001000 N5 W10;
　　A　　B　C

内容	用途	可使用的寄存器
A	起始位	· 所有位型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	变动数	· 所有整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数
C	bit 宽度	

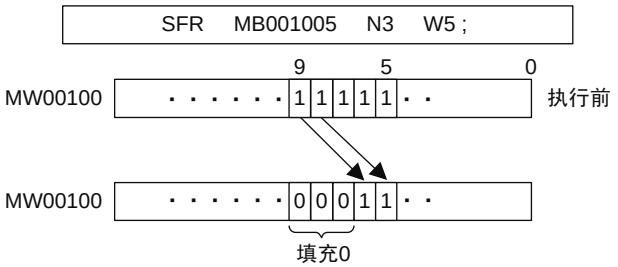
(3) 程序示例

以下是 SFR 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	SFR MB001000 N5 W10;	SHFTR [B]Adr MB001000 [W]Width 00010 [W]Num 00005
L	–	–
F	–	–

（例）使得以 MB001005 （MW00100 的位 5）作为起始的位宽度 5 的数据向右移 3 位



补充　使用 SFR 命令的移动数大于位宽时，指定的位宽数据全部变成 0。

8.8.2 位左移 （SFL）

运动程序	顺序程序
○	○

(1) 动作说明

位左移 （SFL）对于以起始位编号和位宽度指定的位列，向左移动指定的位数。

(2) 格式

SFL MB001000 N5 W10;
A B C

内容	用途	可使用的寄存器
A	起始位	· 所有位型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	变动数	· 所有整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数
C	bit 宽度	

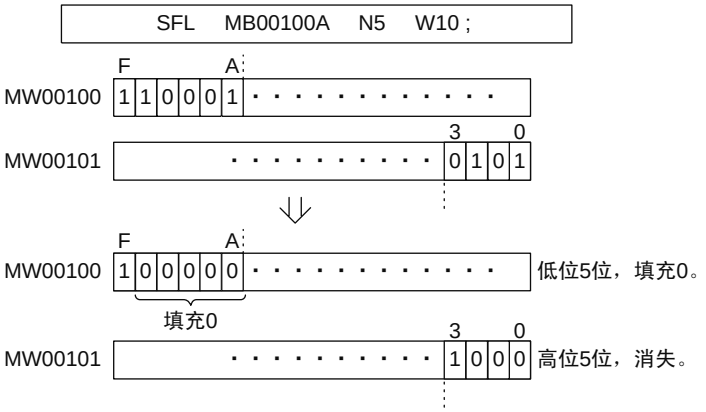
(3) 程序示例

以下是 SFL 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	SFL MB001000 N5 W10;	
L	–	–
F	–	–

（例）使得以 MB00100A （MW00100 的位 A）作为起始的位宽度 10 的数据向左移 5 位



补充

使用 SFR 命令的移动数大于位宽时，指定的位宽数据全部变成 0。

8.8.3 段传输 （BLK）

运动程序	顺序程序
○	○

(1) 动作说明

段传输 （BLK）从传送源寄存器的起始向传送目标寄存器的起始传输指定数的字数据。

(2) 格式

BLK MW00100 DW00100 W10;
A B C

内容	用途	可使用的寄存器
A	输出源起始寄存器	· 所有整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	传输目标起始寄存器	
C	传输程序段数	· 所有整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数

(3) 程序示例

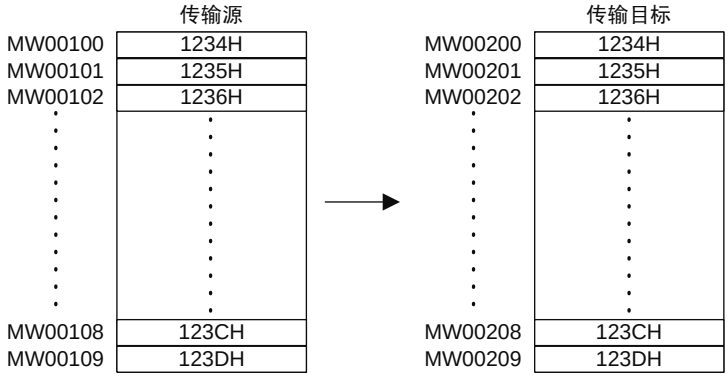
以下是 BLK 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	BLK MW00100 DW00100 W10;	
L	–	–
F	–	–

（例）将 MW00100 ～ MW00109 传输到 MW00200 ～ MW00209。

BLK MW00100 MW00200 W10;



补充

只要传输源与目标寄存器不重叠，传输源数据就会直接被传输到目标寄存器。传输源与传输目标有重叠时，有时不会将传输源数据直接传输到目标寄存器。

8.8.4 清除 （CLR）

运动程序	顺序程序
○	○

(1) 动作说明

清除 （CLR）将起始寄存器指定程序段数的字数据清零。

(2) 格式

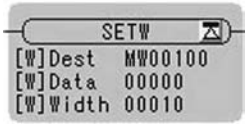
CLR MW00100 W10;
A B

内容	用途	可使用的寄存器
A	数据清除起始	· 所有整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	程序段数	· 所有整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数

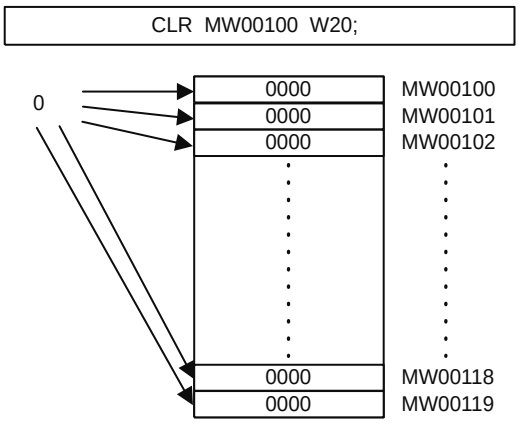
(3) 程序示例

以下是 CLR 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	CLR MW00100 W10;	
L	–	–
F	–	–

（例）将 MW00100 ～ MW00119 的内容清零。



8.8.5 ASCII 转换 1 （ASCII）

运动程序	顺序程序
○	○

(1) 动作说明

ASCII 转换 1 （ASCII）命令将指定的字符 （字符串）转换成 ASCII 代码，存放到指定的存放目标寄存器 （整数型寄存器）。字符区分大小写。

按第 1 个字符 （存放到第 1 个字的下位字节）、第 2 个字符 （存放到第 1 个字的上位字节）的顺序存放。字符串长度为奇数时，存放目标的最后字的上位字节变成 0。最多能够输入 32 个字符。

（注）使用 ASCII 命令需使用以下版本的编程工具。

MP2000 系列控制器	支持版本
全部机型	Ver2.60 或更高

MPE720	支持版本
MPE720 Ver.5	MPE720 Ver.5.38 或更高
MPE720 Ver.6	MPE720 Ver.6.04 或更高 MPE720 Ver.6.04Lite 或更高

(2) 格式

ASCII ‘ABCDEFG’ MW00200;
A B

内容	用途	可使用的寄存器
A	字符串	ASCII 字符型寄存器
B	存放目标寄存器编号	整数型寄存器 （#、C 寄存器除外）

ASCII 命令中能够使用 / 不能使用的字符如下所示。

(a) 能够使用的字符

项目	ASCII 字符
半角英文数字	a ～ z、A ～ Z、0 ～ 9
半角符号	空白字符， !#$%&( )*+,-./:;<=>?@[ ]¥]^_`{|}~

(b) 不能使用的字符

项目	ASCII 字符
单引号	’
双引号	“
双斜线	//
全角字符	所有的全角字符
半角假名字符	所有的半角假名字符

(3) 程序示例

以下是 ASCII 命令的程序示例。

例 (a) 将字符串 ABCD 存放到 MW00100 ～ MW00101 时

```
ASCII ‘ABCD’ MW00100;
```

	高位字节	低位字节	
MW00100	42H(B)	41H(A)	MW00100 = 4241H
MW00101	44H(D)	43H(C)	MW00101 = 4443H

例 (b) 将字符串 “ABCDEFG” 存放到 MW00100 ～ MW00103 时

```
ASCII ‘ABCDEFG’ MW00100;
```

	高位字节	低位字节	
MW00100	42H(B)	41H(A)	MW00100 = 4241H
MW00101	44H(D)	43H(C)	MW00101 = 4443H
MW00102	46H(F)	45H(E)	MW00102 = 4645H
MW00103	00H	47H(G)	MW00103 = 0047H

↑剩余的字节中填充 0。

8.9 基本函数

本节将对三角函数、平方根、BIN、BCD 等基本函数命令进行说明。

8.9.1 正弦 （SIN）

运动程序	顺序程序
○	○

(1) 动作说明

正弦 （SIN）命令可将整数型或实数型数据的正弦值返回至运算结果。倍长整数型数据无法使用。

(2) 格式

MW00100 ＝ SIN(3000) ;
A　　B

内容	用途	单位	可使用的寄存器
A	正弦值输出	—	・ 整数型、实数型的所有寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器
B	角度输入	度 (°)*1	・ 整数型、实数型的所有寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器 ・ 常数

*1. 整数型与实数型时的输入单位及输出结果不同。

- 整数型

  能够在 -327.68 ～ 327.67° 以内的范围使用。将前一个运算结果 （整数型数据）作为输入值，将其运算结果返回至整数型寄存器中 （输入单位 1=0.01° ）。输出结果为运算结果乘以 10000 的值。

- 实数型

  输入前一个运算结果 （实数型数据），将其正弦运算结果返回至实数型寄存器中 （单位 = 度）。

（例）

整数型数据		实数型数据
MW00102 = SIN (MW00100) ; (05000)　(03000)	相等 ⇒ 0.5=SIN30°	MF00102 = SIN (MF00100) ; (0.5)　(30.0)

重要

整数型时，如果输入了 -327.68 ～ 327.67° 范围以外的值，则无法得到正确结果。

(3) 程序示例

以下是 SIN 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	—	—
W	MW00102=SIN(MW00100) ;	SIN [WF]Src MW00100　[WF]Dest MW00102
L	—	—
F	DF00202=SIN(DF00200) ;	SIN [WF]Src DF00200　[WF]Dest DF00202

8.9.2 余弦 （COS）

运动程序	顺序程序
○	○

(1) 动作说明

余弦 （COS）命令可将整数型或实数型数据的余弦值返回至运算结果。
倍长整数型数据无法使用。

(2) 格式

MW00100 ＝ COS (3000) ;
A　　B

内容	用途	单位	可使用的寄存器
A	余弦值输出	—	· 整数型、实数型的所有寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	角度输入	度 (°)*1	· 整数型、实数型的所有寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数

*1. 整数型与实数型时的输入单位及输出结果不同。

- 整数型

  能够在 -327.68 ～ 327.67° 以内的范围使用。将前一个运算结果 （整数型数据）作为输入值，将其运算结果返回至整数型寄存器中 （输入单位 1=0.01° ）。输出结果为运算结果乘以 10000 的值。

- 实数型

  输入前一个运算结果 （实数型数据），将其余弦运算结果返回至实数型寄存器中 （单位 = 度）。

（例）

整数型数据	实数型数据
MW00102 = COS (MW00100) ;　相等 ⇒ 0.5=COS60° (05000)　(06000)	MF00102 = COS (MF00100) ; (0.5)　(60.0)

重要

整数型时，如果输入了 -327.68 ～ 327.67° 范围以外的值，则无法得到正确结果。

(3) 程序示例

以下是 COS 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	—	—
W	MW00102=COS(MW00100) ;	COS [WF]Src MW00100　[WF]Dest MW00102
L	—	—
F	DF00202=COS(DF00200) ;	COS [WF]Src DF00200　[WF]Dest DF00202

8.9.3 正切 （TAN）

运动程序	顺序程序
○	○

(1) 动作说明

正切 （TAN）命令可将指定的变量或常数 （单位 = 度）作为输入值，将其正切运算结果返回至实数型寄存器中。

(2) 格式

MW00100 ＝ TAN(1.0)；
A　　B

内容	用途	单位	可使用的寄存器
A	正切值输出	—	· 所有实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	角度输入	度 (°)*1	· 所有实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数

*1. （例）输入值 （θ=45.0° ）的正切：计算 TAN （θ）=1.0。

```
DF00102=TAN(DF00100);
(1.0)          (45.0)
```

重要

TAN 命令只能使用实数型数据。如果指定了位 / 整数 / 倍长整数，则编译时会发生错误。

(3) 程序示例

以下是 TAN 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	—	—
W	—	—
L	—	—
F	DF00202=TAN(DF00200)；	

8.9.4 反正弦 （ASN）

运动程序	顺序程序
○	○

(1) 动作说明

反正弦 （ASN）命令可将指定的变量或常数作为输入值，将其反正弦运算结果 （单位 = 度）返回至实数型寄存器中。

(2) 格式

MF00100 ＝ ASN(0.5)；
A　　B

内容	用途	单位	可使用的寄存器
A	角度输出	度（°）*1	· 所有实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	正弦值输入	—	· 所有实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数

*1. （例）输入值 （0.5）的反正弦：计算 ASN （0.5）=30.0° 。

```
MF00202=ASN(MF00200);
(30.0)          (0.5)
```

重要

ASN 命令只能使用实数型数据。如果指定了位 / 整数 / 倍长整数，则编译时会发生错误。

(3) 程序示例

以下是 ASN 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	—	—
W	—	—
L	—	—
F	DF00202=ASN(DF00200)；	

8.9.5 反余弦 （ACS）

运动程序	顺序程序
○	○

(1) 动作说明

反余弦 （ACS）命令可将指定的变量或常数作为输入值，将其反余弦运算结果 （单位 = 度）返回至实数型寄存器中。

(2) 格式

MF00100 ＝ ACS (0.5) ;
A　　B

内容	用途	单位	可使用的寄存器
A	角度输出	度 (°)*1	· 所有实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	余弦值输入	—	· 所有实数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数

*1. （例）输入值 （0.5）的反余弦：计算 ACS （0.5）=60.0° 。

```
MF00100 = ACS ( MF00102 ) ;
(60.0)          (0.5)
```

重要

ACS 命令只能使用实数型数据。如果指定了位 / 整数 / 倍长整数，则编译时会发生错误。

(3) 程序示例

以下是 ACS 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	—	—
W	—	—
L	—	—
F	DF00202=ACS (DF00200) ;	

8.9.6 反正切 （ATN）

运动程序	顺序程序
○	○

(1) 动作说明

反正切 （ATN）命令可将整数型或实数型数据的反正切值返回至运算结果。
倍长整数型数据无法使用。

(2) 格式

MW00100 ＝ ATN(100) ;
A　　B

内容	用途	单位	可使用的寄存器
A	角度输出	度（°）*1	・所有整数型、实数型寄存器 （#、C 寄存器除外） ・附加字符的上述寄存器 ・附加字符寄存器
B	正切值输入	—	・所有整数型、实数型寄存器 （#、C 寄存器除外） ・附加字符的上述寄存器 ・附加字符寄存器 ・常数

*1. 整数型与实数型时得输入单位及输出结果不同。

- 整数型

  能够在 -327.68 ～ 327.67 以内的范围使用。将前一个运算结果 （整数型数据）作为输入值，将其运算结果返回至整数型寄存器中 （输入 1=0.01，输出结果为运算结果乘以 100 的值）。

- 实数型

  输入前一个运算结果 （实数型数据），将其反正切运算结果返回至实数型寄存器中。

（例）

整数型数据		实数型数据
MW00100 = ATN (MW00102) ; (04500)　(00100)	相等 ⇒ 45=ATN(1.0)	MF00100 = ATN (MF00102) ; (45.0)　(1.0)

(3) 程序示例

以下是 ATN 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	—	—
W	MW00102=ATN(MW00100) ;	ATAN [WF]Src MW00100　[WF]Dest MW00102
L	—	—
F	DF00202=ATN(DF00200) ;	ATAN [WF]Src DF00200　[WF]Dest DF00202

8.9.7 平方根 （SQT）

运动程序	顺序程序
○	○

(1) 动作说明

平方根 （SQT）命令可将整数或实数的平方根返回至运算结果。
倍长整数型数据无法使用。

(2) 格式

MW00100 ＝ SQT (100) ;
A　　B

内容	用途	可使用的寄存器
A	平方根输出	・ 所有整数型、实数型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器
B	数据输入	・ 所有整数型、实数型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器 ・ 常数

（注）整数型与实数型时的输入单位及输出结果不同。

• 整数型数据

与数学中的平方根运算不同，主要是通过以下算式进行运算。

$$32768 * sign(B) * \sqrt{|B|/32768}$$

sign(B)：数据输入的符号
|B|　　：数据输入的绝对值

即为在数学中的平方根运算结果上乘以 $\sqrt{32768}$ 后得到的值。而且，输入为负数时，运算绝对值的平方根，将其负数作为运算结果。运算误差最大为 ±2。

• 实数型数据

输入前一个运算结果 （实数型数据），以实数型数据的形式保留其平方根。

（例）

	整数型数据	实数型数据
输入为正数时	MW00100 = SQT (MW00102) ;　$\sqrt{64} \times \sqrt{32768} = 1448$ (01448)　(00064)　(8)　(181)	MF00100 = SQT (MF00102) ; (8.0)　(64.0)
输入为负数时	MW00100 = SQT (MW00102) ;　$-\left(\sqrt{64} \times \sqrt{32768}\right) = -1448$ (-01448)　(-00064)　(8)　(181)	MF00100 = SQT (MF00102) ; (-8.0)　(-64.0)

(3) 程序示例

以下是 SQT 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	MW00102=SQT (MW00100) ;	SQRT [WF]Src　MW00100　[WF]Dest　MW00102
L	–	–
F	DF00202=SQT (DF00200) ;	SQRT [WF]Src　DF00200　[WF]Dest　DF00202

8.9.8 BCD → BIN （BIN）

运动程序	顺序程序
○	○

(1) 动作说明

BCD → BIN （BIN）命令可将 BCD 代码的数据转换成二进制 （BIN 代码）。
只能使用整数数据。如果发出了非 BCD 代码数值的指令，则无法得到正确结果。

(2) 格式

MW00100 = BIN(1234H);
A　　B

内容	用途	可使用的寄存器
A	BIN 输出	・ 所有整数型、倍长整数型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器
B	BCD 输入	・ 所有整数型、倍长整数型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器 ・ 附加字符寄存器 ・ 常数

（注）（例 1）

（例 2）

补充　如果发出了非 BCD 代码数据的指令，则无法得到正确结果。

(3) 程序示例

以下是 BIN 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	MW00101=BIN(MW00100);	
L	ML00102=BIN(ML00100);	
F	–	–

8.9.9 BIN→BCD （BCD）

运动程序	顺序程序
○	○

(1) 动作说明

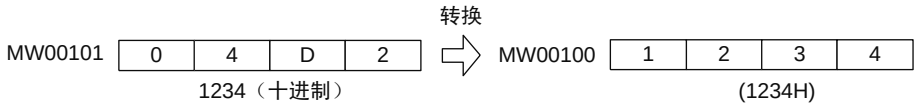
BIN→BCD （BCD）命令可将二进制 （BIN 代码）的数据转换成 BCD 代码。
只能使用整数数据。BIN 数据为 9999 以上或负值时，无法得到正确结果。

(2) 格式

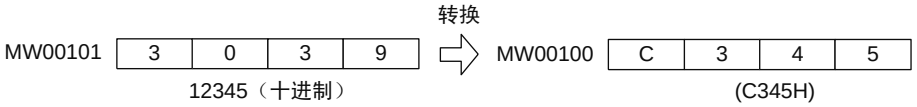
MW00100 = BCD(1234);
A　　B

内容	用途	可使用的寄存器
A	BCD 输出	· 所有整数型、倍长整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器
B	BIN 输入	· 所有整数型、倍长整数型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 附加字符寄存器 · 常数

（注）（例 1）



（例 2）

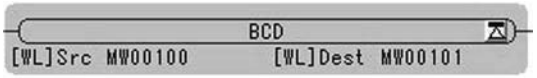
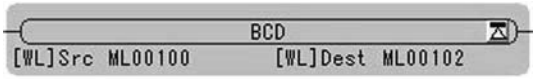


补充　BIN 数据大于 9999 时，无法得到正确结果。

(3) 程序示例

以下是 BCD 命令的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	–	–
W	MW00101=BCD(MW00100);	
L	ML00102=BCD(ML00100);	
F	–	–

8.9.10 指定位 ON （S{ }）

运动程序	顺序程序
○	○

(1) 动作说明

指定位 ON （S{ }）在逻辑运算中的结果为 “ 真 ” 时，接通指定位。即使逻辑运算结果为 “ 假 ”，也不会关闭指定位。

(2) 格式

S {MB001000} ＝ MB001010 & MB001011;
A B

内容	用途	可使用的寄存器
A	指定位	· 所有位型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器
B	逻辑运算式	· 所有位型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 常数

(3) 程序示例

以下是指定位 ON （S{ }）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	S{MB001000}=MB001010&MB001011;	
W	–	–
L	–	–
F	–	–

8.9.11 指定位 OFF （R{ }）

运动程序	顺序程序
○	○

(1) 动作说明

指定位 OFF （R{ }）在逻辑运算中的结果为 “ 真 ” 时，关闭指定位。即使逻辑运算结果为 “ 假 ”，也不会启用指定位。

(2) 格式

```
R {MB001000} = MB001010 & MB001011;
   A             B
```

内容	用途	可使用的寄存器
A	指定位	· 所有位型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器
B	逻辑运算式	· 所有位型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器 · 常数

(3) 程序示例

以下是指定位 OFF （R{ }）的程序示例。

例

type	运动程序 / 顺序程序	梯形图程序
B	R{MB001000}=MB001010&MB001011;	
W	–	–
L	–	–
F	–	–

8.9.12 上升脉冲 （PON）

运动程序	顺序程序
×	○

(1) 动作说明

上升脉冲 （PON）在位输入的状态从 OFF 变化为 ON 时，位输出在 1 次扫描期间保持 ON。用来保存前一次位输出值的寄存器被用作 PON 处理的任务。请设定其它处理中不会使用的寄存器。

（注） 使用 PON 命令需使用以下版本的编程工具。

MP2000 系列控制器	支持版本
全部机型	Ver2.60 或更高

MPE720	支持版本
MPE720 Ver.5	MPE720 Ver.5.38 或更高
MPE720 Ver.6	MPE720 Ver.6.04 或更高 MPE720 Ver.6.04Lite 或更高

(2) 格式

DB000002 = PON(DB000000 DB000001);
A B C

内容	用途	可使用的寄存器
A	位输出	・ 位型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器
B	位输入	・ 位型的所有寄存器 ・ 附加字符的上述寄存器
C	用于保存位输出的上次值	・ 位型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器

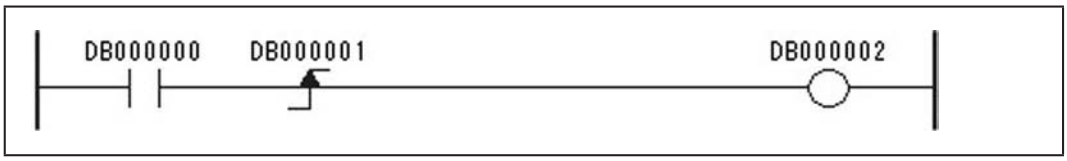
(3) 程序示例

以下是 PON 命令的程序示例。

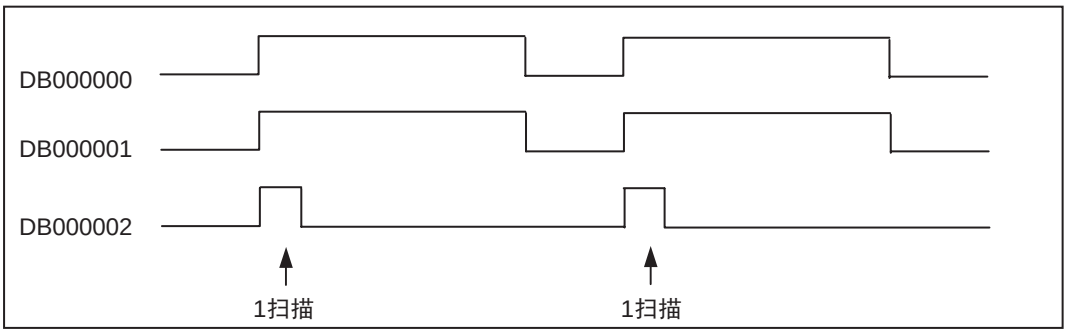
例 (a) 输出到线圈时

```
DB000002=PON(DB000000 DB000001);
```

・ 梯形图等价电路



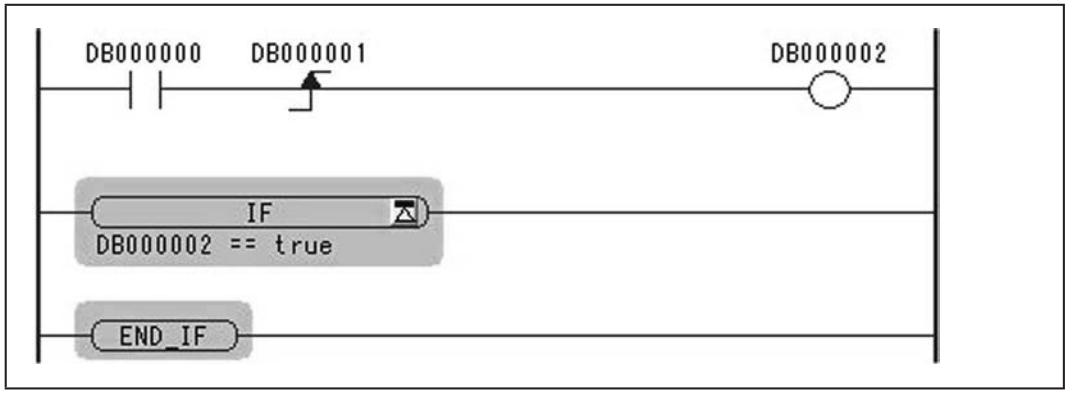
・ 时序图



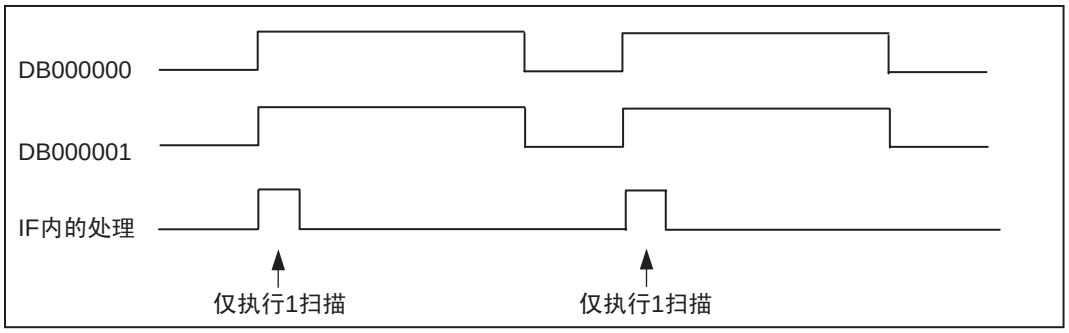
例 (b) 与 IF 命令组合使用时

```
IF PON(DB000000 DB000001) == 1;
        .
        .
IEND;
```

· 梯形图等价电路



· 时序图



8.9.13 下降脉冲 （NON）

运动程序	顺序程序
×	○

(1) 动作说明

下降脉冲 （NON）在位输入的状态从 ON 变化为 OFF 时，位输出在 1 次扫描期间保持 ON。用来保存前一次位输出值得寄存器被用作 NON 处理的任务。请设定其它处理中不会使用的寄存器。

（注）使用 NON 命令需使用以下版本的编程工具。

MP2000 系列控制器	支持版本
全部机型	Ver.2.60 或更高

MPE720	支持版本
MPE720 Ver.5	MPE720 Ver.5.38 或更高
MPE720 Ver.6	MPE720 Ver.6.04 或更高 MPE720 Ver.6.04Lite 或更高

(2) 格式

DB000002 = NON(DB000000 DB000001);
A B C

内容	用途	可使用的寄存器
A	位输出	・ 位型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器
B	位输入	・ 位型的所有寄存器 ・ 附加字符的上述寄存器
C	用于保存位输出的上次值	・ 位型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器

(3) 程序示例

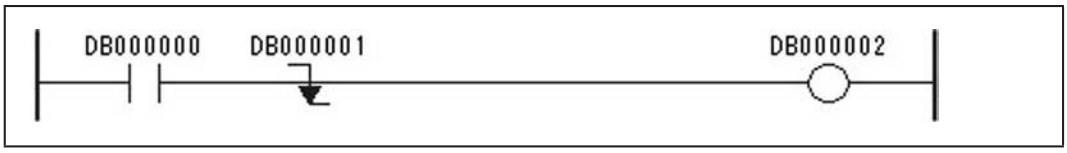
以下是 NON 命令的程序示例。

例

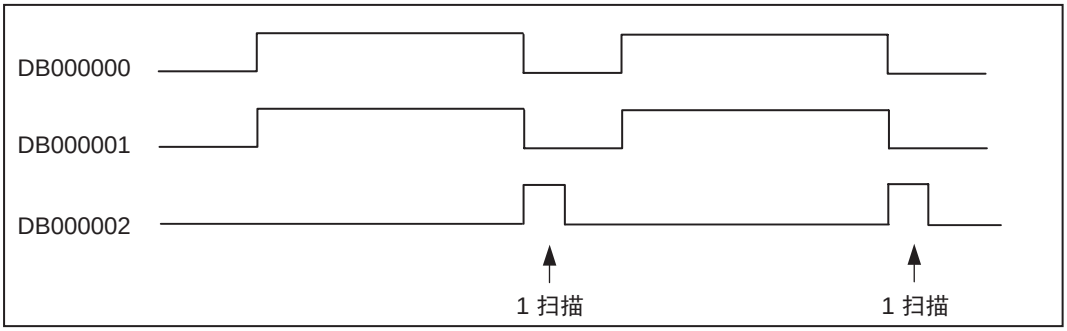
(a) 输出到线圈时

```
DB000002=NON(DB000000 DB000001);
```

・ 梯形图等价电路



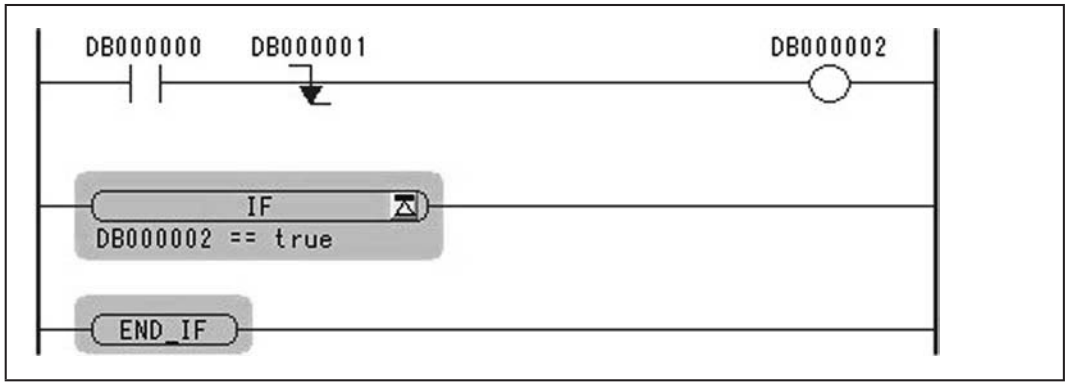
・ 时序图



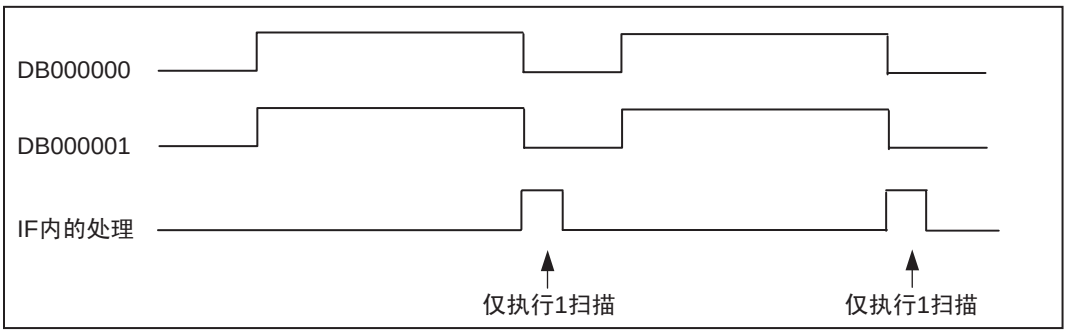
例 (b) 与 IF 命令组合使用时

```
IF NON(DB000000  DB000001) == 1;
                .
                .
IEND;
```

· 梯形图等价电路



· 时序图



8.9.14 接通延时定时器：测量单位＝ 0.01 秒 （TON）

运动程序	顺序程序
×	○

(1) 动作说明

接通延时定时器： 测量单位＝ 0.01 秒 （TON）在位输入为 ON 时，执行时间计数。当 “ 计数值 = 设定值 ” 时，位输出变为 ON。
在计数过程中如果位输入变成 OFF，则定时器停止工作。位输入再次变成 ON 时，计数从头 （0）开始。而且，计数用寄存器中会存放实际的计数时间 （10 ms 单位）。

（注）使用 TON 命令需使用以下版本的编程工具。

MP2000 系列控制器	支持版本
全部机型	Ver2.60 或更高

MPE720	支持版本
MPE720 Ver.5	MPE720 Ver.5.38 或更高
MPE720 Ver.6	MPE720 Ver.6.04 或更高 MPE720 Ver.6.04Lite 或更高

(2) 格式

DB000001 ＝ DB000000 & TON(500 DW00001);
A B C D

内容	用途	可使用的寄存器
A	位输出	・ 位型寄存器 （#、C 寄存器除外） ・ 附加字符的上述寄存器
B	位输入	・ 位型的所有寄存器 ・ 附加字符的上述寄存器
C	设定值	・ 整数型的所有寄存器 ・ 附加字符的上述寄存器 ・ 常数 （0 ～ 65535(655.35S):10 ms 刻度）
D	定时器计数用寄存器	・ 整数型的所有寄存器 ・ 附加字符的上述寄存器

重要

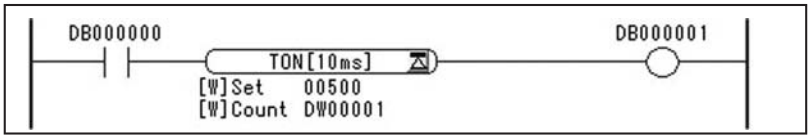
- 调试运行停止中，不执行时间计数。
  解除调试运行停止后，从当前计数值再次开始计数。
- 请务必指定位输入 “DBxxxxxx &”。

(3) 程序示例

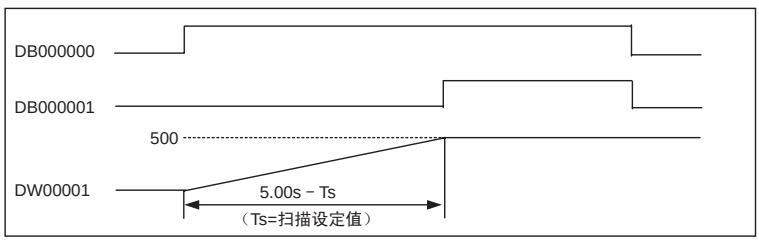
例 以下是 TON 命令的程序示例。

```
DB000001=DB000000 & TON (500 DW00001);
                        ↑ 5 秒设定
```

・ 梯形图等价电路



・ 时序图



8.9.15 断开延时定时器：测量单位＝ 0.01 秒 （TOF）

运动程序	顺序程序
×	○

(1) 动作说明

断开延时定时器： 测量单位＝ 0.01 秒 （TOF）在位输入为 OFF 时，执行时间计数。当 “ 计数值 = 设定值 ” 时，位输出变为 OFF。
在计数过程中如果位输入变成 ON，则定时器停止工作。位输入再次变成 OFF 时，计数从头 （0）开始。而且，计数用寄存器中会存放实际的计数时间 （10 ms 单位）。

（注）使用 TOF 命令需使用以下版本的编程工具。

MP2000 系列控制器	支持版本
全部机型	Ver2.60 或更高

MPE720	支持版本
MPE720 Ver.5	MPE720 Ver.5.38 或更高
MPE720 Ver.6	MPE720 Ver.6.04 或更高 MPE720 Ver.6.04Lite 或更高

(2) 格式

```
DB000001 = DB000000 & TOF(500 DW00001);
   A         B          C     D
```

内容	用途	可使用的寄存器
A	位输出	· 位型寄存器 （#、C 寄存器除外） · 附加字符的上述寄存器
B	位输入	· 位型的所有寄存器 · 附加字符的上述寄存器
C	设定值	· 整数型的所有寄存器 · 附加字符的上述寄存器 · 常数 （0 ～ 65535 （655.35S）：10 ms 刻度）
D	定时器计数用寄存器	· 整数型的所有寄存器 · 附加字符的上述寄存器

重要

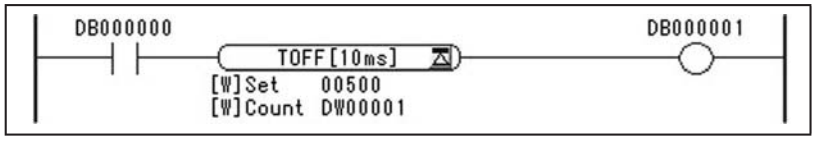
- 调试运行停止中，不执行时间计数。
  解除调试运行停止后，从当前计数值再次开始计数。
- 请务必指定位输入 “DBxxxxxx &”。

(3) 程序示例

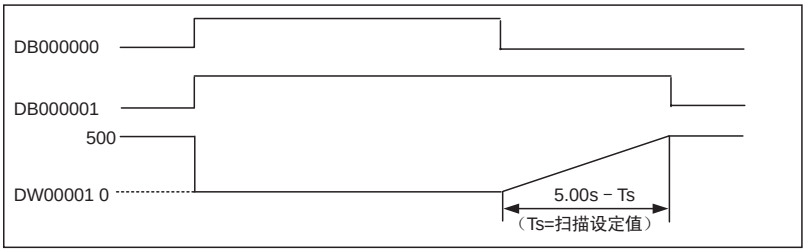
例 以下是 TOF 命令的程序示例。

```
DB000001=DB000000 & TOF (500 DW00001);
```

· 梯形图等价电路



· 时序图



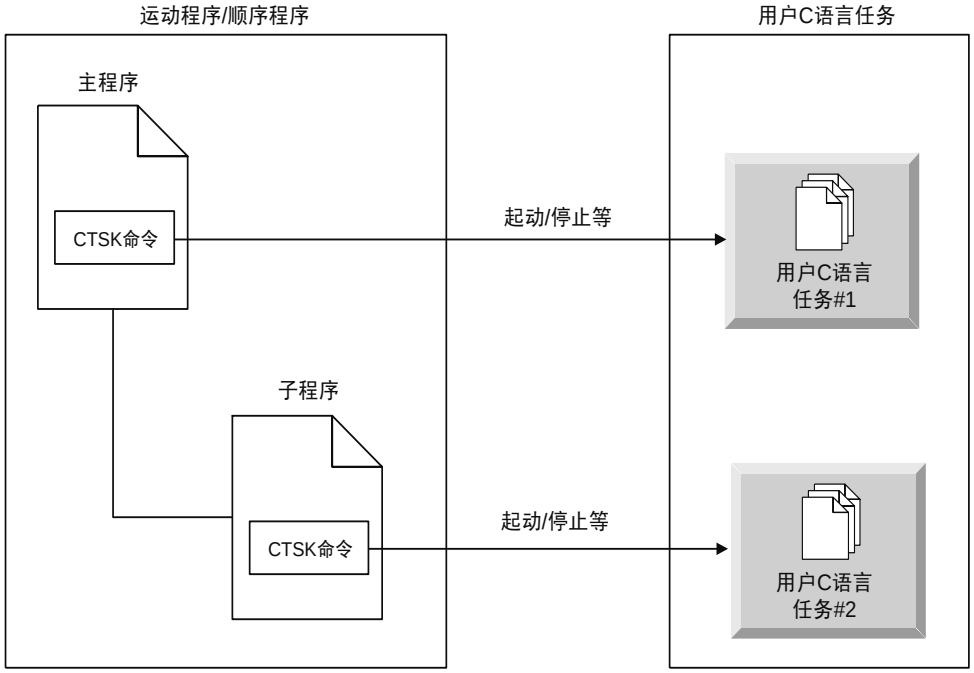
8.10 C 语言控制命令

8.10.1 C 语言任务控制 （CTSK）

运动程序	顺序程序
○	○

(1) 动作说明

C 语言任务控制 （CTSK）是执行用户 C 语言任务启动及停止等控制的命令。



（注）使用 CTSK 命令需使用以下版本的编程工具。

MP2000 系列控制器	支持版本
全部机型	Ver2.60 或更高

MPE720	支持版本
MPE720 Ver.5	MPE720 Ver.5.38 或更高
MPE720 Ver.6	MPE720 Ver.6.04 或更高 MPE720 Ver.6.04Lite 或更高

补充

使用用户 C 语言任务，需使用 “ 机器控制器 MP2000 系列嵌入 C 语言包 ”。
详细内容，请参阅 《机器控制器 MP2000 系列嵌入 C 语言包开发指南》（资料编号：SIJP C880700 25）（日文）。

(2) 格式

```
CTSK EXECUTE TYPE,C_NAME,COMPLETE ERROR ERR_CODE;
```

输入输出定义	No.	名称	输入输出指定	内容	
输入	1	EXECUTE	B-VAL	“CTSK” 函数的执行指定	
	2	TYPE	I-REG	任务控制的类型指定	
				1: WAKEUP	唤醒处于等待 （WAIT）状态的任务。
				2: RESET	仅顺序程序 （L 扫描）有效。 先退出 / 删除任务，再重新生成 / 启动。 任务运行后变成等待 （WAIT）状态。
				3: SUSPEND	中断任务，使其进入 “ 强制等待 （SUSPEND）状态 ”。
				4: RESUME	将处于 “ 强制等待 （SUSPEND）状态 ” 的任务迁移到 “ 可执行 （READY）状态 ”。
	3	C_NAME	地址输入	指定起始寄存器的编号 （MW、DW 的地址）。该寄存器中存放有用户 C 语言任务的名称 （项目名称）。	
输出	1	COMPLETE	B-VAL	“CTSK” 函数的执行完毕	
	2	ERROR	B-VAL	发生错误 （错误内容报告为 ERR_CODE）	
	3	ERR_CODE	L-REG	错误代码	
				0x00000000	无错误
				0x0000006F	DWG 类别错误 • 通过顺序程序 （开始）执行 “CTSK” 函数
				0x00000091	TYPE 的设定异常 • TYPE 的值超出范围 • 执行 TYPE=WAKEUP 时，非 WAIT（等待状态），WAIT-SUSPEND（双重等待状态） • 执行 TYPE=SUSPEND 时，非 WAIT（等待状态），READY（可执行状态）
				0x00000094	不存在 C_NAME 寄存器中指定的任务。
				0x00000096	C_NAME 的寄存器上下限溢出 ※EXECUTE OFF 时，也进行检测
				0xFFFFFFDD	μITRON 检测错误 （非法 ID 编号）*1
				0xFFFFFFCC	μITRON 检测错误 （任务未登录）*1
				0xFFFFFFC1	μITRON 检测错误 （项目状态错误）*1 • 任务处于 DORMANT （休止）状态。 • 任务未处于 SUSPEND 状态时，发出了 RESUME 的指令。
				0xFFFFFFBB	μITRON 检测的错误 （上下文错误）*1 • 无法从任务独立部分开始
				0xFFFFFFB7	μITRON 检测的错误 （队列溢出）*1

*1. μITRON 检测通常不会因系统侧的控制而出错。

（注）1. EXECUTE 输入没有用于信号上升沿，而是作为信号的电平处理，这是为了实现每次扫描时的任务控制。
2. RESET 以外的任务控制可通过运动程序、顺序程序的高速扫描或低速扫描指令执行。不能通过顺序程序的起动指令执行。
3. RESET 只能通过顺序程序 （L 扫描）执行。

(3) 程序示例

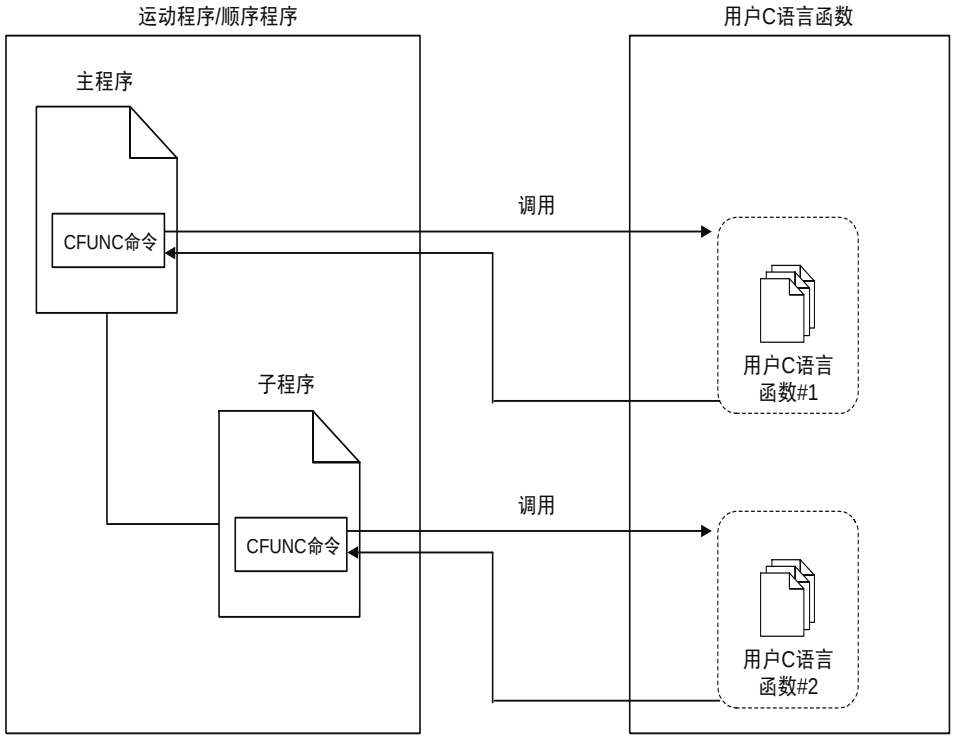
例 以下是 CTSK 命令的程序示例。

```
ASCII ‘ctask1’  DW00010; “ 用户 C 语言任务的任务名称
DW00013 = 0000H;          “NULL 代码
DW00002 = 1;              “WAKEUP
DB000001 = 1;             “ 执行任务控制
CTSK DB000001 DW00002, DA00010, DB000002 DB000003 DL00004;
```

8.10.2 C语言函数调用 （CFUNC）

(1) 动作说明

C语言函数调用 （CFUNC）是执行用户C语言任务调用的命令。



（注）使用CFUNC命令需使用以下版本的编程工具。

MP2000系列控制器	支持版本
全部机型	Ver2.60或更高

MPE720	支持版本
MPE720 Ver.5	MPE720 Ver.5.38 或更高
MPE720 Ver.6	MPE720 Ver.6.04或更高 MPE720 Ver.6.04Lite 或更高

补充

使用用户C语言任务，需使用 “ 机器控制器MP2000系列嵌入C语言包 ”。
详细内容，请参阅 《机器控制器MP2000系列嵌入C语言包开发指南》（资料编号：SIJP C880700 25）（日文）。

(2) 格式

```
CFUNC EXECUTE OPTION1 OPTION2,C_NAME C_ARG1 C_ARG2,
                              COMPLETE ERROR C_RETURN;
```

输入输出定义	No.	名称	输入输出指定	内容
输入	1	EXECUTE	B-VAL	“CFUNC” 函数的执行指定
	2	OPTION1	I-REG	选项指定 1 （备用）
	3	OPTION2	I-REG	选项指定 2 （备用）
	4	C_NAME	地址输入	指定存放用户 C 语言函数的名称 （项目名称）的起始寄存器的编号 （MW、DW 的地址）
	5	C_ARG.1	地址输入	指定调用到用户 C 语言函数的第 1 自变量的起始寄存器的编号（MW、DW 的地址）
	6	C_ARG.2	地址输入	指定调用到用户 C 语言函数的第 2 自变量的起始寄存器的编号（MW、DW 的地址）
输出	1	COMPLETE	B-VAL	“CFUNC” 函数的执行完毕
	2	ERROR	B-VAL	发生错误 • C_NAME，C_ARG.1，C_ARG.2 的寄存器上下限溢出（但，C_ARG.1，C_ARG.2 不考虑大小。） ※EXECUTE OFF 时，也进行检测 • 不存在 C_NAME 寄存器中指定的函数
	3	C_RETURN	L-REG	直接存放用户 C 语言函数的返回值

(3) 程序示例

例 以下是 CFUNC 命令的程序示例。

```
ASCII ‘cfunc1’ DW00010; “用户 C 语言函数的函数名称
DW00013 = 0000H;        “NULL 代码
DB000000 = 1;           “执行用户 C 语言函数
CFUNC DB000000 0 0,DA00010 DA00002 DA00004, DB000003 DB000004 DL00016;
```

9 章

软件工具 （MPE720）

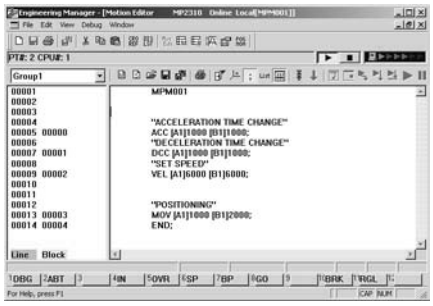
本章对运动程序及顺序程序的软件工具 MPE720 进行说明。

9.1 运动编辑器

本节对运动编辑器进行说明。

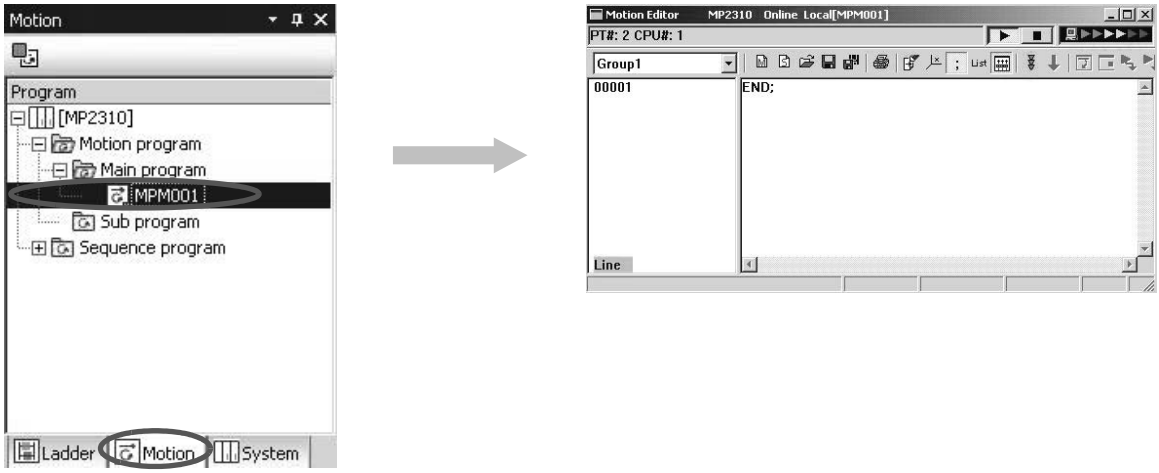
9.1.1 运动编辑器的概要

运动编辑器是输入 / 编辑运动程序和顺序程序时所需的编程工具。具有从程序的文本编辑到编译 （保存）、调试、监视等功能。

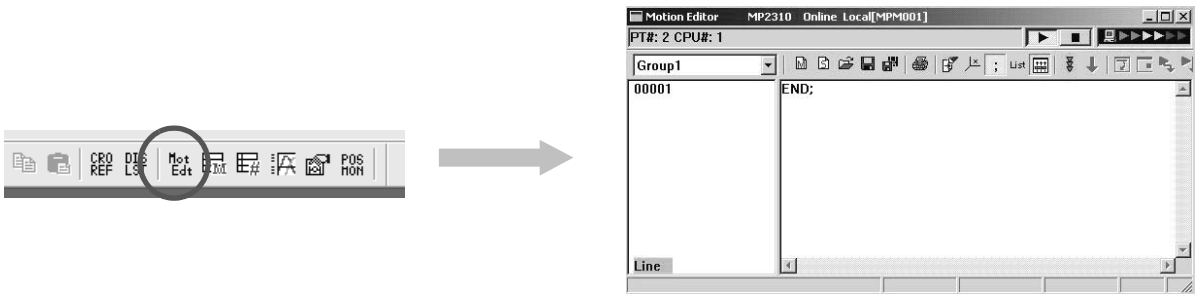


运动编辑器的启动方法有以下 2 种。

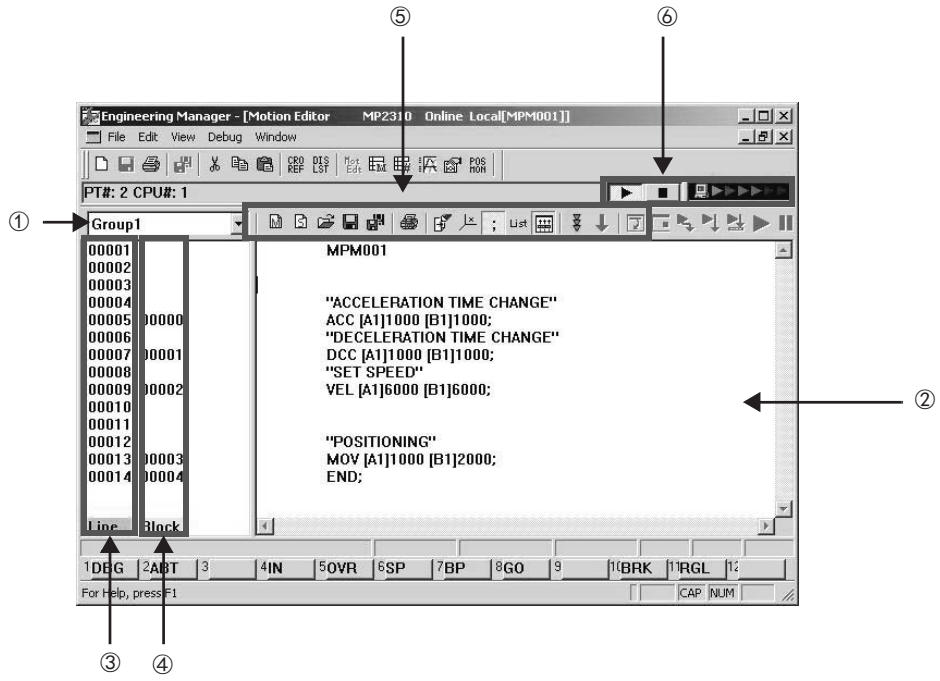
■ 在 ***Motion*** 子窗口中双击对象程序



■ 在 ***Engineering Manager***（工程管理器）中点击 Mot Edt 图标

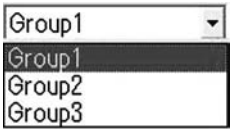


9.1.2 画面说明



①选择分组 （仅限运动程序）

通过分组定义设定的组名称显示在下拉菜单中。
选择正在用于编辑程序的分组。



②程序编辑窗口

是输入程序的文本编辑器。

③行

显示程序的文本行。

④程序段

显示程序段。
运动程序发生运动程序警报时，会报告该程序段。

⑤工具图标

是用于程序编辑的图标。

功能	图标	键操作	说明
剪切		Ctrl+X	剪切 （移动到剪切板）选择范围。
复制		Ctrl+C	复制 （移动到剪切板）选择范围。
粘贴		Ctrl+V	粘贴剪切板中的内容。
位置监视器	POS MON	-	显示 ***Position Monitor*** 对话框。
新建文件 （运动程序）		-	打开新建运动程序。
新建文件 （顺序程序）		-	打开新建顺序程序。
打开文件		-	显示 ***File List*** 对话框。
保存		Ctrl+S	将正在编辑的程序保存到电脑硬盘。 在线编辑过程中，保存到电脑硬盘和下载 （传输）到控制器同时进行。
保存＆闪存保存		-	可连续进行以下操作 • 将正在编辑的程序保存至电脑硬盘。 • 将正在编辑的程序下载至机器控制器。 • 将已下载的程序保存至闪存。 所有已下载至机器控制器的程序将保存至闪存。
程序打印		Ctrl+P	打印正在编辑的程序。
命令输入辅助		F12	显示 ***Motion Command Assist*** 对话框。
位置示教		-	显示 ***Position Teach*** 对话框。 将通过 ***Position Teach*** 对话框指定的轴任务坐标系的当前位置插入到运动编辑器。
是否附加分号切换	;	-	在程序编辑过程中选择 ENT 键输入时是否进行附加分号 （;）的切换。
错误列表对话框显示	List	-	显示保存错误 （编译错误）列表。
自动滚动		-	选择程序运行过程中是否进行与程序的运行行同步的画面滚动。

⑥监视器图标

是用于程序监视器的图标。

功能	图标	键操作	说明
监视器设定		-	设定是否监视程序。 选择 时，在画面中实时显示程序的运行行。
监视显示		-	显示程序的监视状态。

9.2 命令输入辅助

本节对命令输入辅助进行说明。

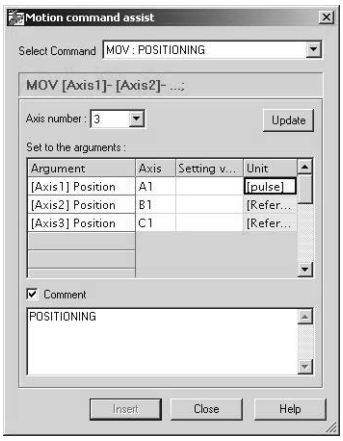
9.2.1 命令输入辅助的概要

命令输入辅助在创建运动程序时，支持输入所需运动命令。
运动命令是称为运动语言的文本格式语言，需按规定格式输入程序。使用 ***Motion command assist***（命令输入辅助）对话框，就能够非常简单地选择输入指定的命令。
命令输入辅助功能适用于以下版本的 MPE720 软件工具。

MPE720	支持版本
MPE720 Ver.5	不支持。
MPE720 Ver.6	Ver6.04 或更高 Ver6.04 Lite 或更高

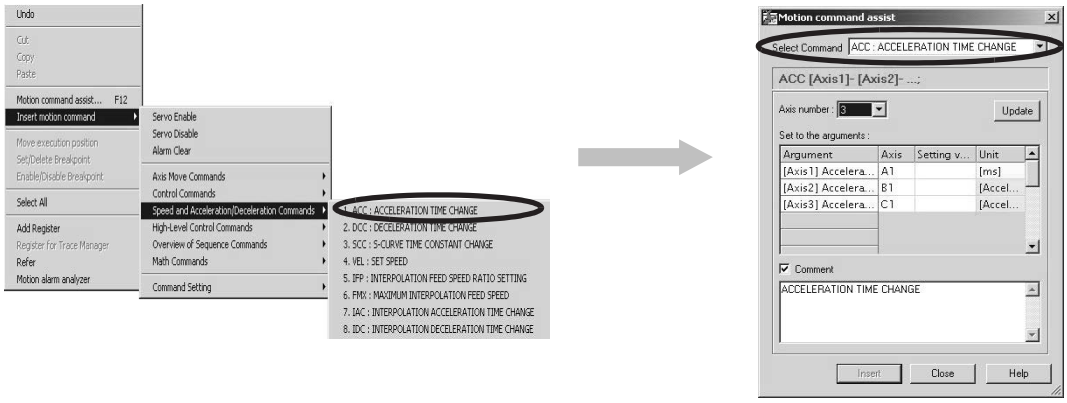
（注）命令辅助功能在 MP2000 系列的全部机型中都能够使用。

命令输入辅助通过 ***Motion Editor***（运动编辑器）画面启动。下页将对命令输入辅助的 2 种起动方法进行介绍。

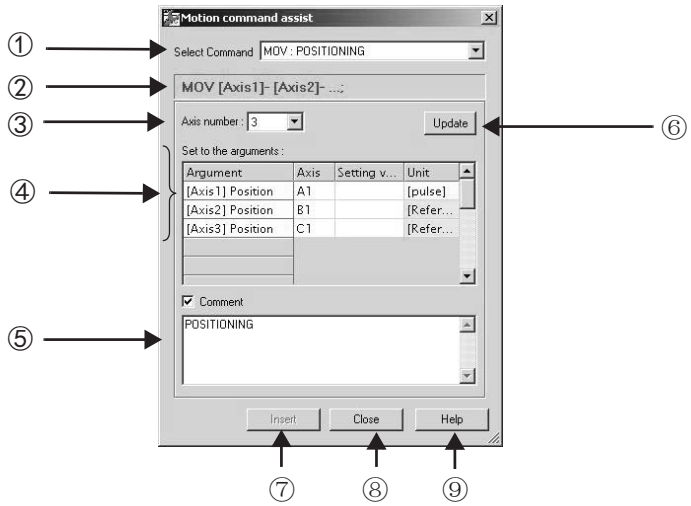


■在右击菜单中选择 ***Motion command assist***（命令输入辅助）或按功能键 F12

■在右击菜单中选择 ***Insert motion command*** （插入命令）→ （要插入的命令）

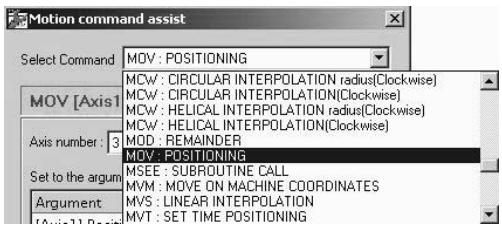


9.2.2 画面说明



①命令选择 （Select Command）

下拉菜单中显示可插入的命令一览。



②命令输入格式

显示选中命令的格式。

例

MOV：定位

```
MOV [Axis1]- [Axis2]- ...;
```

+：加法

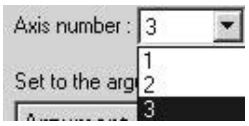
```
ML00106=ML00102 + ML00104;
```

③控制轴数 （Axis number）

轴移动类命令可选择控制轴数 （1 轴 ～ 使用群组定义设定的轴数）。控制轴数固定时，显示固定值并呈灰色 （禁用）。

例

MOV：定位指定控制轴数



EXM：外部定位控制轴数 固定

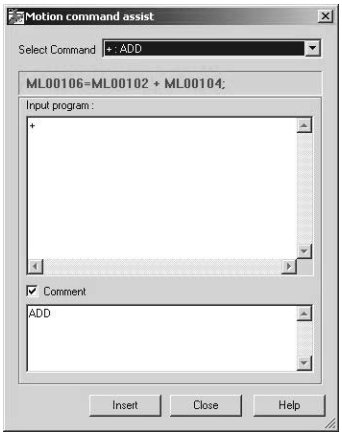
④参数设定 （Set to the arguments）

设定命令的参数。设定项目如下。

项目	内容
自变量 （Argument）	显示设定为自变量的参数名称。无法更改。 自变量可省略时，显示 ***Can be omitted***。
逻辑轴 （Axis）	显示逻辑轴名称。根据需要更改逻辑轴。
设定值 （Setting value）	将常数或寄存器输入为设定值。
单位 （Unit）	显示参数的单位。无法更改。

“ 逻辑轴 ” 的名称通过 ***Group Definition*** 进行定义。

根据各轴运动参数的定义内容显示 “ 单位 ”。未执行运动参数定义时，“ 单位 ” 显示为黄色。放上光标时会显示汽球状提示。请按照指示定义运动参数。如果命令中不需要设定控制轴数和参数，则会如下所示显示程序的输入栏。输入关于上面显示的格式的命令。



⑤注释 / 注释栏

勾选 ***Comment*** （插入注释），则可以在命令上面的行中添加注释。取消勾选，则注释栏呈灰色显示，无法插入注释。

补充 注释插入位置无法更改。

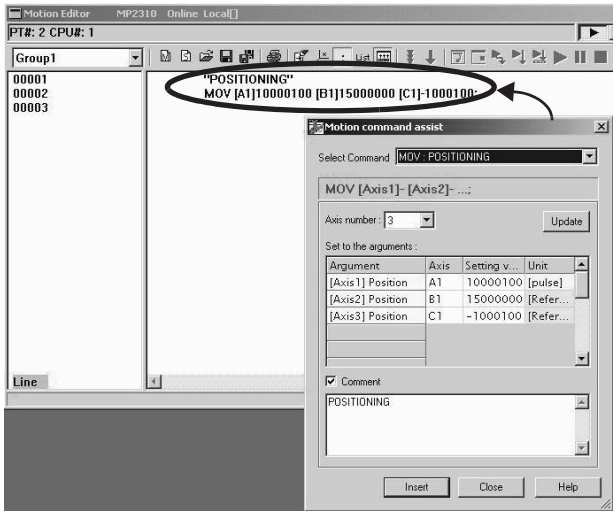
⑥ ***Update*** 按钮

更新 ***Motion command assist*** 对话框显示内容。

补充 更改与单位关联的运动参数后，请通过点击 ***Update*** 按钮更新显示。

⑦ ***Insert*** 按钮

将通过 ***Motion command assist*** 对话框编辑的命令插入 ***Motion Editor*** 窗口中的光标位置。

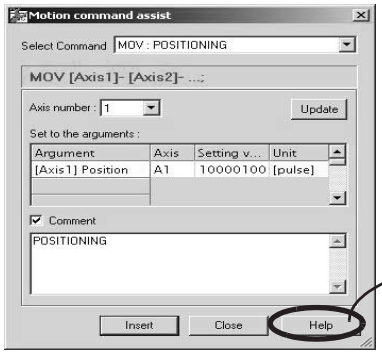


⑧ ***close*** 按钮

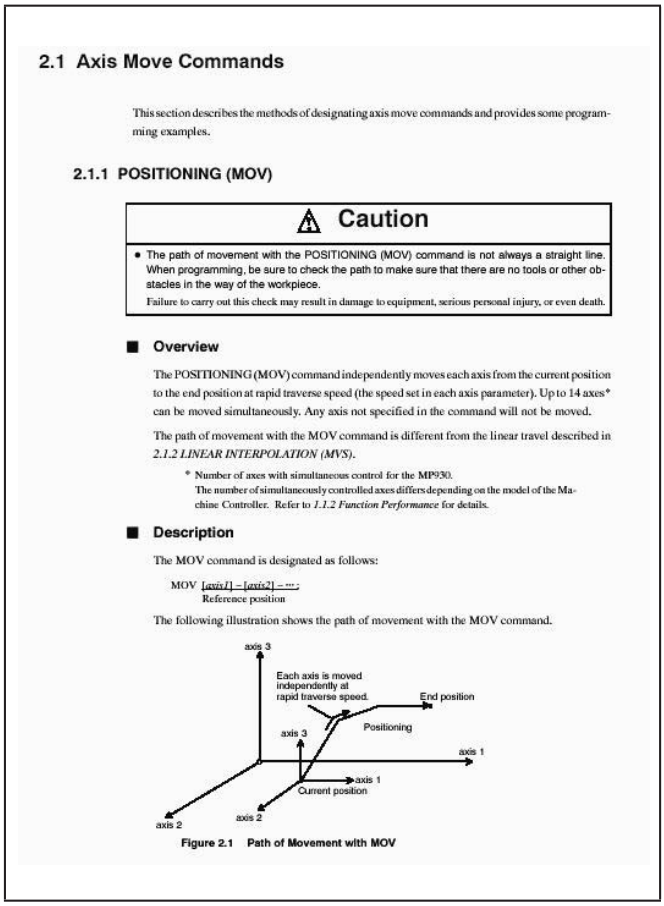
关闭 ***Motion command assist*** 对话框。

⑨ ***Help*** 按钮

显示相应命令的使用手册。



选择 ***Help*** 命令完成跳跃。



2.1 Axis Move Commands

This section describes the methods of designating axis move commands and provides some programming examples.

2.1.1 POSITIONING (MOV)

⚠ Caution

- The path of movement with the POSITIONING (MOV) command is not always a straight line. When programming, be sure to check the path to make sure that there are no tools or other obstacles in the way of the workpiece.

Failure to carry out this check may result in damage to equipment, serious personal injury, or even death.

■ Overview

The POSITIONING (MOV) command independently moves each axis from the current position to the end position at rapid traverse speed (the speed set in each axis parameter). Up to 14 axes* can be moved simultaneously. Any axis not specified in the command will not be moved.

The path of movement with the MOV command is different from the linear travel described in *2.1.2 LINEAR INTERPOLATION (MVS)*.

* Number of axes with simultaneous control for the MP930.
The number of simultaneously controlled axes differs depending on the model of the Machine Controller. Refer to *1.1.2 Function Performance* for details.

■ Description

The MOV command is designated as follows:

MOV [axis1] – [axis2] – ... ;
Reference position

The following illustration shows the path of movement with the MOV command.

Figure 2.1 Path of Movement with MOV

9.3 程序登录功能

本节对 M-EXECUTOR 的程序登录功能进行说明。

9.3.1 程序登录功能的概要

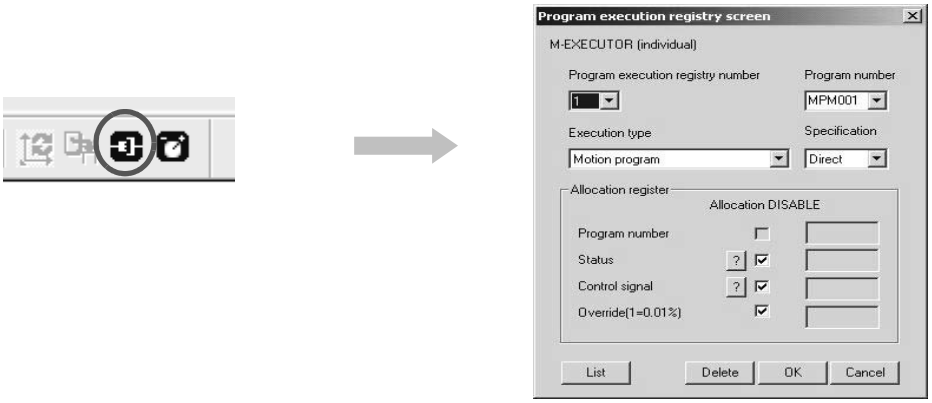
创建的运动程序及顺序程序时，需登录到 MP2000 系统。使用 ***Program execution registry screen*** 对话框能够很简单地将创建的运动程序及顺序程序登录到 MP2000 系统。

程序登录功能可以在以下版本的 MPE720 中使用。

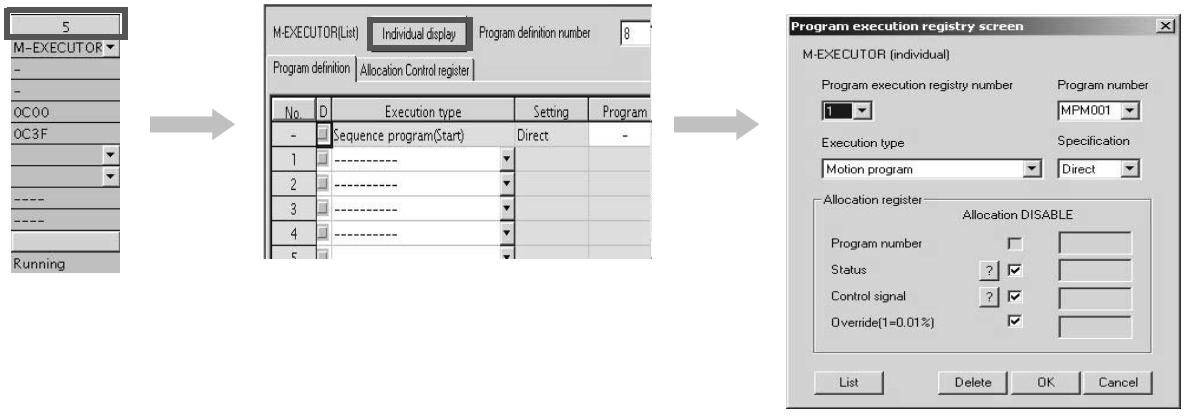
MPE720	支持版本
MPE720 Ver.5	Ver5.38 或更高
MPE720 Ver.6	Ver6.04 或更高 Ver6.04 Lite 或更高

Program execution registry screen （程序登录画面）的启动方法有以下 2 种。

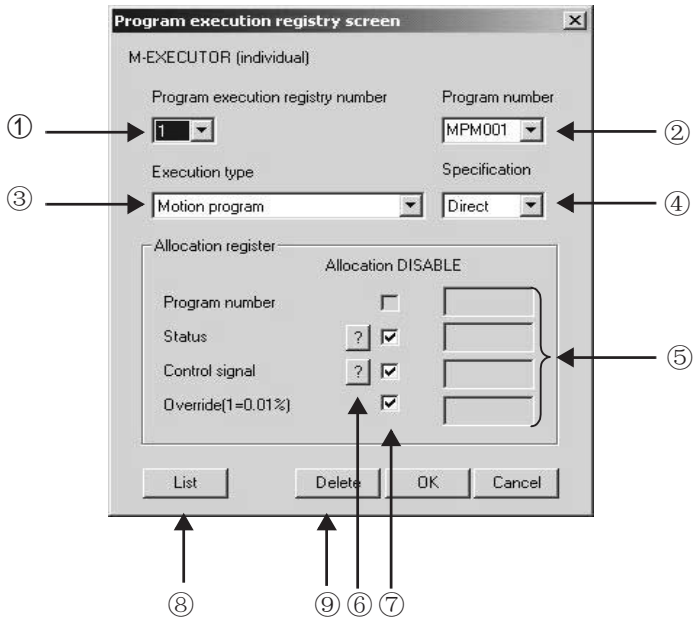
■在 ***Motion Editor*** 窗口中点击 图标。



■通过在 ***Module Configuration Definition*** 窗口中打开 M-EXECUTOR 的详细画面，点击 Individual display 图标。



9.3.2 画面说明



①程序登录编号 （Program execution registry number）

选择程序登录编号。
程序登录编号按照从小到大的顺序依次进行处理。

②程序编号 （Program number）

设定程序编号。

③运行类型 （Execution type）

设定程序的运行类型。

运行类型	所要运行的程序	运行条件
顺序程序 （启动）	顺序程序	接通电源 （仅在接通电源时运行一次）
顺序程序 （L 扫描）		循环启动 （每次低速扫描时运行）
顺序程序 （H 扫描）		循环启动 （每次高速扫描时运行）
运动程序	运动程序	开启控制信号的程序运行开始请求 （通过程序运行开始请求 ON 来运行）

④指定方法 （Specification）

设定程序指定方法。程序指定方法会因程序而异。

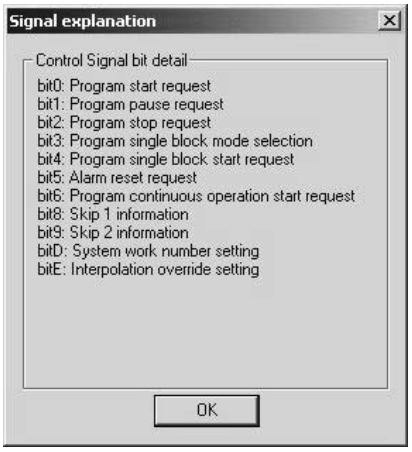
指定方法	运动程序	顺序程序	备注
直接指定	可以	可以	指定程序编号的方法 例：MPM001、SPM002 等
间接指定	可以	不可	指定用于存放程序编号的寄存器的方法 例：OW0C0C 等 （在 OW0C0C 中放入 1 后，执行 MPM001）

⑤分配寄存器 （Allocation register）

设定分配寄存器。分配寄存器和 M-EXECUTOR 控制寄存器会进行实时数据交换。在分配寄存器中可以对 I 寄存器、O 寄存器及 M 寄存器中的任意一个寄存器进行设定。

⑥状态、控制寄存器详细 （“ ? ” 图标）

显示状态、控制寄存器的信号排列。



⑦分配有效 / 无效

设定分配寄存器的有效 / 无效。如不勾选，则定义将变为有效。

⑧ ***List*** 按钮

显示 M-EXECUTOR （一览显示）画面。

⑨ ***Delete*** 按钮

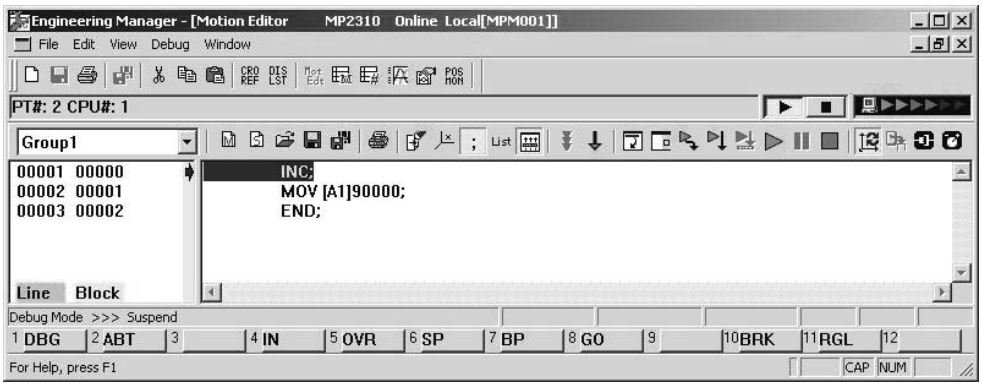
删除定义。

9.4 调试运行

本节对调试运行进行说明。

9.4.1 调试运行的概要

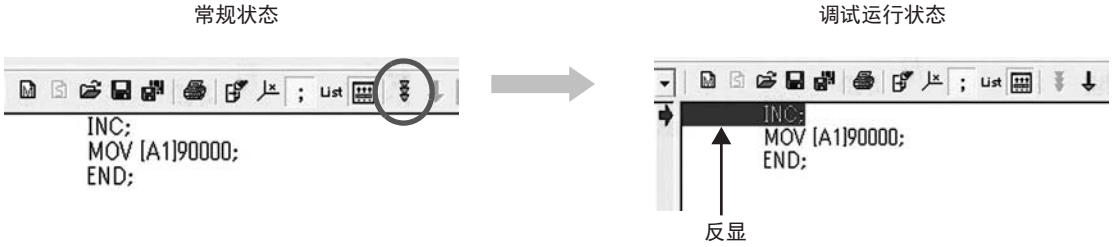
调试运行，是指运行运动程序及顺序程序调试的功能。
使用程序暂停、断点设定、1 步运行 （单段运行）等功能，可检验创建的程序的运行。
调试运行状态下会如下所示在画面上显示程序运行行。



（注） 调试运行在 MP2000 系列的全部机型中都能够使用。

要开始调试运行，在连接到控制器后，点击 ***Motion Editor*** 窗口中的 图标。

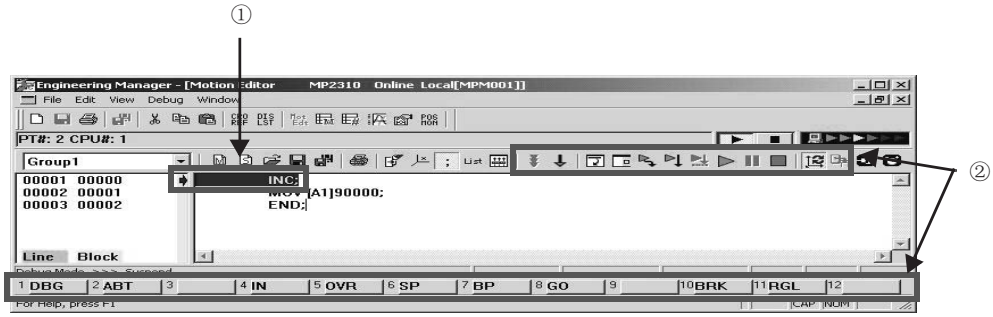
调试运行状态下会如下所示在画面上反显程序运行行。



补充

作为开始调试运行的前提条件，请进行程序的登录。

9.4.2 画面说明



①程序运行行

当前的程序运行行显示为蓝色。运动程序发生警报时，显示为红色。关于运动程序的警报，请参阅“10.2.4 运动程序警报代码 ”。

②工具图标及功能键

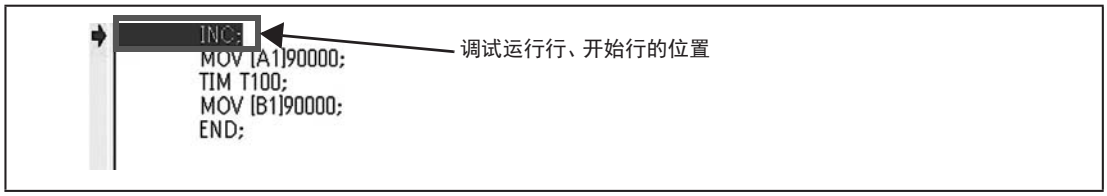
调试运行时使用的图标及功能键如下表所示。

功能	图标	键操作	说明	运动程序	顺序程序
调试模式		F1	开始调试模式。	○	○
常规运行模式		F11	退出调试模式，在正常运行模式下继续运行程序。	○	○
运行开始行的移动		F6	移动运行开始行 （开始点）。	○	○
设定 / 删除断点		F7	设定或删除断点。程序显示中表示断点本身。	○	○
Step In		F4	执行下一个程序段。命令为 MSEE 或 SSEE 时，将跳至指定子程序的起始行执行。	○	○
Step over		F5	执行下一个程序段。命令为 MSEE 或 SSEE 时，将先运行指定子程序，然后执行下一个程序段的 MSEE 或 SSEE 命令。	○	○
运行		F8	调试模式下，继续运行运动程序。	○	○
中断		F10	调试模式下，暂停正在运行的运动程序。	○	○
强制退出		F2	强制中止运行运动程序。	○	×
更新当前位置		—	更新当前的位置坐标。	○	×
设定运动任务		—	设定选中程序的同时执行编号、层级编号、任务。	○	○
断点的有效化 / 无效化	—	—	启用或禁用断点。 可通过调试菜单或鼠标右键菜单进行操作。	○	○
添加快速监视	—	—	在快速参考中登录寄存器。 可通过鼠标右键菜单进行操作。	○	○

（注）○：可使用 ×：不能使用

■调试模式

切换到调试模式。此时调试运动会从程序的起始行开始运行。



补充

切换到调试模式时的调试开始行在运动程序和顺序程序中分别如下所示。

(1) 运动程序

- 针对未启动的运动程序进行操作时
  如上例所示，从程序的起始行开始进行调试运行。
- 针对已启动的运动程序进行操作时
  在轴动作过程中切换操作到调试模式时，会在该轴的动作完成后，再从下一段程序的位置开始进行调试运行。

(2) 顺序程序

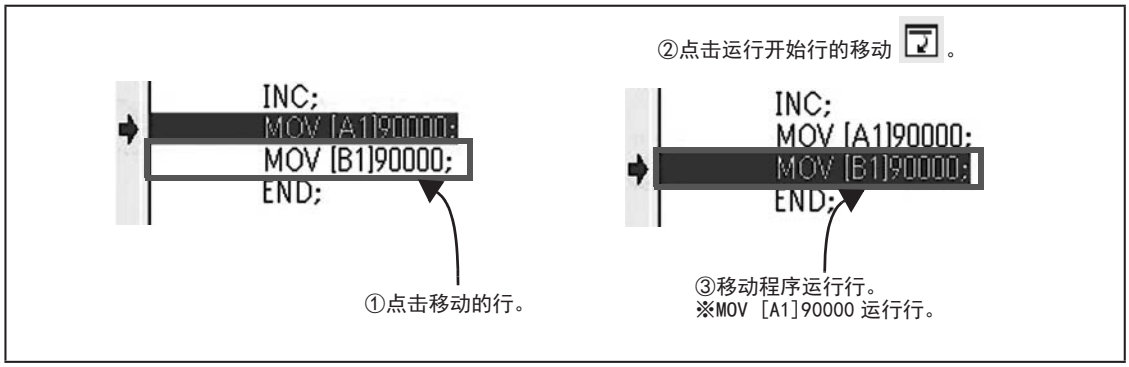
- 针对未启动的顺序程序进行操作时
  无法进行调试运行。
- 针对已启动的顺序程序进行操作时
  如上例所示，从程序的起始行开始进行调试运行。

■常规运行模式

切换到常规运行。此时调试运行被解除，程序从当前运行行重新开始运行。当前设定的断点全部解除。

■运行开始行的移动

移动程序运行行。

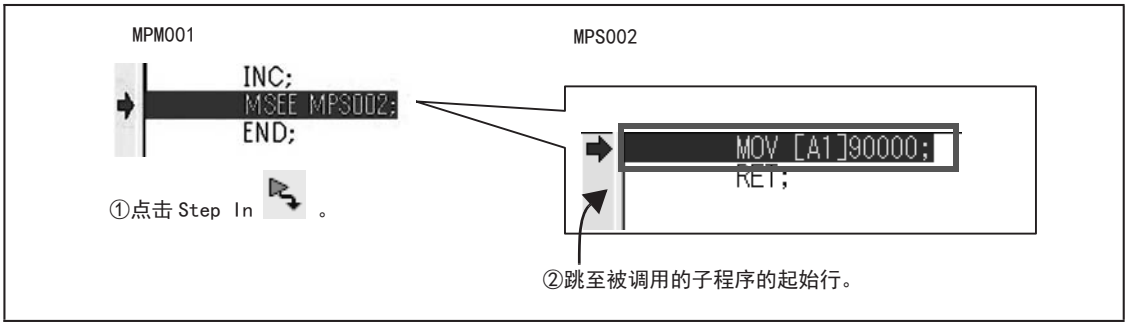


■设定 / 删除断点

断点最多可设定 4 个。
已设定断点的行，如果再次点击，则会删除断点。

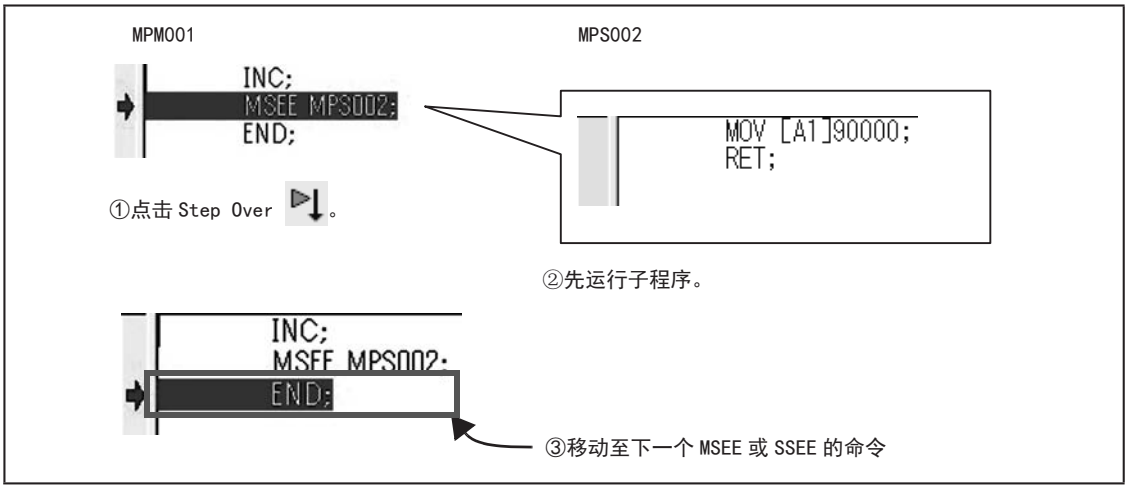
■ Step In

运行 1 行程序。
在执行 MSEE 或 SSEE 命令的同时，进行 Step In 操作时，会跳至被调用的子程序起始行开始运行。



■ Step over

运行 1 行程序。
在执行 MSEE 或 SSEE 命令的同时，进行 Step Over 操作时，会先运行子程序，再移动至下一个 MSEE 或 SSEE 的命令。

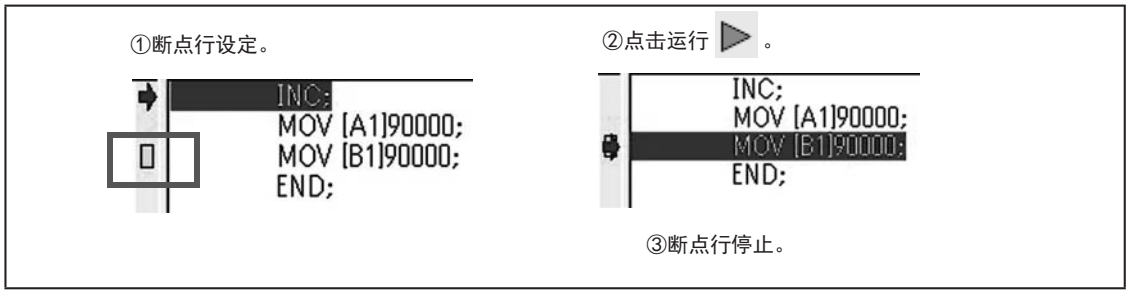


补充

使用 SNGD/SNGE 命令，可将多个处理设定成执行 Step in 和 Step over 的处理单位。

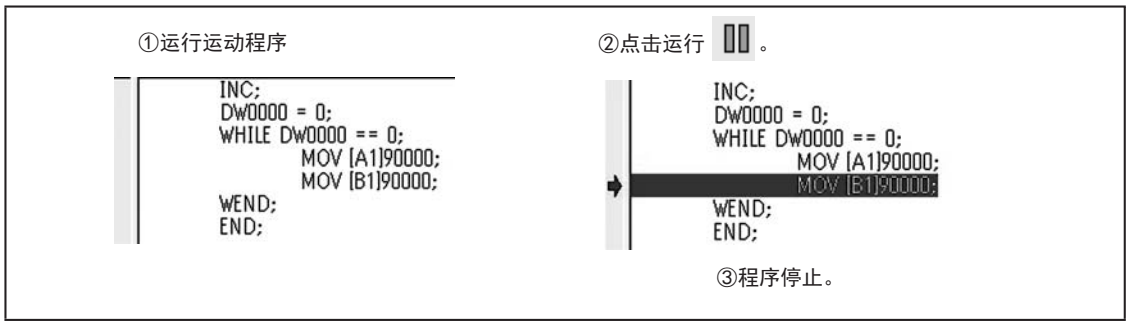
■运行

连续运行程序。到达断点行时，在断点行停止。



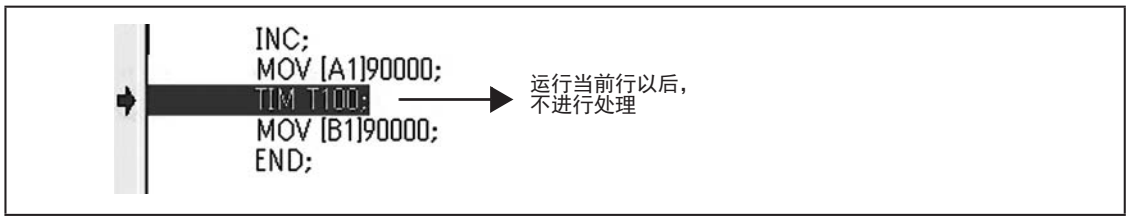
■中断

暂停正在调试运行的程序。重新开始程序时，点击运行图标。



■强制退出

强制退出正在调试运行的程序。



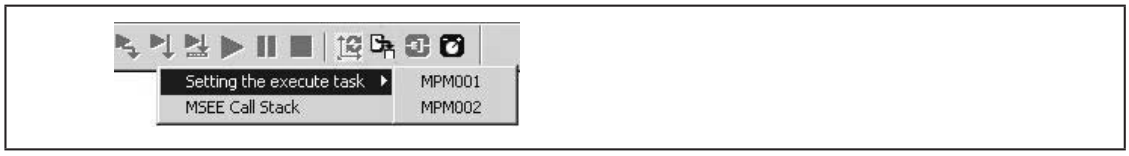
■更新当前位置

该图标的功能与 PLD 命令相同。选择了该图标时，在 Step In、Step Over、运行中通过系统来处理 PLD 命令。

PLD 命令的详细情况，请参阅 “8.3.3 更新程序当前位置 （PLD）”。

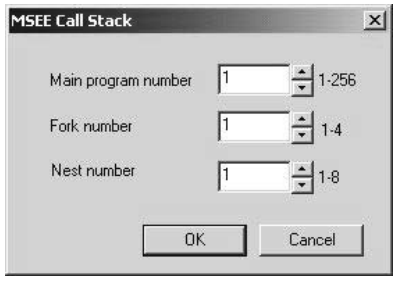
■设定运动任务 （仅限子程序）

设定子程序运行程序监视器、调试运行的子程序信息。显示正在启动的主程序，设定子程序从指定的主程序中调用。



■设定调用栈  （仅限子程序）

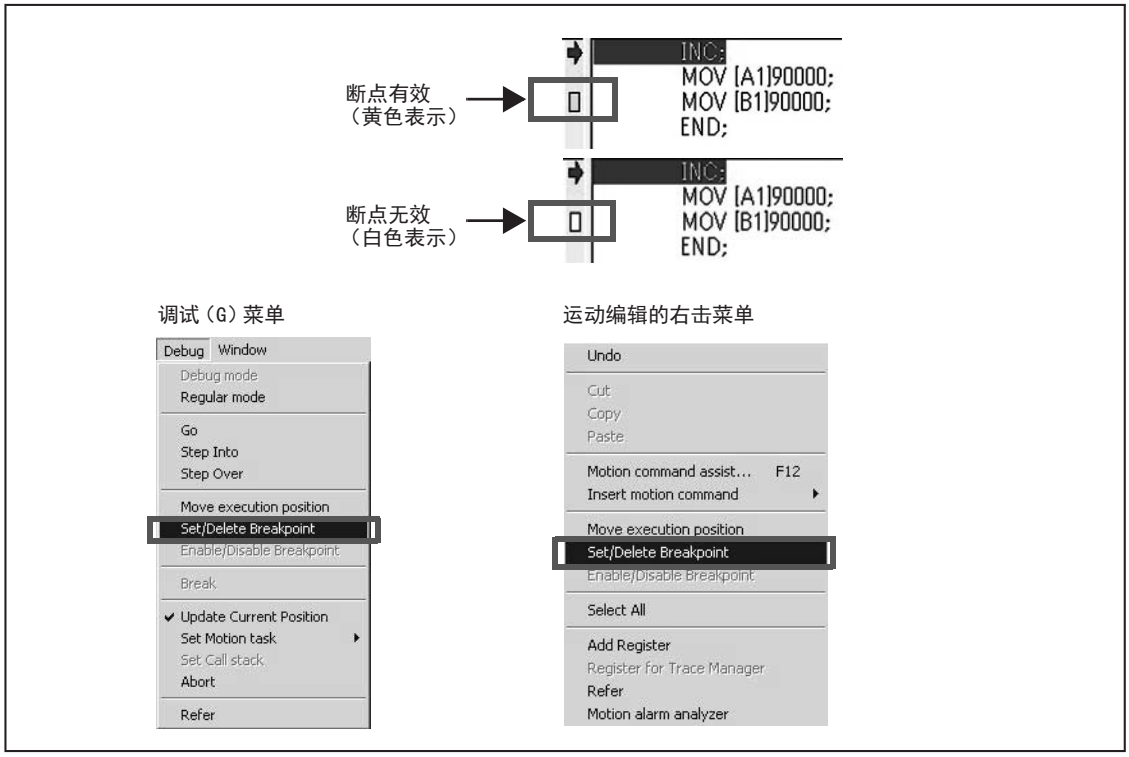
通过 ***Setting the execute task***（设定运动任务）设定详细的子程序信息。



项目	内容
主程序编号	设定调用子程序的主程序编号。
FORK 编号	设定用于调用子程序的主程序的同时执行编号。 例） 运行 MPS004 的程序监视器、调试运行时，在 FORK 编号中设定 “3”。 MPM001 PFORK Label1 Label2 Label3 Label4; Label1: “第1并列 MSEE MPS002; JOINTO LabelX; Label2: “第2并列 MSEE MPS003; JOINTO LabelX; Label3: “第3并列 MSEE MPS004; JOINTO LabelX; Label4: “第4并列 MSEE MPS005; JOINTO LabelX; LabelX: PJOINT; ···· END;
NEST 阶层	设定子程序调用的阶层。 例） 运行 MPS003 的程序监视器、调试运行时，在 NEST 阶层中设定 “2”。 MPM001 MW0000=1; MSEE MPS002; ···· ···· END; MPS002（嵌套1） MW0000=2; MSEE MPS003; ···· ···· RET; MPS003（嵌套2） MW0000=3; ···· ···· RET;

■设定 / 删除断点

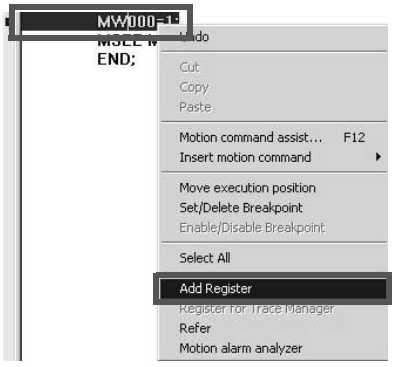
启用或禁用断点。



■添加快速监视

能够将编辑器上的寄存器登录到快速参考。
可通过登录到快速参考来确认寄存器值。

①在监视对象的寄存器上弹出右击菜单，选择 ***Add Register***（追加快速监视）。



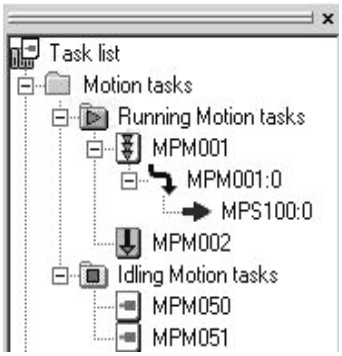
②寄存器被登录到快速参考的监视页面。

9.5 运动任务管理器

本节对运动任务管理器进行说明。

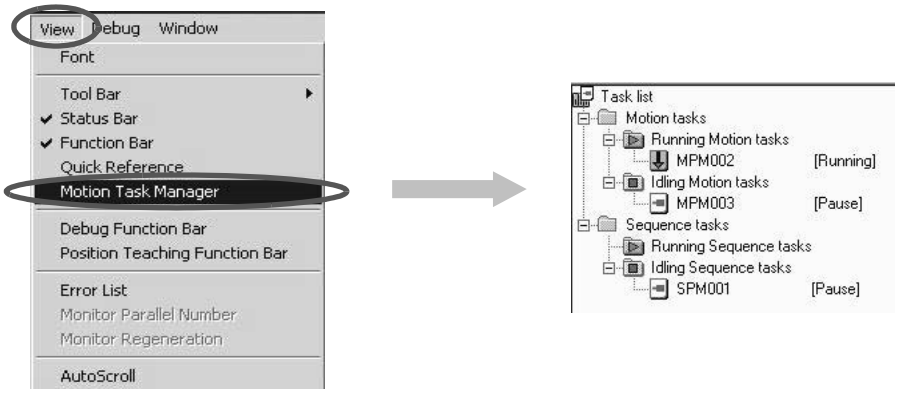
9.5.1 运动任务管理器的概要

运动任务管理器具有监视运动任务一览及任务运行状况的功能。
Motion Task Manager 画面中以树状显示正在执行和停止中的任务。

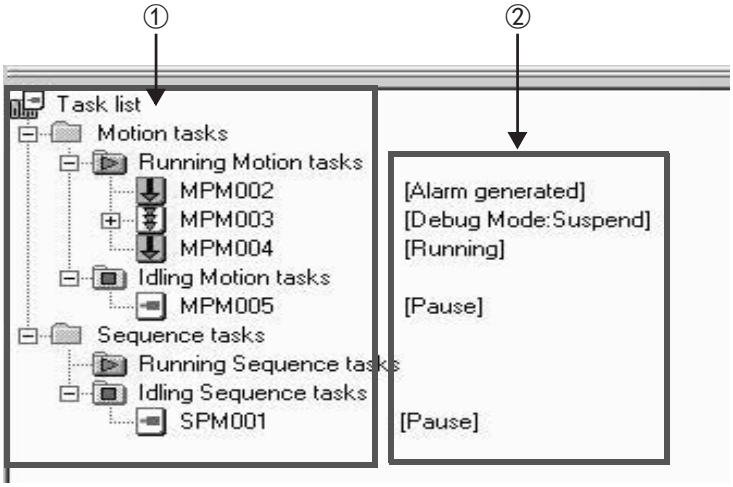


（注）运动任务管理器在 MP2000 系列的全部机型中都能够使用。

启动运动编辑器，应在运动编辑器中选择 ***View*** － ***Motion Task Manager in the Motion Editor***。



9.5.2 画面说明

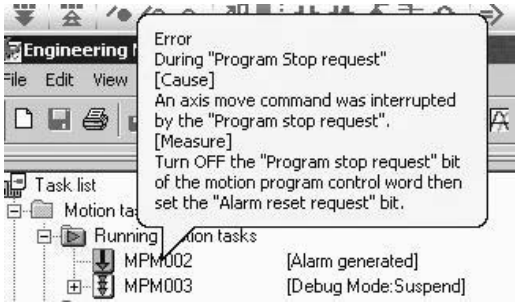


①任务执行状态树状显示

以树状显示运动程序及顺序程序的运行状态。
双击程序能够在运动编辑器中打开程序。

②显示程序状态

显示程序的状态。

显示	状态
停止中	程序已停止。
调试模式中	程序处于调试运行中。
调试模式中：单段停止中	程序在调试模式运行中，处于中断停止中。
运行中	程序正在运行。
警报发生中	正在发生运动程序警报。 将鼠标放置在程序名称上时，可确认警报内容。 

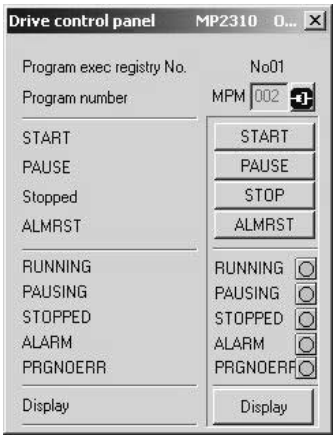
9.6 运行控制面板

本节对运动控制面板进行说明。

9.6.1 运行控制面板的概要

运行已创建的运动程序需登录到 MP2000 系统，再通过用户应用程序发出程序的运行开始信号请求。
在创建该用户应用程序之前，如果不运行已创建的运动程序，则可通过 Drive control panel 对话框进行程序的试运行。通过该对话框可发出运行开始请求及停止请求、警报复位请求等指令。
运行控制面板功能适用于以下版本的 MPE720 软件工具。

MPE720	支持版本
MPE720 Ver.5	Ver5.38 或更高
MPE720 Ver.6	Ver6.04 或更高 Ver6.04 Lite 或更高



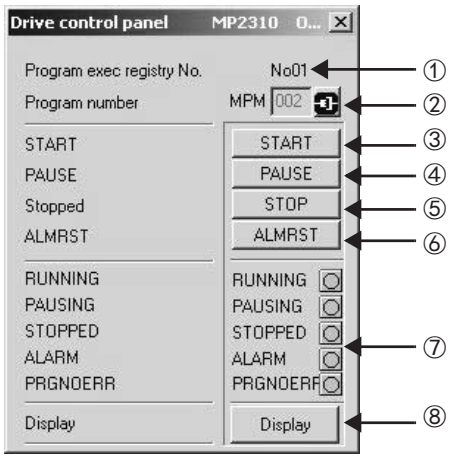
（注） 运行控制面板不能使用调试运行时那样的断点设定、单步执行 （单段执行）功能。

⚠注意

- 轴会随运动程序的运行开始移动，因此，请在通过控制面板运行程序前充分确认运动区域的安全性。
- 请勿通过顺序程序或梯形图程序覆盖保存运动程序的控制寄存器。如果覆盖保存，则可能会导致无法通过本面板执行控制。
- 针对同一轴，请勿在多个程序中同时执行轴移动指令。否则会出现意外动作。

启动运行控制面板时，在 ***Motion Editor*** 窗口中点击 “ ” 图标。

9.6.2 画面说明



①程序登录编号

显示试运行程序的登录编号。
程序登录编号需预先通过 M-EXECUTOR 的 ***Program execution registry*** （程序登录）画面进行设定。

②程序编号

显示试运行程序编号。
程序编号需预先通过 M-EXECUTOR 的 ***Program execution registry*** （程序登录）画面进行设定。

③ ***START*** 按钮

开始试运行。

④ ***PAUSE*** 按钮

进行试运行时，暂停试运行。

⑤ ***STOP*** 按钮

进行试运行时，停止试运行。

⑥ ***ALMRST*** 按钮

发生警报时，执行警报复位。

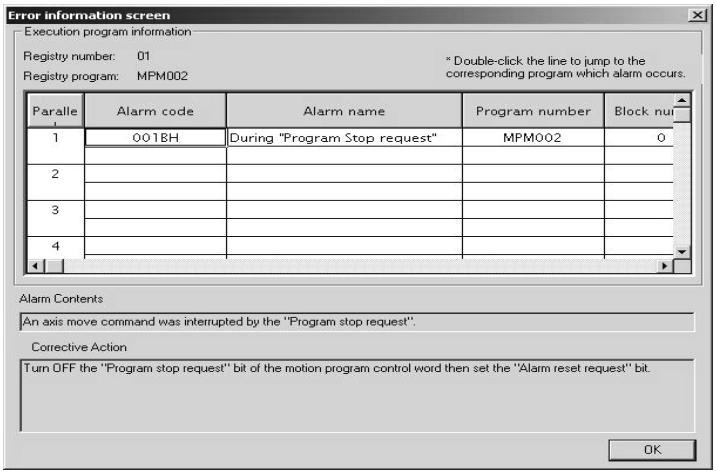
⑦显示运行状态

通过 LED 的点亮来显示试运行的状态。以下是关于各 LED 的说明。

- ***RUNNING*** ： 点击 ***START*** 按钮，LED 点亮。
- ***PAUSING*** ： 点击 ***PAUSE*** 按钮，LED 点亮。
- ***STOPPED*** ： 点击 ***STOP*** 按钮，LED 点亮。
- ***ALARM*** ： 发生程序警报时，LED 点亮。
- ***PRGNOERR***：发生程序编号溢出错误时，LED 点亮。

⑧显示按钮

显示错误信息画面。有关错误信息和错误信息画面的详细信息，请参阅“10.2.4 运动程序警报代码 ”。以下是错误信息画面的示例。



9.7 测试运行功能

本节对测试运行功能进行说明。

9.7.1 测试运行功能的概要

测试运行功能可通过画面连接到 MP2000 控制器上的轴试运行。在无程序的条件下可发出伺服 ON、伺服 OFF、JOG 动作、STEP 动作的指令。测试运行功能可适用于以下版本的 MPE720 软件工具。

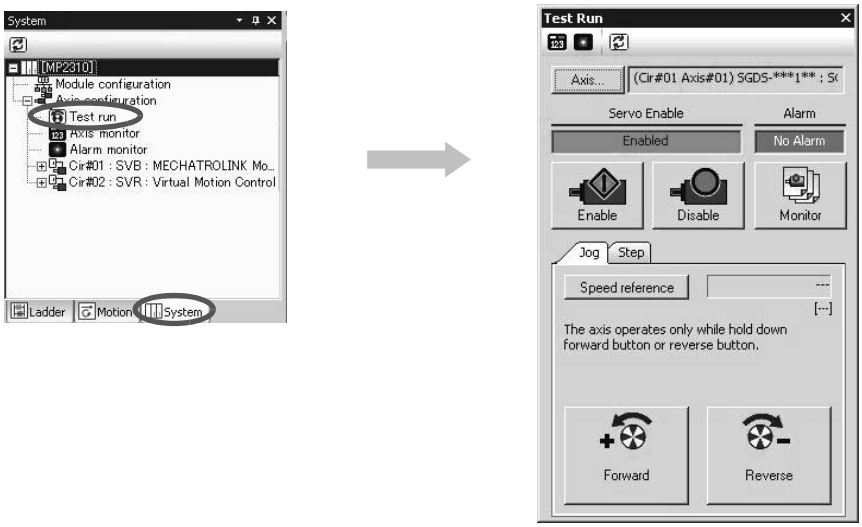
MPE720	支持版本
MPE720 Ver.5	不支持。
MPE720 Ver.6	Ver6.04 或更高 Ver6.04 Lite 或更高

（注）测试运行功能在 MP2000 系列的全部机型中都能够使用。

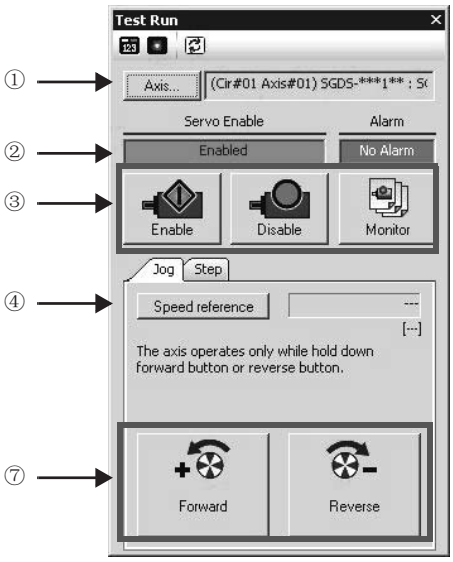
⚠注意

- 轴会发生动作，因此在操作之前，请务必确保安全。
- 运行之前，请确认紧急停止措施，以保证在正常运行中可随时中止轴动作。
- 在通过测试运行功能使轴动作前，请停止正在运行的所有梯形图程序和运动程序。

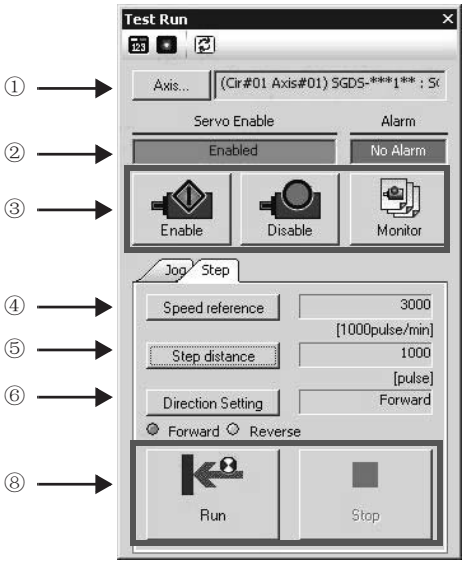
要启动测试运行功能，需在 ***System*** 子窗口中双击 ***Test run*** （测试运行）项目。



9.7.2 画面说明



点击 ***JOG*** 或 ***STEP*** 用以切换对应的选项卡



①轴选择 （Axis）

选择测试运行中进行动作的轴。

②伺服 ON/ 伺服 OFF、警报显示 （Servo Enable、Alarm）

显示伺服 ON/OFF 状态及轴警报状态。

③伺服 ON、伺服 OFF、监视 （Enable、Disable、Monitor）

执行伺服 ON 或伺服 OFF。该操作会更新运动设定参数。
点击 ***Monitor*** 按钮，显示轴警报的详细情况。

④设定速度指令值 （Speed reference）

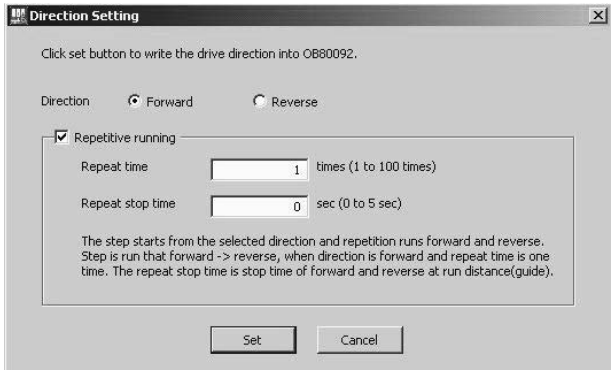
设定速度指令值。该操作会更新运动设定参数。

⑤设定步移动量 （Step distance）

设定 STEP 运行时的 STEP 移动量。该操作会更新运动设定参数。

⑥设定运行方向 （Direction Setting）

显示设定 STEP 运行时的轴运行方向的对话框。
在对话框中指定 ***Forward*** （正转）、 ***Reverse*** （反转）。该操作会更新运动设定参数。
此外，还可在对话框中指定往复的 STEP 运行。



⑦ JOG 运行

执行 JOG 运行。点击 Forward 或 Reverse 按钮后，指定的轴会按选择方向动作。取消点击后，轴停止动作。

⑧ STEP 运行

执行 STEP。点击按钮，则指定的轴会执行 1 次 STEP 动作。与 JOG 运行不同，不需要连续点击按钮。指定了往复运行时，轴会在执行了指定次数的 STEP 往复动作后退出动作。在往复运行中，也可以停止轴动作。

9.8 轴运行监视、轴警报监视

本节将对轴运行监视、轴警报监视进行说明。

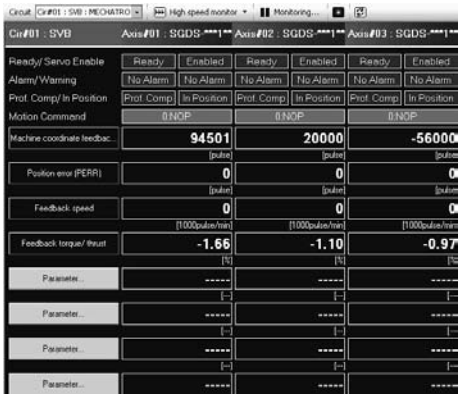
9.8.1 轴运行监视、轴警报监视的概要

轴运行监视，是对连接到 MP2000 控制器上轴的运行状态进行监视的功能。可以显示的运行状态包括，状态显示（准备完成 / 伺服 ON、警报 / 警告、输出 / 定位完成、运动命令）和用户指定的监视参数。
轴警报监视，是对连接到 MP2000 控制器上轴的警报信息进行监视的功能。
轴运行监视、轴警报监视功能适用于以下版本的 MPE720 软件工具。

MPE720	支持版本
MPE720 Ver.5	不支持。
MPE720 Ver.6	Ver6.04 或更高 Ver6.04 Lite 或更高

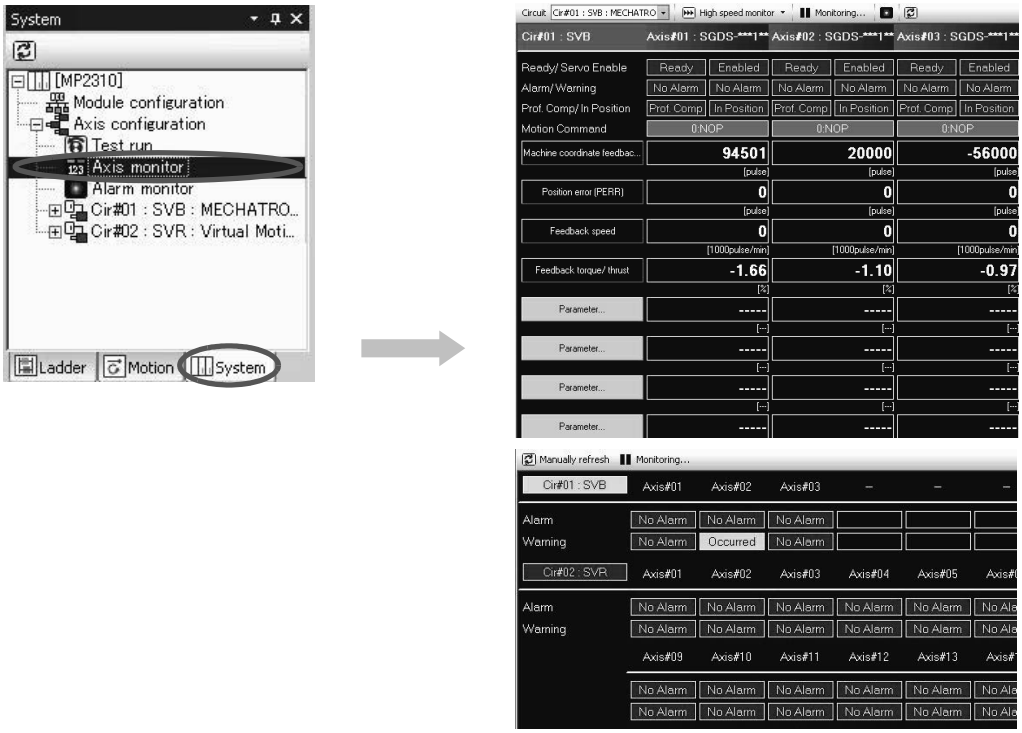
（注）轴运行监视、警报监视在 MP2000 系列的全部机型中都能够使用。

轴运行监视



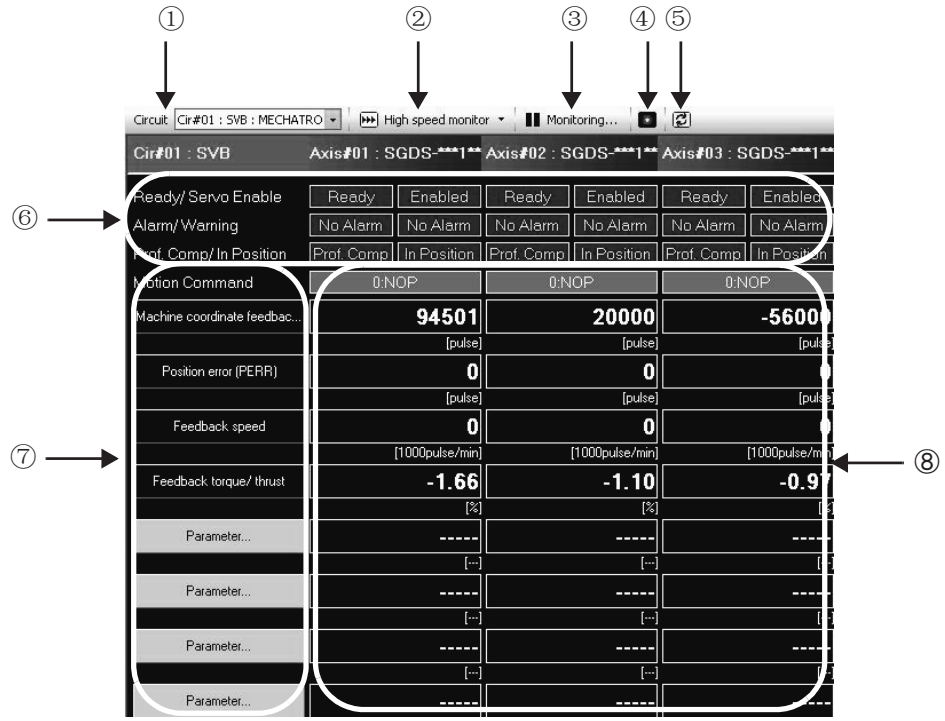
轴警报监视

启动轴运行监视、轴警报监视时，应在 ***System*** 子窗口中双击 ***Axis monitor***（轴运行监视）或 ***Alarm monitor***（轴警报监视）。



9.8.2 画面说明

（1）轴运行监视



①线路选择

显示所选线路的运动监视参数。

②选择监视周期

选择监视周期。

③监视的暂停 / 开始

暂停或开始监视。

④轴警报监视

显示轴警报监视画面。

⑤更新为最新信息

将轴运行监视的画面更新到最新的状态。

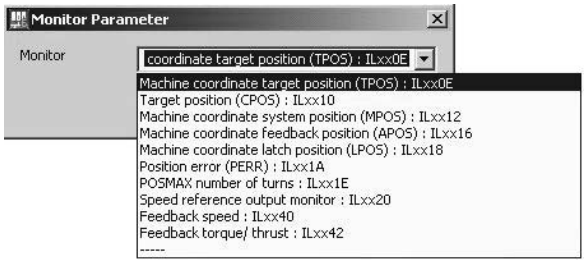
⑥状态显示

显示准备完成 / 伺服 ON、警报 / 警告、输出 / 定位完成、运动命令的状态。状态不同，显示会有所不同。

⑦选择监视参数

最多可选择 8 个监视参数。
默认状态下显示机械坐标系反馈位置 （APOS）、位置偏差 （PERR）、反馈速度、转矩 / 推力指令的监视参数。

点击 Parameter... ，可通过下面画面中的下拉菜单选择监视参数。



下拉菜单中可以显示的监视参数

监视参数	寄存器	单位
机械坐标系目标位置 （TPOS）	ILxx0E	指令单位
机械坐标系计算位置 （CPOS）	ILxx10	指令单位
机械坐标系指令位置 （MPOS）	ILxx12	指令单位
32 Bit 计算位置 （DPOS）	ILxx14	指令单位
机械坐标系反馈位置 （APOS）	ILxx16	指令单位
机械坐标系闩锁位置 （LPOS）	ILxx18	指令单位
位置偏差 （PERR）	ILxx1A	指令单位
POSMAX 转数	ILxx1E	[rev]
速度指令输出值	ILxx20	[pulse/s]
速度反馈	ILxx40	指定的速度单位
转矩 / 推力指令反馈	ILxx42	指定的转矩单位

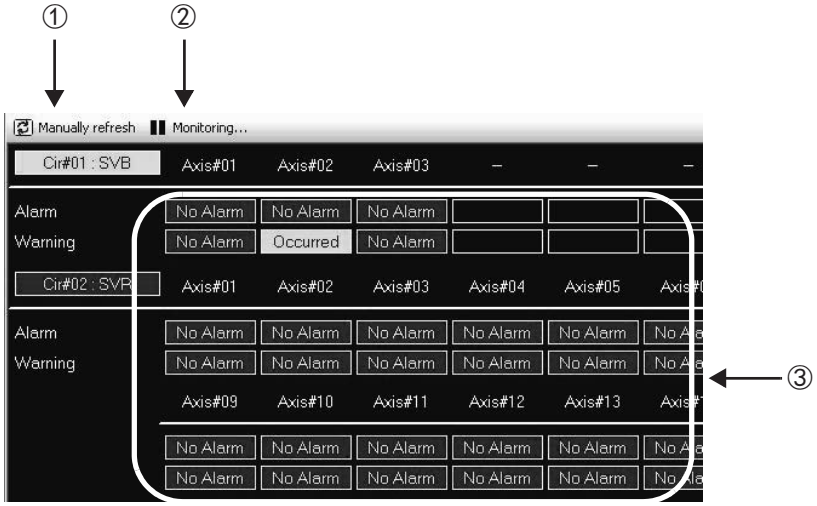
补充

在对话框中，比如，类似 “IW8000” 等直接输入运动寄存器编号，则能够指定下拉菜单中没有显示的监视参数。

⑧显示监视参数

显示指定的监视参数的状态。

(2) 轴警报监视



①手动更新 （Manually refresh）

手动更新警报及警告信息。

②监视的暂停 / 开始 （Monitoring）

暂停或开始监视。

③显示警报及警告

显示警报及警告状态。

显示	轴状态
No Alarm （显示蓝色）	未发生警报或警告
Alarm （显示红色）	正在发生警报
Warning （显示黄色）	正在发生警报

10 章

故障诊断

本章对运动程序的各种故障的原因查明及处理方法（故障诊断）进行说明。

10.1 故障诊断

本节列出了运动程序及顺序程序的故障诊断方法和故障一览。

10.1.1 故障诊断的基本流程

当发生故障时，应尽早查明故障原因并予以处理，从而让系统重新恢复正常。故障诊断时的基本流程如下所示。

STEP1	通过目测，确认以下事项
• 机械的运动状态 （停止时的状态） • 电源的状态 • 输入输出设备的状态 • 配线的状态 • 各种显示状态 （各模块的 LED 显示） • 各种设定开关的状态 （DIPSW 等开关） • 参数设定、程序内容	

↓

STEP2	通过以下操作来观察故障是否发生变化。
• 将控制器设为 STOP 状态。 • 执行警报复位。 • 重新接通电源。	

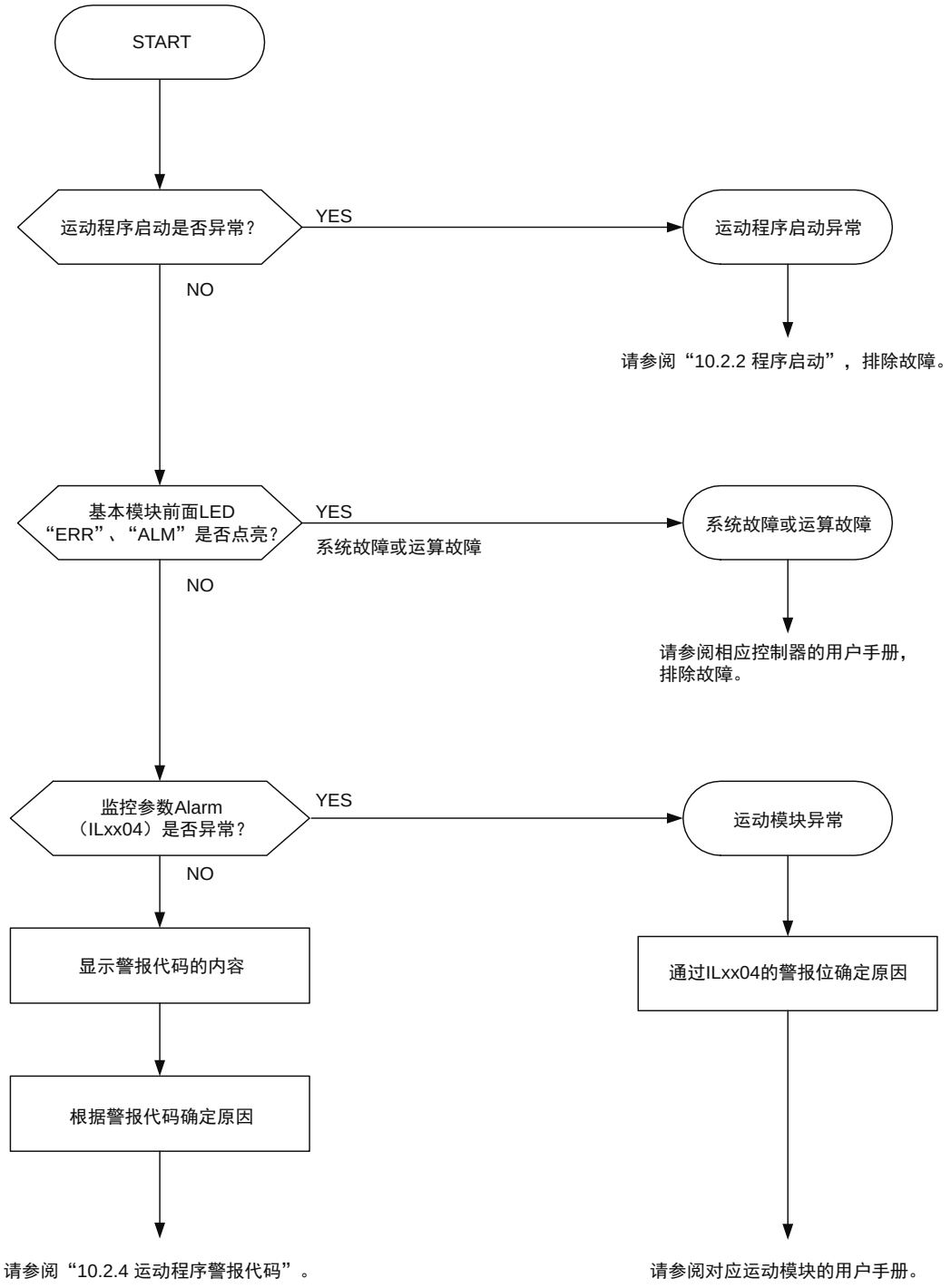
↓

STEP3	从 STEP1、2 来推断故障原因所在
• 是控制器内部还是外部？ • 是顺序控制还是运动控制？ • 是软件还是硬件？	

10. 2 运动程序的故障诊断

10. 2. 1 故障确认流程

如果推断故障原因来自运动程序，请参照以下流程来排除故障。



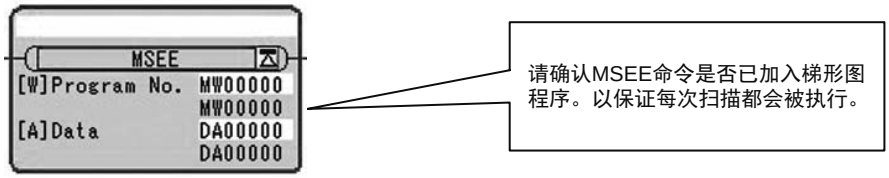
10.2.2 程序启动

运动程序启动存在异常时，请通过以下措施排除故障。

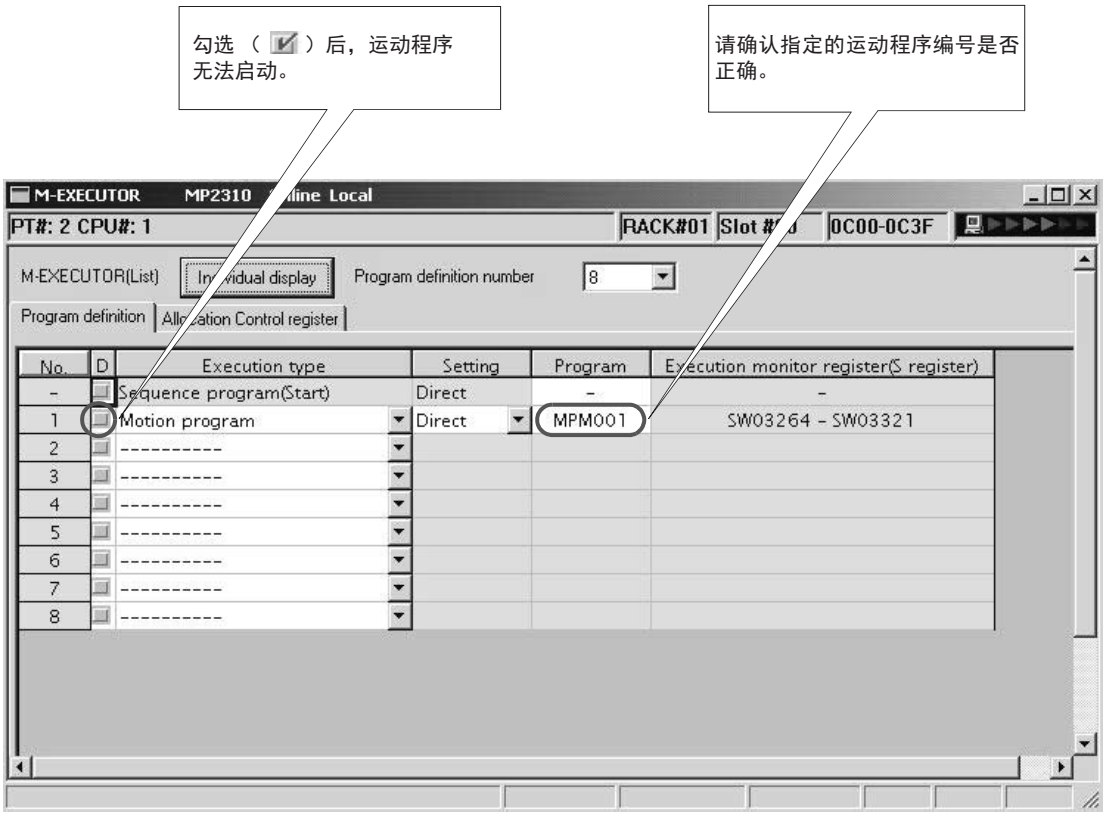
(1) 确认能够正常登录到系统。

启动运动程序需预先登录到系统。登录到系统包括将 MSEE 命令加入到 DWG.H 和登录到 M-EXECUTOR 模块的 2 种方法。
关于上述 2 种方法的详细说明，请参阅 “4.3.2 程序的登录 ”。

- 将 MSEE 命令加入到 DWG.H

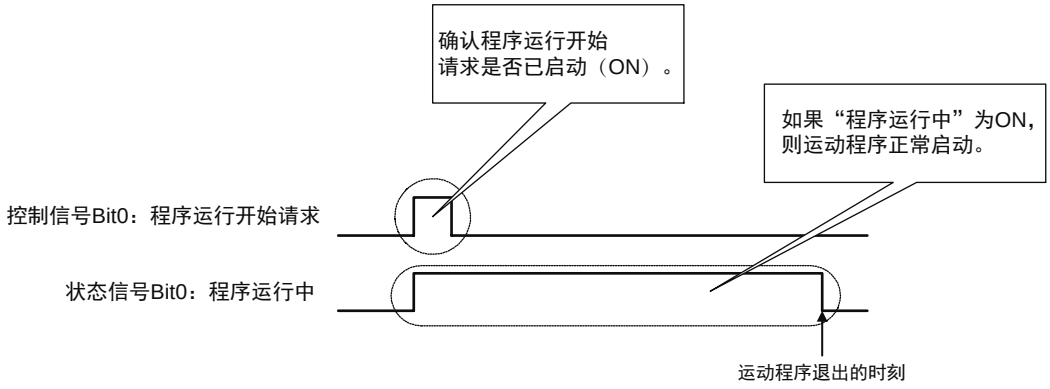


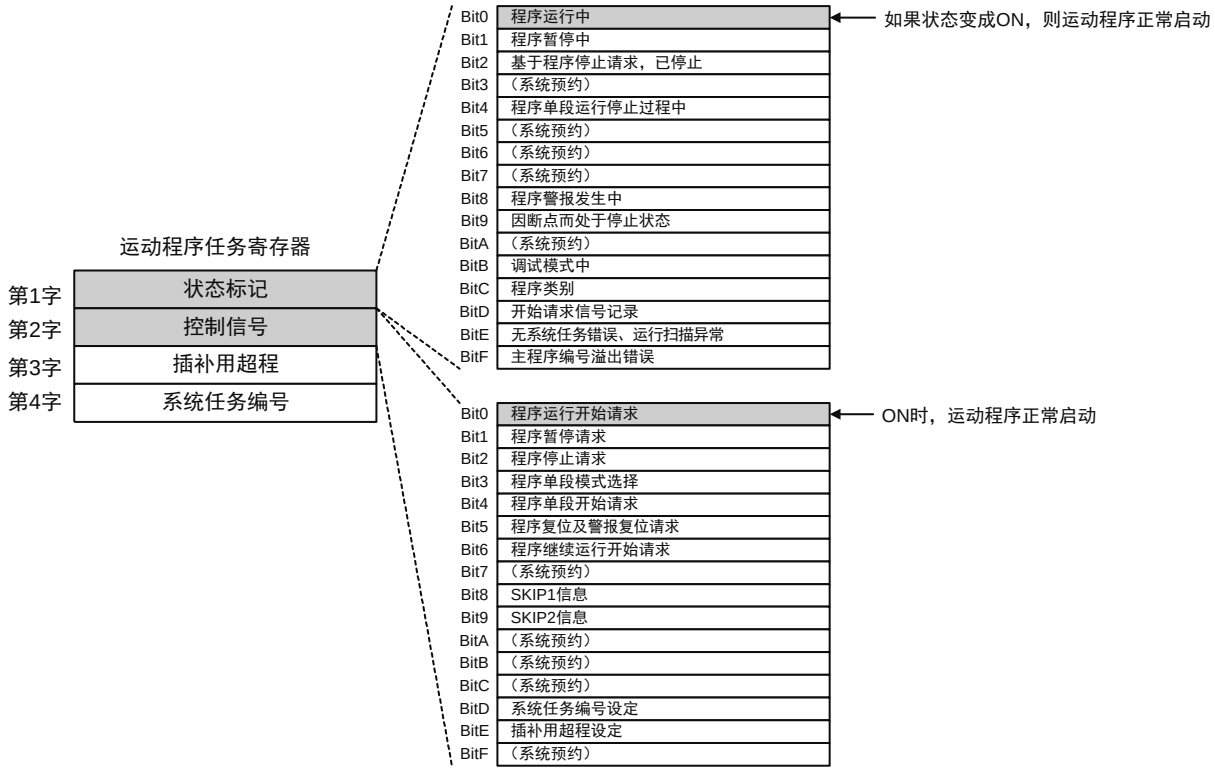
- 登录到 M-EXECUTOR 模块



(2) 请确认程序启动请求和 “ 程序运行中 ” 状态。

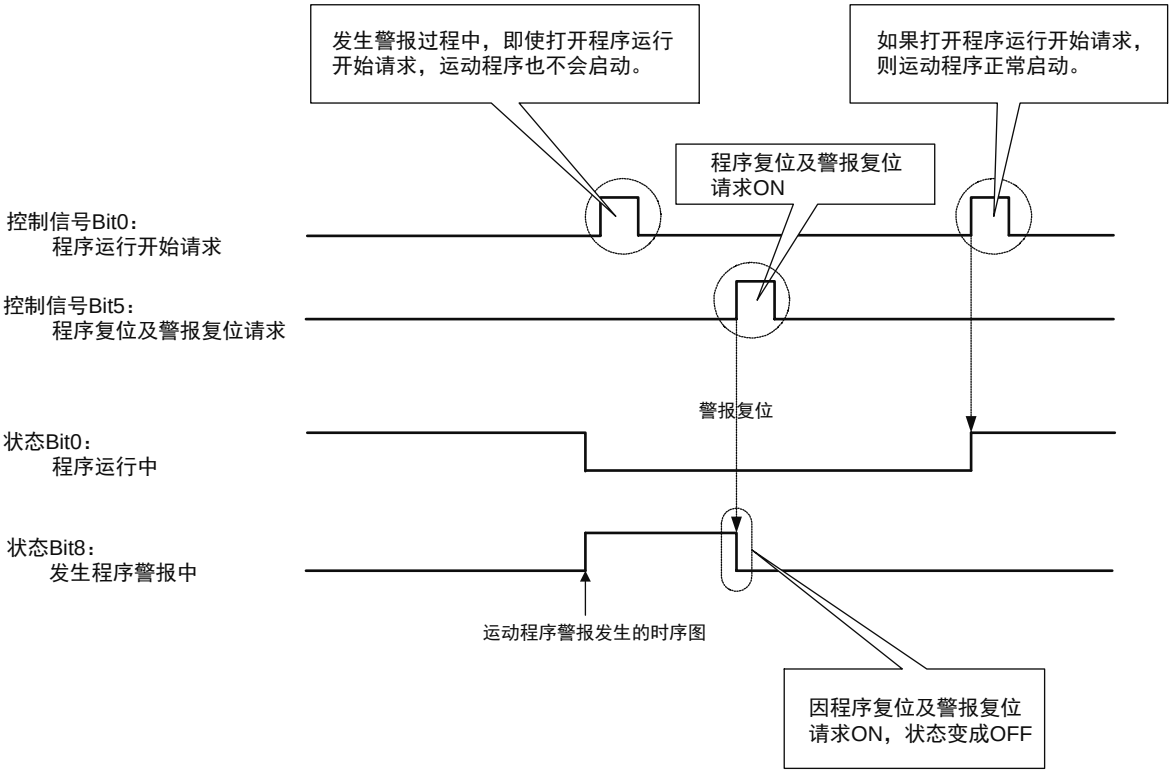
当 MP2000 系统检测到运动程序控制信号 Bit0 “ 程序运行开始请求 ” 发生了 OFF → ON 变化时，启动运动程序。运动程序启动成功后，状态信号 Bit0 “ 程序运行中 ” 会变为 ON。运行运动程序中的 END 命令后，状态信号 Bit0 “ 程序运行中 ” 会变为 OFF。
如果运动程序需要重启，请将运动程序的控制信号 Bit0 置为 OFF 后，在重新切换至 ON。
请使用数据跟踪功能确认以上信号的 ON/OFF 状态是否正确。





(3) 请确认状态 “ 程序警报发生中 ”。

状态 Bit8“ 程序警报发生中 ” 为 ON 时，由于运动程序正在发生警报，因此，运动程序处于无法启动的状态。请参阅 “10.2.4 运动程序警报代码 ” 排除警报原因后，通过启动运动程序控制信号 Bit5“ 程序复位及警报复位请求 ” 执行警报复位，然后重新启动运动程序。



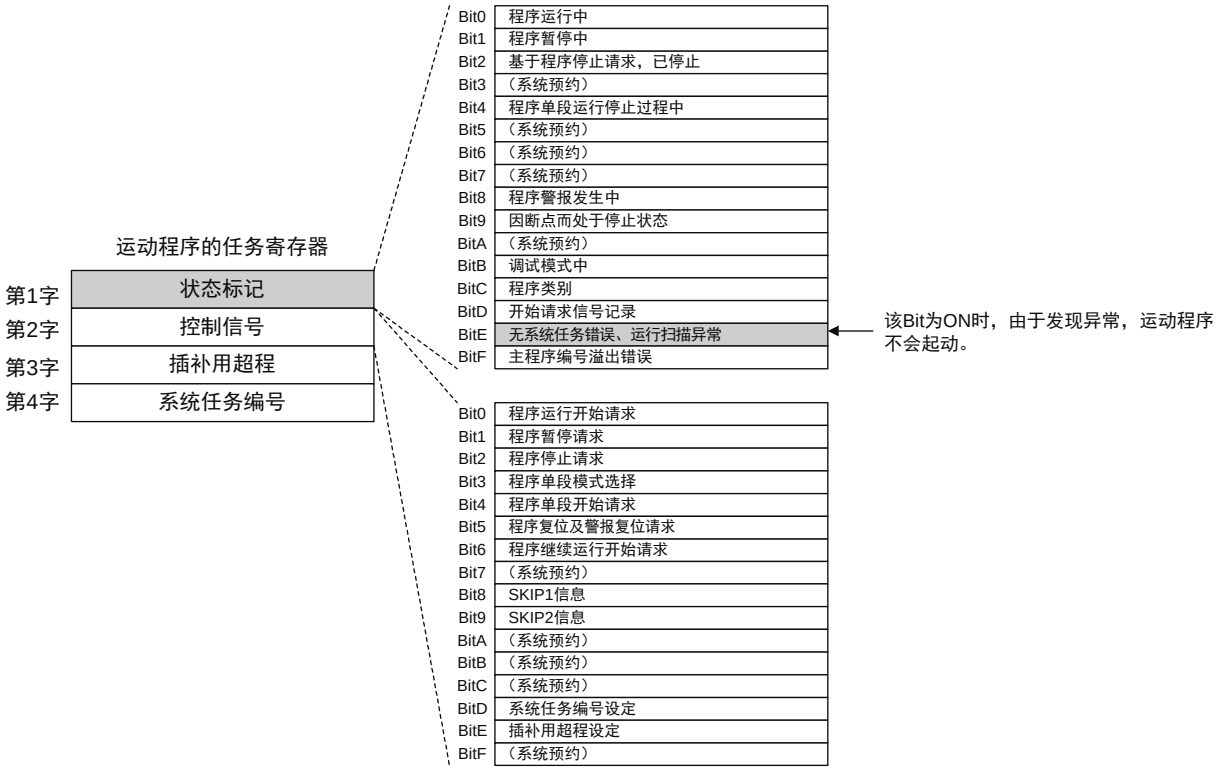
(4) 请确认状态 “ 无系统任务错误、运行扫描异常 ”。

如果运行状态 BitE“ 无系统任务错误、运行扫描异常 ” 为 ON，由于异常的原因会导致运动程序处于无法启动的状态。此时，请对以下项目进行确认。

◆执行任务的个数不少于 16。

◆由系统任务编号指定的任务未处于正在执行状态。

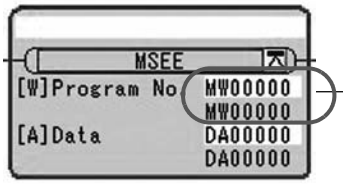
◆高速扫描处理 （DWG.H）中已加入梯形图 MSEE 命令。



(5) 请确认状态 “ 主程序编号溢出错误 ”。

如果运行状态 BitF“ 主程序编号溢出错误 ” 为 ON，由于异常的原因会导致运动程序处于无法启动的状态。此时，请对以下项目进行确认。

◆在 MSEE 命令中指定的运动程序编号超出 1 ～ 256 范围。



运动程序间接指定用寄存器的值超出1～256的范围时，发生“主程序编号溢出错误”。

运动程序的任务寄存器

第1字	状态标记
第2字	控制信号
第3字	插补用超程
第4字	系统任务编号

Bit0	程序运行中
Bit1	程序暂停中
Bit2	基于程序停止请求，已停止
Bit3	（系统预约）
Bit4	程序单段运行停止过程中
Bit5	（系统预约）
Bit6	（系统预约）
Bit7	（系统预约）
Bit8	程序警报发生中
Bit9	因断点而处于停止状态
BitA	（系统预约）
BitB	调试模式中
BitC	程序类别
BitD	开始请求信号记录
BitE	无系统任务错误、运行扫描异常
BitF	主程序编号溢出错误

该Bit为ON时，由于发现异常，运动程序不会起动。

Bit0	程序运行开始请求
Bit1	程序暂停请求
Bit2	程序停止请求
Bit3	程序单段模式选择
Bit4	程序单段开始请求
Bit5	程序复位及警报复位请求
Bit6	程序继续运行开始请求
Bit7	（系统预约）
Bit8	SKIP1信息
Bit9	SKIP2信息
BitA	（系统预约）
BitB	（系统预约）
BitC	（系统预约）
BitD	系统任务编号设定
BitE	插补用超程设定
BitF	（系统预约）

10.2.3 警报代码确认方法

运动程序中发生警报时（状态 Bit8“程序警报发生中”为 ON），根据警报代码可确定发生警报的原因。运动程序的警报代码可通过以下 2 种方法中的任意一种进行确认。

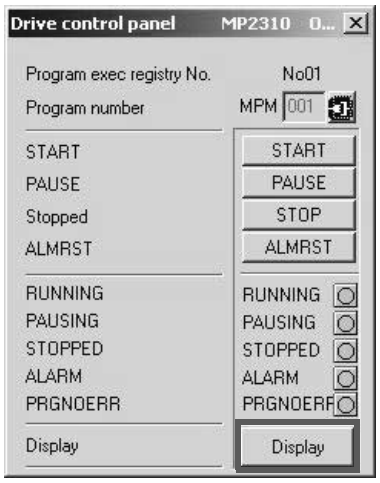
- ◆ 故障息画面
- ◆ S 寄存器

(1) 故障信息画面

显示故障信息画面的方法有 2 种。

(a) 通过运行控制面板启动的方法

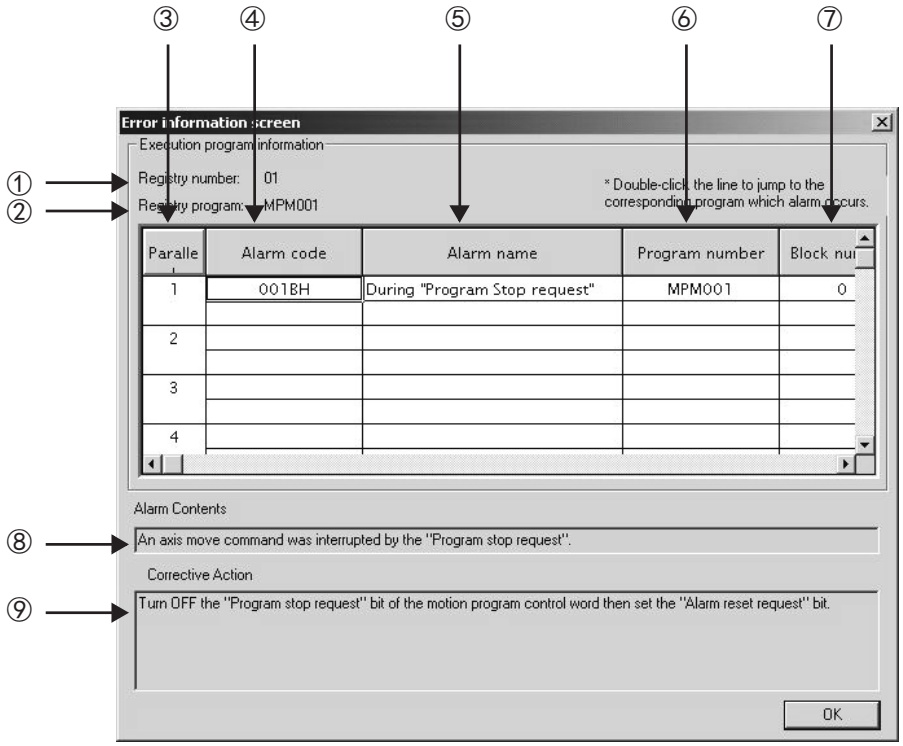
点击运行控制面板上的故障信息 ***Display*** 按钮。



(b) 通过运动编辑器中的鼠标右键菜单启动的方法

在运动编辑器上，在鼠标右键菜单中选择 ***Motion alarm analyzer***（运动警报分析）。

介绍故障信息画面。



①登录编号 （Registry number）

登录到 M-EXECUTOR 程序定义的运动程序发生警报时，会显示 M-EXECUTOR 的登录编号。
使用 MSEE 命令从梯形图程序引用的运动程序发生警报时，会显示 “---”。

②登录程序 （Registry program）

登录到 M-EXECUTOR 程序定义的运动程序发生警报时，会显示登录到 M-EXECUTOR 的程序名称。
使用 MSEE 命令从梯形图程序引用的运动程序发生警报时，会显示 “---”。

③同时执行编号 （Paralle）

运动程序使用同时执行 （PFORK）时，有时会同时发生多个警报。同时执行的详细信息，请参阅 “8.4.3 同时执行 （PFORK、JOINTO、PJOINT）”。

④警报代码 （Alarm code）

显示警报代码。

⑤警报名称 （Alarm name）

显示警报名称。

⑥程序编号 （Program number）

显示正在发生故障的程序名称。

⑦程序段编号 （Block number）

显示发生故障的程序段编号。双击后会跳至发生警报的相应程序。
程序段编号显示在运动编辑器上。

⑧警报内容 （Alarm Contents）

显示警报内容。

⑨处理方法 （Corrective Action）

显示警报的处理方法。

(2)S 寄存器

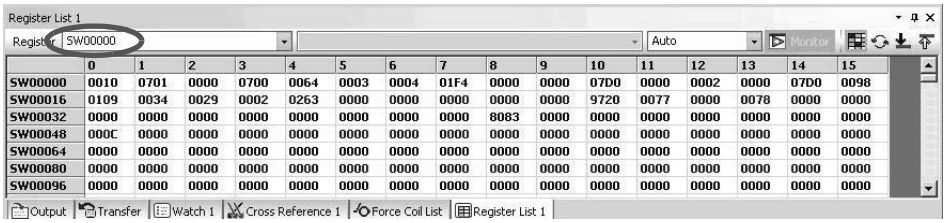
在 S 寄存器 （SW03200 ～ SW04191）的运动程序的运行信息中保存有运动程序的警报代码。
通过使用的系统任务和同时执行编号可区分运动程序警报代码的 S 寄存器编号。
有关 S 寄存器 （SW03200 ～ SW04191）的运动程序的运行信息的一览表，请参阅下页。

补充

请按照以下步骤打开 MPE720 Ver .6 软件工具中的寄存器列表。

1. 在 MPE720 Ver.6 画面中打开 ***Register List*** 子窗口。
   默认设置下，画面下方的子窗口中会显示寄存器列表 ***Register List1*** 选项卡。

2. 在 ***Register*** 栏中输入系统寄存器的起始编号 “SWxxxxx”，则会显示从起始编号开始的系统寄存器的内容。



（注）数据类型默认为 “ 十进制 ”。将光标放到列表上，单击鼠标右键，在弹出菜单中选择 “ 十六进制 ”，则会如上图所示，以十六进制数来显示。

· 系统任务编号 1 ～ 8

系统任务编号		系统任务 1	系统任务 2	系统任务 3	系统任务 4	系统任务 5	系统任务 6	系统任务 7	系统任务 8
运行中主程序编号		SW03200	SW03201	SW03202	SW03203	SW03204	SW03205	SW03206	SW03207
状态		SW03264	SW03322	SW03380	SW03438	SW03496	SW03554	SW03612	SW3670
控制信号		SW03265	SW03323	SW03381	SW03439	SW03497	SW03555	SW03613	SW3671
同时执行 0	程序编号	SW03266	SW03324	SW03382	SW03440	SW03498	SW03556	SW03614	SW3672
	程序段编号	SW03267	SW03325	SW03383	SW03441	SW03499	SW03557	SW03615	SW3673
	警报代码	SW03268	SW03326	SW03384	SW03442	SW03500	SW03558	SW03616	SW3674
同时执行 1	程序编号	SW03269	SW03327	SW03385	SW03443	SW03501	SW03559	SW03617	SW3675
	程序段编号	SW03270	SW03328	SW03386	SW03444	SW03502	SW03560	SW03618	SW3676
	警报代码	SW03271	SW03329	SW03387	SW03445	SW03503	SW03561	SW03619	SW3677
同时执行 2	程序编号	SW03272	SW03330	SW03388	SW03446	SW03504	SW03562	SW03620	SW3678
	程序段编号	SW03273	SW03331	SW03389	SW03447	SW03505	SW03563	SW03621	SW3679
	警报代码	SW03274	SW03332	SW03390	SW03448	SW03506	SW03564	SW03622	SW3680
同时执行 3	程序编号	SW03275	SW03333	SW03391	SW03449	SW03507	SW03565	SW03623	SW3681
	程序段编号	SW03276	SW03334	SW03392	SW03450	SW03508	SW03566	SW03624	SW3682
	警报代码	SW03277	SW03335	SW03393	SW03451	SW03509	SW03567	SW03625	SW3683
同时执行 4	程序编号	SW03278	SW03336	SW03394	SW03452	SW03510	SW03568	SW03626	SW3684
	程序段编号	SW03279	SW03337	SW03395	SW03453	SW03511	SW03569	SW03627	SW3685
	警报代码	SW03280	SW03338	SW03396	SW03454	SW03512	SW03570	SW03628	SW3686
同时执行 5	程序编号	SW03281	SW03339	SW03397	SW03455	SW03513	SW03571	SW03629	SW3687
	程序段编号	SW03282	SW03340	SW03398	SW03456	SW03514	SW03572	SW03630	SW3688
	警报代码	SW03283	SW03341	SW03399	SW03457	SW03515	SW03573	SW03631	SW3689
同时执行 6	程序编号	SW03284	SW03342	SW03400	SW03458	SW03516	SW03574	SW03632	SW3690
	程序段编号	SW03285	SW03343	SW03401	SW03459	SW03517	SW03575	SW03633	SW3691
	警报代码	SW03286	SW03344	SW03402	SW03460	SW03518	SW03576	SW03634	SW3692
同时执行 7	程序编号	SW03287	SW03345	SW03403	SW03461	SW03519	SW03577	SW03635	SW3693
	程序段编号	SW03288	SW03346	SW03404	SW03462	SW03520	SW03578	SW03636	SW3694
	警报代码	SW03289	SW03347	SW03405	SW03463	SW03521	SW03579	SW03637	SW3695
逻辑轴 #1 程序当前位置		SL03290	SL03348	SL03406	SL03464	SL03522	SL03580	SL03638	SL3696
逻辑轴 #2 程序当前位置		SL03292	SL03350	SL03408	SL03466	SL03524	SL03582	SL03640	SL3698
逻辑轴 #3 程序当前位置		SL03294	SL03352	SL03410	SL03468	SL03526	SL03584	SL03642	SL3700
逻辑轴 #4 程序当前位置		SL03296	SL03354	SL03412	SL03470	SL03528	SL03586	SL03644	SL3702
逻辑轴 #5 程序当前位置		SL03298	SL03356	SL03414	SL03472	SL03530	SL03588	SL03646	SL3704
逻辑轴 #6 程序当前位置		SL03300	SL03358	SL03416	SL03474	SL03532	SL03590	SL03648	SL3706
逻辑轴 #7 程序当前位置		SL03302	SL03360	SL03418	SL03476	SL03534	SL03592	SL03650	SL3708
逻辑轴 #8 程序当前位置		SL03304	SL03362	SL03420	SL03478	SL03536	SL03594	SL03652	SL3710
逻辑轴 #9 程序当前位置		SL03306	SL03364	SL03422	SL03480	SL03538	SL03596	SL03654	SL3712
逻辑轴 #10 程序当前位置		SL03308	SL03366	SL03424	SL03482	SL03540	SL03598	SL03656	SL3714
逻辑轴 #11 程序当前位置		SL03310	SL03368	SL03426	SL03484	SL03542	SL03600	SL03658	SL3716
逻辑轴 #12 程序当前位置		SL03312	SL03370	SL03428	SL03486	SL03544	SL03602	SL03660	SL3718
逻辑轴 #13 程序当前位置		SL03314	SL03372	SL03430	SL03488	SL03546	SL03604	SL03662	SL3720
逻辑轴 #14 程序当前位置		SL03316	SL03374	SL03432	SL03490	SL03548	SL03606	SL03664	SL3722
逻辑轴 #15 程序当前位置		SL03318	SL03376	SL03434	SL03492	SL03550	SL03608	SL03666	SL3724
逻辑轴 #16 程序当前位置		SL03320	SL03378	SL03436	SL03494	SL03552	SL03610	SL03668	SL3726

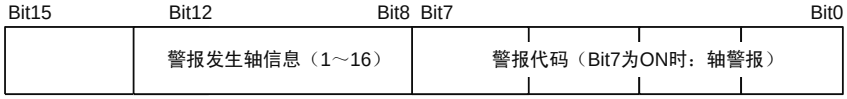
· 系统任务编号 9 ～ 16

系统任务编号		系统任务 9	系统任务 10	系统任务 11	系统任务 12	系统任务 13	系统任务 14	系统任务 15	系统任务 16
运行中主程序编号		SW03208	SW03209	SW03210	SW03211	SW03212	SW03213	SW03214	SW03215
状态		SW03728	SW03786	SW03844	SW03902	SW03960	SW04018	SW04076	SW04134
控制信号		SW03729	SW03787	SW03845	SW03903	SW03961	SW04019	SW04077	SW04135
同时执行 0	程序编号	SW03730	SW03788	SW03846	SW03904	SW03962	SW04020	SW04078	SW04136
	程序段编号	SW03731	SW03789	SW03847	SW03905	SW03963	SW04021	SW04079	SW04137
	警报代码	SW03732	SW03790	SW03848	SW03906	SW03964	SW04022	SW04080	SW04138
同时执行 1	程序编号	SW03733	SW03791	SW03849	SW03907	SW03965	SW04023	SW04081	SW04139
	程序段编号	SW03734	SW03792	SW03850	SW03908	SW03966	SW04024	SW04082	SW04140
	警报代码	SW03735	SW03793	SW03851	SW03909	SW03967	SW04025	SW04083	SW04141
同时执行 2	程序编号	SW03736	SW03794	SW03852	SW03910	SW03968	SW04026	SW04084	SW04142
	程序段编号	SW03737	SW03795	SW03853	SW03911	SW03969	SW04027	SW04085	SW04143
	警报代码	SW03738	SW03796	SW03854	SW03912	SW03970	SW04028	SW04086	SW04144
同时执行 3	程序编号	SW03739	SW03797	SW03855	SW03913	SW03971	SW04029	SW04087	SW04145
	程序段编号	SW03740	SW03798	SW03856	SW03914	SW03972	SW04030	SW04088	SW04146
	警报代码	SW03741	SW03799	SW03857	SW03915	SW03973	SW04031	SW04089	SW04147
同时执行 4	程序编号	SW03742	SW03800	SW03858	SW03916	SW03974	SW04032	SW04090	SW04148
	程序段编号	SW03743	SW03801	SW03859	SW03917	SW03975	SW04033	SW04091	SW04149
	警报代码	SW03744	SW03802	SW03860	SW03918	SW03976	SW04034	SW04092	SW04150
同时执行 5	程序编号	SW03745	SW03803	SW03861	SW03919	SW03977	SW04035	SW04093	SW04151
	程序段编号	SW03746	SW03804	SW03862	SW03920	SW03978	SW04036	SW04094	SW04152
	警报代码	SW03747	SW03805	SW03863	SW03921	SW03979	SW04037	SW04095	SW04153
同时执行 6	程序编号	SW03748	SW03806	SW03864	SW03922	SW03980	SW04038	SW04096	SW04154
	程序段编号	SW03749	SW03807	SW03865	SW03923	SW03981	SW04039	SW04097	SW04155
	警报代码	SW03750	SW03808	SW03866	SW03924	SW03982	SW04040	SW04098	SW04156
同时执行 7	程序编号	SW03751	SW03809	SW03867	SW03925	SW03983	SW04041	SW04099	SW04157
	程序段编号	SW03752	SW03810	SW03868	SW03926	SW03984	SW04042	SW04100	SW04158
	警报代码	SW03753	SW03811	SW03869	SW03927	SW03985	SW04043	SW04101	SW04159
逻辑轴 #1 程序当前位置		SL03754	SL03812	SL03870	SL03928	SL03986	SL04044	SL04102	SL04160
逻辑轴 #2 程序当前位置		SL03756	SL03814	SL03872	SL03930	SL03988	SL04046	SL04104	SL04162
逻辑轴 #3 程序当前位置		SL03758	SL03816	SL03874	SL03932	SL03990	SL04048	SL04106	SL04164
逻辑轴 #4 程序当前位置		SL03760	SL03818	SL03876	SL03934	SL03992	SL04050	SL04108	SL04166
逻辑轴 #5 程序当前位置		SL03762	SL03820	SL03878	SL03936	SL03994	SL04052	SL04110	SL04168
逻辑轴 #6 程序当前位置		SL03764	SL03822	SL03880	SL03938	SL03996	SL04054	SL04112	SL04170
逻辑轴 #7 程序当前位置		SL03766	SL03824	SL03882	SL03940	SL03998	SL04056	SL04114	SL04172
逻辑轴 #8 程序当前位置		SL03768	SL03826	SL03884	SL03942	SL04000	SL04058	SL04116	SL04174
逻辑轴 #9 程序当前位置		SL03770	SL03828	SL03886	SL03944	SL04002	SL04060	SL04118	SL04176
逻辑轴 #10 程序当前位置		SL03772	SL03830	SL03888	SL03946	SL04004	SL04062	SL04120	SL04178
逻辑轴 #11 程序当前位置		SL03774	SL03832	SL03890	SL03948	SL04006	SL04064	SL04122	SL04180
逻辑轴 #12 程序当前位置		SL03776	SL03834	SL03892	SL03950	SL04008	SL04066	SL04124	SL04182
逻辑轴 #13 程序当前位置		SL03778	SL03836	SL03894	SL03952	SL04010	SL04068	SL04126	SL04184
逻辑轴 #14 程序当前位置		SL03780	SL03838	SL03896	SL03954	SL04012	SL04070	SL04128	SL04186
逻辑轴 #15 程序当前位置		SL03782	SL03840	SL03898	SL03956	SL04014	SL04072	SL04130	SL04188
逻辑轴 #16 程序当前位置		SL03784	SL03842	SL03900	SL03958	SL04016	SL04074	SL04132	SL04190

10.2.4 运动程序警报代码

(1) 运动程序警报的构成

警报代码的构成如下图中所示。



(2) 运动程序警报代码一览

以下为运动程序的警报代码一览表。

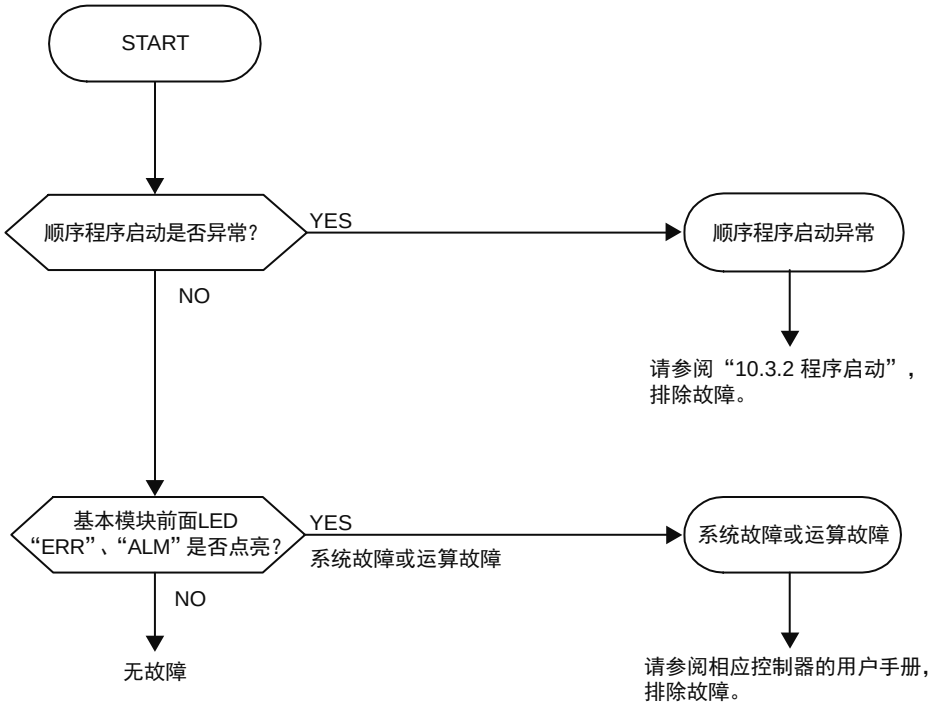
警报代码	警报名称	警报内容	处理方法
02h	除法错误	除数为 0。	修改运动程序。
10h	半径指定为周长警报	在圆弧插补或螺旋插补命令中，将半径指定为转数 (T)。	• 将半径指定改为中心坐标指定，然后执行圆弧插补或螺旋插补命令。 • 不指定转数。
11h	插补进给速度超程	插补进给速度的设定超过了FMX指令的指定范围。	重新设定插补指令的插补进给速度。
12h	未指定插补进给速度	从未进行插补进给速度指定。(设定过一次后，在同一运动程序内可省略)	在插补指令中指定插补进给速度。
13h	加速度参数转换后超出范围	间接指定的加速度参数超出了有效范围。	改变进行间接指定的寄存器的值。
14h	圆弧长度超过 LONG_MAX	在圆弧插补命令或螺旋插补命令中，圆弧长度超出了设定范围。	在圆弧插补或螺旋插补命令中修改指定的圆弧长度。
15h	指定圆弧平面时未指定纵轴	在圆弧插补或螺旋插补命令中，未指定纵轴。	利用 PLN 指令来指定轴。
16h	指定圆弧平面时未指定横轴	在圆弧插补或螺旋插补命令中，未指定横轴。	利用 PLN 指令来指定轴。
17h	指定轴过多	在圆弧插补 (2 轴) 或螺旋插补命令 (3 轴) 中，设定了过多的轴数。	重新设定圆弧插补或螺旋插补命令的轴。
18h	指定转数过多	在圆弧插补或螺旋插补命令中，转数超出了设定范围。	重新设定圆弧插补或螺旋插补命令的转数。
19h	半径超过 LONG_MAX	在圆弧插补或螺旋插补命令中，半径超出了设定范围。	在圆弧插补或螺旋插补命令中修改半径。
1Ah	中心点指定错误	在圆弧插补或螺旋插补命令中，没有正确地进行中心点指定。	在圆弧插补或螺旋插补命令中，正确地进行中心点。
1Bh	正在执行紧急停止指令	根据程序停止请求，轴移动指令停止。	将运动程序控制信号的程序停止请求设为 OFF，并将警报复位请求设为 ON。
1Ch	直线插补移动量超过 LONG_MAX	在直线插补指令中，移动量超出了设定范围。	在直线插补命令中，重新设定移动量。
1Dh	FMX 未定义	在包含插补类指令的运动程序中，未执行 FMX 指令。	执行 FMX 指令。凡是有插补类指令的程序都需要 FMX 指令。
1Eh	地址 T 超出范围	在 IAC/IDC/FMX 指令中，地址超出了设定范围。	在 IAC/IDC/FMX 指令中重新进行设定。
1Fh	地址 P 超出范围	在 IFP 指令中，地址超出了设定范围。	在 IFP 指令中重新进行设定。
21h	PFORK 执行异常	在所要调用的运动程序的 PFORK 的第 2 列和子程序的 PFORK 的第 2 列中，同时存在运动指令。	修改所要调用的运动程序或子程序。
22h	间接指定寄存器范围错误	所指定的寄存器地址超出了寄存器大小的范围。	修改运动程序。
23h	移动量超出范围	在轴移动指令中，带小数点的轴移动量超过了有效范围。	修改轴移动量。

警报代码	警报名称	警报内容	处理方法
80h	逻辑轴使用禁止中	对同一个轴同时发出了多个运动指令。	修改运动程序。
81h	无限长轴指定了超出 POSMAX	在无限长轴指定中，移动距离超出了所设定的 POSMAX。	• 重新设定固定参数的“无限长计数器最大值”。 • 修改运动程序。
82h	轴移动距离超过 LONG_MAX	轴的移动距离超出了设定范围。	修改运动程序。
84h	运动命令重复	对 1 个轴执行了多个指令。	确认是否从其他程序同时向同一个轴发出了指令；如果是，请修改程序。
85h	运动命令响应异常	运动模块发生了与运动指令所规定内容不同的运动命令响应。	• 消除发生警报的原因。 • 如处于伺服 ON 未完状态，请将伺服设为 ON。 • 确认是否从其他程序同时向同一个轴发出了指令；如果是，请修改程序。
87h	VEL 的设定数据超出范围	在VEL指令中，执行了超出设定范围的指令。	修改 VEL 指令。
88h	INP 的设定数据超出范围	在INP命令中，执行了超出设定范围的指令。	修改 INP 指令。
89h	ACC/SCC/DCC 的设定数据范围外	在 ACC/SCC/DCC 命令中，执行了超出设定范围的指令。	修改 ACC/SCC/DCC 指令。
8Ah	MVT 指令中没有时间指定	在 MVT 指令中，T 指定为 0。	修改 MVT 指令。
8Bh	无法执行命令	发出了运动模块无法执行的运动指令。	修改运动程序。
8Ch	输出未完成	在运动模块未处于 ***Distribution Completed***（输出完毕）的状态下就执行了运动指令。	修改运动程序，以确保能在 ***Distribution Completed***（输出完毕）状态下执行运动指令。
8Dh	运动命令异常结束	运动模块变成了 ***Motion command abnormally aborted***（运动命令异常结束）状态。	• 消除轴的异常。 • 修改运动程序。

10. 3 顺序程序的故障诊断

10. 3. 1 故障确认流程

如果推断故障原因来自顺序程序，请参照以下流程排除故障。



10.3.2 程序启动

顺序程序启动存在异常时，请通过以下措施排除故障。

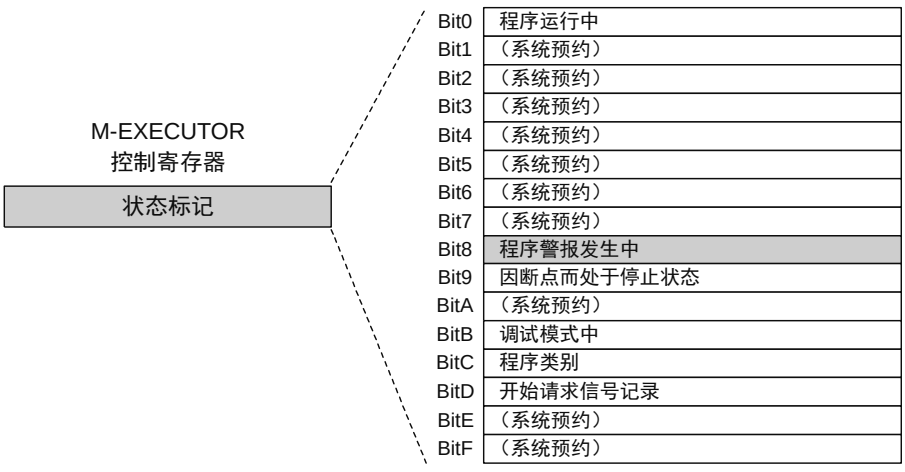
(a) 确认能够正常登录到系统。

顺序程序需预先登录到系统。
运行登录到系统后会登录到 M-EXECUTOR 模块。
关于登录到系统的方法说明，请参阅 “5.2.2 程序的登录 ”。

(b) 请确认状态 “程序警报发生中 ”。

状态 Bit8 “ 程序警报发生中 ” 为 ON 时，由于异常的原因导致顺序程序处于无法启动的状态。此时，请对以下项目进行确认。

- ◆调用程序是否存在。
- ◆调用程序是否是顺序程序。
- ◆是否使用顺序子程序调用 （SSEE）命令调用主程序。
- ◆指定的顺序程序是否超出 1 ～ 256 范围。
- ◆顺序子程序调用 （SSEE）的嵌套是否超过 8 层。



附录

附录 A 运动语言命令

以下为运动语言命令一览。各命令的详细情况，请参阅“8 章　相关命令”。

○	×
可使用	不可使用

A.1 轴设定命令

命令	名称	指令格式	概要	运动程序	顺序程序
ABS	绝对值模式	ABS ; 或 ABS MOV [逻辑轴 1] — [逻辑轴 2] —;	将后面的坐标作为绝对值处理。	○	×
INC	增量模式	INC ; 或 INC MOV [逻辑轴 1] — [逻辑轴 2] —;	将后面的坐标作为增量值处理。	○	×
ACC	加速时间变更	ACC [逻辑轴名称 1] 加速时间 [逻辑轴名称 2] 加速时间 [逻辑轴名称 3] 加速时间 ・・・;	更改定位类命令的加速时间。 最多可同时指定 16 个轴。	○	×
DCC	减速时间变更	DCC [逻辑轴名称 1] 减速时间 [逻辑轴名称 2] 减速时间 [逻辑轴名称 3] 减速时间 ・・・;	更改定位类命令的减速时间。 最多可同时指定 16 个轴。	○	×
SCC	更改 S 字时间常数	SCC [逻辑轴名称 1]S 字时间常数 [逻辑轴名称 2]S 字时间常数 ・・・;	设定移动平均滤波器的时间常数。 最多可同时指定 16 个轴。 滤波器与定位命令 / 插补命令均有关联。	○	×
VEL	更改进给速度	VEL [逻辑轴名称 1] 进给速度 [逻辑轴名称 2] 进给速度 [逻辑轴名称 3] 进给速度 ・・・;	设定定位类命令的速度。 最多可同时指定 16 个轴。	○	×
FMX	设定插补进给最快速度	FMX T 插补进给最高速度;	设定插补类命令的最高速度。 插补加减速时间，是指从 0 开始达到该速度的时间。	○	×
IFP	插补进给速度比率设定	IFP P 插补进给速度比率;	设定插补类命令的速度。 指定相对于最大速度的百分比。	○	×
IAC	插补加速时间更改	IAC T 插补加速时间;	更改插补类命令的加速时间。 指定从 0 到最大速度之间的加速时间。	○	×
IDC	插补减速时间更改	IDC T 插补减速时间;	更改插补类命令的减速时间。 指定从最大速度到速度为 0 之间的减速时间。	○	×

A.2 轴移动命令

命令	名称	指令格式		概要	运动程序	顺序程序
MOV	定位	MOV [逻辑轴名称 1] 指令位置 [逻辑轴名称 2] 指令位置 [逻辑轴名称 3] 指令位置 ・・・;		基于定位速度执行定位命令。最多可设定 16 个轴	○	×
MVS	直线插补	MVS [逻辑轴名称 1] 指令位置 [逻辑轴名称 2] 指令位置 [逻辑轴名称 3] 指令位置 ・・・ F 插补进给速度;		基于插补进给速度 F 执行直线移动。最多可设定 16 个轴。	○	×
MCW	圆弧插补（顺时针）	指定中心位置时	MCW [逻辑轴名称 1] 终点位置 [逻辑轴名称 2] 终点位置 U 中心位置 V 中心位置 T 转数 F 插补进给速度;	同时 2 轴沿半径 R（或指定中心坐标）的圆弧，按切向速度 F 执行插补。 指定中心坐标时，可通过 T －来指定多周圆。 （T －也可省略）	○	×
		指定半径时	MCW [逻辑轴名称 1] 终点位置 [逻辑轴名称 2] 终点位置 R 半径 F 插补进给速度;			
MCC	圆弧插补（逆时针）	指定中心位置时	MCC [逻辑轴名称 1] 终点位置 [逻辑轴名称 2] 终点位置 U 中心位置 V 中心位置 T 转数 F 插补进给速度;			
		指定半径时	MCC [逻辑轴名称 1] 终点位置 [逻辑轴名称 2] 终点位置 R 半径 F 插补进给速度;			

－接下页－

命令	名称	指令格式		概要	运动程序	顺序程序
MCW	螺旋插补（顺时针）	指定中心位置时	MCW［逻辑轴名称 1］终点位置 ［逻辑轴名称 2］终点位置 U 中心位置　V 中心位置 ［逻辑轴名称 3］直线插补的终点位置 T 转数　F 插补进给速度；	以合成了圆弧插补与圆弧插补平面外的直线插补的 3 轴同时移动的方式移动。速度 F 为 3 轴的切向速度。 指定中心坐标时，可通过 T －来指定转数。 （T －也可省略）	○	×
		指定半径时	MCW［逻辑轴名称 1］终点位置 ［逻辑轴名称 2］终点位置 R 半径 ［逻辑轴名称 3］直线插补的终点位置 F 插补进给速度；			
MCC	螺旋插补（逆时针）	指定中心位置时	MCC［逻辑轴名称 1］终点位置 ［逻辑轴名称 2］终点位置 U 中心位置　V 中心位置 ［逻辑轴名称 3］直线插补的终点位置 T 转数　F 插补进给速度；			
		指定半径时	MCC［逻辑轴名称 1］终点位置 ［逻辑轴名称 2］终点位置 R 半径 ［逻辑轴名称 3］直线插补的终点位置 F 插补进给速度；			
ZRN	原点复归	ZRN［逻辑轴名称 1]0 ［逻辑轴名称 2]0 ［逻辑轴名称 3]0・・・；		复位到各轴的原点。	○	×
SKP	带跳过功能的直线插补	SKP［逻辑轴名称 1］指令位置 ［逻辑轴名称 2］指令位置 ［逻辑轴名称 3］指令位置 ・・・ F 插补进给速度 SS 跳过输入信号选择；		直线插补动作中，如果开启 SKIP 信号，则会跳过剩下的移动量，进入下一个程序段。	○	×
MVT	时间指定定位	MVT［逻辑轴名称 1］指令位置 ［逻辑轴名称 2］指令位置 ［逻辑轴名称 3］指令位置 ・・・ T 定位时间（ms）；		对进给速度进行调整，执行定位，以便在指定时间内完成移动动作。	○	×
EXM	外部定位	EXM［逻辑轴名称 1］指令位置 从 D 外部定位信号输入时开始的移动量；		在定位过程中，如果输入外部定位信号，则只会以 D- 指定的移动量（增量值）执行定位，然后执行下一个命令。	○	×

A.3 控制命令

命令	名称	指令格式	概要	运动程序	顺序程序
POS	更改当前值	POS［逻辑轴名称 1］要更改的坐标值 ［逻辑轴名称 2］要更改的坐标值 ・・・;	最多可同时将 16 轴的当前值更改成“所需的坐标值”。后面的移动指令会在这个新的坐标系上移动。	○	×
MVM	机械坐标指令	MVM MOV［逻辑轴名称 1］指令位置 ［逻辑轴名称 2］指令位置 ［逻辑轴名称 3］指令位置・・・ ;	在机械坐标系上移动至目标位置。机械坐标系是指可以自动完成原点复归的坐标系统。该坐标不受 POS 命令的影响。	○	×
PLN	坐标平面指定	PLN［逻辑轴名称 1（纵轴）］ ［逻辑轴名称 2（横轴）］;	平面指定会指定必要命令所需的坐标平面。	○	×
PLD	程序当前位置更新	PLD［逻辑轴名称 1］ ［逻辑轴名称 2］・・・;	通过手动介入等来更新轴变动后的程序当前位置。 最多可指定 16 轴。	○	×
PFN	到位（In position）检查	MVS［逻辑轴名称 1］一 ［逻辑轴名称 2］一・・・PFN; 或 MVS［逻辑轴名称 1］一 ［逻辑轴名称 2］一・・・; PFN［逻辑轴名称 1］ ［逻辑轴名称 2］; MVS［逻辑轴名称 1］一 ［逻辑轴名称 2］一・・・;	对于同一程序段内或上一个程序段的插补移动指令，在进入到位检查宽度后将进入下一个程序段。	○	×
INP	到位检查宽度设定	INP［逻辑轴名称 1］定位范围检测宽度 ［逻辑轴名称 2］定位范围检测宽度 ・・・;	设定定位范围宽度。后面以指令形式发出的 PFN 命令相伴随的插补移动指令，进入到位检查宽度后会跳至下一个程序段。	○	×

A.4 程序控制命令

命令	名称	指令格式	概要	运动程序	顺序程序
IF ELSE IEND	分支	IF （条件式）； （处理 1）； ELSE; （处理 2）； IEND;	满足条件式时，执行处理 1， 不满足时，执行处理 2。	○	○
WHILE WEND	重复	WHILE （条件式）； ・・・； WEND;	满足条件时，重复执行 WHILE ～ WEND 处理。	○	○
PFORK JOINTO PJOINT	同时执行	PFORK 标签 1， 标签 2， 标签 3・・・； 标签 1：处理 1； JOINTO 标签 X； 标签 2：处理 2； JOINTO 标签 X； 标签 3：处理 3； JOINTO 标签 X； 标签 X：PJOINT；	同时执行标签指定的程序段。 运行子程序时，标签最多只能指定 2 个。 同时执行处理中，END，RET 无法使用。	○	×
SFORK JOINTO SJOINT	选择执行	SFORK 条件式 1? 标签 1， 条件式 2? 标签 2， 条件式 3? 标签 3， 条件式 4? 标签 4； 标签 1：处理 1； JOINTO 标签 X； 标签 2：处理 2； JOINTO 标签 X； 标签 3：处理 3； JOINTO 标签 X； 标签 4：处理 4； JOINTO 标签 X； ・・・； 标签 X：SJOINT；	满足条件式 1 时，执行处理 1， 满足条件式 2 时，执行处理 2。	○	○
MSEE	运动程序调用	MSEE MPS □□□;	运行 MPS □□□子程序。	○	×
SSEE	顺序程序调用	SSEE SPS □□□;	运行 SPS □□□子程序。	×	○
UFC	调用运动程序的用户函数	UFC 用户函数名称 输入数据， 输入地址， 输出数据；	从运动程序中调用用户创建的函数。	○	×
FUNC	调用顺序程序的用户函数	FUNC 用户函数名称 输入数据， 输入地址， 输出数据；	从顺序程序中调用用户创建的函数。	×	○
END	程序退出	END;	退出程序。	○	○
RET	子程序退出	RET;	退出子程序。	○	○
TIM	等待时间	TIM T 一；	仅等待 T 指定的时间长度，然后进入到下一个程序段。	○	×
IOW	等待输入 / 输出变量	IOW MB 一 ==・・・;	停止运行运动程序，直到条件式成立。	○	×
EOX	1 次扫描等待	EOX;	该命令可对连续执行的顺序命令进行分割。 EOX 后的程序段会在下一次扫描中运行。	○	×
SNGD/ SNGE	单段无效（SNGD）/ 单段有效 （SNGE）	SNGD; ・・・; SNGE;	该命令可指定是否在调试中进行单步动作。	○	×

A.5 数值运算

命令	名称	指令格式	概要	运动程序	顺序程序
=	代入	（结果）=（运算式）	右式的运算结果代入左边。运算按从左到右的顺序进行 （无优先顺序）。	○	○
+	加法	MW ― = MW ― + MW ―;	执行整数 / 实数的加法。同时存在整数 / 实数时，作为实数进行运算。	○	○
-	减法	MW ― = MW ― - MW ―;	执行整数 / 实数的减法。同时存在整数 / 实数时，作为实数进行运算。	○	○
*	乘法	MW ― = MW ― * MW ―;	执行整数 / 实数的乘法。同时存在整数 / 实数时，作为实数进行运算。	○	○
/	除法	MW ― = MW ― / MW ―;	执行整数 / 实数的除法。同时存在整数 / 实数时，作为实数进行运算。	○	○
MOD	除余	MW ― = MW ― / MW ―; MW ― = MOD;	进行除法运算后执行下一个程序段时，MOD 命令会保存指定寄存器中的余数。	○	○

A.6 逻辑运算

命令	名称	指令格式	概要	运动程序	顺序程序
|	OR（逻辑或）	MB ― = MB ― | MB ―; MB ― = MB ― | 1; MW ― = MW ― | MW ―; MW ― = MW ― | 00FFH;	创建位 / 整数的逻辑或运算。	○	○
&	AND（逻辑与）	MB ― = MB ― & MB ―; MB ― = MB ― & 1; MW ― = MW ― & MW ―; MW ― = MW ― & 00FFH;	创建位 / 整数的逻辑与运算。	○	○
^	XOR（异或）	MW ― = MW ― ^ MW ―; MW ― = MW ― ^ 00FFH;	创建整数的异或运算。	○	○
!	NOT（逻辑非）	MB ― = !MB ―; MB ― = !1; MW ― = !MW ―; MW ― = !00FFH;	创建位或整数的逻辑非运算。	○	○

A.7 数值比较

命令	名称	指令格式	概要	运动程序	顺序程序
==	一致	IF MW — == MW — ; WHILE MW — == MW — ;	用于 IF 或 WHILE 的条件式。左边与右边一致时，取 “ 真 ”。	○	○
<>	不一致	IF MW — <> MW — ; WHILE MW — <> MW — ;	用于 IF 或 WHILE 的条件式。左边与右边不一致时，取 “ 真 ”。	○	○
>	大于	IF MW — > MW — ; WHILE MW — > MW — ;	用于 IF 或 WHILE 的条件式。左边大于右边时，取 “ 真 ”。	○	○
<	小于	IF MW — < MW — ; WHILE MW — < MW —	用于 IF 或 WHILE 的条件式。左边小于右边时，取 “ 真 ”。	○	○
>=	大于等于	IF MW — >= MW — ; WHILE MW — >= MW — ;	用于 IF 或 WHILE 的条件式。左边大于等于右边时，取 “ 真 ”。	○	○
<=	小于等于	IF MW — <= MW — ; WHILE MW — <= MW — ;	用于 IF 或 WHILE 的条件式。左边小于等于右边时，取 “ 真 ”。	○	○

A.8 数据操作

命令	名称	指令格式	概要	运动程序	顺序程序
SFR	位右移	SFR MB — N — W — ;	Bit 变量向右移动指定数值。	○	○
SFL	位左移	SFL MB — N — W — ;	Bit 变量向左移动指定数值。	○	○
BLK	段传输	BLK MW — MW — W — ;	从指定的传输源作为起始，将指定程序段数量的相应区域复制到指定目标。	○	○
CLR	清除	CLR MW — W — ;	将目标寄存器作为起始，将指定范围的区域清零。	○	○
ASCII	ASCII 转换 1	ASCII “ 字符串 ” MW — ;	将指定的字符串转换成 ASCII 码，存放到目标寄存器。	○	○

A.9 基本函数

命令	名称	指令格式	概要	运动程序	顺序程序
SIN	正弦	SIN (MW ー); SIN (90);	计算整数或实数的正弦值。 整数型指定与实数型的指定规格不同。	○	○
COS	余弦	COS (MW ー); COS (90);	计算整数或实数的余弦值。 整数型指定与实数型的指定规格不同。	○	○
TAN	正切	TAN (MF ー); TAN (45.0);	计算实数的正切值。 只能指定实数型。	○	○
ASN	反正弦	ASN (MF ー); ASN (90.0);	计算实数的反正弦值。 只能指定实数型。	○	○
ACS	反余弦	ACS (MF ー); ACS (90.0);	计算实数的反余弦值。 只能指定实数型。	○	○
ATN	反正切	ATN (MW ー); ATN (45);	计算实数的反正切值。 整数型指定与实数型指定的规格不同。	○	○
SQT	平方根	SQT (MW ー); SQT (100);	计算整数或实数的平方根。 整数型指定与实数型指定的规格不同。	○	○
BIN	BCD → BIN	BIN (MW ー);	将 BCD 数据转换成 BIN 数据。	○	○
BCD	BIN → BCD	BCD (MW ー);	将 BIN 数据转换成 BCD 数据。	○	○
S{ }	指定位 ON	S {MB ー} = MB ー & MB ー;	如果逻辑运算结果为 “ 真 ”，则启用指定位。即使逻辑运算结果为 “ 假 ”，也不会关闭指定位。	○	○
R{ }	指定位 OFF	R {MB ー} = MB ー & MB ー;	如果逻辑运算结果为 “ 真 ”，则停用指定位。即使逻辑运算结果为 “ 假 ”，也不会启用指定位。	○	○
PON	上升脉冲	MB ー = PON (MB ー MB ー); 或 IF PON (MB ー MB ー) == 1; ・・・; IEND;	位输入的状态从 OFF 变化为 ON 时，位输出在 1 次扫描期间保持 ON。	×	○
NON	下降脉冲	MB ー = NON (MB ー MB ー); 或 IF NON (MB ー MB ー) == 1; ・・・; IEND;	位输入的状态从 ON 变化为 OFF 时，位输出在 1 次扫描期间保持 ON。	×	○
TON	接通延时定时器	MB ー = MB ー & TON (ー MB ー);	位输入为 ON 时，执行时间计数。 “ 计数值 = 设定值 ” 时，位输出为 ON。 计数时间的单位为 10ms。	×	○
TOF	断开延时定时器	MB ー = MB ー & TOF (ー MB ー);	位输入为 OFF 时，执行时间计数。 “ 计数值 = 设定值 ” 时，位输出为 OFF。 计数时间的单位为 10ms。	×	○

A.10 C 语言控制命令

命令	名称	指令格式	概要	运动程序	顺序程序
CTSK	C 语言任务控制	CTSK EXECUTE TYPE, C_NAME, COMPLETE ERROR ERR_CODE;	控制用户 C 语言任务的启动及停止等动作。	○	○
CFUNC	C 语言函数调用	CFUNC EXECUTE OPTION1 OPTION2, C_NAME C_ARG1 C_ARG2, COMPLETE ERROR C_RETURN;	执行用户 C 语言函数的调用。	○	○

附录 B 示例程序

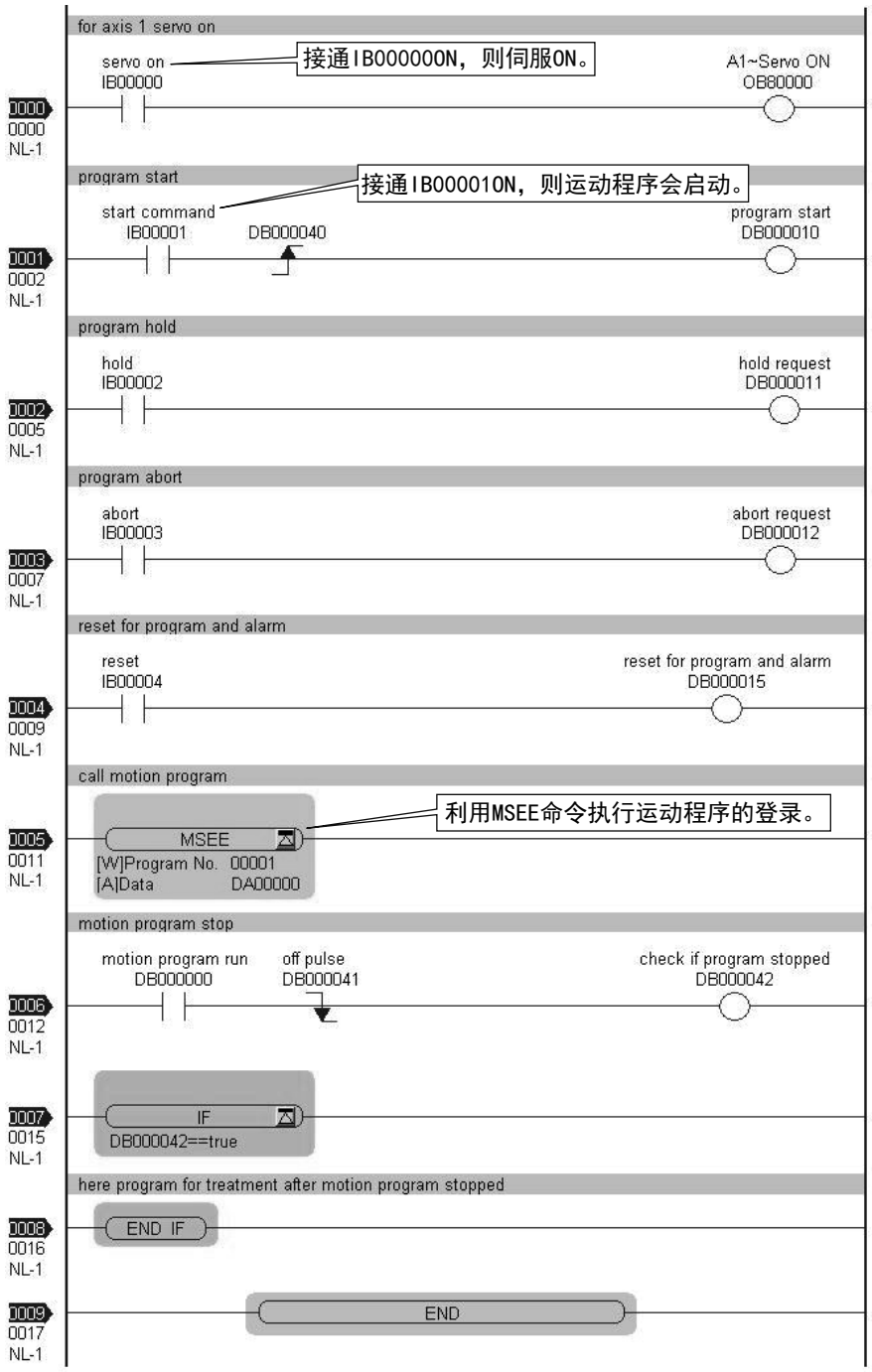
示例程序一览

示例程序	处理内容	参阅项目
运动程序控制用程序	用于运行控制运动程序的示例程序。记载有梯形图程序和顺序程序的示例。	B.1
同时处理	使用 PFORK 命令进行同时处理的示例程序。	B.2
通过运动程序执行速度控制	使用运动程序发出速度控制指令的示例程序。	B.3
使用虚拟轴的简单同步运行	使用 SVR 模块，实现 2 轴同步运行的示例程序。	B.4
顺序程序	使用顺序程序，对单轴伺服电机执行 JOG 及 STEP 动作的示例程序。	B.5

B.1 运动程序控制用程序

用于运行控制运动程序的示例程序。以下分别是使用了梯形图程序的例子和使用了顺序程序的示例。

■ 梯形图程序



■ 顺序程序

```
"----------------------------------------
"   伺服ON指令
"----------------------------------------
OB80000 = IB00000;                          “单轴伺服ON

"----------------------------------------
"   控制信号
"----------------------------------------
DB000010 = PON(IB00001 MB000000);           “程序开始
DB000011 = IB00002;                         “暂停
DB000012 = IB00003;                         “程序停止
DB000015 = IB00004;                         “警报复位

"----------------------------------------
"   运动程序运行停止
"----------------------------------------
IF NON(DB000000 MB000001) == 1;             “程序运行OFF?
;                                           “程序运行停止时的处理
IEND;

END;
```

接通IB00000，则伺服ON。

接通IB00001，则运动程序起动。

重要

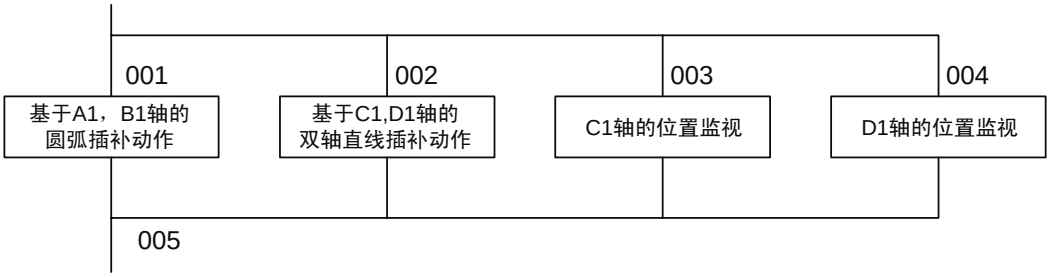
无法将 MSEE 命令加入顺序程序。请另外将 MSEE 命令加入 DWG.H。

B. 2 同时处理

以下为运动程序中，使用 PFORK 命令进行同时处理的示例程序。

```
    ABS;                                          “绝对值模式
    PFORK 001 002 003 004;                        开始同时处理
001:FMX T10000K;                                  “最高插补速度设定
    PLN [A1] [B1];                                “圆弧平面设定
    MCC [A1] 0 [B1]0 U100000 V0 F10000K;          “圆弧插补        处理1：执行圆弧插补
    JOINTO 005;
002:FMX T10000K;                                  “最高插补速度设定
    MVS [C1] 131072 [D1] 20000 F10000K;           “直线插补        处理2：执行直线插补
    JOINTO 005;
003:IOW IL8816>130000;                            “ C1轴位置监视
    OB00000=1;                                    “到达后，OB0000为ON  处理3：C1轴的位置监视
    JOINTO 005;
004:IOW IL8896>12000;                             “ D1轴位置监视
    OB00001=1;                                    “到达后，OB00001为ON  处理4：D1轴的位置监视
    JOINTO 005;
005:PJOINT;
    END;
```

以下是示例程序的动作内容。

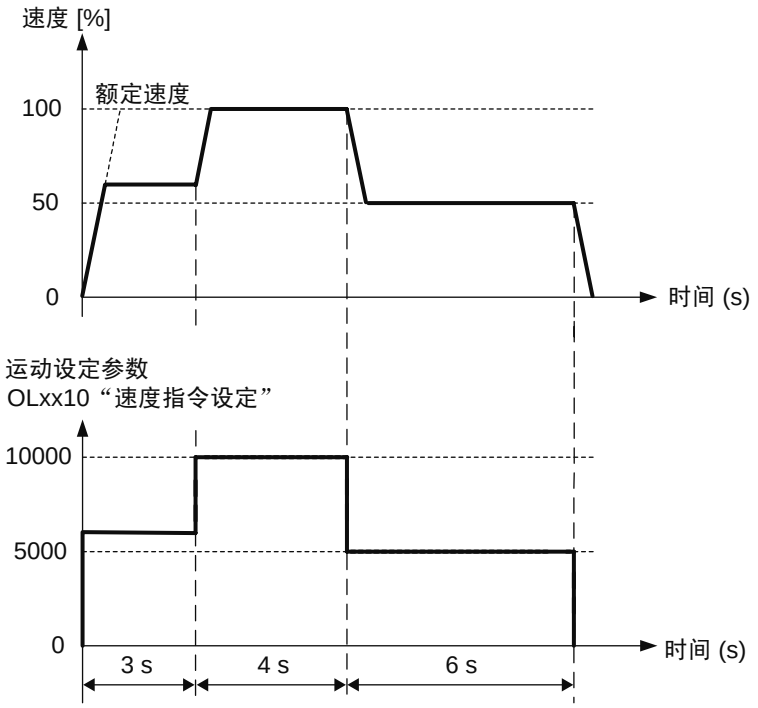


B.3 通过运动程序执行速度控制

以下为通过运动程序执行速度控制的示例程序。
本例将运动设定参数 OWxx03 Bit0 ～ 3“ 速度单位选择 ” 设定为 0.01% （指定额定速度的比例）。

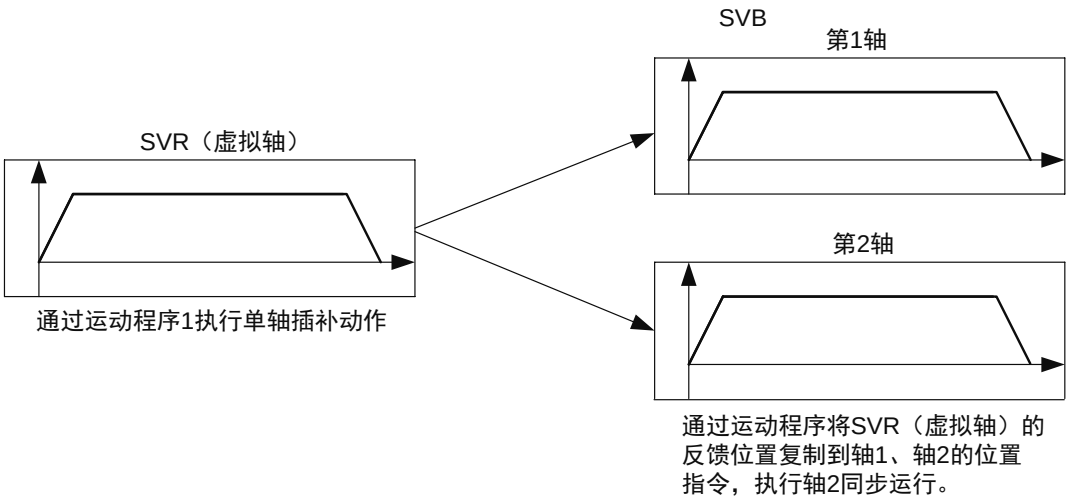
```
OW8008=23;      “速度控制模式
OL8010=6000;    “速度更改  额定速度的 60%
TIM T300;       “等待 3 秒
OL8010=10000;   “速度更改 额定速度
TIM T400;       “ 等待 4 秒
OL8010=5000;    “速度更改  额定速度的 50%
TIM T600;       “等待 6 秒
OW8008=0;       “速度控制模式停止
END;
```

以下是示例程序的动作模式。



B.4 使用虚拟轴的简单同步运行

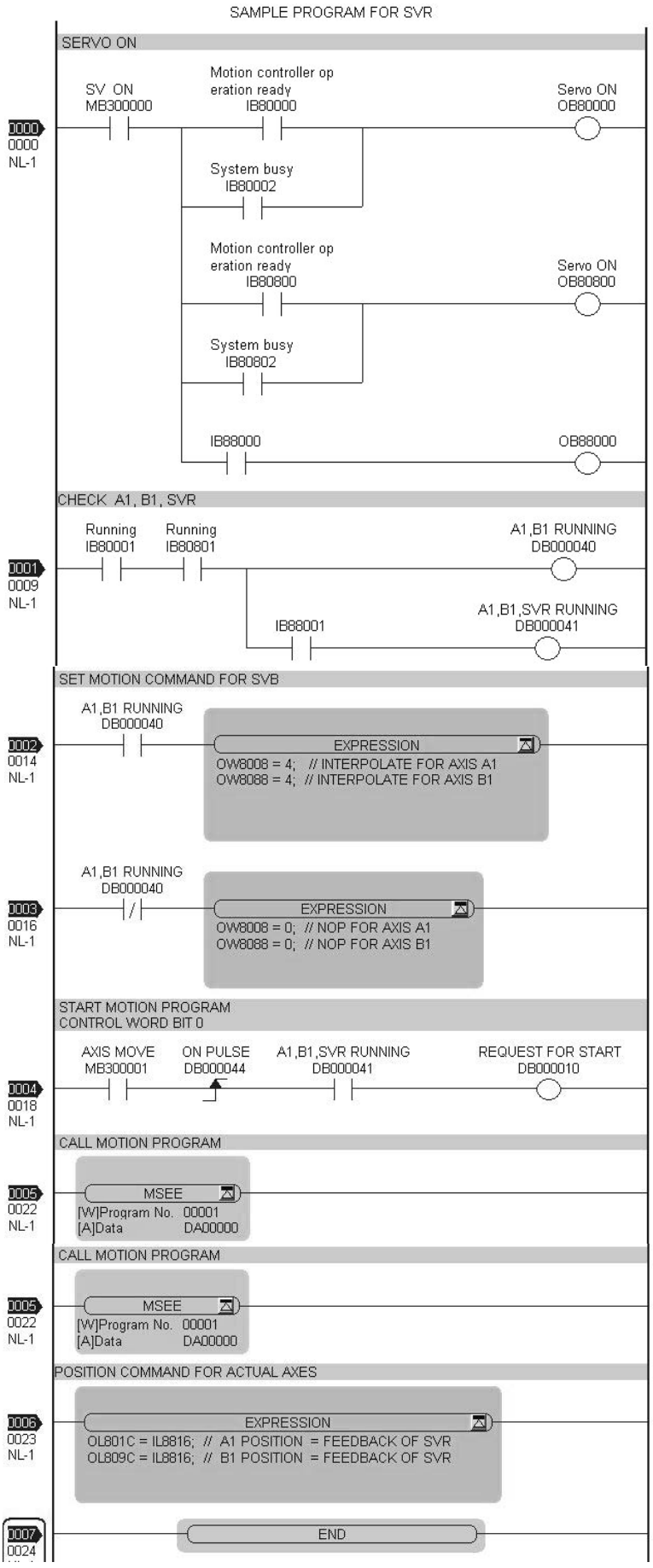
以下是通过运动程序驱动 SVR （虚拟轴），将 SVR （虚拟轴）的反馈位置分配到 2 个实轴，执行 2 轴同步运行的示例程序。



■ 运动程序

```
FMX T10000K;                  “最高插补速度设定  K=1000
INC;                          “增量模式
IAC T500;                     “插补加速时间 =500ms
IDC T500;                     “插补减速时间 =500ms
MVS [SVR] 1000K F10000K;      “移动距离 1000000 的插补动作
END;
```

B. 4 使用虚拟轴的简单同步运行



B.5 顺序程序

以下是使用顺序程序，对单轴伺服电机执行 JOG 及 STEP 动作的示例程序。

顺序主程序（SPM001）

```
“SPM001：主程序 ”

SSEE SPS002;              “ 轴通用设定处理
SSEE SPS003;              “JOG & STEP 动作处理
END;
```

顺序子程序（SPS002）

```
“SPS002：轴通用设定处理 ”

“----------------------------------
  运动命令 0 检测
“----------------------------------
IF IW8008 == 0;
        MB300010 = 1;
ELSE;
        MB300010 = 0;
IEND;

“----------------------------------
“  伺服 ON 指令
“----------------------------------
OB80000 = MB300000 & (IB80000 | IB80002);          “ 伺服 ON

“----------------------------------
“  警报复位
“----------------------------------
OB8000F = MB300001;                                “ 警报复位

“-------------------------------------------------------------------
“  速度单位＆加减速度单位 选择
“
“    Bit0 ～ 3：速度单位选择 (0：指令单位 /s, 1：指令单位 /min, 2：% 指定 )
“    Bit4 ～ 7：加减速度单位选择 (0：指令单位 /s^2, 1：ms 单位 )
“-------------------------------------------------------------------
DW00010 = OW8003 & FF00H;                          “ 功能设定 1 工件
OW8003 = DW00010 | 0011H;                          “ 功能设定 1

“----------------------------------
“  直线加速度 · 减速度设定
“----------------------------------
IF MB300020 == 1;
        OL8036 = 100;                              “ 直线加速度 / 加速时间常数
        OL8038 = 100;                              “ 直线减速度 / 减速时间常数
IEND;

RET;
```

接通 MB300000，则伺服 ON。

顺序子程序（SPS003）

```
“SPS003:JOG &STEP 动作处理”

“----------------------------------------
“  JOG动作
“----------------------------------------
IF IB80001 & ( (DB000010 & !DB000011) | (!DB000010 & DB000011) ) == 1;
        DB000000 = 1;
ELSE;
        DB000000 = 0;
IEND;

DB000001 = PON( DB000000 DB000050 ) & MB300010;    “JOG开始
DB000002 = NON( DB000000 DB000051 );               “JOG停止

IF DB000001 == 1;
        OL8010 = 1000;
        OW8008 = 7;                                “运动命令FEED
IEND;
IF DB000002 == 1;
        OW8008 = 0;                                “运动命令NOP
IEND;

“----------------------------------------
“  STEP动作
“----------------------------------------
IF IB80001 & ( (DB000012 & !DB000013) | (!DB000012 & DB000013) ) == 1;
        DB000008 = 1;
ELSE;
        DB000008 = 0;
IEND;

DB000009 = PON( DB000008 DB000058 ) & MB300010;    “STEP开始
DB00000A = NON( DB000008 DB000059 );               “STEP开始

IF DB000009 == 1;
        OL8010 = 1000;                             “STEP速度设定
        OW8044 = 1000;                             “STEP移动量设定(1000pulse)
        OW8008 = 8;                                “运动命令STEP
IEND;

IF DB00000A == 1;
        OW8008 = 0;                                “运动命令NOP
IEND;

“----------------------------------------
“  反转选择
“----------------------------------------
OB80092 = ( DB000000 & DB000011 ) | ( DB000008 & DB000013 );    “反转选择

RET;
```

接通DB000010，则开始JOG动作（正转）。

接通DB000011，则开始JOG动作（反转）。

接通DB000012，则开始STEP动作（正转）。

接通DB000013，则开始STEP动作（反转）。

附录 C MP900 系列与 MP2000 系列的区别

本节列出了与运动程序相关的 MP900 系列与 MP2000 系列之间的区别。

C.1 运动程序

项目	MP900 系列	MP2000 系列	备注
任务数	无限制	16	同时可运行的程序数
分组定义 1 组最多控制轴数	MP910：28 轴 MP920：48 轴 MP930：14 轴 MP940：1 轴	16 轴	—
梯形 MSEE 命令的任务大小	2 字	4 字	参见 “4.3.3 任务寄存器 ”
插补超程	常时有效	选择有效 / 无效	参见 “4.3.3 任务寄存器 ”
插补超程值指定用寄存器	通过分组定义指定	MSEE 任务的第 3 个字	参见 “4.3.3 任务寄存器 ”
程序运行登录功能 （M-EXECUTOR 模块）	无	有	MP2100、MP2100M、MP2300S、MP2310、MP2400 可使用
PFORK 的嵌套 （同时处理中的 PFORK 执行）	允许	禁止	—
子程序第 2 列的移动命令	禁止	允许	—
从子程序第 2 列的子程序调用 (MSEE)	允许	禁止	—
将实数存放到整数寄存器中时小数点后面的处理	四舍五入	选择舍去 / 四舍五入	MP2000 系列默认为舍去

C.2 顺序程序

项目	MP900 系列	MP2000 系列	备注
可否使用	不可使用	可使用	MP2100、MP2100M、MP2300S、MP2310、MP2400 可使用

C.3 运动程序命令

项目	MP900 系列	MP2000 系列	备注
可否执行 ACC、DCC	• SVA-01、SVA-02、SVB-01 可以执行 • PO-01 不可执行	可执行	—
可否执行 SCC	• SVB-01 可以执行 • SVA-01、SVA-02、PO-01 不可执行	可执行	—
VEL 的速度单位	10^n 指令单位 /min	从 4 个速度单位中选择 • 10^n 指令单位 /min • 指令单位 /s • 0.01% • 0.0001%	• 取决于 MP900 系列与 MP2000 系列的运动模块功能差异 • 参见 “8.1.6 进给速度变更 （VEL）”
指令单位选择＝ pulse 中的 VEL 的速度单位 (10^n 指令单位 /min)	【MP920 PO-01】 100pulse/min 【MP920 PO-01 以外】 1000pulse/min	1000pulse/min	• 取决于 MP900 系列与 MP2000 系列的运动模块功能差异 • 参见 “8.1.6 进给速度变更 （VEL）”
ACC,DCC 的加减速度单位	【MP930 SVB】 相对于进给速度的加减速时间 [ms] 【MP930 SVB 以外】 相对于额定速度的加减速时间 [ms]	从 2 个加减速度单位中选择 • 相对于额定速度的加减速时间 [ms] • 加减速度 [指令单位 /s^2]	• 取决于 MP900 系列与 MP2000 系列的运动模块功能差异 • 参见 “8.1.3 加速时间变更 （ACC）”， “8.1.4 减速时间变更 （DCC）”
EXM 的外部定位移动量	以 pulse 单位进行指定	以指令单位进行指定	取决于 MP900 系列与 MP2000 系列的运动模块功能差异

C.4 分组定义

项目	MP900 系列	MP2000 系列	备注
在线的分组定义保存	禁止	允许	—
自动生成梯形图	有 （通过组定义指定有无创建）	无	—

C.5 调试运行

项目	MP900 系列	MP2000 系列	备注
可停止段的命令	仅移动命令	全部命令	—
忽略程序段停止的方法	在要忽略的各程序段中编写 SNG	在要忽略程序段的区间执行 SNGD，SNDE	参见“8.4.14 单段无效（SNGD）/ 单段有效（SNGE）”
调试运行断点个数	1	4	—

C.6 运动程序警报

项目	MP900 系列	MP2000 系列	备注
运动程序警报存放位置	通过分组定义指定存放位置	存放到 S 寄存器	参见“10.2.3 警报代码确认方法”

附录 D 注意事项

D.1 常规注意事项

(1) 请在更改应用程序时，执行闪存保存

更改运动程序、顺序程序及应用程序时，请执行闪存保存。如在未执行闪存保存的情况下切断控制器电源，则更改后的应用程序会全部丢失。

(2) 请勿在工作状态下调试运行系统

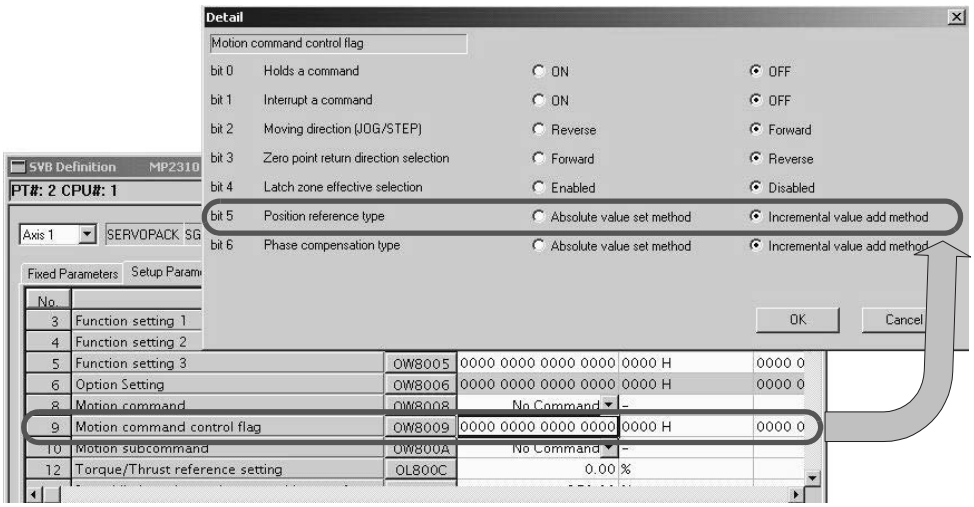
请勿在工作状态下对系统进行调试运行。否则可能会造成定时等发生变化，导致运行异常及装置故障。

进行调试运行时，请使用调试专用的系统。

D.2 运动参数相关注意事项

(1) 请将运动设定参数 OWxx09 Bit5“位置指令型” 设定成增量值叠加方式

使用运动程序时，请将运动设定参数 OWxx09 Bit5“位置指令型” 设定成增量值叠加方式。这是因为运动程序以增量值叠加方式执行位置信息管理。指定了绝对值指令方式后，无法保证运动程序的运行。



运动程序在切换绝对值指令模式与增量值指令模式时使用专用命令 （ABS/INC 命令）。
如发出 ABS 命令，则会被设定成绝对值模式 （ABS 模式）。
如发出 INC 命令，则会被设定成增量模式 （INC 模式）。

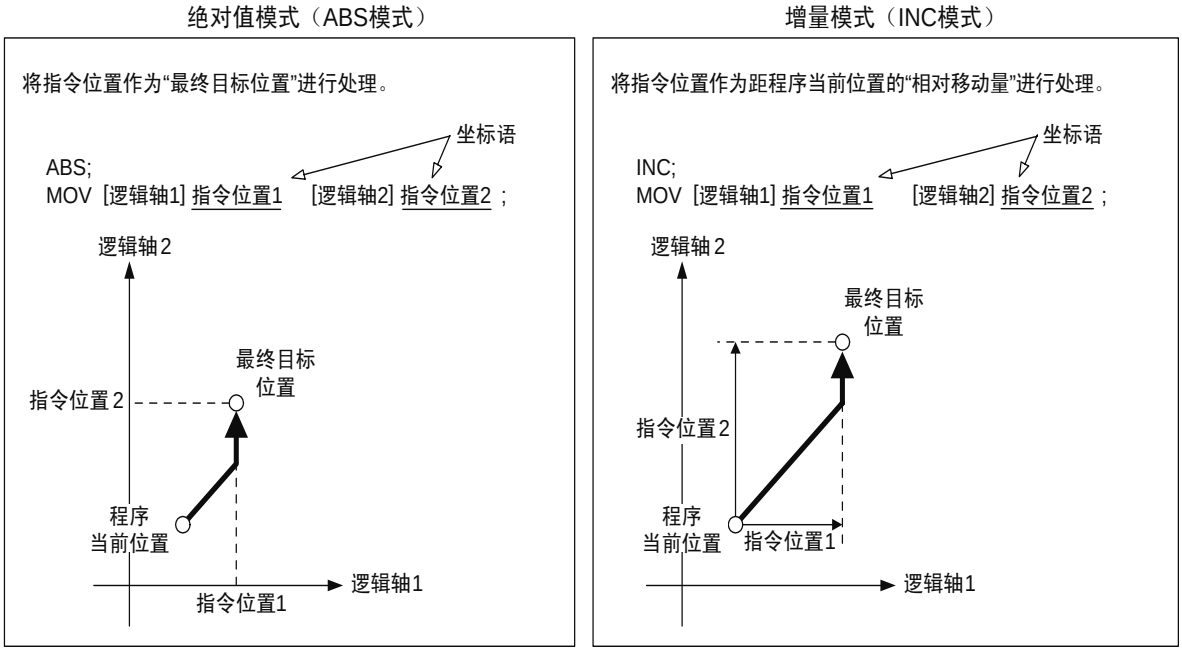
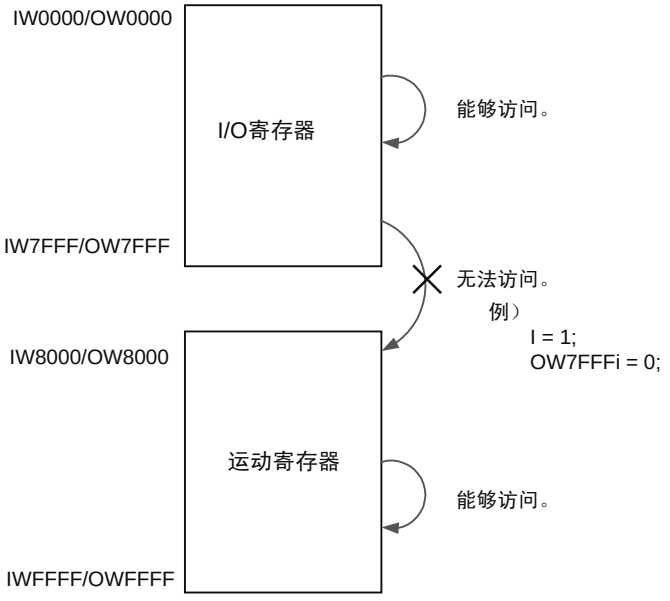


图 D.1 轴移动命令的移动模式

(2) 请勿使用下标从 I/O 寄存器引用运动寄存器

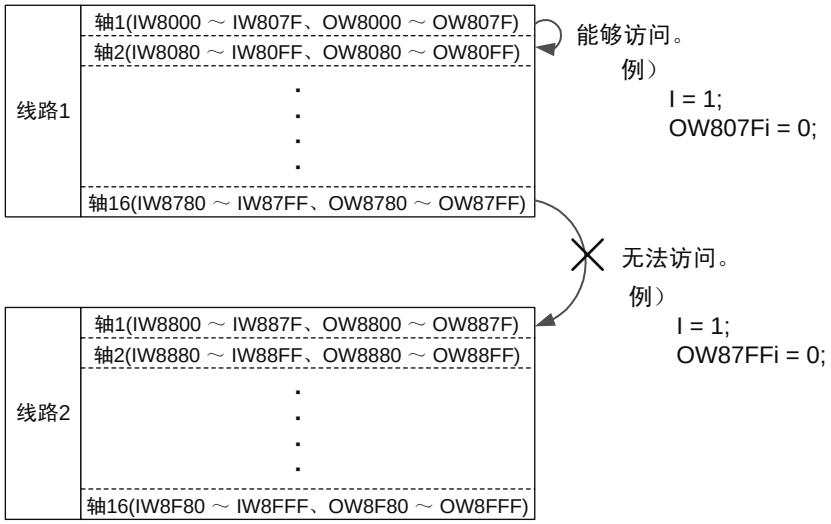
I/O 寄存器与运动寄存器不会被分配到连续的内存中。
使用下标时，请在各种寄存器范围内存取 I/O 寄存器或运动寄存器。



(3) 请勿使用下标访问其它线路的运动寄存器

与 I/O 寄存器和运动寄存器之间关系一样，线路不同的运动寄存器不会被分配到连续的内存中。
使用下标时，请存取各线路的运动寄存器范围内的寄存器。
如果是同一线路，则还可以访问跨轴的运动寄存器。

线路编号	1 轴	2 轴	・・・	16 轴
1	OW8000 ～ OW807F	OW8080 ～ OW80FF	・・・	OW8780 ～ OW87FF
2	OW8800 ～ OW887F	OW8880 ～ OW88FF	・・・	OW8F80 ～ OW8FFF
3	OW9000 ～ OW907F	OW9080 ～ OW90FF	・・・	OW9780 ～ OW97FF
4	OW9800 ～ OW987F	OW9880 ～ OW98FF	・・・	OW9F80 ～ OW9FFF
5	OWA000 ～ OWA07F	OWA080 ～ OWA0FF	・・・	OWA780 ～ OWA7FF
6	OWA800 ～ OWA87F	OWA880 ～ OWA8FF	・・・	OWAF80 ～ OWAFFF
7	OWB000 ～ OWB07F	OWB080 ～ OWB0FF	・・・	OWB780 ～ OWB7FF
8	OWB800 ～ OWB87F	OWB880 ～ OWB8FF	・・・	OWBF80 ～ OWBFFF
9	OWC000 ～ OWC07F	OWC080 ～ OWC0FF	・・・	OWC780 ～ OWC7FF
10	OWC800 ～ OWC87F	OWC880 ～ OWC8FF	・・・	OWCF80 ～ OWCFFF
11	OWD000 ～ OWD07F	OWD080 ～ OWD0FF	・・・	OWD780 ～ OWD7FF
12	OWD800 ～ OWD87F	OWD880 ～ OWD8FF	・・・	OWDF80 ～ OWDFFF
13	OWE000 ～ OWE07F	OWE080 ～ OWE0FF	・・・	OWE780 ～ OWE7FF
14	OWE800 ～ OWE87F	OWE880 ～ OWE8FF	・・・	OWEF80 ～ OWEFFF
15	OWF000 ～ OWF07F	OWF080 ～ OWF0FF	・・・	OWF780 ～ OWF7FF
16	OWF800 ～ OWF87F	OWF880 ～ OWF8FF	・・・	OWFF80 ～ OWFFFF



(4) 通过运动程序执行轴移动过程中，请勿更改运动设定参数 OLxx1C “ 位置指令设定 ”

在通过运动程序执行轴移动过程中，如果通过其它程序 （梯形图程序等）更改 OLxx1C “ 位置指令设定 ”，则轴会按照更改后的指令移动。结果是，通过运动程序指令的位置与实际的轴位置会出现偏差。

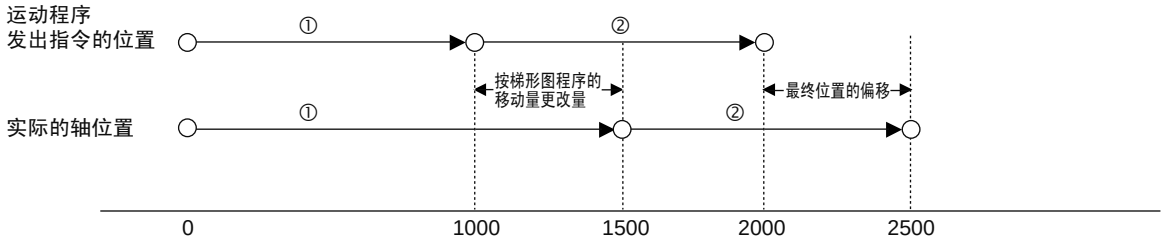
（例）

通过以下运动程序执行 ① 过程中，通过梯形图程序将 OLxx1C “ 位置指令设定 ” 的 A1 轴的移动量从 +1000 更改成 +1500，则 A1 轴会移动到 +1500 的位置。因而，与运动程序中的指令位置 （+1000）之间会产生偏差。之后，运动程序执行 ②。结果是，A1 轴的实际最终位置也不同于基于运动程序的指令位置。

```
INC;
ZRN [A1]0;

MOV [A1]1000;・・・①
MOV [A1]1000;・・・②

END;
```



索引

記号

数字

A

B

C

D

E

F

G

H

I

R

S

T

U

V

W

X

Y

Z

改版记录

资料的改版相关信息与资料编号一起记载于本资料封底的右下角。

资料编号：SICP C880700 38A
© Published in Japan 2014年 1月编制 14-1
└国家或地区 └发行日期 └第一版发行日期

发行日期	改版编号	项目编号	变更内容
2014 年 1 月	-	-	第一版发行 （基于日文说明书 SIJP C880700 38F<7>）

机器控制器 MP2000系列

用户手册
运动程序篇

客户服务热线(帮您解决技术问题)

电话 **400-821-3680**　传真 **021-5385-2008**

周一至周五(节假日除外)9:00～11:30，12:30～16:30　※24小时接收传真

销售

・安川電機(中国)有限公司
上海市湖滨路222号企业天地1号楼22楼
邮编: 200021
电话: 021-53852200
传真: 021-53853299

・安川電機(中国)有限公司　北京分公司
北京市东城区东长安街1号东方广场东方经贸城西三办公楼1011室
邮编: 100738
电话: 010-85184086
传真: 010-85184082

・安川電機(中国)有限公司　广州分公司
广州市天河区体育东路138号金利来数码网络大厦1108-10室
邮编: 510620
电话: 020-38780005
传真: 020-38780565

・安川電機(中国)有限公司　成都分公司
成都市总府路2号时代广场B座711室
邮编: 610016
电话: 028-86719370
传真: 028-86719371

总公司

・株式会社　安川電機
日本福岡県北九州市八幡西区黒崎城石2-1
邮编: 806-0064
电话: 0081-93-645-8800
传真: 0081-93-631-8837

YASKAWA

资料编号　SICP C880700 38A