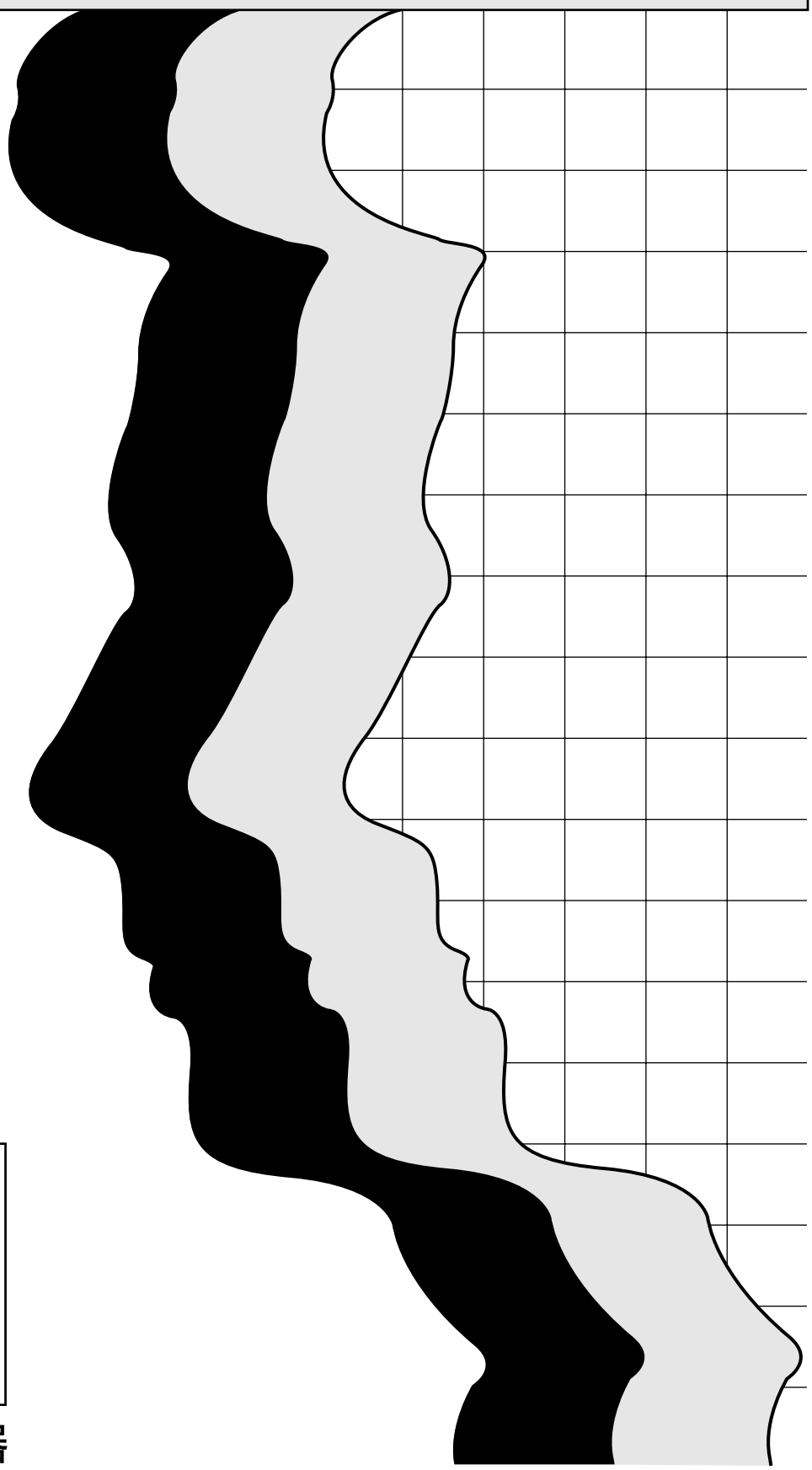
MITSUBISHI

QnA编程

参考手册



MELSEC

可编程控制器

安全警告

(在使用该产品之前必须阅读这些警告)

在使用 MELSEC-QnA 系列产品时，除了要认真地阅读这本手册以及在该手册中指定的相关的手册外，同时还要注意安全以及正确地使用该产品。

这里所陈述的安全警告是适用于该产品的独立的一部分。关于作为 PC 系统整体的安全信息，请参考 CPU 模块用户手册。

请小心地保管该手册以便随时参考，本手册的副本应该交给最终用户。

[系统设计注意事项]

⚠危险

- 安全电路应该安装在可编程控制器的外围，以确保即使在某一个外围供电电源发生故障或者出现可编程控制器错误时，系统作为一个整体仍能继续安全运行。错误的输出与操作可能会导致意外事故。
  1） 以下的电路应该安装在可编程控制器的外面：
  将紧急停止电路保护电路以及象前进/倒退等互逆操作的电路互锁，并将上升/下降位置限制等电路互锁，用以防止出现机械损伤。
  2） 当可编程控制器检测到不正常情况发生时，进程会停止，并且所有的输出会被关掉。这在以下几种情况下会发生：
  - 当供电源供电模块过流或过压时，保护软元件就会被激活。
  - 当 PC CPU 的自诊断功能检测到错误时。
  - 一些错误，比如输入/输出控制错误无法被 PC CPU 检测到，所以可能会存在当这些错误发生时所有的输出都被打开的情况。为了保证在这种情况下机器能够安全运行，就应该在可编程控制器外面提供一个故障安全电路或机械装置。这种故障安全电路的例子请参考 CPU 模块用户手册。

  3） 由于一个输出模块的延迟或者晶体管的故障，输出可能被停在 ON 或 OFF 上。这时就应该提供一个外围电路用来监视输出信号，这些输出信号引起的不正确的操作可能会导致严重的事故。
- 应该安装一个只有在可编程控制器的电源已经打开后，才允许外围电源打开的电路。如果外围电源先打开了，机器就有可能会发生错误的输出与动作而导致事故。
- 当有数据连接通讯错误发生时，以下所示的状态将在故障点被创建。为了确保系统在这些时候能够安全运行，应该在顺控程序中提供一个互锁电路（使用通讯状态信息）。

  错误的输出和操作会导致事故。
  1） 在错误发生之前的数据连接将会被保持。
  2） 在 MELSECNET（II，/B，/10）远程 I/O 站的所有输出将被关掉。
  3） 在 MELSECNET/MINI-S3 远程 I/O 站，所有的输出将被关掉或者输出状态被保持，这依赖于 E.C.模式的设置。

  关于当由这种错误发生时，检查错误站以及运行状态过程的细节，请参考相关的数据连接手册。

[系统设计注意事项]

◇小心
● 不要将控制线或通讯线与主电路或电源线捆扎在一起，或将这些线靠得太近。应该将这些线分开至少 100mm，否则由于噪声可能会发生故障。

[有关安装的警告]

◇小心
● PC 机应该在与本手册的一般规范相符合的环境中使用。 超出一般规范状态的环境中使用 PC 机，会引起电击、着火、故障或破坏/损坏产品。 ● 在安装模块之前,确定模块底座上的模块固定突出部分确实正确地装在基板上的固定孔里。 如果没有正确安装模块，将会导致故障或错误，或者会导致模块倒下。 ● 扩展电缆应该安全地连接在基板和模块连接器上。在安装完后应检查连接是否松动。 连接的不好会引起连接问题以及错误的输入/输出。 ● 将存储卡盒牢固地插入到存储卡盒安装连接器里。在安装完后应检查连接是否松动。 连接的不好会导致错误的操作。 ● 将存储器牢固地插入存储器插槽中。在安装完后应检查连接是否松动。 连接的不好会导致错误的操作。

[有关布线的警告]

◈危险
● 在开始安装和布线工作之前，关闭外围电源供电。 如果操作错误，就会导致电击和设备的损坏。 ● 安装与布线完成后，一定要在打开电源与开始运行之前安上终端的盖子。 如果操作错误，就会导致电击。

◈小心
● 一定要将 FG 和 LG 终端接地，除了 PC 机外至少要执行 3 级接地工作。、 否则，将会有发生电击和故障的危险。 ● 正确给 PC 机布线，检查产品的额定电压和终端布置。 使用与额定电压不相符的电源供电或者没有正确布线，将会引起失火或故障。 ● 多电源供电模块的输出不应并行连接。如果操作错误，会导致电源供电模块过热，引起失火或模块故障。 ● 拧紧终端螺丝钉。 螺丝钉松动将会导致短路、失火或故障。 ● 确定没有异物如芯片或线头等进入模块里。 它将引起失火、错误或故障。 ● 外围的连接器应该弯曲，压焊或者用正确的工具按照正确的方法焊接。 关于弯曲和压焊工具的详细资料，请参考输入/输出模块用户手册。 连接不好会导致短路、失火以及错误的操作。

[有关启动与维护的警告]

⚠危险
● 当电源打开的时候，不要触摸终端。这将引起故障。 ● 确定电池正确地连接。不要试图充电或卸下电池，不要加热电池或将它放在火里，并且不要短接或焊接电池。 不正确的处理电池将会引起电池发热以及破裂，这将会导致失火或人身伤害。 ● 在清洗或重新拧紧终端螺丝之前，一定要关闭电源。 在电源打开时进行这些工作，会引起失效或模块故障。

⚠小心
● 为了确保安全操作，仔细阅读这本手册，使你自己熟知运行中程序的变化过程、强制输出、运行、停止以及暂停操作等。 不正确的操作会导致机器失效以及人身伤害。 ● 不要拆卸或调整任何模块。 这将引起失效、故障、人身伤害或失火。 ● 在安装或卸下模块之前一定要将电源关闭。 在电源打开时，安装或拆卸模块会导致模块失效或故障。 ● 当更换保险丝的时候，一定要使用指定的保险丝。错误容量的保险丝会引起失火。

[处理警告]

⚠小心
● 将该产品作为工业废物进行处理。

版本

*手册的编号在封底的左下角给出。

印刷日期	*手册编号	版本
1997 年 6 月	SH(NA)-080229C-A	第一版

引言

感谢您选用三菱公司的 MELSEC-QnA 系列的通用可编程控制器。请您仔细阅读本手册，以便能最好地使用该设备。本手册的副本应该交给最终用户。

目录

关于手册

QnACPU 相关的手册列在下面的表格中。
请定购您所需要的。

相关的手册

手册名称	手册编号
QnACPU 指南 主要是给第一次使用 QnACPU 的人阅读的。描述了从创建程序和将已经创建的程序写到 CPU，到调试的所有过程。 同时也说明了怎样能够最有效地使用 QnACPU。	IB-66606 （13FJ10）
Q2A（S1）/Q3A/Q4ACPU 用户手册 说明了 Q2ACPU（S1）、Q3ACPU 以及 Q4ACPU 的使用方法，以及电源、存储卡和基板的操作和规格。 （个别购买）	IB-66608 （13J821）
Q4ARCPU 用户手册 说明了 Q4ARCPU 的使用方法，也叙述了总线切换模块、系统控制模块、电源供电模块、存储卡和基板的规格与操作。 （个别购买）	IB-666085 （13J852）
Q2AS（H）CPU（S1）用户手册 说明了 Q2ASCPU、Q2ASCPU-S1、Q2ASHCPU 和 Q2ASHCPU-S1 的使用方法，也叙述了电源供电模块、存储卡和基板的规格与操作。 （个别购买）	SH-3599 （BJ858）
QnACPU 编程手册（通用指令） 说明了如何使用顺控指令、基本指令和应用指令。 （个别购买）	IB-66615 （13JF47）
QnACPU 编程手册（特殊功能模块） 说明了当使用 Q2ACPU（S1）、Q3ACPU 和 Q4ACPU 时特殊功能模块的专用指令。 （个别购买）	IB-66616 （13JF48）
QnACPU 编程手册（AD57 指令） 说明了当使用 Q2ACPU（S1）、Q3ACPU 和 Q4ACPU 时对控制 AD57（S1）型 CRT 控制模块的专用指令。 （个别购买）	IB-66617 （13JF49）
QnACPU 编程手册（PID 控制指令） 说明了当使用 Q2ACPU（S1）、Q3ACPU 和 Q4ACPU 时，PID 控制的专用指令。 （个别购买）	IB-66618 （13JF50）
QnACPU 编程手册（SFC） 说明了 SFC 程序的性能规格、功能、编程、调试以及错误代码。 （个别购买）	IB-66619 （13JF51）
QnA/Q4AR MELSECNET/10 网络系统参考手册 说明了 MELSECNET/10 的一般概念、规格以及部件名称与设置。 （个别购买）	IB-66690 （13JF78）
MELSECNET、MELSECNET/10 数据连接系统参考手册 说明了 MELSECNET（II）、MELSECNET/B 的一般概念、规格以及部件名称与设置。 （个别购买）	IB-66350 （13JF70）
SW2IVD-GPPQ GPP 型功能操作手册（离线） 说明了使用 SW2IVD-GPPQ 时如何创建程序及打印输出数据，以及 SW2IVD-GPPQ 的离线功能，如文件维护。 （与产品一起提供）	IB-66774 （13J921）
SW2IVD-GPPQ GPP 型功能操作手册（在线） 说明了 SW2IVD-GPPQ 的在线功能，包括监视与调试的方法。 （与产品一起提供）	IB-66775 （13J922）
SW2IVD-GPPQ GPP 型功能操作手册（SFC） 说明了 SFC 功能，如 SFC 程序编辑与监视。（与产品一起提供）	IB-66776 （13J923）
SW2IVD-GPPQ GPP 型软件包操作手册（Q6TEL） 说明了系统配置、操作方法等。 （与产品一起提供）	IB-66777

1. 概述

本手册叙述了在为以下 MELSEC QnA 系列的 CPU 模块编程时所需要的程序类型、I/O 处理、软元件等。

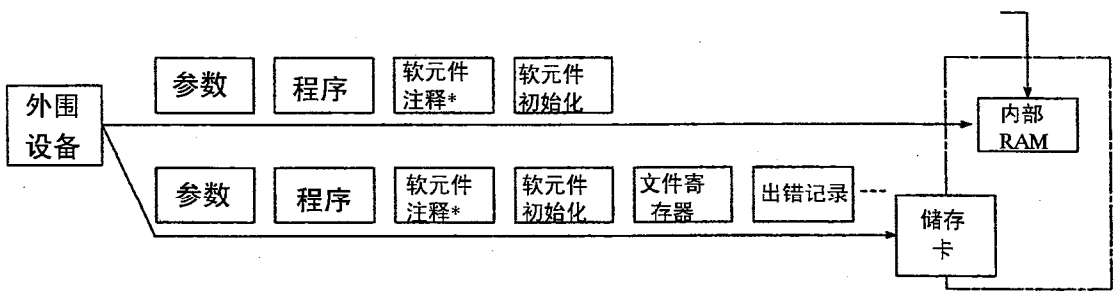
- Q2AS（H）CPU（S1）
- Q2ACPU（S1）
- Q3ACPU
- Q4ACPU
- Q4ARCPU

以上所有的模块在本手册中都被称为“QnACPU” 。
另外，本手册中提及的 GPP 功能软件包是指 SW. –GPPW 或者 SW. -GPPQ。

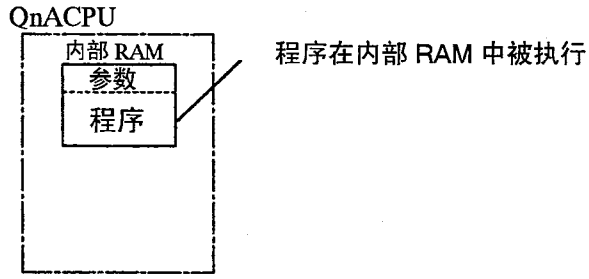
1.1 程序

(1) 用存储卡进行程序管理是可能的

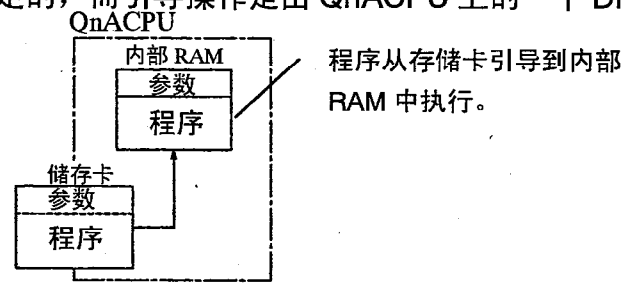
(a) 在外围设备创建的程序可以被存储到 QnACPU 的内部 RAM 中或存储卡上。参数、程序、软元件注释以及软元件初始化设置值都可以存储在内部 RAM 或存储卡中。其他的文件寄存器和缺省的故障履历数据等只能存储在存储卡上（在内部 RAM 中存储是不可能的，参见 2.4 章节）。注意，存储在内部 RAM 中的软元件注释（下图中的“*”）不能用在程序的指令中。



(a) QnACPU 执行存储在其内部 RAM 中的处理程序。

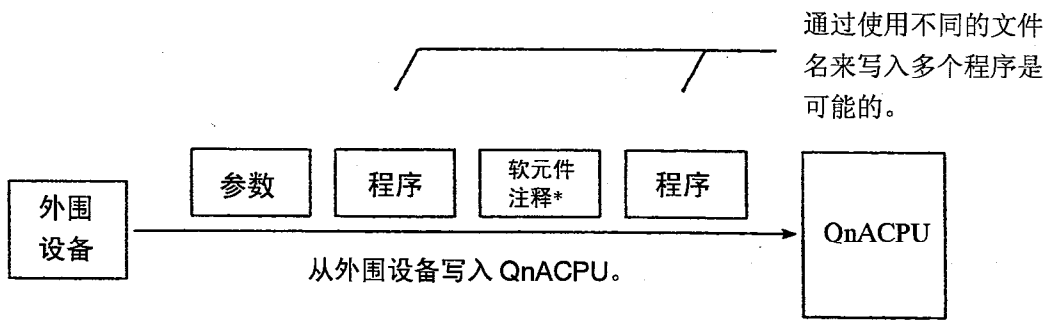


存储在存储卡上的程序只有在它们被读到（引导到）QnACPU 的内部 RAM 中后，才能被执行。（要被读到 QnACPU 中的程序是由参数设置指定的，而引导操作是由 QnACPU 上的一个 DIP 切换设置指定的。）



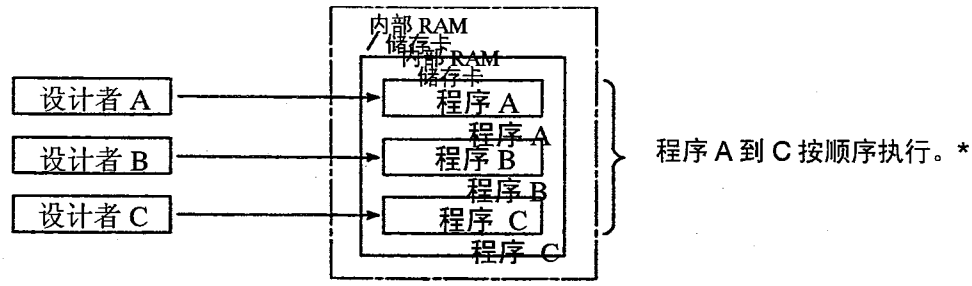
(1) 结构化的程序

QnACPU 程序是以文件的形式存储在内部 RAM 或存储卡中的。因此多个程序可以使用不同的文件名存储在内部 RAM 或存储卡中。

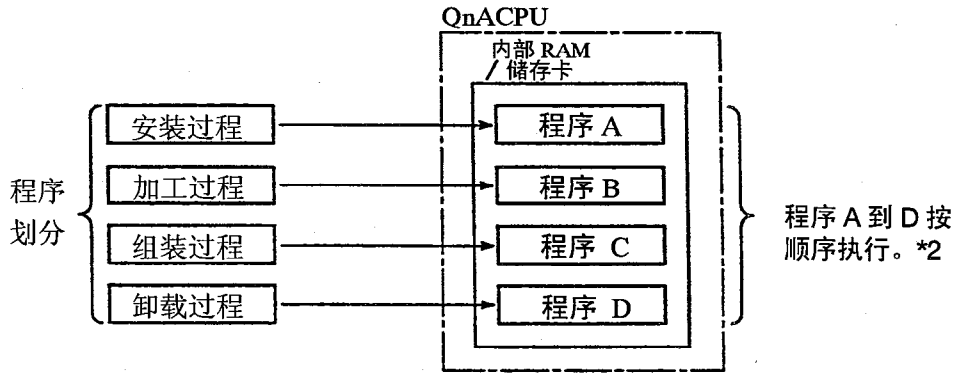


这种结构化的程序允许几个设计者对其分别创建，也允许根据正在处理的工序和功能来对其进行管理和维护。此外，当规格改变时，只需要对相关的程序进行修正和调试。

(a) 将程序的创建划分给几个设计者的例子：



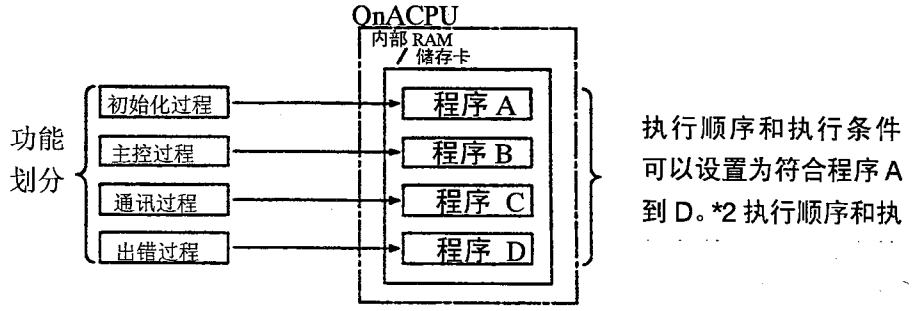
(b) 根据功能进行程序划分的例子：



注释

1. *：参见 3.2 节，了解执行顺序的细节。

(c) 根据工序进行程序划分的例子：



注释

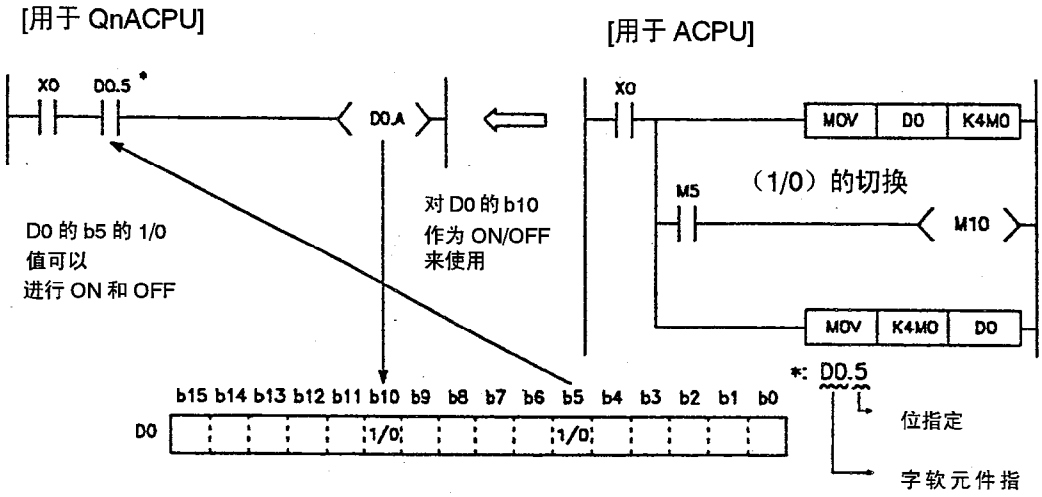
1. *1：根据工序进行划分的程序还可以根据功能进一步划分。
2. *2：参见 3.2 节，了解执行顺序和执行条件的细节。

1.2 方便编程的软元件和指令

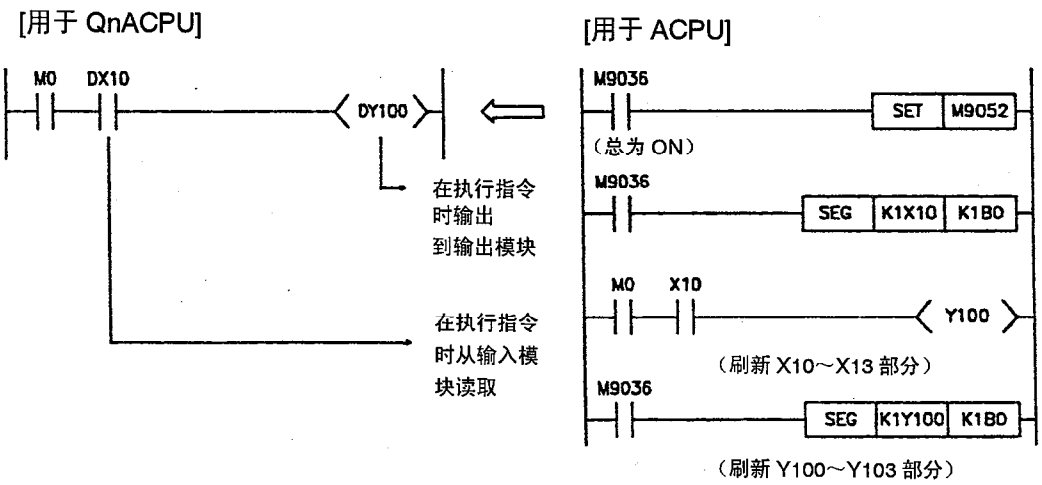
QnACPU 的特点是其软元件和指令使程序的创建变得很方便、容易。以下以几个例子予以说明。

(1) 灵活的软元件指定

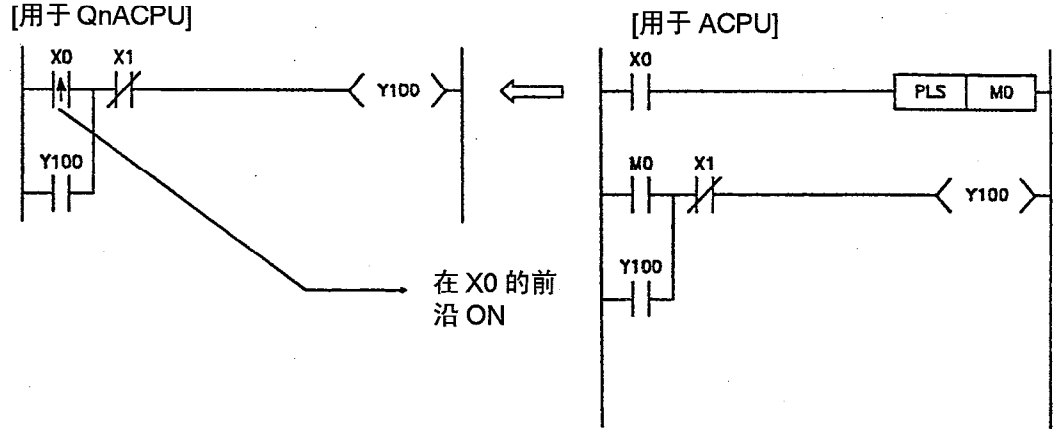
(a)字软元件位可以被指定作为触点或线圈。



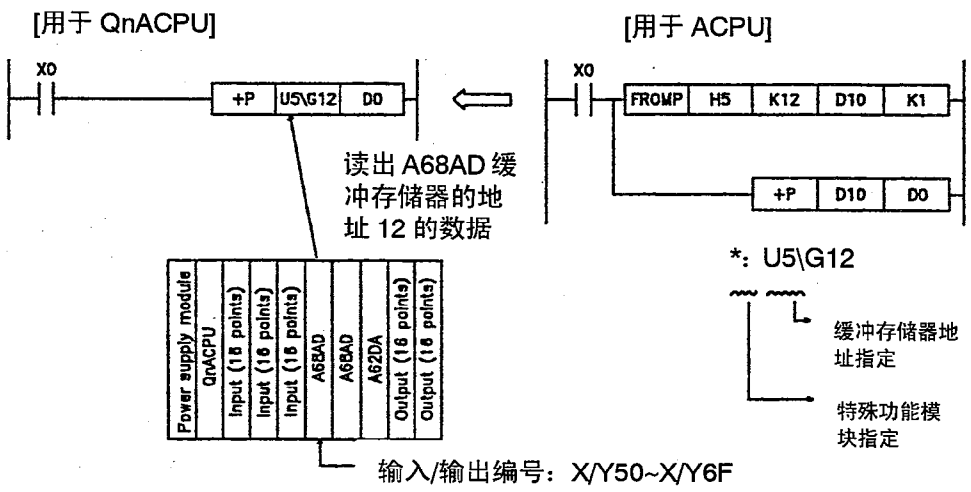
(b) 在一个程序里只使用直接存取输入(DX[])和直接存取输出(DY[])就可对一点单位进行直接处理。



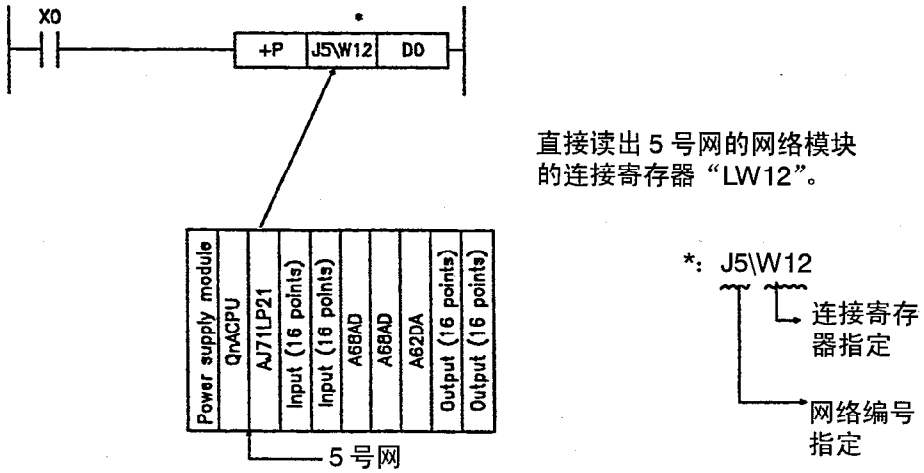
(c) 微分触点（ ）避免了将输入转换成脉冲的操作。



(d) 特殊功能模块的缓冲存储器在编程时可以作为软元件被使用。



(e) 可以对 MELSECNET/10 网络模块的连接软元件（LX，LY，LW，LSB，LSW）进行直接存取而无需刷新设置。

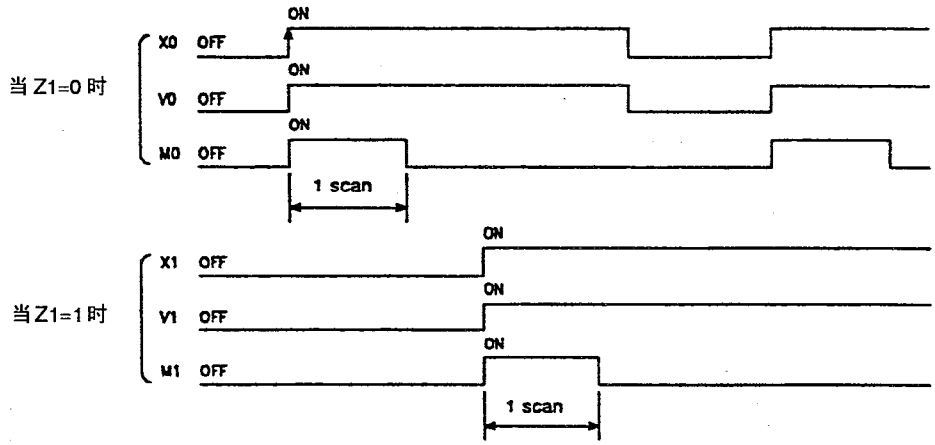


(2) 边缘继电器简化脉冲变换处理

(a) 使用在输入条件的上升沿打开的边缘继电器（V）来简化触点变址的脉冲处理。

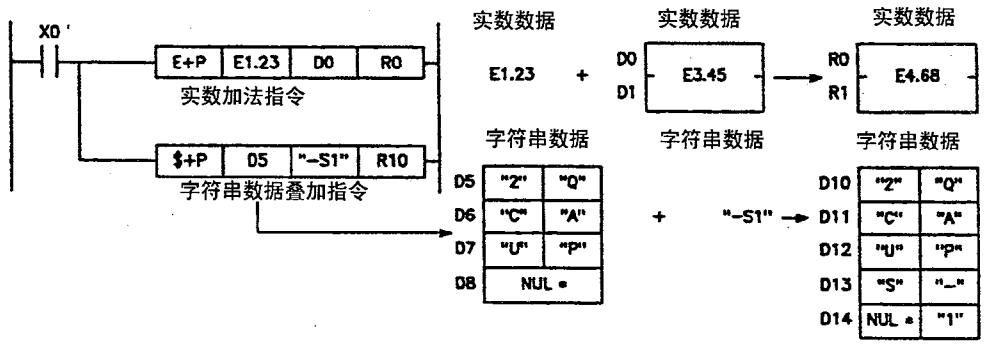
[电路的例子]

[时序图]



(3) 更简单的数据处理

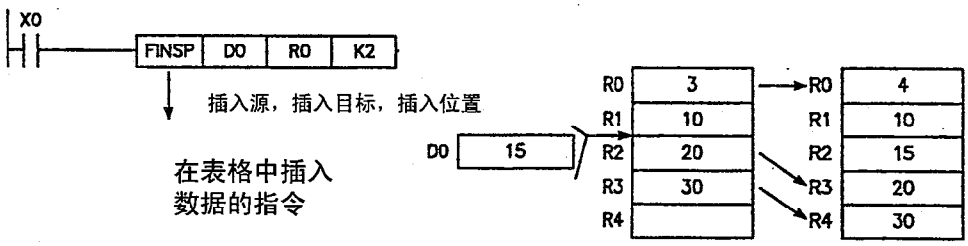
(a) 实数（浮点数据）和字符串常数可以直接用在编程中。



注释

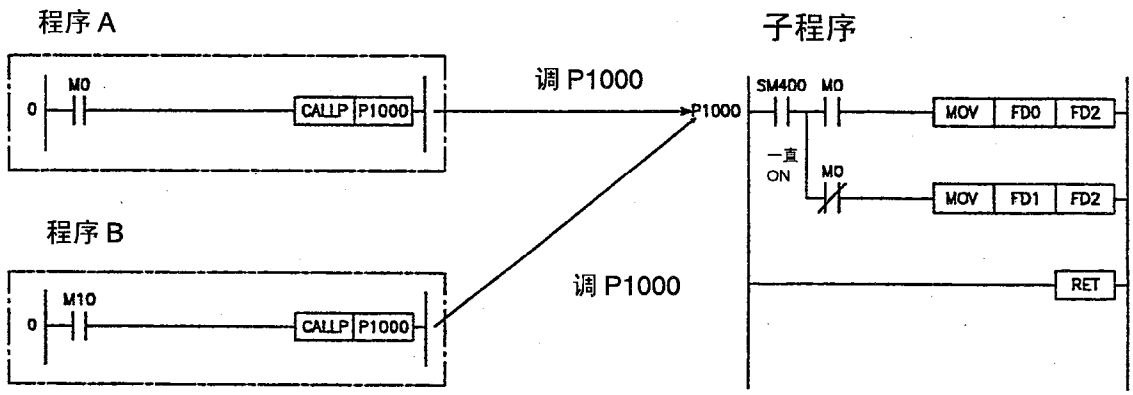
1. *：NUL 表示“00H”（字符串的末尾）

(b) 丰富的数据处理指令，如表格处理指令等，能够高速处理大量数据。

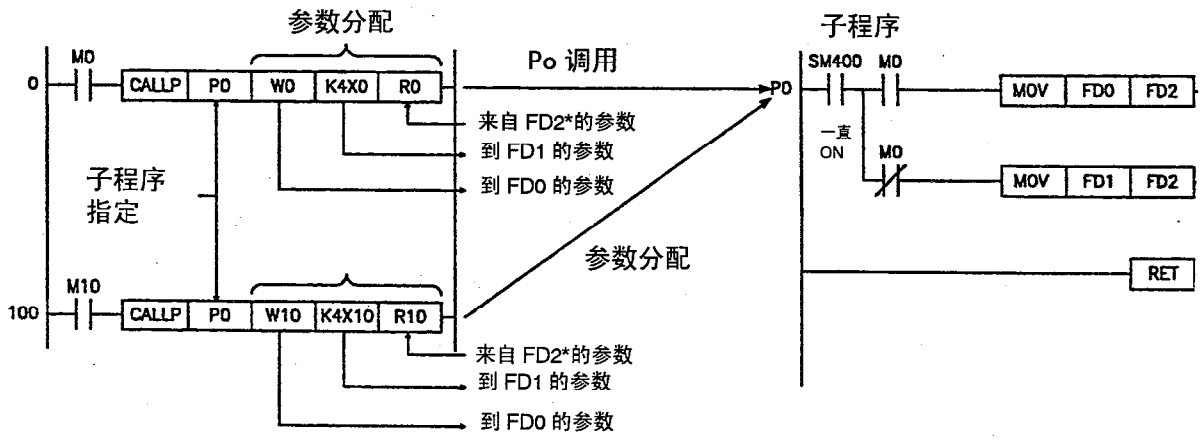


(4) 简单的子程序使用共享

(a) 可以用通用指针在所有的正在执行的顺控程序中调用相同的子程序。



(b) 使用带有参数的子程序调用指令可以简化被多次调用的子程序的创建。



注释

1. QnACPU 自动确定参数输入/输出的条件。

程序“源”数据被作为子程序的输入数据处理。

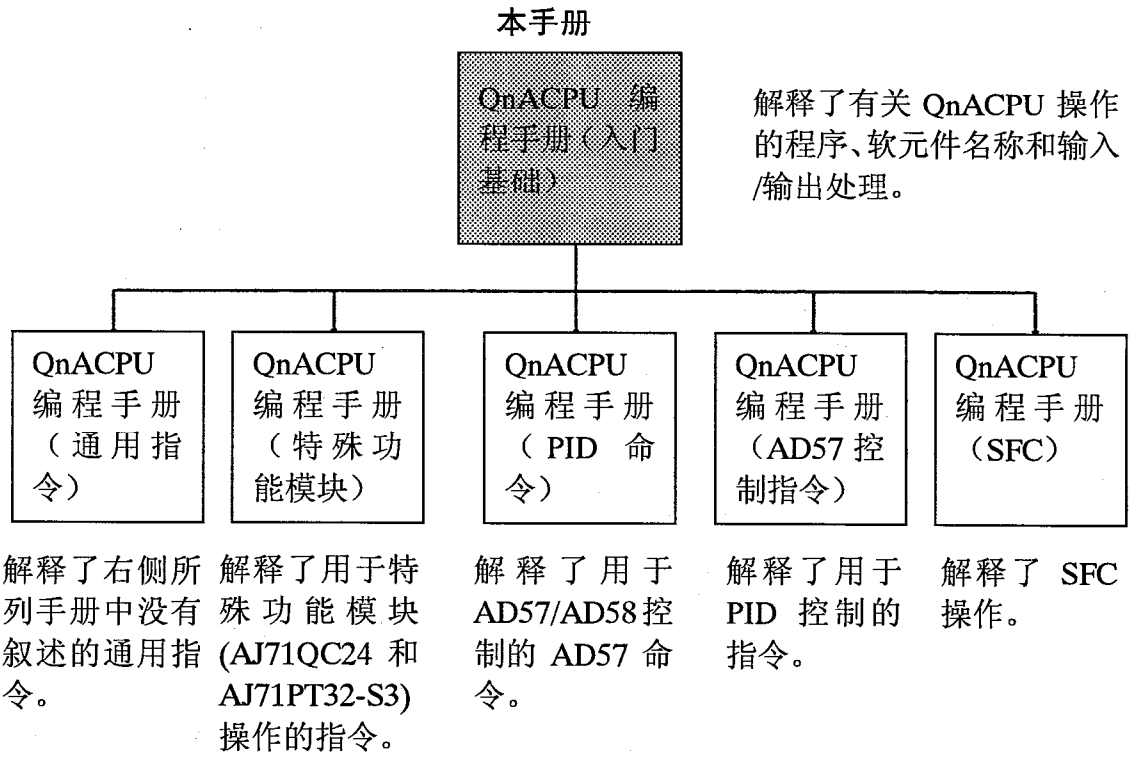
程序“目标”数据被作为子程序的输出数据处理。

1.3 相关的编程手册

除了本手册外，以下所列的 5 本手册也包含有关 QnACPU 操作的指令。

- QnACPU 编程手册（通用指令）
- QnACPU 编程手册（特殊功能模块）
- QnACPU 编程手册（PID 控制指令）
- QnACPU 编程手册（AD57 命令）
- QnACPU 编程手册（SFC）

请使用这本手册了解关于 QnACPU 程序、软元件和输入/输出处理等的基本概念，在实际运用中还需参考其他的手册。



注释

1) 除了以上所指定的编程手册，还有 Q4ARCPU 编程手册（PID 应用版）

2.QnACPU 文件

参数、程序和注释数据等均被指定文件名和扩展名，然后存储在 QnACPU 内部 RAM 或存储卡中。

将数据从一个外围设备写入到 QnACPU 中时，被写的文件是由它们的类型（参数、程序、注释等），而不是它们的扩展名所指定的。

（外围设备自动为指定的文件类型分配相应的扩展名。）

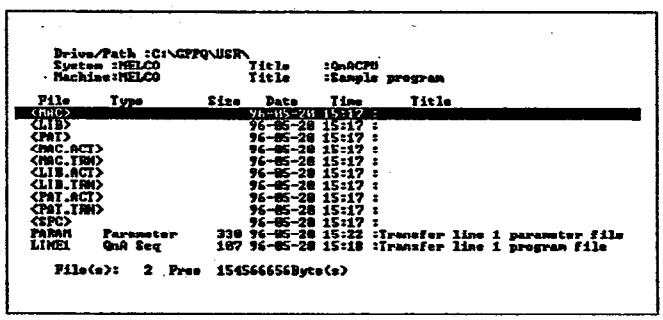
使用不同的文件和扩展名使得在 QnACPU 中可以存储多个文件。

因为 QnACPU 也能够将一个给定的程序当作一个文件来处理，因此创建的程序可以根据它们的“设计者”、“处理”或“功能”通过使用不同的程序文件名而分别独立管理。此外，可以执行存储在 QnACPU 中的多个程序。（QnACPU 程序执行的详细资料请参见第 3 章。）

QnACPU 将文件存储在内部 RAM 和存储卡的空余空间中。

如果当一个写请求发生时（从外围设备），连续的空余存储空间不够容纳某一个文件，那么就不可能将其写入内部 RAM 和存储卡中。（参见 2.6.2 章节）

一个文件的名字、文件大小和创建日期都将附在每个文件中。下面所示的文件列表是显示在外围设备上的。



文件列表显示项目的解释如下。

（1）文件名称

文件名称由文件名字（最大 8 个字符）和扩展名（3 个字符）组成。QnACPU 区分大写字母和小写字母。（在外围设备，所有的字符都转换成大写字母。）

文件扩展名对应于在文件被写入外围设备时所指定的文件类型，它是自动附加在文件名上的。

（2）文件大小

文件大小以字节为单位表示。

文件是以 4 个字节为单位（1 步）存储在内部 RAM 中的，而以 1 个字节为单位存储在存储卡中的。当计算一个文件大小的时候，除寄存器文件以外，请注意至少有 64 个字节（对于程序是 132 个字节）将被加入到所有的用户创建的文件中。

（3）日期和时间

日期和时间是在文件被从外围设备写到 QnACPU 时指明的。

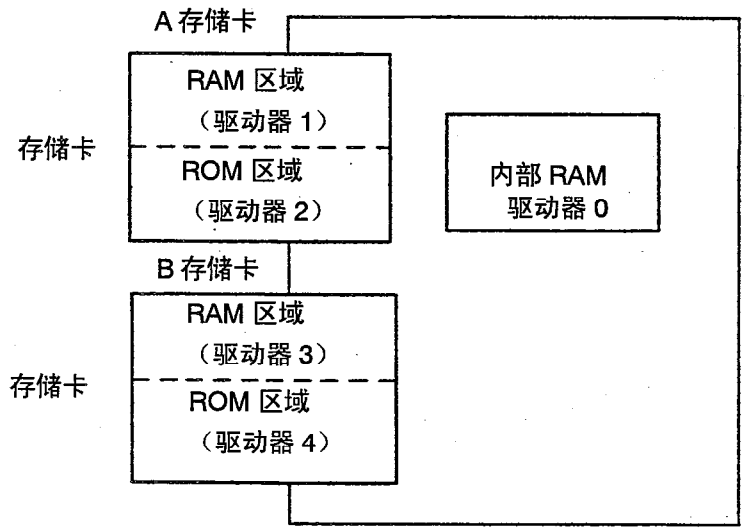
（4）标题

指明用户文件应用等。（最大 16 个字符）

2.1 QnACPU 内部 RAM 和存储卡

指定内部 RAM 和存储卡

QnACPU 有两种类型的文件存储区域：内部 RAM 和存储卡，每一个存储区域被指定一个驱动器编号*。内部 RAM 被指定为驱动器 0，A 存储卡被指定为驱动器 1 和 2，B 存储卡被指定为驱动器 3 和 4。



注释

1）*：当从外围设备向 QnACPU 写入参数数据和程序等时，由驱动器编号指定写入数据的存储器（内部 RAM/存储卡）。

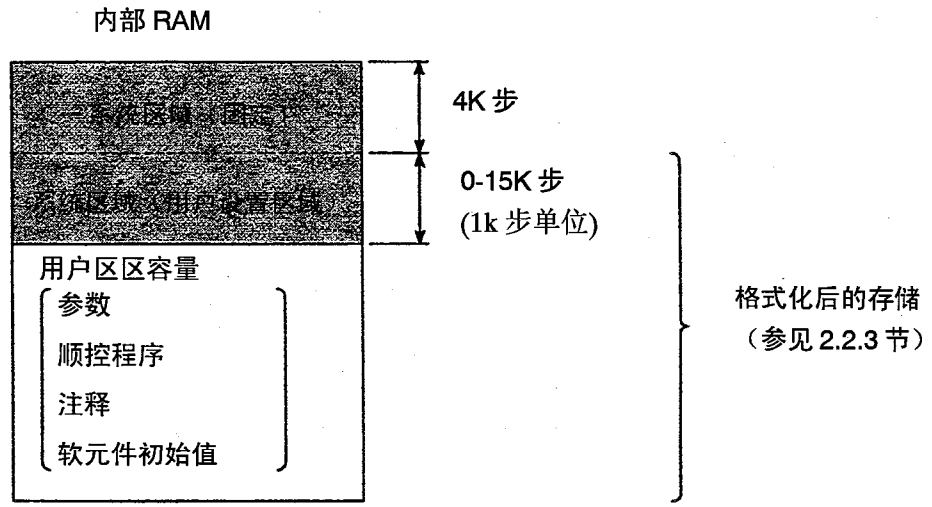
2.2 内部 RAM

本节叙述 QnACPU 内部 RAM 的存储图和容量。

要点
（1） 在初次使用 QnACPU 的内部 RAM 之前，必须先将其格式化。关于内部 RAM 格式化过程的详细资料，请参考“SW□IVD/NX-GPPQ 类型 GPP 功能软件包操作手册（在线）” 。 (2) 程序文件以 1K 步为单位存储在内部 RAM 中。

2.2.1存储器存储图

文件按以下格式存储在内部 RAM 中。



2.2.2格式化注意事项

QnACPU 内部 RAM 只有被外围设备格式化后才能使用。

当格式化内部 RAM 时，指定是否要分配一个系统区域供用户设置。最多可分配 16K 步（以 1K 步为单位）作为用户设置系统区域。

系统区域用户设置区域是用来寄存串行通讯模块，网络中其他站点上的外围设备的监视数据的。

尽管指定一个用户设置区域可以加速对串行通讯模块与其他网络站的监控，但是它也减少了用户文件的可用空间。

2.2.3格式化后的存储器容量

当内部 RAM 被格式化后，外围设备的文件清单所显示的内部 RAM 容量如下所示。

表 2.1　格式化后的存储器容量

CPU 型号名称	存储器容量
Q2ACPU， Q2AS（H）CPU	28K 步（114688 字节）
Q2ACPU-S1， Q2AS（H）CPU-S1	60K 步（245760 字节）
Q3ACPU	92K 步（335872 字节）
Q4ACPU	124K 步（507904 字节）

2.3 存储

本节叙述 QnACPU 存储卡的存储图和存储容量。

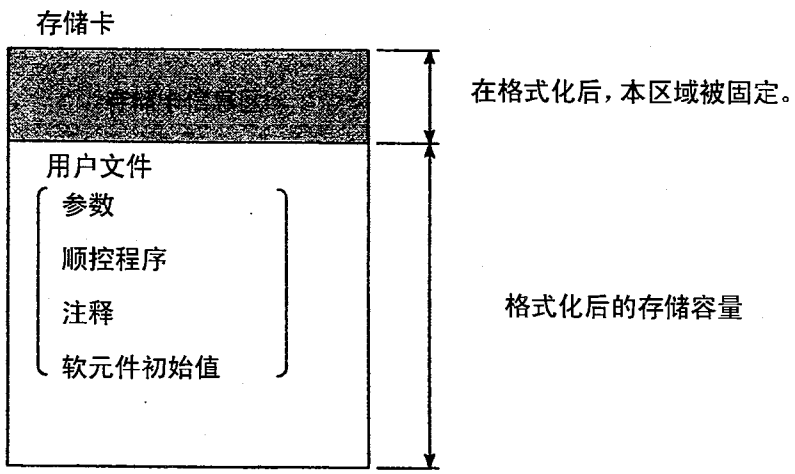
要点

（1）在可以第一次使用 QnACPU 存储卡之前，它必须先将其格式化。只有格式化后的参数数据和程序文件等才能存储在存储卡上。关于存储卡格式化过程的详细资料，请参考“SW 卡[illegible]VD/NX-GPPQ 型 GPP 功能软件包操作手册（在线）”。

（2）程序文件以 512 字节（128 步）步单位存储在内部 RAM 中。

2.3.1 存储图

文件按以下格式存储在存储卡中。



2.3.2格式化后的存储容量

在存储卡被格式化后，外围设备的文件清单所显示的存储卡容量如下所示。

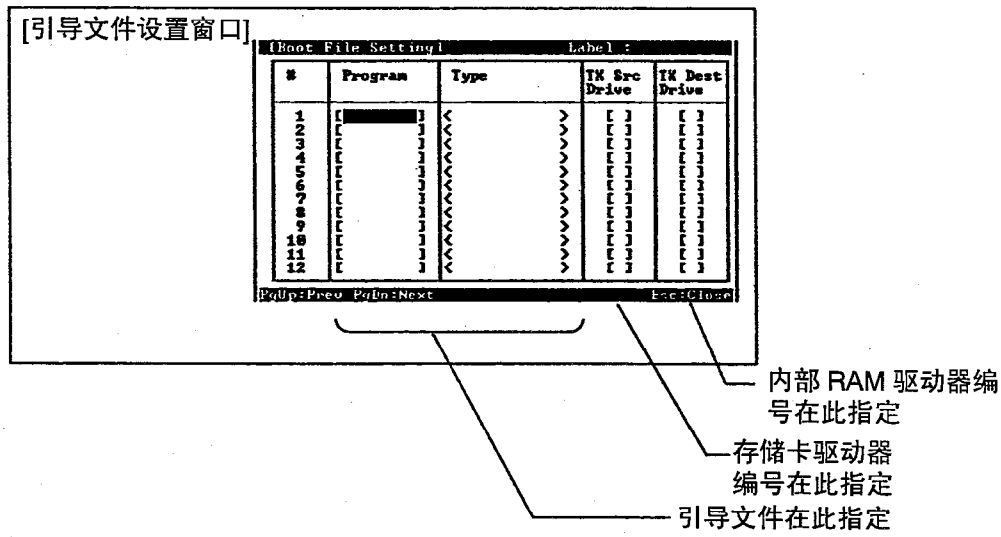
表 2.2 格式化后的存储器容量

存储卡型号名称	存储器容量（K-字节）			最大存储文件数		
	SRAM	E^2PROM	闪存	SRAM	EEPROM	闪存
Q1MEM-64S	59	–	–	118	–	–
Q1MEM-128S	123	–	–	128	–	–
Q1MEM-256S	250.5	–	–	128	–	–
Q1MEM-512S	506	–	–	128	–	–
Q1MEM-1MS	1016.5	–	–	128	–	–
Q1MEM-2MS	2036.0	–	–	256	–	–
Q1MEM-64SE	28.5	29.0	–	57	58	–
Q1MEM-128SE	58.5	59.0	–	117	118	–
Q1MEM-256SE	122.5	123.0	–	128	128	–
Q1MEM-512SE	250.0	250.0	–	128	128	–
Q1MEM-1MSE	505.5	506.0	–	128	128	–
Q1MEM-256SR	122.5	–	*	128	–	128
Q1MEM-512SR	250.0	–	*	128	–	128
Q1MEM-1MSR	505.5	–	*	128	–	128
Q1MEM-2MSR	1016.0	–	*	128	–	128

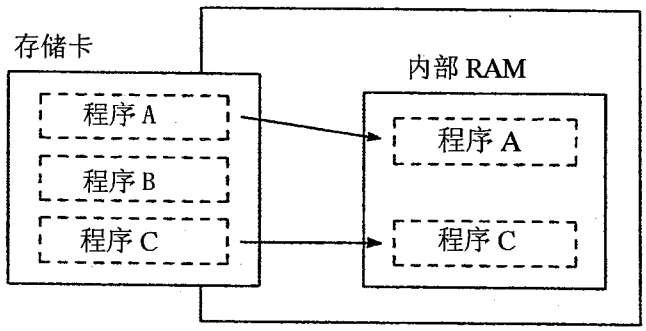
*取决于执行格式化的存储卡读写器的规格。

2.3.3 执行存储卡程序（引导运行）

(1) QnACPU 只运行存储在内部 RAM 中的程序。因此，存储在存储卡中的程序必须使用参数设置指定的引导文件名来引导（读入）到内部 RAM 中。当打开电源或 QnACPU 被复位时，参数中指定的文件就被从存储卡引导（读入）到内部 RAM 中。



(2) 当 QnACPU 处于运行状态时，不能从存储卡向内部 RAM 添加文件，也不能进行文件更改和删除。要想执行这些操作，停止 QnACPU，指定想要的引导文件（通过参数），然后复位 CPU，在复位结束后运行 CPU。



引导处理过程

1. 指定（通过参数设置）要传递的文件。
2. 复位 QnACPU。
3. 指定的文件从存储卡传递到内部 RAM 中。
4. 传递的文件开始执行

2.4 QnACPU 管理的文件类型与存储场所

(1) QnACPU 管理的文件类型与存储场所

能够被存储在 QnACPU 内部 RAM 和存储卡中的文件是根据文件类型决定的，可存储的文件和它们的存储场所如下表 2.3 所示。用户可根据需要决定是否采用存储卡。

表 2.3 QnACPU 中的文件和存储场所

项目	文件类型	文件名*4	存储驱动器*1					限制	参考
			0	1	2	3	4		
程序用	参数	********.QPA	○	○	○	○	○	每个驱动器 1 个文件	第 5 章
	顺控程序/SFC 程序	********.APG	.*2	○*2	○*2	○*2	○*2		第 3 章
软元件用	软元件注释	********.QCD	○*5	○	○	○	○	最大 124 个文件	用户手册*3
	软元件初始值	********.QDI	○	○	○	○	○	最大 124 个文件	4.13 节
	文件寄存器	********.QDR	×	○	.	○	.	最大 124 个文件	4.7 节
	模拟数据	********.QDS	×	○	×	○	×		用户手册*3
	局部软元件	********.QDL	×	○	×	○	×	每个 CPU 一个文件	4.13 节
调试用	采样跟踪数据	********.QTS	×	○	×	○	×		用户手册*3
	状态锁存数据	********.QTL							
	程序跟踪数据	********.QTP							
	SFC 跟踪数据	********.QTR							
故障诊断用	故障履历数据	********.QFD	×	○	×	○	×		用户手册*3

注释

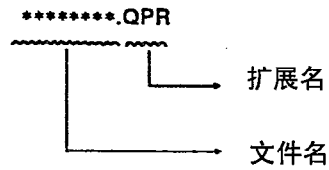
1)*1：上表中使用的符号解释如下。

符号	意思
.*	必须存储。
○	当需要时存储。
×	不能存储。
.	当使用了闪烁存储器时：只能进行文件读取（不能写） 当使用了 EEPROM 时：可用 EROMWR 指令写。

2）*2：QnACPU 运行存储在驱动器 0 中（内部 RAM）的程序。
为了执行存储在驱动器 1-4 的程序，必须设置引导参数。

3）*3：使用的 CPU 型号的用户手册。

4）*4：文件名配置如下：



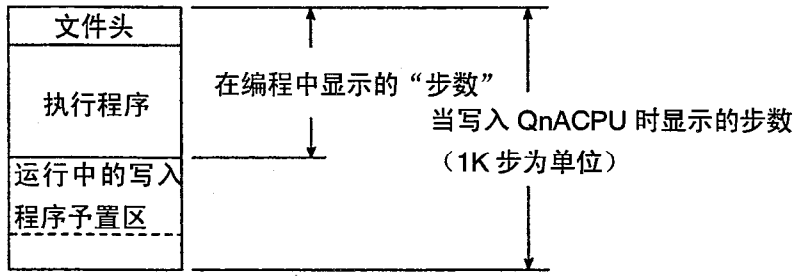
因为外围设备在将文件写入 QnACPU 中之前，先将指定的文件的文件类型转换成扩展名。所以用户不需要考虑扩展名了。

5）*5：存储在驱动器 0 中的软元件注释不能用于顺控程序和 SFC 程序的注释指令（LEDC 等）。

2.5 程序文件配置

程序文件由一个文件头、一个执行程序和运行中写入程序予置区组成。

如下所示，存储在 QnACPU 中的一个程序大小包括以上的成分。



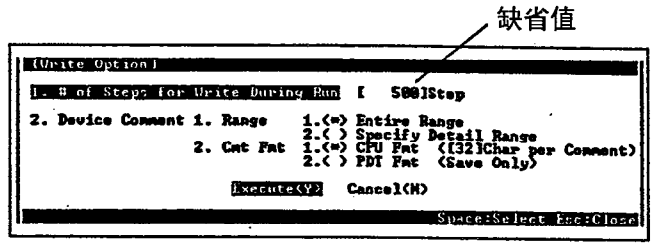
- 文件头：文件名、文件大小和文件创建日期等，都保存在这一区域。固定的 132 字节（33 步）。
- 执行程序：创建的程序保存在这一区域

  1 步 = 4 个字节。

运行时的写所需的步：当外围设备增加了程序步数并将程序在 CPU 运行时写入时所使用的区域。

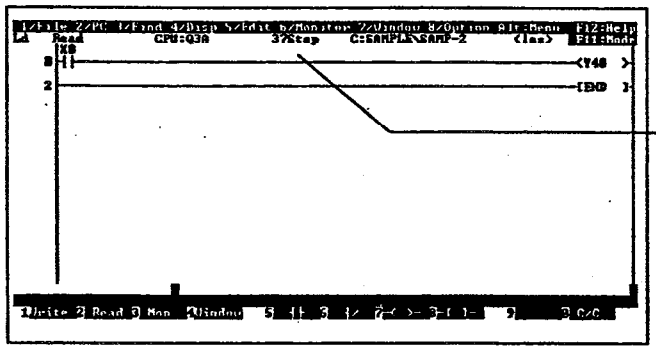
缺省值 = 500 步（2000 个字节）。

可以用外围设备的 PC 菜单的写选项更改该值。



在外围设备编程中，总的文件头的大小和执行程序的步数显示为“使用的步数”。

[梯形图读出屏幕]



注释

根据存储程序的存储器的不同，程序存储容量也就不同。

存储器	程序容量单位
内部存储器	4096 字节（1K 步）
存储卡	512 字节

2.6 文件操作与文件处理注意事项

2.6.1 文件操作

使用 GPP 外围设备的“在线”功能，对于存储在内部 RAM 和存储卡中的文件，可以进行表 2.4 所示的文件操作。

然而，可用的文件操作会根据一个输入码（由外围设备寄存）存在与否、QnACPU“写保护”开关的设置状态和 QnACPU 运行/停止状态而不同。

表 2.4 外围设备的文件操作

文件操作	操作可/不可*1				操作描述
文件列表	A	B	C	D	显示存储在存储器中的文件清单。
读	o	Δ*2	o	o	从存储器中读文件。
写*4	o	Δ*2	o	o	向存储器中写文件。
运行状态中程序写	Δ*2	Δ*2	×	Δ*3	在 QnACPU 运行状态下执行写程序。
重新命名	Δ*2	Δ*2	×	o	存储在存储器中的文件名被更改。
复制	Δ*2	Δ*2	×	Δ*3	存储在存储器中的文件被复制。
删除	Δ*2	Δ*2	×	Δ*3	存储在存储器中的文件被删除。
PC 存储器整理*4	Δ*2	Δ*2	×	×	存储器中不再相邻的文件被重新组织以使它们相邻。
PC 存储器格式化	Δ*2	Δ*2	×	×	执行存储器格式化。

注释

1）*1：在上表的“操作可/不可”项目中用到的代码（A，B，C，D）的解释如下。

操作可/不可

代码	描述	参考
A	当“写禁止”关键字寄存在 CPU 上时	使用的 CPU 型号相应的用户手册（详细信息）
B	当“读/写显示禁止”关键字寄存 CPU 上时	
C	当 CPU 的“系统保护”开关为 ON 时	
D	当 CPU 的运行/单步运行状态有效时	

☞表中使用的符号

符号	描述
o	可以运行
Δ	可以运行，当有一些限制
×	不能运行

2）*2：只有当关键字匹配时，参数和程序文件才有可能执行。

3）*3：以下几种情况的状态是“不能操作”（X）：

☞参数程序+程序文件

☞在运行状态中有新文件创建（当在复制场所中不存在与要复制的文件相同的文件时）

☞在运行状态中复制到驱动器 D

4）*4：为了确保在相邻的区域存放一个指定的文件数据，如果该文件的大小增加时，可能会进行文件移动。

5）对存储卡的 EEPROM 进行读/写与对它的 RAM 进行读写的方法是一样的。然而，写 EEPROM 的时间要比写 RAM 的时间长。

6）不能直接从外围设备向 IC 存储卡的闪烁存储器中写文件。要想这样做，就必须在 GPP 功能外围设备上安装一个存储卡读/写器，并且通过该读/写器进行写操作。

2.6.2 文件处理注意事项

本节讨论关于处理 QnACPU 文件的注意事项。

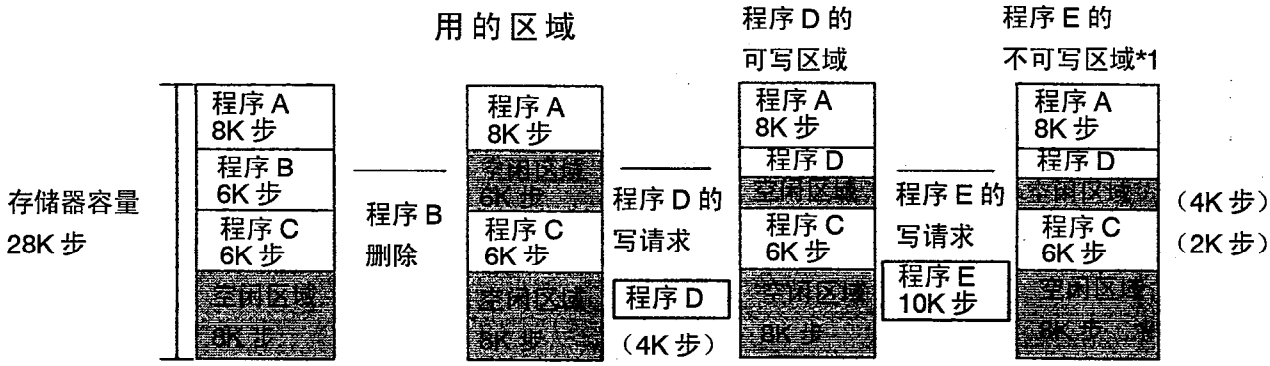
（1） 在存储器中的文件相邻

（a）内部 RAM 和存储卡中的文件基本上是配置于连续区域中的。然而，在进行了多次的文件删除和写操作后，连续区域的容量可能会减小到再也无法存储文件的地步，尽管空余区域的总量是充足的。

（b）当空余区域的总量超过了要写入的文件的大小时，可以执行“PC 存储器整理”功能（GPP 功能外围设备在线模式），从而将所有的不相邻空余区域合并成一个连续区域。

例 1：当不能写入 QnACPU 内部 RAM 时：

此处为了简化说明，不考虑系统文件和参数等占用的区域



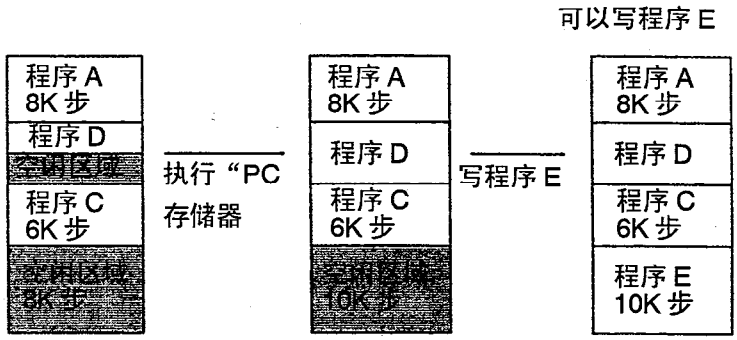
整个空闲区域	8K 步	14K 步	10K 步	10K 步
连续空闲区域	8K 步	8K 步	8K 步	8K 步

注释

1）*1：由于连续的空闲区域只有 8K 步，因此无法写入内部 RAM。

例 2：当执行“PC 存储器整理”功能时：

如果内部 RAM 的状态仍然如前一页的例 1 所示，执行“PC 存储器整理”功能可以保证有 10K 步的连续空闲区域。（下面图表中的“*”所示的）



整个空闲区域	10K 步	10K 步	0K 步
连续空闲区域	8K 步	10K 步	0K 步

（2） 在程序操作中的电源关闭（或复位）

（a）如果在进行不会导致文件移动的文件操作期间关闭电源（或发生复位），那么存储器中的文件将不会丢失。

（b）如果 QnACPU 的待机电池有效，即使在进行会导致文件移动的文件操作期间关闭电源（或发生复位），则存储器中的文件也不会丢失。

只要在关闭电源后不从 QnACPU 上卸下存储卡，存储在存储卡中的文件就不会丢失。

2. QnACPU 文件

要点

（1）以下文件操作会导致文件移动：

- 文件大小改变
- PC 存储器整理功能
- 新文件创建

（2）如果在上述操作过程中发生断电，那么断电前的数据将被保存在 QnACPU 的内部 RAM 中，当电源再打开时，这些数据将被恢复。这需要有电池供电以保存内部 RAM 数据。

（3）当程序大小增加时，在程序运行中进行写操作

（a）QnACPU 程序文件的大小是创建的程序的空间加上运行中写所需要的步。如果使用了外围设备的 GPP 功能在程序运行中执行了写操作，写入文件的大小不能超过初次写入文件时所预留的空间。

（b）如果一个编辑过的文件大小超出其初次写入文件时所预备的空间的话，则无法在运行中执行写操作。

- 如果文件的大小可能由于运行中写操作而增加，那么就在外围设备上多设置一些运行中写操作所需的步。
- 如果在 QnACPU 停止时写入，即使文件大小增加，也可以执行写操作。

(4) 从多个外围设备同时存取一个文件

（a）QnACPU 允许一个文件在被一个 RS-422 连接的外围设备存取的同时又被另一个通过网络或串行通讯模块连接的外围设备存取。

3. 顺控程序配置和执行条件

在 QnACPU 中可以执行顺控程序和 SFC 程序。本章描述顺控程序配置和执行条件。

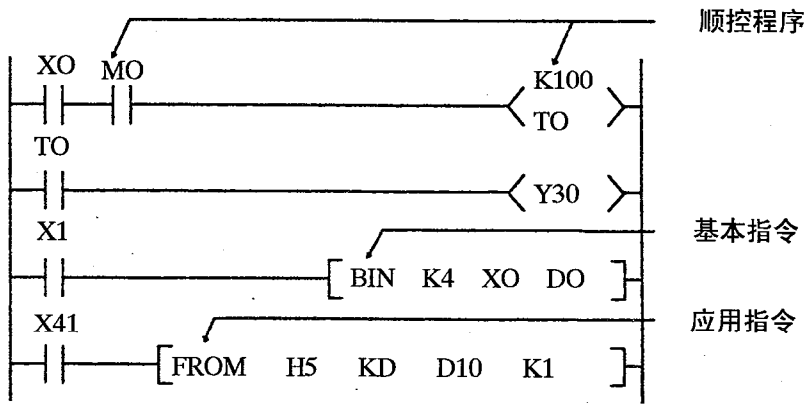
在本手册中没有描述 SFC 程序。

关于 SFC 程序的详细资料，请参见 QnACPU 编程手册（SFC）。

3.1 顺控程序

（1） 顺控程序的定义

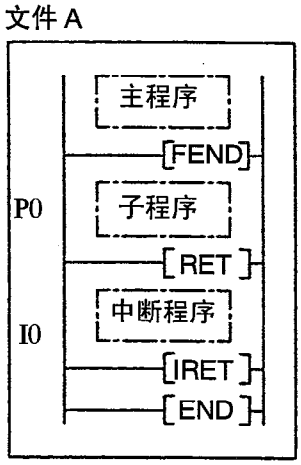
（a）一个顺控程序是用 QnACPU 的顺控指令、基本指令和应用指令等创建起来的。



（b） 有 3 种类型的顺控程序：主程序，子程序和中断程序。

关于这些程序的详细资料请参见本手册的下列章节：

主程序：3.1.1 节
子程序：3.1.2 节
中断程序：3.1.3 节



注释

1）关于 QnACPU 顺控指令、基本指令和应用指令的详细资料，请参见“QnACPU 编程手册（通用指令）” 。

（2） 顺控程序的格式

使用继电器符号语言（梯形图模式）或者逻辑符号语言（指令表模式）都可以进行顺控程序的编程。

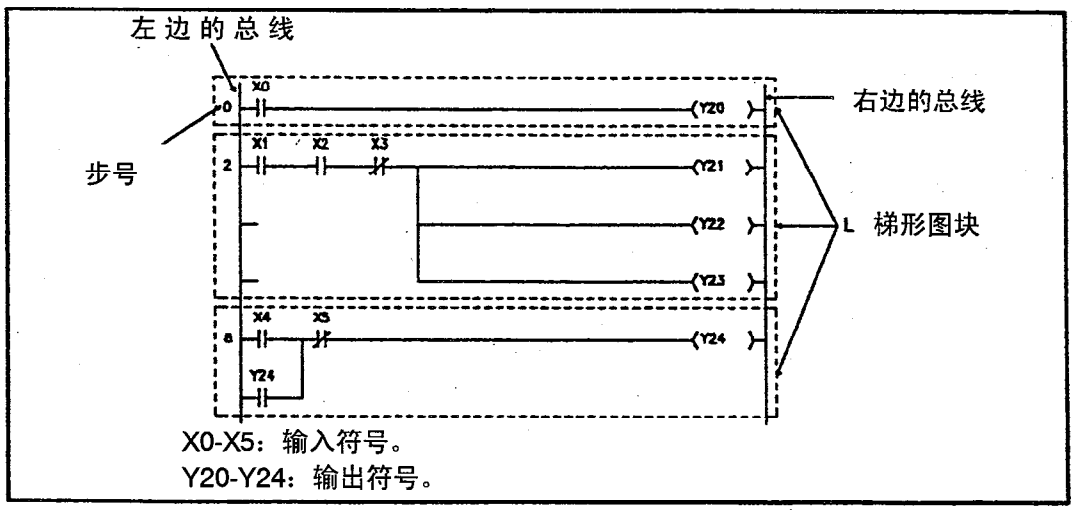
（a） 继电器符号语言

继电器符号语言源于继电器控制电路。

编程表达方式与继电器控制顺序电路相似。

继电器符号语言以梯形图块为单位编程。

一个梯形图块是顺控程序处理的最小单位，从左侧的总线开始，在右侧的总线结束。



3.1 梯形图块

（b） 逻辑符号语言（指令表模式）

逻辑符号语言以专用指令表示继电器符号语言中的触点符号、线圈符号等，以列表形式编程。

（c） 程序处理

顺控程序是按照顺序执行的，从第 0 步开始，到 END 指令结束。

继电器符号语言梯形图块是从左侧的总线开始，从左到右执行的。

当一个梯形图块完成后，就向下执行下一个梯形图块。

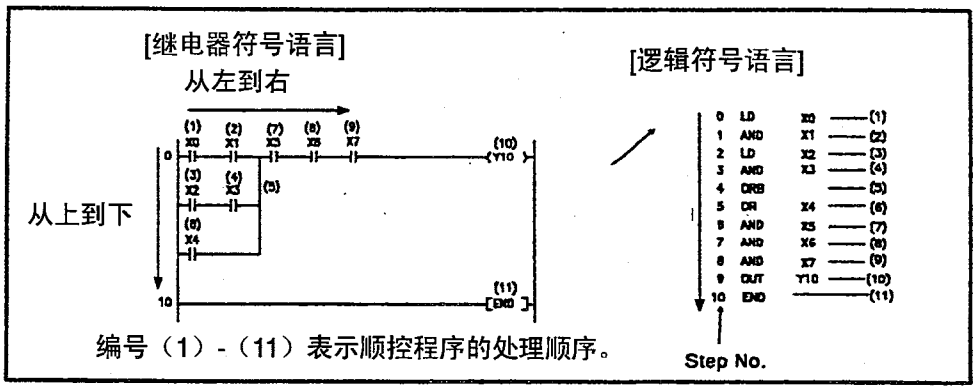


表 3.2 顺控程序处理过程

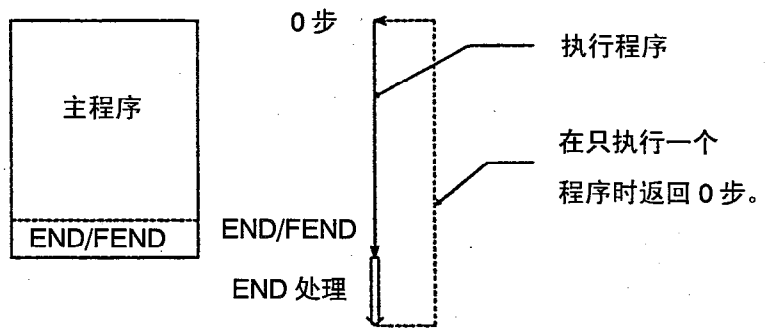
3.1.1 主程序

（1） 主程序的定义

（a） 主程序是从第 0 步开始到 END/FEDN 指令的一段程序。

（b） 主程序是从第 0 步开始执行到 END/FEDN 指令结束。

1）如果只有一个程序被执行，那么处理过程将在 END/FEDN 指令执行完后经过 END 处理，然后又从第 0 步开始执行。



2）如果有多个程序被执行，根据不同的指定的执行条件，在 END/FEND 命令执行完毕后，CPU 的处理会不同。

（2）主程序可被设定的执行条件：

如果有多个程序被执行，在参数中的程序设定一栏中可根据需要将程序设定为以下四种执行类型。

- 初始化执行程序：参见 3.2.1 节
- 扫描执行程序：参见 3.2.2 节
- 低速执行程序：参见 3.2.3 节
- 待机程序：参见 3.2.4 节

3.1.2 子程序

（1） 子程序的定义

（a）一个子程序是从一个指针（P[]）开始的到 RET 指令结束的一段程序。

（b）一个子程序只有在被主程序的一个 CALL（P）指令调用时，才被执行。

（2） 子程序的应用

（a）可以通过将一个子程序作为在一次扫描中被多次执行的程序来减少程序总步数。

（b）可以使用一个只有在某一给定条件符合时才被执行的子程序来减少一个经常执行的程序的步数。

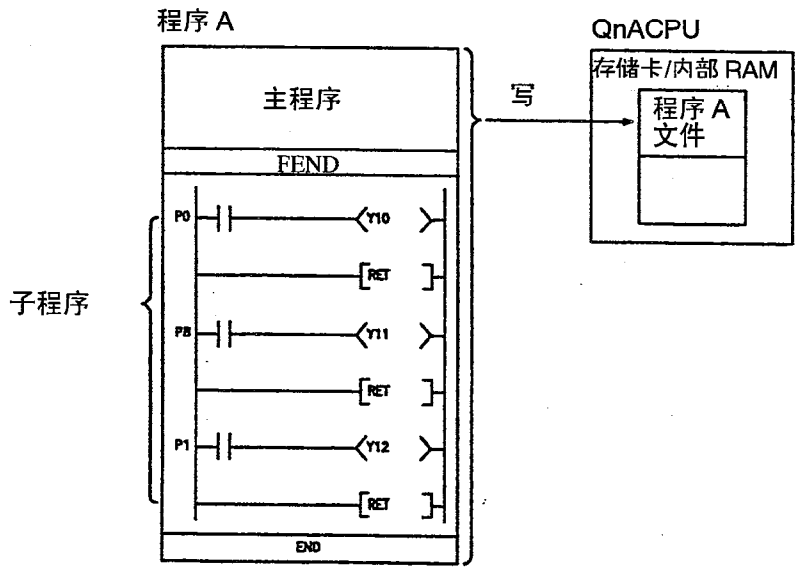
（3） 子程序的管理

子程序创建在主程序之后（在 FEND 指令之后），可作为主程序的一个部分。

子程序也可以作为单独的、独立的程序（待机程序）来管理。（关于待机程序的详细资料请参见 3.2.4 节）。

（a） 当在主程序之后创建时

- 子程序创建在主程序的 FEND 和 END 指令之间。
- 因为对子程序创建的顺序没有限制，因此在创建多个子程序的时候，不需要按照升序设置指针。
- 局部指针和通用指针都可能被用到。*



注释

1）*：关于局部和通用指针的详细资料请参见 4.9 节。

2）关于子程序嵌套的详细资料请参见 4.8 节。

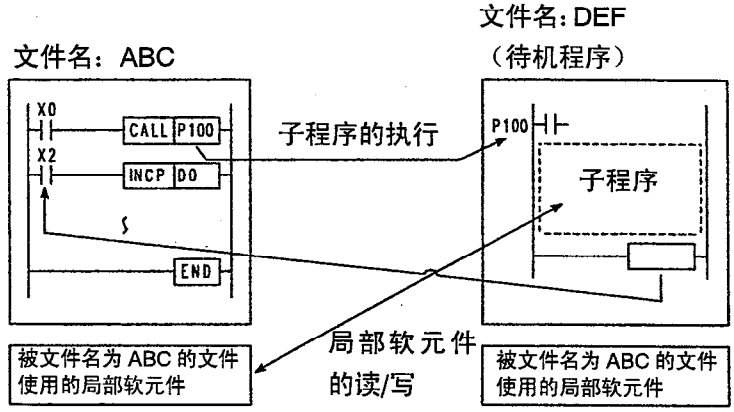
（4）使用由保存有子程序的文件使用的局部软元件

在执行一个子程序的时候，可使用其所在的主程序文件的局部软文件。由特殊继电器“SM776”设置来决定是否使用该局部软元件。

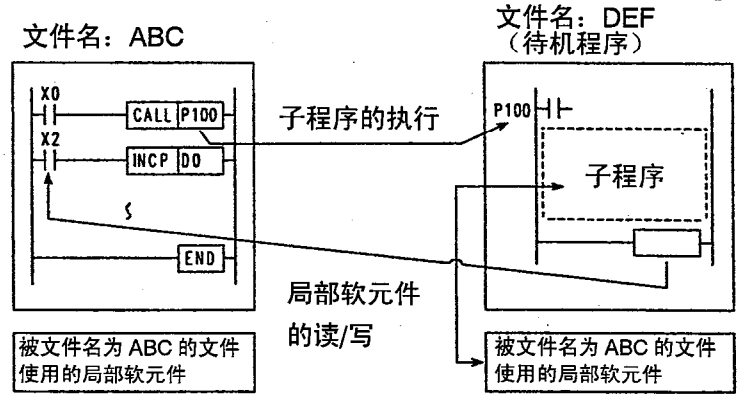
（a）通过设置特殊继电器（SM776）的开/关来实现局部软元件的切换

	SM776
关	用调用子程序的文件所使用的局部软元件来执行计算
开	用子程序所在的主程序的文件使用的局部软元件来执行计算

[在功能版本为 B/A 时*，在“SM776”的操作：“关闭”]



[在功能版本为 A 时*，在“SM776”的操作：“打开”]



* 功能版本 A 不支持局部软元件切换，因此 SM776 的设置无效。
* 功能版本 B 不支持局部软元件切换

（b） 警告

- 如果 SM776 是打开的,当子程序被调用时，他程序的局部软元件数据被读入，并且在 RET 指令执行之后，他程序局部软元件数据被保存。

  因此，当在“SM776：开”设置下执行一个子程序时，扫描时间将会被延长，如下所示。

- Q2ACPU（S1），Q2ASCPU（S1）：560+1.3^{X}（局部软元件的字数）[us]
- Q3ACPU：425+1.0^{X}（局部软元件的字数）[us]
- Q4ACPU，Q2ASHCPU(S1)：220+0.8^{X}（局部软元件的字数）[us]

- 在 QnACPU 或 Q2AS(H)CPU 情况下，以 CPU 为单位设置 SM776 是可行的，但是不能以文件为单位设置 SM776。

- 如果当一个顺控程序执行时改变了 SM776 的开/关设置，就会根据改变后的信息进行控制。

3.1.3 中断程序

(1) 中断程序的定义

(a) 一个中断程序是从中断指针（I[]）*开始，到 IRET 指令结束的一段程序。

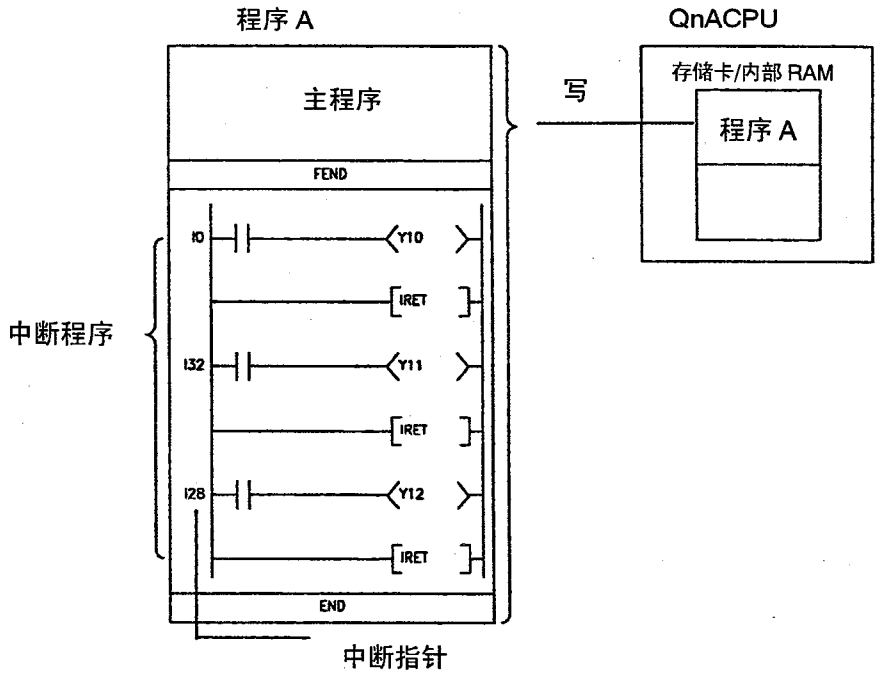
(b) 中断程序只有当一个中断发生时才被执行。*

(2) 中断程序的管理

中断程序在主程序之后（在 FEND 指令之后）创建，并且可作为主程序的一个部分作为一个程序管理。
中断程序也可以作为单独的、独立的程序（待机程序）来管理。（关于待机程序的详细资料请参见 3.2.4 节）。然而，对应于一个中断指针，不能创建多个中断程序。.

(a) 在主程序之后创建时

- 一个中断程序创建在主程序的 FEND 和 END 指令之间。
- 因为对中断程序创建的顺序没有限制，因此在创建多个中断程序的时候，不需要按照升序设置中断指针。



注释

1）*：关于中断要素和中断指针的详细资料请参见 4.10 节。

(3) 执行中断程序

(a) 为了执行一个中断程序，需要 IMASK 和 EI 指令来获得中断许可。*1

如果在“中断许可”状态之前有一个中断发生，那么当“中断许可”状态创建后，该中断的中断程序将被执行。

如果在停止/暂停状态中有一个中断发生，那么当“中断许可”条件创建后又返回到运行状态时，该中断的中断程序将被执行。

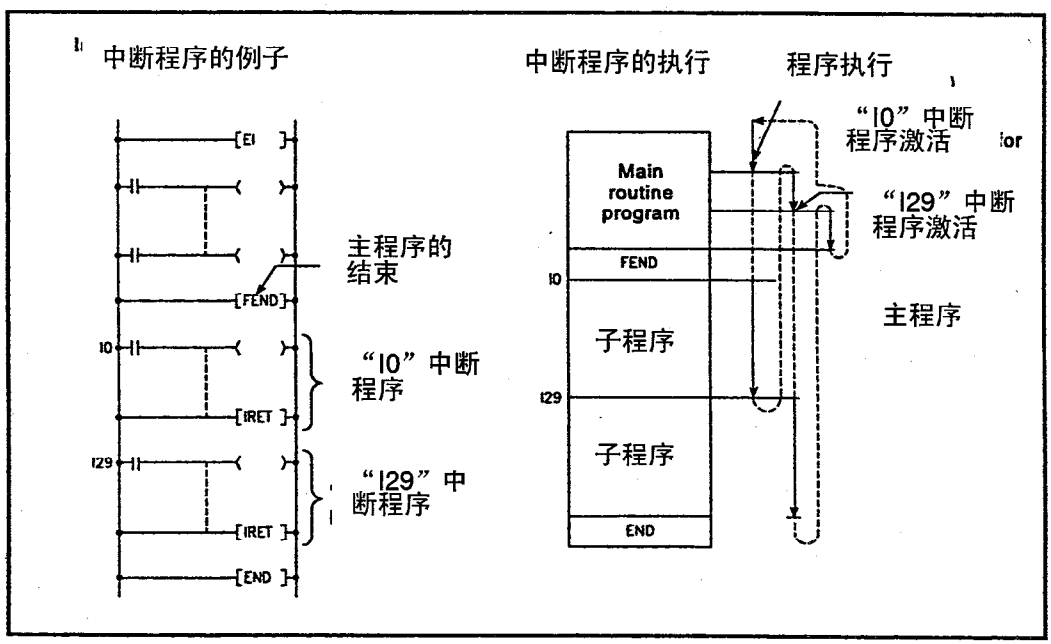


图 3.3 中断程序的执行

(b) 当有一个中断出现时，对应于该中断的中断指针数的中断程序被执行。然而，中断程序的执行根据当时的状况会有以下不同：

1) 当有多个中断发生时：

当有多个中断程序被同时激活时，将从拥有最高优先权的中断指针数的中断程序开始按优先度顺序执行。*2
剩下的中断程序保持为待命状态，直到执行中断程序执行完毕。

2) 当一个指令正在被执行时：

在指令执行期间中断是禁止的。如果在一条指令执行期间有一个中断发生，那么中断程序将在这条指令处理完成后执行。

注释

1）*1：关于 IMASK 和 EI 指令的详细资料请参见“QnACPU 编程手册（通用指令）” 。

2）*2：关于中断程序的优先级的详细资料请参见 4.10 节。

3) 在链接刷新期间的中断：

如果在一个链接刷新操作期间有中断发生，那么该链接刷新操作将被挂起，而执行中断程序。

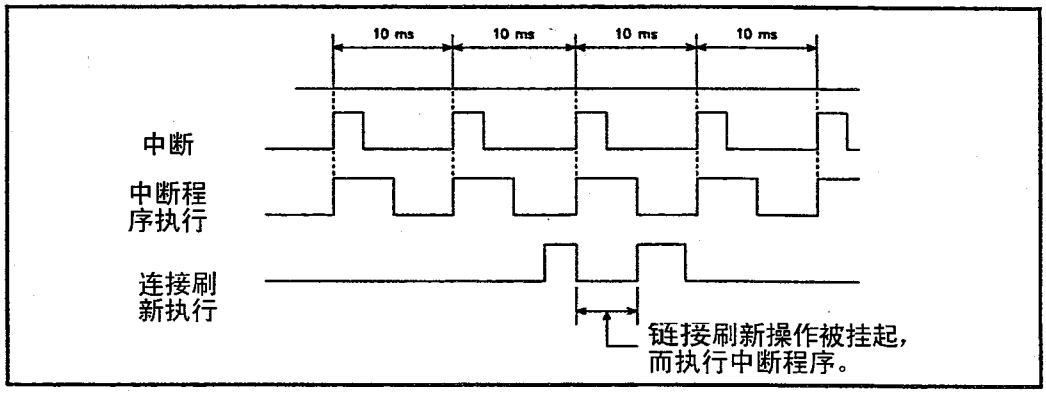


图 3.4 连接刷新操作期间的中断

4) 在 END 处理期间的中断：

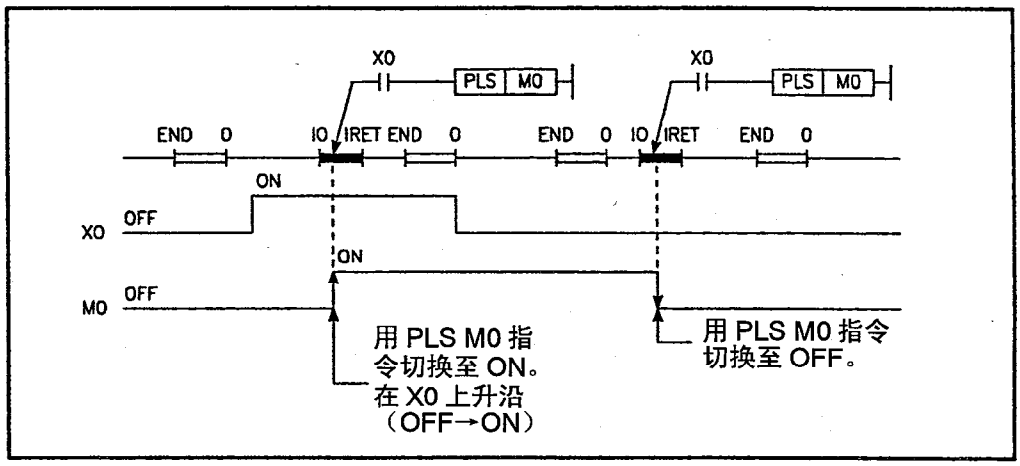
i) 如果在 END 处理期间正在进行一般数据处理的过程中有中断发生，那么在一般数据处理完成后，将执行中断程序。

ii) 如果在恒定扫描的 END 指令后的等待期间有中断发生，那么相对应于该中断的中断程序将被执行。

(c) 关于在从一个扫描执行程序或低速执行程序切换到一个中断程序时，变址寄存器处理的详细资料，请参见 4.6 节。

(4) 程序创建的限制

(a) 被一个中断程序中的 PLS 指令打开的软元件将一直保持打开状态，直到该中断程序再次执行。



(b) “DI”状态（中断禁止）是在执行中断程序期间创建的。
不要在中断程序中执行 EI/DI 指令。

(c) 定时器不能用于中断程序。
由于定时器是用在 OUT T[]指令中来更新当前值并将触点打开和关闭的，因此在中断程序中使用定时器将无法使进行正常的时间计数。

(d) 不能为中断程序设置专用的局部软元件。（参见 4.13.1 节）

(5) 使用保存有中断程序的文件使用的局部软元件

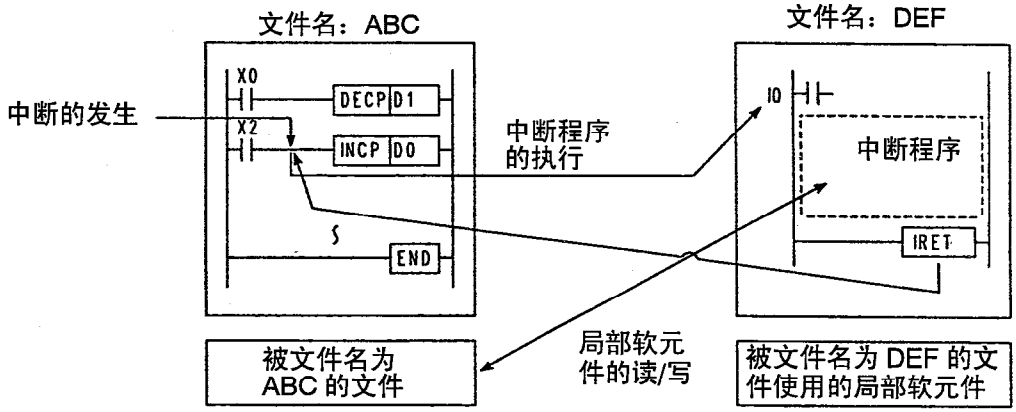
在执行一个中断程序时，可以使用中断程序所在主程序的文件使用的局部软元件。

是否使用该局部软元件是由特殊继电器“SM777”设置的。

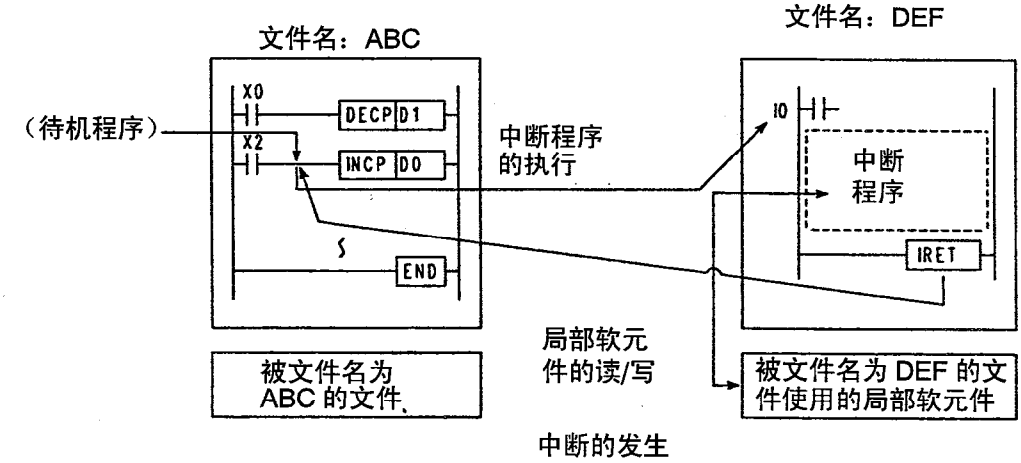
(a) 通过设置特殊继电器 SM777 的开/关实现局部软元件的开关切换

	SM777
关	用在中断程序执行之前的执行的文件所用到的局部软元件来执行计算
开	用中断程序所在主程序文件使用的局部软元件来执行计算

[在功能版本为 B/A 时，“SM777” 的操作：关闭”]



[在功能版本为 A 时，“SM777” 的操作：打开”]

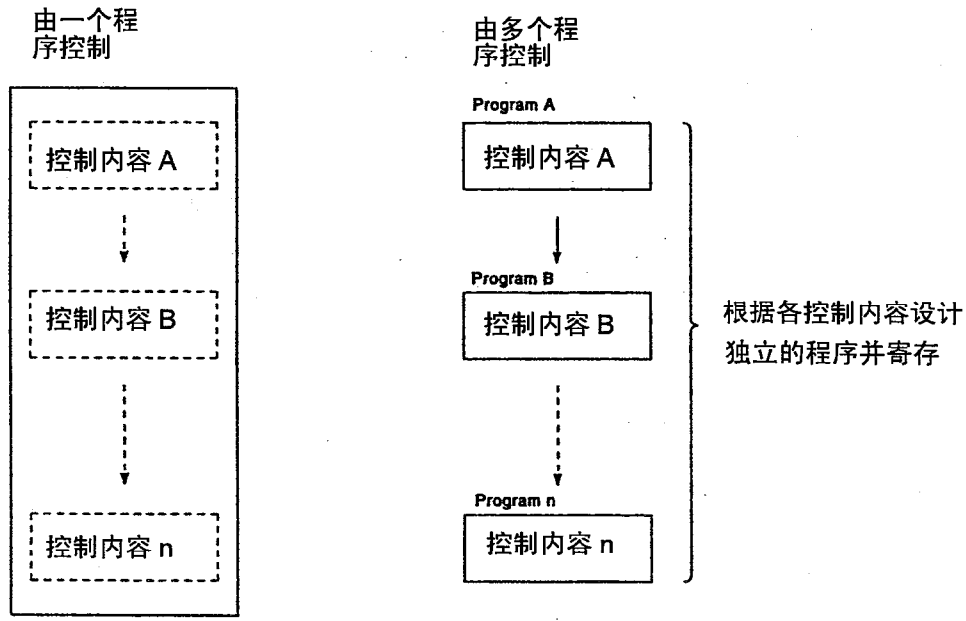


(b) 警告

- 如果 SM777 是打开的,在中断程序被执行之前，读入局部软元件数据，并且在 IRET 指令执行之后，保存局部软元件数据。
  因此，当在“SM777：开”设置下执行一个中断程序时，扫描时间将会延长，如下所示。
- Q2ACPU（S1），Q2ASCPU（S1）：560+1.3*（局部软元件的字数）[us]
- Q3ACPU：425+1.0*（局部软元件的字数）[us]
- Q4ACPU，Q2ASHCPU(S1)：220+0.8*（局部软元件的字数）[us]
- 在 QnACPU 或 Q2AS(H)CPU，以 CPU 为单元来设置 SM777 的开/关是可能的，但不能以文件为单位来进行设置。
- 如果当一个顺控程序执行时改变了 SM777 的开/关设置，就会根据改变后的信息进行控制。

3.2 程序执行条件和运行过程

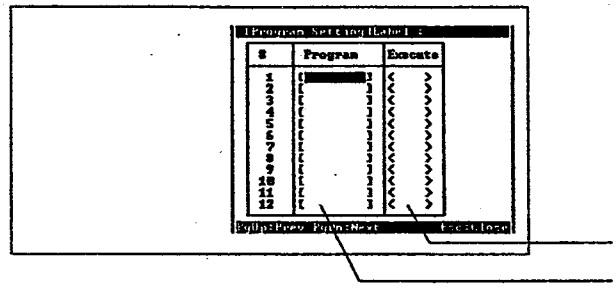
QnACPU 执行的程序保存在 CPU 的内部 RAM 或存储卡中。
程序既可以作为一个程序，也可以被分割成相应于每个控制功能的独立的程序，存储在内部 RAM 或存储卡中。
因此可以将编程过程分给多个程序设计员，他们可以为每个操作分别设计各自独立的程序。



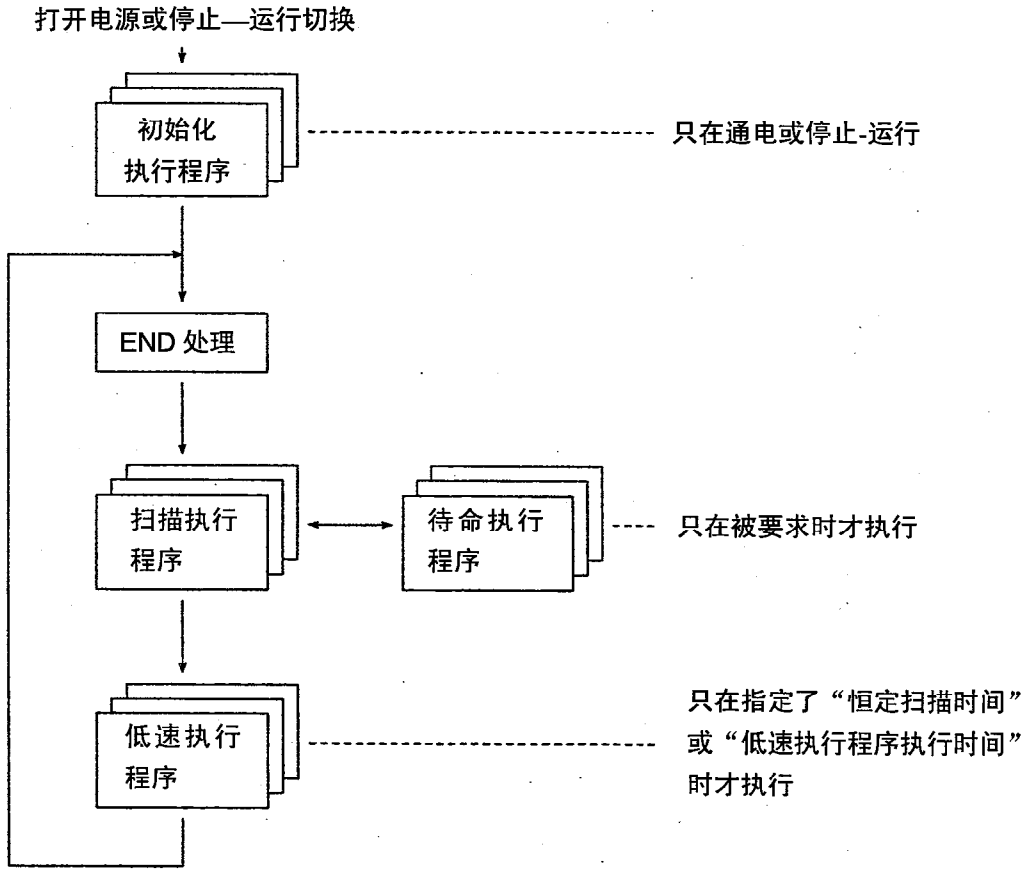
当一个操作被分割成多个程序后，可在参数的程序设置中分别分配一个“执行类型”。
QnACPU 按照在参数中各程序的设置类型和顺序来执行各程序。
有四种执行类型：初始化执行、扫描执行、低速执行和待机。

- 初始化执行：这种程序类型只在打开电源或停止-运行切换发生时执行一次。（参见 3.2.1 节）
- 扫描类型：这种程序类型在初始化程序执行后，每次扫描时执行一次。（参见 3.2.2 节）
- 低速执行：这种程序类型只在进行了恒定扫描时间设置或低速执行类型程序的时间设置时执行。
- 当进行了恒定扫描时间设置时，程序在扫描执行类型程序执行后恒定扫描时间的剩余时间里执行。
- 当为低速执行类型程序的执行设置一个时间时，程序在该设置时间里执行。（参见 3.2.3 节）
- 待机类型：这种类型程序只在被要求执行时才执行。（参见 3.2.4 节）

[程序设置窗口]



发生在打开电源或停止—运行切换时的程序运行步骤如下所示。



要点

(1) QnACPU 不要求指定所有的执行类型。*标明的初始化执行、低速执行和待机类型程序可按要求选用。

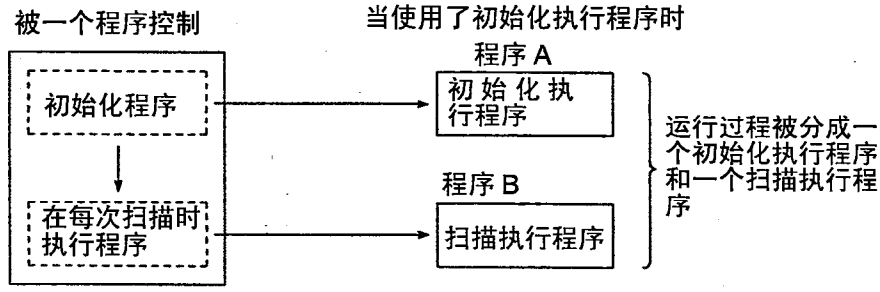
3.2.1 初始化执行程序

（1） 定义

（a） 初始化执行程序只在通电或当停止-运行切换发生时执行一次。

（b） 这种程序执行类型在程序设置参数中指定为“初始化” 。

（c） 初始化执行程序可用于如特殊功能模块的初始化处理等只需要执行一次的控制。

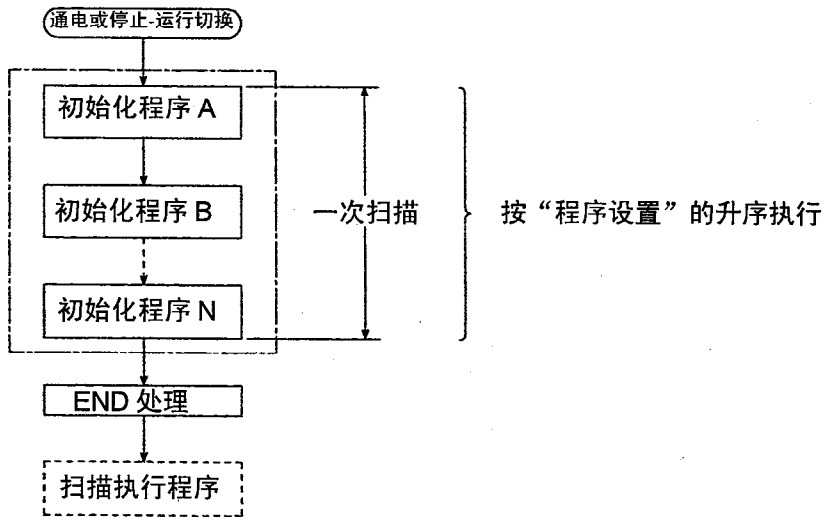


（2） 使用多个初始化执行程序

当使用了多个初始化程序时，它们被按照升序（程序设置参数的设置）一个一个执行。

（3） END 处理

当所有的初始化执行程序完成后，进行 END 处理，然后扫描执行程序从下一个扫描开始执行。



要点

1）*：带有“完成软元件”的指令不能用于初始化执行程序。

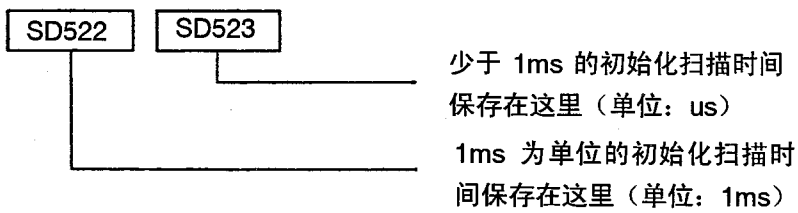
（4） 初始扫描时间

（a）就是初始化执行类型程序运行时间和 END 处理时间的总和。

如果使用了多个初始化执行程序，那么就是所有这些程序被执行的执行时间的总和。

（b） QnACPU 检测初始化扫描时间，并将结果保存在特殊寄存器中（SD522，SD523）。*1

因此初始化时间就可以通过监视 SD522 和 SD523 特殊寄存器来检查了。



如果 SD522 的值是“3”，SD523 的值是“400”，那么初始化扫描时间就是 3.4ms。

（5）初始化执行时间监视器

(a) 初始化执行程序的执行时间可以通过这个定时器（没有缺省设置）来监视。如果需要这种监视，可在参数“PC RAS”设置中，在 10ms 到 2000ms 范围内设置定时器。（设置单位：10ms）

(b) 当低速执行类型程序存在时，请将此值设定成大于实际初始扫描时间与低速执行类型程序的运行时间之和。

(b) 如果初始化执行程序的实际执行时间超过了这个定时器设置，就会发生“WDT 错误”，QnACPU 运行就会停止。

要点

（1）*1：保存在特殊寄存器中的初始化扫描时间的精确度是±0.1ms。即使是在顺控程序中执行了监视定时器复位指令（WDT），初始化扫描时间计数仍将继续。

(2) 当为监视定时器设置指定初始化执行时间时，其计数值将会有10ms 的误差。

因此，如果初始化扫描时间在如下范围内：10ms〈t〈20ms，那么 10ms 的监视定时器设置（t）将会产生一个“WDT 错误”。

3.2.2 扫描执行程序

（1）定义

（a）从初始化执行程序执行之后的扫描开始，扫描执行程序在每次扫描时执行一次。

（b）这种程序执行类型是在程序设置参数中被指定为“扫描”的。

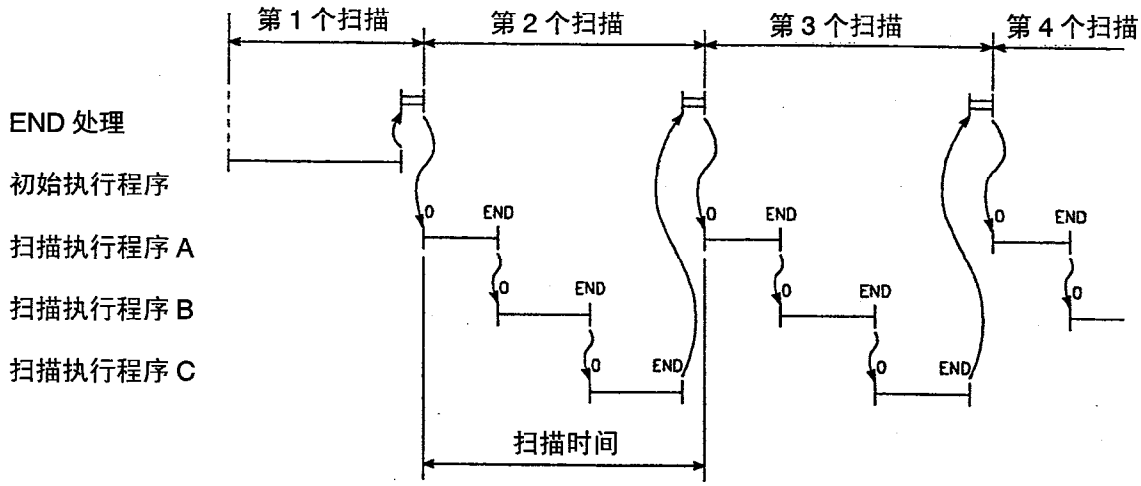
（2）执行多个扫描执行程序

多个扫描执行程序被按照参数中的程序设置升序一个一个地执行。

（3）END 处理

当所有的扫描执行程序完成后，进行 END 处理，并且在从第一个扫描执行程序开始执行。

通过在扫描执行程序结尾处加上一个 COM 指令，可在每一个程序之后进行 END 处理（通用数据处理、链接刷新）。



（4）恒定扫描设置*1

当指定了恒定扫描时，扫描执行程序在每个指定的恒定扫描周期中执行。

注释

1）*1：“恒定扫描”的功能是以固定的时间间隔重复执行扫描类型程序。有关详细资料，请参见使用的 CPU 的用户手册。

要点

1）关于在一个扫描执行程序进行中执行一个中断程序时的变址寄存器处理的详细资料，请参见 4.6 节。

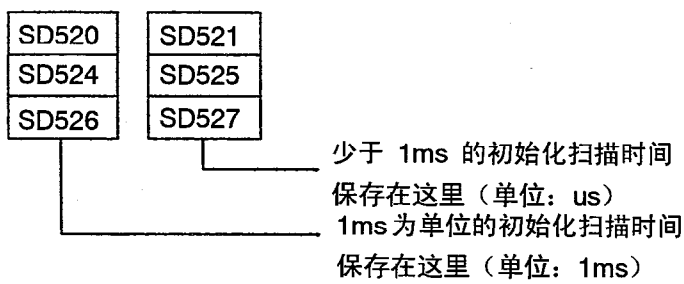
（5） 扫描时间

"扫描时间"是扫描类型程序执行和 END 处理所需的总时间。

如果使用了多个扫描执行程序，那么"扫描时间"就是执行所有这些程序所需的总时间。

QnACPU 检测扫描时间的"当前值"、"最小值"和"最大值"，并将结果保存在特殊寄存器中（SD520，AS521，SD524-SD527）。*

因此初始化扫描时间就可以通过监视 SD520、SD521 和 SD524-SD527 特殊寄存器来检查了。



如果 SD520 的值是"3"，SD521 的值是"400"，那么初始化扫描时间就是 3.4ms。

（6）WDT（监视定时器）

这是监视扫描时间的定时器，它的缺省值是 200ms。

这个 WDT 可以在参数的 PC RAS 被设置在 10ms 到 2000ms 的范围内。（设置单位：10ms）

如果使用了一个低速执行程序，那么 WDT 设置值应该指定为大于扫描时间与低速执行程序时间的总和。

如果扫描时间（扫描执行程序+低速执行程序的执行时间）超过了 WDT

设置值，就会出现一个“WDT 错误”，并且 QnACPU 运行会停止。

要点

（1）*1：保存在特殊寄存器中的扫描时间的精确度是±0.1ms。

即使是在顺控程序中执行了监视定时器复位指令（WDT），扫描时间计数仍将继续。

（2）WDT 测量的误差是 10ms。

因此，如果扫描时间在如下范围内：10ms〈t〈20ms，那么 10ms 的 WDT 设置（t）将会产生一个“WDT 错误”。

3.2.3 低速执行程序

（1）定义

（a）低速执行程序只在“恒定扫描剩余时间”期间或被指定的低速执行程序的执行周期里被执行。

- 在强调控制精度时，可通过在参数设置的 PC RAS 项目中指定恒定扫描时间（设置范围：5~2000ms，设置单位：5ms）来保持一定的扫描周期。
- 为了确保每一次扫描的低速执行程序的执行时间，在参数设定的“PC RAS”项目中指定一个低速执行程序执行时间。（设置范围：1-2000ms，设置单位：1ms）
- 为了执行低速执行程序，必须指定以下设置中的一个：“恒定扫描时间”或“低速执行程序执行时间”。

（b）低速执行程序的执行类型在参数的程序设置中指定为“低速” 。

（c）低速执行程序类型用于不需要在每一个扫描中都执行的程序，例如，打印机输出程序。

（2）执行多个低速执行程序

当使用了多个低速执行程序时，它们被按照升序（参数中的程序设置）一个接一个地执行。

（3）在一个扫描中低速执行程序的执行时间

（a）如果在一次扫描中所有的低速执行程序操作都完成了而还有剩余时间，那么之后执行的处理取决于特殊寄存器 SM330 中的设置状态和低速执行类型程序的执行条件。

- 异步方式（SM330 = 关闭）

  ：在这种方式下，低速执行类型程序继续在剩余时间里运行。

- 同步方式（SM330 = 打开）

  ：在这种方式下，即使有剩余时间，低速执行程序运行也不再继续，而是从下一个扫描又开始运行。

低速执行类型程序的运行方式	SM330 的设置状态	低速执行类型程序的执行条件	
		当设置了“恒定扫描时间”时	当设置了“低速执行程序执行时间”时
异步方式	关	低速执行类型程序被再次执行*1。	低速执行类型程序被再次执行*2。
同步方式	开	产生恒定扫描待机时间*3。	扫描执行类型程序运行开始*4。

*1 如果指定了“恒定扫描时间”，那么低速程序将在恒定扫描的剩余时间里执行。

因此，低速执行程序的执行时间在不同的扫描中是不同的。

但当恒定扫描的剩余时间是 2ms 或更少时，低速执行程序不会被执行，因此，指定的“恒定扫描时间”提供的剩余时间应该大于 2ms。

*2 如果指定了“低速执行程序执行时间”，那么低速执行程序将依照时间设置执行。因此，扫描时间在不同的扫描中是不同的。

*3 如果指定了“恒定扫描时间”，那么在低速 END 处理之后的剩余时间就是待机时间，在恒定扫描时间过去之后，扫描执行类型程序开始执行。

恒定扫描待机时间

= （恒定扫描设置时间）-（扫描时间）

-（低速扫描时间）

这意味着在每个扫描中，扫描时间是固定不变的。

然而，如果恒定扫描之后的剩余时间小于 2ms，则低速执行类型程序将不会被执行。如果使用低速执行类型的程序，就设置恒定扫描时间使剩余时间是 2ms 或更长。

*4 如果指定了“低速执行程序执行时间”，那么扫描执行类型的程序开始运行，而忽略低速 END 处理完成后的剩余时间。

低速程序执行时间中的剩余时间

（忽略）

= （低速程序执行时间的设置时间）—（低速扫描时间）

这意味着在每个扫描时，扫描时间是不同的。

（b） 如果低速执行程序无法在恒定扫描剩余时间或低速执行程序的执行时间进行，那么程序执行将被暂时停止，剩下的程序将在下一个扫描中执行。

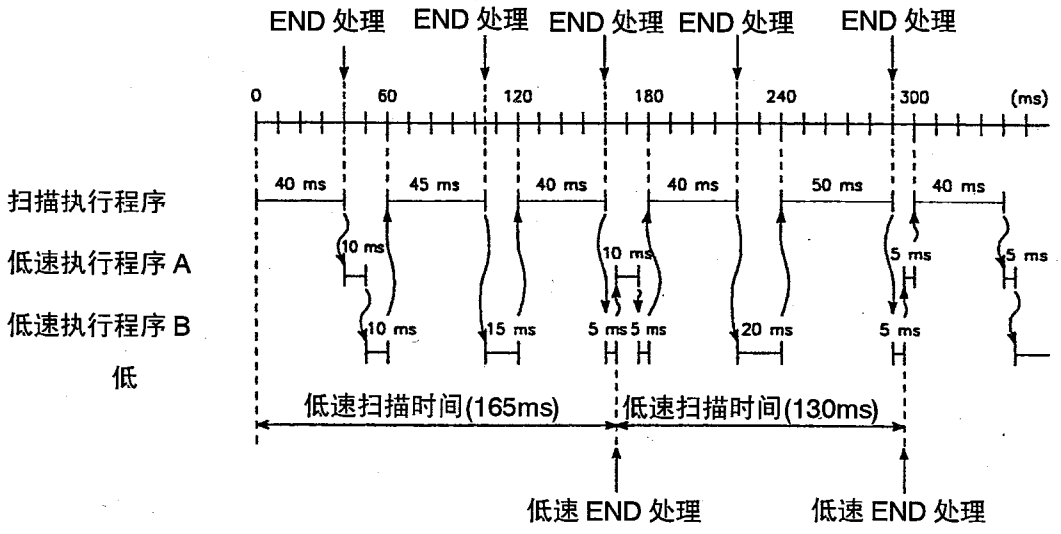
要点
（1）关于在从一个扫描执行程序切换到一个低速执行程序时，变址寄存器处理的详细资料请参见 4.6 节。 （2）关于在一个低速执行程序进行过程中执行一个中断程序时，变址寄存器处理的详细资料请参见 4.6 节。 （3）“低速执行程序执行时间”设置应该是使得[扫描时间]+[低速执行程序的执行时间]的总和小于 WDT 设置值。 （4）COM 指令不能用于低速程序中。

异步方式

（1） “恒定扫描时间设置”

在下列条件下低速执行程序的运行如下所示。

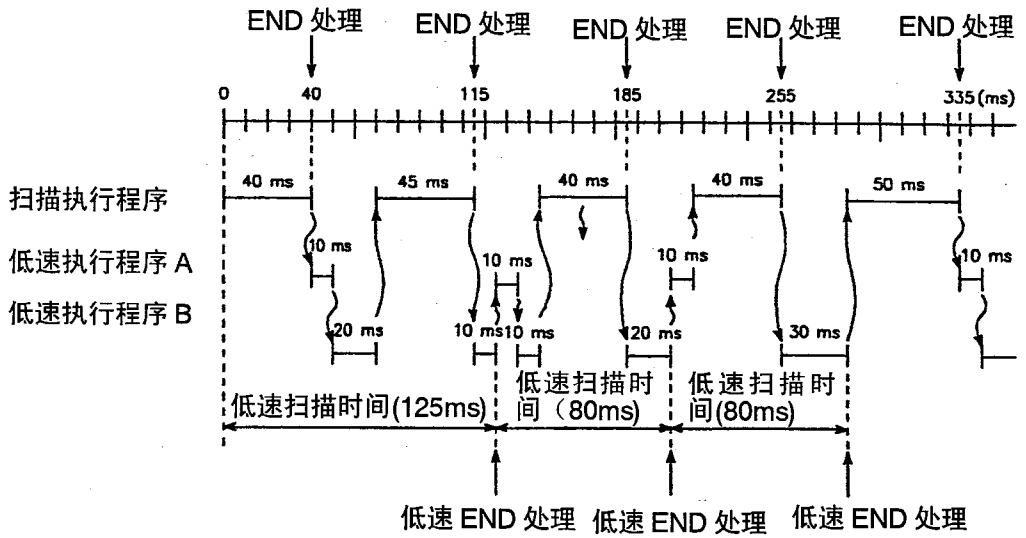
- 恒定扫描时间：60ms
- 总的扫描执行程序时间 ：40ms 到 50ms
- 低速执行程序 A 的执行时间 ：10ms
- 低速执行程序 B 的执行时间 ：30ms
- END 处理 ：0ms（采用 0ms 是为了简化说明）
- 低速 END 处理 ：0ms（采用 0ms 是为了简化说明）



（2）“低速执行程序执行时间”设置

在下列条件下的低速执行程序的运行如下所示。

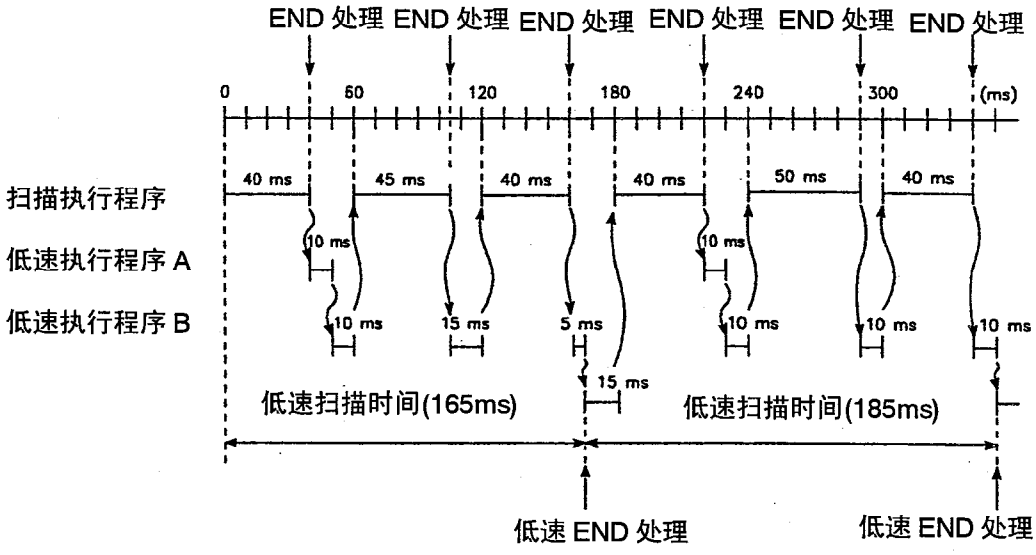
- 低速执行程序执行时间 ：30ms
- 总的扫描执行程序时间 ：40ms 到 50ms
- 低速执行程序 A 的执行时间 ：10ms
- 低速执行程序 B 的执行时间 ：30ms
- END 处理 ：0ms（采用 0ms 是为了简化说明）
- 低速 END 处理 ：0ms（采用 0ms 是为了简化说明）



同步方式

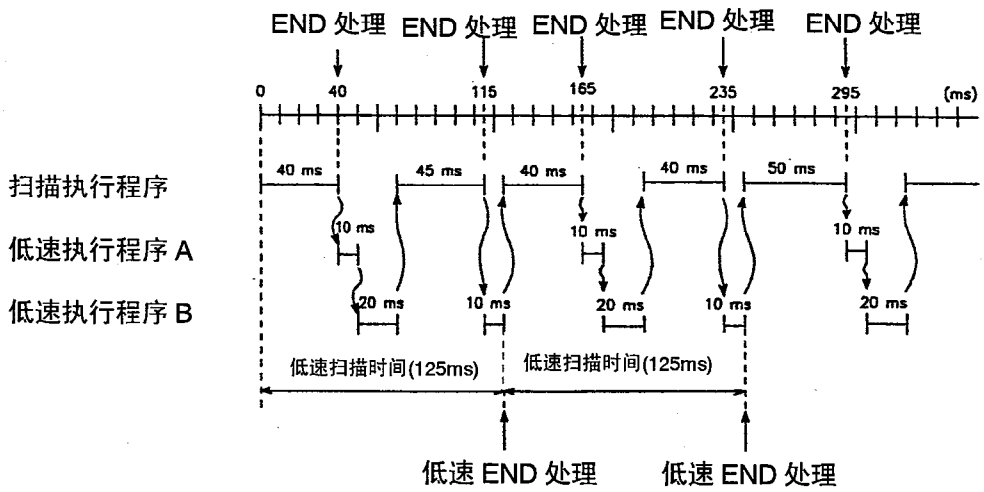
（1） “恒定扫描时间设置”
在下列条件下低速执行程序的运行如下所示。

- 恒定扫描时间 ：60ms
- 总的扫描执行程序时间 ：40ms 到 50ms
- 低速执行程序 A 的执行时间 ：10ms
- 低速执行程序 B 的执行时间 ：30ms
- END 处理 ：0ms（采用 0ms 是为了简化说明）
- 低速 END 处理 ：0ms（采用 0ms 是为了简化说明）



（2）“低速执行程序执行时间”设置
在下列条件下的低速执行程序的运行如下所示。

- 低速执行程序执行时间 ：30ms
- 总的扫描执行程序时间 ：40ms 到 50ms
- 低速执行程序 A 的执行时间 ：10ms
- 低速执行程序 B 的执行时间 ：30ms
- END 处理 ：0ms（采用 0ms 是为了简化说明）
- 低速 END 处理 ：0ms（采用 0ms 是为了简化说明）



（4）END 处理

低速 END 处理在所有的低速执行程序已经执行完后发生。低速处理包括下列一些项目：

- 低速执行程序的特殊继电器/特殊寄存器设置。
- 低速执行程序的运行中写入。
- 低速扫描计时。
- 低速执行程序的监视定时器设置。

当低速 END 处理完成后，又从第一个低速执行程序开始执行。

> 要点
>
> （1）在低速执行程序的执行期间，[最大指令处理时间]+[低速 END 处理时间]可能会与“恒定扫描时间”有出入。

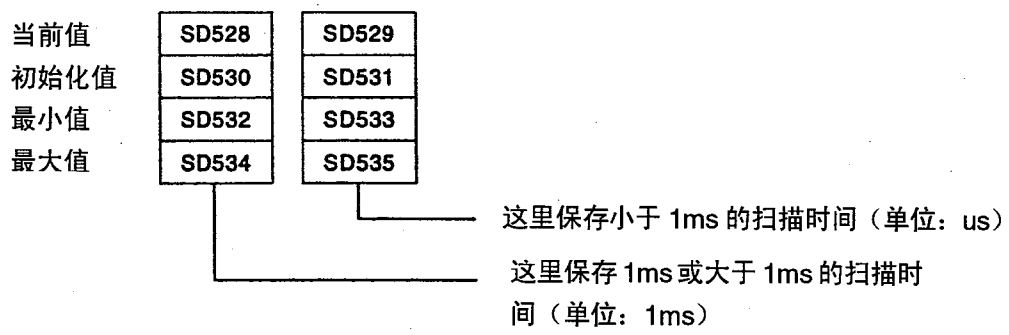
（5）低速扫描时间

（a）“低速扫描时间”就是低速执行程序的执行和低速 END 处理所需的总时间。

如果使用了多个低速执行程序，“低速扫描时间”就是执行所有程序所需的时间加上低速 END 处理时间的总时间。

（b）低速扫描时间由 QnACPU 测量，结果保存在特殊寄存器中（SD528-SD535）。*1

因此，低速扫描时间可以由通过监测 SD528-SD535 特殊寄存器来校验。



如果 SD528 的值是“50”，SD529 的值是“400”，那么低速扫描时间就是 50.4ms。

(6). 低速执行时间监视

低速执行程序的执行周期可以用这个定时器（没有缺省设置）来监视。如果想要这样的监视，那么就在参数的 PC RAS 设置在 10ms 到 2000ms 的范围里指定定时器（设置单位：10ms）。

如果低速执行程序的实际执行时间超过了这个定时器设置，就会出现一个“PRG TIME OVER”错误。（但是即使如此，QnACPU 的运行也不停止。）

要点

（1）*1：存储在特殊寄存器中的扫描时间的精确度是±0.1ms。
即使在顺控程序中执行了一条监视定时器复位指令（WDT），扫描时间计数仍将继续。

（2） 低速执行计时发生在低速 END 处理时。因此，如果低速执行监视器时间（t）指定为“100ms”并且测量到的低速扫描时间（在低速 END 处理时）超过了 100ms，会发生“PRG TIME OVER”错误。

3.2.4 待机程序

（1）定义

（a）待机程序是只在被要求的时候才执行的程序。

（b）待机程序用于以下应用中。

1）将程序库化

子程序和中断程序被转换成待机程序，以便从主程序中分离出来单独管理。

2）改变程序的设定

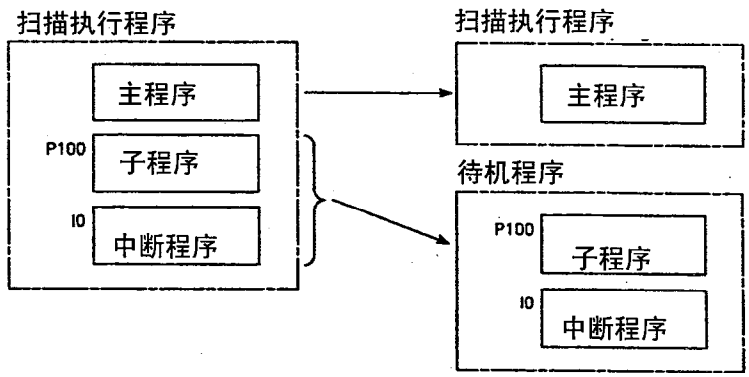
主程序被注册成待机程序，然后需要的程序被转换成扫描执行程序来执行。不需要的程序被转换成待机程序。

（2）将程序放入库中

（a）将程序放入库中

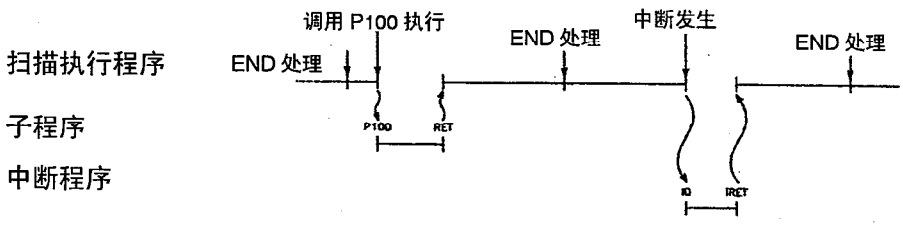
1）用于管理从主程序中分离出来的子程序和中断程序。

可以为一个待机程序创建多个子程序和中断程序。



2）当待机程序执行完成后，就返回到激活待机程序的程序进行处理。

下面所示的是，当一个待机程序的子程序和中断程序被执行时发生的操作。



要点

（1）因为当前值更新和触点的开/关切换是发生在 OUT T[]指令的，所以定时器不能用在待机程序中。

（2）请在子程序被设定为待机程序时，使用通用指针。凡是使用了局部指针的待机程序都是不可执行的。关于通用指针与局部指针的详细资料请参见 4.9.1 节。

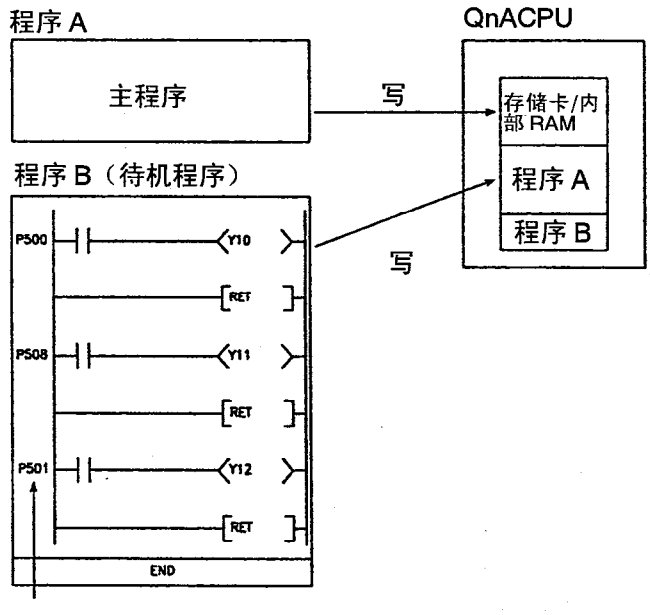
（b）将复数的子程序合并成一个程序时

1）按照顺序创建子程序，从待机程序的第 0 步开始。在子程序的结尾处需要有一条 END 指令。

2）因为对于子程序的创建顺序没有限制，因此当创建多个子程序的时候，不需要将指针数按照升序分配。

3）只使用通用指针。*

带有通用指针的子程序可以被 QnACPU 执行的所有程序调用。



使用通用指针。*

（子程序不需要按照升序创建。）

4）当在子程序中用到了局部软元件时，操作就依照调用子程序的主程序（执行调用/E 调用指令的程序）的局部软元件的值执行。在执行一个待机程序的子程序之前和之后，不保存或复位局部软元件的值。

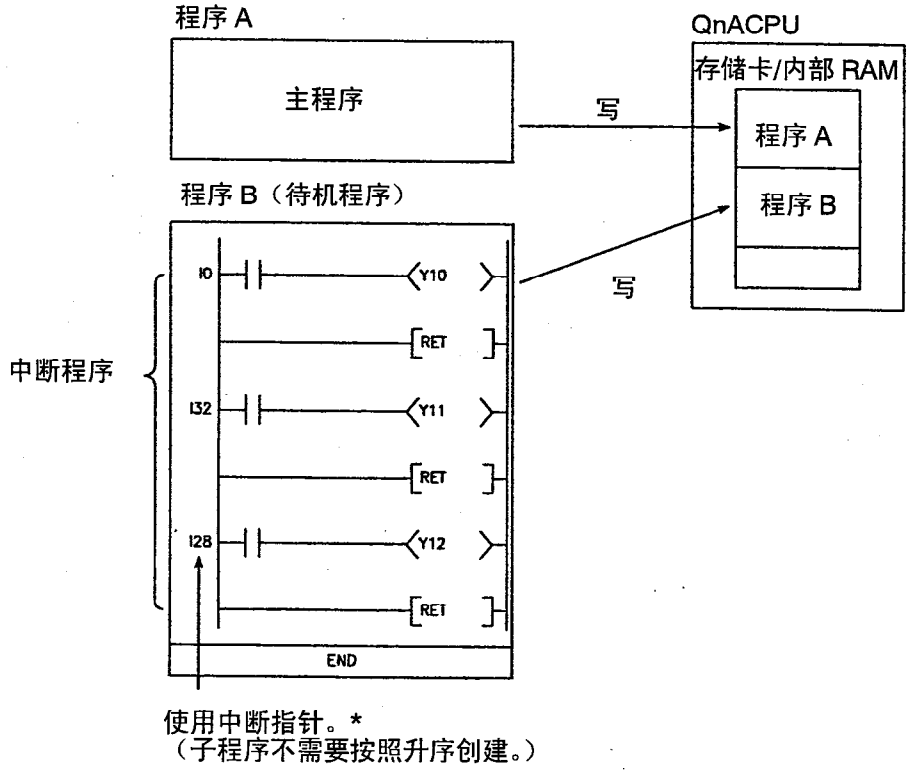
注释

1）*：关于通用指针的详细资料请参见 4.9.2 节。

（c）将复数个中断程序合并成一个程序

1）从待机程序的第 0 步开始，按照顺序创建中断程序。在中断程序的结尾处要有一条 END 指令。

2）因为对于中断程序的创建顺序没有限制，因此当创建多个中断程序的时候，不需要按照升序分配指针。



注释

1）*：关于中断指针的详细资料请参见 4.10 节。

(3). 程序的启动切换

（a）此功能事先创建对应于所有系统的多个程序，只实行需要的程序。通过参数设置指定为“待机”程序的程序可以被转换成扫描执行程序来执行。

QnACPU 使用以下指令来转换程序的类型：

1）PSCAN：将一个待机程序转换成一个扫描执行程序。

2）PLOW：将一个待机程序转换成一个低速执行程序。

3）PSTOP：将一个扫描程序或低速执行程序转换成一个待机程序。

4）POFF：将一个扫描执行程序或低速执行程序转换成一个待机程序。（在输出被关掉之后，进行待机程序的切换。）

<table>
<tr><th>执行指令
改变前的执行类型</th><th>PSCAN</th><th>PSTOP</th><th>POFF</th><th>PLOW</th></tr>
<tr><td>扫描执行类型</td><td>没有改变—保持扫描执行类型</td><td rowspan="2">变成待机类型</td><td rowspan="2">在下一个扫描时输出关闭。
之后的下一个扫描变成待机类型。</td><td rowspan="3">变成低速类型。</td></tr>
<tr><td>初始化执行类型</td><td rowspan="2">变成扫描执行类型</td></tr>
<tr><td>待机类型</td><td>没有改变—保持待机类型</td><td>没有处理。</td></tr>
<tr><td>低速执行类型</td><td>低速执行类型执行停止：从下一个扫描开始变成扫描执行类型。（从第 0 步执行）</td><td>低速执行类型执行停止：从下一个扫描开始变成待机类型。</td><td>低速类型执行停止，并且在下一个扫描关闭输出。之后，从下一个扫描变成待机类型。</td><td>没有改变—保持低速类型。</td></tr>
</table>

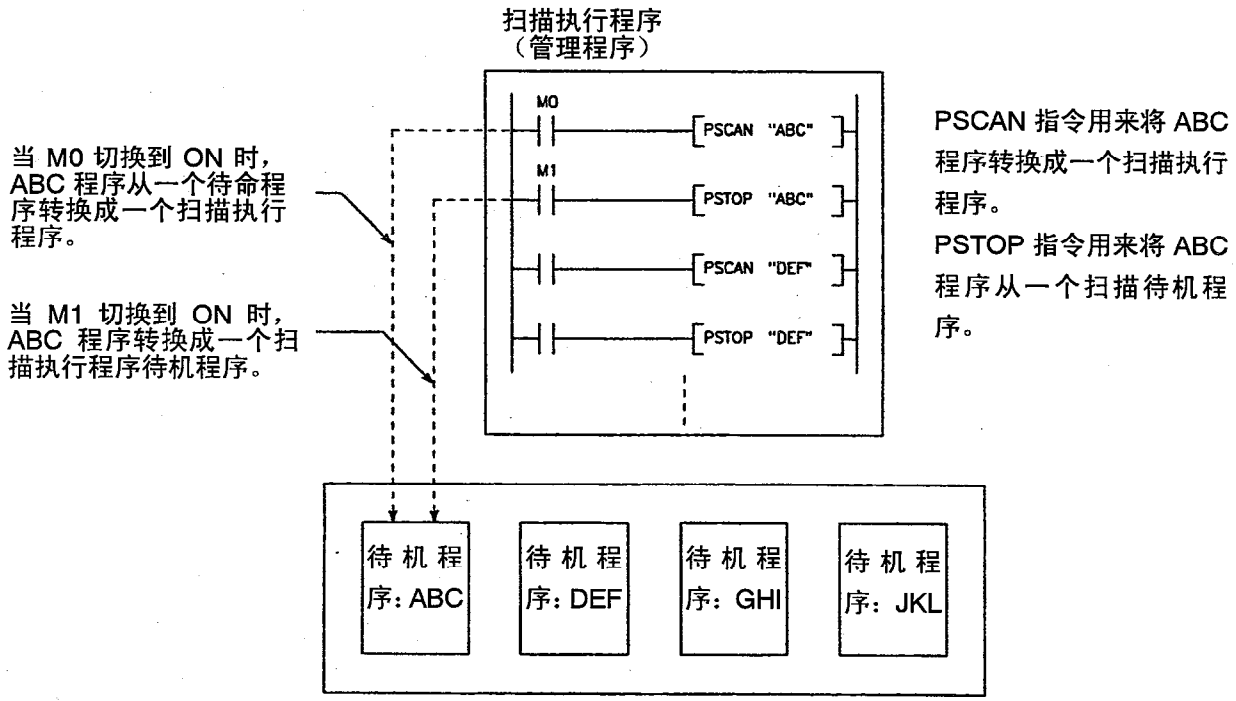
（b）下面的方法可以用来转变一个要被执行的程序。

1）从一个单一的管理程序中选择要执行的程序：

将一个每次执行的扫描执行程序作为管理程序使用，将符合指定条件的待机程序转换成一个扫描执行程序，并执行。

不需要的扫描执行程序可以被此管理程序转换成待机程序。

下面说明了当“ABC”、“DEF”、“GHI”和“JKL”待机程序（在一个单一管理的程序中）被转换时发生的操作。

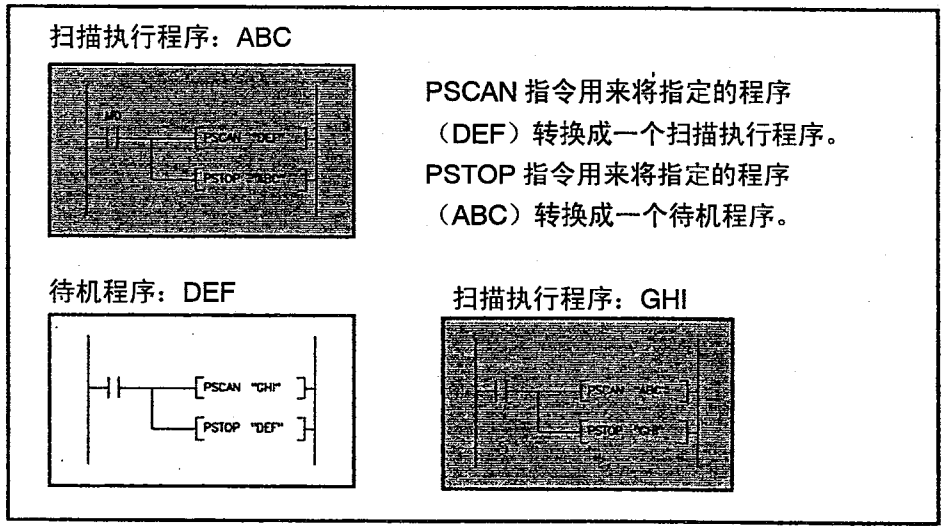


2） 将正在被执行的扫描执行程序转换成另一种类型的程序：

对于正在被执行的扫描执行程序，将要执行的下一个程序从一个待机程序转换成一个扫描执行程序。

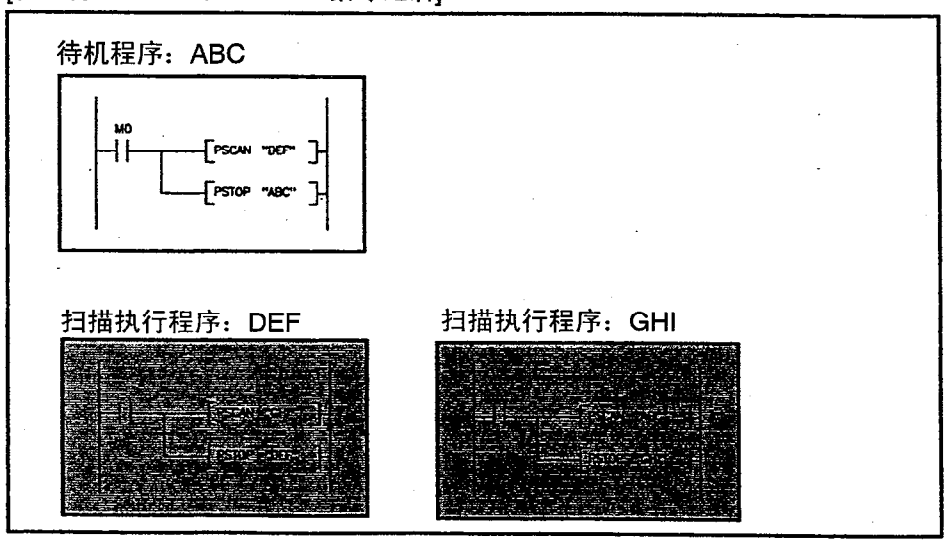
在下面的图解说明中，ABC 与 GHI 程序被指定为扫描执行程序，而 DEF 被指定为待机程序。图解说明显示了当 ABC 和 DEF 程序类型在条件满足时转换所发生的操作。

[在执行 PSCAN 和 PSTOP 指令之前]

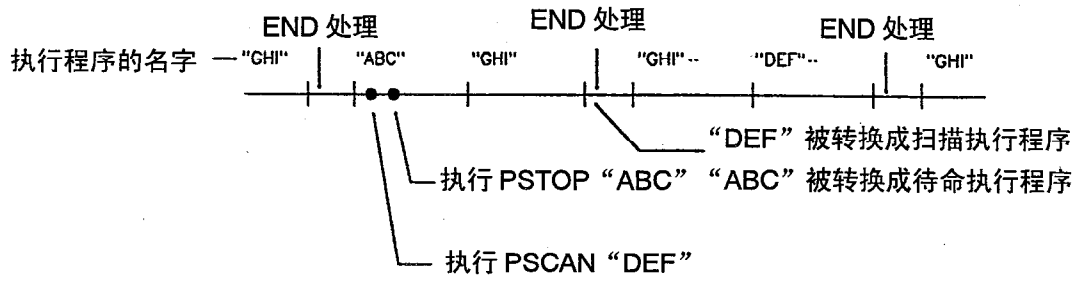


“M0”被切换成 ON

[在执行 PSCAN 和 PSTOP 指令之后]



（c）由于使用 PSCAN 和 PSTOP 指令实现的程序执行类型转换发生在 END 处理时，所以在程序执行期间这样的转换是不可能的。当为同一个扫描中的同一个程序设置了不同的执行类型时，执行类型由最后一个被执行的执行切换命令所指定。



注释

1）*：GHI 和 DEF 程序的执行顺序是由参数中的程序设置项目决定的。

3.3 输入/输出处理与响应延迟

刷新类型的输入/输出处理形式是 QnACPU 的一个特点，在这种形式中，在 END 处理中会发生与输入/输出模块的批量通信。

通过使用顺控程序的直接存取输入/输出命令可以实现一种直接的通信形式，这种直接的通信形式能够在顺控程序指令被执行时直接与输入/输出模块通信。

关于直接输入与直接输出的详细资料请分别参见 4.2.1 和 4.2.2 节。

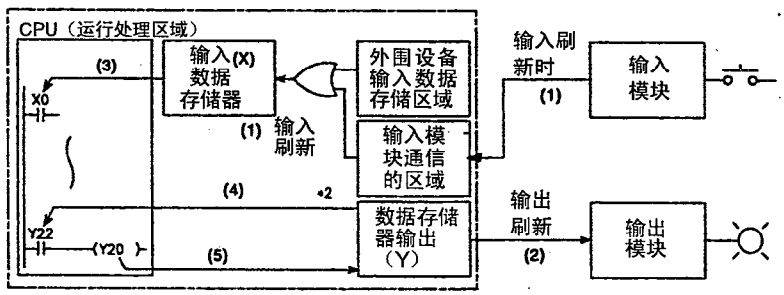
3.3.1 刷新模式

(1) 定义

刷新模式就是 CPU，与输入/输出模块在 END 处理时进行批量通信的模式。

（a） 输入模块的 ON/OFF 情报在 END 处理时被批量存入输入数据存储器中，在顺控程序被执行时被使用。

（b） 作为顺控程序的运行结果之一的输出（Y）在程序运行时被暂时存放于 QnA CPU 内部的输出数据储存器中，然后在 END 处理时被批量地输出到输出模块去。



. 输入刷新：
输入数据在 END 处理时被从输入模块批量读入（(1)），并且经过与外围设备输入区域中的值 OR 运算后保存在输入（X）数据存储器。

. 输出刷新：
在 END 处理时，输出（Y）数据存储器中的数据被批量输出给输出模块（(2)）。

. 当执行了输入触点指令时：
输入信息从输入（X）数据存储器中读入（(3)），并且依此执行顺控程序。

. 当执行了输出触点指令时：
输出信息从输出（Y）数据存储器中读入（(4)），并且依此执行顺控程序。

. 当执行了输出 OUT 指令时：
顺控程序的运行结果（(5)）被保存在输出（Y）数据存储器中。

图 3.5 刷新模式下的输入/输出信息流程

注释

1）*1：参见 3.3.2 节，项目 1)。

2）*2：参见 3.3.2 节，项目 2)。

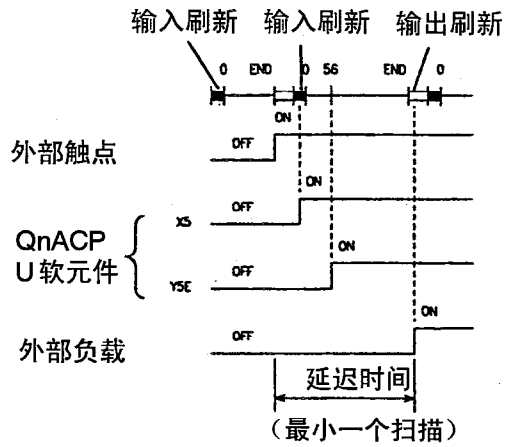
(2) 响应延迟

输入模块输入时间变化可以导致多达2个扫描的输出响应延迟。（参见图 3.6）

梯形图的例子

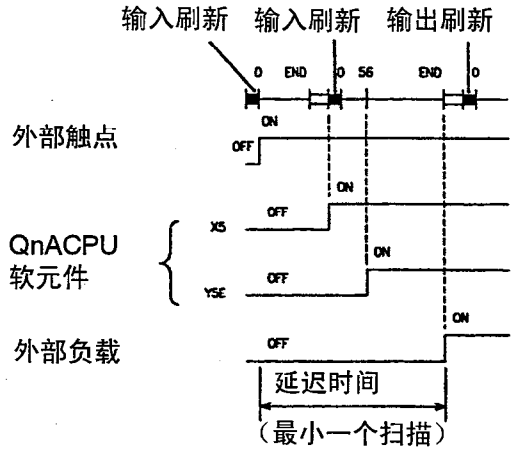
切换 Y5E 输出到 ON 以响应一个 X5 输入 ON 的梯形图。

最早的 Y5E ON



如果在刷新操作之前外部触点立即切换到 ON，就会发生最早的 Y5E ON。此例中 X5 在输入刷新时切换到 ON，Y5E 在第 56 步切换到 ON，并且外部负载在 END 指令执行之后的输出刷新时切换到 ON。在这种情况下，外部触点 ON 和外部负载 ON 之间的时间延迟是 1 个扫描。

最晚的 Y5E ON



如果在刷新操作之后外部触点才立即切换到 ON，就会发生最晚的 Y5E ON。此例中 X5 在下一个输入刷新时切换到 ON，Y5E 在第 56 步切换到 ON，并且外部负载在 END 指令执行之后的输出刷新时切换到 ON。

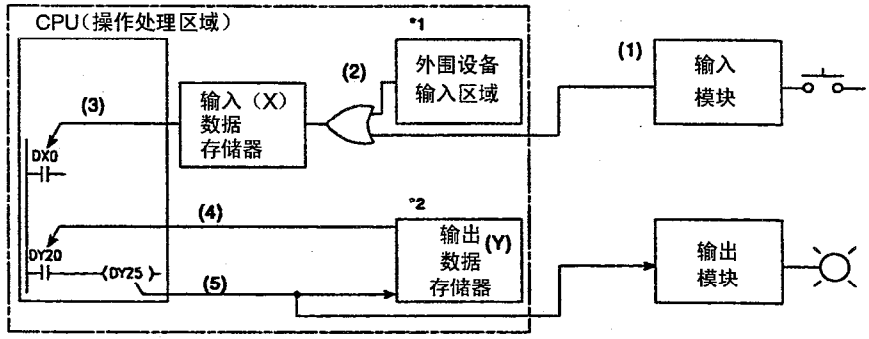
在这种情况下，外部触点 ON 和外部负载 ON 之间的时间延迟是 2 个扫描。

图 3.6 响应输入“X”的改变的输出“Y”的变化

3.3.2 直接模式

(1) 定义

在直接模式下，当执行顺控程序指令时进行与输入/输出模块的通信。对于 QnACPU，可以通过直接存取输入（DX）和直接存取输出（DY）来执行直接模式的 I/O 处理。



. 当执行了输入触点指令时：

对输入模块的输入信息（(1)）以及外围设备输入区域的输入信息（(2)）执行一次 OR 运算，并且将结果保存在输入（X）数据存储器中。这个数据随后用作顺控程序的输入信息（(3)）。

. 当执行了输出触点指令时：

输出信息（(4)）从输出（Y）数据存储器读出，并且执行顺控程序。

. 当执行了输出 OUT 指令时：

顺控程序的运行结果（(5)）输出给输出模块，并保存在输出（Y）数据存储器中。

图 3.7 直接模式下的输入/输出信息流程

注释

1）*1：外围设备输入区域可以通过下列方法切换为 ON 和 OFF：
- 外围设备的测试操作。
- MELSECNET（/B）数据连接系统的连接刷新。
- MELSECNET /10 网络系统的网络刷新。
- 从一个串行通信模块写入。
- 自动刷新 CC-LINK。

2）*2：输出（Y）数据存储器可以通过下列方法切换为 ON 和 OFF：
- 外围设备的测试操作。
- MELSECNET（/B）数据连接系统的连接刷新。
- MELSECNET /10 网络系统的网络刷新。
- 从一个串行通信模块写入。
- 自动刷新 MELSECNET/MINI 或 CC-LINK。

要点

(1) 当输入（X）规定为用于 MELSECNET/MINI 或 CC-LINK 的自动刷新设置的接收数据存储软元件时，使用比与装在主基板或扩展基板上的模块一起使用的 I/O 编号大的 I/O 编号。
如果用于接收数据存储软元件的 I/O 编号在与安装在主基板或扩展基板上的模块一起使用的 I/O 编号范围之内，CPU 模块将输入来自模块的输入 ON/OFF 数据和 MELSECNET/MINI 或 CC-LINK 自动刷新 ON/OFF 数据。
因此，CPU 模块会产生一个输入（X）错误。

(2) 在使用顺控程序指令开关输入（X）时，使用如下所示的输入编号。

. 与安装在主基板或扩展基板上的模块一起使用的输入编号。
. 与 MELSECNET（/B）一起使用的输入编号。
. 与 MELSECNET/MINI 或 CC-LINK 一起使用的输入编号。

当你使用与上述所示的相同输入编号时，输入到输入模块或 MELSECNET（/B）的刷新或者 MELSECNET/MINI 或 CC-LINK 的自动刷新的数据，将覆盖顺控程序指令的 ON/OFF 状态。

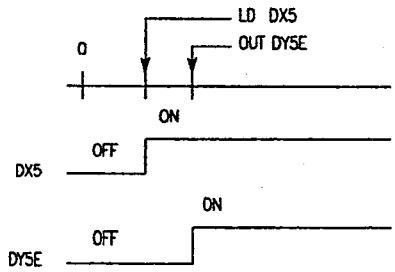
(2) 响应延迟

输入模块的改变可以引起达 1 个扫描的输出响应延迟。（参见图 3.8）

梯形图的例子

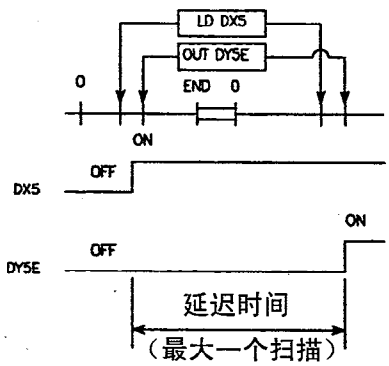
切换 DY5E 输出到 ON 以响应一个 DX5 输入 ON 的梯形图。

最早的 DY5E ON



如果在第 55 步操作之前，DX5 输入立即切换到 ON，就会发生最早的 DY5E ON。也就是说，当第 55 步的 LDDX5 指令被执行时 DX5 是在 ON 状态，那么 DY5E 将在该扫描中切换成 ON。因此，在这种情况下，在 DX5 输入 ON 与 DY5E 输出 ON 之间的最小时间延迟。

最晚的 DY5E ON



如果在第 55 步操作之后，DX5 输入立即切换到 ON，就会发生最慢的 DY5E ON。在这种情况下，DY5E 输出将在下一个扫描中切换成 ON。因此，在这种情况下在 DX5 输入 ON 与 DY5E 输出 ON 之间的最大时间延迟（1 个扫描）。

图 3.8 响应输入“X”的变化的输出“Y”的变化

3.4 在顺控程序中可用的数字值

在 QnACPU 中的数字的和字母的数据是用“0”（OFF）和“1”（ON）数字来表示的。

这种表示方法称为“二进制码”（BIN）。

在十六进制码（HEX）的表示方法中，BIN 数据以 4 个比特位为单位来表示，而且在 QnACPU 中也可以时间 BCD 表示方法（二-十进制码）。

BIN、HEX、BCD 和十进制（DEC）符号的数字表示如下面表 3.1 所示。

表 3.1 BIN、HEX、BCD 和十进制数字表示

DEC（十进制）	HEX（十六进制）	BIN（二进制）	SCD（二-十进制码）
0	0	0	0
1	1	1	1
2	2	10	10
3	3	11	11
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
9	9	1001	1001
10	A	1010	10000
11	B	1011	10001
12	C	1100	10010
13	D	1101	10011
14	E	1110	10100
15	F	1111	10110
16	10	10000	10110
17	11	10001	10111
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
47	2F	101111	1000111

也可能会用到单精度浮点十进制实数。（参见 3.4.4 节）

(1) 到 QnACPU 的外部数字输入

当从一个外部信号源（数字开关等）为 QnACPU 指定数字设置时，可以指定一个 BCD（二-十进制码）设置，它与十进制设置一样。

然而，因为 BCD 方式意味着以与十进制表示相同的方法处理 BIN 表达式，所以基于这样的数值的 QnACPU 操作将与由指定值规定的操作不同。

我们提供了一条 BIN 指令可以使 QnACPU 将 BCD 输入数据转换成 BIN 数据在 QnACPU 中使用。

可以在顺控程序中创建将数字数据转换成 BIN 数据的程序，这样就可以允许从一个外部信号源来指定数字设置，而不需要考虑相应的 BIN 值。

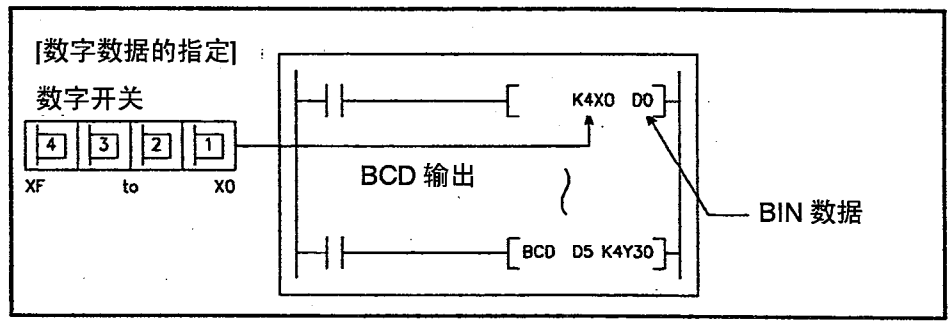


图 3.9 数字切换数据输入到 QnACPU

(2) 从 QnACPU 向外输出数字

数字显示器可以用于显示从 QnACPU 输出的数字数据。然而，因为 QnACPU 使用的是 BIN 数据，所以它不能直接在数码显示器上显示。因此，提供了一条 BCD 指令使 QnACPU 能将 BIN 数据转换成 BCD 数据。可以在顺控程序中创建将 BIN 数据转换成 BCD 数据的程序，这样就可以以十进制数据的方式显示输出数据了。

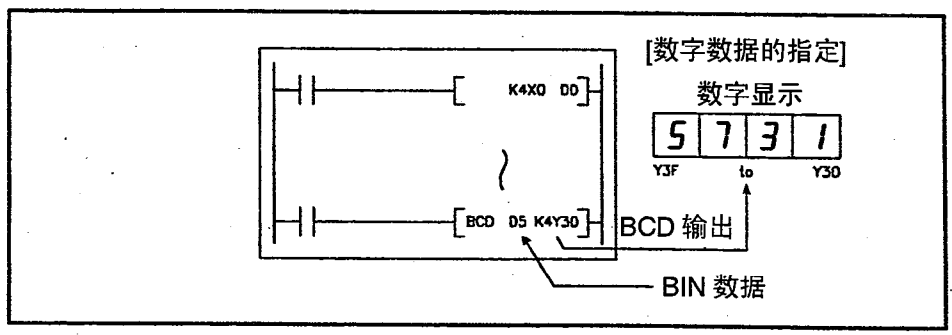


图 3.10 来自 QnACPU 的数据的数字显示

3.4.1 BIN（二进制码）

(1) 二进制码

在二进制码下，数值采用“0”（OFF）和“1”（ON）两个数字来表示。当十进制系统中进行计算时，在 9 之后（8-9-10）会进位到“十”这一行数字上。在二进制系统中，这种情况会发生在 1 之后（0-1-10）。因此，二进制的“10”表示十进制的“2”。二进制值以及它们各自的十进制值如下面图 3.2 所示。

表 3.2 二进制与十进制数值的比较

DEC（十进制）	BIN（二进制）	
0	0000	
1	0001	进位
2	0010	
3	0011	进位
4	0100	
5	0101	
6	0110	
7	0111	
8	1000	进位
9	1001	
10	1010	
11	1011	

(2) 二进制数值的表示

QnACPU 的寄存器（数据寄存器、连接寄存器等）是由 16 个比特位组成的，寄存器的每一位分配一个“2^n”的值。

最高位用来区别“正”和“负”值。

1. 当最高位是“0”时...正值
2. 当最高位是“1”时...负值

QnACPU 的各寄存器的数字表示如下面图 3.11 所示。

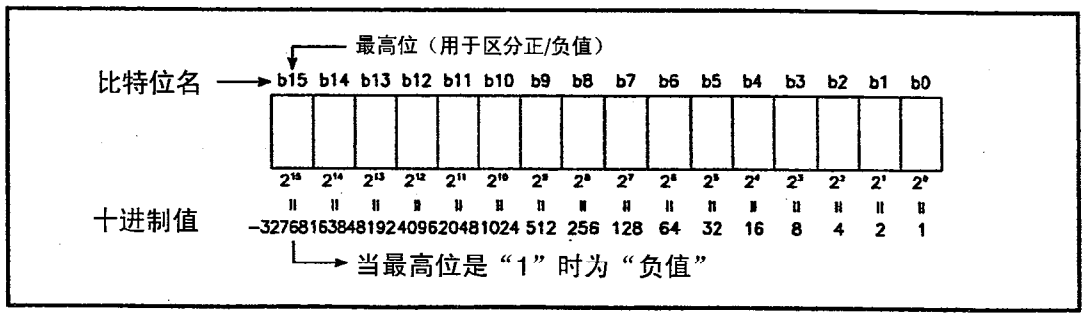


图 3.11 QnACPU 的各寄存器的数值表示

(a) QnACPU 可用的数值数据

如图 3.11 所示，数值显示的范围是-32768 到 32767。因此，在这个范围之内的数值数据可以被保存到 QnACPU 寄存器中。

3.4.2 HEX（十六进制）

(1) 十六进制符号

在十六进制系统中，一个数字表示二进制数据的 4 个比特位。二进制数据的 4 个比特位可以表示 16 个值（0-15）。

在十六进制系统中，从 0 到 15 的值用一个数字来表示。在“9”之后使用字母字符来表示，在“F”之后发生进位，如下所示：

二进制、十六进制和十进制数字表示的比较如下面的表 3.3 所示。

表 3.3 BIN、HEX 和 DEC 数字表示的比较

DEC（十进制）	HEX（十六进制）	BIN（二进制）
0	0	0
1	1	1
2	2	10
3	3	11
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
.	.	.
.	.	.
.	.	.
.	.	.
.	.	.
47	2F	101111

(2) 十六进制数字的表示

QnACPU 的寄存器（数据寄存器、连接寄存器等）是由 16 个比特位组成的。因此，用十六进制码表示时，可以保存的数字范围是 0 到 $FFFF_H$。

3.4.3 BCD（二-十进制码）

(1) BCD 符号

BCD 数字的表示是具有同十进制系统一样的进位形式的二进制表示。当采用十六进制系统时，尽管 BCD 系统中不使用 A-F 字母字符，但是 BCD 表示等效于 4 个二进制位。

二进制、BCD 和十进制数字表示比较如下面表 3.4 所示。

表 3.4 二进制、BCD 和十进制数字表示的比较

DEC（十进制）	BIN（二进制）	BCD（二-十进制码）
0	0	I 0
1	1	I 1
2	10	I 10
3	11	I 11
4	100	I 100
5	101	I 101
6	110	I 110
7	111	I 111
8	1000	I1000
9	1001	I1001
10	1010	1I0000
11	1011	1I0001
12	1100	1I0010

(2) QnACPU 的寄存器（数据寄存器、连接寄存器等）是由 16 个比特位组成的。因此，用 BCD 码表示时，可以保存的数字范围是 0 到 9999。

3.4.4 实数（浮点小数数据）

(1) 实数

实数就是单精度的十进制浮点数据。

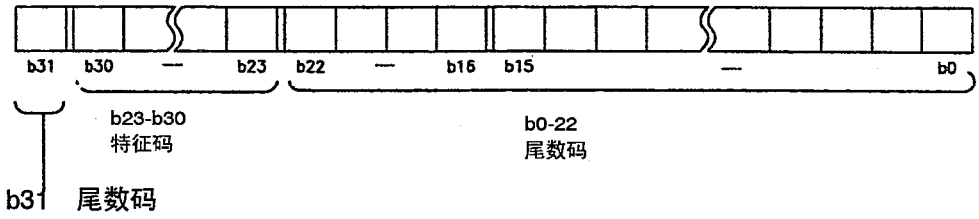
(2) 十进制浮点数据的内部表示

以下解释说明 QnACPU 中的十进制浮点实数数据的内部表示。

使用两个字软元件的十进制浮点数据表示如下所示。

1. [尾数]×2 (特征值)

以下所示以及解释说明的是用于十进制浮点数据内部表示的比特位配置。



. 尾数码：在 b31 表示的尾数码如下。

0：正

1：负

. 特征值：在 b23-b30 的“2^n”中的“n”有几种不同的表示，这取决于 b23-b30 的 BIN 值。

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

. 尾数：对于一个二进制值 1.XXXXXX...，该值的“XXXXXX”部分在 b0-b22 表示（23 位）。

要点

. 外围设备的监视功能允许监视 QnACPU 的十进制浮点数据。

. 对于一个“0”值，在所有的 b0-b31 位都指定为“0”。

. 下面所示是计算例子（nnnnn “X” 代表一个 X 系统数据表示）。

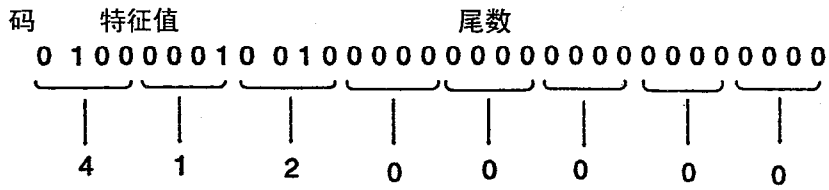
(1) 存储 “10”

尾数码：正→0

特征值：3→82_H→$(10000010)_2$

尾数 ：$(010\ 00000\ 00000\ 00000\ 00000)_2$

因此数据表示将是 41200000_H，如下所示。



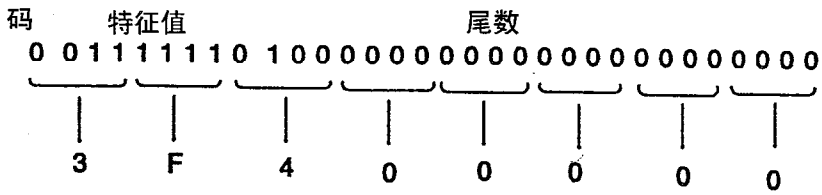
(2) 存储 “0.75”

尾数码：正→0 $(0.75)_{10}$→$(0.11)_2$→(1.100------×2-1)

特征值：

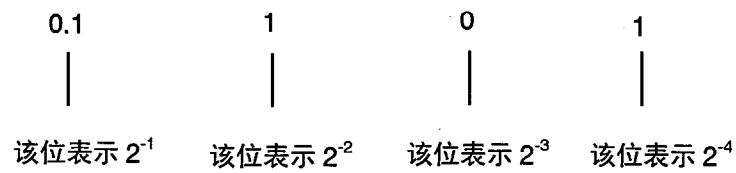
尾数：

因此数据表示将是 $3F400000_H$，如下所示。



注释

在采用二进制系统时，在小数点后面部分的值如下计算：



3.5 字符串数据

(1) 字符串数据

QnACPU 使用 ASCII 码数据。

(2) ASCII 码字符串

ASCII 码字符串如下表所示。“00_H”（空码）用在字符串的结尾。

b8	b7	b6	b5	b4	b3	b2	b1	Column / Row	0	0	0	0	0	0	0	0
									0	0	0	0	1	1	1	1
									0	0	1	1	0	0	1	1
									0	1	0	1	0	1	0	1
									0	1	2	3	4	5	6	7
				0	0	0	0	0	NUL		(SP)	0	@	P	`	p
				0	0	0	1	1			!	1	A	Q	a	q
				0	0	1	0	2			"	2	B	R	b	r
				0	0	1	1	3			#	3	C	S	c	s
				0	1	0	0	4			$	4	D	T	d	t
				0	1	0	1	5			%	5	E	U	e	u
				0	1	1	0	6			&	6	F	V	f	v
				0	1	1	1	7			'	7	G	W	g	w
				1	0	0	0	8			(	8	H	X	h	x
				1	0	0	1	9			)	9	I	Y	i	y
				1	0	1	0	A			*	:	J	Z	j	z
				1	0	1	1	B			+	;	K	[	k	{
				1	1	0	0	C			,	<	L	\	l	|
				1	1	0	1	D			-	=	M	]	m	}
				1	1	1	0	E			.	>	N	^	n	~
				1	1	1	1	F			/	?	O	_	o	

4. 软元件

4.1 软元件列表

4.1.1 软元件列表

QnACPU 中可以使用的软元件的名称和数据范围如下面表 4.1 所示。

表 4.1 软元件列表

类别	类型	软元件名称	缺省值		指定的设置范围参数	参考章节
			点数	使用范围		
内部用户软元件	位软元件	输入	8192 点	X0 到 X1FFFF	固定不变	4.2.1
		输出	8192 点	Y0 到 X1FFFF		4.2.2
		内部继电器	8192 点	M0 到 M8291	可以改变为 28.8K 字或更少。*	4.2.3
		锁存继电器	8192 点	L0 到 L8191		4.2.4
		报警器	2048 点	F0 到 F2047		4.2.5
		边缘继电器	2048 点	V0 到 V2047		4.2.6
		单步继电器*3	8192 点	S0 到 S511 per block		4.2.9
		连接特殊继电器*3	2048 点	SB0 到 SB7FF		4.2.8
		连接继电器	8192 点	B0 到 B1FFF		4.2.7
	字软元件	定时器*1	2048 点	T0 到 T2047		4.2.10
		保持定时器	0 点	(ST0 to ST2047)		
		计数器*1	1024 点	C0 to CC1023		4.2.11
		数据寄存器	12288 点	D0 to D12287		4.2.12
		连接寄存器	8192 点	W0 to W1FFF		4.3.13
		连接特殊寄存器	2048 点	SW0 to SW7FF		4.3.14
内部系统软元件	位软元件	功能输入	5 点	FX0 to FX4	不可能	4.3.1
		功能输出	5 点	FY0 to FY4		4.3.1
		特殊继电器	2048 点	SM0 to SM2047		4.3.2
	字软元件	功能寄存器	5 点	FD0 to FD4		4.3.1
		特殊寄存器	2048 点	SD0 to SD2047		4.3.3
连接直接软元件	位软元件	连接输入	8192 点	Jn\X0 to Jn\X1FFF	不可能	4.4
		连接输出	8192 点	Jn\Y0 to Jn\Y1FFF		
		连接继电器	8192 点	Jn\B0 to Jn\B1FFF		
		连接特殊继电器	512 点	Jn\SB0 to Jn\B1 FFF		
	字软元件	连接寄存器	8192 点	Jn\W0 to Jn\W1FFF		
		连接特殊寄存器	512 点	Jn\SB0 to Jn\B1 FFF		
特殊功能模块软元件	字软元件	缓冲寄存器	16384 点	Un\Go to Un\G16388	不可能	4.5

类别	类型	软元件名称	缺省值		指定的设置范围参数	参考章节
			点数	使用范围		
变址寄存器	字软元件	变址寄存器	16 点	Z0 to Z18	不可能	4.6

文件寄存器	字软元件	文件寄存器	0 点	——	0-1024 K 点（1K 单位）	4.7
嵌套	——	嵌套	15 点	No to N14	不可能	4.8
指针	——	指针	4096 点	P0 to P4095	不可能	4.9
		中断指针	48 点	I0 to 147		4.10
其他	位软元件	SFC 块	320 点	BL0 to BL319	不可能	4.11.1
		SFC 转换软元件	512 点	TR0 to TR511		4.11.2
	——	网络编号	256 点	J1 to J255		4.11.3
		I/O 编号		U0 to UFF		4.11.4
常数	——	十进制常数	K-2147483648-K2147483647			4.12.1
		十六进制常数	H0 to HFFFFFFFF			4.12.2
		实数常数	E±1.17549-38 to E±3.40282+38			4.12.3
		字符串常数	“ABC”, “123”			4.12.4

注释

1）*1：定时器、保持定时器和计数器的“触点”和“线圈”是位软元件，“当前值”则为字软元件。

2）*2：实际的可用点数根据特殊模块而不同。关于缓冲存储器的点数的详细资料，请参阅特殊功能模块手册。

3）*3：输入、输出、单步继电器、连接特殊继电器、连接特殊寄存器保持为各自的缺省值，这些值不能更改。

4.1.2 内部用户软元件内的设置单元

对于除了输入（X）、输出（Y）、单步继电器（S）软元件连接特殊继电器（SB）和连接特殊寄存器 CSN 以外的所有的 QnACPU 内部用户软元件，使用的点数可以通过“软元件设置”参数在 28.8K 字的范围内进行修改。下面讨论进行这些修改时所应该考虑的项目。

(1) 设置范围

（a）软元件点数以 16 点为单位指定。

（b）对于一种类型的软元件最多可以指定 32K 点。内部继电器、锁存继电器、报警器、边缘继电器、连接继电器、定时器、保持定时器和计数器的总的最大点数是 64K 点。对于定时器、保持定时器和计数器，一点当作两点来计算（一点用于线圈，一点用于触点）。

(2) 存储器大小（位软元件大小）

（a）对于位软元件：

对于位软元件，16 点当作一个字来计算。

（位软元件容量）=（M+L+F+B+SB 总点数）/16 （字）

（b）对于定时器（T）、保持定时器（ST）和计数器（C）：

对于定时器、保持定时器和计数器，16 点当作 18 个字来计算。

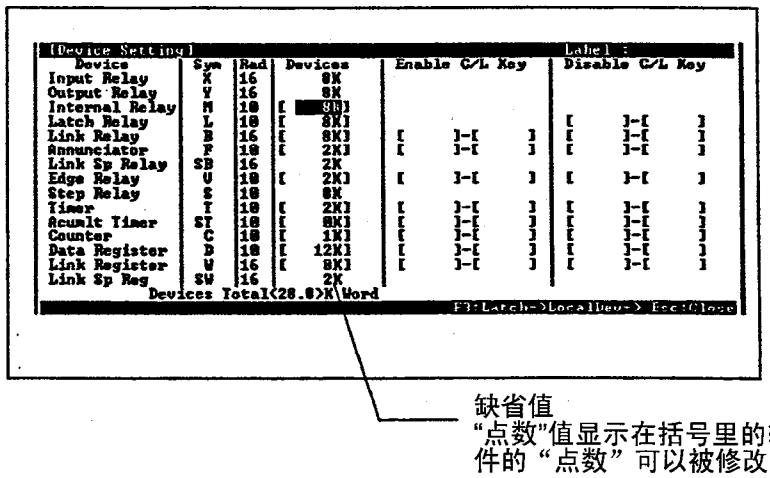
（定时器、保持定时器、计数器容量）= （T，ST，C 总点数）/16 ×18（字）

（c）对于字软元件：

对于数据寄存器（D）、连接寄存器（W）和特殊连接寄存器（SW），16 点当作 16 个字来计算。

（字软元件大小）= （D，W，SW 总点数）/16 ×16（字）

[软元件设置窗口]



缺省值
“点数”值显示在括号里的软元件的“点数”可以被修改。

4.2 内部用户软元件

内部用户软元件是可以由用户自由运用的软元件。

内部用户软元件的可用点数设置（缺省值）事先被指定好。然而，这个设定可以通过一个外围设备的参数设置在 28.8K 的范围内被修改。

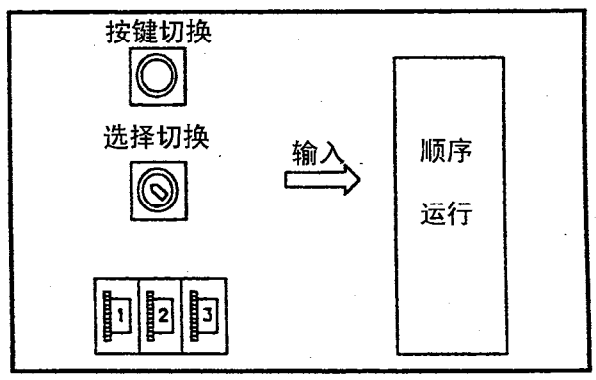
关于内部用户软元件缺省值以及参数指定的设置范围，请参见 4.1 节。

> 要点
>
> （1）当一个内部用户软元件的“可用点数”设置被修改时，根据原先设置参数下创建顺控程序和 SFC 程序不能再象原来那样使用了。因此，在修改了“可用点数”设置后，需从 QnACPU 中读出以上程序并略作修改后再写回到 QnACPU 中。

4.2.1 输入（X）

（1） 定义

（a）输入是从一个外围设备通过按键切换、选择切换、限制切换、数字切换等传送给 QnACPU 的命令或数据。



（b）每个输入可被看作有一点 QnACPU 内的虚拟继电器 Xn 与之对应，而顺控程序正是使用 Xn 的 a/b 触点。

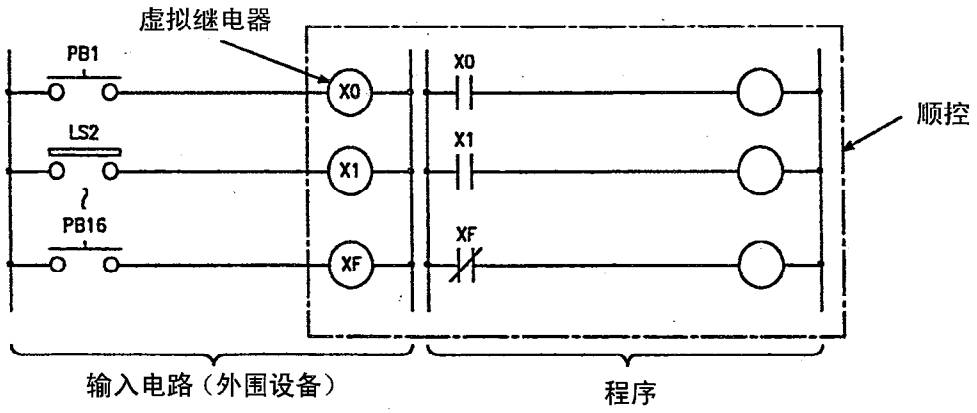


图 4.1 输入（X）的示意图

（c） 对于用在程序中的 Xn 的 a 触点和 b 触点的数量没有限制。

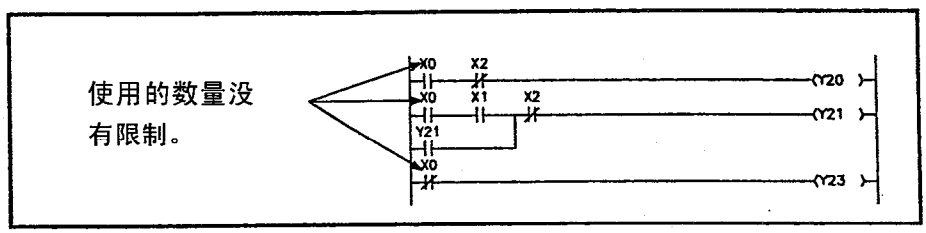


图 4.2 用于程序中的输入（x）

（2） 读取输入

（a）有两种输入类型：“刷新输入”和“直接存取输入”。

1）刷新输入是使用刷新模式从输入模块中读入的 ON/OFF 数据。*1

这些输入在顺控程序中指定为“X[]”。

例如，一个“100”的输入变成“X100”。

2）直接存取输入是使用直接模式，从输入模块中读入的 ON/OFF 数据。*2

这些输入在顺控程序中指定为“DX[]”。

例如，一个“100”的输入变成“DX100”。

（b）可以将同一个输入号指定为一个刷新输入和一个直接存取入。

如果在用作直接存取输入之后用作刷新输入，那么操作将基于在直接存取输入读入的 ON/OFF

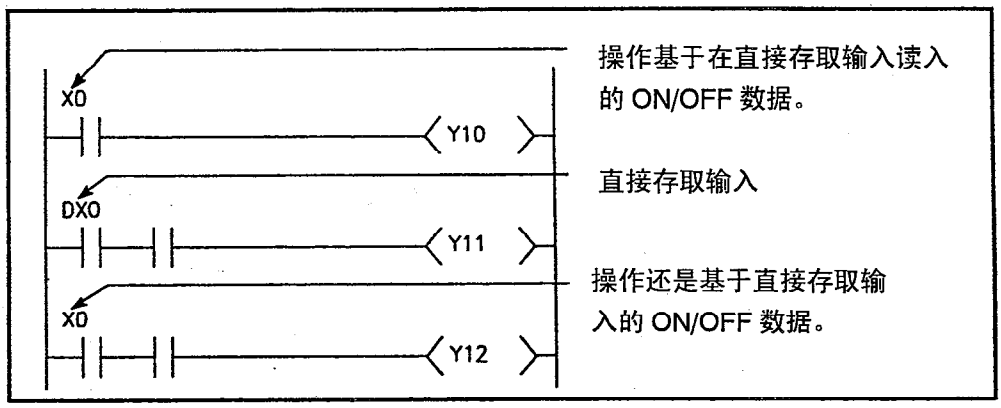


图 4.3 刷新输入和直接存取输入

注释

1）*1：关于刷新模式请参见 3.3.1 节。

2）*2：关于直接模式请参见 3.3.2 节。

要点

（1）直接存取输入只能用于一点单元中。它们不能用点数指定来说明。

LD DX0 ——————可以用

MOV K4DX0 D0 ———不能用

不能用点数指定直接存取输入。

（c）刷新输入和直接存取输入之间的差异

对于直接存取输入，输入模块被执行的指令直接存取，因此，处理速度比刷新输入的处理速度慢。

不仅如此，直接存取输入只能用于（安装在基板和扩展槽中）输入模块和特殊功能模块的输入。刷新输入和直接输

入之间的差别如下面表 4.2 所示。

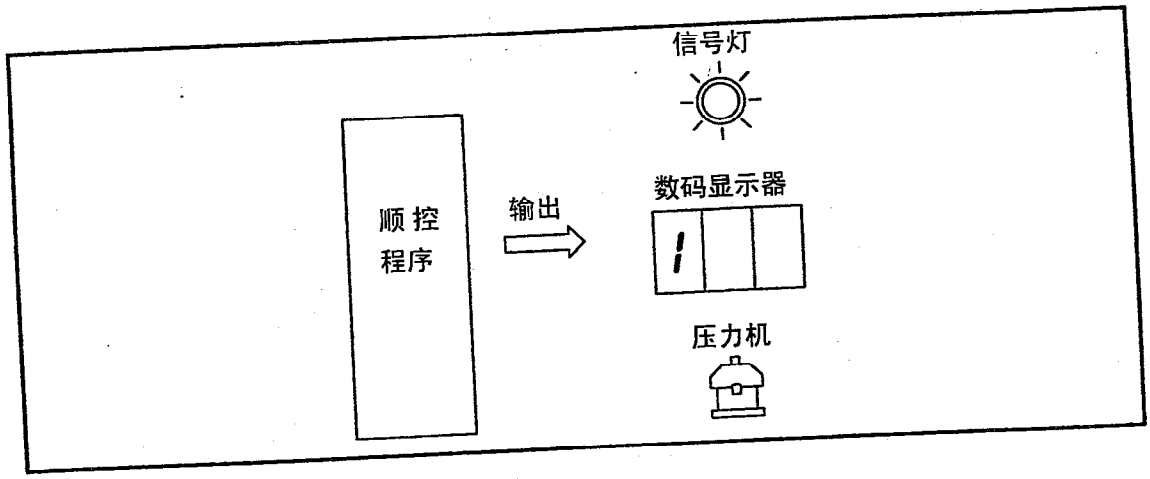
表 4.2 刷新输入和直接存取输入之间的差别

项目	刷新输入	直接存取输入
处理速度	0.075 到 0.2us	大约 10us
安装在基板/扩展槽单元的输入模块	可用	可用
安装在基板/扩展槽单元的特殊功能模块的输入		
安装在基板/扩展槽单元的 I/O 连接模块的输入		
在 MELSECNET/10 网络系统中使用的输入	可用	不可用
在 MELSECNET（II/B）数据连接系统中使用的输入		
在 MELSECNET/MINI-S3 连接中使用的输入		

4.2.2 输出（Y）

（1） 定义

（a）输出是程序输出到外部设备（螺线管、电磁开关器、信号灯、数码显示器等）的控制结果。



（b）输出可作为相当于向外部输出的一个 a 触点来处理。

（c） 用在程序中的 Yn 的 N/O 触点和 N/C 触点的数量没有限制。

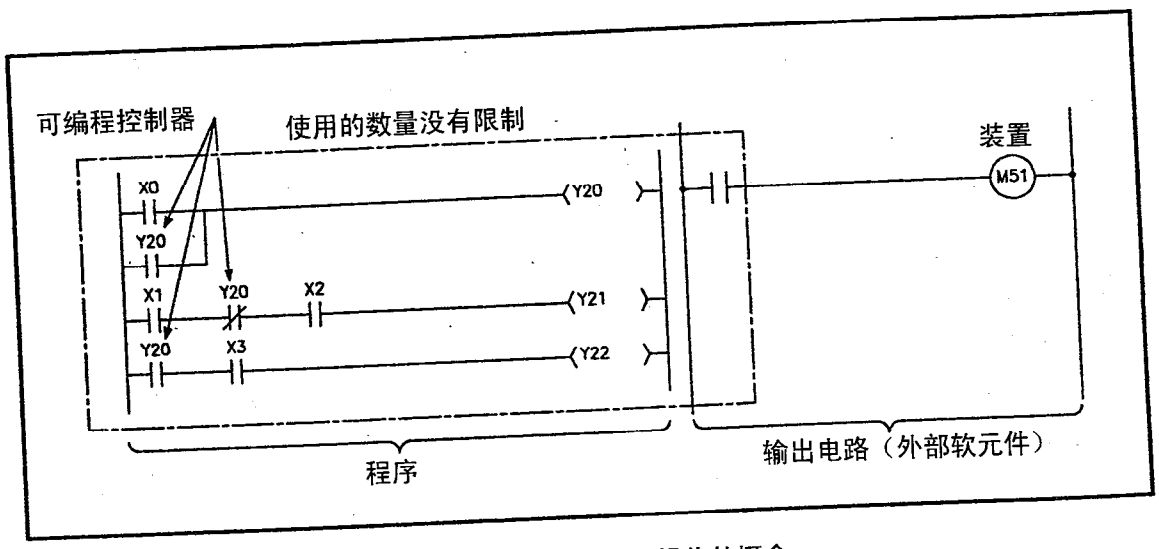
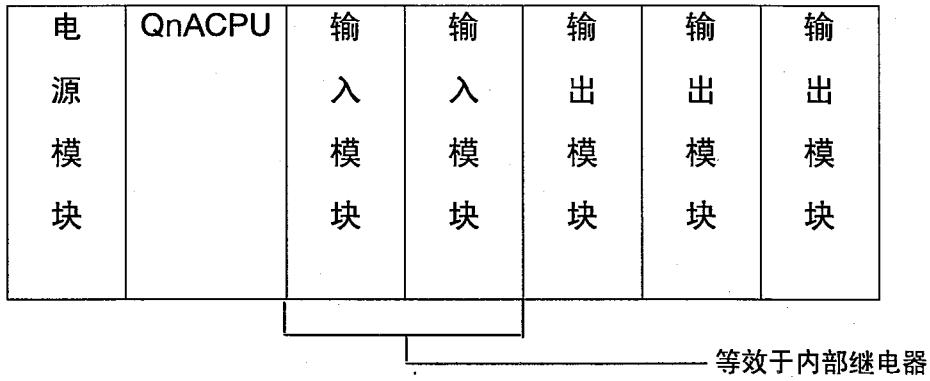


图 4.4 输出（Y）操作的概念

（2）输出作为内部继电器（M）来使用

空闲插槽上和安装有输入模块插槽上的“Y”输入可以用作内部继电器（M）。



（3） 输出方式

（a）有两种输出方式：“刷新输出”和“直接存取输出”。

1）刷新输出是通过使用刷新模式输出到输出模块的 ON/OFF 数据。*1

这些输出在顺控程序中指定为“Y[]”。

例如，一个“100”的输入变成“Y100”。

2）直接存取输出是通过使用直接模式，输出到输出模块中的 ON/OFF 数据。*2

这些输出在顺控程序中指定为“DY[]”。

例如，一个“100”的输入变成“DY100

（b）刷新输出和直接存取输出之间的差别

对于直接存取输出，通过执行指令直接访问输出模块，因此，处理速度比刷新输出的处理速度慢。

另外，直接存取输出只能用于（安装在基板和扩展槽中）输出模块和特殊功能模块使用的输出。刷新输出和直接输出之间的差别如下面表 4.3 所示。

表 4.3 刷新输出和直接存取输出之间的差别

项目	刷新输出	直接存取输出
处理速度	0.075 到 0.2us	大约 10us
安装在基板/扩展槽单元的输出模块	可用	可用
安装在基板/扩展槽单元的特殊功能模块的输出		
安装在基板/扩展槽单元的 I/O 连接模块的输出		
在 MELSECNET/10 网络系统中使用的输出	可用	不可用
在 MELSECNET（II/B）数据连接系统中的输出		
在 MELSECNET/MINI-S3 连接中使用的输出		

注释

1）*1：关于刷新模式的详细资料请参见 3.3.1 节。

2）*2：关于直接模式的详细资料请参见 3.3.2 节。

要点

（1）直接存取输出只能用于一点单元中。

它们不能用点数指定来规定。

OUT DY10 ————————可以用

MOV D0 K4Y10 ————不能用

不能用点数指定直接存取输出。

4.2.3 内部继电器（M）

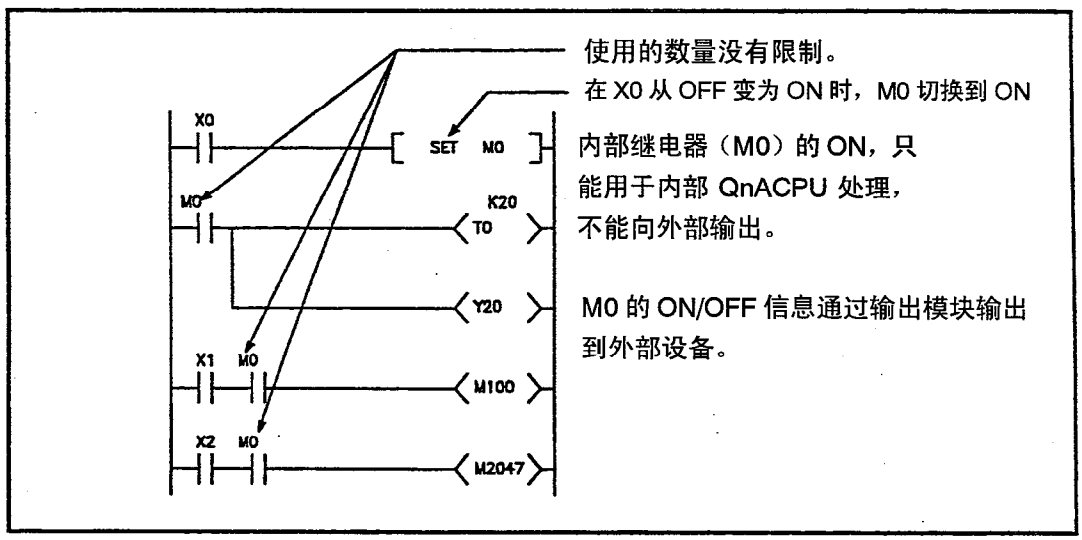
（1） 定义

（a）内部继电器是辅助继电器，它们不能在 PLC 内部被锁存（停电保持）。

下列操作会使所有的内部继电器都被关闭。

- 电源从 OFF 切换到 ON。
- QnACPU 发生复位。
- 执行了 QnACPU 锁存清除操作。

（b）用在程序中的触点（N/O 触点和 N/C 触点）的数量没有限制。



4.5 内部继电器

（2） 向外部输出的过程

（a）输出（Y）用于向一个外部设备输出顺控程序的运行结果。

（b）连接继电器（B）用于通过 MELSECNET/10 向其他的站输出 ON/OFF 信息。

注释

1） 当需要一个锁存（停电保持）时，应该使用锁存继电器（L）。

关于锁存继电器的详细资料请参见 4.2.4 节。

4.2.4 锁存继电器（L）

（1） 定义

（a）锁存继电器是辅助继电器，它们可以在可编程控制器的内部被锁存（停电保持）。

即使是在下列情况下，锁存继电器运行结果（ON/OFF 信息）也会被保存。

-当电源从 OFF 切换到 ON 时。

-当发生了 QnACPU 复位时。

锁存是由 QnACPU 的电池来备份的。

（b） 锁存继电器可以用 QnACPU 的运行/停止键切换来复位。然而，如果在参数中的软元件设置已经将该锁存继电器的锁存清除键为无效时，那么锁存继电器就不能用运行/停止键的操作来复位。

关于设置锁存清除键无效的详细资料，请参见使用的 CPU 模块的用户手册。

（c） 用在程序中的触点（N/O 触点和 N/C 触点）的数量没有限制。

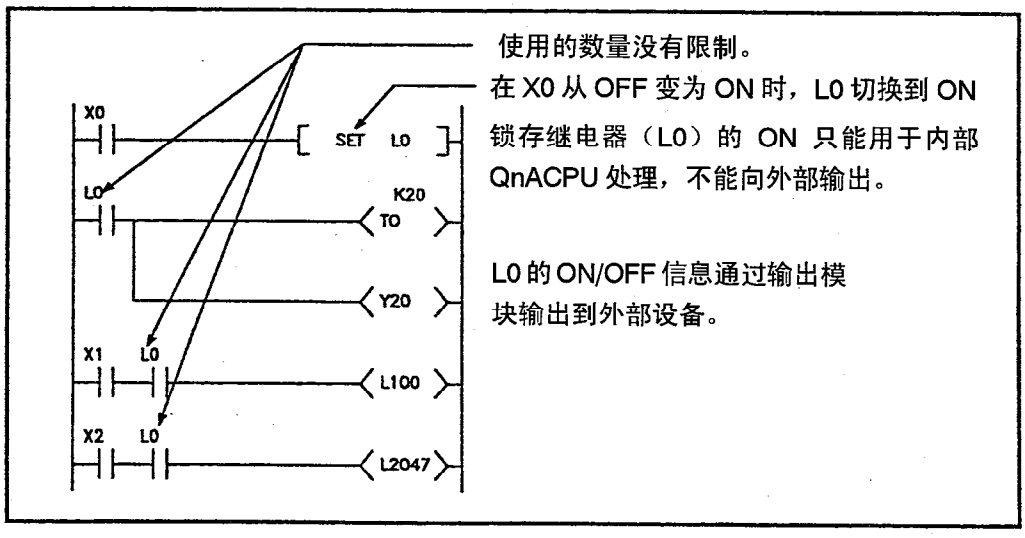


图 4.6 锁存继电器

（2） 外部输出过程

（a）输出（Y）用于向外部设备输出顺控程序的运行结果。

（b）连接继电器（B）用于通过 MELSECNET/10 向其他站输出 ON/OFF 信息。

注释

1）当不需要锁存功能时，应该使用内部继电器（M）。关于内部继电器的详细资料请参见 4.2.3 节。

4.2.5 报警器（F）

(1) 定义

(a) 报警器是用户用于故障检测程序中的软元件。

(b) 当报警器切换成 ON 时，特殊继电器（SM62）会被切换成 ON，并且切换成 ON 的报警器的编号和数量被保存到特殊寄存器中。

- 特殊继电器 ：SM62.....即使只有一个报警器被切换成 ON，也会切换成 ON。
- 特殊寄存器 ：SD62...保存切换成 ON 的第一个报警器的编号。
  ：SD63...保存处于 ON 状态报警器的数量。
  ：SD64 到 SD79...保存按照它们被切换成 ON 的顺序的报警器的编号。SP62 和 SD64 中的报警器号码是一样的。

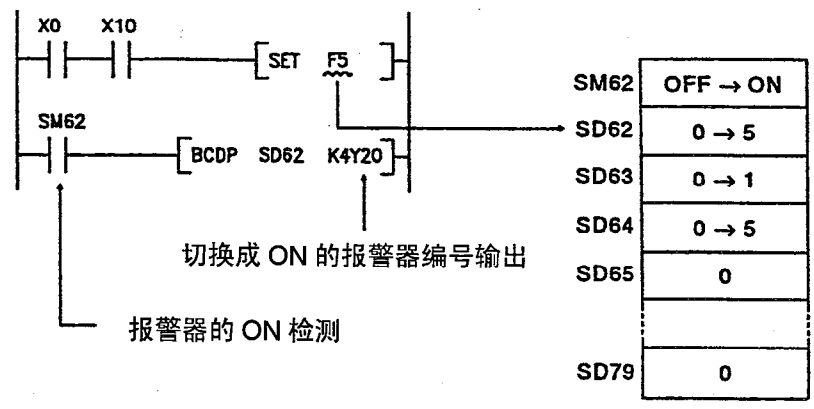
保存在 SD62 中的报警器编号也寄存在“故障履历区域”。

(c) 在故障检测程序中使用报警器可以使用户通过监视特殊继电器和特殊寄存器，检查故障是否存在，以及故障内容（报警器编号）。

举例

输出切换成 ON 报警器（F5）的编号的程序如下所示。

（故障检测程序）



(2) 使报警器 ON 的方法

(a) 可以通过 SET F[]和 OUT F[]指令来控制报警器 ON/OFF。

1) SET F[]指令在输入条件的上升沿(OFF→ON)将报警器切换成 ON，并且即使输入条件被切换成 OFF 时，报警器仍然保持 ON。在使用多个报警器的情况下，使用 OUT F[]指令可以减少扫描时间。

2) 尽管 OUT F[]指令可以根据输入条件的 ON/OFF 操作将报警器切换成 ON 或 OFF，但是它是在每一次扫描时执行的。

要点

(1) 如果报警器通过其他方法而不是通过 SET F[]和 OUT F[]指令被切换成 ON，那么报警器的运作方法与内部继电器相同。
（SM62 不会被切换成 ON，并且报警器的编号不会被保存在 SD62、SD64 到 SD79 中。）

另外，即使一个报警器被 OUT F[]指令切换成 OFF，特殊继电器和特殊寄存器中的内容也不会改变。因此，此时需用 RST F[]指令或 LEDR 指令。（参见下面项目（3）的“使报警器 OFF 的方法和处理内容”）

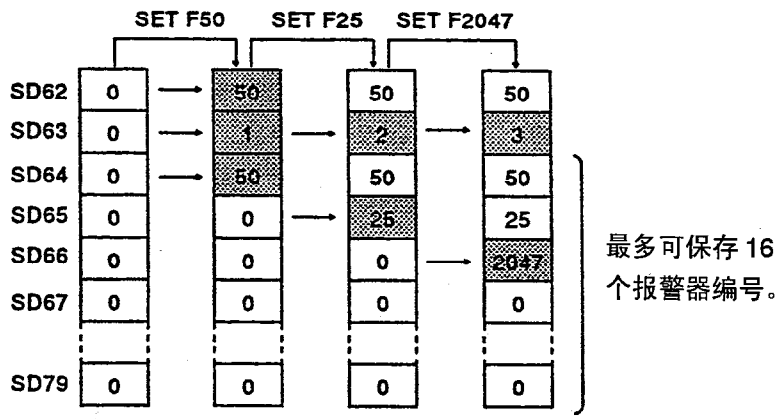
(a) 报警器 ON 时的处理

1) 保存在特殊寄存器中的数据（SD62 到 SD79）

a) 被切换成 ON 的报警器的编号按顺序保存在 SD64 到 SD79 中。

b) 保存在 SD64 中的报警器编号同样被保存在 SD62 中。

c) SD63 的值被加“1”。



2) QnACPU 的处理

a) Q2ACPU（-S1），Q2AS（H）CPU（-S1）在 CPU 前面板上的“用户灯”亮。

b) Q3ACPU，Q4ACPU，Q4ARCPU 保存在 SD62 中的信号器编号显示在 LED 显示器上（CPU 前面板）。

(3) 报警器 OFF 过程和处理内容

(a) 使报警器 OFF 的方法

一个报警器可以通过 RST F□和 LEDR 指令切换成 OFF。

1) 被 SET F□指令切换成 ON 的报警器可以被 RST F□指令切换成 OFF。

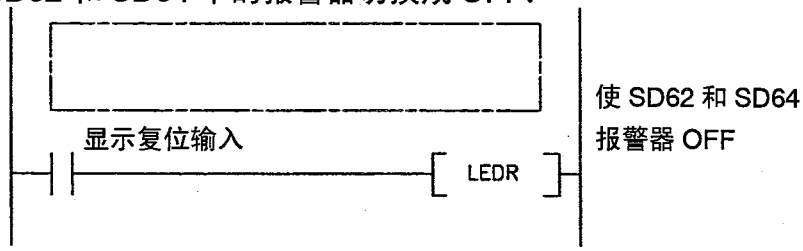
2) LEDR 指令用来将保存在 SD62 和 SD64 中的报警器切换成 OFF。

3) 一个已经被 OUT F□指令切换成 ON 的报警器在 OUT F□指令被切换成 OFF 时，被切换成 OFF。

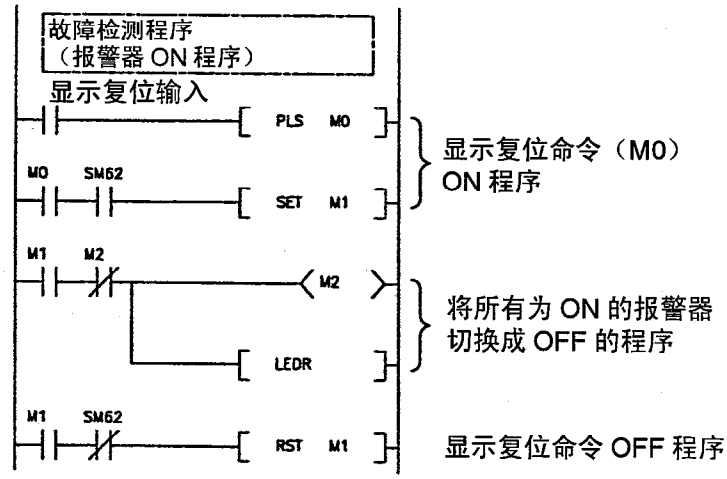
但此时“报警器 OFF 时的处理”（下面的（b）项目）不会发生。

在报警器已经被 OUT F□指令切换成 OFF 后执行 RST F□和 LEDR 指令。

1) 存在 SD62 和 SD64 中的报警器切换成 OFF：



2) 将所有为 ON 的报警器切换成 OFF：



注释

1) 也可以用 BKRST 指令将指定编号范围的报警器切换成 OFF。关于 BKRST 指令的详细资料，请参考 QnACPU 编程手册（通用指令）。

(b) 报警器 OFF 的处理

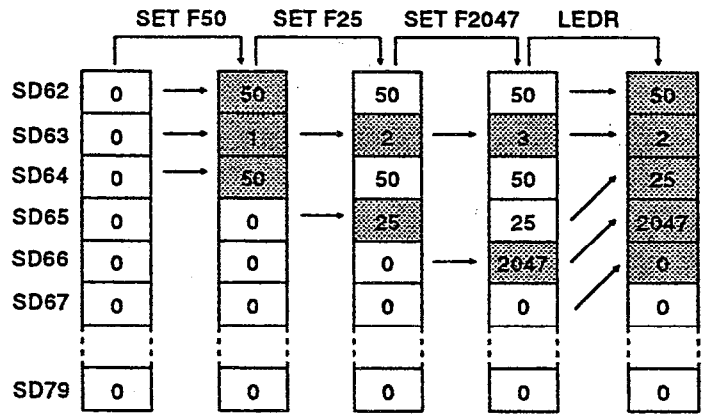
1) LEDR 指令的特殊寄存器（SD62 到 SD79）数据操作

a) 保存在 SD64 中的报警器编号被删除，保存在其后寄存器（SD65 到 SD79）中的报警器编号被向前移，以填充空余空间。

b) 保存在 SD64 中的报警器编号保存到 SD62 中。

c) 从 SD63 值中减“1”。

d) 如果 SD63 的值是“0”，那么 SM62 就被切换成 OFF。



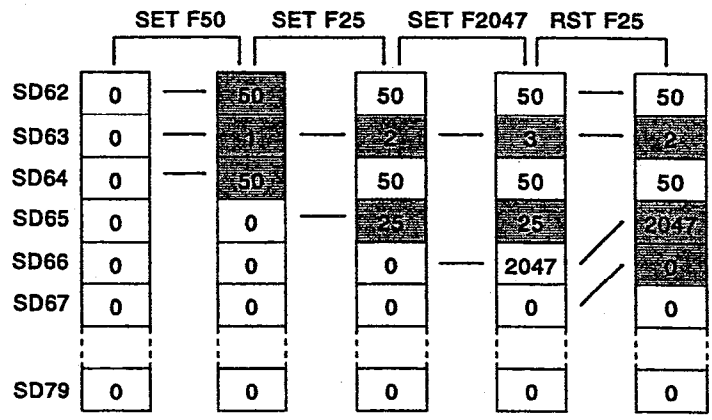
2) 执行 RST F[]指令或使用 OUT F[]指令将警报器关闭时特殊寄存器（SD62~SD79）中的数据：

a) 被切换成 OFF 的报警器的编号被删除，并且随后的所有的报警器编号被向前移。

b) 如果原先保存在 SD64 中的报警器被切换到 OFF，那么保存在 SD64 中的新的报警器编号保存到 SD62 中。

c) 从 SD63 的值中减“1”。

d) 如果 SD63 的值是“0”，那么 SM62 就被切换成 OFF。



3) QnACPU 的处理

a) Q2ACPU（-S1），
Q2AS（H）CPU（-S1）

- 如果从 SD64 到 SD79 的所有报警器被切换成 OFF，那么“用户灯”（在 CPU 的前面板上）切换成 OFF。

b) Q3ACPU，Q4ACPU
Q4ARCPU

- 如果显示报警器的编号（在 CPU 前面板的 LED 显示）切换成 OFF，那保存在 SD62 中的新的报警器编号被显示。
- 如果被切换成 OFF 的报警器编号并不是当前显示器上表示的号码，那么正在显示的编号将不会改变。
- 如果从 SD64 到 SD79 的所有报警器编号被切换成 OFF，那么 LED 显示器将切换 OFF。

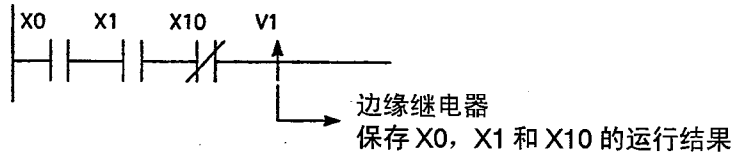
要点

(1) 如果在通过 OUT F[]指令将一个报警器切换成 OFF 之后没有执行 RST F[]指令或 LEDR 指令，那么“报警器 OFF 时的处理”（参见上面的（b））将不会发生。

4.2.6 边缘继电器（V）

(1) 定义

(a) 一个边缘继电器就是从梯形图块的开始保存运行结果（ON/OFF 信息）的软元件。边缘继电器只能用于触点，而不能用于线圈。



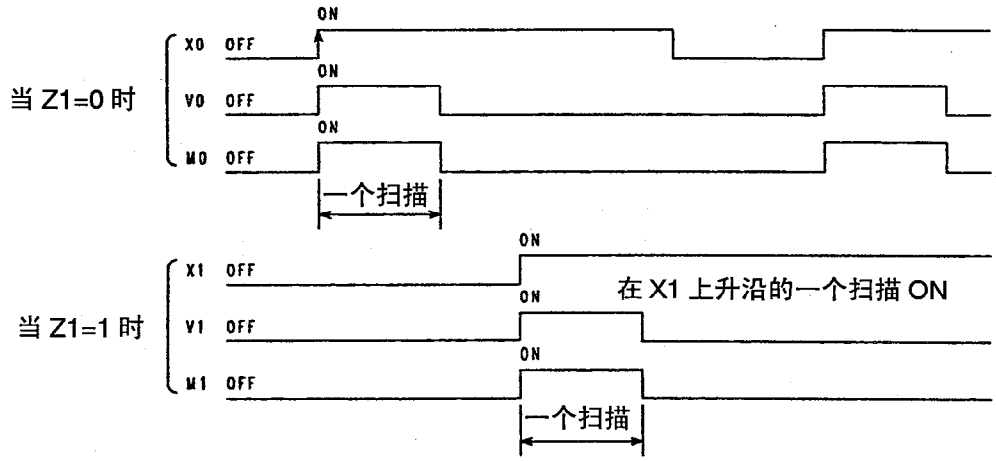
(b) 同一个边缘继电器的编号不能在 QnACPU 执行的程序中使用两次。

(2) 边缘继电器的应用

边缘继电器用于检测使用变址资格配置的程序中的上升沿（OFF→ON）。

[梯形图的例子]

[时序图]



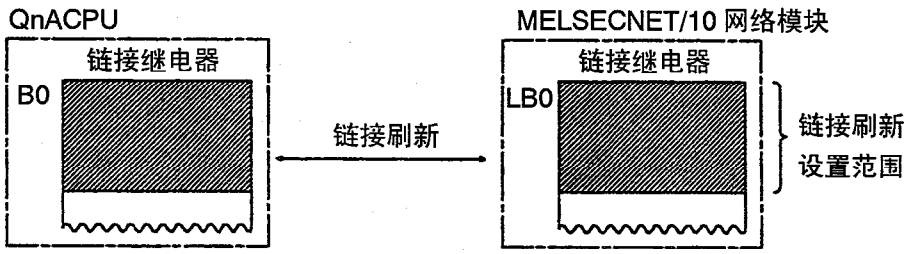
注释

1）*1：X0Z1 的 ON/OFF 信息保存在 V0Z1 的边缘继电器中。
例如，X0 的 ON/OFF 信息保存在 V0 中，而 X1 的 ON/OFF 信息保存在 V1 中。

4.2.7 连接继电器（B）

(1) 定义

(a) 链接继电器是对应于 MELSECNET 数据链接模块和 MELSECNET/10 网络模块的链接继电器（LB）和 QnACPU 之间数据刷新时的 QnACPU 继电器。



在未被 MELSECNET 数据链接系统和 MELSECNET/10 网络系统使用的数据范围的 B 可用作内部继电器或者锁存继电器。

. 锁存范围外... 相当于内部继电器
. 锁存范围内.....相当于锁存继电器

(b) 用于程序中的触点（N/O 触点和 N/C 触点）的数量没有限制。

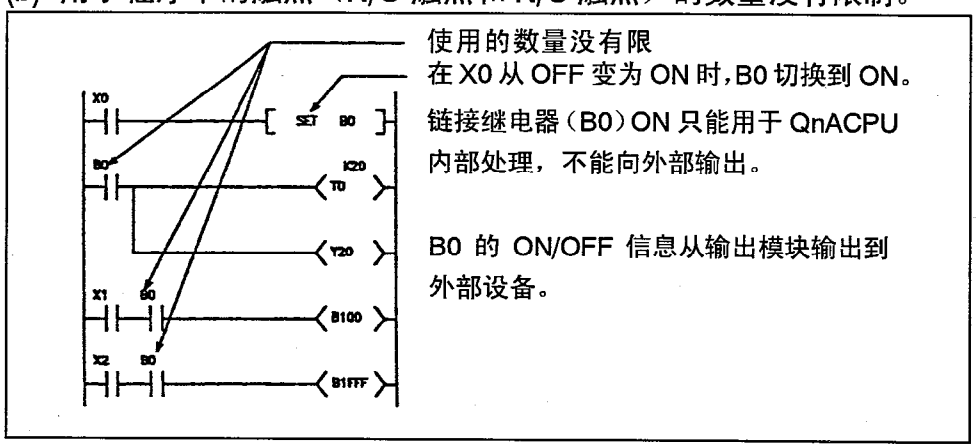


图 4.5 连接继电器

(2) 在 MELSECNET/10 网络系统中使用连接继电器

(a) 如果连接继电器用在 MELSECNET/10 网络系统中，那么主站的 ON/OFF 信息可以被读到其他的站使用。在 MELSECNET/10 网络系统中使用连接继电器允许在主控站和一个寻常站，以及在寻常站之间进行 ON/OFF 信息的传递。

(b) 为了在 MELSECNET/10 网络系统中使用连接继电器，就需要在主控站进行网络参数设置。没有被网络参数设置指定的连接继电器可以用作内部继电器或锁存继电器。

(3) 在 MELSECNET 数据连接系统中使用。

(a) 当连接继电器与一个 MELSECNET 数据连接系统一起使用时，其他站可以读取并使用主站的 ON/OFF 状态。
连接继电器使得能够在一个MELSECNET数据连接系统的一个主站和从站之间，或者在从站之间进行 ON/OFF 信息的交换。

(b) 为了在一个 MELSECNET 数据连接系统中使用，就必须在主站设置连接参数。
没有被连接参数设置指定的连接继电器可以用来替代内部继电器。

注释

1) 关于网络参数的详细资料，请参考 QnA/Q4AR MELSECNET/10 网络系统参考手册。
2) 关于连接参数的详细资料，请参考 MELSECNET 和 MELSECNET/B 数据连接系统参考手册。

4.2.8 特殊连接继电器（SB）

(1) 定义

(a) 特殊连接继电器（SB）用于在 MELSECNET/10 网络模块和用户程序之间进行 ON/OFF 数据交换。

(b) 因为特殊连接继电器是在数据连接中不同状态时被切换成 ON 和 OFF 的，所以它们可作为确定数据连接状态的工具。

(2) 特殊连接继电器点数

每一个 MELSECNET/10 网络模块有 2048 个特殊连接继电器点（SB0 到 SB7FF）。QnACPU 特殊连接继电器的缺省“点数”设置是每个模块 512 点。如下所示：

S80 to SB1FF	第 1 个网络模块
S200 to SB3FF	第 2 个网络模块
S400 to SB5FF	第 3 个网络模块
S600 to SB7FF	第 4 个网络模块

注释

1) 关于用于 QnACPU 的特殊连接继电器的详细资料，请参考 QnACPU 编程手册（通用指令）。

4.2.9 单步继电器（S）

一个单步继电器就是一个 SFC 程序软元件。关于使用单步继电器的步骤的详细资料，请参考 QnACPU 编程手册（SFC）。

要点

因为单步继电器是一个 SFC 程序专用软元件，所以它不能在顺控程序中被用作一个内部继电器。否则，会出现 SFC 错误，并且系统运行将会停止（系统停机）。

4.2.10 定时器（T）

QnACPU 定时器是“加法定时器”类型，当线圈切换成 ON 时开始计时，在当前值等于设置值时结束（时间到）。当“时间到”发生时，触点被切换成 ON。有三种定时器类型：低速定时器、高速定时器以及保持定时器。

要点
在执行 OUT T[]指令时，定时器的线圈会 ON/OFF，当前值会更新，然后其触点回 ON/OFF。 在 END 处理时，定时器的当前值不会更新，其触点不会 ON/OFF。

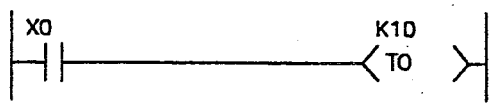
低速定时器

(1) 定义

(a) 低速定时器是只有当线圈 ON 时才运行的定时器。

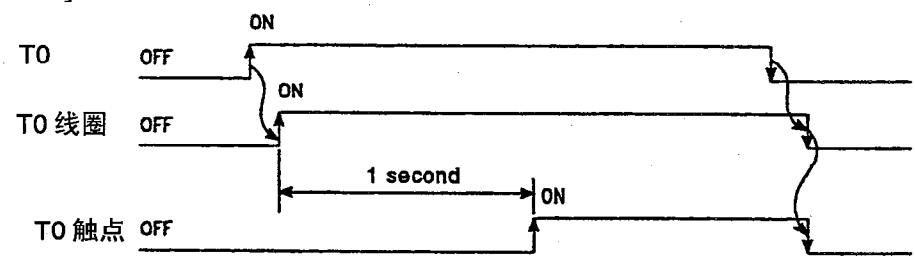
(b) 当定时器的线圈切换成 ON 时，开始计时，并且当出现“时间到”时，触点切换成 ON。当定时器的线圈切换成 OFF 时，当前值变为“0”，并且触点切换成 OFF。

[时序图的例子]



当 X0 切换成 ON 时，T0 线圈切换成 ON，并且在一秒钟之后触点切换成 ON。（在低速定时器以 100ms 为单位计时时）

[时序图]



(2) 测量单位

(a) 低速定时器的缺省的计时单位是 100ms。

(b) 可以以 10ms 为单位在 10ms 到 1000ms 范围内指定计时单位。这一设置可以在“PC 系统设置”参数中指定。

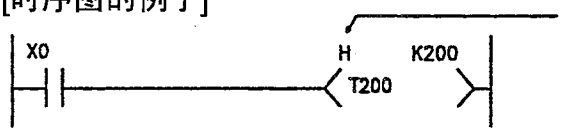
高速定时器

(1) 定义

(a) 高速定时器是只在线圈 ON 时才可运行的定时器。

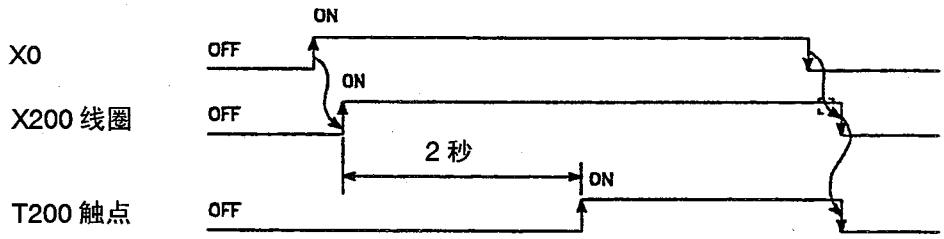
(b) 当定时器的线圈切换成 ON 时，开始计时，并且当出现“时间到”时，触点切换成 ON。当定时器的线圈切换成 OFF 时，当前值变为“0”，并且触点切换成 OFF。

[时序图的例子]



高速定时器显示当 X0 切换成 ON 时，T200 线圈切换成 ON，并且在 2 秒钟之后触点切换成 ON。（高速定时器以 10ms 为单位计时）

[时序图]



(2) 测量单位

(a) 高速定时器的缺省的计时单位是 10ms。

(b) 计时单位可以以 1ms 为单位在 1ms 到 100ms 范围内指定。
这个设置是在 PC 系统的设置参数中指定的。*

保持定时器

(1) 定义

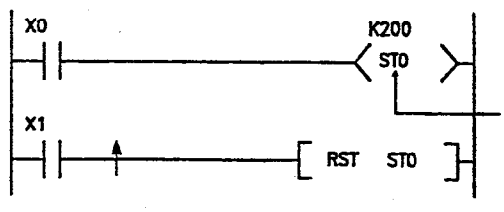
(a) 保持定时器测量“线圈 ON”的时间。

(b) 当定时器的线圈切换成 ON 时，开始计时，并且当发生时间到（线圈 OFF）时，触点切换成 ON。即使定时器的线圈为 OFF 时，(c) 当前值与触点的 ON/OFF 状态也会被保存。当定时器的线圈再次被切换成 ON 时，计时从保存的当前值重新开始。

(c) 有两种保持定时器类型：低速保持定时器和高速保持定时器。

(d) RST F[]指令用于清除（复位）当前值，并且将触点切换成 OFF。

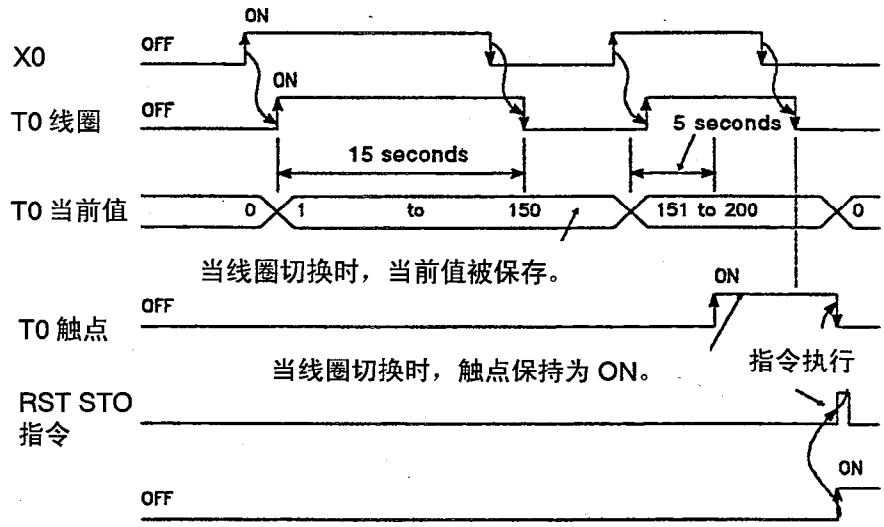
[时序图的例子]



在 20ms 内测量，X0 的 ON 时间。

保持定时器显示当 X1 切换成 ON 时，ST0 触点被复位，并且当前值被清除。

[时序图]



(2) 测量单位

(a) 保持定时器的测量单位设置与低速定时器和高速定时器的设置一样。

. 低速保持定时器：与低速定时器相同
. 高速保持定时器：与高速定时器相同

注释

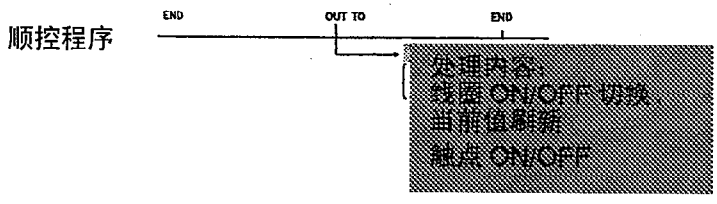
1）*：为了使用保持定时器，必须在 PC 软元件设置参数中指定保持定时器的“占用点数”。

定时器处理和精确度

(a) 当执行了一条 OUT T[]指令时，将会发生以下处理：定时器的线圈 ON/OFF、当前值刷新和触点 ON/OFF 处理。在 END 处理时，不会进行以上处理。

[梯形图的例子]

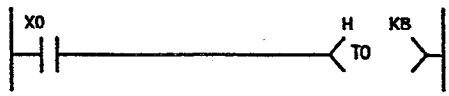
[OUT T0 指令的处理]



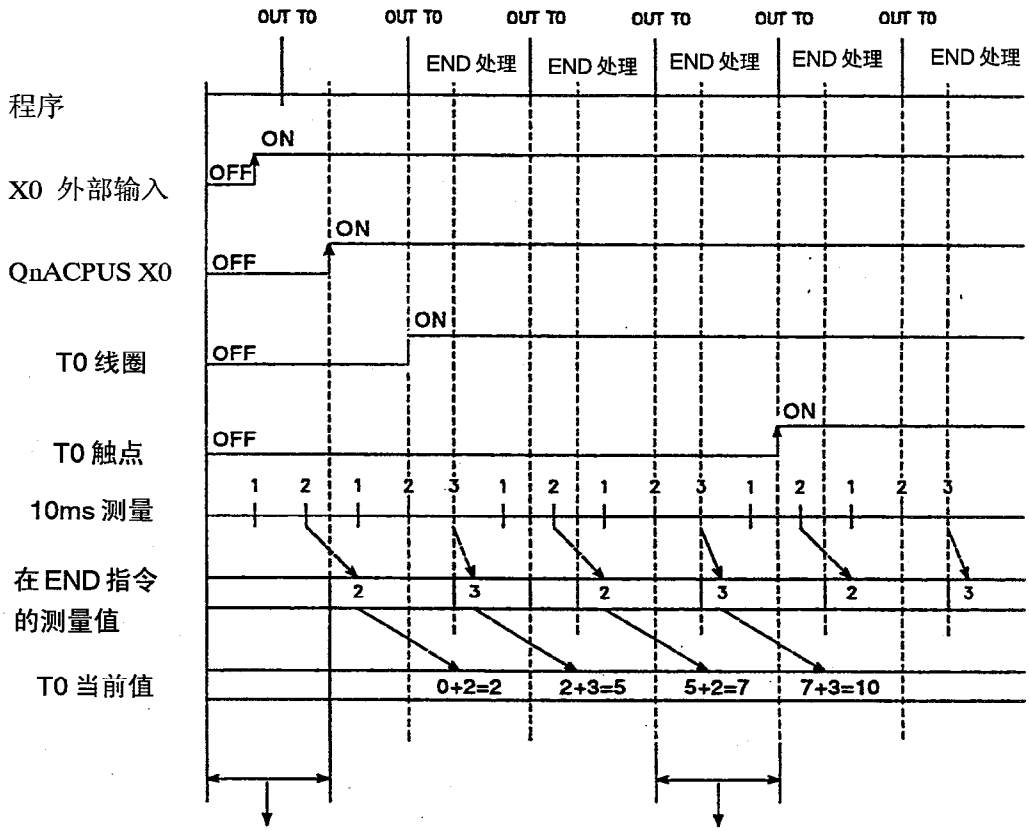
(b) 当执行了 OUT T[]指令时，当前值被累加到在 END 指令测量得到的扫描时间上。

如果在 OUT T[]指令执行时定时器的线圈是 OFF，那么当前值不被刷新。

[梯形图的例子]



[当前值刷新时序]

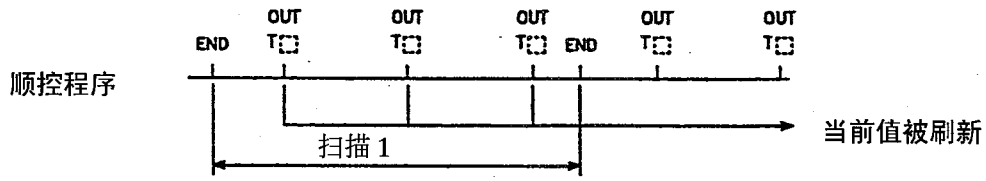


(c) 从输入（X）读取发生的那一点直到输出发生的那一点的定时器的响应精度是+2 个扫描。

使用定时器的注意事项

使用定时器时的几点注意事项：

（a） 在一个扫描中，一个给定的定时器只能被指定（用 OUT T□指令）一次。否则每个 OUT T□指令执行一次，定时器的当前值将被刷新一次，该测量就无意义了。



（b） 当一个定时器（例如 T1）的线圈是 ON 时，不可使用 CJ 指令跳过 OUT T1 指令。如果 OUT T□指令被跳过，那么定时器的当前值将不被刷新。

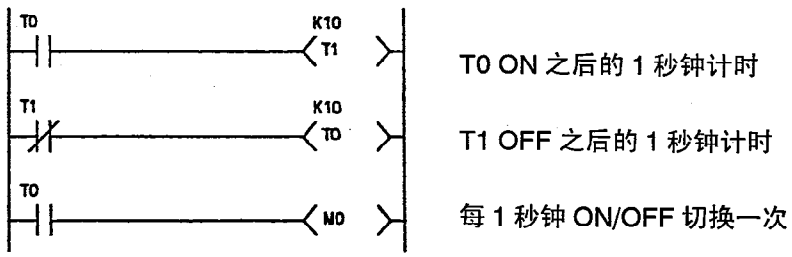
（c） 定时器不能被用在中断程序中。

（d） 如果定时器的设置值是“0”，那么当执行了 OUT T□指令后触点变成 ON。

（e） 在“时间到”之后，即使将设置值改成大于定时器中的当前值，那么“时间到”状态将仍保持有效，并且定时器不动作。

（f） 如果一个定时器用在一个低速执行程序中，那么低速扫描时间将被在 OUT T□指令执行时加算到当前值上。

（g） 使用两个定时器，经常使用如下梯形图作成 ON/OFF 电路。

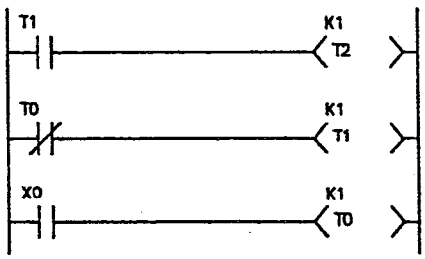


（h） 当创建使用一个定时器触点作为另一个不同定时器计数的触发器的程序时，在计数器随后计数时开始写程序。在下面所示的情况中，如果是按照与定时器计数的顺序相同的顺序进行编程的话，那么所有的定时器将在同一个扫描中处于 ON 状态。

- 当使用了一个设置值比扫描时间短的高速定时器时
- 当使用了一个设置值为“1”的低速定时器时

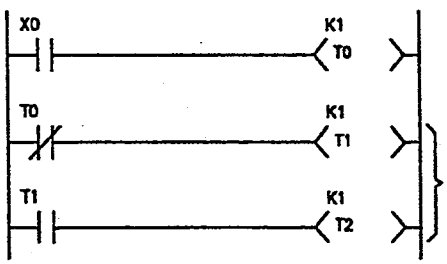
举例

- 当定时器 T0 到 T2 以与它们开始计数的顺序相反的次序编程时：



定时器 T2 是在 T1 触点变成 ON 的扫描之后的那个扫描期间开始计数的。
定时器 T1 是在 T0 触点变成 ON 的扫描之后的那个扫描期间开始计数的。
定时器 T0 是在 X0 变成 ON 时，开始计数的。

- 当定时器 T0 到 T2 以它们开始计数的顺序编程时：



定时器 T0 是在 X0 变成 ON 时，开始计数的。

当 T0 触点变成 ON 时，定时器 T1 和 T2 触点也变成 ON。

4.2.11 计数器（C）

QnACPU 计数器是“加法计数器”类型，当计数值等于设定值时，触点被切换成 ON。有两种计数器类型：对顺控程序中的输入条件的上升沿的数目计数的计数器，以及对中断因素进行数目计数的计数器。

要点

当执行了一条 OUT C[]指令时，就会发生下面的计数器处理：线圈 ON/OFF，当前值更新（计数值+1）以及触点 ON。

计数器的当前值更新和触点的 ON 切换处理不会发生在 END 处理中。

计数器（输入条件上升沿计数器）

（1）定义

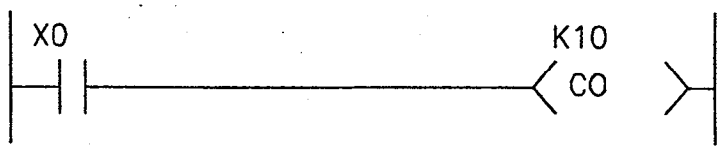
计数器是计算顺控程序中的输入条件上升沿数目的软元件。

（2）计数处理

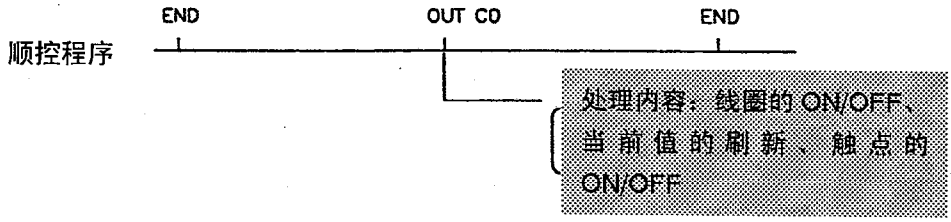
（a）当执行了一条 OUT C[]指令时，就会发生下面的计数器处理：线圈的 ON/OFF 切换、当前值更新（计数值+1）以及触点的 ON/OFF 切换。

计数器的当前值刷新和触点的 ON/OFF 切换处理不会发生在 END 处理中。

[梯形图的例子]



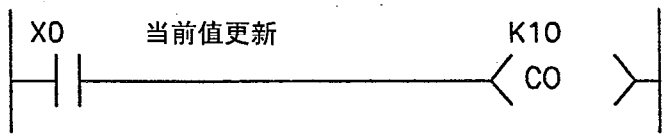
[OUT C0 指令的处理内容（X0：OFF→HON）]



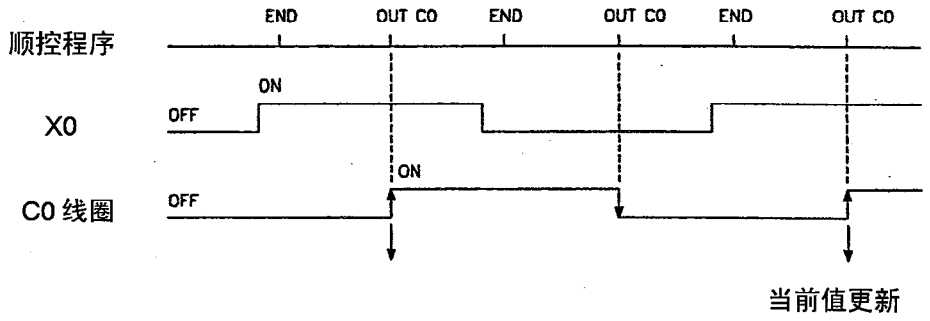
（b）当前值的更新（计数值+1）发生在 OUT C[]指令的上升沿（OFF--ON）。

在下列的 OUT C[]指令状态时，当前值不刷新：OFF、ON→ON、ON→OFF。

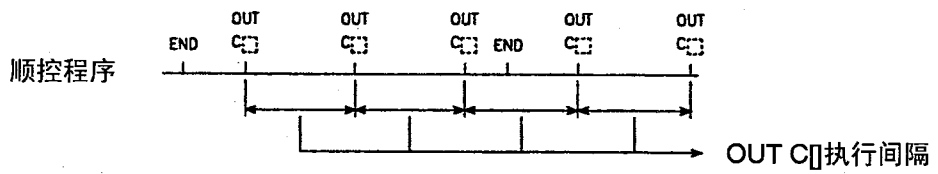
[梯形图的例子]



[当前值刷新时序]



（c）在一个扫描中可以使用多个计数器，以得到最大计数速度。在这种情况下，计数器的输入信号应该采用直接存取输入（DX[]）方式。*1

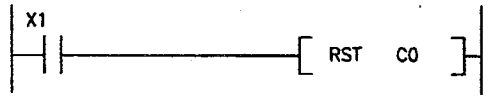


（3） 复位计数器

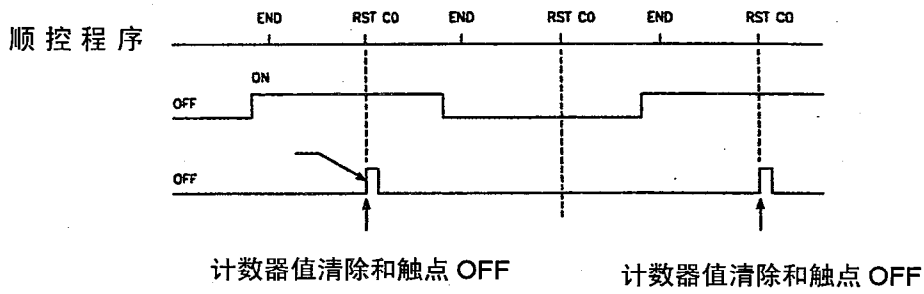
（a）即使在 OUT C 指令切换成 OFF 时，计数器的当前值也不被清除。使用 RST C[]指令来清除计数器的当前值并将触点切换成 OFF。

（b）在执行了 RST C[]指令时，计数值被清除并且触点被切换成 OFF。

[梯形图的例子]



[当前值刷新时序]



（4）最大计数速度

计数器只有在输入条件的 ON/OFF 时间比相应的 OUT C[]指令的执行时间间隔还要长的时候才计数。

最大计数速度用下面的公式计算：

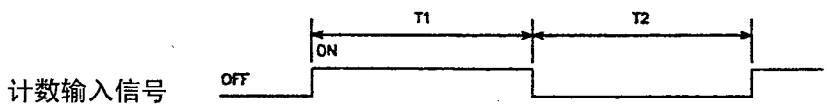
最大计数速度（Cmax）= n/100×1/t [次数/秒]

n：占空比（%）*2
t：OUT C[]指令的执行时间间隔

注释

1）*1：关于直接存取输入的详细资料请参见 4.2.1 节。

2）*2：“占空比”是用百分比值表示的计数输入信号的 ON-OFF 时间比。



当 T1≤T2 时：n=T1/(T1+T2)*100%　　当 T1≤T2 时：n=T1/(T1+T2)*100%

中断计数器

（1）定义

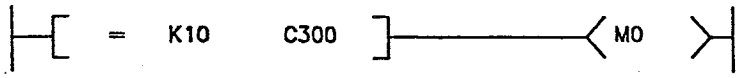
中断计数器是计算中断发生次数的软元件。

（2）计数处理

（a）当有一个中断发生时，中断计数器的当前值被刷新。因此没有必要特意创建一个包含中断计数器功能的程序。

（b）中断计数器操作需要的不只是简单的一个设置值指定。

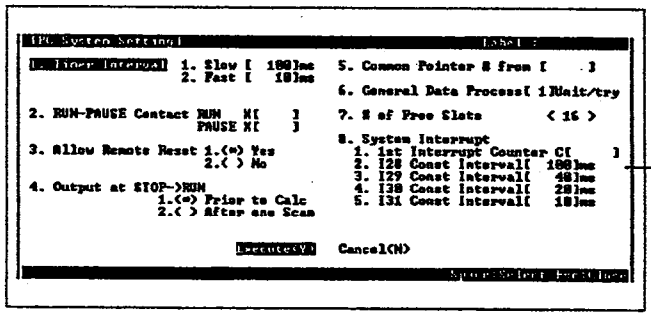
为了控制目的使用中断计数器，就必须使用比较指令（=，≤等）与设置值、进行比较。并根据比较结果将内部继电器（如 M 等）切换成 ON 或 OFF。下面的图表显示了这样一个简单的程序，在该程序中，在 IO 中断输入发生 10 次后之后，M0 被切换成 ON。（在该例中，假定“C300”是对应 IO 的中断计数器。）



（3）设置中断计数器

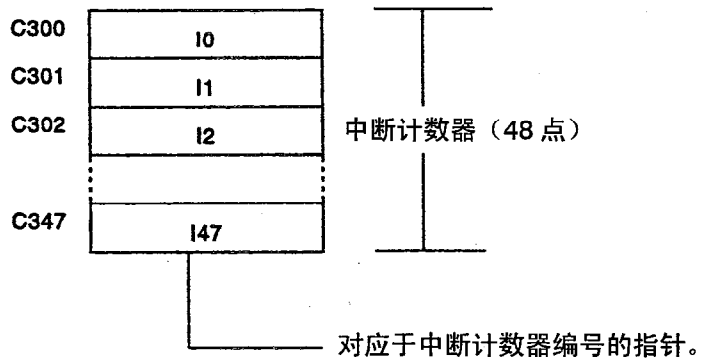
（a）为了使用中断计数器，首先，必须在参数中的“PC 系统设置”中指定中断计数器起始号。然后，再为中断计数器分配 48 点。

[“PC 系统设置”窗口]



在这里指定第一个中断计数器编号。

如果 C300 被指定为第一个中断计数器编号，那么编号 C300-C347 将被分配给中断计数器。



（b）为了使用中断计数器，必须在主程序中允许中断发生。

（4） 注意事项

（a） 一个中断指针不可同时用于执行中断计数器和中断程序操作。另外，指定为中断计数器的中断指针是不能够执行中断程序。

（b） 如果在发生一个中断时，下列所示的处理项目在进行中，那么计数操作将被延迟，直到这些项目处理完成。

即使当这些项目的处理在进行中又发生了相同的中断，也只有一个中断被计数。

- 在顺控程序指令执行期间
- 在 END 处理时的通用数据处理期间
- 在中断程序执行期间

（c） 中断计数器的最大计数速度取决于下面所示项目的最长处理时间。

- 在程序中使用到的指令中处理时间最长的指令
- 在 END 处理时的通用数据处理...最大 2ms
- 中断程序处理时间

最大计数速度=1/. [以上 3 个时间中最长的处理时间]+[500 μ sec× 中断计数器点数]. [PPS]

[举例]

最长的指令处理时间....0.3ms

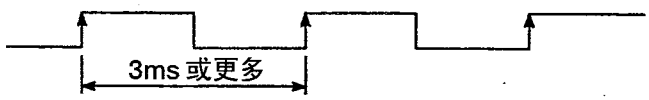
中断程序.....没有

中断计数器点数.....2

在这里，2ms（0.0002sec）的 END 处理时间是最高值：

最大计数速度=1/（0.002+0.0005×2）≈333[PPS]

基于这个最大计数速度,输入脉冲信号必须如下所示:



（d）使用太多的中断计数器将增加顺控程序的处理时间,从而可能导致“WDT 错误”。如果发生了这种情况，那么就减少中断计数器数目或者减少输入脉冲信号的计数速度

（e）中断计数器的计数值可以通过在 FEND 指令之前在顺控程序中使用 RST C[]指令来复位。

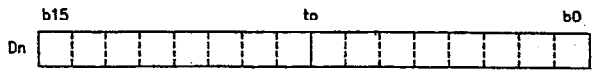
（f）中断计数器的计数值可以通过使用顺控程序的 MOV 指令读出。

4.2.12 据寄存器（D）

（1）定义

（a）数据寄存器是 QnACPU 中存储的数字数据（-32768 到 32767，或者 0000H 到 FFFFH）的存储软元件。

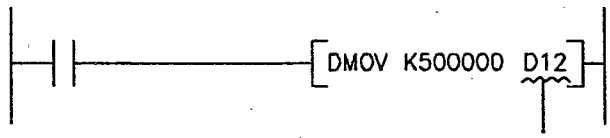
（b）数据寄存器由每点 16 比特位组成，其读和写操作以 16 位



为单位执行。

（c）如果数据寄存器被用为 32 位指令，那么数据将被保存在寄存器 Dn 和 Dn+1。数据的低 16 位保存在顺控程序指定的数据寄存器编号（Dn）中，而数据的高 16 位被保存在指定的寄存器编号+1（Dn+1）中。

例如，如果在 DMOV 指令中指定了寄存器 D12，那么低 16 位数据被保存在 D12 中，而高 16 位数据被保存在 D13 中。



D13	D12
高 16 点	低 16 点

（d） 被顺控程序保存的数据保持到另外一个保存操作发生为止。

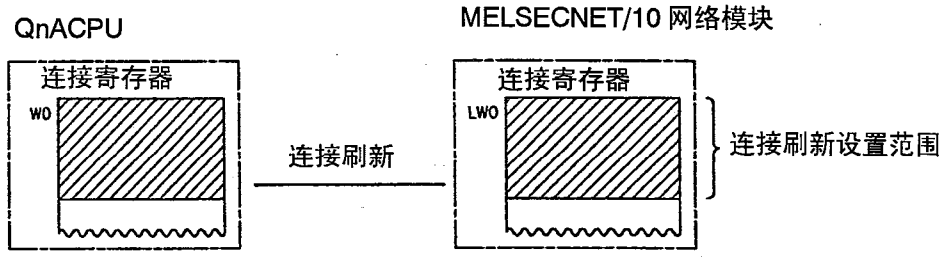
4.2.13 连接寄存器（W）

（1） 定义

（a）连接寄存器是用于网络数据刷新的 QnACPU 内部的存储器。

刷新数据来自于 MELSECNET/10 网络模块和 MELSECNET/10 网络模块的链接寄存器（LW）。

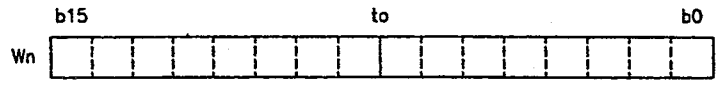
链接寄存器还可用于在 QnACPU 中存储数字数据（-32768 到 32767，或者 0000_H 到 $FFFF_H$）。



MELSECNET 数据连接系统和 MELSECNET/10 网络系统

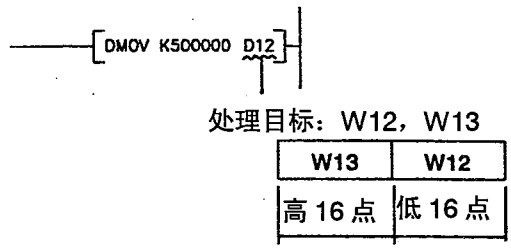
范围外的连接寄存器可以用作数据寄存器。

（b）连接寄存器由每点 16 个比特位组成，读和写操作以 16 位为单位执行。



（c）如果连接寄存器用于 32 位指令，那么数据将被保存在寄存器 Wn 和 Wn+1 中。数据的低 16 位保存在顺控程序指定的数据寄存器编号（Wn）中，而数据的高 16 位被保存在指定的寄存器编号+1（Wn+1）中。

例如，如果在 DMOV 指令中指定了寄存器 W12，那么低 16 位数据被保存在 W12 中，而高 16 位数据被保存在 W13 中。



（d）被顺控程序保存的数据保持到另一个保存操作发生为止。

（2） 在 MELSECNET/10 网络系统中使用连接寄存器

（a）如果连接寄存器被用在一个 MELSECNET/10 网络系统中，那么主站的数字数据可以被其他站读取并在那里使用。

在 MELSECNET/10 网络系统中使用了连接寄存器，就允许在主控站与寻常站，以及在寻常站之间进行数字数据的传递。

（b）为了在 MELSECNET/10 网络系统中使用连接寄存器，就必须在主控站进行网络参数设置。没有在网络参数设置中进行设置的连接寄存器可以被用于数据寄存器。

（3） 在 MELSECNET 数据连接系统中使用连接寄存器

（a）如果连接寄存器被用在 MELSECNET 数据连接系统中，那么主站的数字数据可以被其他的站读取并在那里使用。

在 MELSECNET 数据连接系统中使用了连接寄存器，就允许在主站与一个从站，以及在从站之间进行数字数据的传递。

（b）为了在 MELSECNET 数据连接系统中使用连接寄存器，就必须在主控站进行网络参数设置。没有在网络参数设置中进行设置的连接寄存器可以被用于数据寄存器。

注释

（1）关于网络参数的详细资料，请参考 QnA/Q4AR MELSECNET/10 网络系统参考手册。

（2）关于连接参数的详细资料，请参考 MELSECNET 和 MELSECNET/B 数据连接系统参考手册。

4.2.14 特殊连接寄存器（SW）

（1） 定义

（a）特殊连接寄存器被用在 MELSECNET/10 网络模块和用户程序之间进行数据传递。

（b）因为数据连接时的信息被保存成在特殊连接寄存器（SW）中的数字数据，所以特殊连接寄存器可作为确定故障位置和原因的工具。

（2）特殊连接寄存器点数

每个 MELSECNET/10 网络模块中均有从 SW0 到 SW7FF 为止的 2048 点特殊连接寄存器。QnACPU 特殊寄存器的“点数”缺省设置值是每个模块 512 点。如下面所示：

	特殊连接寄存
SW0 to SW1FF	用于第 1 个网络模块
SW200 to SW3FF	用于第 2 个网络模块
SW400 to SW5FF	用于第 3 个网络模块
SW600 to SW7FF	用于第 4 个网络模块

注释

1）关于用于 QnACPU 的特殊连接寄存器的详细资料，请参考 QnACPU 编程手册（通用指令）。

4.3 内部系统软元件

内部系统软元件是用于系统操作的软元件。内部系统软元件的分配位置和大小是固定的，并且不能被用户改变。

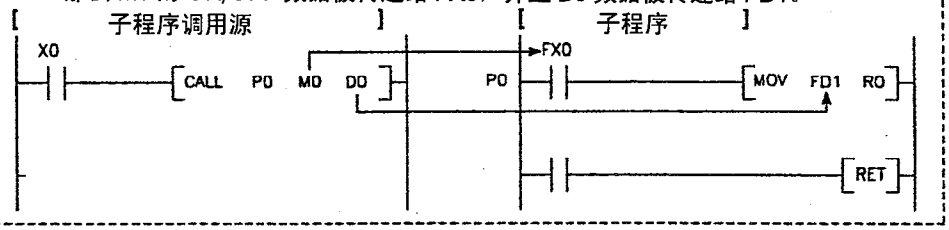
4.3.1 功能软元件（FX，FY，FD）

（1） 定义

(a) 功能软元件是用于带有参数的子程序的软元件，在带有参数的子程序和该子程序的调用源之间进行数据的传递。

举例：

如果 FX0 和 FD1 用于子程序中，并且如果 M0 和 D0 被子程序的调用指令所指定，那么 M0 的 ON/OFF 数据被传递给 FX0，并且 D0 数据被传递给 FD1。



(b) 因为用于每一个子程序调用源的功能软元件都可以被设置，所以在使用子程序时，而无须考虑其他子程序的调用源。

（2） 功能软元件的类型

由三种功能软元件类型：功能输入软元件（FX）、功能输出软元件（FY）以及功能寄存器软元件（FD）。

(a) 功能输入软元件（FX）

- 这些软元件是用于为一个子程序指定 ON/OFF 数据的输入。
- 在子程序中，这些被子程序调用命令指定的比特位数用于计算。
- 所有的 QnACPU 比特位数据软元件都可被使用。

（b）功能输出软元件（FY）

- 对于带有参数的子程序，运行结果被保存在指定的软元件中。
- 所有的位数据指定软元件除了 QnACPU 输入（X，DX）外都可以被使用。

注释

1） 功能软元件只能在带有一个参数的子程序执行期间被监视。

当监视功能软元件时，在带有使用功能软元件的参数的程序中指定步数。

（c） 功能寄存器

- 这些软元件用于指定在子程序调用源和子程序之间的数据传输。
- 功能软元件的输入/输出条件是由 QnACPU 自动确定的。如果子程序数据是源数据，那么该数据就被指定为子程序输入数据。如果子程序数据是目标数据，那么该数据就被指定为子程序输出数据。
- 1 个点占 4 个字。
- QnACPU 的字数据指定软元件可以被使用。

注释

1） 关于功能软元件的使用的详细资料，请参考 QnACPU 编程手册（通用指令）。

4.3.2 特殊继电器（SM）

（1） 定义

（a） 特殊继电器是这样一种固定应用的 QnACPU 内部继电器。它们用于在 QnACPU 系统和用户程序之间进行 ON/OFF 数据的通信。

（2） 特殊继电器分类

特殊继电器是根据它们的应用进行分类的，如下所示。

（a） 用于故障诊断：SM0-SM199
（b） 系统信息：SM200-SM399
（c） 系统时钟/系统计数器：SM400-SM499
（d） 扫描信息：SM500-SM599
（e） 存储卡信息：SM600-SM699
（f） 有关的指令：SM700-SM799
（g） 用于调试：SM800-SM899
（h） 锁存区域：SM900-SM999
（i） 用于 ACPU：SM1000-SM1299

注释

1） 关于可被 QnACPU 使用的特殊继电器的详细资料，请参考 QnACPU 编程手册（通用指令）。

4.3.3 特殊寄存器（SD）

（1） 定义

（a） 特殊寄存器是这样一种固定应用的 QnACPU 内部继器。它们用于在 QnACPU 系统和用户程序之间进行 ON/OFF 数据的通信。

（2） 特殊寄存器分类

特殊寄存器是根据它们的应用进行分类的，如下所示。

（a）用于故障诊断：SM0-SM199
（b）系统信息：SM200-SM399
（c）系统时钟/系统计数器：SM400-SM499
（d）扫描信息：SM500-SM599
（e）存储卡信息：SM600-SM699
（f） 有关的指令：SM700-SM799
（g）用于调试：SM800-SM899
（h）锁存区域：SM900-SM999
（i） 用于 ACPU：SM1000-SM1299

注释

1） 关于可被 QnACPU 使用的特殊继电器的详细资料，请参考 QnACPU 编程手册（通用指令）。

4.4 连接直接软元件（J□\□）

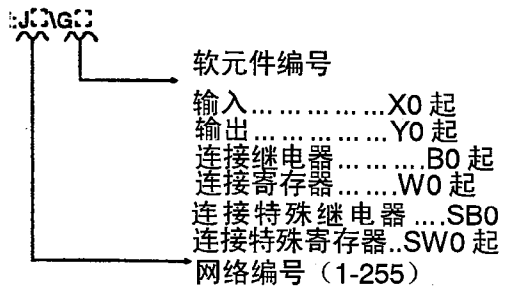
（1）定义

（a）在 QnACPU 和 MELSECNET/10 网络系统模块之间的数据刷新（数据传递）操作在 END 处理时进行。

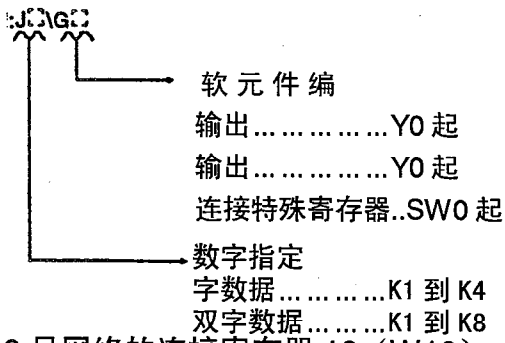
（b）指定方法

1） 连接直接软元件被网络编号和软元件编号所指定。

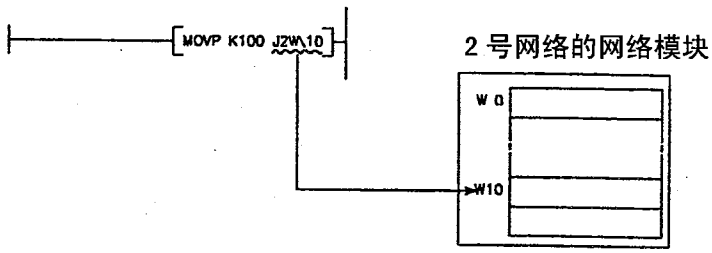
指定方法：



2) 连接直接软元件用网络号码和软元件号来指定。
指定方法：



举例：对于 2 号网络的连接寄存器 10（W10），指定方法就是“J2\W10”。



3) 对于一个位软元件（X，Y，B，SB），需要数字指定。
指定举例：J1/1X0，J10/K4B0

(2) 指定范围

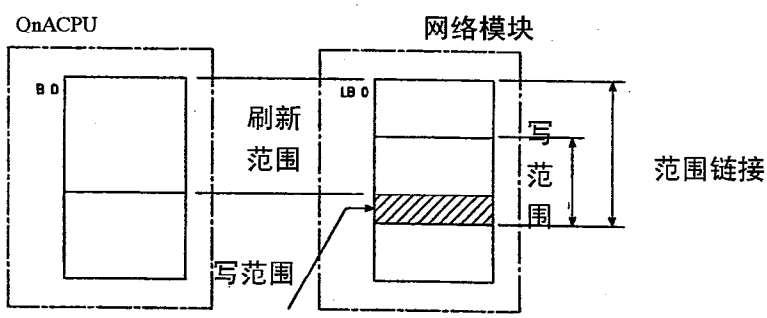
对网络模块中的所有的连接软元件进行连接直接软元件指定是可能的。

[超出网络刷新参数指定的范围的软元件也可以被指定。]

(a) 写操作

1)请在网络刷新范围外而却是连接软元件范围内进行写操作。其中连接软元件范围由“网络参数”的“通用参数设置”来指定，它包含网络刷新范围。

然而，当在刷新范围之外的一个输出被转成 ON 时，即使被设置成 STOP 状态，它也将不被刷新，因此它将不变成 OFF。



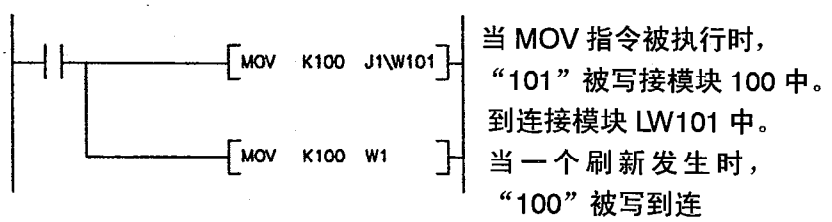
2) 尽管写操作在连接软元件范围（由刷新参数指定）的“刷新范围”部分也是可能的，但是当一个刷新操作发生时，连接模块的链接软元件数据将被重写。为了避免这种情形发生，请先在刷新范围内的连接软元件（QnACPU 中）中写入相应的数据（参照下例）。

[刷新参数设置]

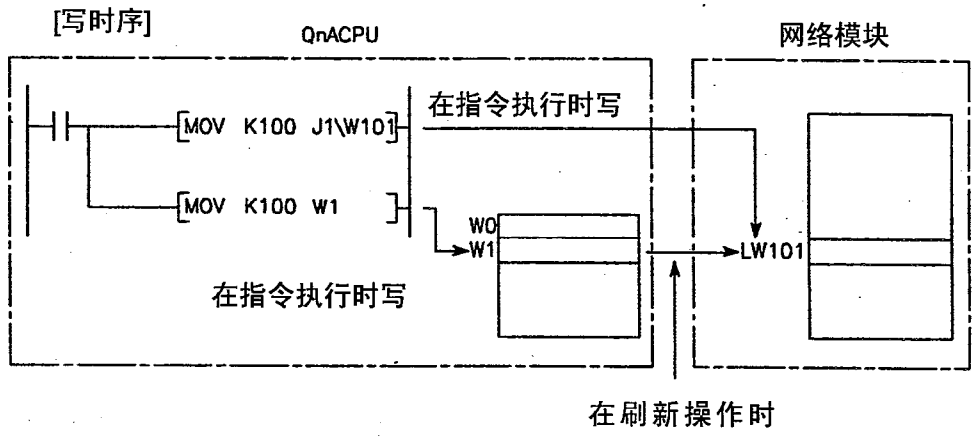
网络编号：1

QnACPU（W0 到 W3F）〈---〉网络模块（LW100 到 LW13F）

[顺控程序]



[写时序]



3) 当使用一个连接直接软元件将数据写到另一个站的写范围时，从那个站接收到的数据将替代被写的数据。

b) 读操作

在网络模块的全部连接软元件范围内，被连接直接软元件读取是可能的。

(3)“连接直接软元件”和“连接刷新”之间的差别

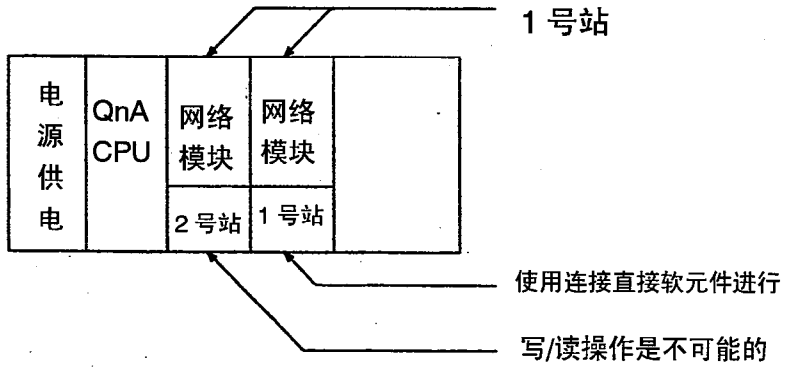
“连接直接软元件”和“连接刷新”之间的差别如下表 4.4 所示。

表 4.4 “连接直接软元件”和“连接刷新”之间的差别

项目		连接直接软元件	连接刷新
程序符号方法	连接继电器	J□/K4B0 或后面的	B0 或后面的
	连接寄存器	J□W0 或后面的	W0 或后面的
	连接特殊继电器	J□/K4SB0 或后面的	SB0 或后面的
	连接特殊寄存器	J□SW0 或后面的	SW0 或后面的
步数		2 步	1 步
网络模块存取范围		所有的网络模块连接软元件	参数指定的范围
存取数据保证范围		字单位（16 比特位）	

要点

(1) 每一个网络编号只有一个网络模块能对连接直接软元件进行写/读操作。
如果在相同的网络编号下安装了两个或更多的网络模块，那么具有最低的第一个 I/O 编号的网络模块将是能够使用连接直接软元件进行写/读出操作的模块。
例如，如果 1 号站和 2 号这点网络模块被安装在 1 号网络中，如下图所示，那么 2 号站的网络模块将进行连接直接软元件的操作



注释

1) 系统参考手册。
2) 关于网络参数、通用参数和网络刷新参数的详细资料，请参考以下手册：
详细信息关于 MELSECNET/10 网络系统的详细资料，请参
QnA/Q4AR MELSECNET/10 网络
设置过程
SW□IVD-GPPQ 类型 GPP 功能软件包操作手册（离线）

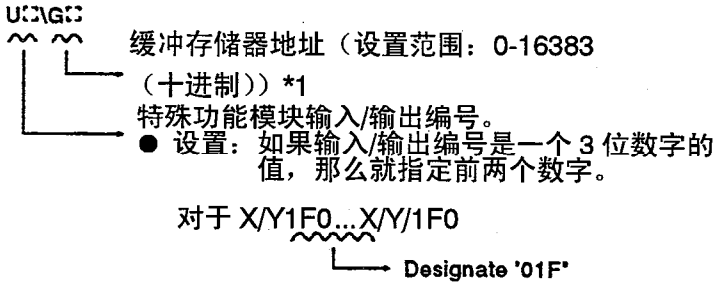
4.5 特殊功能模块软元件（U□\G□）

(1) 定义

(a) 特殊功能模块软元件允许 QnACPU 直接存取安装在主基板和扩展槽上的特殊功能模块的缓冲寄存器。而不可存取安装在 MELSECNET/10 网络系统或 MELSECNET（II，/B）数据连接系统的远程站上的特殊功能模块中的缓冲寄存器中。

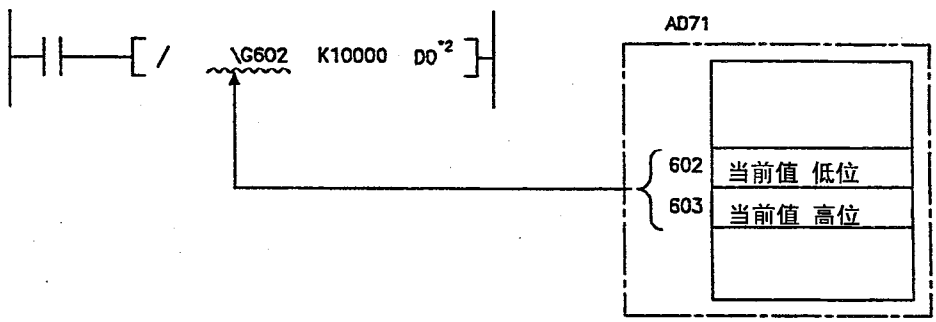
(b) 特殊功能模块软元件由特殊功能模块输入/输出编号和缓冲存储器地址来指定。

指定方式：



● 设置范围：00H 到 FEH

为了将安装在 0 基板的 0 号槽的 AD71 定位模块的 X 轴的当前值（缓冲存储器：602，603）转换成“mm”单位（1/1000），并且将它保存到 D0 和 D1 中，可作如下指定：



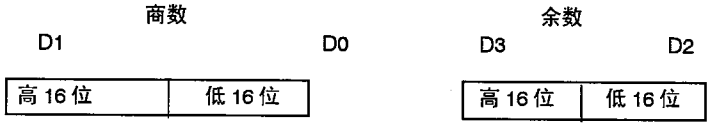
(2) 处理速度

特殊功能模块软元件的处理速度是“FROM/TO 指令的处理速度”和“指令处理速度”的总和。
如果在一个顺控程序中使用了两次或两次以上相同特殊功能模块的相同缓冲存储器，那么可以通过使用 FROM 指令将该缓冲存储器的数据读入到 QnACPU 中来提高处理速度。

注释

1）*1：关于缓冲存储器地址及其应用的详细资料，请参考使用的特殊功能模块手册。

2）*2：商数和余数被保存在 D0 和 D3 中。



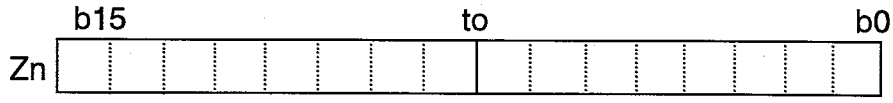
4.6 变址寄存器（Z）

（1）定义

（a） 变址寄存器在顺控程序中用作间接设置（变址修饰）指定。

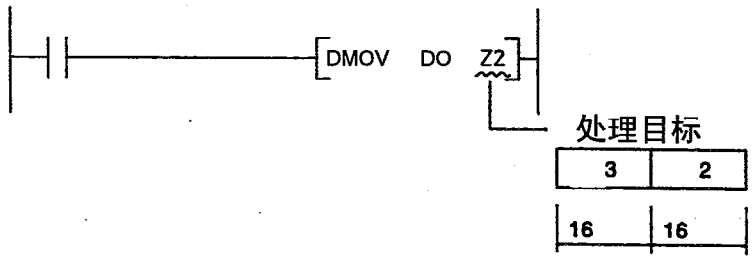
（b） 有 16 个变址寄存器（0-Z15）。

（c） 变址寄存器由每点 16 位组成，其读和写操作以 16 位为单位。



(d) 如果变址寄存器被用于 32 位指令，那么数据将被保存在寄存器 Zn 和 Zn+1。数据的低 16 位保存在顺控程序指定的变址寄存器编号（Zn）中，而数据的高 16 位被保存在指定的变址寄存器编号+1（Zn+1）中。

例如，如果在 DMOV 指令中指定了寄存器 Z2，那么低 16 位数据被保存在 Z2 中，而高 16 位数据被保存在 Z3 中。



(2) 在程序切换时的变址寄存器处理

当从一个扫描执行程序或低速执行程序切换到另一种程序类型时，变址寄存器（Z0-Z15）的数据被保存（保护）起来。

这个数据在切换回到扫描执行程序或低速执行程序时，被复位。

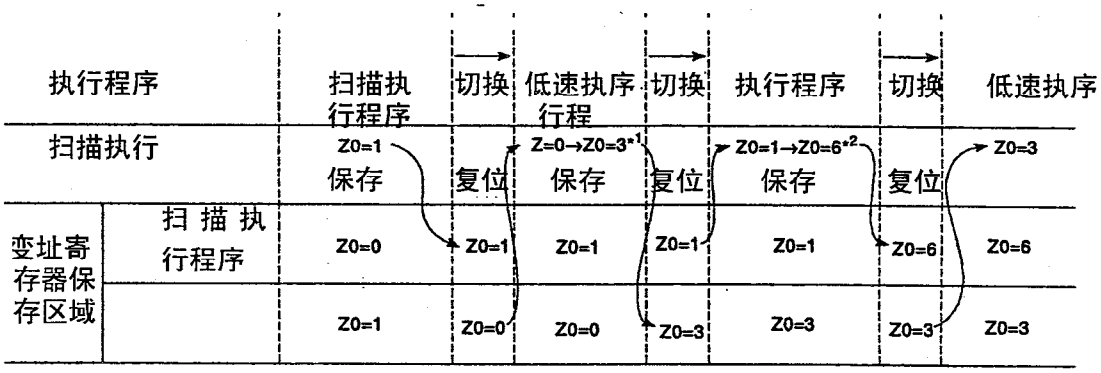
注释

1）关于使用变址寄存器的变址修饰的详细资料，请参考 QnACPU 编程手册（通用指令）。

（a） 在扫描执行程序和低速执行程序之间进行切换

1） 当从一个扫描执行程序切换到一个低速执行程序时，扫描执行程序的变址寄存器数据被保存，并且低速执行程序的变址寄存器数据被复位。

2） 当从一个低速执行程序切换到一个扫描执行程序时，低速执行程序的变址寄存器数据被保存，并且扫描执行程序的变址寄存器数据被复位。



*1：对于一个低速执行程序，Z0 被改变成 3。

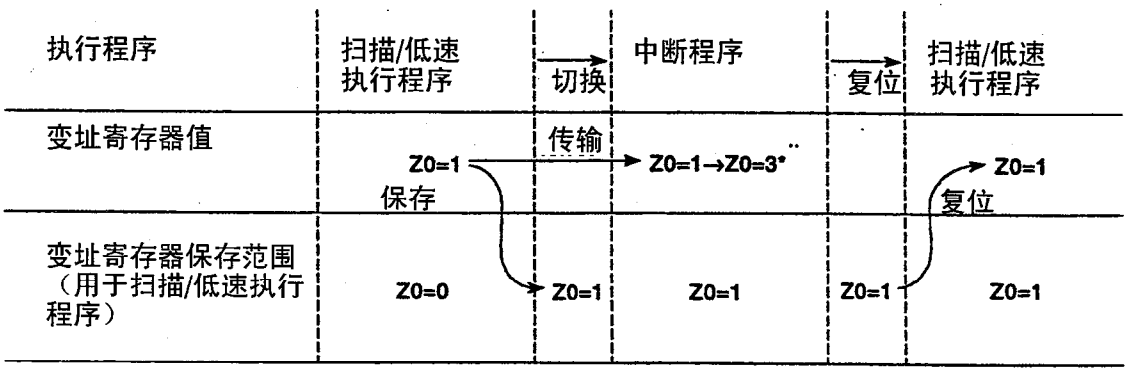
*2：对于一个扫描执行程序，Z0 被改变成 6。

字软元件应该被用于在扫描执行程序和低速执行程序之间进行变址寄存器数据的交换。

（b） 在扫描/低速执行程序和中断程序之间进行切换

1） 当扫描/低速执行程序被切换成中断程序时，扫描/低速执行程序的变址寄存器值首先被保存，然后传送给中断程序。

2） 当中断程序被切换成扫描/低速执行程序时，保存的变址寄存器值被复位。



*：在中断程序，Z0 被改变成 3。

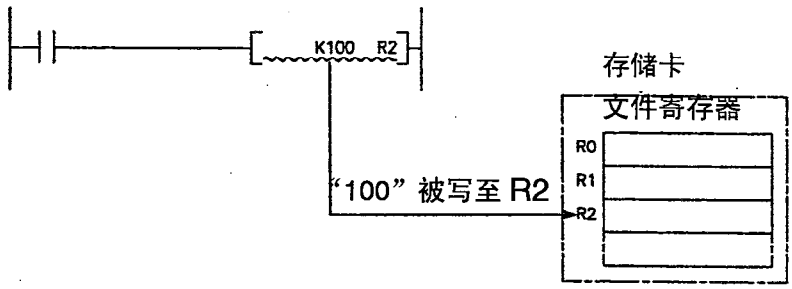
在将中断程序的变址寄存器数据向扫描执行程序或低速执行程序传递时，请使用字软元件。

4.7 文件寄存器（R）

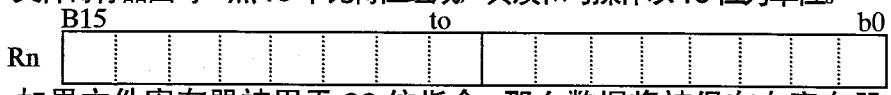
(3) 定义

（a） 文件寄存器就是数据寄存器的扩展软元件。

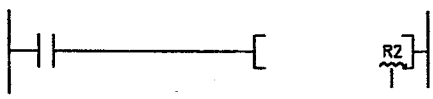
（b） 文件寄存器被保存在 QnACPU 存储卡的文件中。
因此，当使用文件存储器时，需要有存储卡。



(c) 文件寄存器由每一点16个比特位组成，其读和写操作以16位为单位。



(d) 如果文件寄存器被用于 32 位指令，那么数据将被保存在寄存器 Rn 和 Rn+1。
数据的低 16 位保存在顺控程序指定的数据寄存器编号（Rn）中，而数据的高 16 位被保存在指定的寄存器编号+1（Rn+1）中。
例如，如果在 DMOV 指令中指定了寄存器 R2，那么低 16 位数据被保存在 R2 中，而高 16 位数据被保存在 R3 中。



处理对象： R2，R3

R2	R3
高 16 位	低 16 位

(2) 文件寄存器容量

每一个文件最大可以被扩展成一个以 1 块（32K 字）为单位的 32 个块（1018K 字）。（但最后的块只有 26 字）
然而，允许的扩展块数目根据被使用的存储卡的容量大小以及保存在存储卡中的顺控程序的大小而不同，

注释

1) 关于 QnACPU 存储卡的详细资料，请参见 2.3 节。

(3) 不同存储卡类型不同的存储卡存取方法

采用了以下三种类型的存储卡来保存一个文件寄存器。存储卡的存取方法根据存储器类型的不同而不同。

RAM

（a）使用一个程序进行读/写操作是允许的。
（b）通过软元件设置的 PC 读/写操作是允许的。
（c）文件寄存器数据可以通过下列方法中的任意一个被改变。
1）通过 GPP 的在线测试操作
2）通过 QC24 和 QE71 的专用协议的批量写操作命令
3）从一个 GOT900 系列的软元件写操作或随机写操作命令

E^2PROM

（a）通过命令进行读操作是允许的，写操作是不允许的。

（b）通过软元件设置的 PC 读/写操作是允许的。

（c）文件寄存器数据可以通过下列方法中的任意一个来改变。

1）通过 QC24 和 QE71 专用协议的批写命令

（CPU 必须在 STOP/PAUSE 状态下。）

下面的 CPU 软件版本支持这种方法。

CPU 类型	软件版本号
QnA	L 或更新的
Q2AS（H）	T 或更新的
Q4AR	S 或更新的

闪烁 ROM

（a）使用程序进行读操作是允许的，但是使用程序进行写操作是不允许的。

（b）通过软元件设置的 PC 读/写操作是不允许的。

（c）通过一个读/写器进行文件读/写操作是允许的。

（有关详细资料，请参考 GPP 的操作手册。）

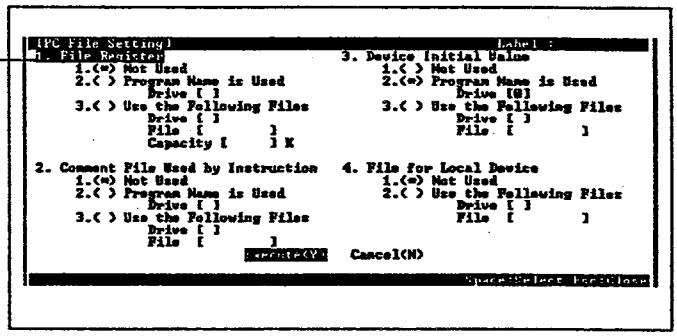
(4) 指定使用的文件寄存器

存储卡可以容纳总共 124 个文件寄存器。

要在顺控程序中使用的存储卡文件寄存器是由 PC 的文件设置参数决定的。

[PC 文件设置屏幕]

指定要使用的文件寄存器

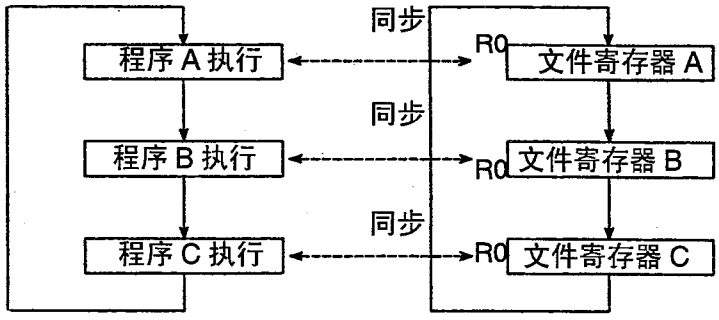


(a) 当选择了“不使用”时
选择该选项是为了指定在顺控程序中将要使用那些文件寄存器。
QDRSET 指令被用来指定将要使用的那些文件寄存器。

(b) 当选择了“使用程序名”时
当文件寄存器具有与使用的顺控程序相同的文件名时，选择该选项。
如果执行中程序改变了，那么文件寄存器就自动改变以符合新的程序名。
在一些情况下，将文件寄存器作为局域软元件使用是很方便的，这些局域软元件只能与当前正在被执行的程序一起使用。

举例：

当文件寄存器（A-C）具有与将要使用的程序（A-C）相同的名字时，操作如下所示。
- 在程序 A 执行时...文件寄存器 A 被存取
- 在程序 B 执行时...文件寄存器 B 被存取
- 在程序 C 执行时...文件寄存器 C 被存取



在以下所有情况下，在一个扫描结束时被执行的文件名字的文件寄存器被存取。

然而，如果在扫描结束时被执行的程序没有文件寄存器，那么就不能存取文件寄存器。

- 从一个外围设备存取
- 从网络中的的另一个站存取
- 从一个串行通讯模块存取

(c) 使用下列文件

当一个给定的文件寄存器要被所有的执行程序共享时，选择该选项。

通过指定文件寄存器“驱动器”、“文件”和“容量”设置，当一个 QnACPU RUN 状态被创建时，将创建参数指定的文件寄存器。

(5) 在 QnACPU 中寄存文件寄存器文件

（a） 如果在 PC 文件设置屏幕上选择了不同于“使用下列文件”的选项，那么文件寄存器文件必须寄存在 QnACPU 中。（如果文件寄存器具有与将要使用的程序相同的文件名，那么它们应该被寄存在由 PC 文件设置参数指定的驱动中。）

1） 当文件寄存器文件没有被寄存到 QnACPU 中时：

即使对文件寄存器进行读/写操作，也不会有错误出现。然而，从文件寄存器读出的所有数据都为“FFFFH”。

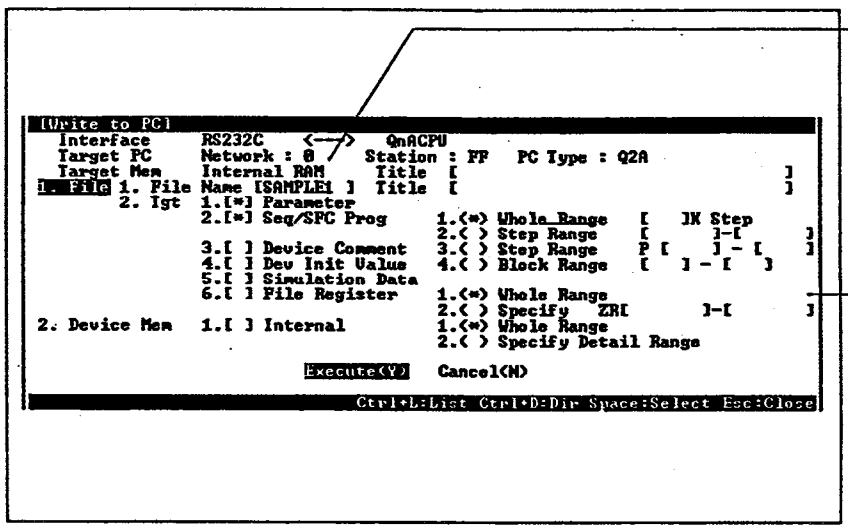
2） 对设置范围之外的文件寄存器的读/写操作：

即使对这些文件寄存器进行读/写操作，也不会有错误出现。然而，从文件寄存器读出的所有数据都为“FFFFH”。

（b）为了在 QnACPU 中寄存文件寄存器文件，在外围设备的“PC 写操作屏幕”上指定文件名和文件寄存器的大小，然后将该数据写到 QnACPU 中。

文件寄存器大小从 ZR0 起指定，以 1K 点（1024 点）为单位。*1

[PC 写操作屏幕]



(6) 文件寄存器大小检查

（a）如果在 QnACPU 中使用了文件寄存器，那么当文件寄存器的大小等于或大于实际需要的范围时，会对文件寄存器进行写/读操作。

- 一个文件寄存器的大小检查应该在使用了文件寄存器的程序的第 0 步执行。
- 在使用了 QDRSET 指令切换到另一个文件寄存器文件后，执行文件大小检查。
- 当使用 RSET 指令切换块时，要在执行 RSET 指令之前检查确定切换目标块具有 1K 点或更多的空间。

（文件寄存器大小） 〉[32K 点×（切换块编号）+1K 点]

（b）可用的文件寄存器大小可以在文件寄存器容量存储寄存器（SD647）中检查。*2

文件寄存器大小以 1K 点为单位保存在 SD647 中。

（一个文件寄存器大小的“小于 1K 点”的剩余部分不被保存。为了确保能准确地检查“使用范围”，一定要以 1K 点（1024 点）为单位指定文件寄存器设置。）

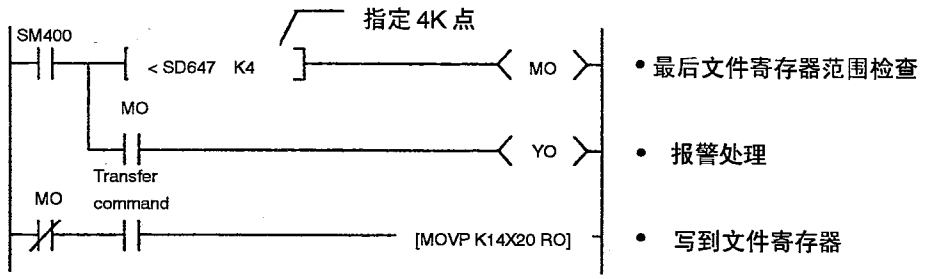
（c）检查文件寄存器大小

1）首先要弄清楚每一个顺控程序的文件寄存器的大小。

2）从 SD647（顺控程序）中的总的文件寄存器大小设置，可以确定文件寄存器的大小是否超出了使用的点数。

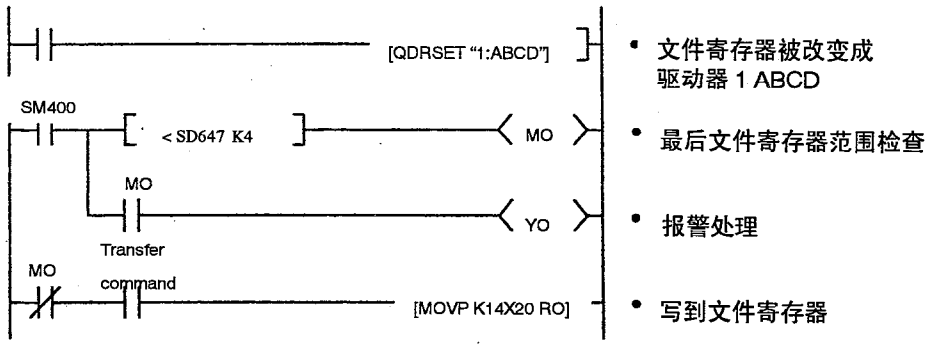
[程序举例 1]

在每一个程序的开头检查文件寄存器的“使用范围”。



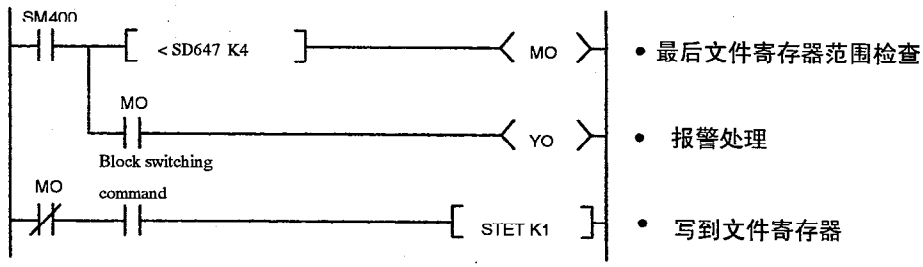
[程序举例 2]

在执行了 QDRSET 指令后检查文件寄存器的“使用范围”。



[程序举例 3]

块切换



注释

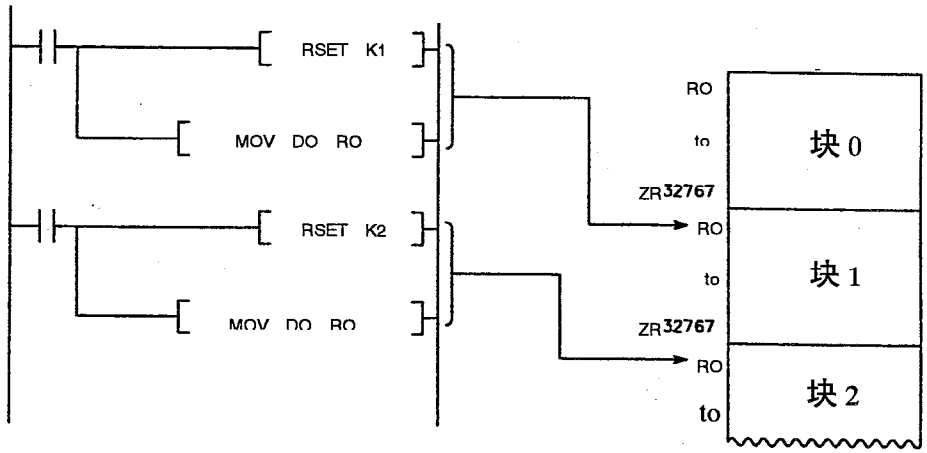
1）*1：即使文件寄存器设置范围没有从 ZR0 指定，文件的创建范围也将从 ZR0 开始，并在最后的文件编号结束。
例如，如果文件寄存器的写范围被指定为 ZR1000 到 ZR2047，那么文件的创建范围将是 ZR0 到 ZR2047。
由于在这种情况下，从 ZR0 到 ZR999 的数据将是不定值的，所以请先从 ZR0 赋值。

2）*2：当切换到另一个文件寄存器时，新的文件寄存器文件的大小被保存在 SD647 中。

(7) 文件寄存器指定方法

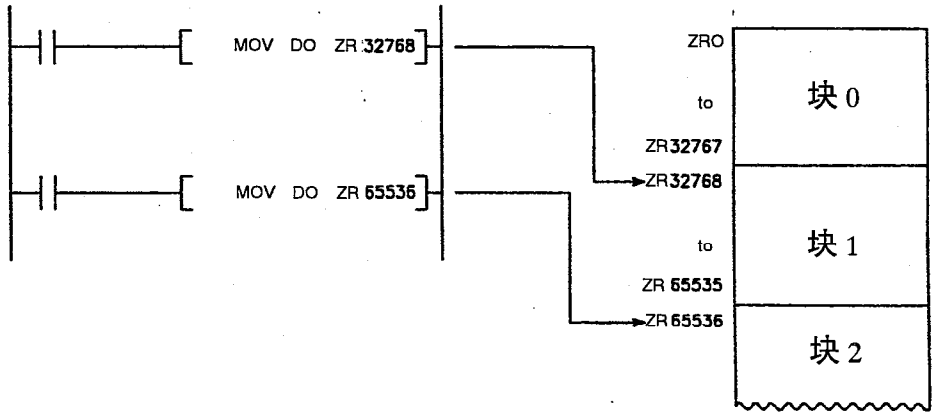
（a）块切换格式

块切换格式以 32K 点（R0-R32767）为单位，指定了文件寄存器的点数。
如果使用了多个块，那么就使用 RSET 指令来切换到另一个块编号进行进一步的文件寄存器的设置。
在每一个块中的 R0 到 R32767 范围内指定设置。



(b) 序列编号存取格式

该格式用于指定超出 32K 点数的没有切换块编号的文件寄存器设置。多个文件寄存器块可以被用作一个连续的文件寄存器。



4.8 嵌套（N）

（1） 定义

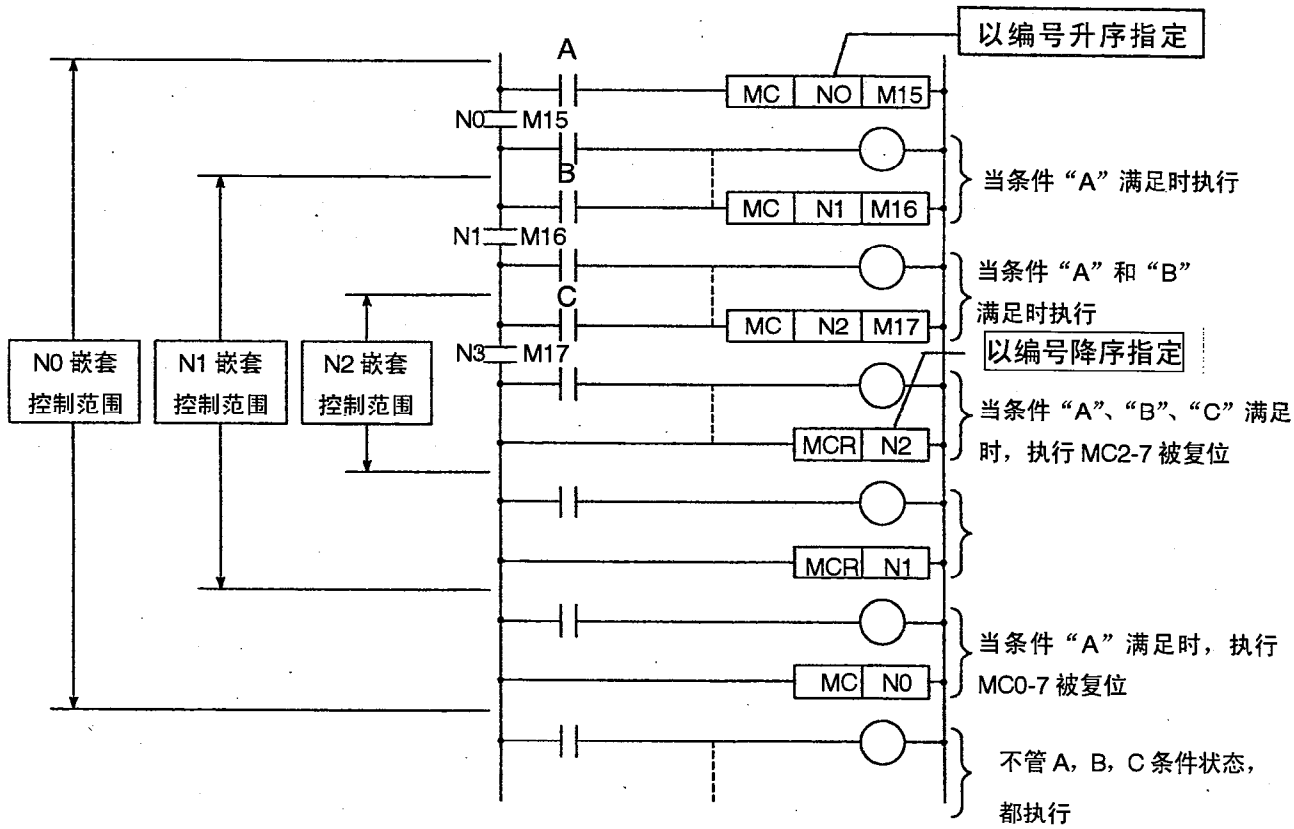
嵌套软元件是与主控制指令一起使用的。

（2） 主控制指定方法

主控制指令被用来打开和关闭梯形图的通用总线，这样梯形图的切换就可以通过顺控程序被有效地执行。

它是用 MC 和 MCR 主控制指令来指定的。

关于如何使用主控制的详细资料，请参考 QnACPU 编程手册（通用指令）。



4.9 指针

（1） 定义

指针是用在跳转命令中的软元件。总共有 4096 个指针可以使用（对所有的程序）。

（2） 指针应用

（a） 指针用在跳转命令（CJ、SCJ、JMP）中以指定跳转目标和标号（跳转目标的开头）。

（b） 指针用于子程序调用命令（调用、调用 P）中以指定调用目标和标号（子程序的开头）。

（3） 指针类型

有两种指针类型：独立于 QnACPU 程序使用的“局部指针”，以及用于从 QnACPU 中执行的所有程序中调用子程序的“通用指针”。（关于“局部指针”的详细资料请参见 4.9.1 节，关于“通用指针”的详细资料请参见 4.9.2 节。）

4.9.1 局部指针

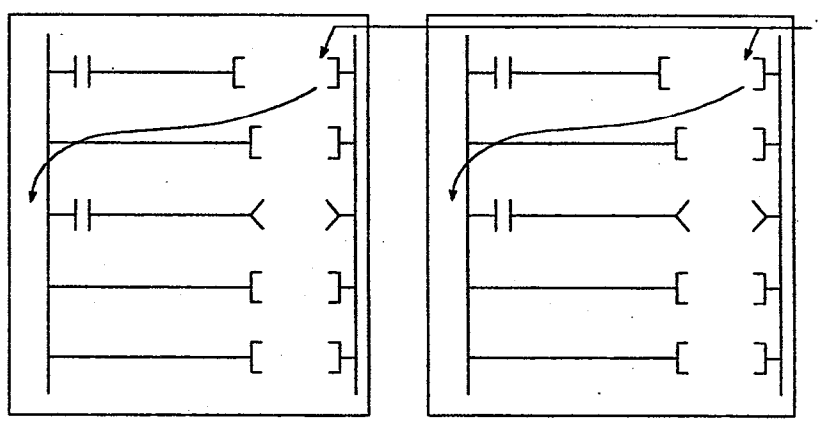
（1） 定义

（a） 局部指针是可以独立使用在 QnACPU 程序跳转命令和子 程序调用命令的指针。局部指针不能在其他的程序跳转命令和子程序调用命令中使用。

（b） 相同的指针编号可以被用在不同的程序中。

程序 B　　程序 B

使用了相同的指针



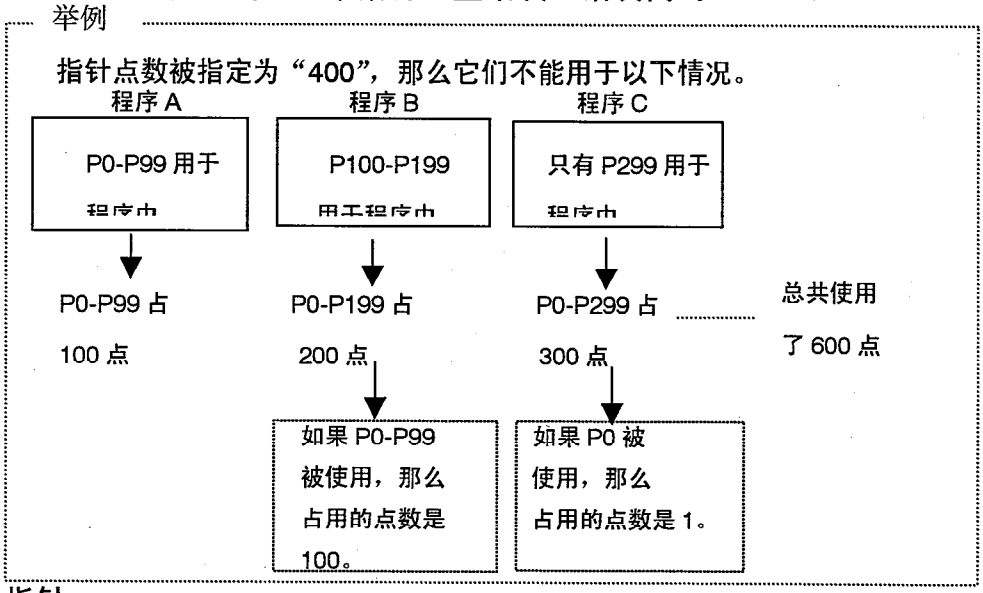
(2) 局部指针点数

最多可达指定点数的局部指针可以在所有的程序中被分配使用。最高局部指针编号就是在每一个程序中的“已用的点数”的上限。

因此，当局部指针在几个程序中使用时，指针设置应该从 P0 开始。

如果总的指针数目（对所有的程序）超过了指定的范围，那么就会出现一个指针配置错误（错误代码：4020）。

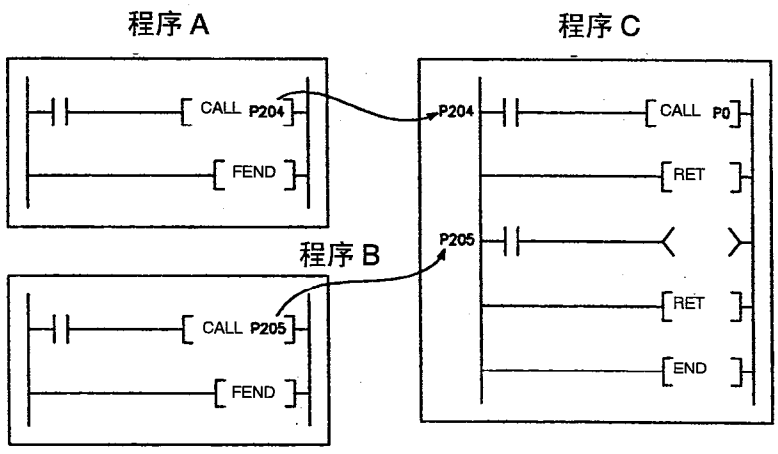
举例



4.9.2 通用指针

(1) 定义

（a）通用指针用于从所有的在 QnACPU 中执行的程序中调用子程序



(b) 相同的指针编号不能再用作标号。这样的使用将导致一个指针配置错误（错误代码：4021）。

(2) 使用的通用指针范围

为了使用通用指针，必须在 PC 系统的设置参数中指定第一个通用指针编号。

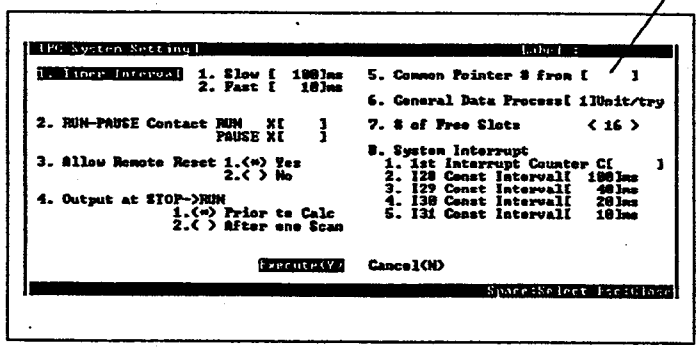
在此编号之后的所有指针都将变成通用指针。然而，只有在局部指针范围之后的指针编号才能被指定（通过参数设置）

程序 A	程序 B	程序 C
P0-P99 用于程序中	P0-P99 用于程序中	P0-P199 用于程序中
P0-P99 占 100 点	P0-P99 占 100 点	P0-P199 占 200 点

总共使用了 400 点。

在 P400 后面的所有指针可以被

[PC 系统设置窗口]



在这里设置通用指针的开头编号。

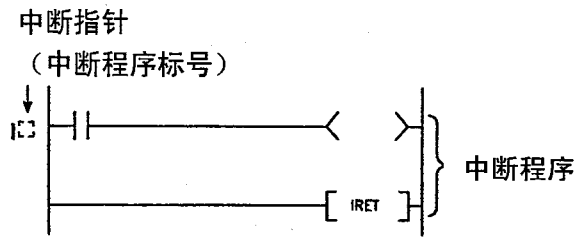
要点

(1) 在跳转命令中，不允许跳转到其他程序的通用指针处。通用指针应该只和子程序调用命令一起使用。

4.10 中断指针

(1) 定义

(a) 中断指针被用作中断程序开头处的标号。



(b) 所有执行程序中可使用共 48 个中断指针（10-147）。

(2) 中断指针编号和中断因素

(a) 如下所示，有四种类型的中断因素。

- AI61 因素....来自 AI61 中断模块的中断输入。
- 顺控起动发生器模块因素

  ... 来自特殊功能模块，例如计算机连接模块等的中断输入，该特殊功能模块可以为 QnACPU（不包括 AI61）指定一个中断开始。
- 内部时间因素

  ... 由 QnACPU 的内部定时器产生的固定周期的中断。
- 错误中断

  .... 由继续顺控程序运行的错误产生的中断。

(b) 表 4.5 所示是中断指针编号和中断因素的列表。

表 4.5 中断指针编号和中断因素的列表

编号	中断因素		优先级别	中断编号	中断因素		优先级别
AI61 中断模块因素		第 1 点	29	I16	顺控起动发生器模块因素*1	第 1 块	17
I1		第 2 点	30	I17		第 2 块	18
I2		第 3 点	31	I18		第 3 块	19
I3		第 4 点	32	I19		第 4 块	20
I4		第 5 点	33	I20		第 5 块	21
I5		第 6 点	34	I21		第 6 块	22
I6		第 7 点	35	I22		第 7 块	23
I7		第 8 点	36	I23		第 8 块	24
I8		第 9 点	37	I24		第 9 块	25
I9		第 10 点	38	I25		第 10 块	26
I10		第 11 点	39	I26		第 11 块	27
I11		第 12 点	40	I27		第 12 块	28
I12		第 13 点	41	I28	内部定时器因素*2	100ms	48
I13		第 14 点	42	I29		40ms	47
I14		第 15 点	43	I30		20ms	46
I15		第 16 点	44	I31		10ms	45

中断编号	中断因素		优先级别
I32	错误因素 *3*4	停止操作的错误	1
I33		空余	——
I34		单位检验错误。保险丝断开 单位错误	2
I35		操作错误 SFCP 操作错误	3
I36		ICM.操作错误 文件操作错误	4
37		扩展 INS.错误	5
I38		程序时间结束	6
I39		CHK 指令执行报警器检测	7
I40-I46		空闲	——
I47		作为 EDRSET 指令的标号使用	

注释

1）*1：第 1 到第 12 块是按顺序分配的，是从安装在最接近 QnACPU 的顺控起动发生器模块开始的。

2）*2：所示的内部时间是缺省的设置时间。
这些时间可以在从 5ms 到 1000ms 的范围内（PC 系统设置参数）以 5ms 为单位进行指定。

3）*3：当带有“132（停止运行的错误）”的错误中断发生时，直到 132 处理完成，QnACPU 才会停止运行，。

4）*4：当电源被打开时以及在 QnACPU 的复位期间，禁止错误中断的执行。当使用了中断指针号 132 到 139 时，使用 IMASK 指令来设置中断许可状态。

4.11 他软元件

4.11.1 SFC 块软元件（BL）

该软元件用于检查被 SFC 程序指定的块是否激活。

关于使用 SFC 块软元件的详细资料，请参考 QnACPU 编程手册(SFC)。

4.11.2 SFC 变换软元件(TR)

该软元件用于检查一个强制的变换是否在一个指定的 SFC 程序块中被指定了指定的变换条件。

关于使用 SFC 变换软元件的详细资料，请参考 QnACPU 编程手册（SFC）。

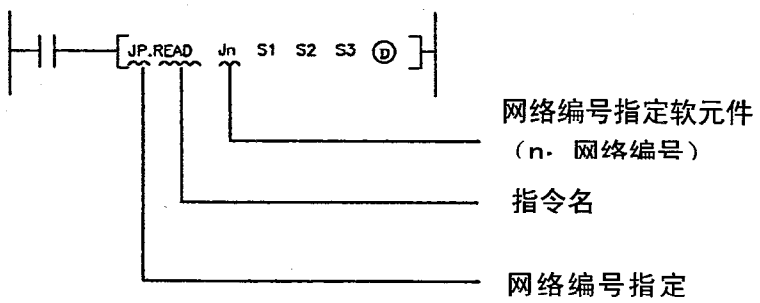
4.11.3 网络编号指定软元件(J)

（1）定义

网络编号指定软元件被用于指定在数据连接指令中的网络编号。

（2）指定网络编号指定软元件

网络编号指定软元件在数据连接指令中被指定，如下所示。



注释

1) 关于数据连接指令的详细资料，请参考 QnACPU 编程手册(通用指令)。

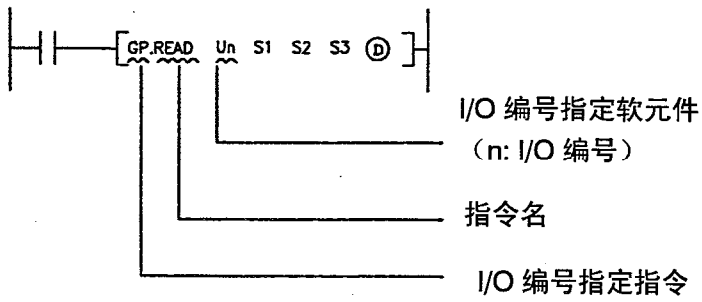
4.11.4 I/O 编号指定软元件(U)

（1） 定义

I/O 编号指定软元件与数据连接指令或特殊功能模块指令一起使用以指定 I/O 编号。

（2） 指定 I/O 编号指定软元件

I/O 编号指定软元件与数据连接指令或特殊功能模块指令一起指定，如下所示。



注释

1) 关于特殊功能模块指令的详细资料，请参考 QnACPU 编程手册（特殊功能模块）。

4.11.5 宏指令参数软元件(VD)

（1）定义

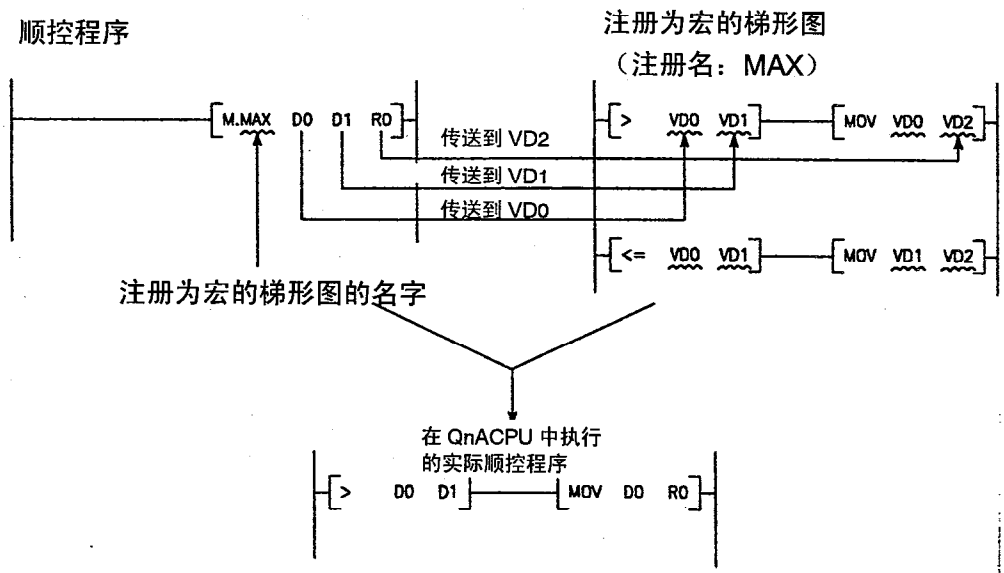
宏指令参数软元件用于注册为宏的梯形图。

当 VD[]设置为注册为宏的梯形图中的指定时，在宏指令被执行时，就会被转换成指定的软元件。

（2）指定宏指令参数软元件

宏指令参数软元件用于指定那些在外围设备的宏注册中注册为宏指令的梯形图中设置的软元件。*

当在一个顺控程序中使用了宏指令时，按升序将软元件指定为对应于与注册为宏的梯形图一起使用的指令参数软元件。



注释

1）*：通过宏指令参数软元件，VD0 到 VD4 可以被用在一个注册为宏指令的梯形图中。

4.12 常量

4.12.1 十进制常量（K）

（1） 定义

十进制常量是在顺控程序中指定十进制数据的软元件。

它们被指定为“K[]”设置（例如，K1234），并且以二进制（BIN）码形式被保存在 QnACPU 中。

关于二进制码的详细资料请参见 3.4.1 节。

（2） 指定范围

十进制常量的设置范围如下所示：

- 对于字数据（16 位）...K-32768 到 K32767
- 对于两个字的数据(32 位)...K-2147483648 到 K2147483647

4.12.2 十六进制常量（H）

（1） 定义

十六进制常量是在顺控程序中指定十六进制或 BCD 数据的软元件。（对于 BCD 数据指定，使用了 0-9 数字指定。）

十六进制常量被指定为“H[]”设置（例如，H1234），

关于十六进制码的详细资料请参见 3.4.3 节。

（2） 指定范围

十六进制常量的设置范围如下所示：

- 对于字数据（16 位）...H0 到 HFFFF（对于 BCD， H0 到 H9999）
- 对于两个字数据（32 位）...H0 到 HFFFFFFFF（对于 BCD，H0 到 H99999999）

4.12.3 实数（E）

（1） 定义

实数是在顺控程序中指定实数的软元件。

实数被指定为“E[]”设置（例如，E1.234），

关于实数的详细资料请参见 3.4.4 节。

（2） 指定范围

实数的设置范围是-1.0×2^{127}到-1.0×2^{-126}，0，1.0×2^{-126}到1.0×2^{127}。

（3） 指定方法

实数可以在顺控程序中通过一个“正常表达式”或一个“指数表达式”来指定。

正常表达式……规定值被指定为 F 或例如，10.2345 变成 E10.2345。

指数表达式……将指定值与一个“$\times10^{n}$”的指数相乘。例如，1234 变成 E1.234+3。*

注释

1）*：在上面例子中的“+3”代表一个10^{n}值（10^{3}）。

4.12.4 字符串（”）

（1） 定义

字符串常量是用于在顺控程序中指定字符串的软元件。

它们被引号指定（例如，“ABCD1234”）。

（2） 可以使用的字符串

所有的 ASCII 码字符都可以被用于字符串。

（3） 指定的字符串数目

字符串从指定的字符到 NUL 码（00H）。

4.13 软元件的方便使用

当在 QnACPU 中执行了多个程序时，在内部用户软元件中的局部软元件可以被用在每一个程序的独立执行中。

另外，软元件的初始化设置可以被用来指定软元件和特殊功能模块数据设置，而不需要使用程序。

4.13.1 全局软元件以及局部软元件

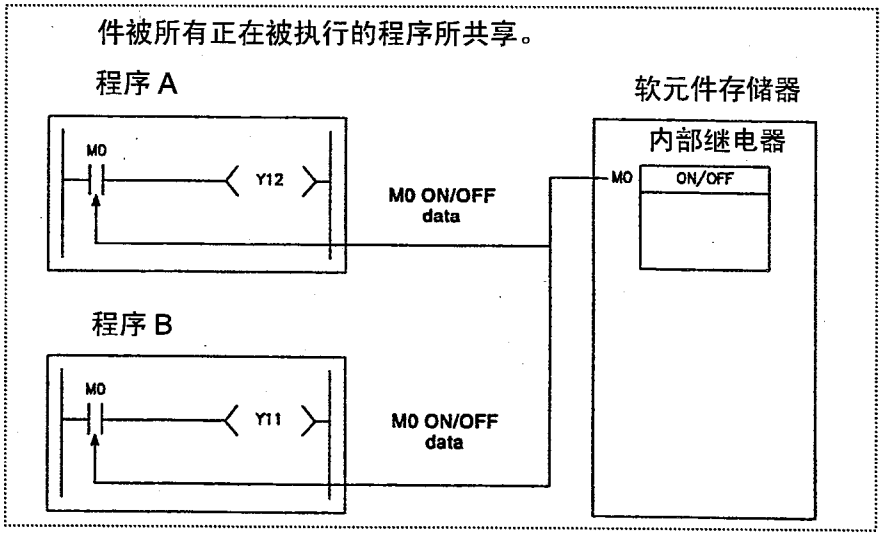
在 QnACPU 中，可以执行并保存许多程序。

在 QnACPU 软元件中，那些可以被所有的程序共享的软元件是“全局软元件”，而那些被每一个程序独立使用的软元件是“局部软元件”。

（1） 全局软元件

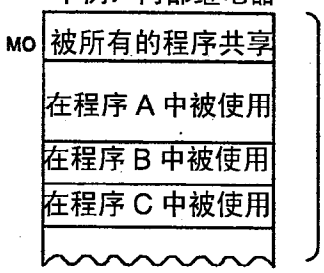
（a） 全局软元件是能够被在 QnACPU 中正在执行的所有程序共享的软元件。全局软元件被保存在 QnACPU 的软元件存储器中，所有的程序使用相同的软元件。

全局软元



(b) 在执行多个程序时，必须事先指定所有程序的“共享范围”以及每一个程序的“独立范围”。

举例：内部继电器



必须为每一个程序指定“使用范围”。

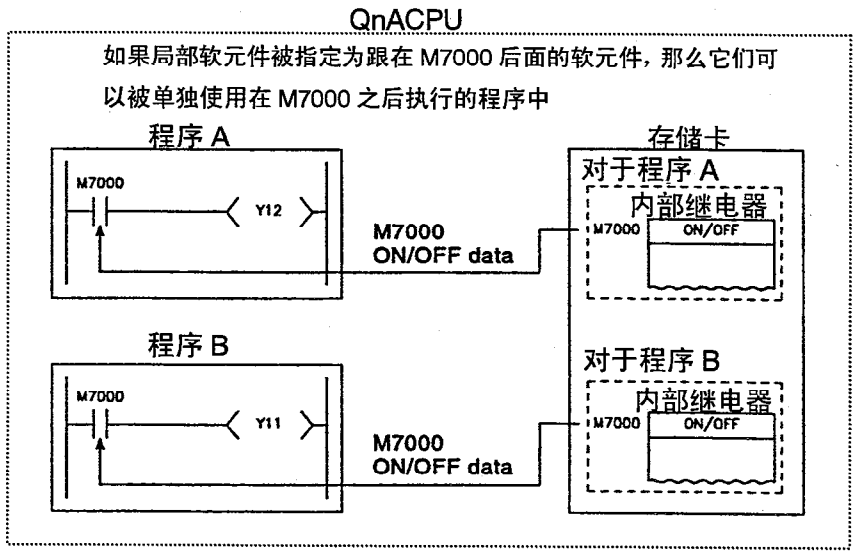
（2）局部软元件

（a）局部软元件是被各个程序独立使用的软元件。

使用局部软元件时不需要考虑其他的程序，只需考虑对应的程序编程。

然而，因为局部软元件被存储在存储卡中，所以为了要使用局部软元件就需要有一块存储卡。

没有存储卡，局部软元件不能使用。



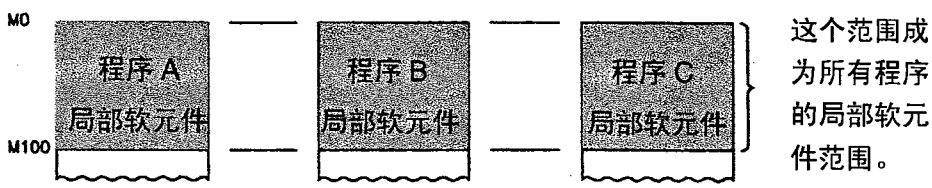
（b）有五种软元件类型可以被用作局部软元件：内部继电器（M）、边缘继电器（V）、定时器（T，ST）、计数器（C）以及数据寄存器（D）。

（c）局部软元件的指定

1）为了将以上软元件用作局部软元件，就必须在软元件设置参数中指定一个局部软元件的使用范围。

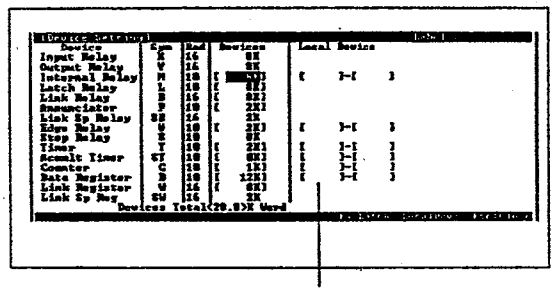
请注意，为局部软元件指定的范围适用于所有的程序，并且不能特别设定为某一个程序的局部软元件。

例如，如果局部软元件范围被指定为 M0 到 M100，那么这个范围将应用于所有程序的局部软元件。



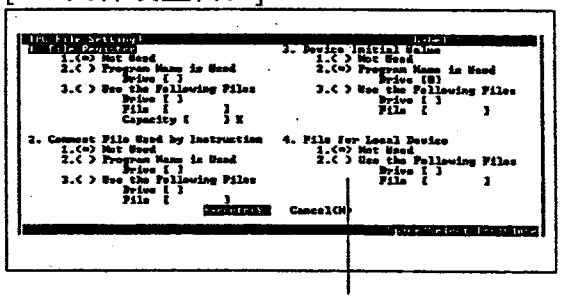
2） 当局部软元件设置被指定时，局部软元件数据要被保存的驱动器和文件名必须在参数模式的 PC 文件设置中指定。

[驱动器范围设置]



局部软元件范围设置区域

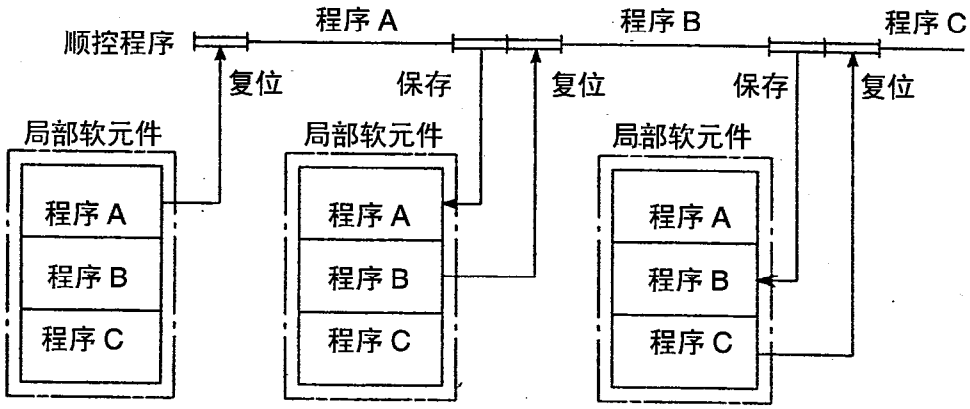
[PC 文件设置窗口]



局部软元件文件设置区域

（d）当使用了局部软元件时，在保存在存储卡中的局部软元件文件数据和 QnACPU 的软元件区域的数据之间会进行数据交换。因此扫描时间被这个数据交换时间延长了。

- Q02ACPU（S1） 560+1.3×（局部软元件的字数）*
- Q02ASCPU（S1） ×程序数目（us）
- Q03ACPU 320+1.0×（局部软元件的字数）*
  ×程序数目（us）
- Q04ACPU 220+0.8×（局部软元件的字数）*
- Q04ARCPU ×程序数目（us）
- Q02ASHCPU(S1)



（e）当 CPU 从 STOP 切换到 RUN 时，局部软元件中的数据都被清除。

注释

1) *：关于局部软元件的“字数”的详细资料，请参见 4.1.2 节（项目 2）。

要点

（1） 关于在中断程序中使用局部软元件，请参见 3.1.3 条。

（2） 除非特别指定为“局部软元件”，否则所有的软元件都是全局软元件。

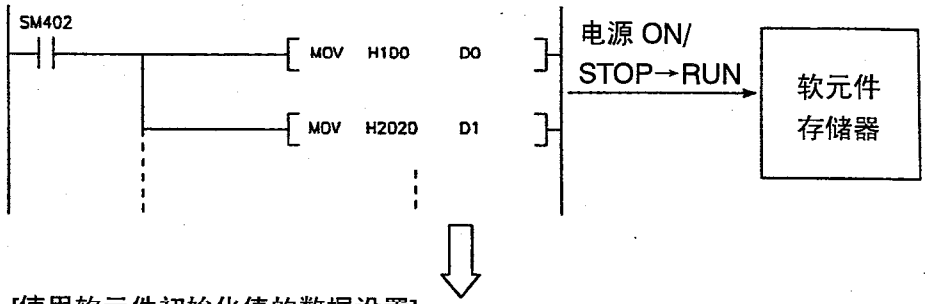
（3） 关于在子程序中局部软元件的使用，请参见 3.1.2 条。

4.13.2 软元件初始化值

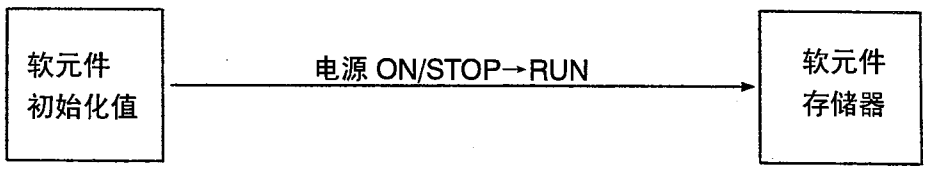
（1） 定义

（a）使用软元件初始化值，用于程序的数据可以不需用命令设置被保存到软元件或特殊功能模块的缓冲存储器中。因此可以省去初始化处理程序。

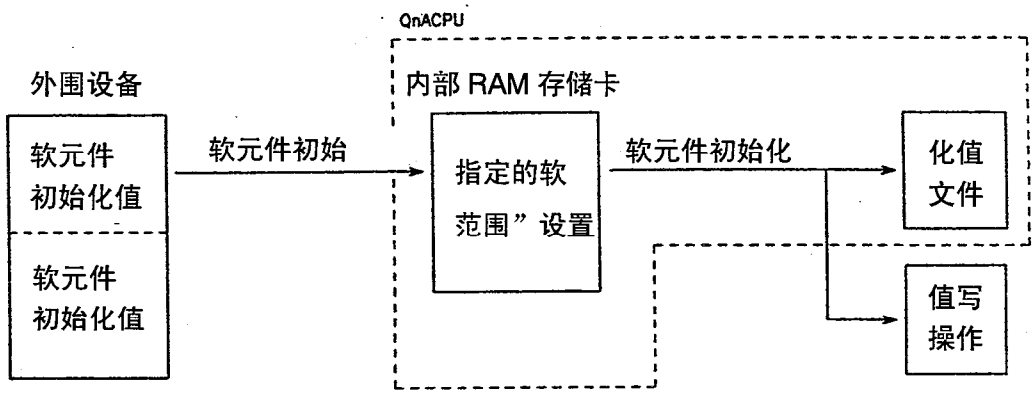
[使用初始化程序的数据设置]



[使用软元件初始化值的数据设置]



（b）为了使用软元件初始化值，就必须在外围设备事先创建软元件初始化数据，并且该数据必须被保存为 QnACPU 存储卡中的一个软元件初始化值文件。在电源 ON 时，或从 STOP 切换到 RUN 时，QnACPU 将数据从软元件初始化文件写到指定的软元件或特殊功能模块的缓冲存储器中。



(c) 软元件初始化值可以被用于以下软元件中：

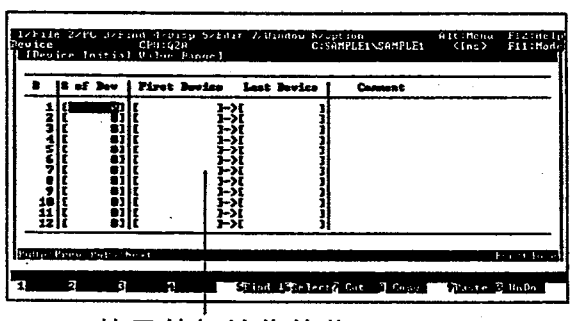
1）定时器当前值（T）
2）保持定时器当前值（ST）
3）计数器当前值（C）
4）数据寄存器（D）
5）特殊寄存器（SD）
6）连接寄存器（W）
7）特殊连接寄存器（SW）
8）文件寄存器（R0-R32767）
9）特殊功能模块软元件（U□\G□）
10）连接直接软元件（J□\W□,J□\SW□）

（2） 使用软元件初始化值的过程

（a）在软元件初始化值设置屏幕的软元件模式中，指定软元件初始化值范围的设置。

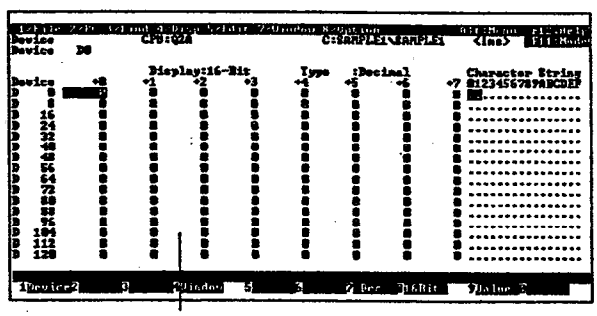
（b）在软元件模式屏幕指定软元件初始化值数据。

[软元件初始化设置屏幕]



软元件初始化值范围

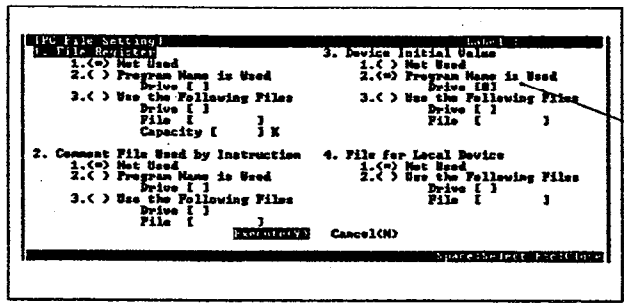
[软元件模式]



软元件初始化值数据

（c）在参数模式下的 PC 文件设置中，指定软元件初始化值数据要被保存到的文件的文件名。

[PC 文件设置窗口]



(d) 将软元件初始化值数据和参数设置写入 QnACPU 中。

（3）使用软元件初始化值的注意事项

（a）在软元件初始化值数据和锁存范围数据都存在的情况下，软元件初始化值数据具有优先权。因此，在电源打开时，锁存范围数据被软元件初始化值数据覆盖。

（b）软元件初始化值不能用于下列区域。

1）在一个特殊功能模块的缓冲存储器区域中的需要通道切换的区域。

例如：AJ71E71 以太网接口模块的缓冲存储器中的通道 2 区域。

2）在写序列固定的特殊功能模块区域。

例如：A68AD 模拟/数字转换器模块的初始化设置。

3）在 STOP→RUN 切换时不希望有设置的区域（在电源打开时被复位，并且被程序和修改的数据）。

注释

1）关于“软元件初始化值范围”和“软元件初始化值数据”的设置过程的详细资料，请参考 SW[]IVD-GPPQ GPP 功能软件包操作手册（在线）

2）关于将软元件初始化值写到 QnACPU 中的过程的详细资料，请参考 SW[]IVD-GPPQ GPP 功能软件包操作手册（在线）。

备忘录

QnA编程

参考手册

型号	QnA-P(KISO)-CH
	SH(NA)-080229C-A

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : 1-8-12, OFFICE TOWER Z 14F HARUMI CHUO-KU 104-6212,TELEX : J24532 CABLE MELCO TOKYO
NAGOYA WORKS : 1-14 , YADA-MINAMI 5 , HIGASHI-KU, NAGOYA , JAPAN