第5章 三菱FX2N系列 PLC的功能指令

本章导读

– 本章主要介绍了FX2N的功能指令及其编程方法，功能指令编号为FNC00～FNC246，将常用功能指令归类讲述，从5.6节起只作简讲，未介绍的可查阅附录表B.2。将以表格形式归纳功能指令格式、类型及使用要素。选择合适的功能指令，将使编程更加方便和快捷。要求掌握各类功能指令及其编程方法，掌握GPPW内装的的模拟仿真、时序图等功能，来帮助学习功能指令。

5.1 功能指令的基本规则 1

FX2N系列PLC的功能指令一览表见附录表B.2。一条基本逻辑指令只完成一个特定的操作，而一条功能指令却能完成一系列的操作，相当于执行了一个子程序，所以功能指令功能更强大，编程更精练，它能用于运动控制、模拟量控制等场合。基本指令和其梯形图符号之间是互相对应的。而功能指令采用梯形图和助记符相结合的形式，意在表达本指令要做什么。有些功能指令在整个程序中只能使用一次，介绍到此类指令时会特别强调。

5.1.1 功能指令的表示

1. 功能指令的梯形图表示

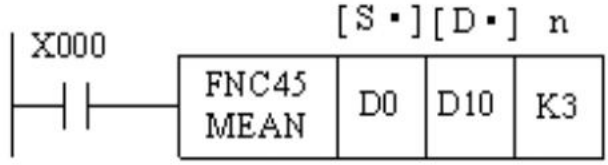
用功能框表示功能指令，即在功能框中用通用的助记

5.1.1 功能指令的表示 2

符形式来表示，如图5.1（a）所示，该指令的含义如图5.1（b）所示。图5.1（a）中X000常开接点是功能指令的执行条件，其后的方框即为功能指令。由图可见，功能指令同一般的汇编指令相似，也是由操作码和操作数两大部分组成。

（1）操作码部分

功能框第一段为操作码部分，表达了该指令做什么。



（a）功能指令 MEAN 梯形图

$$\frac{(D0)+(D1)+(D2)}{3} \rightarrow D10$$

（b） X000 ON，MEAN 指令含义

图 5.1 功能指令 MEAN 举例

5.1.1 功能指令的表示 3

一般功能指令都是以指定的功能号来表示，如FNC45。但是，为了便于记忆，每个功指令都有一个助记符，对应FNC45的助记符是MEAN，表示“求平均值”。这样就能见名知义，比较直观。在编程器或FXGP软件中输入功能指令时，输入的是功能号FNC45，显示的却是助记符MEAN。不过，在FXGP软件中也可直接输入助记符MEAN。

注意：本书在介绍各功能指令时，将以图5.1（a）的形式同时给出功能号和对应的助记符，但并不意味着在FXGP软件中输入功能指令时要两者一起送，而是按上述介绍，只要送入其中一个就行了。

（2）操作数部分

5.1.1 功能指令的表示 4

功能框的第一段之后都为操作数部分，表达了参加指令操作的操作数在那里。操作数部分部分组成：

源操作数（源）　目标操作数（目）　数据个数

源操作数：D0、D1和D2，数据个数K3指示源有3个；

目操作数：D10。

当X000接通时，MEAN指令的含义如图5.1（b）所示，即要取出D0～D2的连续3个数据寄存器中的内容作算术平均后送入D10寄存器中。当X000断开时，此指令不执行。

操作数排列次序：源在前，目在后，数据个数在最后

有些功能指令还要求多个操作数，也有的功能指令不需要操作数。

5.1.1 功能指令的表示 5

2. 功能指令的要素描述

功能指令的要素描述将按表图的格式给出。如对图5.1（a）这条MEAN指令的要素描述如表5.1所示。表中使用符号的说明：

① 求平均值指令：指令的名称

② FNC45：指令的功能号

表 5.1 MEAN 指令概要

求平均值指令		操作数	程序步
P	FNC45	[S·]	MEAN, MEAN(P) 7 步
	MEAN	K,H / KnX / KnY / KnM / KnS / T / C / D / V, Z	
16	MEAN(P)	n [D·]	

5.1.1 功能指令的表示 6

③ MEAN　指令的助记符

④（P）　指令的执行形式，（P）表示可使用脉冲执行方式，在执行条件满足时仅执行一个扫描周期；缺省的为连续执行型。

⑤（D）　指令的数据长度可为32位，缺省为16位。

⑥ [S·]　源操作数，简称源，指令执行后不改变其内容的操作数。当源不止一个时，用[S1·]、[S2·]等来表示。有“·”表示能用变址方式，缺省为无“·”，表示不能使用变址方式。

⑦ [D·]　目标操作数，简称目，指令执行后将改变其内容的操作数。当目不止一个时，用[D1·]、[D2·]等来表示。有“·”表示能使用变址方式，缺省为无“·”，表

5.1.1 功能指令的表示 7

示不能使用变址方式。

⑧ m、n　其它操作数，常用来表示常数或对源和目作出补充说明。表示常数时，K后跟的为十进制数，H后跟的为十六进制数。

⑨ 程序步　指令执行所需的步数。一般来说，功能指令的功能号和助记符占一步，每个操作数占2～4步(16位操作数是2步，32位操作数是4步)。因此，一般16位指令为7步，32位指令为13步。

5.1.2 功能指令的数据长度 1

1. 字元件与双字元件

（1）字元件

1个字元件是由16位的存储单元构成，最高位(第15位)

5.1.2 功能指令的数据长度 2

为符号位，第0～14位为数值位。图5.2所示为16位数据寄存器D0图示。

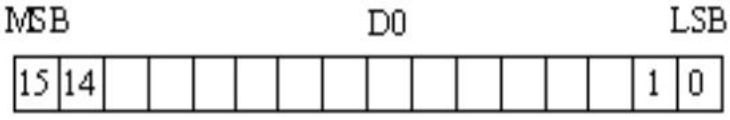


图 5.2 字元件

（2）双字元件

可以使用两个字元件组成双字元件，以组成32位数据操作数。双字元件是由相邻的寄存器组成，在图5.3中由D11和D10组成。低16位数据存放在低位元件D10中，

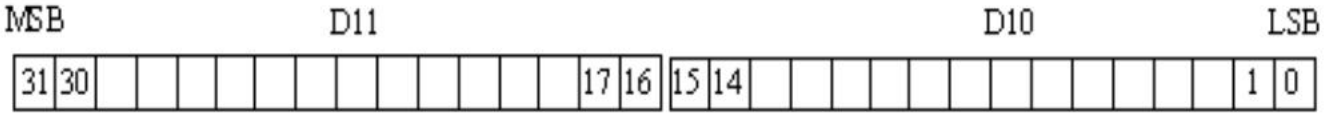


图 5.3 双字元件

5.1.2 功能指令的数据长度 3

高16位数据存放在高位元件D10中，存放原则是：低对低，高对高。双字元件中第31位为符号位，第0～30位为数值位。

注意：在指令中使用双字元件时，一般只用其低位地址表示这个元件，但高位元件也将同时被指令使用。建议用偶数作为双字元件的地址，此点会用图5.6来说明。

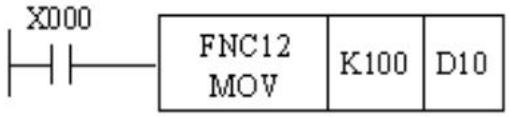
功能指令中的操作数是指操作数本身或操作数的地址。功能指令能够处理16位或32位的数据。

2. 功能指令中的16位数据

因为几乎所有寄存器的二进制位数都是16位，所以功能指令中16位的数据都是以缺省形式给出。如图5.4 所示即为一条16位MOV指令：

5.1.2 功能指令的数据长度 4

MOV指令的含义是，当X000接通时，将十进制数100传送到16位的数据寄存器D10中去。当X000断开时，该指令被跳过不执行，源和目的内容都不变。

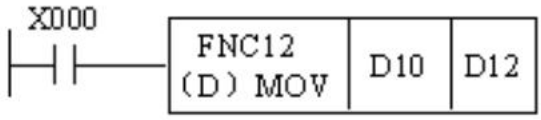


X000 ON，100 → D10

图 5.4 16位 MOV 指令

3. 功能指令中的32位数据

功能指令也能处理32位数据，这时需要在指令前缀符号（D），如图5.5 所示即为一条32位MOV指令：



X000 ON，(D11)→D13，(D10)→(D12)

图 5.5 32位 MOV 指令

5.1.2 功能指令的数据长度 5

凡是能前缀显式符号（D）的功能指令，就能处理32位数据。32位数据是由两个相邻寄存器构成的，但在指令中写出的是低位地址，源和目都是这样表达的。所以对图5.5所示32位MOV指令含义应该这样来理解：当X000接通时，将由D11和D10组成的32位源数据传送到由D13和D12组成的目标地址中去。

要避免出现类似图5.6所示指令的错误：源由D11和D10组成，而目由D12和D11组成，这里D11是源、目重复使用，就会引起出错。所以建议32位数据首地址用偶地址。

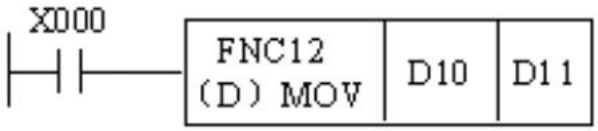


图 5.6 错误的 32 位 MOV 指令

5.1.2 功能指令的数据长度 6

注意:32位计数器C200～C255不能作为16位指令操作数

4. 功能指令中的位元件

位元件:只有ON或OFF两种状态，用一个二进制位就能表达的元件。如X、Y、M、S等。功能指令中也能使用由只含一个bit的位元件，以及位元件组合。

位元件组合成位组合元件的方法：将多个位元件按四位一组的原则来组合，也就是说用4位BCD码来表示1位十进制数，这样就能在程序中使用十进制数据了。组合方法的助记符是：

Kn＋最低位位元件号

如KnX、KnY、KnM即是位元件组合，其中“K”表示后面跟的是十进制数，“n”表示四位一组的组数，

5.1.2 功能指令的数据长度 7

16位数据：K1～K4，
32位数据：K1～K8。
数据中的最高位是符号位。如：
K2M0：由M0～M3和M4～M7两组位元件组成一个8位数据，其中M7是最高位，M0是最低位。
K4M10：由M10～M25四组位元件组成一个16位数据，其中M25是最高位，M10是最低位。

注意：

① 当一个16位数据传送到目元件K1M0～K3M0时，由于目标元件不到16位，所以将只传送16位数据中的低位数据，高位数据将不传送。32位数据传送也一样。

② 由于数据只能是16位或32位这两种格式，因此当用

5.1.2 功能指令的数据长度 8

K1～K3组成字时，其高位不足16位部分均作0处理。如执行图5.7所示指令时，源数据只有12位，而目标寄存器D20是16位的，传送结果D20的高4位自动添0，如图5.8所示。这时最高位的符号位必然是0，也就是说，只能是正数（符号位的判别是：正0负1）。

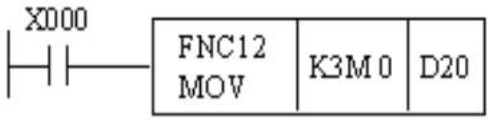


图 5.7 源数据不足 16 位

图 5.8 目高 4 位自动添 0

5.1.2 功能指令的数据长度 9

③ 由位元件组成组合位元件时，最低位元件号可以任意给定。如X000、X001和Y005均可。但习惯上采用以0结尾的位元件，如X000、X010和Y020等。

5.1.3 功能指令的执行方式

功能指令的两种执行方式：连续执行和脉冲执行方式。

1. 功能指令的连续执行方式

缺省为连续执行方式，如图5.9。PLC是以循环扫描方式工作的，如果执行条件X000接通，指令在每个扫描周期中都要被重复执行一次，这种情况对大多数指令都是允许的。

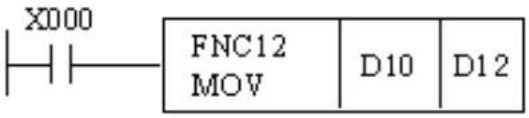


图 5.9 连续执行的 MOV 指令

5.1.3 功能指令的执行方式 2

2．功能指令的脉冲执行方式

对于某些功能指令，如XCH、INC和DEC等，用连续执行方式在实用中可能会带来问题。如图5.10所示是一条INC指令，是对目标元件（D10、D11）进行加1操作的。假设该指令以连续方式工作的话，那么只要X000是接通的，则每个扫描周期都会对目标元件加1，而这在许多实际的控制中是不允许的。为了解决这类问题，设置了指令的脉冲执行方式，并在指令助记符的后面后缀符号“P”来表示此方式，如图5.10所示。

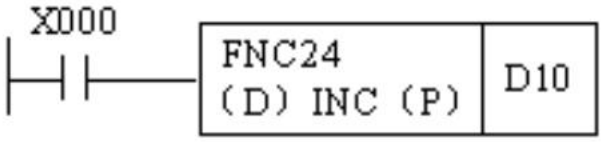


图 5.10 脉冲执行方式的 INC 指令

5.1.3 功能指令的执行方式 3

注意：在图5.10中INC后加“（P）”，仅表示指令还有脉冲执行方式；在INC前加“（D）”，也仅表示指令还有32位操作方式。但在FXGP中输入时应该这么送：DINCP　D10，即加在前后缀的括号不必送的 。对于在本书中，以这种方式表达的所有其它功能指令都要这样来理解。

在脉冲执行方式下，指令INC只在条件X000从断开变为接通时才执行一次对目标元件的加1操作。即每当X000来了一个上升沿，才会执行加1；而在其它情况下，即使X000始终是接通的，都不会执行加1指令。

在不需要每个扫描周期都执行指令时，可以采用脉冲执行方式的指令，这样还能缩短程序的执行时间。

5.1.4 变址操作 1

FX2N的16个变址寄存器V和Z都是16位的（FX0N和FX0S只有两个变址寄存器V和Z），即V0～V7、Z0～Z7。除了能作为通用数据寄存器之外，主要用于运算操作数地址的修改，在传送、比较等指令中用来改变操作对象的元件地址，循环程序中也常使用变址寄存器。变址方法是将V、Z放在各种寄存器的后面，充当操作数地址的偏移量。操作数的实际地址就是寄存器的当前值和V或Z内容的相加后的和。

当源或目的寄存器用[S•]或[D•]表示时，就能进行作变址操作。当进行32位数据操作时，V、Z自动组对成32位（V，Z）来使用，这时Z为低16位，而V充当高16位。可以用变址寄存器进行变址的软元件是X、Y、M、S、P、T、C、D、K、H、KnX、KnY、KnM、KnS。

5.1.4 变址操作 2

例5.1 如图5.11所示的梯形图中，求执行加法操作后源和目操作数的实际地址。

解： 第一行指令执行10→V，第二行指令执行20→Z，所以变址寄存器的值为，V＝10，Z＝20。第三行指令执行（D5V）＋（D15Z）→（D40Z），

[S1·]为D5V：	D(5+10)＝D15	源操作数1的实际地址
[S2·]为D15Z：	D(15+20)＝D35	源操作数2的实际地址
[D·]为D40Z：	D(40+20)＝D60	目操作数的实际地

所以，第三行指令实际执行（D15）＋（D35）→（D60），即D15的内容和D35的内容相加，结果送入D60中去。

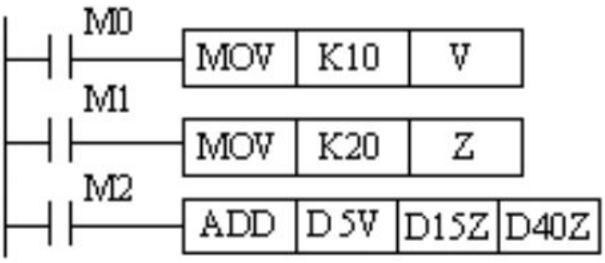


图 5.11 变址操作举例

5.2 程序流向控制指令 1

FX2N系列PLC的功能指令中程序流向控制指令共有10条，功能号是FNC00～FNC09，程序流向控制指令汇总如附录表B.2所示。通常情况下，PLC的控制程序是顺序逐条执行的，但是在许多场合下却要求按照控制要求改变程序的流向。这些场合有：条件跳转、转子与返回、中断调用与返回、循环、警戒时钟与主程序结束。

5.2.1 条件跳转指令

1. 指令用法说明

条件跳转指令为CJ或CJ（P）后跟标号，其用法是当跳转条件成立时跳过一段指令，跳转至指令中所标明的标号处继续执行，若条件不成立则继续顺序执行。这样可以减少扫描时间并使“双线圈操作”成为可能。

5.2 程序流向控制指令 2

条件跳转指令的助记符、功能号、操作数和程序步等指令概要如表5.2所示。由表5.2可见，能够充当目标操作数的只有标号P0～P127。

例5.2 梯形图如图5.12所示，阅读此程序，试分析：

（1）程序的可能流向；

（2）程序中的“双线圈操作”是否可能。

解： （1）分析图5.12所示程序的流向如下。

①若M0接通，则CJ P0的跳转条件成立，程序将跳转到标号为P0处。因为M0常闭是断开的，所以 CJ P1的跳转条件不成立，

表 5.2 CJ 指令概要

条件跳转指令		操作数	程序步
P	FNC00 CJ CJ(P)	[D·]：P0、P1……P127	CJ、CJ(P) 3步
16		P63即END	标号P 1步

5.2 程序流向控制指令 3

程序顺序执行。按照M3的状态对Y000进行处理。

②若M0断开，则CJ　P0的跳转条件不成立，程序会按照指令的顺序执行下去。执行到P0标号处时，由于M0常闭是接通的，则CJ　P1的跳转条件成立，因此程序就会跳转到P1标号处。

（2）Y000为双线圈输出。

程序在执行过程中，M0常开和M0常闭是一对约束。使线圈Y000驱动逻辑任何时候只有一个会发生，所以在图5.12所示梯形图中Y000为双线圈输出是可以的。

2. 跳转程序中软元件状态与标号

（1）被跳过程序段中软元件的状态

被跳过的程序段中的各种继电器和状态器、定时器等将保持跳转发生前的状态不变。**正在工作的定时器T192～T199高速计数器C235~C255不管有无跳转仍将**

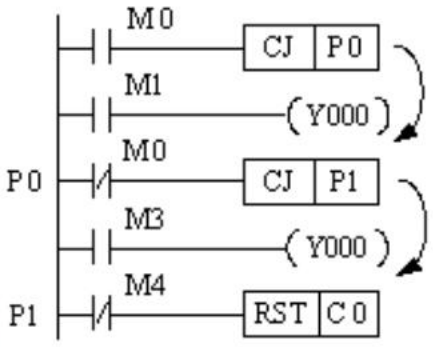


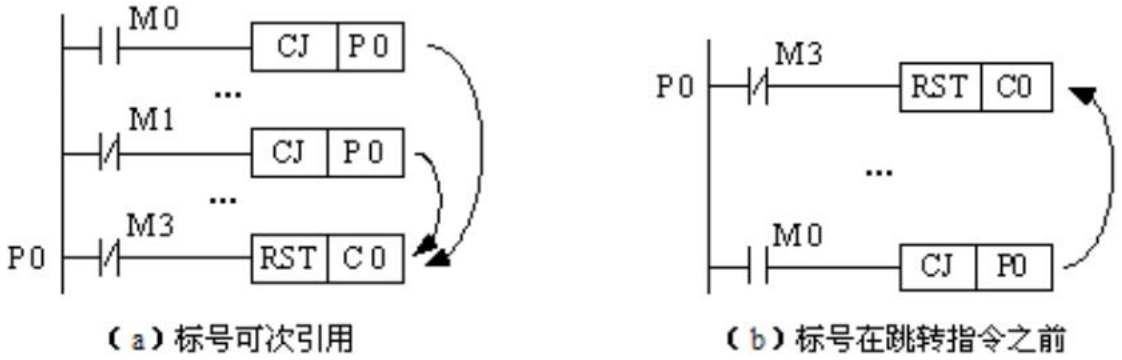
图 5.12 CJ 指令示例

5.2 程序流向控制指令 4

连续工作，输出接点也能动作。掉电保持计数器、定时器，其当前值被锁定。程序继续执行时，它们将继续工作。复位优先，即使复位指令在被跳过程序段中，条件满足，复位也将执行。

（2）标号不能重复使用，但能多次引用。

同一标号不能重复使用，但可多次被引用，即可从不同的地方跳转到同一标号处，如图5.13（a）。标号也可以出现在跳转指令之前，如图5.13（b）。



（a）标号可次引用　　（b）标号在跳转指令之前

图 5.13 跳转指令标号使用

5.2 程序流向控制指令 5

当M0接通时，程序也允许向回跳转。但是如果M0接通时间超过100ms，会引起警戒时钟出错，但不会影响程序的执行。

标号共有128个，其中P63相当于END，不能作为真正的标号使用。这样，当要跳过最后一段程序结束时，就可以在此段程序前设置一条CJ P63指令。也可以理解为CJ P63就是跳转到程序的最开始处。而且标号P63不必出现在程序中。

3. 无条件跳转与条件跳转的脉冲执行方式

（1）构造无条件跳转指令

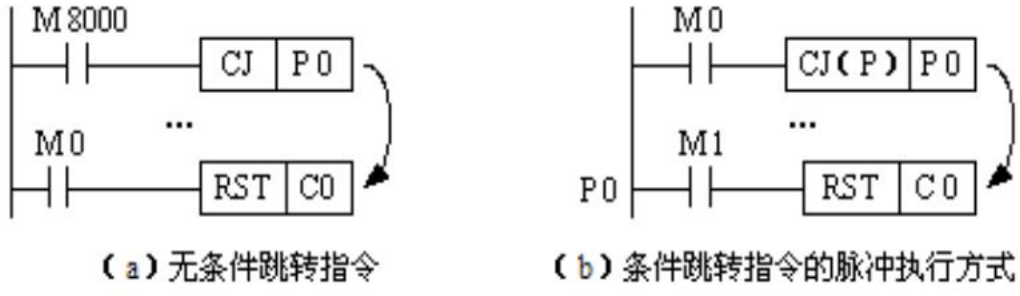
可用条件跳转指令来构造无条件跳转指令，使用某个始终成立的条件使条件跳转变成无条件跳转。常用的是

5.2 程序流向控制指令 6

M8000，因为只要PLC处于RUN状态，则M8000总是接通的，无条件跳转梯形图如图5.14（a），条件跳转指令CJ P0的驱动条件始终成立，因此就可以将这条指令看成是无条件跳转。

（2）条件跳转指令的脉冲执行方式

条件跳转指令脉冲执行方式如图5.14（b）。



（a）无条件跳转指令　　（b）条件跳转指令的脉冲执行方式

图 5.14 跳转指令的用法

5.2 程序流向控制指令 7

4. 跳转与主控区之间的相关问题

①如果跳转的区域包括整个主控区(MC—MCR)，则将不受任何限制，可以随意跳转而不必考虑主控区问题。

②如果跳转从主控区外跳到主控区内时，这时主控指令的目标接点，应被当作接通来处理。比如说被跳过的主控指令为MC N0 M10，则M10仍被看作是接通的。

③如果跳转发生在主控区内，当主控接点为断开时，跳转指令因没有执行到而不能跳转。

④如果跳转从主控区内跳到主控区外时，当主控接点断开时，由于没有执行到跳转指令，因此不能跳转。当主控接点接通时，可以跳转，这时MCR指令被忽略。

⑤如果跳转从一个主控区内跳到另一个主控区内，而且

5.2 程序流向控制指令 8

源主控接点是接通的，则跳转可以进行，不管目标主控接点原状态如何，均被看做接通，MCR N0被忽略。

5.2.2 转子与返回指令

子程序也是为一些特定的控制目的编制的相对独立的模块，供主程序调用。为了区别于主程序，将主程序排在前边，子程序排在后边，并以主程序结束指令FEND（FNC06）给以分隔。

1．指令用法说明

（1）子程序调用指令：CALL或CALL（P） 标号，标号是被调用子程序的入口地址，以P0～P127（不包括P63)来表示。

子程序返回用SRET指令。

5.2.2 转子与返回指令 2

转子与返回指令的指令概要如表5.4所示。子程序调用和返回的梯形图如图5.15（a）。当M0接通时，调用子程序P0，程序将跳转到P0标号所指向的那条程序，同时将调用指令下一条指令的地址作为断点保存。此后从P0开始逐条顺序执行子程序，直至遇到SRET指令时,程

表 5.4 转子与返回指令概要

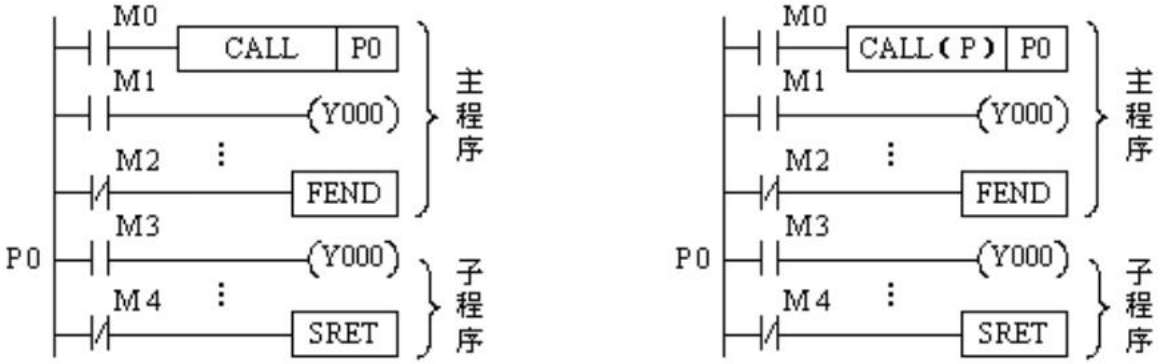
指令名称	助记符	操作数	程序步
转子	FNC 01 CALL CALL (P)	[D▪]：P0、P1……P62 嵌套 5 级	CALL , CALL (P) 3 步 标号 P 1 步
返回	FNC 02 SRET	无	SRET 1 步

5.2.2 转子与返回指令 3

序将返回到主程序的断点处，继续顺序执行主程序，即执行指令LD M1，OUT　Y000。

（2）子程序的位置与标号使用

子程序P0安排在主程序结束指令FEND之后，标号P0和子程序返回指令SRET间的程序构成了P0子程序的内容



（a） 子程序调用与返回例　　（b）CALL 指令的脉冲执行方式

图 5.15 CALL 指令的用法

5.2.2 转子与返回指令 4

主程序带有多个子程序时，子程序要依次放在主程序结束指令FEND之后，并以不同的标号相区别。**FX1N、FX2N和FX2NC**子程序标号范围为P0～P127（**FX1S为P0～P63**），这些标号与条件转移中的标号相同，在条件转移中已使用了的标号，子程序不能再用。同一标号只能用一次，不同的CALL指令可多次调用同一标号。

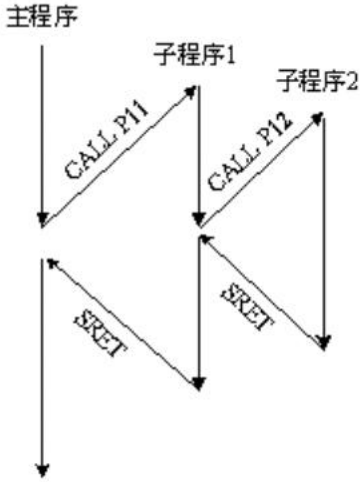
（3）CALL指令的脉冲执行方式

脉冲执行方式如图5.15（b），只有在M0上升沿时，程序才会转子。CJ是跳转，CALL也是一种跳转，不过CJ跳转是“有去无回”，而CALL的跳转则是“有去有回”的，子程序结束后将会回到主程序的断点处继续执行原来的程序。为了两者区别，把后者称谓“调用”，更适当。

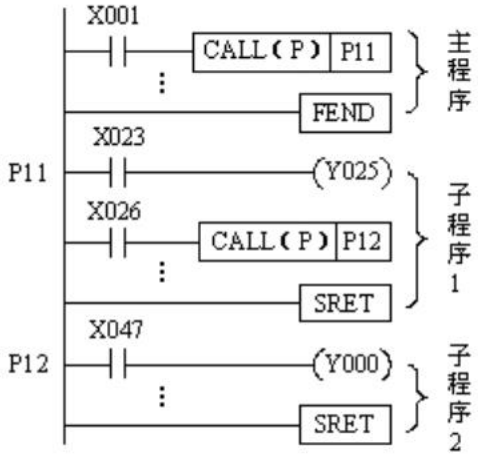
5.2.2 转子与返回指令 5

2. 子程序的嵌套

子程序嵌套示意如图5.16（a），梯形图如图5.16（b）
子程序嵌套：主程序调用子程序1，在执行子程序1时，子程序1又调用另一个子程序2，称为子程序嵌套。



（a）子程序嵌套示意



（b）子程序嵌套梯形图

图 5.16 子程序嵌套

5.2.2 转子与返回指令 6

子程序嵌套总数可有5级。

注意：在子程序和中断子程序中使用的定时器范围为：T192～T199和T246～T249。在此之外的定时器，虽然在子程序中或许也能使用，但不能保证其运行的正确性，所以请别使用在此之外的定时器。

5.2.3 中断与返回指令

中断是CPU与外设“打交道”(数据传送)的一种方式，慢速的外设远远跟不上高速CPU的节拍，要“拖累”CPU。为此可采用数据传送的中断方式，使CPU与外设并行工作的，平时CPU在执行主程序，当外设需要数据传送服务时，才去向CPU发出中断请求。在允许中断的情况下，CPU可以响应外设的中断请求，从主程序中被拉出来，去执行一段中断服务子程序，比如给了外设一批数据

5.2.3 中断与返回指令 2

后，就不再管外设，返回主程序。以后都是这样，每当外设需要数据传送服务时，又会向CPU发中断请求。可见CPU只有在执行中断服务子程序短暂的时间里才同外设打交道，所以CPU的工作效率就大大提高了。

1. 指令用法说明

有关中断指令概要如表5.5所示。

（1）中断返回指令：　　FNC 03　IRET

表 5.5 有关中断指令概要

指令名称	功能号　助记符	操作数	程序步
中断返回	FNC 03　IRET	[D·]：无	IRET　1步
开中断	FNC 04　EI	[D·]：无	EI　1步
关中断	FNC 05　DI	[D·]：无	DI　1步

5.2.3 中断与返回指令 3

（2）开中断指令（中断允许）：FNC 04 EI

（3）关中断指令（中断禁止）：FNC 05 DI

2.3.4节已经介绍了FX2N系列PLC有3类中断：

外部中断

内中断（即内部定时器中断）

高速计数器外部计数中断

FX2N系列PLC可以多达15个中断源，15个中断源可以同时向CPU发中断请求信号，这时CPU要通过中断判优，来决定响应那一个中断。15个中断源的优先级由中断号决定，中断号小者其优先级为高。另外，外中断的优先级整体上高于内中断的优先级。这样，在主程序的执行过程中，就可根据不同中断服务子程序中PLC所要完成工作的优先级高低决定能否响应中断。对可以响应中

5.2.3 中断与返回指令 4

断的程序段用中断允许指令EI来开中断，对不允许中断的程序段用中断指令DI来关中断。程序中允许中断响应的区间应该由EI指令开始，DI指令结束，如图5.17所示。在此区间之外时，即使有中断请求，CPU也不会立即响应。通常情况下，在执行某个中断服务程序时，将禁止其它中断。

中断返回指令必须用**IRET**，不能用子程序返回指令SRET。IRET指令除了能从中断服务程序返回以外，还要通知CPU本次中断已经结束。

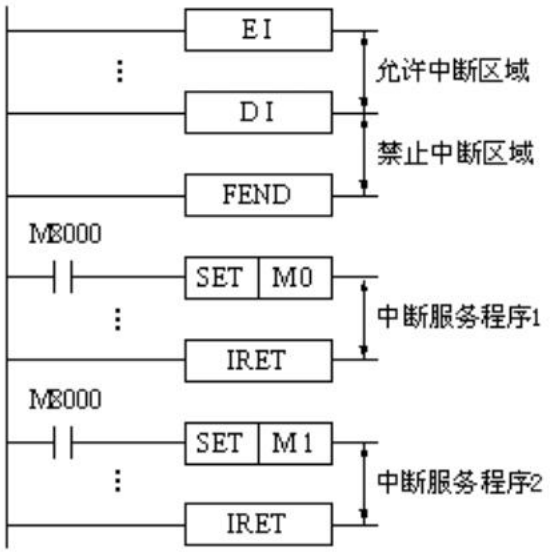


图 5.17 中断区间示意

5.2.3 中断与返回指令 5

中断与子程序区别：子程序调用是事先在程序中用CALL给定的，但中断调用要求响应时间小于机器的扫描周期，所以就不能事先在程序中给定，而是由外设（中断源）随机地通过硬件向CPU发中断请求，才把CPU拉到中断服务子程序中去的。整个中断是一个软硬件结合的过程。

2. 中断指针

为区别内外中断及标明中断子程序的入口，规定了中断标号。中断标号是I开头的，又称为I指针。子程序的标号是P开头的，又称为P指针。I指针分为3种。

（1）外中断指针

外中断指针的格式如图5.19（a），I00～150，共6点。外中断是外部信号引起的，对应的外部输入口为X000

5.2.3 中断与返回指令 6

～X005。指针格式中的最后一位是选择是上升沿请求中断，还是下降沿请求中断。

（2）内中断指针

内中断指针的格式如图5.19（b）所示，I6□□～I8□□，共3点。内中断为内部定时时间到信号中断，由指定编号为6～8的专用定时器控制。设定时间在10～99ms间选取，每隔设定时间就会中断一次。

（a）外中断指针格式　（b）内中断指针格式　（c）高速计数器中断指针格式

图 5.19 中断指针格式

5.2.3 中断与返回指令 7

（3）高速外部计数中断指针

高速计数器中断指针的格式如图5.19（c）所示,共6点:I010～I060。这6个中断指针分别表示由高速计数器（C235～C255）的当前值实现的中断。

例5.3 中断指令的梯形图如图5.17和图5.18所示，试解答：

（1）指出I001中断的含义，并分析此中断的过程；

（2）指出I699中断的含义，并分析此中断的过程；

（3）指出I020中断的含义，并分析此中断的过程。

解：（1）I001表示X000为中断请求信号，且上升沿有效。因此在允许中断区间，如果输入X000从OFF→ON变化时（上升沿），则程序从主程序转移到标号为I001处，开始执行中断服务程序1，直至遇IRET指令时返回主程序。

（2）I699表示内部定时时间到中断，每隔99ms就执行I699开始

5.2.3 中断与返回指令 8

的中断服务程序一次，直至遇IRET指令时返回主程序。

（3）I020表示高速计数器C236计数到中断，从X1输入计数脉冲，每当C236当前值为10时，产生中断，执行标号为I020开始的中断服务程序一次，D1就被加1，直至遇IRET指令时返回主程序。

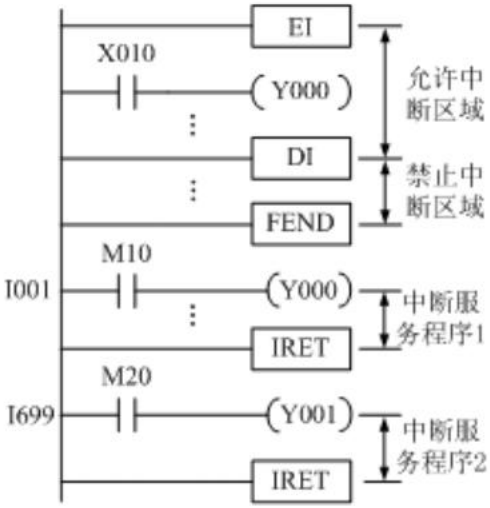


图 5.17 内外中断示意梯形图

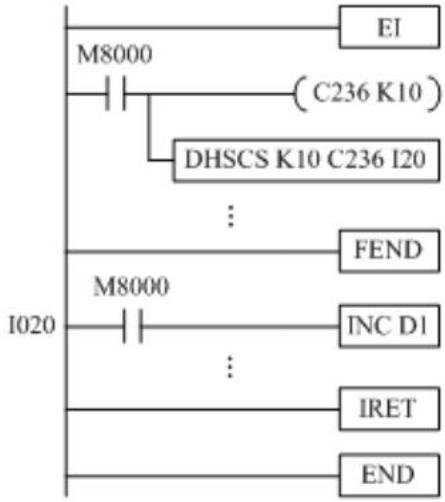


图 5.18 高速计数器中断示意梯形图

5.2.3 中断与返回指令 9

3. 中断请求信号的宽度与中断屏蔽寄存器

（1）中断请求信号宽度

中断请求信号宽度，即中断请求信号的持续时间必须大于200μs，宽度不足的请求信号可能得不到正确响应。

（2）中断屏蔽寄存器

M8050～M8058这10个特殊功能辅助继电器是同中断有关的，其功能如表5.5和表5.6所示。可用程序设置其为ON或OFF，当其为ON时，即使已经用EI指令开中断了，也会屏蔽相关的中断，不妨将M8050～M8059称之为中断屏蔽寄存器。DI则是中断屏蔽总开关。

5.2.3 中断与返回指令 10

表 5.5 内外中断屏蔽寄存器

特殊寄存器名	状态	功能	特殊寄存器名	状态	功能
M8050	ON	禁止 I00X 中断	M8055	ON	禁止 I50X 中断
M8051	ON	禁止 I10X 中断	M8056	ON	禁止 I6XX 中断
M8052	ON	禁止 I20X 中断	M8057	ON	禁止 I7XX 中断
M8053	ON	禁止 I30X 中断	M8058	ON	禁止 I8XX 中断
M8054	ON	禁止 I40X 中断			

表 5.6 高速计数器中断屏蔽寄存器

特殊寄存器名	状态	功能	特殊寄存器名	状态	功能
M8059	ON	禁止 I010 中断	M8059	ON	禁止 I040 中断
	ON	禁止 I020 中断		ON	禁止 I050 中断
	ON	禁止 I030 中断		ON	禁止 I060 中断

5.2.4 主程序结束指令 1

主程序结束指令： FNC 06 FEND

有关主程序结束指令的概要如表5.7。

FEND指令表示主程序结束，此后CPU将进行输入/输出处理、警戒时钟刷新，完成后回到第0步。子程序和中断服务程序都必须写在FEND后，没有它们时也可以没有FEND指令。但是程序的最后必须用END指令结尾。所以，子程序及中断服务程序必须写在FEND指令与END指令之间。

表 5.7 主程序结束指令概要

指令名称	功能号 助记符	操作数	程序步
主程序结束	FNC 06 FEND	[D·]：无	FEND 1步

5.2.4 主程序结束指令 2

若FEND指令在CALL或CALL（P）指令执行之后，SRET指令执行之前出现，或将FEND指令置于FOR-NEXT循环之中，则被认为出错。一个完整的PLC程序可以没有子程序，也可以没有中断服务程序，但必定要有主程序。

5.2.5 警戒时钟指令

警戒时钟刷新指令： FNC 07 WDT（P）

警戒时钟刷新指令概要如表5.8所示。

表 5.8 警戒时钟刷新指令概要

指令名称	功能号 助记符	操作数	程序步
警戒时钟刷新	FNC 07 WDT (P)	[D·]：无	WDT，WDT (P) 1步

5.2.5 警戒时钟指令 1

WDT指令用于1.3.2介绍过的警戒定时器刷新，CPU从第0步扫描到END或FEND指令时，将使警戒定时器复位。如果扫描时间，因干扰超过了D8000 中设定的警戒定时器定时时间，警戒定时器不再被复位，用户程序将会停止执行，PLC面板上的CPU-E出错指示灯将会点亮。为此，可将WDT指令插到合适的程序步中来刷新警戒定时器，以使顺序程序得以继续执行到END。这样就可以将一个运行时间大于警戒定时器定时值的程序用WDT指令分成几部分，使每部分的执行时间都小于警戒定时器定时值。

存储在D8000中的警戒定时器定时时间由PLC的监控程序写入，同时也允许用户改写其内容。若希望扫描周期

5.2.5 警戒时钟指令 2

时间改写为160ms，可以用功能指令MOV来改写D8000的内容，如图5.20所示。此外，WDT指令还可用于：

（1）当程序用CJ指令向后跳转时，即对应的P标号步序小于CJ指令的步序，为防止出错，应在P标号之后插入WDT指令，如图5.21。

（2）可将WDT指令置于FOR-NEXT循环之中，以防止死循环或循环时间超时而停止运行。

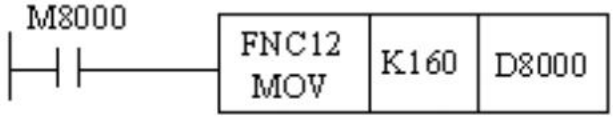


图 5.20 利用 MOV 指令改写警戒时钟

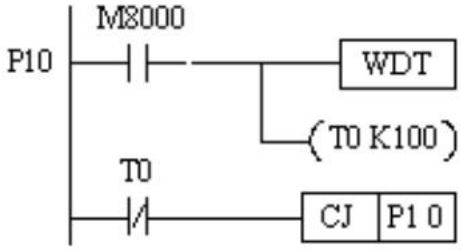


图 5.21 P 标号后插入 WDT 指令

5.2.6 循环指令 1

1. 指令用法说明

（1）循环体起点指令：　　FNC 08　FOR（16）

（2）循环体终点指令：　　FNC 09　NEXT

循环指令概要如表5.9所示。能够充当源操作数的为如表中[S·]所指定的范围内的所有软元件。

循环指令可反复执行某段程序，只要将这一段程序放在

表 5.9 循环指令概要

指令名称	功能号助记符	操作数	程序步
循环开始	FNC08 FOR	[S·] K,H　KnX　KnY　KnM　KnS　T　C　D　V, Z	FOR　3步
循环结束	FNC09 NEXT	无	NEXT 1步

5.2.6 循环指令 2

FOR－NEXT间，待执行完指定的循环次数后，才执行NEXT下一条指令。

在梯形图中判断FOR/NEXT指令配对的原则是：与NEXT指令之前相距最近的FOR指令是一对循环指令，FOR/NEXT对是唯一的。

FOR指令和NEXT指令间包含的程序，称为循环体，循环体内的程序就是要反复循环执行的操作。

如果在循环体内又包含了另外一个完整的循环，则称为循环的嵌套。图5.22中循环C的循环体中包含了循环B的全部，循环B的循环体中包含了循环A的全部，这是三重循环的嵌套，循环指令最多允许5层嵌套。嵌套循环程序的执行总是由内向外，逐层循环的。

5.2.6 循环指令 3

循环次数由FOR后的数值指定，表5.9 中[S·] 区间内的元件都可以。循环次数范围为1～32767，如循环次数＜1时，被当作1处理，FOR--NEXT循环一次。

注意:应避免下述会出错情况。

① NEXT指令出现在FOR指令之前。

② FOR和NEXT指令必须成对使用，缺一不可。

③ NEXT指令出现在FEND或END指令之后。

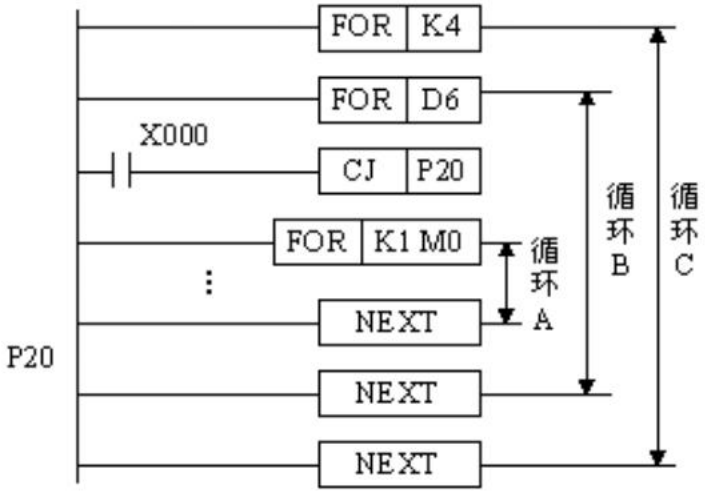


图 5.22 FOR NEXT 指令例

5.2.6 循环指令 4

注意：应避免下述会出错情况。

① NEXT指令出现在FOR指令之前。

② FOR和NEXT指令必须成对使用，缺一不可。

③ NEXT指令出现在FEND或END指令之后。

5.3 数据传送指令

在FX2N系列PLC中设置了8条数据传送指令，2条数据比较指令，其功能号是FNC10～FNC19。

传送指令:

MOV（传送）　　SMOV（BCD码移位传送）

CML （取反传送）　　BMOV（数据块传送）

FMOV（多点传送）　　XCH（数据交换）

BCD（二进制数转换成BCD码并传送）

BIN（BCD码转换为二进制数并传送）

比较指令:

CMP（比较）

ZCP（区间比较）

5.3.1 比较指令 1

比较指令： **FNC10 CMP [S1·] [S2·] [D·]**

其中[S1·]、 [S2·]为两个比较的源操作数，[D·]为比较结果标志软元件，指令中给出的是标志软元件的首地址。

比较指令的概要如表5.10所示。比较指令CMP可对两个数进行代数减法操作，将源操作数[S1·]和[S2·]的数据进

表 5.10 比较指令概要

比较指令		操作数	程序步
P	FNC 10 CMP CMP(P)	[S1·][S2·] K,H / KnX / KnY / KnM / KnS / T / C / D / V, Z X / Y / M / S [D·]	CMP CMP(P) 7步 (D) CMP (D) CMP (P) 13步
D			

5.3.1 比较指令 2

行比较，结果送到目标操作数[D·]中，再将比较结果写入指定的相邻三个标志软元件中。指令中所有源数据均作为二进制数处理。

图5.23所示为比较指令CMP的梯形图，对应的指令为：CMP K100 D10 M0。

在图5.23中，如X010接通，则将执行比较操作，即将100减去D10中的内容，再将比较结果写入相邻三个标志软元件M0～M2中。标志位操作规则是：

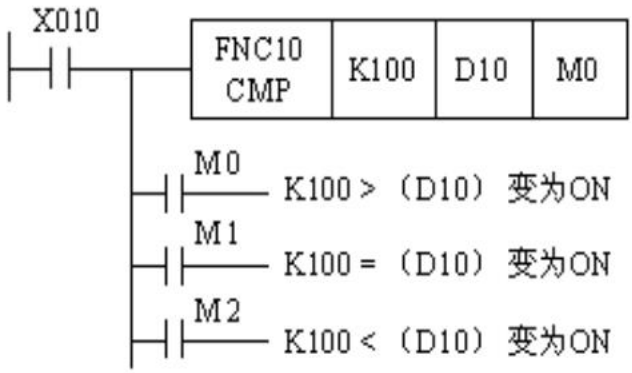


图 5.23 比较指令 CMP 举例

5.3.1 比较指令 3

若K100＞（D10），则M0被置1；
若K100＝（D10），则M1被置1；
若K100＜（D10），则M2被置1。
可见CMP指令执行后，标志位中必有一个被置1，而其余二个均为0。
CMP指令在作32位操作时，使用前缀（D）：
（D）CMP [S1·] [S2·] [D·]。
CMP指令也可有脉冲操作方式，使用后缀（P）：
（D）CMP（P） [S1·] [S2·] [D·]，只有在驱动条件由OFF→ON时进行一次比较。
注意：指令中的三个操作数必须按表5.10所示编写，如果缺操作数，或操作元件超出此表中指定范围等都要引起出错。清除比较结果，可用RST或ZRST复位指令。

5.3.2 区间比较指令 1

区间比较指令：**FNC11 ZCP [S1·] [S2·] [S3·] [D·]**

[S1·]和[S2·]为区间起点和终点，[S3·]为另一比较软元件，[D·]为标志软元件（首地址）。ZCP指令可将某个指定的源数据[S3·]与一个区间的数据进行代数比较，[S1·]和[S2·]分别为区间的下限和上限，比较结果送到目标操作数[D·]中，[D·]由3个连续的标志位软元件组成。

表 5.11 区间比较指令概要

区间比较指令		操作数	程序步
P	FNC 11 ZCP ZCP（P）	[S1·] [S2·] [S3·] K,H KnX KnY Kn M KnS T C D V, Z X Y M S [D·]	ZCP ZCP (P) 9步 (D)ZCP (D) ZCP(P) 17步
D			

5.3.2 区间比较指令 2

图5.24 所示为区间比较指令示例梯形图，对应指令为：ZCP K100 K200 C0 M0。

如果X010接通，则将执行区间比较操作，即将C0的内容与区间的上下限去比较，比较结果写入相邻三个标志位软元件M0～M2中。标志位操作规则是：

若K100＞C0，则M0被置1；

若K100＜C0＜K200，则M1被置1；

若K200＜C0，则M2被置1。

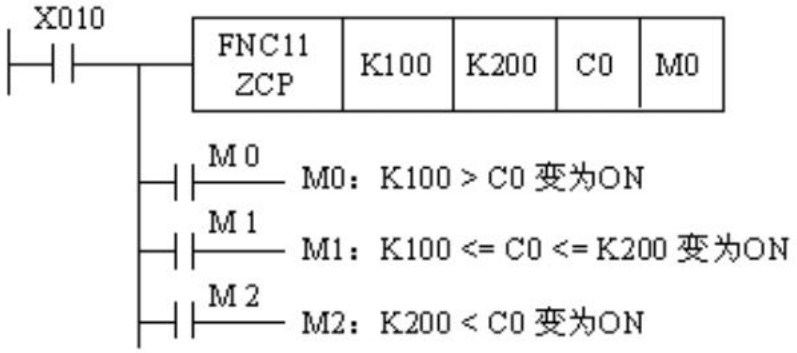


图 5.24 区间比较指令 ZCP 举例

5.3.2 区间比较指令 3

ZCP指令的32位方式：

（D）ZCP [S1·] [S2·] [S3·] [D·]。

ZCP指令的脉冲方式：

（D）ZCP（P）[S1·] [S2·] [S3·] [D·]

有关ZCP指令操作数等注意事项同CMP指令。

5.3.3 传送指令

数据传送指令： **FNC12 MOV [S·] [D·]**

[S·]为源数据，[D·]为目软元件。

功能：将源数据传送到目软元件中去。

数据传送指令概要如表5.12。能充当源操作数的为如表中[S•]所指定的范围内的所有软元件；能够充当目操作数的软元件要除去常数K、H和输入继电器位组合。

5.3.3 传送指令 2

表 5.12 数据传送指令概要

传送指令		操作数	程序步
P	FNC 12 MOV	[S·] K,H KnX KnY KnM KnS T C D V, Z	MOV MOV (P) 5步
D	MOV(P)	[D·]	(D) MOV (D) MOV (P) 9步

图5.25为MOV的示例梯形图，对应的指令为：
MOV D10 D20。
如X010接通，将D10的内容传送到D20中去，传送结果

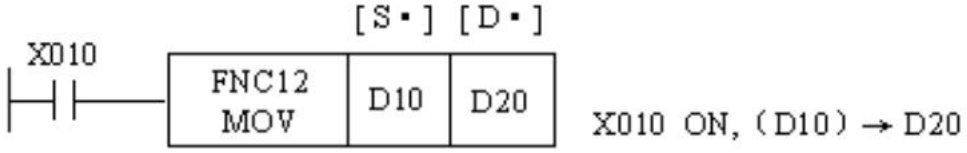


图 5.25 数据传送指令 MOV 举例

5.3.3 传送指令 3

D10内容保持不变，D20中内容被D10内容转化为二进制后取代。MOV指令的32位脉冲方式：

（D）MOV（P） [S·] [D·]。

5.3.4 移位传送指令

移位传送指令：**FNC13 SMOV [S·] m1 m2 [D·] n**

[S·]为源数据，m1为被传送的起始位，m2为传送位数

表 5.13 移位传送指令概要

移位传送指令		操作数	程序步
P	FNC 13 SMOV	[S·] K,H KnX KnY KnM KnS T C D V, Z n m1,m2 [D·]	SMOV
16	SMOV(P)		SMOV (P) 11步

5.3.4 移位传送指令 2

，[D·]为目软元件，n为传送的目起始位。

功能：将[S·]第m1位开始的m2个数移位到[D·]的第n位开始的m2个位置去，ml、m2和n取值均为：1～4。分开的BCD码重新分配组合，一般用于多位BCD拨盘开关的数据输入。

图5.26为SMOV的示例梯形图，对应指令为：

SMOV D10 K4 K2 D20 K3

移位传送示意图如图5.27。设D10=BCD码4321，D20=BCD码9008。如X010接通，执行移位传送指令。

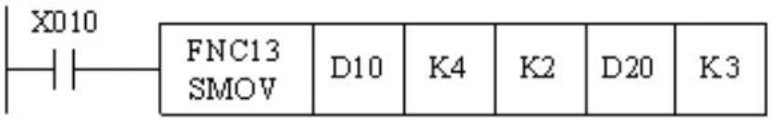


图 5.26 移位传送指令 SMOV 举例

5.3.4 移位传送指令 3

将D10中的二进制数转换成BCD码4321；然后将第4位（m1=K4）开始的共2位（m2=K2）BCD码4和3，分别移到D20的第3位（n=K3）和第2位的BCD码位置上去，所以移位传送后D20=9438。

移位传送指令只能对16位数据进行操作,所以BCD码值超过9999时将会出错。

SMOV指令脉冲方式：

SMOV（P）[S·] m1 m2 [D·] n。

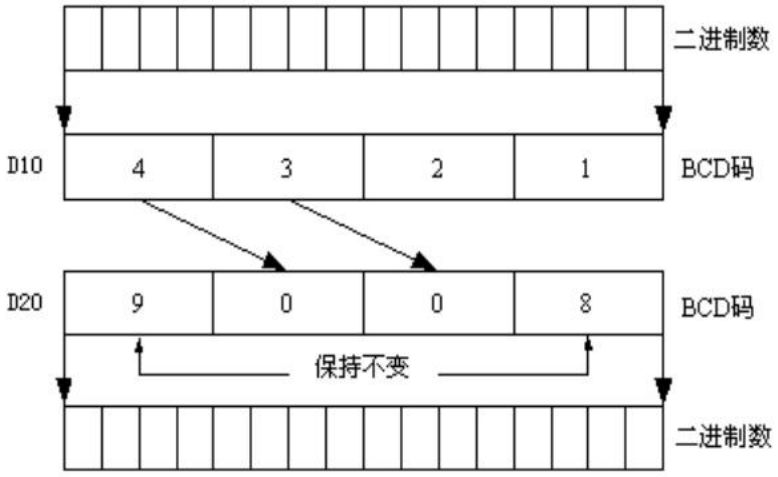


图 5.27 移位传送示意图

5.3.5 取反传送指令 1

取反传送指令： **FNC14 CML [S·] [D·]**

[S·]为源数据，[D·]为目软元件。

功能:将[S·]按二进制的位取反后送到目[D·]中。

取反传送指令概要如表5.14。

图5.28为取反传送指令CML示例梯形图，对应指令为：CML D10 K1Y001。

表 5.14 取反传送指令概要

取反传送指令		操作数	程序步
P	FNC 14	[S·]	CML
	CML	K,H KnX KnY KnM KnS T C D V, Z	CML (P) 5步
D	CML (P)	[D·]	(D) CML
			(D) CML (P) 9步

5.3.5 取反传送指令 2

在图5.28中，如X010接通，则将执行取反传送指令。首先将D10中的各个位取反。然后根据K1Y001指定，将D10的低4位送到Y004、Y003、Y002、Y001四位目元件中去，因此Y005以上的输出继电器不会有任何变化。如果被取反的软元件是K或H型的都将被变换成二进制数后，再取反传送。

CML指令32位脉冲格式：

（D）CML（P）[S·]　[D·]。

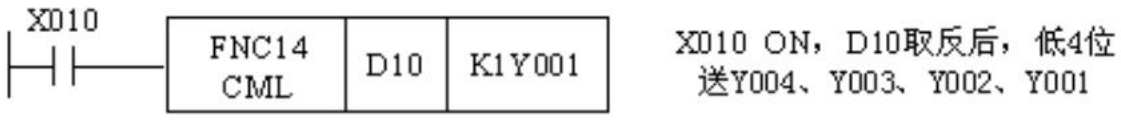


图 5.28　取反传送指令 CML 举例

5.3.6 块传送指令 1

块传送指令：**FNC15 BMOV [S·] [D·] n**

[S·]为源软元件，[D·]为目标软元件，n为数据块个数。

功能：将源中的n个数据组成的数据块传送到指定的目中去。如果元件号超出允许元件号的范围，数据仅传送到允许范围内。

块传送指令概要如表5.15。

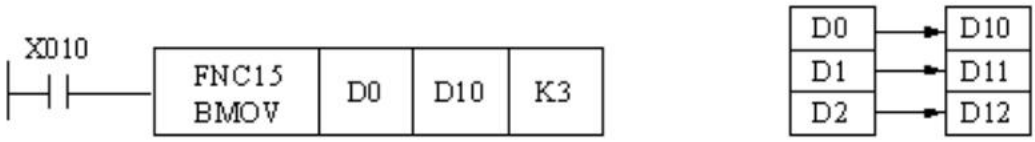
表 5.15 块传送指令概要

块传送指令		操作数	程序步
P	FNC 15 BMOV	[S·] K,H KnX KnY KnM KnS T C D V, Z	BMOV
16	BMOV (P)	n [D·]	BMOV (P) 7步

5.3.6 块传送指令 2

图5.29（a）为块传送指令示例梯形图，对应指令为：BMOV D0 D10 K3。

在图5.29（a）中，如X010接通，执行块传送指令。K3指定数据块个数为3，将D0～D2内容传送到D10～D12，如图5.29（b）。当源、目类型相同时，传送顺序自动决定。如源、目类型不同，只要位数相同就可正确传送。如源、目软元件号超出允许范围，则只对符合规定的数据传送。 BMOV指令没有32位操作方式，但有脉冲方式：BMOV（P）[S·] [D·] n。



（a）块传送指令举例

（b）块传送示意

图 5.29 块传送指令 BMOV

5.3.7 多点传送指令 1

多点传送指令： FNC16 FMOV [S·] [D·] n

[S·]为源软元件，[D·]为目软元件，n为目软元件个数。

功能：将一个源中的数据传送到指定的n个目中去。指令中给出的是目的首地址。常用于对某一段数据寄存器清零或置相同的初始值。

多点传送指令概要如表5.16。

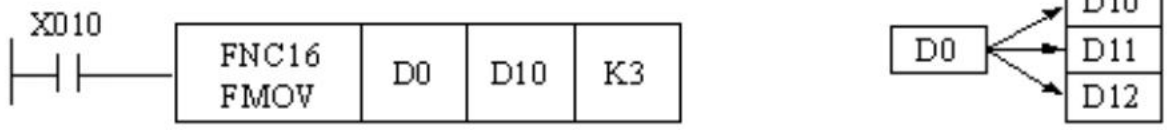
表 5.16 块传送指令概要

多点传送指令		操作数	程序步
P	FNC 15 FMOV	[S·] K,H KnX KnY KnM KnS T C D V, Z	FMOV
16	FMOV (P)	n [D·]	FMOV (P) 7步

5.3.7 多点传送指令 2

图5.30（a）为多点传送指令示例梯形图，对应指令：FMOV D0 D10 K3。

在图5.30（a）中，如X010接通，按K3指定目元件个数为3，则将D0中的内容传送到D10～D12中去，如图5.30（b）。传送后D0中的内容不变，而D10～D12内容被D0内容取代。如果目软元件号超出允许范围，则只对符合规定的数据进行传送。 FMOV指令没有32位操作方式，但有脉冲方式：FMOV（P）[S·] [D·] n



（a）多点传送指令举例　　（b）多点传送示意

图 5.30 多点传送指令 FMOV

5.3.8 数据交换指令 1

数据交换指令： FNC17 XCH [D1·] [D2·]

[D1·]，[D2·]为两个目软元件。

功能：将两个指定的目软元件的内容交换。

数据交换指令概要如表5.17。

图5.31为数据交换指令示例梯形图，对应指令为：

XCH D10 D20

表 5.17 数据交换指令概要

数据交换指令		操作数	程序步
P	FNC 17	[D1·] K,H KnX KnY KnM KnS T C D V, Z [D2·]	XCH
	XCH		XCH (P) 5步
D	XCH (P)		DXCH
			DXCH (P) 9步

5.3.8 数据交换指令 2

在图5.31中，如X010接通，则将执行数据交换指令。将D10的内容传送到D20中去，而D20中的内容则传送到D10中去，两个软元件的内容互换。

注意：按图5.31中的梯形图，数据在每个扫描周期都要交换1次，而经过两次交换后D10和D20的内容将复原。解决的办法是使用XCH指令的脉冲方式，只有在驱动条件由OFF→ON时进行一次交换操作。

XCH指令的32位脉冲方式为：

（D）XCH（P）[D1·]　[D2·]。

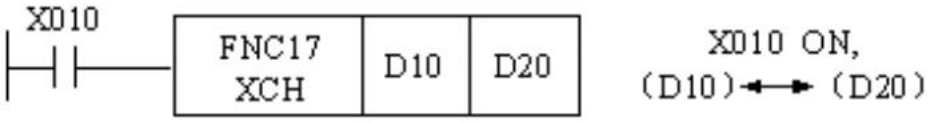


图 5.31　数据交换指令 XCH 举例

5.3.9 BCD变换指令 1

BCD码变换指令：**FNC18 BCD [S·] [D·]**

[S·]为被转换的软元件，[D·]为目标软元件。

功能：将指定软元件的内容转换成BCD码并送到指定的目软元件中去。再译成7段码，就能输出驱动LED。

BCD码变换指令概要如表5.18。

图5.32为BCD码变换指令的示例梯形图，对应指令为：BCD D10 K2Y000。

表 5.18 BCD 码变换指令概要

BCD 码变换指令		操作数	程序步
P	FNC 18	[S·] K,H KnX KnY Kn M KnS T C D V, Z	BCD
	BCD		BCD (P) 5 步
D	BCD (P)	[D·]	(D)BCD
			(D)BCD (P) 9 步

5.3.9 BCD变换指令 2

在图5.32中，如X010接通，则将执行BCD码变换指令，即将D10中的二进制数转换成BCD码，然后将低八位内容送到Y007～Y000中去。指令执行过程的示意如图5.33所示。注意，如果超出了BCD码变换指令能够转换

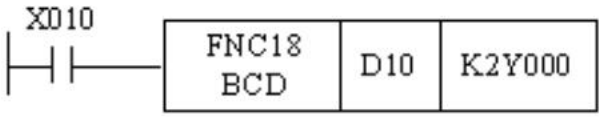


图 5.32 BCD 码变换指令 BCD 举例

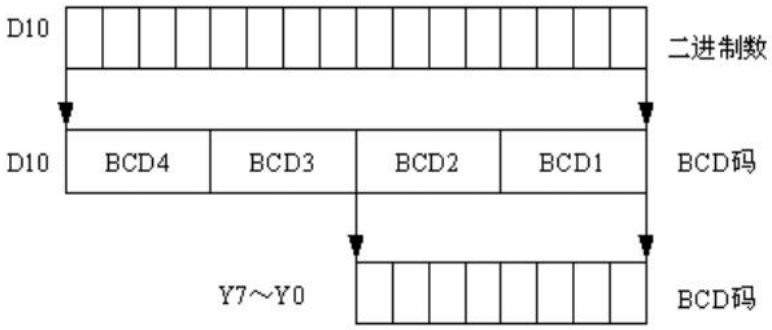


图 5.33 BCD 码变换指令执行示意

5.3.9 BCD变换指令 3

的最大数据范围就会出错，16位操作时为0～9999；32位操作时为0～99999999。BCD码变换指令的32位脉冲操作格式为：（D）BCD（P）[S·]　[D·]。

5.3.10 BIN变换指令

BIN变换指令：　**FNC19　BIN　[S·]　[D·]**

[S·]为被转换的软元件，[D·]为目软元件。BIN变换指令概要如表5.19。

表 5.19　BIN 变换指令概要

BIN 变换指令		操作数	程序步
P	FNC 19	[S·]	BIN
	BIN	K,H　KnX　KnY　Kn M　KnS　T　C　D　V, Z	BIN (P)　5 步
D	BIN (P)	[D·]	(D) BIN
			(D) BIN (P)　9 步

5.3.10 BIN变换指令 2

功能：将指定软元件中的BCD码转换成二进制数并送到指定的目软元件中去。此指令作用正好与BCD变换指令相反，用于将软元件中的BCD码转换成二进制数

图5.34为BIN变换指令的示例梯形图，对应指令为：

BIN K2X000 D10。

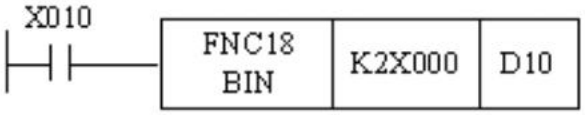


图 5.34 BIN 变换指令举例

这条指令可将BCD拨盘的设定值通过X007～X000输入到PLC中去。如X010接通，则将执行BIN变换指令，把从X007～X000上输入的两位BCD码，变换成二进制数，传送到D10的低八位中。

指令执行过程如图5.35，设输入的BCD码＝63，如直接输入，是二进制01100011（十进制99），就会出错。

5.3.10 BIN变换指令 3

如用BIN变换指令输入，将会先把BCD码63转化成二进制00111111，就不会出错了。

注意：如[S·]中内容不是BCD码就会出错，也不能是常数K，因为在操作前，程序自动将其变换成二进制数。

BIN变换指令32位脉冲方式：（D）BIN（P）[S·]　[D·]

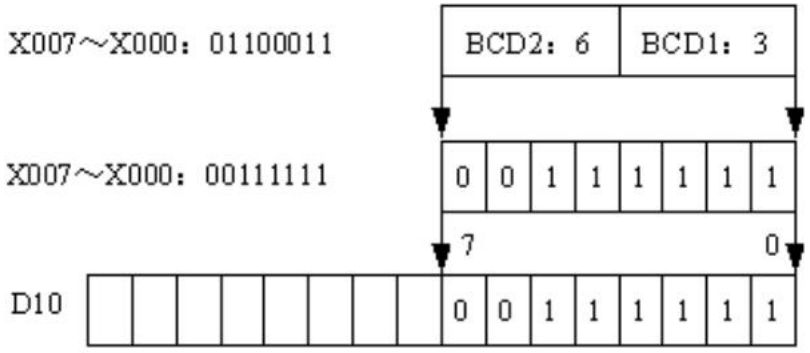


图 5.35　BIN 变换指令执行示意

5.4 算术和逻辑运算指令 1

10条算逻运算指令的功能号是FNC20～FNC29。

5.4.1 BIN加法指令

二进制加法指令：FNC20 ADD [S1·] [S2·] [D·]

[S1·]、[S2·]为加数源元件，[D·]为和的目元件。

功能：将指定的两个源中有符数，进行二进制加法，然后将和送入指定的目中。

二进制加法指令概要如表5.20。

表 5.20 二进制加法指令概要

加法指令		操作数	程序步
P	FNC 20	[S1·] [S2·]	ADD
	ADD	K,H KnX KnY Kn M KnS T C D V, Z	ADD(P) 7步
D	ADD(P)	[D·]	(D) ADD
			(D) ADD (P) 13 步

5.4.1 BIN加法指令 2

加法指令影响三个标志位：
①若相加和＝0，零标志位M8020=1；
②若发生进位，则进位标志位M8022=1；
③若发生借位，则借位标志M8021=1。
若浮点数标志位M8023置1，则可进行浮点加法运算。
图5.36为加法指令ADD的示例梯形图，对应的指令为：
ADD K10 D10 D20。
在图5.36中，如X010接通，执行加法运算，即将10与D10中的内容相加，结果送入D20中，并根据运算的结果使相应的标志位置1。

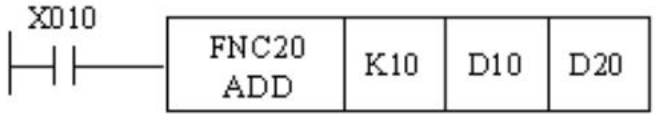


图 5.36 加法指令 ADD 举例

ADD指令32位方式：（D）ADD D10 D20 D30。
指令中给出的源、目是其首地址，如对加数1来说，低

5.4.1 BIN加法指令 3

16位在D10中，高16位在相邻下一数据寄存器D11中，两者组成一个32位的加数1。同理，D21和D20组成了另一个加数2；D31和D30组成和数。为避免重复使用某软元件，建议用偶数元件号。源、目可用相同的元件号，如：ADD Dl0 D20 D10。

ADD指令脉冲方式：ADD（P） Dl0 D20 D10。

5.4.2 BIN减法指令

二进制减法指令：**FNC21 SUB [S1·] [S2·] [D·]**

[S1·]、[S2·]为被减数和减数源元件，[D·]为差目元件。

功能：将指定两个源软元件中有符数，进行二进制代数减法，相减结果差送入指定的目元件中。

二进制减法指令概要如表5.21。

5.4.2 BIN减法指令 2

SUB指令进行的是二进制有符数减法代数运算，减法指令影响标志位：

①相减结果为0，零标志位M8020=1；

②相减发生借位，借位标志M8021=1；

③若相减发生进位，进位标志M8022=1。若将浮点数标志位M8023置1，则可以进行浮点数减法运算。

表 5.21 二进制减法指令概要

减法指令		操作数	程序步
P	FNC 21	[S1·] [S2·]	SUB
	SUB	K,H KnX KnY KnM KnS T C D V, Z	SUB(P) 7步
D	SUB(P)	[D·]	(D) SUB
			(D) SUB (P) 13步

5.4.2 BIN减法指令 3

图5.37为减法指令SUB的示例梯形图，对应的指令为：SUB K10 D10 D20。

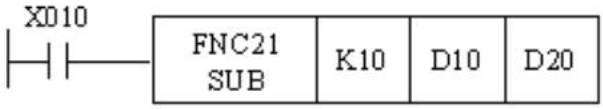


图 5.37 减法指令 SUB 举例

在图5.37中，如X010接通，执行减法运算，将10与D10中的内容相减，结果送入D20中，并根据运算的结果使相应的标志位置1。

SUB指令的32位脉冲操作格式为：

（D）SUB（P）[S·] [D·]，

这时，指令中给出的是源、目软元件的首地址。

5.4.3 BIN乘法指令

二进制乘法指令：**FNC22 MUL [S1·] [S2·] [D·]**

[S1·]、[S2·]为被乘数和乘数的源元件，[D·]为乘积的目元件的首地址。

5.4.3 BIN乘法指令 2

功能：将指定的两个源软元件中的数，进行二进制有符数乘法，然后将相乘的积送入指定的目标元件中。

二进制乘法指令概要如表5.22。

图5.38为乘法指令示例梯形图，对应的指令为：

MUL　D10　D20　D30。

表 5.22　二进制乘法指令概要

乘法指令		操作数	程序步
P	FNC 22	[S1・] [S2・]	MUL
	MUL	K,H | KnX | KnY | KnM | KnS | T | C | D | *Z | V	MUL（P）　7步
D	MUL（P）	*:16位时可用　[D・]	(D) MUL
			(D) MUL (P) 13 步

5.4.3 BIN乘法指令 3

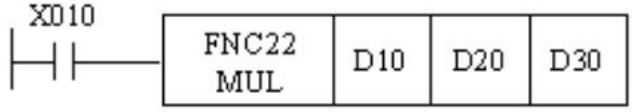


图 5.38 乘法指令 MUL 举例

在图5.38中，如X010接通，执行有符数乘法运算，将D10与D20中的两内容相乘，积送入D31和D30中两个目单元中去。MUL指令进行的是有符数乘法，被乘数和乘数最高位是符号位，分为16位和32位操作两种情况：

（1）16位乘法运算

源都是16位的，但积却是32位的。积将按照“高对高，低对低”的原则存放到目中，即积的低16位存放到指令中给出的低地址目元件中，高16位存放到高一号地址的目元件中。如果积用位元件（Y、M、S）组合来存放，则目元件要用K8来给定，小于K8将得不到32位的积，如用K4则只能得到低16位。

5.4.3 BIN乘法指令 4

16位乘法允许使用脉冲执行方式：

MUL（P） [S1·] [S2·] [D·]。

（2）32位乘法运算

32位的脉冲方式的MUL指令：

（D）MUL（P） D10 D20 D30。

指令中的源都是32位的，被乘数的32位在D11和D10中，乘数的32位在D21和D20中；但是积却是64位的，并将存放到D33、D32、D31和D30中。如果积用位元件（Y、M、S）组合来存放，即使用K8来指定，也只能得到积的低32位，积的高32位将丢失。解决的办法是先用字元件存放积，然后再传送到位元件组合。

若将浮点数标志位M8023置1，可进行浮点数乘法运算

5.4.4 BIN除法指令 1

二进制除法指令：**FNC23 DIV [S1·] [S2·] [D·]**

[S1·]、[S2·]为存放被除数和除数源元件，[D·]为商和余数的目元件首地址。功能：将指定两个源元件中的数，进行二进制有符数除法，将相除的商和余数送入从首地址开始的相应的目标元件中。

二进制除法指令概要如表5.23 。

表 5.23 二进制除法指令概要

除法指令		操作数	程序步
P	FNC 23	[S1·] [S2·]	DIV
	DIV	K,H KnX KnY KnM KnS T C D *Z V	DIV(P) 7步
D	DIV(P)	*:16位时可用 [D·]	(D) DIV
			(D) DIV (P) 13步

5.4.4 BIN除法指令 2

图5.39为除法指令DIV示例梯形图，对应的指令为DIV D10 D20 D30。

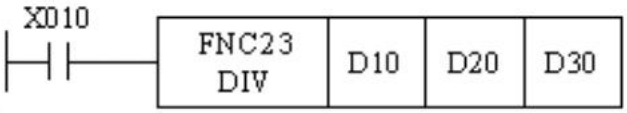


图 5.39 除法指令 DIV 举例

在图5.39中，如X010接通，执行除法运算，将D10与D20中的两内容相除，商送入D30中，而余数放入D31中。

DIV指令分为16位和32位操作两种情况。

（1）16位除法运算

16位除法运算的源、目都是16位的，虽然商是不会超过16位。如商用位元件组合来存放，能得到相应指定的位数的商，如用K4M0指定能得到16位，但这时余数将丢失。解决的办法是先用字元件存放商和余数，然后再传送到位元件组合去。

5.4.4 BIN除法指令 3

16位除法脉冲方式：DIV（P） [S1·] [S2·] [D·]。

（2）32位除法运算

32位脉冲方式DIV指令:（D）DIV（P）D10 D20 D30。指令中的源、目都为32位，给出的都只是它们的首地址。被除数的32位在D11和D10中，除数的32位在D21和D20中；商的32位在D31和D30中，余数的32位在D33和D32中。都是按照“高对高，低对低”的原则存放的。如果商用位元件组合来存放，能得到相应指定位数的商，如用K8M0指定能得到32位，但余数将丢失。解决的办法是先用字元件存放商和余数，然后再传送到位元件组合去。除法运算中除数不能为0，否则要出错。若将浮点数标志位M8023置1，则可进行浮点数除法运算。

5.4.5 BIN加1指令 1

二进制加1指令： **FNC24 INC [D·]**

[D·]是要加1的目元件。

功能：将指定的目软元件的内容增加1。

二进制加1指令概要如表5.24 。

图5.40为加1指令INC的示例梯形图，对应的指令为：INC（P）D10。

表 5.24 二进制加 1 指令概要

加 1 指令		操作数	程序步
P	FNC 24	K,H KnX KnY KnM KnS T C D V, Z	INC
	INC	[D▪]	INC(P) 3步
D	INC(P)		(D) INC
			(D) INC (P) 5步

5.4.5 BIN加1指令 2

INC指令常使用的是脉冲操作方式。在图5.40中，如X010由OFF→ON时，则将执行一次加1运算，即将老的D10内容加1后作为新的D10内容。

INC指令不影响标志位。比如，用INC指令进行16位操作时，当正数32767再加1时，将会变为-32768；这种情况下进位或借位标志都不受影响。

INC指令最常用于循环次数、变址操作等情况。

图 5.40 加 1 指令 INC 举例

5.4.5 BIN加1指令 2

DEC指令经常使用的是脉冲操作方式。在图5.41中，如X010由OFF→ON时，则将执行一次减1运算，即将老的D10内容减1后作为新的D10内容。

DEC指令不影响标志位。比如，用DEC指令进行16位操作时，当负数－32768再减1时，将会变为32767；这种情况下进位或借位标志都不受影响。

DEC指令也常用于循环次数、变址操作等情况。

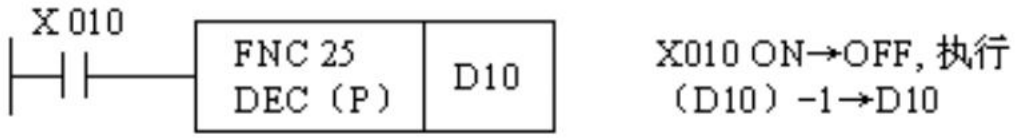


图 5.41 减 1 指令 DEC 举例

5.4.7 逻辑“与”指令 1

逻辑“与”指令：**FNC26 WAND [S1·] [S2·] [D·]** [S1·]、[S2·]为两个相“与”的源元件，[D·]为放相“与”结果的目元件。功能：将指定两源元件中数，进行二进制按位“与”，然后将相“与”结果送入指定的目软元件中。

逻辑“与”指令概要如表5.26 。

图5.42为逻辑“与”指令示例梯形图，对应的指令为：

WAND D10 D20 D30

表5.26 逻辑“与”指令概要

逻辑“与”指令		操作数	程序步
P	FNC 26	[S1·] [S2·]	WAND
	WAND	K,H KnX KnY KnM KnS T C D V, Z	WAND(P) 7步
D	WAND(P)	[D·]	(D) AND
			(D) AND (P) 13步

5.4.7 逻辑“与”指令 2

WAND前面的“W”表示16位字操作，以与“与”基本指的数据宽度仅一位的AND指令相区别。

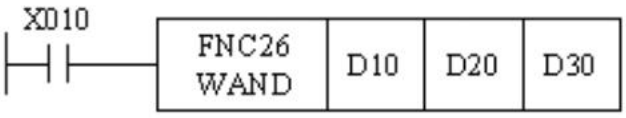


图 5.42 逻辑“与”指令 WAND 举例

如X010接通，则将执行逻辑“与”运算，即将D10“与”D20中的内容，进行二进制按位“与”，相“与D10＝12，D20＝10，则送入D30的相“与”结果为8，相“与”的示意如图5.43所示。“与”运算的规则是：“全1出1，有0出0”。在D10与D20相“与”运算中，只有第3位满足“全1出1”，在第2至第0位相“与”中，至少有一位是0，所以相“与”结果都是“有0出0”。

5.4.8 逻辑“或”指令 1

逻辑“或”指令： **FNC27 WOR [S1·] [S2·] [D·]**

[S1·]、[S2·]为两个相“或”的源元件，[D·]为放相“或”结果的目元件。

功能：将指定的两个源元件中的数，进行二进制按位“或”，然后将相“或”结果送入指定的目标元件中。

逻辑“或”指令概要如表5.27。

表5.27 “逻辑或”指令概要

逻辑“或”指令		操作数	程序步
P	FNC 27	[S1·] [S2·]	WOR
	WOR	K,H KnX KnY KnM KnS T C D V, Z	WOR(P) 7步
D	WOR(P)	[D·]	(D) OR
			(D) OR (P) 13步

5.4.7 逻辑“或”指令 2

图5.44为逻辑“或”指令示例梯形图，对应的指令为：WOR D10 D20 D30。

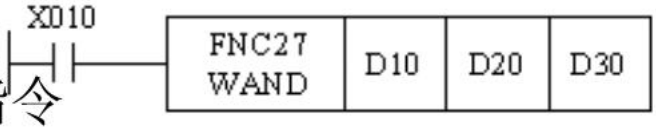


图 5.44 逻辑“或”指令 WOR 举例

WOR前面的“W”表示16位字操作，以与“或”基本指令中的数据宽度仅一位的OR指令相区别。

如X010接通，则将执行逻辑“或”运算，即将D10“或”D20中的内容，进行二进制按位“或”，相“或”结果将送入D30中。假设D10中的数据为12，D20中的数据为10，则送入D30的相“或”结果为14，相“或”的示意如图5.45所示。或运算的规则是：“全0出0，有1出1”。在D10或D20相“或”运算中，只有第0位满足“全0出0”，在

5.4.7 逻辑“或”指令 3

第3至第1位相“或”中，至少有一位是1，所以相“或”结果都是“有1出1”。

逻辑“或”指令的32位脉冲操作格式为：

（D）OR（P）[S1·] [S2·] [D·]。指令中给出的[S1·]、[S2·]和[D·]分别为源和目软元件的首地址。

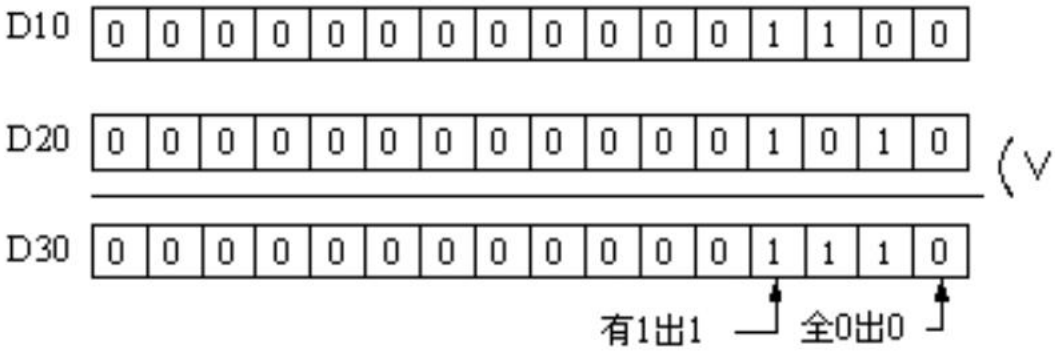


图 5.45 逻辑“或”指令示意

5.4.9 逻辑“异或”指令 1

逻辑“异或”指令: **FNC28 WXOR [S1·] [S2·] [D·]**

[S1·]、[S2·]为两个相”异或”的源元件，[D·]为放相“异或”结果的目标元件。

功能：将指定的两个源软元件中的数，进行二进制按位“异或”，然后将相“异或”结果送入指定的目元件中。

表 5.28 逻辑“异或”指令概要

逻辑“异或”指令		操作数	程序步
P	FNC 28	[S1·] [S2·]	WXOR
	WXOR	K,H KnX KnY Kn M KnS T C D V, Z	WXOR(P) 7步
D	WXOR(P)	[D·]	(D) XOR
			(D) XOR (P) 13步

5.4.9 逻辑“异或”指令 2

图5.46为逻辑“异或”指令示例梯形图，对应的指令为：WXOR D10 D20 D30。 WOR D10 D20 D30。WXOR前面的“W”表示16位操作。在图5.46中，如果X010接通，则将执行逻辑“异或”运算，即将D10“异或”D20中的内容，进行二进制按位“异或”，相“异或”结果将送入D30中。设D10＝12，D20＝10，则送入D30相“异或”结果为6，相“异或”的示意如图5.47所示。“异或”运算可以理解为不考虑进位的按位加，其规则是：“相同出0，相异出1”。在D10与D20相“异或”运算中，第3和第0位满足“相同出0”，第2和第1位满足“相异出1”。

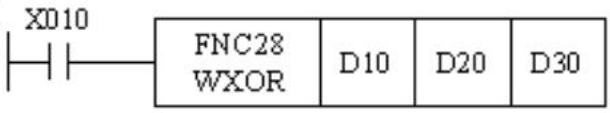


图 5.46 逻辑“异或”指令 WXOR 举例

5.4.9 逻辑“异或”指令 3

逻辑“异或”指令的32位脉冲操作格式为：

（D）XOR（P）[S1·] [S2·] [D·]。

同样，指令中给出的[S1·]、[S2·]和[D·]分别为源和目元件的首地址。

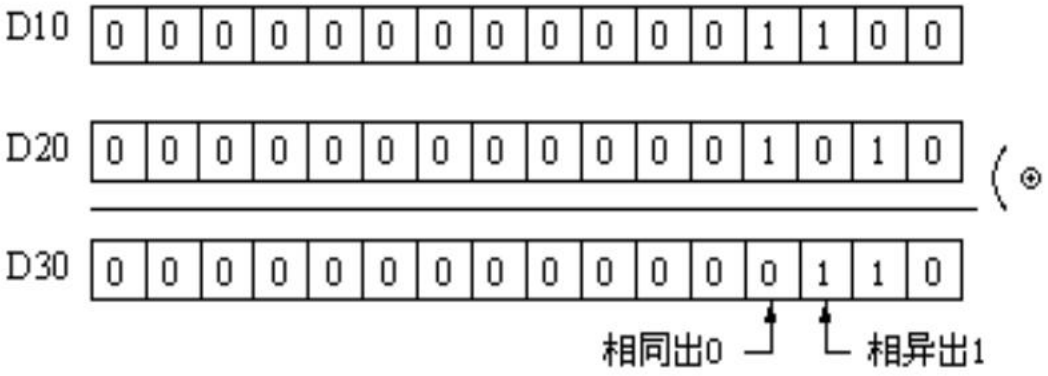


图 5.47 逻辑“异或”指令示意

5.4.10 求补指令 1

求补指令： FNC29 NEG [D·]。

[D·]为存放求补结果的目元件。

功能：将将指定的目元件[D·]中的数，进行二进制求补运算，然后将求补结果再送入目元件中。

求补指令概要如表5.29。

表 5.29 求补指令概要

求补指令		操作数	程序步
P	FNC 29 NEG	K,H KnX KnY Kn M KnS T C D V, Z [D·]	NEG NEG(P) 3步
D	NEG(P)		(D) NEG (D) NEG (P) 5步

5.4.10 求补指令 2

图5.48为求补指令NEG的示例梯形图，对应的指令为：

NEG D10

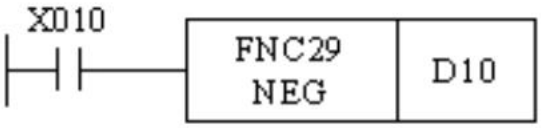


图 5.48 求补指令 NEG 举例

如果X010接通，执行求补运算，即将D10中的二进制数，进行“连同符号位求反加1”，再将求补结果送入D10中求补示意如图5.49。设D10＝H000C，求补就要对它进行“连同符号位求反加1”，最高位的符号位也得参加“求反加1”，求补结果为HFFF4再存入D10中。

图 5.49 求补指令示意

5.4.10 求补指令 3

求补同求补码是不同的，求补码的规则是：
“符号位不变，数值位求反加1”，
对H000C求补码的结果将是H7FF4，两者的结果不一样。求补指令是绝对值不变的变号运算，求补前的H000C的真值是十进制＋12，而求补后的HFFF4的真值是十进制－12。
求补指令的32位脉冲操作格式为：
（D）NEG（P） [D·]，
[D·]为目软元件的首地址。求补指令一般使用其脉冲执行方式，否则每个扫描周期都将执行一次求补操作。

注意：5.5～5.8中的功能指令，不再做课件，需要时请查阅教材相关章节。

本章小结

本章介绍FX2N的功能指令，功能指令使控制变得更加灵活、方便，使其功能变得更强大。

FX2N的功能指令可以归纳为：

程序流向控制	数据比较与传送
算术与逻辑运算	循环移位与移位
数据处理	高速处理
方便类	外部I/O
FX系列外围设备	外部F2设备
浮点数	定位

时钟运算和接点比较等几大类。在第7章与第8章的案例中也会用到定位、浮点数运算和接点比较功能指令。要

本章小结

注意功能指令的使用条件和源、目操作数的选用范围和选用方法，要注意有些功能指令在整个程序中只能使用一次。

目前主流的Windows平台下的三菱GPPW和FXGP编程软件，是学习和使用三菱PLC是离不开的得力助手，对学习功能指令也是一个很好的“老师”。当遇到对指令的疑惑时，如程序步、源、目操作数的范围、有否脉冲执行方式以及有否32位操作数方式等等，可以用它们来进行上机验证。在学习功能指令组成的梯形图时，要充分利用GX Simulator对其进行模拟仿真，以加深对梯形图的理解。